

Azure Pipelines task reference

Article • 09/26/2023

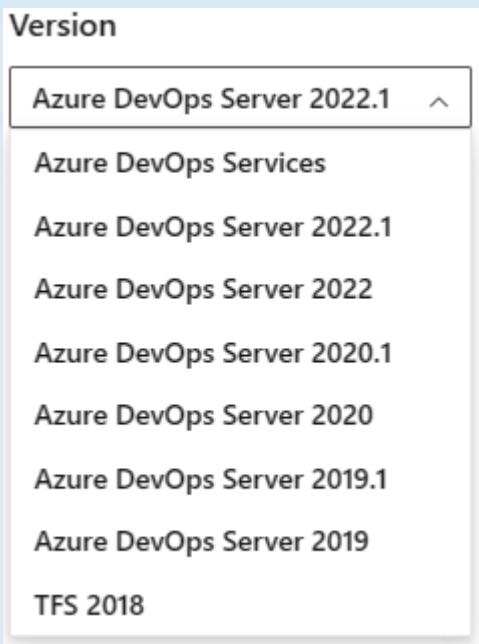
A task performs an action in a pipeline. For example, a task can build an app, interact with Azure resources, install a tool, or run a test. Tasks are the building blocks for defining automation in a pipeline.

The articles in this section describe the built-in tasks for Azure Pipelines and specify the semantics for attributes that hold special meaning for each task.

Please refer to the [YAML Reference for *steps.task*](#) for details on the general attributes supported by tasks.

For how-tos and tutorials about authoring pipelines using tasks, including creating custom tasks, custom extensions, and finding tasks on the Visual Studio Marketplace, see [Tasks concepts](#) and [Azure Pipelines documentation](#).

Important



To view the task reference for tasks available for your platform, make sure that you select the correct Azure DevOps version from the version selector which is located above the table of contents. Feature support differs depending on whether you are working from Azure DevOps Services or an on-premises version of Azure DevOps Server.

To learn which on-premises version you are using, see [Look up your Azure DevOps platform and version](#).

Build tasks

Task	Description
.NET Core DotNetCoreCLI@2 DotNetCoreCLI@1 DotNetCoreCLI@0	Build, test, package, or publish a dotnet application, or run a custom dotnet command.
Android Build AndroidBuild@1	AndroidBuild@1 is deprecated. Use Gradle.
Android Signing AndroidSigning@3 AndroidSigning@2 AndroidSigning@1	Sign and align Android APK files.
Ant Ant@1	Build with Apache Ant.
Azure IoT Edge AzureIoTEdge@2	Build and deploy an Azure IoT Edge image.
CMake CMake@1	Build with the CMake cross-platform build system.
Container Build ContainerBuild@0	Container Build Task.
Docker Docker@2 Docker@1 Docker@0	Build or push Docker images, login or logout, start or stop containers, or run a Docker command.
Docker Compose DockerCompose@0	Build, push or run multi-container Docker applications. Task can be used with Docker or Azure Container registry.
Download GitHub Nuget Packages DownloadGitHubNugetPackage@1	Restore your nuget packages using dotnet CLI.
Go Go@0	Get, build, or test a Go application, or run a custom Go command.
Gradle Gradle@3 Gradle@2 Gradle@1	Build using a Gradle wrapper script.
Grunt Grunt@0	Run the Grunt JavaScript task runner.

Task	Description
gulp gulp@1 gulp@0	Run the gulp Node.js streaming task-based build system.
Index sources and publish symbols PublishSymbols@2 PublishSymbols@1	Index your source code and publish symbols to a file share or Azure Artifacts symbol server.
Jenkins queue job JenkinsQueueJob@2	Queue a job on a Jenkins server.
Jenkins Queue Job JenkinsQueueJob@1	Queue a job on a Jenkins server.
Maven Maven@4 Maven@3 Maven@2 Maven@1	Build, test, and deploy with Apache Maven.
MSBuild MSBuild@1	Build with MSBuild.
Prepare Analysis Configuration SonarQubePrepare@5 SonarQubePrepare@4	Prepare SonarQube analysis configuration.
Publish Quality Gate Result SonarQubePublish@5 SonarQubePublish@4	Publish SonarQube's Quality Gate result on the Azure DevOps build result, to be used after the actual analysis.
Run Code Analysis SonarQubeAnalyze@5 SonarQubeAnalyze@4	Run scanner and upload the results to the SonarQube server.
Visual Studio build VSBuild@1	Build with MSBuild and set the Visual Studio version property.
Xamarin.Android XamarinAndroid@1	Build an Android app with Xamarin.
Xamarin.iOS XamariniOS@2 XamariniOS@1	Build an iOS app with Xamarin on macOS.
Xcode Xcode@5 Xcode@4	Build, test, or archive an Xcode workspace on macOS. Optionally package an app.

Task	Description
Xcode Build Xcode@3 Xcode@2	Build an Xcode workspace on macOS.
Xcode Package iOS XcodePackageiOS@0	Generate an .ipa file from Xcode build output using xcrun (Xcode 7 or below).

Deploy tasks

Task	Description
App Center distribute AppCenterDistribute@3 AppCenterDistribute@2 AppCenterDistribute@1 AppCenterDistribute@0	Distribute app builds to testers and users via Visual Studio App Center.
ARM template deployment AzureResourceManagerTemplateDeployment@3	Deploy an Azure Resource Manager (ARM) template to all the deployment scopes.
Azure App Service Classic (Deprecated) AzureWebPowerShellDeployment@1	Create or update Azure App Service using Azure PowerShell.
Azure App Service deploy AzureRmWebAppDeployment@4 AzureRmWebAppDeployment@3 AzureRmWebAppDeployment@2	Deploy to Azure App Service a web, mobile, or API app using Docker, Java, .NET, .NET Core, Node.js, PHP, Python, or Ruby.
Azure App Service manage AzureAppServiceManage@0	Start, stop, restart, slot swap, slot delete, install site extensions or enable continuous monitoring for an Azure App Service.
Azure App Service Settings AzureAppServiceSettings@1	Update/Add App settings an Azure Web App for Linux or Windows.
Azure CLI AzureCLI@2 AzureCLI@1	Run Azure CLI commands against an Azure subscription in a PowerShell Core/Shell script when running on Linux agent or PowerShell/PowerShell Core/Batch script when running on Windows agent.
Azure CLI Preview AzureCLI@0	Run a Shell or Batch script with Azure CLI commands against an azure subscription.
Azure Cloud Service deployment AzureCloudPowerShellDeployment@2 AzureCloudPowerShellDeployment@1	Deploy an Azure Cloud Service.

Task	Description
Azure Container Apps Deploy AzureContainerApps@1 AzureContainerApps@0	An Azure DevOps Task to build and deploy Azure Container Apps.
Azure Database for MySQL deployment AzureMysqlDeployment@1	Run your scripts and make changes to your Azure Database for MySQL.
Azure file copy AzureFileCopy@5 AzureFileCopy@4 AzureFileCopy@3 AzureFileCopy@2 AzureFileCopy@1	Copy files to Azure Blob Storage or virtual machines.
Azure Function on Kubernetes AzureFunctionOnKubernetes@1 AzureFunctionOnKubernetes@0	Deploy Azure function to Kubernetes cluster.
Azure Functions Deploy AzureFunctionApp@2 AzureFunctionApp@1	Update a function app with .NET, Python, JavaScript, PowerShell, Java based web applications.
Azure Functions for container AzureFunctionAppContainer@1	Update a function app with a Docker container.
Azure Key Vault AzureKeyVault@2 AzureKeyVault@1	Download Azure Key Vault secrets.
Azure Monitor alerts (Deprecated) AzureMonitorAlerts@0	Configure alerts on available metrics for an Azure resource (Deprecated).
Azure PowerShell AzurePowerShell@5 AzurePowerShell@4 AzurePowerShell@3 AzurePowerShell@2 AzurePowerShell@1	Run a PowerShell script within an Azure environment.
Azure resource group deployment AzureResourceGroupDeployment@2	Deploy an Azure Resource Manager (ARM) template to a resource group and manage virtual machines.
Azure Resource Group Deployment AzureResourceGroupDeployment@1	Deploy, start, stop, delete Azure Resource Groups.
Azure Spring Apps AzureSpringCloud@0	Deploy applications to Azure Spring Apps and manage deployments.

Task	Description
Azure SQL Database deployment SqlAzureDacpacDeployment@1	Deploy an Azure SQL Database using DACPAC or run scripts using SQLCMD.
Azure VM scale set deployment AzureVmssDeployment@0	Deploy a virtual machine scale set image.
Azure Web App AzureWebApp@1	Deploy an Azure Web App for Linux or Windows.
Azure Web App for Containers AzureWebAppContainer@1	Deploy containers to Azure App Service.
Build machine image PackerBuild@1 PackerBuild@0	Build a machine image using Packer, which may be used for Azure Virtual machine scale set deployment.
Check Azure Policy compliance AzurePolicyCheckGate@0	Security and compliance assessment for Azure Policy.
Chef Chef@1	Deploy to Chef environments by editing environment attributes.
Chef Knife ChefKnife@1	Run scripts with Knife commands on your Chef workstation.
Copy files over SSH CopyFilesOverSSH@0	Copy files or build artifacts to a remote machine over SSH.
Deploy to Kubernetes KubernetesManifest@1 KubernetesManifest@0	Use Kubernetes manifest files to deploy to clusters or even bake the manifest files to be used for deployments using Helm charts.
IIS web app deploy IISWebAppDeploymentOnMachineGroup@0	Deploy a website or web application using Web Deploy.
IIS Web App deployment (Deprecated) IISWebAppDeployment@1	Deploy using MSDeploy, then create/update websites and app pools.
IIS web app manage IISWebAppManagementOnMachineGroup@0	Create or update websites, web apps, virtual directories, or application pools.
Invoke REST API InvokeRESTAPI@1 InvokeRESTAPI@0	Invoke a REST API as a part of your pipeline.
Kubectl Kubernetes@1 Kubernetes@0	Deploy, configure, update a Kubernetes cluster in Azure Container Service by running kubectl commands.

Task	Description
Manual intervention ManualIntervention@8	Pause deployment and wait for manual intervention.
Manual validation ManualValidation@0	[PREVIEW] Pause a pipeline run to wait for manual interaction. Works only with YAML pipelines.
MySQL database deploy MysqlDeploymentOnMachineGroup@1	Run scripts and make changes to a MySQL Database.
Package and deploy Helm charts HelmDeploy@0	Deploy, configure, update a Kubernetes cluster in Azure Container Service by running helm commands.
PowerShell on target machines PowerShellOnTargetMachines@3	Execute PowerShell scripts on remote machines using PSSession and Invoke-Command for remoting.
PowerShell on Target Machines PowerShellOnTargetMachines@2 PowerShellOnTargetMachines@1	Execute PowerShell scripts on remote machine(s).
Service Fabric application deployment ServiceFabricDeploy@1	Deploy an Azure Service Fabric application to a cluster.
Service Fabric Compose deploy ServiceFabricComposeDeploy@0	Deploy a Docker Compose application to an Azure Service Fabric cluster.
SQL Server database deploy SqlDacpacDeploymentOnMachineGroup@0	Deploy a SQL Server database using DACPAC or SQL scripts.
SQL Server database deploy (Deprecated) SqlServerDacpacDeployment@1	Deploy a SQL Server database using DACPAC.
SSH SSH@0	Run shell commands or a script on a remote machine using SSH.
Windows machine file copy WindowsMachineFileCopy@2 WindowsMachineFileCopy@1	Copy files to remote Windows machines.

Package tasks

Task	Description
Cargo authenticate (for task runners)	Authentication task for the cargo client used for installing Cargo crates distribution.

Task	Description
CargoAuthenticate@0	
CocoaPods CocoaPods@0	Install CocoaPods dependencies for Swift and Objective-C Cocoa projects.
Conda environment CondaEnvironment@1 CondaEnvironment@0	This task is deprecated. Use <code>conda</code> directly in script to work with Anaconda environments.
Download Github Npm Package DownloadGithubNpmPackage@1	Install npm packages from GitHub.
Maven Authenticate MavenAuthenticate@0	Provides credentials for Azure Artifacts feeds and external maven repositories.
npm Npm@1 Npm@0	Install and publish npm packages, or run an npm command. Supports npmjs.com and authenticated registries like Azure Artifacts.
npm authenticate (for task runners) npmAuthenticate@0	Don't use this task if you're also using the npm task. Provides npm credentials to an .npmrc file in your repository for the scope of the build. This enables npm task runners like gulp and Grunt to authenticate with private registries.
NuGet NuGetCommand@2	Restore, pack, or push NuGet packages, or run a NuGet command. Supports NuGet.org and authenticated feeds like Azure Artifacts and MyGet. Uses NuGet.exe and works with .NET Framework apps. For .NET Core and .NET Standard apps, use the .NET Core task.
NuGet authenticate NuGetAuthenticate@1 NuGetAuthenticate@0	Configure NuGet tools to authenticate with Azure Artifacts and other NuGet repositories. Requires NuGet >= 4.8.5385, dotnet >= 6, or MSBuild >= 15.8.166.59604.
NuGet command NuGet@0	Deprecated: use the "NuGet" task instead. It works with the new Tool Installer framework so you can easily use new versions of NuGet without waiting for a task update, provides better support for authenticated feeds outside this organization/collection, and uses NuGet 4 by default.
NuGet Installer NuGetInstaller@0	Installs or restores missing NuGet packages. Use NuGetAuthenticate@0 task for latest capabilities.
NuGet packager NuGetPackager@0	Deprecated: use the "NuGet" task instead. It works with the new Tool Installer framework so you can easily use new versions of NuGet without waiting for a task update, provides better support for authenticated feeds outside this organization/collection, and uses NuGet 4 by default.

Task	Description
NuGet publisher NuGetPublisher@0	Deprecated: use the "NuGet" task instead. It works with the new Tool Installer framework so you can easily use new versions of NuGet without waiting for a task update, provides better support for authenticated feeds outside this organization/collection, and uses NuGet 4 by default.
NuGet Restore NuGetRestore@1	Restores NuGet packages in preparation for a Visual Studio Build step.
PyPI publisher PyPIServerPublisher@0	Create and upload an sdist or wheel to a PyPI-compatible index using Twine.
Python pip authenticate PipAuthenticate@1 PipAuthenticate@0	Authentication task for the pip client used for installing Python distributions.
Python twine upload authenticate TwineAuthenticate@1 TwineAuthenticate@0	Authenticate for uploading Python distributions using twine. Add '-r FeedName/EndpointName --config-file \$(PYPIRC_PATH)' to your twine upload command. For feeds present in this organization, use the feed name as the repository (-r). Otherwise, use the endpoint name defined in the service connection.
Universal packages UniversalPackages@0	Download or publish Universal Packages.
Xamarin Component Restore XamarinComponentRestore@0	This task is deprecated. Use 'NuGet' instead.

Test tasks

Task	Description
App Center test AppCenterTest@1	Test app packages with Visual Studio App Center.
Azure Load Testing AzureLoadTest@1	Automate performance regression testing with Azure Load Testing.
Container Structure Test ContainerStructureTest@0	Uses container-structure-test (https://github.com/GoogleContainerTools/container-structure-test) to validate the structure of an image based on four categories of tests - command tests, file existence tests, file content tests and metadata tests.
Mobile Center Test VSMobileCenterTest@0	Test mobile app packages with Visual Studio Mobile Center.

Task	Description
Publish code coverage results PublishCodeCoverageResults@2 PublishCodeCoverageResults@1	Publish any of the code coverage results from a build.
Publish test results PublishTestResults@1	Publish test results to Azure Pipelines.
Publish Test Results PublishTestResults@2	Publish test results to Azure Pipelines.
Run functional tests RunVisualStudioTestsusingTestAgent@1	Deprecated: This task and its companion task (Visual Studio Test Agent Deployment) are deprecated. Use the 'Visual Studio Test' task instead. The VSTest task can run unit as well as functional tests. Run tests on one or more agents using the multi-agent job setting. Use the 'Visual Studio Test Platform' task to run tests without needing Visual Studio on the agent. VSTest task also brings new capabilities such as automatically rerunning failed tests.
Visual Studio Test VSTest@2 VSTest@1	Run unit and functional tests (Selenium, Appium, Coded UI test, etc.) using the Visual Studio Test (VsTest) runner. Test frameworks that have a Visual Studio test adapter such as MsTest, xUnit, NUnit, Chutzpah (for JavaScript tests using QUnit, Mocha and Jasmine), etc. can be run. Tests can be distributed on multiple agents using this task (version 2).
Visual Studio test agent deployment DeployVisualStudioTestAgent@2	DeployVisualStudioTestAgent@2 is deprecated. Use the Visual Studio Test task to run unit and functional tests.
Visual Studio Test Agent Deployment DeployVisualStudioTestAgent@1	Deploy and configure Test Agent to run tests on a set of machines.
Xamarin Test Cloud XamarinTestCloud@1	[Deprecated] Test mobile apps with Xamarin Test Cloud using Xamarin.UITest. Instead, use the 'App Center test' task.

Tool tasks

Task	Description
.NET Core SDK/runtime installer DotNetCoreInstaller@1 DotNetCoreInstaller@0	Acquire a specific version of the .NET Core SDK from the internet or local cache and add it to the PATH.

Task	Description
Docker CLI installer DockerInstaller@0	Install Docker CLI on agent machine.
Duffle tool installer DuffleInstaller@0	Install a specified version of Duffle for installing and managing CNAB bundles.
Go tool installer GoTool@0	Find in cache or download a specific version of Go and add it to the PATH.
Helm tool installer HelmInstaller@1 HelmInstaller@0	Install Helm on an agent machine.
Install Azure Func Core Tools FuncToolsInstaller@0	Install Azure Func Core Tools.
Java tool installer JavaToolInstaller@0	Acquire a specific version of Java from a user-supplied Azure blob or the tool cache and sets JAVA_HOME.
Kubectl tool installer KubectlInstaller@0	Install Kubectl on agent machine.
Kubelogin tool installer KubeloginInstaller@0	Helps to install kubelogin.
Node.js tool installer NodeTool@0	Finds or downloads and caches the specified version spec of Node.js and adds it to the PATH.
NuGet tool installer NuGetToolInstaller@1 NuGetToolInstaller@0	Acquires a specific version of NuGet from the internet or the tools cache and adds it to the PATH. Use this task to change the version of NuGet used in the NuGet tasks.
Use .NET Core UseDotNet@2	Acquires a specific version of the .NET Core SDK from the internet or the local cache and adds it to the PATH. Use this task to change the version of .NET Core used in subsequent tasks. Additionally provides proxy support.
Use Node.js ecosystem UseNode@1	Set up a Node.js environment and add it to the PATH, additionally providing proxy support.
Use Python version UsePythonVersion@0	Use the specified version of Python from the tool cache, optionally adding it to the PATH.
Use Ruby version UseRubyVersion@0	Use the specified version of Ruby from the tool cache, optionally adding it to the PATH.
Visual Studio test platform installer VisualStudioTestPlatformInstaller@1	Acquire the test platform from nuget.org or the tool cache. Satisfies the 'vstest' demand and can be used for running

Task	Description
	tests and collecting diagnostic data using the Visual Studio Test task.

Utility tasks

Task	Description
Archive files ArchiveFiles@2	Compress files into .7z, .tar.gz, or .zip.
Archive Files ArchiveFiles@1	Archive files using compression formats such as .7z, .rar, .tar.gz, and .zip.
Azure Network Load Balancer AzureNLBManagement@1	Connect or disconnect an Azure virtual machine's network interface to a Load Balancer's back end address pool.
Bash Bash@3	Run a Bash script on macOS, Linux, or Windows.
Batch script BatchScript@1	Run a Windows command or batch script and optionally allow it to change the environment.
Cache Cache@2	Cache files between runs.
Cache (Beta) CacheBeta@1 CacheBeta@0	Cache files between runs.
Command Line CmdLine@2 CmdLine@1	Run a command line script using Bash on Linux and macOS and cmd.exe on Windows.
Copy and Publish Build Artifacts CopyPublishBuildArtifacts@1	CopyPublishBuildArtifacts@1 is deprecated. Use the Copy Files task and the Publish Build Artifacts task instead.
Copy files CopyFiles@2	Copy files from a source folder to a target folder using patterns matching file paths (not folder paths).
Copy Files CopyFiles@1	Copy files from source folder to target folder using minimatch patterns (The minimatch patterns will only match file paths, not folder paths).
cURL Upload Files cURLUploader@2 cURLUploader@1	Use cURL's supported protocols to upload files.

Task	Description
Decrypt file (OpenSSL) DecryptFile@1	Decrypt a file using OpenSSL.
Delay Delay@1	Delay further execution of a workflow by a fixed time.
Delete files DeleteFiles@1	Delete folders, or files matching a pattern.
Deploy Azure Static Web App AzureStaticWebApp@0	Build and deploy an Azure Static Web App.
Download artifacts from file share DownloadFileshareArtifacts@1	Download artifacts from a file share, like \share\drop.
Download build artifacts DownloadBuildArtifacts@1 DownloadBuildArtifacts@0	Download files that were saved as artifacts of a completed build.
Download GitHub Release DownloadGitHubRelease@0	Downloads a GitHub Release from a repository.
Download package DownloadPackage@1 DownloadPackage@0	Download a package from a package management feed in Azure Artifacts.
Download Pipeline Artifacts DownloadPipelineArtifact@2 DownloadPipelineArtifact@1 DownloadPipelineArtifact@0	Download build and pipeline artifacts.
Download secure file DownloadSecureFile@1	Download a secure file to the agent machine.
Extract files ExtractFiles@1	Extract a variety of archive and compression files such as .7z, .rar, .tar.gz, and .zip.
File transform FileTransform@2 FileTransform@1	Replace tokens with variable values in XML or JSON configuration files.
FTP upload FtpUpload@2 FtpUpload@1	Upload files using FTP.
GitHub Comment GitHubComment@0	Write a comment to your GitHub entity i.e. issue or a pull request (PR).
GitHub Release GitHubRelease@1	Create, edit, or delete a GitHub release.

Task	Description
GitHubRelease@0	
Install Apple certificate InstallAppleCertificate@2	Install an Apple certificate required to build on a macOS agent machine.
Install Apple Certificate InstallAppleCertificate@1 InstallAppleCertificate@0	Install an Apple certificate required to build on a macOS agent.
Install Apple provisioning profile InstallAppleProvisioningProfile@1	Install an Apple provisioning profile required to build on a macOS agent machine.
Install Apple Provisioning Profile InstallAppleProvisioningProfile@0	Install an Apple provisioning profile required to build on a macOS agent.
Install SSH key InstallSSHKey@0	Install an SSH key prior to a build or deployment.
Invoke Azure Function AzureFunction@1 AzureFunction@0	Invoke an Azure Function.
Jenkins download artifacts JenkinsDownloadArtifacts@1	Download artifacts produced by a Jenkins job.
Node.js tasks runner installer NodeTaskRunnerInstaller@0	Install specific Node.js version to run node tasks.
PowerShell PowerShell@2 PowerShell@1	Run a PowerShell script on Linux, macOS, or Windows.
Publish build artifacts PublishBuildArtifacts@1	Publish build artifacts to Azure Pipelines or a Windows file share.
Publish Pipeline Artifacts PublishPipelineArtifact@1 PublishPipelineArtifact@0	Publish (upload) a file or directory as a named artifact for the current run.
Publish Pipeline Metadata PublishPipelineMetadata@0	Publish Pipeline Metadata to Evidence store.
Publish To Azure Service Bus PublishToAzureServiceBus@1 PublishToAzureServiceBus@0	Sends a message to Azure Service Bus using a service connection (no agent is required).
Python script PythonScript@0	Run a Python file or inline script.

Task	Description
Query Azure Monitor alerts AzureMonitor@1	Observe the configured Azure Monitor rules for active alerts.
Query Classic Azure Monitor alerts AzureMonitor@0	Observe the configured classic Azure Monitor rules for active alerts.
Query work items queryWorkItems@0	Execute a work item query and check the number of items returned.
Review App ReviewApp@0	Use this task under deploy phase provider to create a resource dynamically.
Service Fabric PowerShell ServiceFabricPowerShell@1	Run a PowerShell script in the context of an Azure Service Fabric cluster connection.
Shell script ShellScript@2	Run a shell script using Bash.
Update Service Fabric App Versions ServiceFabricUpdateAppVersions@1	Automatically updates the versions of a packaged Service Fabric application.
Update Service Fabric manifests ServiceFabricUpdateManifests@2	Automatically update portions of application and service manifests in a packaged Azure Service Fabric application.
Xamarin License XamarinLicense@1	[Deprecated] Upgrade to free version of Xamarin: https://store.xamarin.com .

Open source

These tasks are open source [on GitHub](#). Feedback and contributions are welcome. See [Pipeline task changelog](#) for a list of task changes, including a historical record of task updates.

FAQ

What are task input aliases?

Inputs to a task are identified by a `label`, `name`, and may include one or more optional `aliases`. The following example is an excerpt from the [source code](#) for the **Known Hosts Entry** input of the [InstallSSHKey@0](#) task.

JSON

```
{  
    "name": "hostName",  
    "aliases": [  
        "knownHostsEntry"  
    ],  
    "label": "Known Hosts Entry"  
    ...  
}
```

Before YAML pipelines were introduced in 2019, pipelines were created and edited using a UI based pipeline editor, and only the `label` was used by pipeline authors to reference a task input.

The screenshot shows the configuration interface for the 'Install SSH key' task. At the top left is a back arrow and the task name. Below are several input fields:

- Known Hosts Entry ***: A red box highlights this field, which contains a placeholder text 'sample known hosts entry line'.
- SSH Public Key**: An empty text input field.
- SSH Passphrase**: An empty text input field.
- SSH Key ***: An empty text input field containing the placeholder 'sample SSH key'.
- Advanced**: A dropdown menu icon.

At the bottom left is a link 'About this task' and a large blue 'Add' button.

When YAML pipelines were introduced in 2019, pipeline authors using YAML started using the task input `name` to refer to a task input. In some cases, the task input names weren't descriptive, so `aliases` were added to provide additional descriptive names for task inputs.

For example, the `InstallSSHKey@0` task has a **Known Hosts Entry** input named `hostName` that expects an entry from a `known_hosts` file. The **Known Hosts Entry** label in the classic pipeline designer makes this clear, but it isn't as clear when using the `hostName`

name in a YAML pipeline. Task input aliases were introduced to allow task authors to provide descriptive names for their previously authored tasks, and for the `InstallSSHKey@0` task, a `knownHostsEntry` alias was added  , while keeping the original `hostName` name for compatibility with existing pipelines using that name.

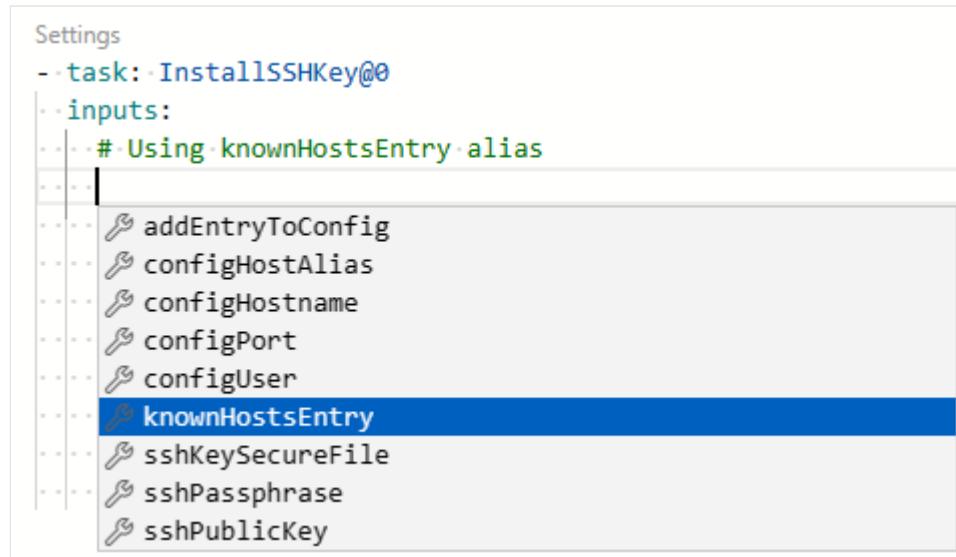
Any items in a task input's `aliases` are interchangeable with the `name` in a YAML pipeline. The following two YAML snippets are functionally identical, with the first example using the `knownHostsEntry` alias and the second example using `hostName`.

```
yml
- task: InstallSSHKey@0
  inputs:
    # Using knownHostsEntry alias
    knownHostsEntry: 'sample known hosts entry line'
    # Remainder of task inputs omitted

- task: InstallSSHKey@0
  inputs:
    # Using hostName name
    hostName: 'sample known hosts entry line'
    # Remainder of task inputs omitted
```

Starting with Azure DevOps Server 2019.1, the [YAML pipeline editor was introduced](#), which provides an intellisense type functionality.

The YAML pipeline editor uses the [Yamlschema - Get](#) REST API to retrieve the schema used for validation in the editor. If a task input has an alias, the schema promotes the alias to the primary YAML name for the task input, and the alias is suggested by the intellisense.



The following example is the **Known Hosts Entry** task input for the `InstallSSHKey@0` task from the YAML schema, with `knownHostsEntry` listed in the name position and `hostName` in the `aliases` collection.

JSON

```
"properties": {  
    "knownHostsEntry": {  
        "type": "string",  
        "description": "Known Hosts Entry",  
        "ignoreCase": "key",  
        "aliases": [  
            "hostName"  
        ]  
    },  
},
```

Because the intellisense in the YAML pipeline editor displays `knownHostsEntry`, and the YAML generated by the [task assistant](#) uses `knownHostsEntry` in the generated YAML, the task reference displays the `alias` from the task source code as the YAML name for a task input. If a task has more than one alias (there are a few that have two aliases), the first alias is used as the name.

Why did the task reference change?

The Azure Pipelines tasks reference documentation moved to its current location to support the following improvements.

- Task articles are generated using the task source code from the [Azure Pipelines tasks open source repository](#).
- Task input names and aliases are generated from the task source so they are always up to date.
- YAML syntax blocks are generated from the task source so they are up to date.
- Supports community contributions with integrated user content such as enhanced task input descriptions, remarks and examples.
- Provides task coverage for all supported Azure DevOps versions.
- Updated every sprint to cover the latest updates.

To contribute, see [Contributing to the tasks content](#).

Where can I learn step-by-step how to build my app?

[Build your app](#)

Can I add my own build tasks?

Yes: [Add a build task](#)

What are installer tasks?

To learn more about tool installer tasks, see [Tool installers](#).

DotNetCoreCLI@2 - .NET Core v2 task

Article • 09/26/2023

Build, test, package, or publish a dotnet application, or run a custom dotnet command.

Syntax

YAML

```
# .NET Core v2
# Build, test, package, or publish a dotnet application, or run a custom
dotnet command.
- task: DotNetCoreCLI@2
  inputs:
    command: 'build' # 'build' | 'push' | 'pack' | 'publish' | 'restore' |
'reun' | 'test' | 'custom'. Required. Command. Default: build.
    #publishWebProjects: true # boolean. Optional. Use when command =
publish. Publish web projects. Default: true.
    #projects: # string. Optional. Use when command = build || command =
restore || command = run || command = test || command = custom ||
publishWebProjects = false. Path to project(s).
    #custom: # string. Required when command = custom. Custom command.
    #arguments: # string. Optional. Use when command = build || command =
publish || command = run || command = test || command = custom. Arguments.
    #restoreArguments: # string. Optional. Use when command = restore.
  Arguments.
    #publishTestResults: true # boolean. Optional. Use when command = test.
  Publish test results and code coverage. Default: true.
    #testRunTitle: # string. Optional. Use when command = test. Test run
title.
    #zipAfterPublish: true # boolean. Optional. Use when command = publish.
  Zip published projects. Default: true.
    #modifyOutputPath: true # boolean. Optional. Use when command = publish.
  Add project's folder name to publish path. Default: true.
    #packagesToPush: '$(Build.ArtifactStagingDirectory)/*.nupkg' # string.
  Alias: searchPatternPush. Required when command = push. Path to NuGet
package(s) to publish. Default: $(Build.ArtifactStagingDirectory)/*.nupkg.
    #nuGetFeedType: 'internal' # 'internal' | 'external'. Required when
command = push. Target feed location. Default: internal.
    #publishVstsFeed: # string. Alias: feedPublish. Required when command =
push && nuGetFeedType = internal. Target feed.
    #publishFeedCredentials: # string. Alias: externalEndpoint. Required
when command = push && nuGetFeedType = external. NuGet server.
    #packagesToPack: '**/*.csproj' # string. Alias: searchPatternPack.
  Required when command = pack. Path to csproj or nuspec file(s) to pack.
  Default: **/*.csproj.
    #configuration: '$(BuildConfiguration)' # string. Alias:
  configurationToPack. Optional. Use when command = pack. Configuration to
Package. Default: $(BuildConfiguration).
    #packDirectory: '$(Build.ArtifactStagingDirectory)' # string. Alias:
```

```
outputDir. Optional. Use when command = pack. Package Folder. Default:  
$(Build.ArtifactStagingDirectory).  
    #nobuild: false # boolean. Optional. Use when command = pack. Do not  
build. Default: false.  
    #includesymbols: false # boolean. Optional. Use when command = pack.  
Include Symbols. Default: false.  
    #includesource: false # boolean. Optional. Use when command = pack.  
Include Source. Default: false.  
    # Feeds and authentication  
    #feedsToUse: 'select' # 'select' | 'config'. Alias: selectOrConfig.  
Required when command = restore. Feeds to use. Default: select.  
    #vstsFeed: # string. Alias: feedRestore. Optional. Use when  
selectOrConfig = select && command = restore. Use packages from this Azure  
Artifacts feed.  
    #includeNuGetOrg: true # boolean. Optional. Use when selectOrConfig =  
select && command = restore. Use packages from NuGet.org. Default: true.  
    #nugetConfigPath: # string. Optional. Use when selectOrConfig = config  
&& command = restore. Path to NuGet.config.  
    #externalFeedCredentials: # string. Alias: externalEndpoints. Optional.  
Use when selectOrConfig = config && command = restore. Credentials for feeds  
outside this organization/collection.  
    # Advanced  
    #noCache: false # boolean. Optional. Use when command = restore. Disable  
local cache. Default: false.  
    #restoreDirectory: # string. Alias: packagesDirectory. Optional. Use  
when command = restore. Destination directory.  
    #verbosityRestore: 'Detailed' # '-' | 'Quiet' | 'Minimal' | 'Normal' |  
'Detailed' | 'Diagnostic'. Optional. Use when command = restore. Verbosity.  
Default: Detailed.  
    # Advanced  
    #publishPackageMetadata: true # boolean. Optional. Use when command =  
push && nuGetFeedType = internal && command = push. Publish pipeline  
metadata. Default: true.  
    # Pack options  
    #versioningScheme: 'off' # 'off' | 'byPrereleaseNumber' | 'byEnvVar' |  
'byBuildNumber'. Required when command = pack. Automatic package versioning.  
Default: off.  
    #versionEnvVar: # string. Required when versioningScheme = byEnvVar &&  
command = pack. Environment variable.  
    #majorVersion: '1' # string. Alias: requestedMajorVersion. Required when  
versioningScheme = byPrereleaseNumber && command = pack. Major. Default: 1.  
    #minorVersion: '0' # string. Alias: requestedMinorVersion. Required when  
versioningScheme = byPrereleaseNumber && command = pack. Minor. Default: 0.  
    #patchVersion: '0' # string. Alias: requestedPatchVersion. Required when  
versioningScheme = byPrereleaseNumber && command = pack. Patch. Default: 0.  
    # Advanced  
    #buildProperties: # string. Optional. Use when command = pack.  
Additional build properties.  
    #verbosityPack: 'Detailed' # '-' | 'Quiet' | 'Minimal' | 'Normal' |  
'Detailed' | 'Diagnostic'. Optional. Use when command = pack. Verbosity.  
Default: Detailed.  
    # Advanced  
    #workingDirectory: # string. Optional. Use when command != restore &&  
command != push && command != pack && command != pack && command != push &&  
command != restore. Working directory.
```

Inputs

`command` - Command

`string`. Required. Allowed values: `build`, `push` (nuget push), `pack`, `publish`, `restore`, `run`, `test`, `custom`. Default value: `build`.

The dotnet command to run. Specify `custom` to add arguments or use a command not listed here.

Important

The **NuGet Authenticate** task is the new recommended way to authenticate with Azure Artifacts and other NuGet repositories. The `restore` and `push` commands of this .NET Core CLI task no longer take new features and only critical bugs are addressed. See remarks for details.

`publishWebProjects` - Publish web projects

`boolean`. Optional. Use when `command = publish`. Default value: `true`.

If this input is set to `true`, the `projects` property value is skipped, and the task tries to find the web projects in the repository and run the publish command on them. Web projects are identified by the presence of either a `web.config` file or a `wwwroot` folder in the directory. In the absence of a `web.config` file or a `wwwroot` folder, projects that use a web SDK, like `Microsoft.NET.Sdk.Web`, are selected.

`projects` - Path to project(s)

`string`. Optional. Use when `command = build || command = restore || command = run || command = test || command = custom || publishWebProjects = false`.

The path to the `.csproj` file(s) to use. You can use wildcards (e.g. `**/*.csproj` for all `.csproj` files in all subfolders). For more information, see the [file matching patterns reference](#).

`custom` - Custom command

`string`. Required when `command = custom`.

The command to pass to `dotnet.exe` for execution. For a full list of available commands, see the [dotnet CLI documentation](#).

`arguments` - Arguments

`string`. Optional. Use when `command = build || command = publish || command = run || command = test || command = custom`.

Specifies the arguments for the selected command. For example, build configuration, output folder, and runtime. The arguments depend on the command selected.

This input currently only accepts arguments for `build`, `publish`, `run`, `test`, and `custom`. If you would like to add arguments for a command not listed, use `custom`.

`restoreArguments` - Arguments

`string`. Optional. Use when `command = restore`.

Writes the additional arguments to be passed to the `restore` command.

`publishTestResults` - Publish test results and code coverage

`boolean`. Optional. Use when `command = test`. Default value: `true`.

Enabling this option will generate a `test results` TRX file in `$(Agent.TempDirectory)`, and the results will be published to the server.

This option appends `--logger trx --results-directory $(Agent.TempDirectory)` to the command line arguments.

Code coverage can be collected by adding the `--collect "Code coverage"` option to the command line arguments.

`testRunTitle` - Test run title

`string`. Optional. Use when `command = test`.

Provides a name for the test run.

`zipAfterPublish` - Zip published projects

`boolean`. Optional. Use when `command = publish`. Default value: `true`.

If this input is set to `true`, folders created by the publish command will be zipped and deleted.

`modifyOutputPath` - Add project's folder name to publish path

`boolean`. Optional. Use when `command = publish`. Default value: `true`.

If this input is set to `true`, folders created by the publish command will have the project file name prefixed to their folder names when the output path is specified explicitly in arguments. This is useful if you want to publish multiple projects to the same folder.

`feedsToUse` - Feeds to use

Input alias: `selectOrConfig`. `string`. Required when `command = restore`. Allowed values: `select` (Feed(s) I select here), `config` (Feeds in my NuGet.config). Default value: `select`.

You can either select a feed from Azure Artifacts and/or [NuGet.org](#) here, or you can commit a `nuget.config` file to your source code repository and set its path using the `nugetConfigPath` input.

`vstsFeed` - Use packages from this Azure Artifacts feed

Input alias: `feedRestore`. `string`. Optional. Use when `selectOrConfig = select && command = restore`.

Includes the selected feed in the generated `NuGet.config`. You must have Package Management installed and licensed to select a feed here. `projectName / feedName` are used for project-scoped feeds. Only `FeedName` is used for organization-scoped feeds. Note: This is not supported for the test command.

`includeNuGetOrg` - Use packages from NuGet.org

`boolean`. Optional. Use when `selectOrConfig = select && command = restore`. Default value: `true`.

Includes `NuGet.org` in the generated `NuGet.config`.

`nugetConfigPath` - Path to NuGet.config

`string`. Optional. Use when `selectOrConfig = config && command = restore`.

The `NuGet.config` in your repository that specifies the feeds from which to restore packages.

`externalFeedCredentials` - Credentials for feeds outside this organization/collection

Input alias: `externalEndpoints`. `string`. Optional. Use when `selectOrConfig = config && command = restore`.

The credentials to use for external registries located in the selected `NuGet.config`. For feeds in this organization/collection, leave this input blank; the build's credentials are used automatically.

`noCache` - Disable local cache

`boolean`. Optional. Use when `command = restore`. Default value: `false`.

Prevents NuGet from using packages from local machine caches.

`restoreDirectory` - Destination directory

Input alias: `packagesDirectory`. `string`. Optional. Use when `command = restore`.

Specifies the folder in which packages are installed. If no folder is specified, packages are restored into the default NuGet package cache.

`verbosityRestore` - Verbosity

`string`. Optional. Use when `command = restore`. Allowed values: `-`, `Quiet`, `Minimal`, `Normal`, `Detailed`, `Diagnostic`. Default value: `Detailed`.

Specifies the amount of detail displayed in the output for the `restore` command.

`packagesToPush` - Path to NuGet package(s) to publish

Input alias: `searchPatternPush`. `string`. Required when `command = push`. Default value: `$(Build.ArtifactStagingDirectory)/*.nupkg`.

The pattern to match or path to `nupkg` files to be uploaded. Multiple patterns can be separated by a semicolon, and you can make a pattern negative by prefixing it with `!`. Example: `**/*.nupkg; !**/*.Tests.nupkg`.

nuGetFeedType - Target feed location

`string`. Required when `command = push`. Allowed values: `internal` (This organization/collection), `external` (External NuGet server (including other organizations/collections)). Default value: `internal`.

Specifies whether the target feed is internal or external.

publishVstsFeed - Target feed

Input alias: `feedPublish`. `string`. Required when `command = push && nuGetFeedType = internal`.

Specifies a feed hosted in this organization. You must have Package Management installed and licensed to select a feed here.

publishPackageMetadata - Publish pipeline metadata

`boolean`. Optional. Use when `command = push && nuGetFeedType = internal && command = push`. Default value: `true`.

Associates this build/release pipeline's metadata (run #, source code information) with the package.

publishFeedCredentials - NuGet server

Input alias: `externalEndpoint`. `string`. Required when `command = push && nuGetFeedType = external`.

The NuGet service connection that contains the external NuGet server's credentials.

packagesToPack - Path to csproj or nuspec file(s) to pack

Input alias: `searchPatternPack`. `string`. Required when `command = pack`. Default value: `**/*.csproj`.

The pattern to search for `.csproj` or `.nuspec` files to pack.

You can separate multiple patterns with a semicolon, and you can make a pattern negative by prefixing it with `!`. Example: `**/*.csproj;!**/*.Tests.csproj`.

configuration - Configuration to Package

Input alias: `configurationToPack`. `string`. Optional. Use when `command = pack`. Default

value: `$(BuildConfiguration)`.

When using a `.csproj` file, this input specifies the configuration to package.

packDirectory - Package Folder

Input alias: `outputDir`. `string`. Optional. Use when `command = pack`. Default value: `$(Build.ArtifactStagingDirectory)`.

The folder where packages will be created. If this folder is empty, packages will be created alongside the `csproj` file.

nobuild - Do not build

`boolean`. Optional. Use when `command = pack`. Default value: `false`.

Specifies that the task will not build the project before packing. This task corresponds to the `--no-build` parameter of the `build` command.

includesymbols - Include Symbols

`boolean`. Optional. Use when `command = pack`. Default value: `false`.

Creates symbol NuGet packages. This task corresponds to the `--include-symbols` command line parameter.

includesource - Include Source

`boolean`. Optional. Use when `command = pack`. Default value: `false`.

Includes source code in the package. This task corresponds to the `--include-source` command line parameter.

versioningScheme - Automatic package versioning

`string`. Required when `command = pack`. Allowed values: `off`, `byPrereleaseNumber` (Use the date and time), `byEnvVar` (Use an environment variable), `byBuildNumber` (Use the build number). Default value: `off`.

This task cannot be used with included referenced projects. If you choose `Use the date and time`, this will generate a [SemVer](#)-compliant version formatted as `x.Y.Z-ci-datetime` where you choose `X`, `Y`, and `Z`.

If you choose `Use an environment variable`, you must select an environment variable and ensure it contains the version number you want to use.

If you choose `Use the build number`, this will use the build number to version your package. **Note:** Under `Options`, set the build number format to `$(BuildDefinitionName)_$(Year:yyyy).$(Month).$(DayOfMonth)$(Rev:r)`.

`versionEnvVar` - Environment variable

`string`. Required when `versioningScheme = byEnvVar && command = pack`.

Specifies the variable name without `$`, `$env`, or `%`.

`majorVersion` - Major

Input alias: `requestedMajorVersion`. `string`. Required when `versioningScheme = byPrereleaseNumber && command = pack`. Default value: `1`.

The `X` in version `X.Y.Z`.

`minorVersion` - Minor

Input alias: `requestedMinorVersion`. `string`. Required when `versioningScheme = byPrereleaseNumber && command = pack`. Default value: `0`.

The `Y` in version `X.Y.Z`.

`patchVersion` - Patch

Input alias: `requestedPatchVersion`. `string`. Required when `versioningScheme = byPrereleaseNumber && command = pack`. Default value: `0`.

The `Z` in version `X.Y.Z`.

`buildProperties` - Additional build properties

`string`. Optional. Use when `command = pack`.

Specifies a list of `token = value` pairs, separated by semicolons, where each occurrence of `$token$` in the `.nuspec` file will be replaced with the given value. Values can be strings in quotation marks.

verbosityPack - Verbosity

`string`. Optional. Use when `command = pack`. Allowed values: `-`, `Quiet`, `Minimal`, `Normal`, `Detailed`, `Diagnostic`. Default value: `Detailed`.

Specifies the amount of detail displayed in the output for the `pack` command.

workingDirectory - Working directory

`string`. Optional. Use when `command != restore && command != push && command != pack && command != pack && command != push && command != restore`.

The current working directory where the script is run. `Empty` is the root of the repo (build) or artifacts (release), which is `$(System.DefaultWorkingDirectory)`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

ⓘ Important

The **NuGet Authenticate** task is the new recommended way to authenticate with Azure Artifacts and other NuGet repositories. The `restore` and `push` commands of this .NET Core CLI task no longer take new features and only critical bugs are addressed.

Why is my build, publish, or test step failing to restore packages?

Most `dotnet` commands, including `build`, `publish`, and `test` include an implicit `restore` step. This will fail against authenticated feeds, even if you ran a successful

`dotnet restore` in an earlier step, because the earlier step will have cleaned up the credentials it used.

To fix this issue, add the `--no-restore` flag to the `Arguments` textbox.

In addition, the `test` command does not recognize the `feedRestore` or `vstsFeed` arguments, and feeds specified in this manner will not be included in the generated `NuGet.config` file when the implicit `restore` step runs. It's recommended that an explicit `dotnet restore` step be used to restore packages. The `restore` command respects the `feedRestore` and `vstsFeed` arguments.

Why am I getting NU1507 warnings with [Package Source Mapping](#) although when building on my machine it has no warnings?

The various commands that do a NuGet restore or access a NuGet feed build a special temporary `NuGet.config` file that add NuGet authentication for azure artifacts NuGet feeds. The way this is done is in conflict with the schema that Package Source Mapping uses to map the packages to the sources and breaks the Package Source Mapping configuration in the `NuGet.config` file in your repository. To work around this conflict you can use the [NuGet Authenticate](#) task to authenticate and afterwards the custom command to invoke the desired dotnet command without the `NuGet.config` modification.

YAML

```
# Authenticate Azure DevOps NuGet feed
- task: NuGetAuthenticate@1
  displayName: 'Authenticate Azure DevOps NuGet feed'

# Restore project
- task: DotNetCoreCLI@2
  inputs:
    command: 'custom'
    custom: 'restore'

# Build project
- task: DotNetCoreCLI@2
  inputs:
    command: 'custom'
    custom: 'build'
    arguments: '--no-restore'
```

Why should I check in a NuGet.config?

Checking a `NuGet.config` into source control ensures that a key piece of information needed to build your project—the location of its packages—is available to every developer that checks out your code.

However, for situations where a team of developers works on a large range of projects, it's also possible to add an Azure Artifacts feed to the global `NuGet.config` on each developer's machine. In these situations, using the `Feeds I select here` option in the NuGet task replicates this configuration.

Troubleshooting

Project using Entity Framework has stopped working on Hosted Agents

.NET Core does not have Entity Framework(EF) built-in. You will have to either install EF before beginning execution or add `global.json` to the project with required .NET Core SDK version. This will ensure that correct SDK is used to build EF project. If the required version is not present on the machine, add the `UseDotNetV2` task to your pipeline to install the required version. For more information, see [Get the Entity Framework Core runtime](#).

Examples

- [Build examples](#)
- [Push examples](#)
- [Push examples](#)
- [Pack examples](#)
- [Publish examples](#)
- [Restore examples](#)
- [Test examples](#)

Build examples

Build a project

YAML

```
# Build project
- task: DotNetCoreCLI@2
```

```
inputs:  
  command: 'build'
```

Build Multiple Projects

YAML

```
# Build multiple projects  
- task: DotNetCoreCLI@2  
  inputs:  
    command: 'build'  
    projects: |  
      src/proj1/proj1.csproj  
      src/proj2/proj2.csproj  
      src/other/other.sln    # Pass a solution instead of a csproj.
```

Push examples

Push NuGet packages to internal feed

YAML

```
# Push non test NuGet packages from a build to internal organization Feed  
- task: DotNetCoreCLI@2  
  inputs:  
    command: 'push'  
    searchPatternPush:  
      '$(Build.ArtifactStagingDirectory)/*.nupkg;!$(Build.ArtifactStagingDirectory)  
      )/*.Tests.nupkg'  
    feedPublish: 'FabrikamFeed'
```

Push NuGet packages to external feed

YAML

```
# Push all NuGet packages from a build to external Feed  
- task: DotNetCoreCLI@2  
  inputs:  
    command: 'push'  
    nugetFeedType: 'external'  
    externalEndPoint: 'MyNuGetServiceConnection'
```

Pack examples

Pack a NuGetPackage to a specific output directory

YAML

```
# Pack a NuGet package to a test directory
- task: DotNetCoreCLI@2
  inputs:
    command: 'pack'
    outputDir: '$(Build.ArtifactStagingDirectory)/TestDir'
```

Pack a Symbol Package

YAML

```
# Pack a symbol package along with NuGet package
- task: DotNetCoreCLI@2
  inputs:
    command: 'pack'
    includesymbols: true
```

Publish examples

Publish projects to specified folder

YAML

```
# Publish projects to specified folder.
- task: DotNetCoreCLI@2
  displayName: 'dotnet publish'
  inputs:
    command: 'publish'
    publishWebProjects: false
    projects: '**/*.csproj'
    arguments: '-o $(Build.ArtifactStagingDirectory)/Output'
    zipAfterPublish: true
    modifyOutputPath: true
```

Restore examples

YAML

```
#Restore packages with the .NET Core CLI task
- task: DotNetCoreCLI@2
  displayName: 'dotnet restore'
  inputs:
```

```
command: 'restore'
feedsToUse: 'select'
feedRestore: ' projectName/feedName '
projects: '**/*.csproj'
includeNuGetOrg: true
```

Test examples

Run tests in your repository

YAML

```
# Run tests and auto publish test results.
- task: DotNetCoreCLI@2
  inputs:
    command: 'test'
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Build

DotNetCoreCLI@1 - .NET Core v1 task

Article • 09/26/2023

Build, test and publish using dotnet core command-line.

Syntax

YAML

```
# .NET Core v1
# Build, test and publish using dotnet core command-line.
- task: DotNetCoreCLI@1
  inputs:
    command: 'build' # 'build' | 'publish' | 'restore' | 'test' | 'run'.
    Required. Command. Default: build.
    #publishWebProjects: true # boolean. Optional. Use when command = publish. Publish Web Projects. Default: true.
    #projects: # string. Optional. Use when command != publish || publishWebProjects = false. Project(s).
    #arguments: # string. Arguments.
    #zipAfterPublish: true # boolean. Optional. Use when command = publish. Zip Published Projects. Default: true.
```

Inputs

`command` - Command

`string`. Required. Allowed values: `build`, `publish`, `restore`, `test`, `run`. Default value: `build`.

The dotnet command to run. Specify `custom` to add arguments or use a command not listed here.

`publishWebProjects` - Publish Web Projects

`boolean`. Optional. Use when `command = publish`. Default value: `true`.

If this input is set to `true`, the `projects` property value is skipped, and the task tries to find the web projects in the repository and run the publish command on them. Web projects are identified by the presence of either a `web.config` file or a `wwwroot` folder in the directory. In the absence of a `web.config` file or a `wwwroot` folder, projects that use a web SDK, like `Microsoft.NET.Sdk.Web`, are selected.

`projects` - Project(s)

`string`. Optional. Use when `command != publish || publishWebProjects = false`.

The path to the `.csproj` file(s) to use. You can use wildcards (e.g. `**/*.csproj` for all `.csproj` files in all subfolders). For more information, see the [file matching patterns reference](#).

`arguments` - Arguments

`string`.

Specifies the arguments for the selected command. For example, build configuration, output folder, and runtime. The arguments depend on the command selected.

This input currently only accepts arguments for `build`, `publish`, `run`, `test`, and `custom`. If you would like to add arguments for a command not listed, use `custom`.

`zipAfterPublish` - Zip Published Projects

`boolean`. Optional. Use when `command = publish`. Default value: `true`.

If this input is set to `true`, a folder created by the publish command will be zipped and deleted.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

For a newer version of this task, see [DotNetCoreCLI@2](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.0.0 or greater
Task category	Build

See also

- [DotNetCoreCLI@2](#)

DotNetCoreCLI@0 - .NET Core v0 task

Article • 09/26/2023

Use this task to build, test, package, or publish a dotnet application, or to run a custom dotnet command. For package commands, this task supports NuGet.org and authenticated feeds like Package Management and MyGet.

If your .NET Core or .NET Standard build depends on NuGet packages, make sure to add two copies of this step: one with the `restore` command and one with the `build` command.

This task is deprecated; use [DotNetCoreCLI@2](#).

Syntax

YAML

```
# .NET Core v0
# Build, test and publish using dotnet core command-line.
- task: DotNetCoreCLI@0
  inputs:
    command: 'build' # 'build' | 'publish' | 'restore' | 'test' | 'run'.
    Required. Command. Default: build.
    #publishWebProjects: true # boolean. Optional. Use when command = publish. Publish Web Projects. Default: true.
    #projects: # string. Optional. Use when command != publish || publishWebProjects = false. Project(s).
    #arguments: # string. Arguments.
    #zipAfterPublish: true # boolean. Optional. Use when command = publish. Zip Published Projects. Default: true.
```

Inputs

`command` - Command

`string`. Required. Allowed values: `build`, `publish`, `restore`, `test`, `run`. Default value: `build`.

The dotnet command to run. Specify `custom` to add arguments or use a command not listed here.

`publishWebProjects` - Publish Web Projects

`boolean`. Optional. Use when `command = publish`. Default value: `true`.

If this input is set to `true`, the `projects` property value is skipped and the task tries to find the web projects in the repository and run the `publish` command on them. Web projects are identified by the presence of either a `web.config` file or a `wwwroot` folder in the directory. In the absence of a `web.config` file or a `wwwroot` folder, projects that use a web SDK, like `Microsoft.NET.Sdk.Web`, are selected.

`projects` - Project(s)

`string`. Optional. Use when `command != publish || publishWebProjects = false`.

The path to the `.csproj` file(s) to use. You can use wildcards (e.g. `**/*.csproj` for all `.csproj` files in all subfolders). For more information, see the [file matching patterns reference](#).

`arguments` - Arguments

`string`.

Specifies the arguments for the selected command. For example, build configuration, output folder, and runtime. The arguments depend on the command selected.

This input currently only accepts arguments for `build`, `publish`, `run`, `test`, and `custom`. If you want to add arguments for a command not listed, use `custom`.

`zipAfterPublish` - Zip Published Projects

`boolean`. Optional. Use when `command = publish`. Default value: `true`.

If this input is set to `true`, the folder created by the `publish` command will be zipped and deleted.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

ⓘ Important

`DotNetCorCLI@0` is deprecated. For a newer supported version, see [DotNetCoreCLI@2](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.95.0 or greater
Task category	Build

See also

- [DotNetCoreCLI@2](#)

DotNetCoreInstaller@1 - .NET Core SDK/runtime installer v1 task

Article • 09/26/2023

Use this task to acquire a specific version of the .NET Core SDK from the internet or local cache and add it to the PATH.

This task is deprecated.

Syntax

YAML

```
# .NET Core SDK/runtime installer v1
# Acquire a specific version of the .NET Core SDK from the internet or local
cache and add it to the PATH.
- task: DotNetCoreInstaller@1
  inputs:
    packageType: 'sdk' # 'runtime' | 'sdk'. Required. Package to install.
    Default: sdk.
    version: '2.2.x' # string. Required. Version. Default: 2.2.x.
    #includePreviewVersions: false # boolean. Include Preview Versions.
    Default: false.
    # Advanced
    #installationPath: '$(Agent.ToolsDirectory)/dotnet' # string. Path To
    Install .Net Core. Default: $(Agent.ToolsDirectory)/dotnet.
    #performMultiLevelLookup: false # boolean. Perform Multi Level Lookup.
    Default: false.
```

Inputs

packageType - Package to install

`string`. Required. Allowed values: `runtime` (Only Runtime), `sdk` (SDK (contains runtime)). Default value: `sdk`.

Specifies whether to install only Runtime or the full SDK.

version - Version

`string`. Required. Default value: `2.2.x`.

Specifies the version of .NET Core SDK or Runtime to install.

Use the following format to specify the version:

- 2.x: Installs the latest in major version.
- 2.2.x: Installs the latest in major and minor version.
- 2.2.104: Installs the exact version.

Find the value of `version` for installing SDK/Runtime in the [releases-index file](#).

`includePreviewVersions` - Include Preview Versions

`boolean`. Default value: `false`.

Specifies if you want preview versions to be included while searching for latest versions. This setting is ignored if you specify an exact version, such as `3.0.100-preview3-010431`.

`installationPath` - Path To Install .Net Core

`string`. Default value: `$(Agent.ToolsDirectory)/dotnet`.

Specifies where .NET Core SDK/Runtime should be installed. Different paths can have the following impact on .NET's behavior:

- **`$(Agent.ToolsDirectory)`**: This determines the version to be cached on the agent since this directory isn't cleaned up across pipelines. All pipelines running on the agent, would have access to the versions installed previously using the agent.
- **`$(Agent.TempDirectory)`**: This can ensure that a pipeline doesn't use any cached version of .NET core since this folder is cleaned up after each pipeline.
- **Any other path**: You can configure any other path, given the agent process has access to the path. This will change the state of the machine and impact all processes running on it.

You can also configure the Multi-Level Lookup setting. This setting can configure .NET host to probe for a suitable version.

`performMultiLevelLookup` - Perform Multi Level Lookup

`boolean`. Default value: `false`.

This input is only applicable to Windows-based agents. This input configures the behavior of .NET host processes for looking up a suitable shared framework. `False` means that only versions present in the folder specified in this task would be looked by the host process. `True` means that the host will attempt to look in pre-defined global locations using Multi-level Lookup.

For Windows, the default global locations are:

- C:\Program Files\dotnet (64-bit processes)
- C:\Program Files (x86)\dotnet (32-bit process)

For more information, see [Multi-level SharedFX Lookup](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

This task is deprecated. Use [@UseDotNet2](#).

What's new in this task version.

- Support for installing multiple versions side by side.
- Support for patterns in version to fetch latest in minor/major version.
- Restrict Multi-level lookup.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: DotNetCore
Command restrictions	Any
Settable variables	Any

Requirement	Description
Agent version	All supported agent versions.
Task category	Tool

DotNetCoreInstaller@0 - .NET Core SDK/runtime installer v0 task

Article • 09/26/2023

Use this task to acquire a specific version of the .NET Core SDK from the internet or local cache and add it to the PATH.

Syntax

```
# .NET Core SDK/runtime installer v0
# Acquire a specific version of the .NET Core SDK from the internet or local
cache and add it to the PATH.
- task: DotNetCoreInstaller@0
  inputs:
    packageType: 'sdk' # 'runtime' | 'sdk'. Required. Package to install.
    Default: sdk.
    version: '1.0.4' # string. Required. Version. Default: 1.0.4.
```

Inputs

`packageType` - Package to install

`string`. Required. Allowed values: `runtime`, `sdk` (SDK (contains runtime)). Default value: `sdk`.

Specifies whether to install only Runtime or the full SDK.

`version` - Version

`string`. Required. Default value: `1.0.4`.

Specifies the exact version of the .NET Core SDK or Runtime to install.

Find the value of `version-sdk` for installing the SDK, or `version-runtime` for installing Runtime from any releases [in GitHub](#).

Note

The task won't work with new versions of .NET Core. Upgrade to `@UseDotNet2` of the task so you can download the latest versions of .NET Core.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: DotNetCore
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Tool

AndroidBuild@1 - Android Build v1 task

Article • 09/26/2023

Use this task to build an Android app using Gradle and (optionally) start the emulator for unit tests.

The `AndroidBuild@1` task is deprecated. Use the [Gradle task](#) instead.

Syntax

YAML

```
# Android Build v1
# AndroidBuild@1 is deprecated. Use Gradle.
- task: AndroidBuild@1
  inputs:
    #gradleWrapper: # string. Location of Gradle Wrapper.
    #gradleProj: # string. Project Directory.
    #gradleArguments: 'build' # string. Gradle Arguments. Default: build.
    # Android Virtual Device (AVD) Options
    avdName: 'AndroidBuildEmulator' # string. Required. Name. Default:
    AndroidBuildEmulator.
    #createAvd: AndroidBuildEmulator # boolean. Create AVD. Default:
    AndroidBuildEmulator.
    #emulatorTarget: 'android-19' # string. Required when createAvd = true.
    AVD Target SDK. Default: android-19.
    #emulatorDevice: 'Nexus 5' # string. Optional. Use when createAvd =
    true. AVD Device. Default: Nexus 5.
    #avdAbi: 'default/armeabi-v7a' # string. Required when createAvd = true.
    AVD ABI. Default: default/armebi-v7a.
    #avdForce: false # boolean. Optional. Use when createAvd = true.
    Overwrite Existing AVD. Default: false.
    #avdOptionalArgs: # string. Optional. Use when createAvd = true. Create
    AVD Optional Arguments.
    # Emulator Options
    #startEmulator: false # boolean. Start and Stop Android Emulator.
    Default: false.
    #emulatorTimeout: '300' # string. Required when startEmulator = true.
    Timeout in Seconds. Default: 300.
    #emulatorHeadless: false # boolean. Optional. Use when startEmulator =
    true. Headless Display. Default: false.
    #emulatorOptionalArgs: '-no-snapshot-load -no-snapshot-save' # string.
    Optional. Use when startEmulator = true. Emulator Optional Arguments.
    Default: -no-snapshot-load -no-snapshot-save.
    #deleteAvd: false # boolean. Optional. Use when startEmulator = true.
    Delete AVD. Default: false.
```

Inputs

`gradleWrapper` - Location of Gradle Wrapper
`string`.

The location of the `gradlew` wrapper that is used for the build. Agents on Windows (including Microsoft-hosted agents) must use the `gradlew.bat` wrapper. Agents on Linux or macOS can use the `gradlew` shell script. Learn more about [the Gradle Wrapper](#).

`gradleProj` - Project Directory
`string`.

The relative path from the repo root to the root directory of the application. This is most likely to be where the `build.gradle` file is located.

`gradleArguments` - Gradle Arguments
`string`. Default value: `build`.

Provides any options to pass to the Gradle command line. Learn more about the [Gradle command line ↗](#).

`avdName` - Name
`string`. Required. Default value: `AndroidBuildEmulator`.

The name of the Android Virtual Device (AVD) to be started or created.

You must deploy your own agent to use this string. You cannot use a Microsoft-hosted pool if you want to create an AVD.

`createAvd` - Create AVD
`boolean`. Default value: `AndroidBuildEmulator`.

Creates the named Android Virtual Device (AVD).

`emulatorTarget` - AVD Target SDK
`string`. Required when `createAvd = true`. Default value: `android-19`.

The Android SDK version that the Android Virtual Device (AVD) targets.

emulatorDevice - AVD Device

`string`. Optional. Use when `createAvd = true`. Default value: `Nexus 5`.

The device pipeline that may be used. This can be a device index or an Id.

avdAbi - AVD ABI

`string`. Required when `createAvd = true`. Default value: `defaultarmeabi-v7a`.

The Application Binary Interface (ABI) to use for the Android Virtual Device (AVD). Learn more about [ABI Management](#).

avdForce - Overwrite Existing AVD

`boolean`. Optional. Use when `createAvd = true`. Default value: `false`.

Overwrites an existing AVD by passing `--force` to the `android create avd` command.

avdOptionalArgs - Create AVD Optional Arguments

`string`. Optional. Use when `createAvd = true`.

Creates additional arguments to pass to `android create avd`.

startEmulator - Start and Stop Android Emulator

`boolean`. Default value: `false`.

Starts and stops the Android emulator after the Android Build task finishes.

You must deploy your own agent to use this boolean. You cannot use a Microsoft-hosted pool if you want to use an emulator. Learn more about [Azure Pipeline agents](#).

emulatorTimeout - Timeout in Seconds

`string`. Required when `startEmulator = true`. Default value: `300`.

Defines how long (in seconds) the build will wait for the emulator to start.

emulatorHeadless - Headless Display

`boolean`. Optional. Use when `startEmulator = true`. Default value: `false`.

Starts the emulator with no GUI (headless mode) by using the `-no-skin -no-audio -no-window` value.

emulatorOptionalArgs - Emulator Optional Arguments

`string`. Optional. Use when `startEmulator = true`. Default value: `-no-snapshot-load -no-snapshot-save`.

Provides additional arguments to pass to the `emulator` command.

deleteAvd - Delete AVD

`boolean`. Optional. Use when `startEmulator = true`. Default value: `false`.

Deletes the AVD upon task completion.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: AndroidSDK
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any

Requirement	Description
Agent version	1.83.0 or greater
Task category	Build

AndroidSigning@3 - Android Signing v3 task

Article • 09/26/2023

Use this task in a pipeline to sign and align Android APK files.

Syntax

```
# Android Signing v3
# Sign and align Android APK files.
- task: AndroidSigning@3
  inputs:
    apkFiles: '**/*.apk' # string. Alias: files. Required. APK files.
  Default: **/*.apk.
  # Signing Options
    #apksign: true # boolean. Sign the APK. Default: true.
    apksignerKeystoreFile: # string. Alias: keystoreFile. Required when
    apksign = true. Keystore file.
    #apksignerKeystorePassword: # string. Alias: keystorePass. Optional. Use
    when apksign = true. Keystore password.
    #apksignerKeystoreAlias: # string. Alias: keystoreAlias. Optional. Use
    when apksign = true. Alias.
    #apksignerKeyPassword: # string. Alias: keyPass. Optional. Use when
    apksign = true. Key password.
    #apksignerVersion: 'latest' # string. Optional. Use when apksign = true.
    apksigner version. Default: latest.
    #apksignerArguments: '--verbose' # string. Optional. Use when apksign =
    true. apksigner arguments. Default: --verbose.
    #apksignerFile: # string. Alias: apksignerLocation. Optional. Use when
    apksign = true. apksigner location.
  # Zipalign Options
    #zipalign: true # boolean. Zipalign. Default: true.
    #zipalignVersion: 'latest' # string. Optional. Use when zipalign = true.
    Zipalign version. Default: latest.
    #zipalignFile: # string. Alias: zipalignLocation. Optional. Use when
    zipalign = true. Zipalign location.
```

Inputs

apkFiles - APK files

Input alias: files. string. Required. Default value: **/*.apk.

The relative path from the repo root to the APK(s) you want to sign. You can use [wildcards](#) to specify multiple files. For example:

- `outputs\apk*.apk` to sign all .APK files in the `outputs\apk\` subfolder.
- `**/bin/*.apk` to sign all .APK files in all `bin` subfolders.

`apksign` - Sign the APK

`boolean`. Default value: `true`.

Signs the APK with a provided Android Keystore file. Unsigned APKs can only run in an emulator. APKs must be signed to run on a device.

`apkSignerKeystoreFile` - Keystore file

Input alias: `keystoreFile`. `string`. Required when `apksign = true`.

The file path to the Android Keystore file that is used to sign the APK. This file must be uploaded to the [secure files](#) library, where it is securely stored with encryption. The Android Keystore file is removed from the agent machine when the pipeline completes.

The file can either be checked into source control or placed on the build machine directly by an administrator. It is recommended to encrypt the keystore file in source control and use the `Decrypt File` task to decrypt the file during the build.

`apkSignerKeystorePassword` - Keystore password

Input alias: `keystorePass`. `string`. Optional. Use when `apksign = true`.

The key password for the provided Android Keystore file.

ⓘ Important

Use a new variable with its lock enabled on the Variables pane to encrypt this value.

See [secret variables](#).

`apkSignerKeystoreAlias` - Alias

Input alias: `keystoreAlias`. `string`. Optional. Use when `apksign = true`.

The alias that identifies the public/private key pair to be used in the Android Keystore file.

`apksignerKeyPassword` - Key password

Input alias: `keyPass`. `string`. Optional. Use when `apksign = true`.

The key password for the alias and keystore file.

Important

Use a new variable with its lock enabled on the Variables pane to encrypt this value.

See [secret variables](#).

`apksignerVersion` - apksigner version

`string`. Optional. Use when `apksign = true`. Default value: `latest`.

The Android SDK build-tools version that the `apksigner` executable uses for the task.

`apksignerArguments` - apksigner arguments

`string`. Optional. Use when `apksign = true`. Default value: `--verbose`.

Provides options to pass to the `apksigner` command line. See the [apksigner documentation](#).

`apksignerFile` - apksigner location

Input alias: `apksignerLocation`. `string`. Optional. Use when `apksign = true`.

Specifies the location of the apksigner executable used during signing. This defaults to the apksigner found in the Android SDK version folder that your application builds against.

`zipalign` - Zipalign

`boolean`. Default value: `true`.

Select if you want to zipalign your package. This reduces the amount of RAM consumed by an app.

`zipalignVersion` - Zipalign version

`string`. Optional. Use when `zipalign = true`. Default value: `latest`.

The Android SDK build-tools version that the `zipalign` executable uses for the task.

`zipalignFile` - Zipalign location

Input alias: `zipalignLocation`. `string`. Optional. Use when `zipalign = true`.

Specifies the location of the `zipalign` executable used during signing. This defaults to the `zipalign` found in the Android SDK version folder that your application builds against.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task in a pipeline to sign and align Android APK files.

This version of the task uses apksigner instead of jarsigner to sign APKs.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: JDK
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : Setting variables is disabled

Requirement	Description
Agent version	2.182.1 or greater
Task category	Build

AndroidSigning@2 - Android Signing v2 task

Article • 09/26/2023

Use this task in a pipeline to sign and align Android APK files.

Syntax

YAML

```
# Android Signing v2
# Sign and align Android APK files.
- task: AndroidSigning@2
  inputs:
    apkFiles: '**/*.apk' # string. Alias: files. Required. APK files.
  Default: **/*.apk.
  # Signing Options
    #jarsign: true # boolean. Sign the APK. Default: true.
    jarsignerKeystoreFile: # string. Alias: keystoreFile. Required when
  jarsign = true. Keystore file.
    #jarsignerKeystorePassword: # string. Alias: keystorePass. Optional. Use
  when jarsign = true. Keystore password.
    #jarsignerKeystoreAlias: # string. Alias: keystoreAlias. Optional. Use
  when jarsign = true. Alias.
    #jarsignerKeyPassword: # string. Alias: keyPass. Optional. Use when
  jarsign = true. Key password.
    #jarsignerArguments: '-verbose -sigalg MD5withRSA -digestalg SHA1' #
  string. Optional. Use when jarsign = true. Jarsigner arguments. Default: -
  verbose -sigalg MD5withRSA -digestalg SHA1.
  # Zipalign Options
    #zipalign: true # boolean. Zipalign. Default: true.
    #zipalignFile: # string. Alias: zipalignLocation. Optional. Use when
  zipalign = true. Zipalign location.
```

Inputs

apkFiles - APK files

Input alias: files. string. Required. Default value: **/*.apk.

The relative path from the repo root to the APK(s) you want to sign. You can use [wildcards](#) to specify multiple files. For example:

- outputs\apk*.apk to sign all .APK files in the outputs\apk\ subfolder.
- **/bin/*.apk to sign all .APK files in all bin subfolders.

`jarsign` - Sign the APK

`boolean`. Default value: `true`.

Signs the APK with a provided keystore file. Unsigned APKs can only run in an emulator. APKs must be signed to run on a device.

`jarsignerKeystoreFile` - Keystore file

Input alias: `keystoreFile`. `string`. Required when `jarsign = true`.

The file path to the Android Keystore file that is used to sign the APK. This file must be uploaded to the [secure files](#) library, where it is securely stored with encryption. The Android Keystore file is removed from the agent machine when the pipeline completes.

The file can either be checked into source control or placed on the build machine directly by an administrator. It is recommended to encrypt the keystore file in source control and use the `Decrypt File` task to decrypt the file during the build.

`jarsignerKeystorePassword` - Keystore password

Input alias: `keystorePass`. `string`. Optional. Use when `jarsign = true`.

The password for the provided Android Keystore file.

ⓘ Important

Use a new variable with its lock enabled on the Variables tab to encrypt this value.

See [secret variables](#).

`jarsignerKeystoreAlias` - Alias

Input alias: `keystoreAlias`. `string`. Optional. Use when `jarsign = true`.

The alias that identifies the public/private key pair to be used in the Android Keystore file.

`jarsignerKeyPassword` - Key password

Input alias: `keyPass`. `string`. Optional. Use when `jarsign = true`.

The key password for the alias and Android Keystore file.

ⓘ Important

Use a new variable with its lock enabled on the Variables tab to encrypt this value.

See [secret variables](#).

`jarsignerArguments` - Jarsigner arguments

`string`. Optional. Use when `jarsign = true`. Default value: `-verbose -sigalg MD5withRSA -digestalg SHA1`.

Provides options to pass to the `jarsigner` command line.

`zipalign` - Zipalign

`boolean`. Default value: `true`.

Select this boolean if you want to zipalign your package. This reduces the amount of RAM consumed by an app.

`zipalignFile` - Zipalign location

Input alias: `zipalignLocation`. `string`. Optional. Use when `zipalign = true`.

Specifies the location of the zipalign executable used during signing. This defaults to the zipalign found in the Android SDK version folder that your application builds against.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build

Requirement	Description
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: JDK
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : Setting variables is disabled
Agent version	2.182.1 or greater
Task category	Build

AndroidSigning@1 - Android Signing v1 task

Article • 09/26/2023

Use this task in a pipeline to sign and align Android APK files.

Syntax

YAML

```
# Android Signing v1
# Sign and align Android APK files.
- task: AndroidSigning@1
  inputs:
    files: # string. Required. APK Files.
  # Signing Options
    #jarsign: true # boolean. Sign the APK. Default: true.
    keystoreFile: # string. Required when jarsign = true. Keystore File.
    #keystorePass: # string. Optional. Use when jarsign = true. Keystore Password.
    #keystoreAlias: # string. Optional. Use when jarsign = true. Alias.
    #keyPass: # string. Optional. Use when jarsign = true. Key Password.
    #jarsignerArguments: '-verbose -sigalg MD5withRSA -digestalg SHA1' # string. Optional. Use when jarsign = true. Jarsigner Arguments. Default: -verbose -sigalg MD5withRSA -digestalg SHA1.
  # Zipalign Options
    #zipalign: true # boolean. Zipalign. Default: true.
    #zipalignLocation: # string. Optional. Use when zipalign = true. Zipalign Location.
```

Inputs

`files` - APK Files

`string`. Required.

The relative path from the repo root to the APK(s) you want to sign. You can use [wildcards](#) to specify multiple files. For example:

- `outputs\apk*.apk` to sign all .APK files in the `outputs\apk\` subfolder.
- `**/bin/*.apk` to sign all .APK files in all `bin` subfolders.

The default value: `/apk` Argument aliases: `apkFiles`

jarsign - Sign the APK

`boolean`. Default value: `true`.

Signs the APK with a provided Android Keystore file. Unsigned APKs can only run in an emulator. APKs must be signed to run on a device.

keystoreFile - Keystore File

`string`. Required when `jarsign = true`.

The file path to the Android Keystore file that is used to sign the APK. This file must be uploaded to the [secure files](#) library, where it is securely stored with encryption. The Android Keystore file is removed from the agent machine when the pipeline completes.

The file can either be checked into source control or placed on the build machine directly by an administrator. It is recommended to encrypt the keystore file in source control and use the [Decrypt File](#) task to decrypt the file during the build.

Argument aliases: `apkSignerKeystoreFile`

keystorePass - Keystore Password

`string`. Optional. Use when `jarsign = true`.

The key password for the provided Android Keystore file.

ⓘ Important

Use a new variable with its lock enabled on the Variables pane to encrypt this value.

See [secret variables](#).

Argument aliases: `apkSignerKeystorePassword`

keystoreAlias - Alias

`string`. Optional. Use when `jarsign = true`.

The alias that identifies the public/private key pair to be used in the Android Keystore file.

Argument aliases: `apkSignerKeystoreAlias`

`keyPass` - Key Password

`string`. Optional. Use when `jarsign = true`.

The key password for the alias and Android Keystore file.

Important

Use a new variable with its lock enabled on the Variables pane to encrypt this value.

See [secret variables](#).

`jarsignerArguments` - Jarsigner Arguments

`string`. Optional. Use when `jarsign = true`. Default value: `-verbose -sigalg MD5withRSA -digestalg SHA1`.

Provides options to pass to the `jarsigner` command line.

`zipalign` - Zipalign

`boolean`. Default value: `true`.

Select this boolean if you want to zipalign your package. This reduces the amount of RAM consumed by an app.

`zipalignLocation` - Zipalign Location

`string`. Optional. Use when `zipalign = true`.

Specifies the location of the zipalign executable used during signing. This defaults to the zipalign found in the Android SDK version folder that your application builds against.

Argument aliases: `zipalignFile`

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: JDK, AndroidSDK
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.98.1 or greater
Task category	Build

Ant@1 - Ant v1 task

Article • 09/26/2023

Use this task to build with Apache Ant.

Syntax

YAML

```
# Ant v1
# Build with Apache Ant.
- task: Ant@1
  inputs:
    buildFile: 'build.xml' # string. Alias: antBuildFile. Required. Ant
    build file. Default: build.xml.
    #options: # string. Options.
    #targets: # string. Target(s).
    # JUnit Test Results
    #publishJUnitResults: true # boolean. Publish to Azure Pipelines.
    Default: true.
    testResultsFiles: '**/TEST-*.xml' # string. Required when
    publishJUnitResults = true. Test results files. Default: **/TEST-*.xml.
    #testRunTitle: # string. Optional. Use when publishJUnitResults = true.
    Test run title.
    # Code Coverage
    #codeCoverageToolOptions: 'None' # 'None' | 'Cobertura' | 'JaCoCo'.
    Alias: codeCoverageTool. Code coverage tool. Default: None.
    codeCoverageClassDirectories: '.' # string. Alias:
    classFilesDirectories. Required when codeCoverageTool != None. Class files
    directories. Default: ..
    #codeCoverageClassFilter: # string. Alias: classFilter. Optional. Use
    when codeCoverageTool != None. Class inclusion/exclusion filters.
    #codeCoverageSourceDirectories: # string. Alias: srcDirectories.
    Optional. Use when codeCoverageTool != None. Source files directories.
    #codeCoverageFailIfEmpty: false # boolean. Alias: failIfCoverageEmpty.
    Optional. Use when codeCoverageTool != None. Fail when code coverage results
    are missing. Default: false.
    # Advanced
    #antHomeDirectory: # string. Alias: antHomeUserInputPath. Set ANT_HOME
    path.
    javaHomeOption: 'JDKVersion' # 'JDKVersion' | 'Path'. Alias:
    javaHomeSelection. Required. Set JAVA_HOME by. Default: JDKVersion.
    #jdkVersionOption: 'default' # 'default' | '1.11' | '1.10' | '1.9' |
    '1.8' | '1.7' | '1.6'. Alias: jdkVersion. Optional. Use when
    javaHomeSelection = JDKVersion. JDK version. Default: default.
    #jdkUserInputDirectory: # string. Alias: jdkUserInputPath. Required when
    javaHomeSelection = Path. JDK path.
    #jdkArchitectureOption: 'x64' # 'x86' | 'x64'. Alias: jdkArchitecture.
    Optional. Use when jdkVersion != default. JDK architecture. Default: x64.
```

Inputs

`buildFile` - Ant build file

Input alias: `antBuildFile`. `string`. Required. Default value: `build.xml`.

The relative path from the repository root to the Ant build file.

For more information about build files, see [Using Apache Ant](#).

`options` - Options

`string`.

Provides options to pass to the Ant command line. You can provide your own properties (for example, `-DmyProperty=myPropertyValue`) and also use built-in variables (for example, `-DcollectionId=$(system.collectionId)`). Alternatively, the built-in variables are already set as environment variables during the build and can be passed directly (for example, `-DcollectionIdAsEnvVar=%SYSTEM_COLLECTIONID%`).

See [Running Apache Ant](#).

`targets` - Target(s)

`string`.

An optional, space-separated list of targets to build. If not specified, the `default` target will be used. If no `default` target is defined, Ant 1.6.0 and later will build all top-level tasks.

See [Using Apache Ant Targets](#).

`publishJUnitResults` - Publish to Azure Pipelines

`boolean`. Default value: `true`.

Select this option to publish JUnit test results produced by the Ant build to Azure Pipelines. Each test results file matching `Test Results Files` will be published as a test run in Azure Pipelines.

`testResultsFiles` - Test results files

`string`. Required when `publishJUnitResults = true`. Default value: `**/TEST-*.xml`.

The test results file path. Wildcards can be used. For more information, see the [file matching patterns reference](#). For example, `**/TEST-*.xml` for all XML files whose name starts with `TEST-`.

`testRunTitle` - Test run title

`string`. Optional. Use when `publishJUnitResults = true`.

Provides a name for the test run.

`codeCoverageToolOptions` - Code coverage tool

Input alias: `codeCoverageTool`. `string`. Allowed values: `None`, `Cobertura`, `Jacoco`. Default value: `None`.

Selects the code coverage tool.

If you are using the [Microsoft-hosted agents](#), then the tools are set up for you. If you are using the on-premises [Windows agent](#), you must ensure the agent is set up for either JaCoco or Cobertura.

- JaCoCo - ensure that `jacocoant.jar` is available in the `lib` folder of Ant installation.
Learn more about [JaCoCo Ant tasks](#).
- Cobertura - ensure that an environment variable `COBERTURA_HOME` points to the Cobertura .jar files location. Learn more about [Cobertura with Ant tasks](#).

After you select one of these tools, the following arguments appear:

`codeCoverageClassFilesDirectories` - Class files directories

Input alias: `classFilesDirectories`. `string`. Required when `codeCoverageTool != None`.

Default value: `..`.

The comma-separated list of relative paths from the Ant build file to directories containing class files and archive files (`.jar`, `.war`, etc.). Code coverage is reported for class files in these directories. For example: `target/classes,target/testClasses`.

`codeCoverageClassFilter` - Class inclusion/exclusion filters

Input alias: `classFilter`. `string`. Optional. Use when `codeCoverageTool != None`.

The comma-separated list of filters to include or exclude classes from collecting code coverage. For example: `+:com., +:org., - :my.app*..`

`codeCoverageSourceDirectories` - Source files directories

Input alias: `srcDirectories`. `string`. Optional. Use when `codeCoverageTool != None`.

The comma-separated list of relative paths from the Ant build file to source code directories. Code coverage reports will use these to highlight source code. For example:

`src/java,src/Test.`

`codeCoverageFailIfEmpty` - Fail when code coverage results are missing

Input alias: `failIfCoverageEmpty`. `boolean`. Optional. Use when `codeCoverageTool != None`. Default value: `false`.

Fails the build if the code coverage did not produce any results to publish.

`antHomeDirectory` - Set ANT_HOME path

Input alias: `antHomeUserInputPath`. `string`.

If set, overrides any existing ANT_HOME environment variable with the given path.

`javaHomeOption` - Set JAVA_HOME by

Input alias: `javaHomeSelection`. `string`. Required. Allowed values: `JDKVersion` (JDK Version), `Path`. Default value: `JDKVersion`.

Sets JAVA_HOME either by selecting a JDK version that will be discovered during builds or by manually entering a JDK path.

`jdkVersionOption` - JDK version

Input alias: `jdkVersion`. `string`. Optional. Use when `javaHomeSelection = JDKVersion`. Allowed values: `default`, `1.11` (JDK 11), `1.10` (JDK 10 (out of support)), `1.9` (JDK 9 (out of support)), `1.8` (JDK 8), `1.7` (JDK 7), `1.6` (JDK 6 (out of support)). Default value: `default`.

Attempts to discover the path to the selected JDK version and sets JAVA_HOME accordingly.

`jdkUserInputDirectory` - JDK path

Input alias: `jdkUserInputPath`. `string`. Required when `javaHomeSelection = Path`.

Sets JAVA_HOME to the given path.

jdkArchitectureOption - JDK architecture

Input alias: `jdkArchitecture`. `string`. Optional. Use when `jdkVersion != default`.

Allowed values: `x86`, `x64`. Default value: `x64`.

Optionally supplies the architecture (x86, x64) of the JDK.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to build with Apache Ant.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: ant
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.89.0 or greater
Task category	Build

AppCenterDistribute@3 - App Center distribute v3 task

Article • 09/26/2023

Use this task to distribute app builds to testers and users via Visual Studio App Center.

Syntax

YAML

```
# App Center distribute v3
# Distribute app builds to testers and users via Visual Studio App Center.
- task: AppCenterDistribute@3
  inputs:
    serverEndpoint: # string. Required. App Center service connection.
    appSlug: # string. Required. App slug.
    appFile: # string. Alias: app. Required. Binary file path.
    #buildVersion: # string. Build version.
    releaseNotesOption: 'input' # 'input' | 'file'. Alias:
    releaseNotesSelection. Required. Create release notes. Default: input.
      releaseNotesInput: # string. Required when releaseNotesSelection =
    input. Release notes.
      #releaseNotesFile: # string. Required when releaseNotesSelection = file.
    Release notes file.
    #isMandatory: false # boolean. Require users to update to this release.
    Default: false.
    destinationType: 'groups' # 'groups' | 'store'. Required. Release
    destination. Default: groups.
    #distributionGroupId: # string. Alias: destinationGroupIds. Optional.
    Use when destinationType = groups. Destination IDs.
    #destinationStoreId: # string. Required when destinationType = store.
    Destination ID.
    #isSilent: # boolean. Optional. Use when destinationType = groups. Do
    not notify testers. Release will still be available to install.
    # Symbols
    #symbolsOption: 'Apple' # 'Apple' | 'Android' | 'UWP'. Alias:
    symbolsType. Symbols type. Default: Apple.
    #symbolsPath: # string. Optional. Use when symbolsType == AndroidNative
    || symbolsType = Windows. Symbols path.
    #appxsymPath: # string. Optional. Use when symbolsType = UWP. Symbols
    path (*.appxsym).
    #symbolsDsymFiles: # string. Alias: dsymPath. Optional. Use when
    symbolsType = Apple. dSYM path.
    #symbolsMappingTxtFile: # string. Alias: mappingTxtPath. Optional. Use
    when symbolsType = Android. Mapping file.
    #nativeLibrariesPath: # string. Optional. Use when symbolsType ==
    Android. Native Library File Path.
```

```
#symbolsIncludeParentDirectory: # boolean. Alias: packParentFolder.  
Optional. Use when symbolsType = Apple. Include all items in parent folder.
```

Inputs

`serverEndpoint` - App Center service connection

`string`. Required.

Selects the service connection for Visual Studio App Center. To create one, click the `Manage` link and create a new service connection.

`appSlug` - App slug

`string`. Required.

The app slug is in the format of `{username}/{app_identifier}`. To locate `{username}` and `{app_identifier}` for an app, click on its name from [App Center](#), and the resulting URL is in the format of `https://appcenter.ms/users/**{username}**/apps/**{app_identifier}**`. If you are using orgs, the app slug is of the format `{orgname}/{app_identifier}`.

`appFile` - Binary file path

Input alias: `app`. `string`. Required.

The relative path from the repo root to the APK/AAB or IPA file you want to publish.

`buildVersion` - Build version

`string`.

The build version of the uploading binary which needs to be specified for `.zip` and `.msi`. This value will be ignored unless the platform is WPF or WinForms.

`symbolsOption` - Symbols type

Input alias: `symbolsType`. `string`. Allowed values: `Apple`, `Android`, `UWP`. Default value: `Apple`.

Includes symbol files to receive symbolicated stack traces in App Center Diagnostics.

symbolsPath - Symbols path

`string`. Optional. Use when `symbolsType == AndroidNative || symbolsType = Windows`.

The relative path from the repo root to the symbols folder.

appxsymPath - Symbols path (*.appxsym)

`string`. Optional. Use when `symbolsType = UWP`.

The relative path to the APPXSYM symbols file. Path may contain [wildcards](#).

symbolsDsymFiles - dSYM path

Input alias: `dsymPath`. `string`. Optional. Use when `symbolsType = Apple`.

The relative path from the repo root to dSYM folder. Path may contain [wildcards](#).

symbolsMappingTxtFile - Mapping file

Input alias: `mappingTxtPath`. `string`. Optional. Use when `symbolsType = Android`.

The relative path from the repo root to Android's `mapping.txt` file.

nativeLibrariesPath - Native Library File Path

`string`. Optional. Use when `symbolsType == Android`.

The relative path from the repo root to the additional native libraries you want to publish (e.g. .so files).

symbolsIncludeParentDirectory - Include all items in parent folder

Input alias: `packParentFolder`. `boolean`. Optional. Use when `symbolsType = Apple`.

Uploads the selected symbols file or folder and all other items inside the same parent folder. This is required for React Native apps.

releaseNotesOption - Create release notes

Input alias: `releaseNotesSelection`. `string`. Required. Allowed values: `input` (Enter Release Notes), `file` (Select Release Notes File). Default value: `input`.

The release notes will be attached to the release and shown to testers on the installation page.

releaseNotesInput - Release notes

`string`. Required when `releaseNotesSelection = input`.

The release notes for this version.

releaseNotesFile - Release notes file

`string`. Required when `releaseNotesSelection = file`.

Selects a UTF-8 encoded text file which contains the release notes for this version.

isMandatory - Require users to update to this release

`boolean`. Default value: `false`.

The App Center Distribute SDK required to mandate an update. Testers are automatically prompted to update.

destinationType - Release destination

`string`. Required. Allowed values: `groups`, `store`. Default value: `groups`.

Each release is distributed to either groups or a store.

distributionGroupId - Destination IDs

Input alias: `destinationGroupIds`. `string`. Optional. Use when `destinationType = groups`.

The IDs of the distribution groups who will receive the build release. Leave it empty to use the default group, and use commas or semicolons to separate multiple IDs.

destinationStoreId - Destination ID

`string`. Required when `destinationType = store`.

The IDs of the distribution store that will receive the build release.

isSilent - Do not notify testers. Release will still be available to install.

`boolean`. Optional. Use when `destinationType = groups`.

Testers do not receive an email for new releases.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to distribute app builds to testers and users through App Center.

- [Sign up with App Center](#) ↗ first.
- For details about using this task, see the App Center documentation article [Deploy Azure DevOps Builds with App Center](#).

Examples

This example pipeline builds an Android app, runs tests, and publishes the app using App Center Distribute.

YAML

```
# Android
# Build your Android project with Gradle.
# Add steps that test, sign, and distribute the APK, save build artifacts,
and more:
# https://learn.microsoft.com/azure/devops/pipelines/ecosystems/android

pool:
  vmImage: 'macOS-latest'
steps:

- script: sudo npm install -g appcenter-cli
- script: appcenter login --token {YOUR_TOKEN}

- task: Gradle@2
  inputs:
    workingDirectory: ''
    gradleWrapperFile: 'gradlew'
    gradleOptions: '-Xmx3072m'
    publishJUnitResults: false
```

```

testResultsFiles: '**/TEST-*.xml'
tasks: build

- task: CopyFiles@2
  inputs:
    contents: '**/*.apk'
    targetFolder: '$(build.artifactStagingDirectory)'

- task: PublishBuildArtifacts@1
  inputs:
    pathToPublish: '$(build.artifactStagingDirectory)'
    artifactName: 'outputs'
    artifactType: 'container'

# Run tests using the App Center CLI
- script: appcenter test run espresso --app "{APP_CENTER_SLUG}" --devices "{DEVICE}" --app-path {APP_FILE} --test-series "master" --locale "en_US" --build-dir {PAT_ESPRESSO} --debug

# Distribute the app
- task: AppCenterDistribute@3
  inputs:
    serverEndpoint: 'AppCenter'
    appSlug: '$(APP_CENTER_SLUG)'
    appFile: '${APP_FILE}' # Relative path from the repo root to the APK or IPA file you want to publish
    symbolsOption: 'Android'
    releaseNotesOption: 'input'
    releaseNotesInput: 'Here are the release notes for this version.'
    destinationType: 'groups'

```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Deploy

AppCenterDistribute@2 - App Center distribute v2 task

Article • 09/26/2023

Use this task to distribute app builds to testers and users via Visual Studio App Center.

This task is deprecated; use [AppCenterDistribute@3](#).

Syntax

YAML

```
# App Center distribute v2
# Distribute app builds to testers and users via Visual Studio App Center.
- task: AppCenterDistribute@2
  inputs:
    serverEndpoint: # string. Required. App Center service connection.
    appSlug: # string. Required. App slug.
    appFile: # string. Alias: app. Required. Binary file path.
    releaseNotesOption: 'input' # 'input' | 'file'. Alias:
    releaseNotesSelection. Required. Create release notes. Default: input.
    releaseNotesInput: # string. Required when releaseNotesSelection =
input. Release notes.
    #releaseNotesFile: # string. Required when releaseNotesSelection = file.
    Release notes file.
    #isMandatory: false # boolean. Require users to update to this release.
    Default: false.
    #distributionGroupId: # string. Alias: destinationIds | destinationId.
    Destination IDs.
    # Symbols
    #symbolsOption: 'Apple' # 'Apple'. Alias: symbolsType. Symbols type.
    Default: Apple.
    #symbolsPath: # string. Optional. Use when symbolsType == AndroidNative
    || symbolsType = Windows. Symbols path.
    #symbolsPdbFiles: '**/*.pdb' # string. Alias: pdbPath. Optional. Use
    when symbolsType = UWP. Symbols path (*.pdb). Default: **/*.pdb.
    #symbolsDsymFiles: # string. Alias: dsymPath. Optional. Use when
    symbolsType = Apple. dSYM path.
    #symbolsMappingTxtFile: # string. Alias: mappingTxtPath. Optional. Use
    when symbolsType = AndroidJava. Mapping file.
    #symbolsIncludeParentDirectory: # boolean. Alias: packParentFolder.
    Include all items in parent folder.
```

Inputs

serverEndpoint - App Center service connection

`string`. Required.

Selects the service connection for Visual Studio App Center. To create one, click the [Manage](#) link and create a new service connection.

appSlug - App slug

`string`. Required.

The app slug is in the format of `{username}/{app_identifier}`. To locate `{username}` and `{app_identifier}` for an app, click on its name from [App Center](#), and the resulting URL is in the format of `https://appcenter.ms/users/**{username}**/apps/**{app_identifier}**`. If you are using orgs, the app slug is of the format `{orgname}/{app_identifier}`.

appFile - Binary file path

Input alias: `app`. `string`. Required.

The relative path from the repo root to the APK or IPA file you want to publish.

symbolsOption - Symbols type

Input alias: `symbolsType`. `string`. Allowed values: `Apple`. Default value: `Apple`.

Includes symbol files to receive symbolicated stack traces in App Center Diagnostics.

symbolsPath - Symbols path

`string`. Optional. Use when `symbolsType == AndroidNative || symbolsType = Windows`.

The relative path from the repo root to the symbols folder.

symbolsPdbFiles - Symbols path (*.pdb)

Input alias: `pdbPath`. `string`. Optional. Use when `symbolsType = UWP`. Default value: `**/*.pdb`.

The relative path from the repo root to PDB symbols files. Path may contain [wildcards](#).

symbolsDsymFiles - dSYM path

Input alias: `dsymPath`. `string`. Optional. Use when `symbolsType = Apple`.

The relative path from the repo root to dSYM folder. Path may contain [wildcards](#).

symbolsMappingTxtFile - Mapping file

Input alias: `mappingTxtPath`. `string`. Optional. Use when `symbolsType = AndroidJava`.

The relative path from the repo root to Android's `mapping.txt` file.

symbolsIncludeParentDirectory - Include all items in parent folder

Input alias: `packParentFolder`. `boolean`.

Uploads the selected symbols file or folder and all other items inside the same parent folder. This is required for React Native apps.

releaseNotesOption - Create release notes

Input alias: `releaseNotesSelection`. `string`. Required. Allowed values: `input` (Enter Release Notes), `file` (Select Release Notes File). Default value: `input`.

Release notes are attached to the release and shown to testers on the installation page.

releaseNotesInput - Release notes

`string`. Required when `releaseNotesSelection = input`.

The release notes for this version.

releaseNotesFile - Release notes file

`string`. Required when `releaseNotesSelection = file`.

Selects a UTF-8 encoded text file which contains the release notes for this version.

isMandatory - Require users to update to this release

`boolean`. Default value: `false`.

The App Center Distribute SDK required to mandate update. Testers are automatically prompted to update.

`distributionGroupId` - Destination IDs

Input alias: `destinationIds` | `destinationId`. `string`.

The IDs of the distribution stores or groups who will receive the build release. Leave it empty to use the default group.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

This task is deprecated. Use [AppCenterDistribute@3](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Deploy

AppCenterDistribute@1 - App Center distribute v1 task

Article • 09/26/2023

Use this task to distribute app builds to testers and users via App Center and Visual Studio App Center.

This task is deprecated; use [AppCenterDistribute@3](#).

Syntax

YAML

```
# App Center distribute v1
# Distribute app builds to testers and users via Visual Studio App Center.
- task: AppCenterDistribute@1
  inputs:
    serverEndpoint: # string. Required. App Center service connection.
    appSlug: # string. Required. App slug.
    appFile: # string. Alias: app. Required. Binary file path.
    releaseNotesOption: 'input' # 'input' | 'file'. Alias:
    releaseNotesSelection. Required. Create release notes. Default: input.
    releaseNotesInput: # string. Required when releaseNotesSelection =
    input. Release notes.
    #releaseNotesFile: # string. Required when releaseNotesSelection = file.
    Release notes file.
    #isMandatory: false # boolean. Require users to update to this release.
    Default: false.
    #distributionGroupId: # string. Alias: destinationId. Destination ID.
    # Symbols
    #symbolsOption: 'Apple' # 'Apple'. Alias: symbolsType. Symbols type.
    Default: Apple.
    #symbolsPath: # string. Optional. Use when symbolsType == AndroidNative
    || symbolsType = Windows. Symbols path.
    #symbolsPdbFiles: '**/*.pdb' # string. Alias: pdbPath. Optional. Use
    when symbolsType = UWP. Symbols path (*.pdb). Default: **/*.pdb.
    #symbolsDsymFiles: # string. Alias: dsymPath. Optional. Use when
    symbolsType = Apple. dSYM path.
    #symbolsMappingTxtFile: # string. Alias: mappingTxtPath. Optional. Use
    when symbolsType = AndroidJava. Mapping file.
    #symbolsIncludeParentDirectory: # boolean. Alias: packParentFolder.
    Include all items in parent folder.
```

Inputs

serverEndpoint - App Center service connection

`string`. Required.

Selects the service connection for Visual Studio App Center. To create one, click the `Manage` link and create a new service connection.

appSlug - App slug

`string`. Required.

The app slug is in the format of `{username}/{app_identifier}`. To locate `{username}` and `{app_identifier}` for an app, click on its name from [App Center](#). The resulting URL is in the format of `https://appcenter.ms/users/**{username}**/apps/**{app_identifier}**`. If you are using orgs, the app slug is of the format `{orgname}/{app_identifier}`.

appFile - Binary file path

Input alias: `app`. `string`. Required.

The relative path from the repo root to the APK or IPA file you want to publish.

symbolsOption - Symbols type

Input alias: `symbolsType`. `string`. Allowed values: `Apple`. Default value: `Apple`.

Includes symbol files to receive symbolicated stack traces in App Center Diagnostics.

symbolsPath - Symbols path

`string`. Optional. Use when `symbolsType == AndroidNative || symbolsType = Windows`.

The relative path from the repo root to the symbols folder.

symbolsPdbFiles - Symbols path (*.pdb)

Input alias: `pdbPath`. `string`. Optional. Use when `symbolsType = UWP`. Default value: `**/*.pdb`.

The relative path from the repo root to `.pdb` symbols files. Path may contain [wildcards](#).

symbolsDsymFiles - dSYM path

Input alias: `dsymPath`. `string`. Optional. Use when `symbolsType = Apple`.

The relative path from the repo root to the dSYM folder. Path may contain [wildcards](#).

symbolsMappingTxtFile - Mapping file

Input alias: `mappingTxtPath`. `string`. Optional. Use when `symbolsType = AndroidJava`.

The relative path from the repo root to Android's `mapping.txt` file.

symbolsIncludeParentDirectory - Include all items in parent folder

Input alias: `packParentFolder`. `boolean`.

Uploads the selected symbols file or folder and all other items inside the same parent folder. This is required for React Native apps.

releaseNotesOption - Create release notes

Input alias: `releaseNotesSelection`. `string`. Required. Allowed values: `input` (Enter Release Notes), `file` (Select Release Notes File). Default value: `input`.

The release notes are attached to the release and shown to testers on the installation page.

releaseNotesInput - Release notes

`string`. Required when `releaseNotesSelection = input`.

The release notes for this version.

releaseNotesFile - Release notes file

`string`. Required when `releaseNotesSelection = file`.

Selects a UTF-8 encoded text file which contains the release notes for this version.

isMandatory - Require users to update to this release

`boolean`. Default value: `false`.

The App Center Distribute SDK required to mandate an update. Testers are automatically prompted to update.

`distributionGroupId` - Destination ID

Input alias: `destinationId`. `string`.

The IDs of the distribution stores or groups who will receive the build release. Leave it empty to use the default group.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

This task is deprecated. Use [AppCenterDistribute@3](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Deploy

AppCenterDistribute@0 - App Center Distribute v0 task

Article • 09/26/2023

Use this task to distribute app builds to testers and users via App Center.

Syntax

YAML

```
# App Center distribute v0
# Distribute app builds to testers and users via App Center.
- task: AppCenterDistribute@0
  inputs:
    serverEndpoint: # string. Required. App Center connection.
    appSlug: # string. Required. App slug.
    appFile: # string. Alias: app. Required. Binary file path.
    releaseNotesOption: 'input' # 'input' | 'file'. Alias:
      releaseNotesSelection. Required. Create release notes. Default: input.
      releaseNotesInput: # string. Required when releaseNotesSelection =
        input. Release notes.
      #releaseNotesFile: # string. Required when releaseNotesSelection = file.
      Release notes file.
    #distributionGroupId: # string. Distribution group ID.
    # Symbols
    #symbolsOption: 'Apple' # 'Apple'. Alias: symbolsType. Symbols type.
    Default: Apple.
    #symbolsPath: # string. Optional. Use when symbolsType == AndroidNative
    || symbolsType = Windows. Symbols path.
    #symbolsPdbFiles: '**/*.pdb' # string. Alias: pdbPath. Optional. Use
    when symbolsType = UWP. Symbols path (*.pdb). Default: **/*.pdb.
    #symbolsDsymFiles: # string. Alias: dsymPath. Optional. Use when
    symbolsType = Apple. dSYM path.
    #symbolsMappingTxtFile: # string. Alias: mappingTxtPath. Optional. Use
    when symbolsType = AndroidJava. Mapping file.
    #symbolsIncludeParentDirectory: # boolean. Alias: packParentFolder.
    Include all items in parent folder.
```

Inputs

`serverEndpoint` - App Center connection

string. Required.

Selects the service endpoint for your Visual Studio App Center connection. To create one, click the [Manage](#) link and create a new service endpoint.

appSlug - App slug

`string`. Required.

The app slug is in the format of `{username}/{app_identifier}`. To locate `{username}` and `{app_identifier}` for an app, click on its name from [App Center](#), and the resulting URL is in the format of `https://appcenter.ms/users/**{username}**/apps/**{app_identifier}**`. If you are using orgs, the app slug is of the format `{orgname}/{app_identifier}`.

appFile - Binary file path

Input alias: `app`. `string`. Required.

The relative path from the repo root to the APK or IPA file you want to publish.

symbolsOption - Symbols type

Input alias: `symbolsType`. `string`. Allowed values: `Apple`. Default value: `Apple`.

Includes symbol files to receive symbolicated stack traces in App Center Diagnostics.

symbolsPath - Symbols path

`string`. Optional. Use when `symbolsType == AndroidNative || symbolsType = Windows`.

The relative path from the repo root to the symbols folder.

symbolsPdbFiles - Symbols path (*.pdb)

Input alias: `pdbPath`. `string`. Optional. Use when `symbolsType = UWP`. Default value: `**/*.pdb`.

The relative path from the repo root to `.pdb` symbols files. Path may contain [wildcards](#).

symbolsDsymFiles - dSYM path

Input alias: `dsymPath`. `string`. Optional. Use when `symbolsType = Apple`.

The relative path from the repo root to the dSYM folder. Path may contain [wildcards](#).

`symbolsMappingTxtFile` - Mapping file

Input alias: `mappingTxtPath`. `string`. Optional. Use when `symbolsType = AndroidJava`.

The relative path from the repo root to Android's `mapping.txt` file.

`symbolsIncludeParentDirectory` - Include all items in parent folder

Input alias: `packParentFolder`. `boolean`.

Uploads the selected symbols file or folder and all other items inside the same parent folder. This is required for React Native apps.

`releaseNotesOption` - Create release notes

Input alias: `releaseNotesSelection`. `string`. Required. Allowed values: `input` (Enter Release Notes), `file` (Select Release Notes File). Default value: `input`.

The release notes will be attached to the release and shown to testers on the installation page.

`releaseNotesInput` - Release notes

`string`. Required when `releaseNotesSelection = input`.

The release notes for this version.

`releaseNotesFile` - Release notes file

`string`. Required when `releaseNotesSelection = file`.

Selects a UTF-8 encoded text file which contains the release notes for this version.

`distributionGroupId` - Distribution group ID

`string`.

The IDs of the distribution groups who will receive the build release.. Leave it empty to use the default group.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

This task is deprecated. Use [AppCenterDistribute@3](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Deploy

AppCenterTest@1 - App Center test v1 task

Article • 09/26/2023

Test app packages with Visual Studio App Center.

Syntax

YAML

```
# App Center test v1
# Test app packages with Visual Studio App Center.
- task: AppCenterTest@1
  inputs:
    appFile: # string. Alias: app. Required. Binary application file path.
    artifactsDirectory: '$(Build.ArtifactStagingDirectory)/AppCenterTest' # string. Alias: artifactsDir. Required. Artifacts directory. Default: $(Build.ArtifactStagingDirectory)/AppCenterTest.
    # Prepare Tests
    #prepareTests: true # boolean. Alias: enablePrepare. Prepare tests.
    Default: true.
    frameworkOption: 'appium' # 'appium' | 'espresso' | 'calabash' |
    'uitest' | 'xcuitest'. Alias: framework. Required when enablePrepare = true.
    Test framework. Default: appium.
    #appiumBuildDirectory: # string. Alias: appiumBuildDir. Required when
    enablePrepare = true && framework = appium. Build directory.
    #espressoBuildDirectory: # string. Alias: espressoBuildDir. Optional.
    Use when enablePrepare = true && framework = espresso. Build directory.
    #espressoTestApkFile: # string. Alias: espressoTestApkPath. Optional.
    Use when enablePrepare = true && framework = espresso. Test APK path.
    #calabashProjectDirectory: # string. Alias: calabashProjectDir. Required
    when enablePrepare = true && framework = calabash. Project directory.
    #calabashConfigFile: # string. Optional. Use when enablePrepare = true
    && framework = calabash. Cucumber config file.
    #calabashProfile: # string. Optional. Use when enablePrepare = true &&
    framework = calabash. Profile to run.
    #calabashSkipConfigCheck: false # boolean. Optional. Use when
    enablePrepare = true && framework = calabash. Skip Configuration Check.
    Default: false.
    #uitestBuildDirectory: # string. Alias: uitestBuildDir. Required when
    enablePrepare = true && framework = uitest. Build directory.
    #uitestStorePath: # string. Optional. Use when enablePrepare = true &&
    framework = uitest. Store file.
    #uitestStorePassword: # string. Alias: uitestStorePass. Optional. Use
    when enablePrepare = true && framework = uitest. Store password.
    #uitestKeyAlias: # string. Optional. Use when enablePrepare = true &&
    framework = uitest. Key alias.
    #uitestKeyPassword: # string. Alias: uitestKeyPass. Optional. Use when
    enablePrepare = true && framework = uitest. Key password.
```

```

#uiTestToolsDirectory: # string. Alias: uitestToolsDir. Optional. Use
when enablePrepare = true && framework = uitest. Test tools directory.
#signInfo: # string. Optional. Use when framework = calabash ||
framework = uitest. Signing information.
#xcUITestBuildDirectory: # string. Alias: xcuitestBuildDir. Optional.
Use when enablePrepare = true && framework = xcuitest. Build directory.
#xcUITestIpaFile: # string. Alias: xcuitestTestIpaPath. Optional. Use
when enablePrepare = true && framework = xcuitest. Test IPA path.
#prepareOptions: # string. Alias: prepareOpts. Optional. Use when
enablePrepare = true. Additional options.
# Run Tests
#runTests: true # boolean. Alias: enableRun. Run tests. Default: true.
credentialsOption: 'serviceEndpoint' # 'serviceEndpoint' | 'inputs'.
Alias: credsType. Required when enableRun = true. Authentication method.
Default: serviceEndpoint.
#serverEndpoint: # string. Required when enableRun = true && credsType =
serviceEndpoint. App Center service connection.
#username: # string. Required when enableRun = true && credsType =
inputs. App Center username.
#password: # string. Required when enableRun = true && credsType =
inputs. App Center password.
appSlug: # string. Required when enableRun = true. App slug.
devices: # string. Required when enableRun = true. Devices.
#series: 'master' # string. Optional. Use when enableRun = true. Test
series. Default: master.
#dsymDirectory: # string. Alias: dsymDir. Optional. Use when enableRun =
true. dSYM directory.
localeOption: 'en_US' # 'da_DK' | 'nl_NL' | 'en_GB' | 'en_US' | 'fr_FR'
| 'de_DE' | 'ja_JP' | 'ru_RU' | 'es_MX' | 'es_ES' | 'user'. Alias: locale.
Required when enableRun = true. System language. Default: en_US.
#userDefinedLocale: # string. Optional. Use when enableRun = true &&
locale = user. Other locale.
#loginOptions: # string. Alias: loginOpts. Optional. Use when enableRun
= true && credsType = inputs. Additional options for login.
#runOptions: # string. Alias: runOpts. Optional. Use when enableRun =
true. Additional options for run.
#skipWaitingForResults: false # boolean. Alias: async. Optional. Use
when enableRun = true. Do not wait for test result. Default: false.
# Advanced
#cliFile: # string. Alias: cliLocationOverride. App Center CLI location.
#showDebugOutput: false # boolean. Alias: debug. Enable debug output.
Default: false.

```

Inputs

appFile - Binary application file path

Input alias: `app`. `string`. Required.

The relative path from the repo root to the APK or IPA file you want to test.

`artifactsDirectory` - Artifacts directory

Input alias: `artifactsDir`. `string`. Required. Default value:

`$(Build.ArtifactStagingDirectory)/AppCenterTest.`

Specifies where to place the artifacts produced by the prepare step and used by the run step. This directory will be created if it does not already exist.

`prepareTests` - Prepare tests

Input alias: `enablePrepare`. `boolean`. Default value: `true`.

When set to `true`, this input prepares the tests.

`frameworkOption` - Test framework

Input alias: `framework`. `string`. Required when `enablePrepare = true`. Allowed values:

`appium`, `espresso`, `calabash`, `uitest` (Xamarin UI Test), `xcuitest`. Default value: `appium`.

`appiumBuildDirectory` - Build directory

Input alias: `appiumBuildDir`. `string`. Required when `enablePrepare = true && framework = appium`.

The path to the directory with the Appium tests.

`espressoBuildDirectory` - Build directory

Input alias: `espressoBuildDir`. `string`. Optional. Use when `enablePrepare = true && framework = espresso`.

The path to the Espresso output directory.

`espressoTestApkFile` - Test APK path

Input alias: `espressoTestApkPath`. `string`. Optional. Use when `enablePrepare = true && framework = espresso`.

The path to the APK file with the Espresso tests. If not set, `build-dir` is used to discover it. A wildcard is allowed.

`calabashProjectDirectory` - Project directory

Input alias: `calabashProjectDir`. `string`. Required when `enablePrepare = true &&`

```
framework = calabash.
```

The path to the Calabash workspace directory.

calabashConfigFile - Cucumber config file

```
string. Optional. Use when enablePrepare = true && framework = calabash.
```

The path to the Cucumber configuration file, usually cucumber.yml.

calabashProfile - Profile to run

```
string. Optional. Use when enablePrepare = true && framework = calabash.
```

The profile to run. This value must exist in the Cucumber configuration file.

calabashSkipConfigCheck - Skip Configuration Check

```
boolean. Optional. Use when enablePrepare = true && framework = calabash. Default value: false.
```

When set to `true`, this input skips the configuration check specified by the Cucumber profile.

uitestBuildDirectory - Build directory

Input alias: `uitestBuildDir`. `string`. Required when `enablePrepare = true && framework = uitest`.

The path to the directory with the built test assemblies.

uitestStorePath - Store file

```
string. Optional. Use when enablePrepare = true && framework = uitest.
```

The path to the store file that is used to sign the app.

uitestStorePassword - Store password

Input alias: `uitestStorePass`. `string`. Optional. Use when `enablePrepare = true && framework = uitest`.

The password of the store file that is used to sign the app. To encrypt this value, use a new variable with its lock enabled on the Variables tab.

uitestKeyAlias - Key alias

`string`. Optional. Use when `enablePrepare = true && framework = uitest`.

Specifies the alias that identifies the public/private key pair that is used in the store file.

uiTestKeyPassword - Key password

Input alias: `uitestKeyPass`. `string`. Optional. Use when `enablePrepare = true && framework = uitest`.

Specifies the key password for the alias and store file. To encrypt this value, use a new variable with its lock enabled on the Variables tab.

uiTestToolsDirectory - Test tools directory

Input alias: `uitestToolsDir`. `string`. Optional. Use when `enablePrepare = true && framework = uitest`.

The path to the directory with the Xamarin UI test tools that contain *test-cloud.exe*.

signInfo - Signing information

`string`. Optional. Use when `framework = calabash || framework = uitest`.

Signs the test server.

xcUITestBuildDirectory - Build directory

Input alias: `xcuitestBuildDir`. `string`. Optional. Use when `enablePrepare = true && framework = xcuitest`.

The path to the build output directory (usually `$(ProjectDir)/Build/Products/Debug-iphoneos`).

xcUITestIpaFile - Test IPA path

Input alias: `xcuitestTestIpaPath`. `string`. Optional. Use when `enablePrepare = true && framework = xcuitest`.

The path to the .ipa file with the XCUI Test tests.

prepareOptions - Additional options

Input alias: `prepareOpts`. `string`. Optional. Use when `enablePrepare = true`.

The additional arguments that are passed to the App Center test prepare step.

runTests - Run tests

Input alias: `enableRun`. `boolean`. Default value: `true`.

Runs the tests.

credentialsOption - Authentication method

Input alias: `credsType`. `string`. Required when `enableRun = true`. Allowed values: `serviceEndpoint` (App Center service connection), `inputs` (Credentials). Default value: `serviceEndpoint`.

Uses the App Center service connection or enters the credentials to connect to Visual Studio App Center.

serverEndpoint - App Center service connection

`string`. Required when `enableRun = true && credsType = serviceEndpoint`.

Selects the service connection for Visual Studio App Center. If needed, click the Manage link to create a new service connection.

username - App Center username

`string`. Required when `enableRun = true && credsType = inputs`.

Create your username by visiting the [App Center sign in page](#), and provide the value here.

password - App Center password

`string`. Required when `enableRun = true && credsType = inputs`.

Set your password by visiting the [App Center sign in page](#), and provide the value here. Variables defined in build or release pipelines as `$(passwordVariable)` are accepted. You may mark the variable type as `secret` to secure it.

`appSlug` - App slug

`string`. Required when `enableRun = true`.

The app slug is in the format of `<username>/<app_identifier>`. To locate the `<username>` and `<app_identifier>` for an app, click its name from [Visual Studio App Center](#). The resulting URL is in the format

`https://appcenter.ms/users/<username>/apps/<app_identifier>`.

`devices` - Devices

`string`. Required when `enableRun = true`.

Identifies the devices this test will run against. Copy and paste this string when you define a new test run from the Visual Studio App Center Test beacon.

`series` - Test series

`string`. Optional. Use when `enableRun = true`. Default value: `master`.

The series name for organizing the test runs (for example: master, production, beta).

`dsymDirectory` - dSYM directory

Input alias: `dsymDir`. `string`. Optional. Use when `enableRun = true`.

The path to the iOS symbol files.

`localeOption` - System language

Input alias: `locale`. `string`. Required when `enableRun = true`. Allowed values: `da_DK` (Danish (Denmark)), `nl_NL` (Dutch (Netherlands)), `en_GB` (English (United Kingdom)), `en_US` (English (United States)), `fr_FR` (French (France)), `de_DE` (German (Germany)), `ja_JP` (Japanese (Japan)), `ru_RU` (Russian (Russia)), `es_MX` (Spanish (Mexico)), `es_ES` (Spanish (Spain)), `user` (Other). Default value: `en_US`.

Utilize if your language isn't displayed. Select `Other` and enter its locale, such as `en_US`.

`userDefinedLocale` - Other locale

`string`. Optional. Use when `enableRun = true && locale = user`.

Enters any two-letter ISO-639 language code along with any two-letter ISO 3166 country code in the format `<language>_<country>`, such as `en_US`.

loginOptions - Additional options for login

Input alias: `loginOpts`. `string`. Optional. Use when `enableRun = true && credsType = inputs`.

The additional arguments that are passed to the Visual Studio App Center login step.

runOptions - Additional options for run

Input alias: `runOpts`. `string`. Optional. Use when `enableRun = true`.

The additional arguments that are passed to the Visual Studio App Center test run.

skipWaitingForResults - Do not wait for test result

Input alias: `async`. `boolean`. Optional. Use when `enableRun = true`. Default value: `false`.

Executes a command asynchronously and exits when the tests are uploaded without waiting for the test results.

cliFile - App Center CLI location

Input alias: `cliLocationOverride`. `string`.

The path to the Visual Studio App Center CLI on the build or release agent.

showDebugOutput - Enable debug output

Input alias: `debug`. `boolean`. Default value: `false`.

Adds `--debug` to the Visual Studio App Center CLI.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

This task lets you run test suites against an application binary (.apk or .ipa file) using App Center Test.

- [Sign up with App Center](#) first.
- For details about using this task, see the App Center documentation article [Using Azure DevOps for UI Testing](#).

Examples

This example runs Espresso tests on an Android app using the App Center Test task.

YAML

```
steps:
- task: AppCenterTest@1
  displayName: 'Espresso Test - Synchronous'
  inputs:
    appFile: 'Espresso/espresso-app.apk'
    artifactsDirectory: '$(Build.ArtifactStagingDirectory)/AppCenterTest'
    frameworkOption: espresso
    espressoBuildDirectory: Espresso
    serverEndpoint: 'myAppCenterServiceConnection'
    appSlug: 'xplatbg1/EspressoTests'
    devices: a84c93af
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Test

ArchiveFiles@2 - Archive files v2 task

Article • 09/26/2023

Archive files using compression formats such as .7z, .tar, .gz, and .zip.

Syntax

YAML

```
# Archive files v2
# Compress files into .7z, .tar.gz, or .zip.
- task: ArchiveFiles@2
  inputs:
    rootFolderOrFile: '$(Build.BinariesDirectory)' # string. Required. Root
    folder or file to archive. Default: $(Build.BinariesDirectory).
    #includeRootFolder: true # boolean. Prepend root folder name to archive
    paths. Default: true.
    # Archive
    archiveType: 'zip' # 'zip' | '7z' | 'tar' | 'wim'. Required. Archive
    type. Default: zip.
    #sevenZipCompression: 'normal' # 'ultra' | 'maximum' | 'normal' | 'fast'
    | 'fastest' | 'none'. Optional. Use when archiveType = 7z. 7z compression.
    Default: normal.
    #tarCompression: 'gz' # 'gz' | 'bz2' | 'xz' | 'none'. Optional. Use when
    archiveType = tar. Tar compression. Default: gz.
    archiveFile: '$(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip' #
    string. Required. Archive file to create. Default:
    $(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip.
    #replaceExistingArchive: true # boolean. Replace existing archive.
    Default: true.
    #verbose: false # boolean. Force verbose output. Default: false.
    #quiet: false # boolean. Force quiet output. Default: false.
```

Inputs

`rootFolderOrFile` - Root folder or file to archive

`string`. Required. Default value: `$(Build.BinariesDirectory)`.

Name of the root folder or the file path to files to add to the archive. For folders, everything in the named folder is added to the archive.

`includeRootFolder` - Prepend root folder name to archive paths

`boolean`. Default value: `true`.

Prepends the root folder name to file paths in the archive. Otherwise, all file paths will start one level lower.

For example, if the root folder is: `/home/user/output/classes/` and the file path: `com/acme/Main.class`. The resulting archive will contain: `classes/com/acme/Main.class`. Otherwise, the resulting archive will contain: `com/acme/Main.class`.

`archiveType` - Archive type

`string`. Required. Allowed values: `zip`, `7z`, `tar`, `wim`. Default value: `zip`.

Specifies a compression format. Valid formats include:

- `zip` - Default. Choose this format for all zip compatible types such as `.zip`, `.jar`, `.war`, `.ear`
- `7z` - 7-Zip format, (`.7z`)
- `tar` - tar format, use for compressed tars including `.tar.gz`, `.tar.bz2`, `.tar.xz`
- `wim` - wim format, `.wim`

Example, to create an archive named `foo.jar`:

- Select compression format `zip`
- Specify the name of the archive file to create: `foo.jar`

`sevenZipCompression` - 7z compression

`string`. Optional. Use when `archiveType = 7z`. Allowed values: `ultra`, `maximum`, `normal`, `fast`, `fastest`, `none`. Default value: `normal`.

Set compression level or `None` to create an uncompressed `.7z` file.

`tarCompression` - Tar compression

`string`. Optional. Use when `archiveType = tar`. Allowed values: `gz`, `bz2`, `xz`, `none`. Default value: `gz`.

Set a compression format or choose `None` to create an uncompressed `.tar` file.

- `gz` - Default format for gzip compression (`.tar.gz`, `.tar.tgz`, `.taz`)
- `bz2` - bzip2 compression (`.tar.bz2`, `.tz2`, `.tbz2`)
- `xz` - xz compression (`.tar.xz`, `.txz`)

`archiveFile` - Archive file to create

`string`. Required. Default value:

`$(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip`.

Specify the name of the archive file to create. For example, to create `foo.tgz`:

- Set archive type: `tar`
- Set tar compression: `gz`

`replaceExistingArchive` - Replace existing archive

`boolean`. Default value: `true`.

By default, overwrites an existing archive. Otherwise, when set to `false`, uncompressed tar files are added to the existing archive.

Supported file formats that can be added to an existing archive:

- `zip`
- `7z`
- `tar` - Only uncompressed
- `wim`

`verbose` - Force verbose output

`boolean`. Default value: `false`.

If set to true, forces tools to use verbose output. Overrides the 'quiet' setting.

`quiet` - Force quiet output

`boolean`. Default value: `false`.

If set to `true`, forces tools to use quiet output. The `verbose` setting (or equivalent) can override this setting.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to create an archive file from a source folder. Standard archive formats are supported including .zip, .jar, .war, .ear, .tar, .7z, and more.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : Setting variables is disabled
Agent version	2.182.1 or greater
Task category	Utility

ArchiveFiles@1 - Archive Files v1 task

Article • 09/26/2023

Archive files using compression formats such as .7z, .rar, .tar.gz, and .zip.

Syntax

YAML

```
# Archive Files v1
# Archive files using compression formats such as .7z, .rar, .tar.gz, and
.zip.
- task: ArchiveFiles@1
  inputs:
    rootFolder: '$(Build.BinariesDirectory)' # string. Required. Root folder
(or file) to archive. Default: $(Build.BinariesDirectory).
    #includeRootFolder: true # boolean. Prefix root folder name to archive
paths. Default: true.
    # Archive
    archiveType: 'default' # 'default' | '7z' | 'tar' | 'wim'. Required.
Archive type. Default: default.
    #tarCompression: 'gz' # 'gz' | 'bz2' | 'xz' | 'none'. Optional. Use when
archiveType = tar. Tar compression. Default: gz.
    archiveFile: '$(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip' #
string. Required. Archive file to create. Default:
$(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip.
    #replaceExistingArchive: true # boolean. Replace existing archive.
Default: true.
```

Inputs

rootFolder - Root folder (or file) to archive

`string`. Required. Default value: `$(Build.BinariesDirectory)`.

Name of the root folder or file to archive. For folders, everything in the named folder is added to the archive.

includeRootFolder - Prefix root folder name to archive paths

`boolean`. Default value: `true`.

By default, prepends the root folder name to file paths within the archive. When set to `false`, all file paths will start one level lower.

For example, if the root folder path is: `/home/user/output/classes/` and the file path `com/acme/Main.class`. The resulting archive will contain `classes/com/acme/Main.class`. Otherwise, the resulting archive will contain `com/acme/Main.class`.

`archiveType` - Archive type

`string`. Required. Allowed values: `default` (zip), `7z`, `tar`, `wim`. Default value: `default`.

Specifies a compression format.

For example, to create an archive named `foo.jar`:

- Set compression format: `zip`
- Set the archive name: `foo.jar`

For all tar files (including compressed ones), choose `tar`.

`tarCompression` - Tar compression

`string`. Optional. Use when `archiveType = tar`. Allowed values: `gz`, `bz2`, `xz`, `none`. Default value: `gz`.

Selects a compression scheme or `none` to create an uncompressed tar file.

`archiveFile` - Archive file to create

`string`. Required. Default value:

`$(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip`.

Specify the name of the archive file to create.

For example, to create `foo.tgz`:

- Set archive type: `tar`
- Set tar compression: `gz`

`replaceExistingArchive` - Replace existing archive

`boolean`. Default value: `true`.

Overwrites an existing archive. If not specified, files are added to the archive.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

There is a newer version of the Archive Files task available.

- [Archive Files v2](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Utility

See also

- [Archive Files v2](#)

AzureResourceManagerTemplateDeployment@3 - ARM template deployment v3 task

Article • 09/26/2023

Use this task to deploy an Azure Resource Manager (ARM) template to all deployment scopes.

Syntax

YAML

```
# ARM template deployment v3
# Deploy an Azure Resource Manager (ARM) template to all the deployment
scopes.
- task: AzureResourceManagerTemplateDeployment@3
  inputs:
    # Azure Details
    deploymentScope: 'Resource Group' # 'Management Group' | 'Subscription'
    | 'Resource Group'. Required. Deployment scope. Default: Resource Group.
    azureResourceManagerConnection: # string. Alias: ConnectedServiceName.
    Required. Azure Resource Manager connection.
    #subscriptionId: # string. Alias: subscriptionName. Required when
    deploymentScope != Management Group. Subscription.
    #action: 'Create Or Update Resource Group' # 'Create Or Update Resource
    Group' | 'DeleteRG'. Required when deploymentScope = Resource Group. Action.
    Default: Create Or Update Resource Group.
    #resourceGroupName: # string. Required when deploymentScope = Resource
    Group. Resource group.
    #location: # string. Required when action = Create Or Update Resource
    Group || deploymentScope != Resource Group. Location.
    # Template
    #templateLocation: 'Linked artifact' # 'Linked artifact' | 'URL of the
    file'. Required when action = Create Or Update Resource Group ||
    deploymentScope != Resource Group. Template location. Default: Linked
    artifact.
    #csmFileLink: # string. Required when templateLocation = URL of the file
    && action = Create Or Update Resource Group || deploymentScope != Resource
    Group. Template link.
    #csmParametersFileLink: # string. Optional. Use when templateLocation =
    URL of the file && action = Create Or Update Resource Group ||
    deploymentScope != Resource Group. Template parameters link.
    #csmFile: # string. Required when templateLocation = Linked artifact &&
    action = Create Or Update Resource Group || deploymentScope != Resource
    Group. Template.
    #csmParametersFile: # string. Optional. Use when templateLocation =
    Linked artifact && action = Create Or Update Resource Group ||
```

```
deploymentScope != Resource Group. Template parameters.  
  #overrideParameters: # string. Optional. Use when action = Create Or  
  Update Resource Group || deploymentScope != Resource Group. Override  
  template parameters.  
  #deploymentMode: 'Incremental' # 'Incremental' | 'Complete' |  
  'Validation'. Required when action = Create Or Update Resource Group ||  
  deploymentScope != Resource Group. Deployment mode. Default: Incremental.  
  # Advanced  
  #deploymentName: # string. Optional. Use when action = Create Or Update  
  Resource Group || deploymentScope != Resource Group. Deployment name.  
  #deploymentOutputs: # string. Optional. Use when action = Create Or  
  Update Resource Group || deploymentScope != Resource Group. Deployment  
  outputs.  
  #addSpnToEnvironment: false # boolean. Optional. Use when action =  
  Create Or Update Resource Group || deploymentScope != Resource Group. Access  
  service principal details in override parameters. Default: false.  
  #useWithoutJSON: false # boolean. Optional. Use when action = Create Or  
  Update Resource Group || deploymentScope != Resource Group. Use individual  
  output values without JSON.Stringify applied. Default: false.
```

Inputs

`deploymentScope` - Deployment scope

`string`. Required. Allowed values: `Management Group`, `Subscription`, `Resource Group`.
Default value: `Resource Group`.

The scope of the deployment. Learn more about [deployment scopes](#).

`azureResourceManagerConnection` - Azure Resource Manager connection

Input alias: `ConnectedServiceName`. `string`. Required.

Specifies the Azure Resource Manager service connection with access to the selected deployment scope.

`subscriptionId` - Subscription

Input alias: `subscriptionName`. `string`. Required when `deploymentScope != Management Group`.

Specifies the Azure subscription.

ⓘ Important

The specified value must be the subscription ID and not the subscription name.

action - Action

`string`. Required when `deploymentScope = Resource Group`. Allowed values: `Create Or Update Resource Group`, `DeleteRG` (Delete resource group). Default value: `Create Or Update Resource Group`.

The action to be performed on the Azure resources or resource group.

resourceGroupName - Resource group

`string`. Required when `deploymentScope = Resource Group`.

Provides the name of a resource group.

location - Location

`string`. Required when `action = Create Or Update Resource Group || deploymentScope != Resource Group`.

Resource Group deployment scopes: The location to deploy the resource group. If the resource group already exists in the Azure subscription, then this value will be ignored.
Other deployment scopes: The location to store deployment metadata.

templateLocation - Template location

`string`. Required when `action = Create Or Update Resource Group || deploymentScope != Resource Group`. Allowed values: `Linked artifact`, `URL of the file`. Default value: `Linked artifact`.

The location of the Template and the Parameters JSON files. Choose **Linked artifact** if the files are part of the linked code/build artifacts. For linked artifacts, you can also specify the path to a Bicep file. Choose **URL of the file** if the JSON files are located at any publicly accessible http/https URLs. To use a file stored in a private storage account, retrieve and include the shared access signature (SAS) token in the URL of the template. Example: `<blob_storage_url>/template.json?`. To upload a parameters file to a storage account and generate a SAS token, you could use [Azure file copy task](#) or follow the steps using [PowerShell](#) or [Azure CLI](#).

csmFileLink - Template link

`string`. Required when `templateLocation = URL of the file && action = Create Or Update Resource Group || deploymentScope != Resource Group`.

Specifies the URL of the template file. An example URL:

```
https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-vm-simple-windows/azuredeploy.json
```

To deploy a template stored in a private storage account, retrieve and include the shared access signature (SAS) token in the URL of the template. Example:

<blob_storage_url>/template.json?<SASToken>. To upload a template file (or a linked template) to a storage account and generate a SAS token, use the [Azure file copy](#) task or follow the steps using [PowerShell](#) or [Azure CLI](#).

To view the template parameters in a grid, click on  next to the override template parameters text box. This feature requires that CORS rules are enabled at the source. If the templates are in an Azure storage blob, refer to [Cross-Origin Resource Sharing](#) to enable CORS.

csmParametersFileLink - Template parameters link

`string`. Optional. Use when `templateLocation = URL of the file && action = Create Or Update Resource Group || deploymentScope != Resource Group`.

Specifies the URL of the parameters file. An example URL:

```
https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-vm-simple-windows/azuredeploy.parameters.json
```

To use a file stored in a private storage account, retrieve and include the shared access signature (SAS) token in the URL of the template. Example:

<blob_storage_url>/template.json?<SASToken>. To upload a template file (or a linked template) to a storage account and generate a SAS token, use the [Azure file copy](#) task or follow the steps using [PowerShell](#) or [Azure CLI](#).

To view the template parameters in a grid, click on  next to Override template parameters text box. This feature requires that CORS rules are enabled at the source. If the templates are in an Azure storage blob, refer to [Cross-Origin Resource Sharing](#) to enable CORS.

csmFile - Template

`string`. Required when `templateLocation = Linked artifact && action = Create Or Update Resource Group || deploymentScope != Resource Group`.

Specifies the path or a pattern pointing to the Azure Resource Manager template. Learn more about [Azure Resource Manager templates](#). To get started immediately, use [this](#)

[sample template](#). Supports Bicep files when the Azure CLI version > 2.20.0.

csmParametersFile - Template parameters

`string`. Optional. Use when `templateLocation = Linked artifact && action = Create Or Update Resource Group || deploymentScope != Resource Group`.

Specify the path or a pattern pointing for the parameters file for the Azure Resource Manager template. Supports Bicep files when the Azure CLI version > 2.20.0.

overrideParameters - Override template parameters

`string`. Optional. Use when `action = Create Or Update Resource Group || deploymentScope != Resource Group`.

Specifies the template parameters to override.

To view the template parameters in a grid, click on `...` next to the Override Parameters textbox. This feature requires that CORS rules are enabled at the source. If the templates are in the Azure storage blob, reference this string to enable CORS, or type the template parameters to override in the textbox.

Example: `-storageName fabrikam -adminUsername $(vmusername) -adminPassword (ConvertTo-SecureString -String '$(password)' -AsPlainText -Force) -azureKeyVaultName $(fabrikamFibre)`.

If the parameter value has multiple words, enclose the words in quotes, even if you're passing the value by using variables. For example, `-name "parameter value" -name2 "$(var)"`. To override object type parameters, use stringified JSON objects. For example, `-options ["option1"] -map {"key1": "value1" }`.

deploymentMode - Deployment mode

`string`. Required when `action = Create Or Update Resource Group || deploymentScope != Resource Group`. Allowed values: `Incremental`, `Complete`, `Validation` (Validation only). Default value: `Incremental`.

The `Incremental` mode handles deployments as incremental updates to the resource group. It leaves unchanged resources that exist in the resource group but are not specified in the template.

`Complete` mode deletes resources that are not in your template. Complete mode takes relatively more time than incremental mode. If the task times out, consider increasing

the timeout or changing to the [Incremental](#) mode.

Warning

Complete mode will delete all the existing resources in the resource group that are not specified in the template. Do review if the resource group you're deploying to doesn't contain any necessary resources that are not specified in the template.

[Validate](#) mode enables you to find problems with the template before creating actual resources.

Note

The [Validate](#) mode always creates a resource group, even if no resources are deployed. Learn more about [deployment modes](#).

deploymentName - Deployment name

`string`. Optional. Use when `action = Create Or Update Resource Group || deploymentScope != Resource Group`.

Specifies the name of the resource group deployment to create.

deploymentOutputs - Deployment outputs

`string`. Optional. Use when `action = Create Or Update Resource Group || deploymentScope != Resource Group`.

Provides a name for the variable for the output variable, which contains the outputs section of the current deployment object in string format. You can use the [ConvertFrom-Json](#) PowerShell cmdlet to parse the JSON object and access the individual output values. Learn more about [deployment outputs](#).

addSpnToEnvironment - Access service principal details in override parameters

`boolean`. Optional. Use when `action = Create Or Update Resource Group || deploymentScope != Resource Group`. Default value: `false`.

Adds the service principal ID and key of the Azure endpoint chosen to be the script's execution environment. The variables `$servicePrincipalId` and `$servicePrincipalKey` can be in override parameters, such as `-key $servicePrincipalKey`.

`useWithoutJSON` - Use individual output values without `JSON.Stringify` applied

`boolean`. Optional. Use when `action = Create Or Update Resource Group || deploymentScope != Resource Group`. Default value: `false`.

Individual output values are being converted via `JSON.Stringify` by default. If you want to use the output values as it is without converting them via `JSON.Stringify`, enable this option. For more details refer to [this ↗](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

ⓘ Note

This task supports Bicep files when the Azure CLI version > 2.20.0.

- Added support for deployment at all the deployment scopes.
 - Removed all the VM related actions.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.

Requirement	Description
Command restrictions	Any
Settable variables	Any
Agent version	2.119.1 or greater
Task category	Deploy

AzureWebPowerShellDeployment@1 - Azure App Service Classic (Deprecated) v1 task

Article • 09/26/2023

This task creates or updates Azure App Service using Azure PowerShell.

This task is deprecated.

Syntax

YAML

```
# Azure App Service Classic (Deprecated) v1
# Create or update Azure App Service using Azure PowerShell.
- task: AzureWebPowerShellDeployment@1
  inputs:
    ConnectedServiceName: # string. Required. Azure Subscription (Classic).
    WebSiteLocation: # string. Required. Web App Location.
    WebSiteName: # string. Required. Web App Name.
    #Slot: # string. Slot.
    Package: # string. Required. Web Deploy Package.
    #doNotDelete: false # boolean. Set DoNotDelete flag. Default: false.
    #AdditionalArguments: # string. Additional Arguments.
```

Inputs

`ConnectedServiceName` - Azure Subscription (Classic)

`string`. Required.

Specifies the Azure Classic subscription to target for deployment.

`WebSiteLocation` - Web App Location

`string`. Required.

Specifies a location for the website.

`WebSiteName` - Web App Name

`string`. Required.

Specifies the website name or selects it from the list.

Note: Only the websites associated with the default app service plan for the selected region are listed.

Slot - Slot

`string`.

Specifies the slot.

Package - Web Deploy Package

`string`. Required.

The path to the Visual Studio Web Deploy package under the default artifact directory.

doNotDelete - Set DoNotDelete flag

`boolean`. Default value: `false`.

When set to `true`, additional files in the Web Deployment package are preserved while publishing the website.

AdditionalArguments - Additional Arguments

`string`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release

Requirement	Description
Runs on	Agent
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: azureps
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.103.0 or greater
Task category	Deploy

AzureRmWebAppDeployment@4 - Azure App Service deploy v4 task

Article • 09/26/2023

Use this task to deploy to Azure App Service a web, mobile, or API app using Docker, Java, .NET, .NET Core, Node.js, PHP, Python, or Ruby.

ⓘ Note

Use [AzureFunctionApp@1](#) to deploy Azure Functions apps.

Syntax

YAML

```
# Azure App Service deploy v4
# Deploy to Azure App Service a web, mobile, or API app using Docker, Java,
.NET, .NET Core, Node.js, PHP, Python, or Ruby.
- task: AzureRmWebAppDeployment@4
  inputs:
    ConnectionType: 'AzureRM' # 'AzureRM' | 'PublishProfile'. Required.
    Connection type. Default: AzureRM.
    azureSubscription: # string. Alias: ConnectedServiceName. Required when
    ConnectionType = AzureRM. Azure subscription.
    #PublishProfilePath: '$(System.DefaultWorkingDirectory)/**/*.*.pubxml' #
    string. Required when ConnectionType = PublishProfile. Publish profile path.
    Default: $(System.DefaultWorkingDirectory)/**/*.*.pubxml.
    #PublishProfilePassword: # string. Required when ConnectionType =
    PublishProfile. Publish profile password.
    appType: 'webApp' # 'webApp' | 'webAppLinux' | 'webAppContainer' |
    'webAppHyperVContainer' | 'functionApp' | 'functionAppLinux' |
    'functionAppContainer' | 'apiApp' | 'mobileApp'. Alias: WebAppKind. Required
    when ConnectionType = AzureRM. App Service type. Default: webApp.
    WebAppName: # string. Required when ConnectionType = AzureRM. App
    Service name.
    #deployToSlotOrASE: false # boolean. Alias: DeployToSlotOrASEFlag.
    Optional. Use when ConnectionType = AzureRM && WebAppKind != "".
    Deploy to
    Slot or App Service Environment. Default: false.
    #ResourceGroupName: # string. Required when DeployToSlotOrASEFlag =
    true. Resource group.
    #SlotName: 'production' # string. Required when DeployToSlotOrASEFlag =
    true. Slot. Default: production.
    #DockerNamespace: # string. Required when WebAppKind = webAppContainer
    || WebAppkind = functionAppContainer || WebAppkind = webAppHyperVContainer.
    Registry or Namespace.
    #DockerRepository: # string. Required when WebAppKind = webAppContainer
```

```
|| WebAppkind = functionAppContainer || WebAppkind = webAppHyperVContainer.  
Image.  
    #DockerImageTag: # string. Optional. Use when WebAppKind =  
webAppContainer || WebAppkind = functionAppContainer || WebAppkind =  
webAppHyperVContainer. Tag.  
    #VirtualApplication: # string. Optional. Use when WebAppKind !=  
webAppLinux && WebAppKind != webAppContainer && WebAppkind !=  
functionAppContainer && WebAppKind != functionApp && webAppKind !=  
functionAppLinux && WebAppKind != "". Virtual application.  
    #packageForLinux: '$(System.DefaultWorkingDirectory)/**/*.zip' # string.  
Alias: Package. Required when ConnectionType = PublishProfile || WebAppKind  
= webApp || WebAppKind = apiApp || WebAppKind = functionApp || WebAppKind =  
mobileApp || WebAppKind = webAppLinux || webAppKind = functionAppLinux.  
Package or folder. Default: $(System.DefaultWorkingDirectory)/**/*.zip.  
    #RuntimeStack: # string. Optional. Use when WebAppKind = webAppLinux.  
Runtime Stack.  
    #RuntimeStackFunction: # 'DOTNET|2.2' | 'DOTNET|3.1' | 'JAVA|8' |  
'JAVA|11' | 'NODE|8' | 'NODE|10' | 'NODE|12' | 'NODE|14' | 'PYTHON|3.6' |  
'PYTHON|3.7' | 'PYTHON|3.8'. Optional. Use when WebAppKind =  
functionAppLinux. Runtime Stack.  
    #StartupCommand: # string. Optional. Use when WebAppKind = webAppLinux  
|| WebAppKind = webAppContainer || WebAppkind = functionAppContainer ||  
WebAppKind = functionAppLinux || WebAppkind = webAppHyperVContainer. Startup  
command.  
    # Post Deployment Action  
    #ScriptType: # 'Inline Script' | 'File Path'. Optional. Use when  
ConnectionType = AzureRM && WebAppKind != "" && WebAppKind !=  
webAppContainer && WebAppkind != functionAppContainer. Deployment script  
type.  
    #InlineScript: ':: You can provide your deployment commands here. One  
command per line.' # string. Required when ScriptType == Inline Script &&  
ConnectionType = AzureRM && WebAppKind != "" && WebAppKind !=  
webAppContainer && WebAppkind != functionAppContainer. Inline Script.  
Default: :: You can provide your deployment commands here. One command per  
line..  
    #ScriptPath: # string. Required when ScriptType == File Path &&  
ConnectionType = AzureRM && WebAppKind != "" && WebAppKind !=  
webAppContainer && WebAppkind != functionAppContainer. Deployment script  
path.  
    # File Transforms & Variable Substitution Options  
    #WebConfigParameters: # string. Optional. Use when WebAppKind !=  
webAppContainer && WebAppkind != functionAppContainer && WebAppKind !=  
webAppLinux && webAppKind != functionAppLinux && Package NotEndsWith .war.  
Generate web.config parameters for Python, Node.js, Go and Java apps.  
    #enableXmlTransform: false # boolean. Alias: XmlTransformation.  
Optional. Use when WebAppKind != webAppContainer && WebAppkind !=  
functionAppContainer && WebAppKind != webAppLinux && webAppKind !=  
functionAppLinux && Package NotEndsWith .war. XML transformation. Default:  
false.  
    #enableXmlVariableSubstitution: false # boolean. Alias:  
XmlVariableSubstitution. Optional. Use when WebAppKind != webAppContainer &&  
WebAppkind != functionAppContainer && WebAppKind != webAppLinux &&  
webAppKind != functionAppLinux && Package NotEndsWith .war. XML variable  
substitution. Default: false.  
    #JSONFiles: # string. Optional. Use when WebAppKind != webAppContainer
```

```

    && WebAppkind != functionAppContainer && WebAppKind != webAppLinux &&
    webAppKind != functionAppLinux && Package NotEndsWith .war. JSON variable
    substitution.

        # Application and Configuration Settings
            #AppSettings: # string. Optional. Use when ConnectionType = AzureRM. App
            settings.

                #ConfigurationSettings: # string. Optional. Use when ConnectionType =
                AzureRM. Configuration settings.

                    # Additional Deployment Options
                        #enableCustomDeployment: false # boolean. Alias: UseWebDeploy. Optional.
                        Use when ConnectionType = AzureRM && WebAppKind != webAppLinux && WebAppKind
                        != webAppContainer && WebAppkind != functionAppContainer && webAppKind !=
                        functionAppLinux && WebAppKind != "" && Package NotEndsWith .war && Package
                        NotEndsWith .jar. Select deployment method. Default: false.

                            #DeploymentType: 'webDeploy' # 'webDeploy' | 'zipDeploy' | 'runFromZip'.
                            Required when UseWebDeploy == true && ConnectionType = AzureRM && WebAppKind
                            != webAppLinux && WebAppKind != webAppContainer && WebAppkind !=
                            functionAppContainer && webAppKind != functionAppLinux && WebAppKind != ""
                            && Package NotEndsWith .war && Package NotEndsWith .jar. Deployment method.
                            Default: webDeploy.

                                #TakeAppOfflineFlag: true # boolean. Optional. Use when UseWebDeploy ==
                                true && DeploymentType != runFromZip && ConnectionType = AzureRM &&
                                WebAppKind != webAppLinux && WebAppKind != webAppContainer && WebAppkind !=
                                functionAppContainer && webAppKind != functionAppLinux && WebAppKind != ""
                                && Package NotEndsWith .war && Package NotEndsWith .jar. Take App Offline.
                                Default: true.

                                    #SetParametersFile: # string. Optional. Use when UseWebDeploy == true &&
                                    DeploymentType == webDeploy && ConnectionType = AzureRM && WebAppKind !=
                                    webAppLinux && WebAppKind != webAppContainer && WebAppkind !=
                                    functionAppContainer && webAppKind != functionAppLinux && WebAppKind != ""
                                    && Package NotEndsWith .war && Package NotEndsWith .jar. SetParameters file.

                                        #RemoveAdditionalFilesFlag: false # boolean. Optional. Use when
                                        UseWebDeploy == true && DeploymentType == webDeploy && ConnectionType =
                                        AzureRM && WebAppKind != webAppLinux && WebAppKind != webAppContainer &&
                                        WebAppkind != functionAppContainer && webAppKind != functionAppLinux &&
                                        WebAppKind != "" && Package NotEndsWith .war && Package NotEndsWith .jar.
                                        Remove additional files at destination. Default: false.

                                            #ExcludeFilesFromAppDataFlag: true # boolean. Optional. Use when
                                            UseWebDeploy == true && DeploymentType == webDeploy && ConnectionType =
                                            AzureRM && WebAppKind != webAppLinux && WebAppKind != webAppContainer &&
                                            WebAppkind != functionAppContainer && webAppKind != functionAppLinux &&
                                            WebAppKind != "" && Package NotEndsWith .war && Package NotEndsWith .jar.
                                            Exclude files from the App_Data folder. Default: true.

                                                #AdditionalArguments: '-retryAttempts:6 -retryInterval:10000' # string.
                                                Optional. Use when UseWebDeploy == true && DeploymentType == webDeploy &&
                                                ConnectionType = AzureRM && WebAppKind != webAppLinux && WebAppKind !=
                                                webAppContainer && WebAppkind != functionAppContainer && webAppKind !=
                                                functionAppLinux && WebAppKind != "" && Package NotEndsWith .war && Package
                                                NotEndsWith .jar. Additional arguments. Default: -retryAttempts:6 -
                                                retryInterval:10000.

                                                    #RenameFilesFlag: true # boolean. Optional. Use when UseWebDeploy ==
                                                    true && DeploymentType == webDeploy && ConnectionType = AzureRM &&
                                                    WebAppKind != webAppLinux && WebAppKind != webAppContainer && WebAppkind !=
                                                    functionAppContainer && webAppKind != functionAppLinux && WebAppKind != ""

```

```
&& Package NotEndsWith .war && Package NotEndsWith .jar. Rename locked  
files. Default: true.
```

Inputs

`ConnectionType` - Connection type

`string`. Required. Allowed values: `AzureRM` (Azure Resource Manager), `PublishProfile` (Publish Profile). Default value: `AzureRM`.

Specify the service connection type to use to deploy the Web App.

Specify `Publish Profile` for using Visual Studio created [Publish profiles](#).

`azureSubscription` - Azure subscription

Input alias: `ConnectedServiceName`. `string`. Required when `ConnectionType = AzureRM`.

Specify the Azure Resource Manager subscription for the deployment.

`PublishProfilePath` - Publish profile path

`string`. Required when `ConnectionType = PublishProfile`. Default value: `$(System.DefaultWorkingDirectory)/**/*.pubxml`.

The path of the [publish profile](#) created from Visual Studio.

`PublishProfilePassword` - Publish profile password

`string`. Required when `ConnectionType = PublishProfile`.

It is recommended to store a password in a secret variable and use that variable here e.g. `$(Password)`.

`appType` - App Service type

Input alias: `WebAppKind`. `string`. Required when `ConnectionType = AzureRM`. Allowed values: `webApp` (Web App on Windows), `webAppLinux` (Web App on Linux), `webAppContainer` (Web App for Containers (Linux)), `webAppHyperVContainer` (Web App for Containers (Windows)), `functionApp` (Function App on Windows (Not Recommended, Use Azure Functions Task)), `functionAppLinux` (Function App on Linux (Not Recommended, Use Azure Functions Task)), `functionAppContainer` (Function App for

Containers (Linux) (Not Recommended, Use Azure Functions for container Task)), `apiApp` (API App), `mobileApp` (Mobile App). Default value: `webApp`.

Choose from Web App On Windows, Web App On Linux, Web App for Containers, Function App, Function App on Linux, Function App for Containers and Mobile App.

`WebAppName` - App Service name

`string`. Required when `ConnectionType = AzureRM`.

Specify the name of an existing Azure App Service. App services based on the selected app type will only be listed when using the task assistant.

`deployToSlotOrASE` - Deploy to Slot or App Service Environment

Input alias: `DeployToSlotOrASEFlag`. `boolean`. Optional. Use when `ConnectionType = AzureRM && WebAppKind != ""`. Default value: `false`.

Specify the option to deploy to an existing deployment slot or Azure App Service environment. For both targets, the task requires a Resource Group name. If the deployment target is a slot, by default the deployment is to the **Production** slot. Any other existing slot name can be provided. If the deployment target is an Azure App Service environment, leave the slot name as `Production` and specify just the Resource Group name.

`ResourceGroupName` - Resource group

`string`. Required when `DeployToSlotOrASEFlag = true`.

The Resource group name is required when the deployment target is either a deployment slot or an App Service Environment.

Specify the Azure Resource group that contains the Azure App Service specified above.

`SlotName` - Slot

`string`. Required when `DeployToSlotOrASEFlag = true`. Default value: `production`.

Specify an existing slot other than the Production slot.

`DockerNamespace` - Registry or Namespace

`string`. Required when `WebAppKind = webAppContainer || WebAppkind =`

```
functionAppContainer || WebAppkind = webAppHyperVContainer.
```

A globally unique top-level domain name for your specific registry or namespace. Note: The fully qualified image name will be of the format: {registry or namespace}/{repository}:{tag}. For example, myregistry.azurecr.io/nginx:latest.

DockerRepository - Image

```
string. Required when WebAppKind = webAppContainer || WebAppkind =  
functionAppContainer || WebAppkind = webAppHyperVContainer.
```

The name of the repository where the container images are stored. Note: The fully qualified image name will be of the format: {registry or namespace}/{repository}:{tag}. For example, myregistry.azurecr.io/nginx:latest.

DockerImageTag - Tag

```
string. Optional. Use when WebAppKind = webAppContainer || WebAppkind =  
functionAppContainer || WebAppkind = webAppHyperVContainer.
```

Tags are the mechanism that registries use to apply version information to Docker images. Note: The fully qualified image name will be of the format: {registry or namespace}/{repository}:{tag}. For example, myregistry.azurecr.io/nginx:latest.

VirtualApplication - Virtual application

```
string. Optional. Use when WebAppKind != webAppLinux && WebAppKind !=  
webAppContainer && WebAppkind != functionAppContainer && WebAppKind != functionApp  
&& webAppKind != functionAppLinux && WebAppKind != "".
```

Specify the name of the Virtual Application that has been configured in the Azure portal. This option is not required for deployments to the website root. The Virtual Application must have been configured before deployment of the web project.

packageForLinux - Package or folder

Input alias: `Package`. `string`. Required when `ConnectionType = PublishProfile` || `WebAppKind = webApp` || `WebAppKind = apiApp` || `WebAppKind = functionApp` || `WebAppKind = mobileApp` || `WebAppKind = webAppLinux` || `webAppKind = functionAppLinux`. Default value: `$(System.DefaultWorkingDirectory)/**/*.zip`.

The file path to the package, or to a folder containing app service contents generated by MSBuild or a compressed zip or war file.

Variables are **Build** and **Release**. **Wildcards** are supported.

For example, `$(System.DefaultWorkingDirectory)/*/*/*.zip` or
`$(System.DefaultWorkingDirectory)/*/*/*.war`.

RuntimeStack - Runtime Stack

`string`. Optional. Use when `WebAppKind = webAppLinux`.

Specify the framework and version for Function App on Linux.

RuntimeStackFunction - Runtime Stack

`string`. Optional. Use when `WebAppKind = functionAppLinux`. Allowed values: `DOTNET|2.2` (`DOTNET|2.2` (functionapp v2)), `DOTNET|3.1` (`DOTNET|3.1` (functionapp v3)), `JAVA|8` (`JAVA|8` (functionapp v2/v3)), `JAVA|11` (`JAVA|11` (functionapp v3)), `NODE|8` (`NODE|8` (functionapp v2)), `NODE|10` (`NODE|10` (functionapp v2/v3)), `NODE|12` (`NODE|12` (functionapp v3)), `NODE|14` (`NODE|14` (functionapp v3)), `PYTHON|3.6` (`PYTHON|3.6` (functionapp v2/v3)), `PYTHON|3.7` (`PYTHON|3.7` (functionapp v2/v3)), `PYTHON|3.8` (`PYTHON|3.8` (functionapp v3)).

Specify the framework and version. Refer to the [Azure Functions runtime versions overview](#) for supported runtime versions. Old values like `DOCKER|microsoft/azure-functions-*` are deprecated. Please use the new values from dropdown.

StartupCommand - Startup command

`string`. Optional. Use when `WebAppKind = webAppLinux || WebAppKind = webAppContainer || WebAppkind = functionAppContainer || WebAppKind = functionAppLinux || WebAppkind = webAppHyperVContainer`.

Specify the Startup command. For example:

`dotnet exec filename.dll`

`dotnet filename.dll`.

ScriptType - Deployment script type

`string`. Optional. Use when `ConnectionType = AzureRM && WebAppKind != "" &&`

`WebAppKind != webAppContainer && WebAppkind != functionAppContainer`. Allowed values: `Inline Script`, `File Path` (Script File Path).

Customizes the deployment by providing a script that runs on the Azure App Service after successful deployment. Choose inline deployment script or the path and name of a script file. Learn more about [Azure App Service Deployment](#).

`InlineScript` - Inline Script

`string`. Required when `ScriptType == Inline Script && ConnectionType = AzureRM && WebAppKind != "" && WebAppKind != webAppContainer && WebAppkind != functionAppContainer`. Default value: `:: You can provide your deployment commands here. One command per line..`

The script to execute. You can provide your deployment commands here, one command per line. See the following example.

`ScriptPath` - Deployment script path

`string`. Required when `ScriptType == File Path && ConnectionType = AzureRM && WebAppKind != "" && WebAppKind != webAppContainer && WebAppkind != functionAppContainer`.

The path and name of the script to execute.

`WebConfigParameters` - Generate web.config parameters for Python, Node.js, Go and Java apps

`string`. Optional. Use when `WebAppKind != webAppContainer && WebAppkind != functionAppContainer && WebAppKind != webAppLinux && webAppKind != functionAppLinux && Package NotEndsWith .war`.

A standard `Web.config` will be generated and deployed to Azure App Service if the application does not have one. The values in `web.config` can be edited and vary based on the application framework. For example, for `node.js` applications, `web.config` will have a `Startup` file and `iis_node` module values. This edit feature is only for the generated `web.config`. Learn more about [Azure App Service Deployment](#).

`AppSettings` - App settings

`string`. Optional. Use when `ConnectionType = AzureRM`.

Edits web app application settings using the syntax `-key value`. Values containing spaces must be enclosed in double quotes. Examples: `-Port 5000 -RequestTimeout 5000` and `-WEBSITE_TIME_ZONE "Eastern Standard Time"`. To provide two or more key values, the key values must be separated by a space. Example: `-key1 "Value1" -Key2 "Value2"`.

ConfigurationSettings - Configuration settings

`string`. Optional. Use when `ConnectionType = AzureRM`.

Edits web app configuration settings using the syntax `-key value`. Values containing spaces must be enclosed in double quotes. Example: `-phpVersion 5.6 -linuxFxVersion node|6.11`.

enableCustomDeployment - Select deployment method

Input alias: `UseWebDeploy`. `boolean`. Optional. Use when `ConnectionType = AzureRM && WebAppKind != webAppLinux && WebAppKind != webAppContainer && WebAppkind != functionAppContainer && webAppKind != functionAppLinux && WebAppKind != "" && Package NotEndsWith .war && Package NotEndsWith .jar`. Default value: `false`.

If unchecked or false, the task auto-detects the best deployment method based on the app type, package format, and other parameters. Check this option in the task assistant to view the supported deployment methods, and choose one for deploying your app.

DeploymentType - Deployment method

`string`. Required when `UseWebDeploy == true && ConnectionType = AzureRM && WebAppKind != webAppLinux && WebAppKind != webAppContainer && WebAppkind != functionAppContainer && webAppKind != functionAppLinux && WebAppKind != "" && Package NotEndsWith .war && Package NotEndsWith .jar`. Allowed values: `webDeploy` (Web Deploy), `zipDeploy` (Zip Deploy), `runFromZip` (Run From Package). Default value: `webDeploy`.

Determines the deployment method for the app.

TakeAppOfflineFlag - Take App Offline

`boolean`. Optional. Use when `UseWebDeploy == true && DeploymentType != runFromZip && ConnectionType = AzureRM && WebAppKind != webAppLinux && WebAppKind != webAppContainer && WebAppkind != functionAppContainer && webAppKind !=`

```
functionAppLinux && WebAppKind != "" && Package NotEndsWith .war && Package  
NotEndsWith .jar. Default value: true.
```

Specify this option to take the Azure App Service offline by placing an `app_offline.htm` file in the root directory before the synchronization operation begins. The file will be removed after the synchronization completes successfully.

SetParametersFile - SetParameters file

```
string. Optional. Use when UseWebDeploy == true && DeploymentType == webDeploy &&  
ConnectionType = AzureRM && WebAppKind != webAppLinux && WebAppKind !=  
webAppContainer && WebAppkind != functionAppContainer && webAppKind !=  
functionAppLinux && WebAppKind != "" && Package NotEndsWith .war && Package  
NotEndsWith .jar.
```

The location of the `SetParameters.xml` file to use.

RemoveAdditionalFilesFlag - Remove additional files at destination

```
boolean. Optional. Use when UseWebDeploy == true && DeploymentType == webDeploy &&  
ConnectionType = AzureRM && WebAppKind != webAppLinux && WebAppKind !=  
webAppContainer && WebAppkind != functionAppContainer && webAppKind !=  
functionAppLinux && WebAppKind != "" && Package NotEndsWith .war && Package  
NotEndsWith .jar. Default value: false.
```

Specify 'true' to delete files on the Azure App Service that have no matching files in the App Service package or folder. This will also remove all files related to any extension installed on this Azure App Service. To prevent this, select the `Exclude files from App_Data folder` checkbox.

ExcludeFilesFromAppDataFlag - Exclude files from the App_Data folder

```
boolean. Optional. Use when UseWebDeploy == true && DeploymentType == webDeploy &&  
ConnectionType = AzureRM && WebAppKind != webAppLinux && WebAppKind !=  
webAppContainer && WebAppkind != functionAppContainer && webAppKind !=  
functionAppLinux && WebAppKind != "" && Package NotEndsWith .war && Package  
NotEndsWith .jar. Default value: true.
```

Specify the option to prevent files in the `App_Data` folder from being deployed to/deleted from the Azure App Service.

AdditionalArguments - Additional arguments

```
string. Optional. Use when UseWebDeploy == true && DeploymentType == webDeploy &&
ConnectionType = AzureRM && WebAppKind != webAppLinux && WebAppKind != webAppContainer && WebAppkind != functionAppContainer && webAppKind != functionAppLinux && WebAppKind != "" && Package NotEndsWith .war && Package NotEndsWith .jar. Default value: -retryAttempts:6 -retryInterval:10000.
```

Additional Web Deploy arguments following the syntax `-key:value`. These will be applied when deploying the Azure App Service. Examples:

```
disableLink:AppPoolExtension -disableLink:ContentExtension. Learn more about Web Deploy Operation Settings.
```

RenameFilesFlag - Rename locked files

```
boolean. Optional. Use when UseWebDeploy == true && DeploymentType == webDeploy && ConnectionType = AzureRM && WebAppKind != webAppLinux && WebAppKind != webAppContainer && WebAppkind != functionAppContainer && webAppKind != functionAppLinux && WebAppKind != "" && Package NotEndsWith .war && Package NotEndsWith .jar. Default value: true.
```

Specify the default value to enable the msdeploy flag `MSDEPLOY_RENAME_LOCKED_FILES=1` in Azure App Service application settings. If set, the option enables msdeploy to rename files that are locked during app deployment.

enableXmlTransform - XML transformation

Input alias: `XmlTransformation`. `boolean`. Optional. Use when `WebAppKind != webAppContainer && WebAppkind != functionAppContainer && WebAppKind != webAppLinux && webAppKind != functionAppLinux && Package NotEndsWith .war`. Default value: `false`.

The config transforms will be run for `*.Release.config` and `*.<EnvironmentName>.config` on the `*.config` file. Configuration transformations run before variable substitution. [XML transformations](#) are supported only for the Windows platform.

enableXmlVariableSubstitution - XML variable substitution

Input alias: `XmlVariableSubstitution`. `boolean`. Optional. Use when `WebAppKind != webAppContainer && WebAppkind != functionAppContainer && WebAppKind != webAppLinux && webAppKind != functionAppLinux && Package NotEndsWith .war`. Default value: `false`.

Variables defined in the build or release pipeline will be matched against the key or name entries in the `configSections`, `appSettings`, `applicationSettings`, and `connectionStrings` sections of any configuration file and `parameters.xml` file. Variable substitution runs after configuration transformations.

If the same variables are defined in the release pipeline and in the stage, the stage variables will supersede the release pipeline variables. Learn more about [XML variable substitution]](/azure/devops/pipelines/tasks/transforms-variable-substitution#xml-variable-substitution).

`JSONFiles` - JSON variable substitution

`string`. Optional. Use when `WebAppKind != webAppContainer && WebAppkind != functionAppContainer && WebAppKind != webAppLinux && webAppKind != functionAppLinux && Package NotEndsWith .war.`

Provides a newline-separated list of JSON files to substitute the variable values. File names must be relative to the root folder. To substitute JSON variables that are nested or hierarchical, specify them using `JSONPath` expressions. For example, to replace the value of `ConnectionString` in the sample below, define a variable named `Data.DefaultConnection.ConnectionString` in the build or release pipeline (or release pipelines stage).

```
JSON

{
  "Data": {
    "DefaultConnection": {
      "ConnectionString": "Server=(localdb)\SQLEXPRESS;Database=MyDB;Trusted_Connection=True"
    }
  }
}
```

A variable substitution runs after configuration transformations. Note: Build and release pipeline variables are excluded from substitution. Learn more about [JSON variable substitution](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

`AppServiceApplicationUrl`

Application URL of the selected App Service.

Remarks

- [Prerequisites for the task](#)
- [Usage notes](#)
- [Deployment methods](#)
- [Troubleshooting](#)
- [FAQs](#)

Use this task to deploy to a range of App Services on Azure. The task works on cross-platform agents running Windows, Linux, or Mac and uses several different [underlying deployment technologies](#).

The task works for [ASP.NET](#), [ASP.NET Core](#), [PHP](#), [Java](#), [Python](#), [Go](#), and [Node.js](#) based web applications.

The task can be used to deploy to a range of Azure App Services such as:

- [Web Apps on both Windows and Linux](#)
- [Web Apps for Containers](#)
- [Function Apps on both Windows and Linux](#)
- [Function Apps for Containers](#)
- [WebJobs](#)
- Apps configured under [Azure App Service Environments](#)

Prerequisites for the task

The following prerequisites must be set up in the target machine(s) for the task to work correctly.

- **App Service instance.** The task is used to deploy a Web App project or Azure Function project to an existing Azure App Service instance, which must exist before the task runs. The App Service instance can be created from the [Azure portal](#) and [configured](#) there. Alternatively, the [Azure PowerShell task](#) can be used to run [AzureRM PowerShell scripts](#) to provision and configure the Web App.

- **Azure Subscription.** To deploy to Azure, an Azure subscription must be [linked to the pipeline](#). The task does not work with the Azure Classic service connection, and it will not list these connections in the settings of the task.

Usage notes

- The task works with the [Azure Resource Manager APIs](#) only.
- To ignore SSL errors, define a variable named `VSTS_ARM_REST_IGNORE_SSL_ERRORS` with value `true` in the pipeline. If you are deploying to a slot configured to auto-swap, the swap will fail unless you set `SCM_SKIP_SSL_VALIDATION` or `'SCM_SKIP_ASE_SSL_VALIDATION'` to `1` in the app services configuration settings.
- For .NET apps targeting Web App on Windows, avoid deployment failure with the error `ERROR_FILE_IN_USE` by ensuring that **Rename locked files** and **Take App Offline** settings are enabled. For zero downtime deployment, use the slot swap option.
- When deploying to an App Service that has Application Insights configured, and you have enabled **Remove additional files at destination**, ensure you also enable **Exclude files from the App_Data folder** in order to maintain the Application insights extension in a safe state. This is required because the Application Insights continuous web job is installed into the App_Data folder.

Deployment methods

Several deployment methods are available in this task. Web Deploy (msdeploy.exe) is the default. To change the deployment option, expand **Additional Deployment Options** and enable **Select deployment method** to choose from additional package-based deployment options.

Based on the type of Azure App Service and agent, the task chooses a suitable deployment technology. The different deployment technologies used by the task are:

- [Web Deploy](#)
- [Kudu REST APIs](#)
- [Container Registry](#)
- [Zip Deploy](#)
- [Run From Package](#)
- [War Deploy](#)

By default, the task tries to select the appropriate deployment technology based on the input package type, App Service type, and agent operating system.

Auto Detect Logic

For windows based agents.

App Service type	Package type	Deployment Method
WebApp on Linux or Function App on Linux	Folder/Zip/jar War	Zip Deploy War Deploy
WebApp for Containers (Linux) or Function App for Containers (Linux)	Update the App settings	NA
WebApp on Windows, Function App on Windows, API App, or Mobile App	War Jar MsBuild package type or deploy to virtual application Folder/Zip	War Deploy Zip Deploy Web Deploy <pre>if postDeploymentScript == true, Zip Deploy else, Run From Package</pre>

On non-Windows agents (for any App Service type), the task relies on [Kudu REST APIs](#) to deploy the app.

Web Deploy

[Web Deploy](#) (msdeploy.exe) can be used to deploy a Web App on Windows or a Function App to the Azure App Service using a Windows agent. Web Deploy is feature-rich and offers options such as:

- **Rename locked files:** Rename any file that is still in use by the web server by enabling the msdeploy flag `MSDEPLOY_RENAME_LOCKED_FILES=1` in the Azure App Service settings. This option, if set, enables msdeploy to rename files that are locked during app deployment.
- **Remove additional files at destination:** Deletes files in the Azure App Service that have no matching files in the App Service artifact package or folder being deployed.
- **Exclude files from the App_Data folder:** Prevent files in the App_Data folder (in the artifact package/folder being deployed) being deployed to the Azure App Service
- **Additional Web Deploy arguments:** Arguments that will be applied when deploying the Azure App Service. Example: `-disableLink:AppPoolExtension -`

`disableLink:ContentExtension`. For more examples of Web Deploy operation settings, see [Web Deploy Operation Settings](#).

Install Web Deploy on the agent using the [Microsoft Web Platform Installer](#). Web Deploy 3.5 must be installed without the bundled SQL support. There is no need to choose any custom settings when installing Web Deploy. Web Deploy is installed at `C:/Program Files (x86)/IIS/Microsoft Web Deploy V3`.

Kudu REST APIs

[Kudu REST APIs](#) work on both Windows and Linux automation agents when the target is a Web App on Windows, Web App on Linux (built-in source), or Function App. The task uses Kudu to copy files to the Azure App service.

Container Registry

Works on both Windows and Linux automation agents when the target is a Web App for Containers. The task updates the app by setting the appropriate container registry, repository, image name, and tag information. You can also use the task to pass a startup command for the container image.

Zip Deploy

Expects a .zip deployment package and deploys the file contents to the **wwwroot** folder of the App Service or Function App in Azure. This option overwrites all existing contents in the **wwwroot** folder. For more information, see [Zip deployment for Azure Functions](#).

Run From Package

Expects the same deployment package as Zip Deploy. However, instead of deploying files to the **wwwroot** folder, the entire package is mounted by the Functions runtime and files in the **wwwroot** folder become read-only. For more information, see [Run your Azure Functions from a package file](#).

War Deploy

Expects a .war deployment package and deploys the file content to the **wwwroot** folder or **webapps** folder of the App Service in Azure.

Troubleshooting

Error: Could not fetch access token for Azure. Verify if the Service Principal used is valid and not expired.

The task uses the service principal in the service connection to authenticate with Azure. If the service principal has expired or doesn't have permissions to the App Service, the task fails with this error. Verify the validity of the service principal used and that it's present in the app registration. For more information, see [Use role-based access control to manage access to your Azure subscription resources](#). This blog post  also contains more information about using service principal authentication.

SSL error

If you want to use a certificate in App Service, the certificate must be signed by a trusted certificate authority. If your web app gives you certificate validation errors, you're probably using a self-signed certificate. Set a variable named

`VSTS_ARM_REST_IGNORE_SSL_ERRORS` to the value `true` in the build or release pipeline to resolve the error.

A release hangs for long time and then fails

This problem could be the result of insufficient capacity in your App Service plan. To resolve this problem, you can scale up the App Service instance to increase available CPU, RAM, and disk space or try with a different App Service plan.

5xx error codes

If you're seeing a 5xx error, [check the status of your Azure service](#) .

Azure Function suddenly stopped working

Azure Functions may suddenly stop working if more than one year has passed since the last deployment. If you deploy with "RunFromPackage" in "deploymentMethod", a SAS with an expiration date of 1 year is generated and set as the value of "WEBSITE_RUN_FROM_PACKAGE" in the application configuration. Azure Functions uses this SAS to reference the package file for function execution, so if the SAS has expired, the function will not be executed. To resolve this issue, deploy again to generate a SAS with an expiration date of one year.

Error: No package found with specified pattern

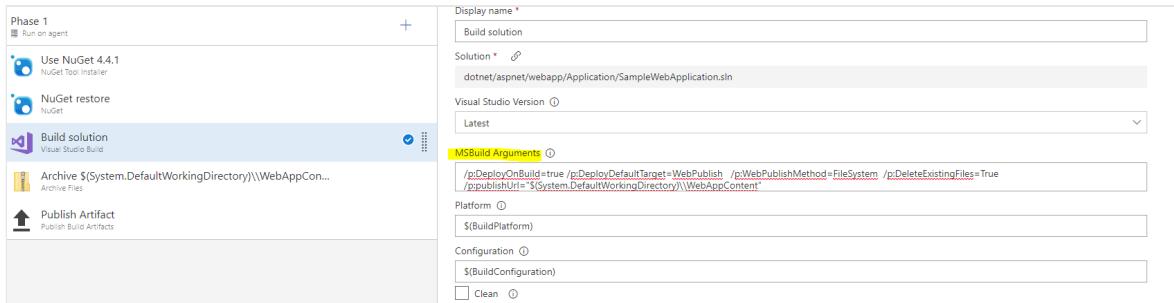
Check if the package mentioned in the task is published as an artifact in the build or a previous stage and downloaded in the current job.

Error: Publish using zip deploy option is not supported for msBuild package type

Web packages created via the MSBuild task (with default arguments) have a nested folder structure that can be deployed correctly only by Web Deploy. The publish-to-zip deployment option can't be used to deploy those packages. To convert the packaging structure, take these steps:

1. In the Build solution task, change the **MSBuild Arguments** to

```
/p:DeployOnBuild=true /p:DeployDefaultTarget=WebPublish  
/p:WebPublishMethod=FileSystem /p:DeleteExistingFiles=True  
/p:publishUrl="$(System.DefaultWorkingDirectory)\\WebAppContent":
```

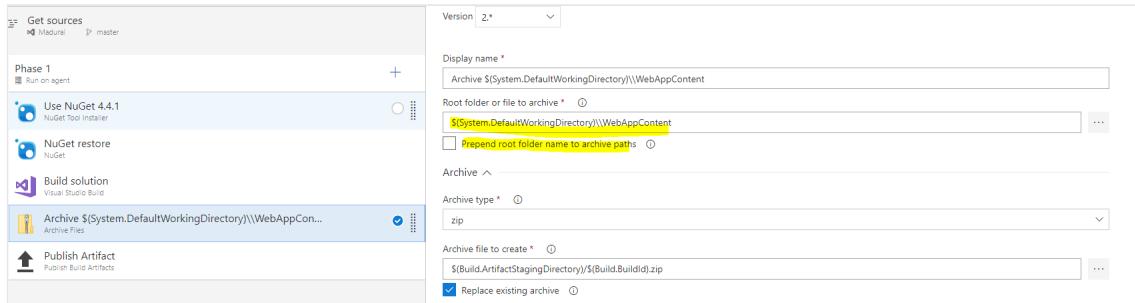


2. Add an Archive task and change the values as follows:

- a. Change **Root folder or file to archive** to

```
$(System.DefaultWorkingDirectory)\\WebAppContent.
```

- b. Clear the **Prepend root folder name to archive paths** check box:

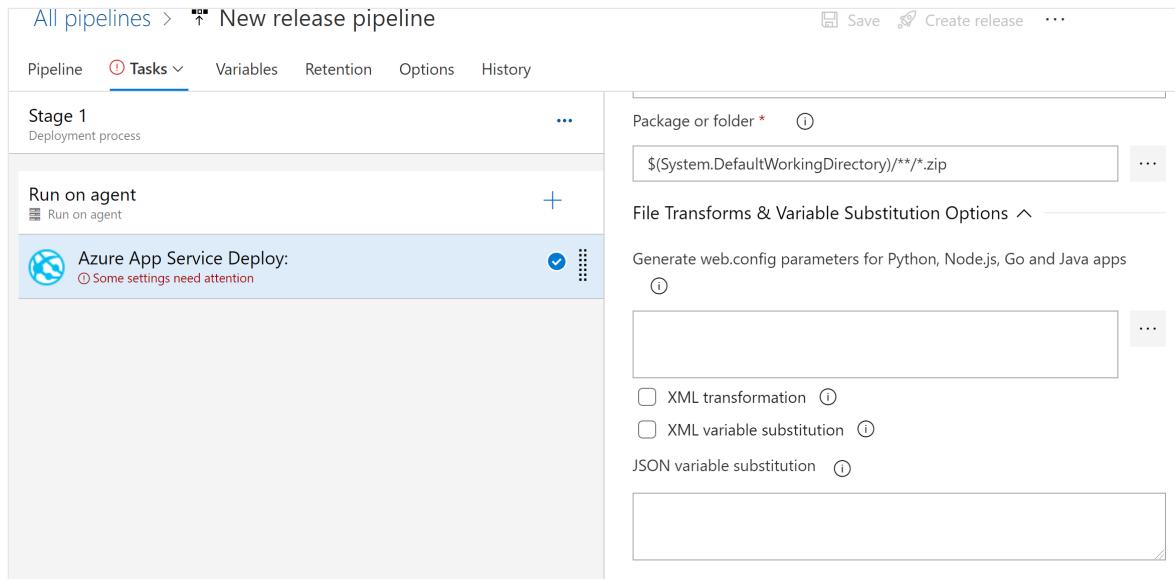


Web app deployment on Windows is successful but the app is not working

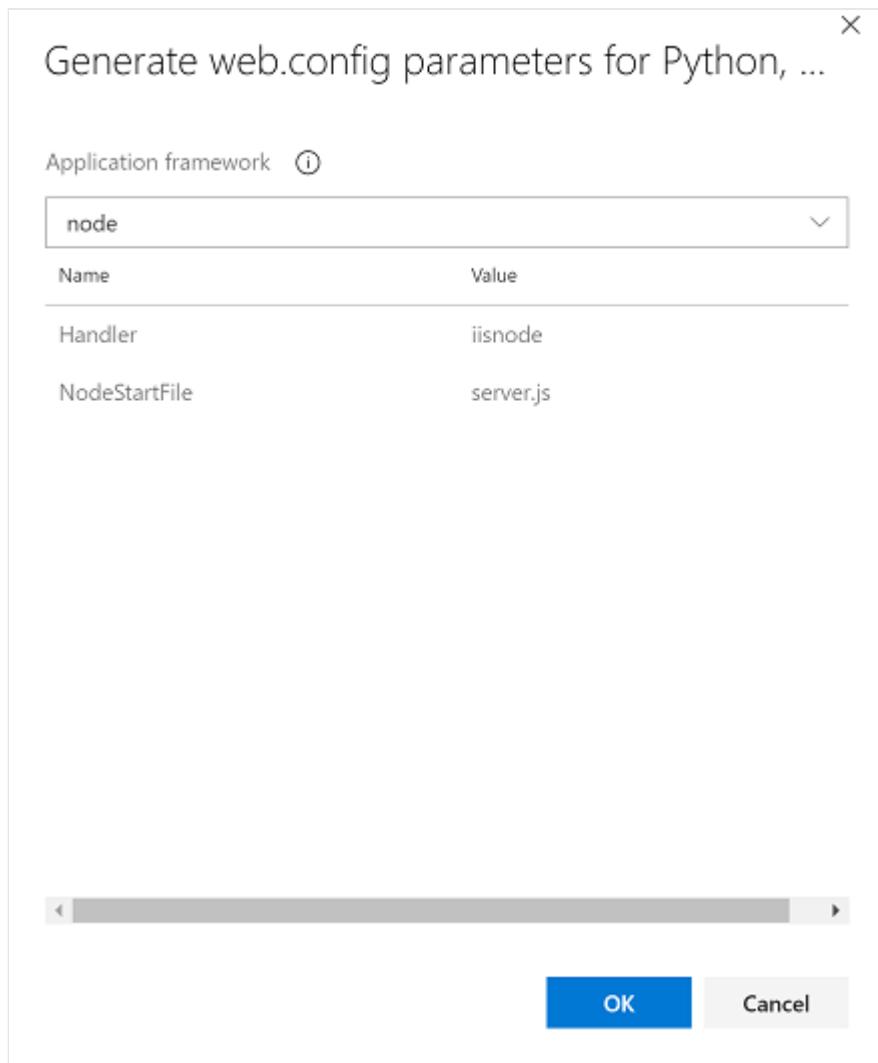
This may be because web.config is not present in your app. You can either add a web.config file to your source or auto-generate one using the File Transforms and

Variable Substitution Options of the task.

- Click on the task and go to Generate web.config parameters for Python, Node.js, Go and Java apps.



- Click on the more button Generate web.config parameters for Python, Node.js, Go and Java apps to edit the parameters.



- Select your application type from the drop down.
- Click on OK. This will populate web.config parameters required to generate web.config.

ⓘ Note

This section is deprecated and has been replaced with the [File Transform task](#).

ERROR_FILE_IN_USE

When deploying .NET apps to Web App on Windows, deployment may fail with error code *ERROR_FILE_IN_USE*. To resolve the error, ensure *Rename locked files* and *Take App Offline* options are enabled in the task. For zero downtime deployments, use slot swap.

You can also use *Run From Package* deployment method to avoid resource locking.

Web Deploy Error

If you are using web deploy to deploy your app, in some error scenarios Web Deploy will show an error code in the log. To troubleshoot a web deploy error, see [Web Deploy error codes](#).

Web app deployment on App Service Environment (ASE) is not working

- Ensure that the Azure DevOps build agent is on the same VNET (subnet can be different) as the Internal Load Balancer (ILB) of ASE. This will enable the agent to pull code from Azure DevOps and deploy to ASE.
- If you are using Azure DevOps, the agent doesn't need to be accessible from the internet but needs only outbound access to connect to Azure DevOps Service.
- If you are using TFS/Azure DevOps Server deployed in a Virtual Network, the agent can be completely isolated.
- The build agent must be configured with the DNS configuration of the Web App it needs to deploy to. Since the private resources in the Virtual Network don't have entries in Azure DNS, this needs to be added to the hosts file on the agent machine.
- If a self-signed certificate is used for the ASE configuration, the `-allowUntrusted` option needs to be set in the deploy task for MSDeploy. It is also recommended to set the variable `VSTS_ARM_REST_IGNORE_SSL_ERRORS` to true. If a certificate from a

certificate authority is used for ASE configuration, this should not be necessary. If you are deploying to a slot configured to auto-swap, the swap will fail unless you set `SCM_SKIP_SSL_VALIDATION` or `SCM_SKIP_ASE_SSL_VALIDATION` to `1` in the app services configuration settings.

FAQs

How should I configure my service connection?

This task requires an [Azure Resource Manager service connection](#).

How should I configure web job deployment with Application Insights?

When you're deploying to an App Service, if you have [Application Insights](#) configured and you've enabled `Remove additional files at destination`, you also need to enable `Exclude files from the App_Data folder`. Enabling this option keeps the Application Insights extension in a safe state. This step is required because the Application Insights continuous WebJob is installed into the App_Data folder.

How should I configure my agent if it's behind a proxy while I'm deploying to App Service?

If your self-hosted agent requires a web proxy, you can inform the agent about the proxy during configuration. Doing so allows your agent to connect to Azure Pipelines or Azure DevOps Server through the proxy. [Learn more about running a self-hosted agent behind a web proxy](#).

Examples

- [Deploy to specific app type](#)
- [Deploy to Azure Web App Linux container](#)
- [Deploy a web app to a Windows App Service across deployment slots](#)
- [Sample Post deployment script](#)

Deploy to specific app type

To deploy to a specific app type, set `appType` to any of the following accepted values:

`webApp` (Web App on Windows), `webAppLinux` (Web App on Linux), `webAppContainer`

(Web App for Containers - Linux), `functionApp` (Function App on Windows), `functionAppLinux` (Function App on Linux), `functionAppContainer` (Function App for Containers - Linux), `apiApp` (API App), `mobileApp` (Mobile App). If not mentioned, `webApp` is taken as the default value.

To enable any advanced deployment options, add the parameter `enableCustomDeployment: true` and include the following parameters as needed.

yml

```
# deploymentMethod: 'runFromPackage' # supports zipDeploy as well
# appOffline: boolean    # Not applicable for 'runFromPackage'
# setParametersFile: string
# removeAdditionalFilesFlag: boolean
# additionalArguments: string
```

Deploy to Azure Web App Linux container

The following YAML example deploys to an Azure Web App container (Linux).

yml

```
pool:
  vmImage: ubuntu-latest

variables:
  azureSubscriptionEndpoint: Contoso
  DockerNamespace: contoso.azurecr.io
  DockerRepository: aspnetcore
  WebAppName: containersdemoapp

steps:
  - task: AzureRMWebAppDeployment@4
    displayName: Azure App Service Deploy
    inputs:
      appType: webAppContainer
      ConnectedServiceName: $(azureSubscriptionEndpoint)
      WebAppName: $(WebAppName)
      DockerNamespace: $(DockerNamespace)
      DockerRepository: $(DockerRepository)
      DockerImageTag: $(Build.BuildId)
```

Deploy a web app to a Windows App Service across deployment slots

The following example deploys a web app to a Windows App Service across deployment slots.

```
yml

pool:
  vmImage: 'windows-latest'

variables:
  solution: '**/*.sln'
  buildPlatform: 'Any CPU'
  buildConfiguration: 'Release'

stages:
- stage: DeployDevStage
  displayName: 'Deploy App to Dev Slot'
  jobs:
    - job: DeployApp
      displayName: 'Deploy App'
      steps:
        - task: DownloadPipelineArtifact@2
          inputs:
            buildType: 'current'
            artifactName: 'drop'
            targetPath: '$(System.DefaultWorkingDirectory)'
        - task: AzureRmWebAppDeployment@4
          inputs:
            ConnectionType: 'AzureRM'
            azureSubscription: 'Fabrikam Azure Subscription - PartsUnlimited'
            appType: 'webApp'
            WebAppName: 'partsunlimited'
            deployToSlotOrASE: true
            ResourceGroupName: 'rgPartsUnlimited'
            SlotName: 'Dev'
            packageForLinux: '$(System.DefaultWorkingDirectory)/**/*.zip'

- stage: DeployStagingStage
  displayName: 'Deploy App to Staging Slot'
  dependsOn: DeployDevStage
  jobs:
    - job: DeployApp
      displayName: 'Deploy App'
      steps:
        - task: DownloadPipelineArtifact@2
          inputs:
            buildType: 'current'
            artifactName: 'drop'
            targetPath: '$(System.DefaultWorkingDirectory)'
        - task: AzureRmWebAppDeployment@4
          inputs:
            appType: webApp
            ConnectionType: AzureRM
            ConnectedServiceName: 'Fabrikam Azure Subscription -
```

```
PartsUnlimited'
    ResourceGroupName: 'rgPartsUnlimited'
    WebAppName: 'partsunlimited'
    Package: '$(System.DefaultWorkingDirectory)/**/*.zip'
    deployToSlotOrASE: true
    SlotName: 'staging'
```

Sample Post deployment script

The task provides an option to customize the deployment by providing a script that will run on the Azure App Service after the app's artifacts have been successfully copied to the App Service. You can choose to provide either an inline deployment script or the path and name of a script file in your artifact folder.

This is very useful when you want to restore your application dependencies directly on the App Service. Restoring packages for Node, PHP, and python apps helps to avoid timeouts when the application dependency results in a large artifact being copied over from the agent to the Azure App Service.

An example of a deployment script is:

```
ps

@echo off
if NOT exist requirements.txt (
    echo No Requirements.txt found.
    EXIT /b 0
)
if NOT exist "$(PYTHON_EXT)/python.exe" (
    echo Python extension not available >&2
    EXIT /b 1
)
echo Installing dependencies
call "$(PYTHON_EXT)/python.exe" -m pip install -U setuptools
if %errorlevel% NEQ 0 (
    echo Failed to install setuptools >&2
    EXIT /b 1
)
call "$(PYTHON_EXT)/python.exe" -m pip install -r requirements.txt
if %errorlevel% NEQ 0 (
    echo Failed to install dependencies>&2
    EXIT /b 1
)
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.104.1 or greater
Task category	Deploy

See also

- This task is open source [on GitHub](#). Feedback and contributions are welcome.

AzureRmWebAppDeployment@3 - Azure App Service deploy v3 task

Article • 09/26/2023

Use this task to deploy to Azure App Service a web, mobile, or API app using Docker, Java, .NET, .NET Core, Node.js, PHP, Python, or Ruby.

ⓘ Note

Use [AzureFunctionApp@1](#) to deploy Azure Functions apps.

Syntax

YAML

```
# Azure App Service deploy v3
# Deploy to Azure App Service a web, mobile, or API app using Docker, Java,
.NET, .NET Core, Node.js, PHP, Python, or Ruby.
- task: AzureRmWebAppDeployment@3
  inputs:
    azureSubscription: # string. Alias: ConnectedServiceName. Required.
    Azure subscription.
    appType: 'app' # 'app' | 'applinux' | 'functionapp' | 'api' |
'mobileapp'. Alias: WebAppKind. Required. App type. Default: app.
    WebAppName: # string. Required. App Service name.
    #DeployToSlotFlag: false # boolean. Optional. Use when WebAppKind != "".
    Deploy to slot. Default: false.
    #ResourceGroupName: # string. Required when DeployToSlotFlag = true.
    Resource group.
    #SlotName: # string. Required when DeployToSlotFlag = true. Slot.
    #ImageSource: 'Registry' # 'Registry' | 'Builtin'. Optional. Use when
    WebAppKind = applinux || WebAppKind = linux. Image Source. Default:
    Registry.
    #AzureContainerRegistry: # string. Required when ImageSource =
    AzureContainerRegistry. Registry.
    #AzureContainerRegistryLoginServer: # string. Optional. Use when
    ImageSource = invalidimagesource. Registry Login Server Name.
    #AzureContainerRegistryImage: # string. Required when ImageSource =
    AzureContainerRegistry. Image.
    #AzureContainerRegistryTag: # string. Optional. Use when ImageSource =
    AzureContainerRegistry. Tag.
    #DockerRepositoryAccess: 'public' # 'private' | 'public'. Required when
    ImageSource = invalidImage. Repository Access. Default: public.
    #dockerRegistryConnection: # string. Alias:
    RegistryConnectedServiceName. Required when DockerRepositoryAccess = private
    || ImageSource = PrivateRegistry. Registry Connection.
```

```
#PrivateRegistryImage: # string. Required when ImageSource =
PrivateRegistry. Image.
#PrivateRegistryTag: # string. Optional. Use when ImageSource =
PrivateRegistry. Tag.
#DockerNamespace: # string. Required when WebAppKind != app &&
WebAppKind != functionapp && WebAppKind != api && WebAppKind != mobileapp &&
ImageSource = Registry. Registry or Namespace.
#DockerRepository: # string. Required when WebAppKind != app &&
WebAppKind != functionapp && WebAppKind != api && WebAppKind != mobileapp &&
ImageSource = Registry. Image.
#DockerImageTag: # string. Optional. Use when WebAppKind != app &&
WebAppKind != functionapp && WebAppKind != api && WebAppKind != mobileapp &&
ImageSource = Registry. Tag.
#VirtualApplication: # string. Optional. Use when WebAppKind != linux &&
WebAppKind != applinux && WebAppKind != "". Virtual application.
#Package: '$(System.DefaultWorkingDirectory)/**/*.zip' # string.
Required when WebAppKind != linux && WebAppKind != applinux && WebAppKind !=
"". Package or folder. Default:
$(System.DefaultWorkingDirectory)/**/*.zip.
#packageForLinux: '$(System.DefaultWorkingDirectory)/**/*.zip' # string.
Alias: BuiltinLinuxPackage. Required when WebAppKind != app && WebAppKind !=
functionapp && WebAppKind != api && WebAppKind != mobileapp && ImageSource =
Builtin. Package or folder. Default:
$(System.DefaultWorkingDirectory)/**/*.zip.
#RuntimeStack: # string. Required when WebAppKind != app && WebAppKind !=
functionapp && WebAppKind != api && WebAppKind != mobileapp &&
ImageSource = Builtin. Runtime Stack.
#StartupCommand: # string. Optional. Use when WebAppKind = applinux |||
WebAppKind = linux. Startup command.
# Output
#WebAppUri: # string. Optional. Use when WebAppKind != "". App Service
URL.
# Post Deployment Action
#ScriptType: # 'Inline Script' | 'File Path'. Optional. Use when
WebAppKind != "". Deployment script type.
#InlineScript: ':: You can provide your deployment commands here. One
command per line.' # string. Required when ScriptType == Inline Script &&
WebAppKind != "". Inline Script. Default: :: You can provide your deployment
commands here. One command per line..
#ScriptPath: # string. Required when ScriptType == File Path &&
WebAppKind != "". Deployment script path.
# File Transforms & Variable Substitution Options
#GenerateWebConfig: false # boolean. Optional. Use when WebAppKind !=
linux && WebAppKind != applinux && WebAppKind != "" && Package NotEndsWith
.war. Generate Web.config. Default: false.
#WebConfigParameters: # string. Required when GenerateWebConfig == true
&& WebAppKind != linux && WebAppKind != applinux && WebAppKind != "" &&
Package NotEndsWith .war. Web.config parameters.
#enableXmlTransform: false # boolean. Alias: XmlTransformation.
Optional. Use when WebAppKind != linux && WebAppKind != applinux &&
WebAppKind != "" && Package NotEndsWith .war. XML transformation. Default:
false.
#enableXmlVariableSubstitution: false # boolean. Alias:
XmlVariableSubstitution. Optional. Use when WebAppKind != linux &&
WebAppKind != applinux && WebAppKind != "" && Package NotEndsWith .war. XML
```

```

variable substitution. Default: false.

#JSONFiles: # string. Optional. Use when WebAppKind != linux &&
WebAppKind != applinux && WebAppKind != "" && Package NotEndsWith .war. JSON
variable substitution.

# Application and Configuration Settings
#AppSettings: # string. App settings.
#ConfigurationSettings: # string. Configuration settings.
# Additional Deployment Options
#TakeAppOfflineFlag: false # boolean. Optional. Use when WebAppKind != linux && WebAppKind != applinux && WebAppKind != "". Take App Offline.
Default: false.

#UseWebDeploy: false # boolean. Optional. Use when WebAppKind != linux && WebAppKind != applinux && WebAppKind != "". Publish using Web Deploy.
Default: false.

#SetParametersFile: # string. Optional. Use when UseWebDeploy == true &&
WebAppKind != linux && WebAppKind != applinux && WebAppKind != "".
SetParameters file.

#RemoveAdditionalFilesFlag: false # boolean. Optional. Use when
UseWebDeploy == true && WebAppKind != linux && WebAppKind != applinux && WebAppKind != "".
Remove additional files at destination. Default: false.

#ExcludeFilesFromAppDataFlag: false # boolean. Optional. Use when
UseWebDeploy == true && WebAppKind != linux && WebAppKind != applinux && WebAppKind != "".
Exclude files from the App_Data folder. Default: false.

#AdditionalArguments: # string. Optional. Use when UseWebDeploy == true &&
WebAppKind != linux && WebAppKind != applinux && WebAppKind != "".
Additional arguments.

#RenameFilesFlag: false # boolean. Optional. Use when UseWebDeploy ==
true && WebAppKind != linux && WebAppKind != applinux && WebAppKind != "".
Rename locked files. Default: false.

```

Inputs

`azureSubscription` - Azure subscription

Input alias: `ConnectedServiceName`. `string`. Required.

Specify the Azure Resource Manager subscription for the deployment.

`appType` - App type

Input alias: `WebAppKind`. `string`. Required. Allowed values: `app` (Web App), `applinux` (Linux Web App), `functionapp` (Function App (Not Recommended, Use Azure Functions Task)), `api` (API App), `mobileapp` (Mobile App). Default value: `app`.

Specify the type of web app to deploy.

Note: Specify Linux Web App for built-in platform images or custom container image deployments.

WebAppName - App Service name

`string`. Required.

Specify the name of an existing Azure App Service. App services based on the selected app type will only be listed when using the task assistant.

DeployToSlotFlag - Deploy to slot

`boolean`. Optional. Use when `WebAppKind != ""`. Default value: `false`.

Use this option to deploy to an existing slot other than the Production slot. If this option is not selected, then the Azure App Service will be deployed to the Production slot.

ResourceGroupName - Resource group

`string`. Required when `DeployToSlotFlag = true`.

Specify the Azure Resource group that contains the Azure App Service specified above.

SlotName - Slot

`string`. Required when `DeployToSlotFlag = true`.

Specify an existing slot other than the Production slot.

ImageSource - Image Source

`string`. Optional. Use when `WebAppKind = applinux || WebAppKind = linux`. Allowed values: `Registry` (Container Registry), `Builtin` (Built-in Image). Default value: `Registry`.

App Service on Linux offers two different options to publish your application:

Custom image deployment or app deployment with a built-in platform image.

AzureContainerRegistry - Registry

`string`. Required when `ImageSource = AzureContainerRegistry`.

A globally unique top-level domain name for your specific registry.

Note: A fully qualified image name will be of the format: `<registry>/<repository>:<tag>`. For example, `myregistry.azurecr.io/nginx:latest`.

AzureContainerRegistryLoginServer - Registry Login Server Name

`string`. Optional. Use when `ImageSource = invalidImageSource`.

Specify an Azure container registry login server name.

AzureContainerRegistryImage - Image

`string`. Required when `ImageSource = AzureContainerRegistry`.

The name of the repository where the container images are stored.

Note: A fully qualified image name will be of the format: `<registry>/<repository>`:

`<tag>`. For example, `myregistry.azurecr.io/nginx:latest`.

AzureContainerRegistryTag - Tag

`string`. Optional. Use when `ImageSource = AzureContainerRegistry`.

This is the mechanism that registries use to give Docker images a version.

Note: A fully qualified image name will be of the format: `<registry>/<repository>`:

`<tag>`. For example, `myregistry.azurecr.io/nginx:latest`.

DockerRepositoryAccess - Repository Access

`string`. Required when `ImageSource = invalidImage`. Allowed values: `private`, `public`.

Default value: `public`.

Specify the Docker repository access.

dockerRegistryConnection - Registry Connection

Input alias: `RegistryConnectedServiceName`. `string`. Required when

`DockerRepositoryAccess = private || ImageSource = PrivateRegistry`.

Specify the registry connection.

PrivateRegistryImage - Image

`string`. Required when `ImageSource = PrivateRegistry`.

The name of the repository where the container images are stored.

Note: A fully qualified image name will be of the format: <registry> <repository>: <tag>. For example, myregistry.azurecr.io/nginx:latest.

PrivateRegistryTag - Tag

`string`. Optional. Use when `ImageSource = PrivateRegistry`.

Tags are the mechanism that registries use to give Docker images a version.

Note: A fully qualified image name will be of the format: '<registry>/<repository>: <tag>'. For example, myregistry.azurecr.io/nginx:latest.

DockerNamespace - Registry or Namespace

`string`. Required when `WebAppKind != app && WebAppKind != functionapp && WebAppKind != api && WebAppKind != mobileapp && ImageSource = Registry`.

A globally unique top-level domain name for your specific registry or namespace.

Note: A fully qualified image name will be of the format: <registry or namespace>/<repository>:<tag>. For example, myregistry.azurecr.io/nginx:latest.

DockerRepository - Image

`string`. Required when `WebAppKind != app && WebAppKind != functionapp && WebAppKind != api && WebAppKind != mobileapp && ImageSource = Registry`.

The name of the repository where the container images are stored.

Note: A fully qualified image name will be of the format: '<registry or namespace>/<repository>:<tag>'. For example, myregistry.azurecr.io/nginx:latest.

DockerImageTag - Tag

`string`. Optional. Use when `WebAppKind != app && WebAppKind != functionapp && WebAppKind != api && WebAppKind != mobileapp && ImageSource = Registry`.

This is the mechanism that registries use to give Docker images a version.

Note: A fully qualified image name will be of the format: '<registry or namespace>/<repository>:<tag>'. For example, myregistry.azurecr.io/nginx:latest.

VirtualApplication - Virtual application

`string`. Optional. Use when `WebAppKind != linux && WebAppKind != applinux && WebAppKind != ""`.

Specify the name of the Virtual Application that has been configured in the Azure portal. The option is not required for deployments to the App Service root.

Package - Package or folder

`string`. Required when `WebAppKind != linux && WebAppKind != applinux && WebAppKind != ""`. Default value: `$(System.DefaultWorkingDirectory)/**/*.zip`.

The file path to the package or a folder containing app service contents generated by MSBuild or a compressed zip or war file.

Variables are [Build](#) and [Release](#). [Wildcards](#) are supported.

For example, `$(System.DefaultWorkingDirectory)/*/*/*.zip` or
`$(System.DefaultWorkingDirectory)/*/*/*.war`.

packageForLinux - Package or folder

Input alias: `BuiltinLinuxPackage`. `string`. Required when `WebAppKind != app && WebAppKind != functionapp && WebAppKind != api && WebAppKind != mobileapp && ImageSource = Builtin`. Default value: `$(System.DefaultWorkingDirectory)/**/*.zip`.

The file path to the package or a folder containing app service contents generated by MSBuild or a compressed zip or war file.

Variables are [Build](#) and [Release](#). [Wildcards](#) are supported.

For example, `$(System.DefaultWorkingDirectory)/*/*/*.zip` or
`$(System.DefaultWorkingDirectory)/*/*/*.war`.

RuntimeStack - Runtime Stack

`string`. Required when `WebAppKind != app && WebAppKind != functionapp && WebAppKind != api && WebAppKind != mobileapp && ImageSource = Builtin`.

Specify the framework and version.

StartupCommand - Startup command

`string`. Optional. Use when `WebAppKind = applinux || WebAppKind = linux`.

Specify the startup command.

WebAppUri - App Service URL

`string`. Optional. Use when `WebAppKind` != "".

Specify a name for the output variable that is generated for the URL of the Azure App Service. The variable can be used in subsequent tasks.

ScriptType - Deployment script type

`string`. Optional. Use when `WebAppKind` != "" . Allowed values: `Inline Script`, `File Path` (Script File Path).

Customizes the deployment by providing a script that will run on the Azure App service once the task has completed the deployment successfully . For example, this can restore packages for Node, PHP, and Python applications. Learn more about [Azure App Service Deployment](#).

InlineScript - Inline Script

`string`. Required when `ScriptType == Inline Script && WebAppKind != ""`. Default value: :: You can provide your deployment commands here. One command per line..

ScriptPath - Deployment script path

`string`. Required when `ScriptType == File Path && WebAppKind != ""`.

GenerateWebConfig - Generate Web.config

`boolean`. Optional. Use when `WebAppKind != linux && WebAppKind != applinux && WebAppKind != "" && Package NotEndsWith .war`. Default value: `false`.

A standard `Web.config` will be generated and deployed to Azure App Service if the application does not have one. The values in `web.config` can be edited and vary based on the application framework. For example, for the `node.js` application, `web.config` will have startup file and `iis_node` module values. Learn more about [Azure App Service Deployment](#).

WebConfigParameters - Web.config parameters

`string`. Required when `GenerateWebConfig == true && WebAppKind != linux &&`

```
WebAppKind != applinux && WebAppKind != "" && Package NotEndsWith .war..
```

Edits values like startup files in the generated `web.config` file. This edit feature is only for the generated `web.config`. Learn more about [Azure App Service Deployment](#).

AppSettings - App settings

```
string.
```

Edits web app application settings following the syntax `-key value`. Values containing spaces should be enclosed in double quotes. Examples: `-Port 5000 -RequestTimeout 5000 -WEBSITE_TIME_ZONE "Eastern Standard Time"`.

ConfigurationSettings - Configuration settings

```
string.
```

Edits web app configuration settings following the syntax `-key value`. Values containing spaces should be enclosed in double quotes.

Examples: `-phpVersion 5.6 -linuxFxVersion: node|6.11`.

TakeAppOfflineFlag - Take App Offline

```
boolean. Optional. Use when WebAppKind != linux && WebAppKind != applinux && WebAppKind != "". Default value: false.
```

Use this option to take the Azure App Service offline by placing an `app_offline.htm` file in the root directory of the App Service before the sync operation begins. The file will be removed after the sync operation completes successfully.

UseWebDeploy - Publish using Web Deploy

```
boolean. Optional. Use when WebAppKind != linux && WebAppKind != applinux && WebAppKind != "". Default value: false.
```

`Publish using Web Deploy` options are supported only when using Windows agent. On other platforms, the task relies on [Kudu REST APIs](#) to deploy the Azure App Service, and following options are not supported.

SetParametersFile - SetParameters file

```
string. Optional. Use when UseWebDeploy == true && WebAppKind != linux &&
```

```
WebAppKind != applinux && WebAppKind != "".
```

The location of the `SetParameters.xml` file to use.

RemoveAdditionalFilesFlag - Remove additional files at destination

```
boolean. Optional. Use when UseWebDeploy == true && WebAppKind != linux &&  
WebAppKind != applinux && WebAppKind != "". Default value: false.
```

Use this option to delete files on the Azure App Service that have no matching files in the App Service package or folder.

Note: This will also remove all files related to any extension installed on this Azure App Service. To prevent this, select the `Exclude files from App_Data folder` checkbox.

ExcludeFilesFromAppDataFlag - Exclude files from the App_Data folder

```
boolean. Optional. Use when UseWebDeploy == true && WebAppKind != linux &&  
WebAppKind != applinux && WebAppKind != "". Default value: false.
```

Use this option to prevent files in the `App_Data` folder from being deployed to/deleted from the Azure App Service.

AdditionalArguments - Additional arguments

```
string. Optional. Use when UseWebDeploy == true && WebAppKind != linux &&  
WebAppKind != applinux && WebAppKind != "".
```

The additional Web Deploy arguments following the syntax `-key:value`.

These will be applied when deploying the Azure App Service. Examples: `-`

```
disableLink:AppPoolExtension -disableLink:ContentExtension.
```

See more examples of [Web Deploy Operation Settings ↗](#).

RenameFilesFlag - Rename locked files

```
boolean. Optional. Use when UseWebDeploy == true && WebAppKind != linux &&  
WebAppKind != applinux && WebAppKind != "". Default value: false.
```

Use this option to enable msdeploy flag `MSDEPLOY_RENAME_LOCKED_FILES=1` in Azure App Service application settings. The option enables msdeploy to rename locked files that are locked during app deployment.

`enableXmlTransform` - XML transformation

Input alias: `XmlTransformation`. `boolean`. Optional. Use when `WebAppKind != linux && WebAppKind != applinux && WebAppKind != "" && Package NotEndsWith .war`. Default value: `false`.

The config transforms will be run for `*.Release.config` and `*.<EnvironmentName>.config` on the `*.config file`.

Config transforms will be run prior to the Variable Substitution.

XML transformations are supported only for Windows platform.

`enableXmlVariableSubstitution` - XML variable substitution

Input alias: `XmlVariableSubstitution`. `boolean`. Optional. Use when `WebAppKind != linux && WebAppKind != applinux && WebAppKind != "" && Package NotEndsWith .war`. Default value: `false`.

Variables defined in the build or release pipeline will be matched against the `key` or `name` entries in the `appSettings`, `applicationSettings`, and `connectionStrings` sections of any config file and `parameters.xml`. Variable Substitution is run after config transforms.

Note: If the same variables are defined in the release pipeline and in the environment, then the environment variables will supersede the release pipeline variables.

`JSONFiles` - JSON variable substitution

`string`. Optional. Use when `WebAppKind != linux && WebAppKind != applinux && WebAppKind != "" && Package NotEndsWith .war`.

Provides a new lines-separated list of JSON files to substitute the variable values. Files names are to be provided relative to the root folder.

To substitute JSON variables that are nested or hierarchical, specify them using `JSONPath` expressions.

For example, to replace the value of `ConnectionString` in the sample below, you need to define a variable as `Data.DefaultConnection.ConnectionString` in the build/release pipeline (or the release pipeline's environment).

JSON

```
{  
  "Data": {  
    "DefaultConnection": {  
      "ConnectionString": "Server=(localdb)\\SQLEXPRESS;Database=MyDB;Trusted_Connection=True"  
    }  
  }  
}
```

Variable Substitution is run after configuration transforms.

Note: pipeline variables are excluded in substitution.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

What's new in Version 3.0: Supports File Transformations (XDT) Supports Variable Substitutions(XML, JSON) Learn more about [Azure App Service Deployment](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any

Requirement	Description
Settable variables	Any
Agent version	2.104.1 or greater
Task category	Deploy

AzureRmWebAppDeployment@2 - Azure App Service Deploy v2 task

Article • 09/26/2023

Use this task to update Azure App Service using Web Deploy/[Kudu REST APIs](#).

Syntax

YAML

```
# Azure App Service deploy v2
# Update Azure App Service using Web Deploy / Kudu REST APIs.
- task: AzureRmWebAppDeployment@2
  inputs:
    ConnectedServiceName: # string. Required. Azure Subscription.
    WebAppName: # string. Required. App Service name.
    #DeployToSlotFlag: false # boolean. Deploy to slot. Default: false.
    #ResourceGroupName: # string. Required when DeployToSlotFlag = true.
    Resource group.
    #SlotName: # string. Required when DeployToSlotFlag = true. Slot.
    #VirtualApplication: # string. Virtual Application.
    Package: '$(System.DefaultWorkingDirectory)/**/*.{zip}' # string.
    Required. Package or Folder. Default:
    $(System.DefaultWorkingDirectory)/**/*.{zip}.
    # Output
    #WebAppUri: # string. App Service URL.
    # Additional Deployment Options
    #UseWebDeploy: true # boolean. Publish using Web Deploy. Default: true.
    #SetParametersFile: # string. Optional. Use when UseWebDeploy == true.
    SetParameters File.
    #RemoveAdditionalFilesFlag: false # boolean. Optional. Use when
    UseWebDeploy == true. Remove Additional Files at Destination. Default:
    false.
    #ExcludeFilesFromAppDataFlag: false # boolean. Optional. Use when
    UseWebDeploy == true. Exclude Files from the App_Data Folder. Default:
    false.
    #AdditionalArguments: # string. Optional. Use when UseWebDeploy == true.
    Additional Arguments.
    #TakeAppOfflineFlag: false # boolean. Take App Offline. Default: false.
```

Inputs

`ConnectedServiceName` - Azure Subscription

`string`. Required.

Specify the Azure Resource Manager subscription for the deployment.

WebAppName - App Service name

`string`. Required.

Specify the name of an existing Azure App Service.

DeployToSlotFlag - Deploy to slot

`boolean`. Default value: `false`.

Use this option to deploy to an existing slot other than the Production slot.

ResourceGroupName - Resource group

`string`. Required when `DeployToSlotFlag = true`.

Specify the Azure Resource group that contains the Azure App Service specified above.

SlotName - Slot

`string`. Required when `DeployToSlotFlag = true`.

Specify an existing slot other than the Production slot.

VirtualApplication - Virtual Application

`string`.

Specify the name of the Virtual Application that has been configured in the Azure portal. The option is not required for deployments to the App Service root.

Package - Package or Folder

`string`. Required. Default value: `$(System.DefaultWorkingDirectory)/**/*.zip`.

The folder or file path to the App Service package or folder. Variables include [Build](#) and [Release](#). [Wildcards](#) are supported.

For example, `$(System.DefaultWorkingDirectory)/*/*/*.zip`.

WebAppUri - App Service URL

`string`.

Specify a name for the output variable that is generated for the URL of the App Service. The variable can be used in subsequent tasks.

UseWebDeploy - Publish using Web Deploy

`boolean`. Default value: `true`.

`Publish using web deploy` options are supported only when using a Windows agent. On other platforms, the task relies on [Kudu REST APIs](#) to deploy the App Service, and the following options are not supported.

SetParametersFile - SetParameters File

`string`. Optional. Use when `UseWebDeploy == true`.

Specify the location of the `SetParameters.xml` file to use.

RemoveAdditionalFilesFlag - Remove Additional Files at Destination

`boolean`. Optional. Use when `UseWebDeploy == true`. Default value: `false`.

Use this option to delete files on the Azure App Service that have no matching files in the App Service package or folder.

ExcludeFilesFromAppDataFlag - Exclude Files from the App_Data Folder

`boolean`. Optional. Use when `UseWebDeploy == true`. Default value: `false`.

Exclude files in the `App_Data` folder from being deployed to the Azure App Service.

AdditionalArguments - Additional Arguments

`string`. Optional. Use when `UseWebDeploy == true`.

Additional Web Deploy arguments following the syntax `-key:value`.

These will be applied when deploying the Azure App Service. Example: `-`

`disableLink:AppPoolExtension` `-disableLink:ContentExtension`.

Learn more about [Web Deploy operation settings](#).

TakeAppOfflineFlag - Take App Offline

`boolean`. Default value: `false`.

Use this option to take the Azure App Service offline by placing an `app_offline.htm` file in the root directory of the App Service before the sync operation begins. The file will be removed after the sync operation completes successfully.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.102.0 or greater
Task category	Deploy

AzureAppServiceManage@0 - Azure App Service manage v0 task

Article • 09/26/2023

Start, stop, restart, slot swap, slot delete, install site extensions, or enable continuous monitoring for an Azure App Service.

Syntax

YAML

```
# Azure App Service manage v0
# Start, stop, restart, slot swap, slot delete, install site extensions or
enable continuous monitoring for an Azure App Service.
- task: AzureAppServiceManage@0
  inputs:
    azureSubscription: # string. Alias: ConnectedServiceName. Required.
    Azure subscription.
    #Action: 'Swap Slots' # 'Swap Slots' | 'Start Azure App Service' | 'Stop
    Azure App Service' | 'Restart Azure App Service' | 'Start Swap With Preview'
    | 'Complete Swap' | 'Cancel Swap' | 'Delete Slot' | 'Install Extensions' |
    'Enable Continuous Monitoring' | 'Start all continuous webjobs' | 'Stop all
    continuous webjobs'. Action. Default: Swap Slots.
    WebAppName: # string. Required. App Service name.
    #SpecifySlotOrASE: false # boolean. Alias: SpecifySlot. Optional. Use
    when Action != Swap Slots && Action != Delete Slot && Action != Start Swap
    With Preview && Action != Complete Swap && Action != Cancel Swap. Specify
    Slot or App Service Environment. Default: false.
    #ResourceGroupName: # string. Required when Action = Swap Slots ||
    Action = Delete Slot || SpecifySlot = true || Action = Start Swap With
    Preview || Action = Complete Swap || Action = Cancel Swap. Resource group.
    #SourceSlot: # string. Required when Action = Swap Slots || Action =
    Start Swap With Preview || Action = Complete Swap. Source Slot.
    #SwapWithProduction: true # boolean. Optional. Use when Action = Swap
    Slots || Action = Start Swap With Preview || Action = Complete Swap. Swap
    with Production. Default: true.
    #TargetSlot: # string. Required when SwapWithProduction = false. Target
    Slot.
    #PreserveVnet: false # boolean. Optional. Use when Action = Swap Slots
    || Action = Start Swap With Preview || Action = Complete Swap. Preserve
    Vnet. Default: false.
    #Slot: 'production' # string. Required when Action = Delete Slot ||
    Action = Cancel Swap || SpecifySlot = true. Slot. Default: production.
    #ExtensionsList: # string. Required when Action = Install Extensions.
    Install Extensions.
    #OutputVariable: # string. Optional. Use when Action = Install
    Extensions. Output variable.
    #AppInsightsResourceGroupName: # string. Required when Action == Enable
```

```
Continuous Monitoring. Resource Group name for Application Insights.  
#ApplicationInsightsResourceName: # string. Required when Action ==  
Enable Continuous Monitoring. Application Insights resource name.  
# Advanced Settings  
#ApplicationInsightsWebTestName: # string. Optional. Use when Action ==  
Enable Continuous Monitoring. Application Insights web test name.
```

Inputs

`azureSubscription` - Azure subscription

Input alias: `ConnectedServiceName`. `string`. Required.

Selects the Azure Resource Manager subscription.

`Action` - Action

`string`. Allowed values: `Swap Slots`, `Start Azure App Service` (Start App Service), `Stop Azure App Service` (Stop App Service), `Restart Azure App Service` (Restart App Service), `Start Swap With Preview`, `Complete Swap` (Complete Swap With Preview), `Cancel Swap` (Cancel Swap With Preview), `Delete Slot`, `Install Extensions`, `Enable Continuous Monitoring`, `Start all continuous webjobs`, `Stop all continuous webjobs`. Default value: `Swap Slots`.

Optional. Defines the action to perform on the App Service. You can start, stop, restart, slot swap, start swap with a preview, complete swap with a preview, cancel swap with a preview, install site extensions, or enable continuous monitoring for an Azure App Service.

`WebAppName` - App Service name

`string`. Required.

Enters or selects the name of an existing Azure App Service.

`SpecifySlotOrASE` - Specify Slot or App Service Environment

Input alias: `SpecifySlot`. `boolean`. Optional. Use when `Action != Swap Slots && Action != Delete Slot && Action != Start Swap With Preview && Action != Complete Swap && Action != Cancel Swap`. Default value: `false`.

ResourceGroupName - Resource group

```
string. Required when Action = Swap Slots || Action = Delete Slot || SpecifySlot = true || Action = Start Swap With Preview || Action = Complete Swap || Action = Cancel Swap.
```

Enters or selects the Azure Resource Group that contains the Azure App Service specified above.

SourceSlot - Source Slot

```
string. Required when Action = Swap Slots || Action = Start Swap With Preview || Action = Complete Swap.
```

Used as source slot when `action == Swap Slots`. The swap action directs destination slot's traffic to the source slot.

SwapWithProduction - Swap with Production

```
boolean. Optional. Use when Action = Swap Slots || Action = Start Swap With Preview || Action = Complete Swap. Default value: true.
```

Swaps the traffic of the source slot with production. If you don't select this option, then you need to provide the source and target slot names.

TargetSlot - Target Slot

```
string. Required when SwapWithProduction = false.
```

Use as the destination slot when `action == Swap Slots`. The swap action directs the destination slot's traffic to the source slot.

PreserveVnet - Preserve Vnet

```
boolean. Optional. Use when Action = Swap Slots || Action = Start Swap With Preview || Action = Complete Swap. Default value: false.
```

Preserves the virtual network settings.

Slot - Slot

```
string. Required when Action = Delete Slot || Action = Cancel Swap || SpecifySlot = true. Default value: production.
```

ExtensionsList - Install Extensions

`string`. Required when `Action = Install Extensions`.

Site extensions run on Microsoft Azure App Service. You can install a set of tools as a site extension and better manage your Azure App Service. Restart the App Service so the latest changes take effect.

OutputVariable - Output variable

`string`. Optional. Use when `Action = Install Extensions`.

Provides the variable name for the selected extension's local installation path.

This field is now deprecated and will be removed. Use the

`LocalPathsForInstalledExtensions` variable from the Output Variables section in subsequent tasks.

AppInsightsResourceGroupName - Resource Group name for Application Insights

`string`. Required when `Action == Enable Continuous Monitoring`.

Enters or selects the resource group where your Application Insights resource is available.

ApplicationInsightsResourceName - Application Insights resource name

`string`. Required when `Action == Enable Continuous Monitoring`.

Selects the Application Insights resource where continuous monitoring data is recorded.

If your Application Insights resource is not listed here and you want to create a new resource, select **+New**. Once you create the resource in the Azure portal, come back here and select **Refresh**.

ApplicationInsightsWebTestName - Application Insights web test name

`string`. Optional. Use when `Action == Enable Continuous Monitoring`.

Optional. Enters the Application Insights web test name you want to create or update.

If you don't provide a web test name, the default test name is used.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

`LocalPathsForInstalledExtensions`

This input is the local installation paths for the extensions you select.

If you select multiple extensions, the output is a comma-separated list of local paths for each of the extensions you select. The output lists the paths in the order they appear in the Install Extensions field.

Remarks

Use this task to start, stop, restart, slot swap, Swap with Preview, install site extensions, or enable continuous monitoring for an Azure App Service.

What happens during a swap

When you swap two slots (usually from a staging slot into the production slot), make sure that the production slot is always the target slot. This way, the swap operation doesn't affect your production app.

Also at any point of the swap (or swap with preview) operation, all work of initializing the swapped apps happens on the source slot. The target slot remains online while the source slot is being prepared and warmed up, regardless of where the swap succeeds or fails.

For more information, see [Set up staging environments in Azure App Service](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None

Requirement	Description
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.102.0 or greater
Task category	Deploy

AzureAppServiceSettings@1 - Azure App Service Settings v1 task

Article • 09/26/2023

Updates or adds app service settings in an Azure Web App for Linux or Windows.

Syntax

YAML

```
# Azure App Service Settings v1
# Update/Add App settings an Azure Web App for Linux or Windows.
- task: AzureAppServiceSettings@1
  inputs:
    azureSubscription: # string. Alias: ConnectedServiceName. Required.
    Azure subscription.
    appName: # string. Required. App Service name.
    resourceGroupName: # string. Required. Resource group.
    #slotName: 'production' # string. Slot. Default: production.
    # Application and Configuration Settings
    #appSettings: # string. App settings.
    #generalSettings: # string. General settings.
    #connectionStrings: # string. Connection Strings.
```

Inputs

`azureSubscription` - Azure subscription

Input alias: `ConnectedServiceName`. `string`. Required.

Selects the Azure Resource Manager subscription.

`appName` - App Service name

`string`. Required.

Enters or selects the name of an existing Azure App Service.

`resourceGroupName` - Resource group

`string`. Required.

Enters or selects the Azure Resource Group that contains the Azure App Service specified above.

slotName - Slot

`string`. Default value: `production`.

Enters or selects an existing slot. If you don't select a slot, changes are made to production.

appSettings - App settings

`string`.

Application settings in JSON syntax. Enclose values containing spaces in double quotes. For more information, see [Configure app settings](#).

The following is an example of the JSON syntax:

JSON

```
[  
  {  
    "name": "key1",  
    "value": "valueabcd",  
    "slotSetting": false  
  },  
  {  
    "name": "key2",  
    "value": "valueefgh",  
    "slotSetting": true  
  }]
```

generalSettings - General settings

`string`.

General settings in JSON syntax. Enclose values containing spaces in double quotes. For a list of the available properties, see the [App Service SiteConfig object documentation](#). For more information, see [Configure general settings](#).

The following is an example of the JSON syntax:

JSON

```
[  
  {  
    "alwaysOn": true,  
    "webSocketsEnabled": false  
  }  
]
```

connectionStrings - Connection Strings

string.

Connection strings in JSON syntax. Enclose values containing spaces in double quotes. For more information, see [Configure connection strings](#).

The following is an example of the JSON syntax:

JSON

```
[  
  {  
    "name": "key1",  
    "value": "valueabcd",  
    "type": " MySql ",  
    "slotSetting": false  
  },  
  {  
    "name": "key2",  
    "value": "valueefgh",  
    "type": " Custom ",  
    "slotSetting": true  
  }  
]
```

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to configure App settings, connection strings and other general settings in bulk using JSON syntax on your web app or any of its deployment slots. The task works on cross platform Azure Pipelines agents running Windows, Linux or Mac. The task works for ASP.NET, ASP.NET Core, PHP, Java, Python, Go and Node.js based web applications.

Examples

The following example YAML snippet deploys a web application to an Azure Web App service running on windows.

YAML

```
variables:
  azureSubscription: Contoso
  WebApp_Name: sampleWebApp
  # To ignore SSL error uncomment the below variable
  # VSTS_ARM_REST_IGNORE_SSL_ERRORS: true

steps:
  - task: AzureWebApp@1
    displayName: Azure Web App Deploy
    inputs:
      azureSubscription: $(azureSubscription)
      appName: $(WebApp_Name)
      package: $(System.DefaultWorkingDirectory)/**/*.zip

  - task: AzureAppServiceSettings@1
    displayName: Azure App Service Settings
    inputs:
      azureSubscription: $(azureSubscription)
      appName: $(WebApp_Name)
      # To deploy the settings on a slot, provide slot name as below. By
      default, the settings would be applied to the actual Web App (Production
      slot)
      # slotName: staging
      appSettings: |
        [
          {
            "name": "APPINSIGHTS_INSTRUMENTATIONKEY",
            "value": "$(Key)",
            "slotSetting": false
          },
          {
            "name": "MYSQL_DATABASE_NAME",
            "value": "$(DB_Name)",
            "slotSetting": false
          }
        ]
```

```

        ]
    generalSettings: |
        [
            {
                "alwaysOn": true,
                "webSocketsEnabled": false
            }
        ]
    connectionStrings: |
        [
            {
                "name": "MysqlCredentials",
                "value": "$(MySQL_ConnectionString)",
                "type": " MySql",
                "slotSetting": false
            }
        ]

```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.104.1 or greater
Task category	Deploy

AzureCLI@2 - Azure CLI v2 task

Article • 09/26/2023

Run Azure CLI commands against an Azure subscription in a PowerShell Core/shell script when running on Linux agent. Or, run Azure CLI commands against an Azure subscription in a PowerShell/PowerShell Core/batch script when running on Windows agent.

Syntax

YAML

```
# Azure CLI v2
# Run Azure CLI commands against an Azure subscription in a PowerShell
Core/Shell script when running on Linux agent or PowerShell/PowerShell
Core/Batch script when running on Windows agent.
- task: AzureCLI@2
  inputs:
    azureSubscription: # string. Alias: connectedServiceNameARM. Required.
    Azure Resource Manager connection.
    scriptType: # 'ps' | 'pscore' | 'batch' | 'bash'. Required. Script Type.
    scriptLocation: 'scriptPath' # 'inlineScript' | 'scriptPath'. Required.
    Script Location. Default: scriptPath.
    scriptPath: # string. Required when scriptLocation = scriptPath. Script
    Path.
    #inlineScript: # string. Required when scriptLocation = inlineScript.
    Inline Script.
    #arguments: # string. Alias: scriptArguments. Script Arguments.
    #powerShellErrorActionPreference: 'stop' # 'stop' | 'continue' |
    'silentlyContinue'. Optional. Use when scriptType = ps || scriptType =
    pscore. ErrorActionPreference. Default: stop.
    # Advanced
    #addSpnToEnvironment: false # boolean. Access service principal details
    in script. Default: false.
    #useGlobalConfig: false # boolean. Use global Azure CLI configuration.
    Default: false.
    #workingDirectory: # string. Alias: cwd. Working Directory.
    #failOnStandardError: false # boolean. Fail on Standard Error. Default:
    false.
    #powerShellIgnoreLASTEXITCODE: false # boolean. Optional. Use when
    scriptType = ps || scriptType = pscore. Ignore $LASTEXITCODE. Default:
    false.
```

Inputs

`azureSubscription` - Azure Resource Manager connection

Input alias: `connectedServiceNameARM`. `string`. Required.

Select an Azure Resource Manager service connection for the deployment.

`scriptType` - Script Type

`string`. Required. Allowed values: `ps` (PowerShell), `pscore` (PowerShell Core), `batch`, `bash` (Shell).

Type of script. Select a `bash` or `pscore` script when running on Linux agent. Or, select a `batch`, `ps`, or `pscore` script when running on Windows agent. A `pscore` script can run on cross-platform agents (Linux, macOS, or Windows).

`scriptLocation` - Script Location

`string`. Required. Allowed values: `inlineScript` (Inline script), `scriptPath` (Script path). Default value: `scriptPath`.

Path to the script.

`scriptPath` - Script Path

`string`. Required when `scriptLocation = scriptPath`.

Fully qualified path of the script. Use `.ps1`, `.bat`, or `.cmd` when using Windows-based agent. Use `.ps1` or `.sh` when using Linux-based agent or a path relative to the default working directory.

`inlineScript` - Inline Script

`string`. Required when `scriptLocation = inlineScript`.

You can write your scripts inline here. When using Windows agent, use PowerShell, PowerShell Core, or batch scripting. Use PowerShell Core or shell scripting when using Linux-based agents. For batch files, use the prefix `call` before every Azure command. You can also pass predefined and custom variables to this script by using arguments.

The following is an example for PowerShell/PowerShellCore/shell.

```
az --version
```

```
az account show
```

The following is an example for batch.

```
call az --version
call az account show
```

arguments - Script Arguments

Input alias: `scriptArguments`. `string`.

Arguments passed to the script.

powerShellErrorActionPreference - ErrorActionPreference

`string`. Optional. Use when `scriptType = ps || scriptType = pscore`. Allowed values: `stop`, `continue`, `silentlyContinue`. Default value: `stop`.

Prepends the line `$ErrorActionPreference = 'VALUE'` at the top of your PowerShell/PowerShell Core script.

addSpnToEnvironment - Access service principal details in script

`boolean`. Default value: `false`.

Adds the service principal ID, service principal key, and tenant ID of the Azure endpoint you chose to the script's execution environment. You can use the `servicePrincipalId`, `servicePrincipalKey` and `tenantId` variables in your script.

This is honored only when the Azure endpoint has service principal authentication scheme.

The following list shows the syntax to access environment variables based on the script type.

- PowerShell script syntax: `$env:servicePrincipalId`
- Batch script syntax: `%servicePrincipalId%`
- Shell script syntax: `$servicePrincipalId`

`useGlobalConfig` - Use global Azure CLI configuration

`boolean`. Default value: `false`.

If this input is false, this task will use its own [Azure CLI configuration directory](#). Use this task to run Azure CLI tasks in *parallel* releases.

`workingDirectory` - Working Directory

Input alias: `cwd`. `string`.

Current working directory where the script is run. If left blank, this input is the root of the repo (build) or artifacts (release), which is `$(System.DefaultWorkingDirectory)`.

`failOnStandardError` - Fail on Standard Error

`boolean`. Default value: `false`.

If this input is true, this task will fail when any errors are written to the StandardError stream. Clear the checkbox to ignore standard errors and instead rely on exit codes to determine the status.

`powerShellIgnoreLASTEXITCODE` - Ignore \$LASTEXITCODE

`boolean`. Optional. Use when `scriptType = ps || scriptType = pscore`. Default value: `false`.

If this input is false, the line `if ((Test-Path -LiteralPath variable:\LASTEXITCODE)) { exit $LASTEXITCODE }` is appended to the end of your script. This will propagate the last exit code from an external command as the exit code of PowerShell. Otherwise, the line is not appended to the end of your script.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

What's new in Version task version 2.0

- Support for PowerShell and PowerShell Core script.
- PowerShell Core works with cross-platform agents (Linux, macOS, or Windows), make sure the agent has PowerShell version 6 or more.
- Powershell script works with only Windows agent, make sure the agent has PowerShell version 5 or below.

Prerequisites

- A Microsoft Azure subscription.
- [Azure Resource Manager service connection](#) to your Azure account.
- Microsoft hosted agents have Azure CLI pre-installed. However if you are using private agents, [install Azure CLI](#) on the computer(s) that run the build and release agent. If an agent is already running on the machine on which the Azure CLI is installed, restart the agent to ensure all the relevant stage variables are updated.

Examples

The following example lists the version of Azure CLI and gets the details of the subscription.

YAML

```
- task: AzureCLI@2
  displayName: Azure CLI
  inputs:
    azureSubscription: <Name of the Azure Resource Manager service
connection>
    scriptType: ps
    scriptLocation: inlineScript
    inlineScript: |
      az --version
      az account show
```

The following example illustrates how to pass arguments to your script.

- Passing arguments to inline scripts:

YAML

```

- task: AzureCLI@2
  inputs:
    azureSubscription: <Azure_Resource_Manager_Service_Connection>
    scriptType: 'ps'
    scriptLocation: 'inlineScript'
    arguments: '$(AZURE_STORAGE_ACCOUNT) $(AZURE_STORAGE_KEY)'
    inlineScript: './scripts/publish.ps1 $1 $2'

```

- Passing arguments with script path:

YAML

```

- task: AzureCLI@2
  inputs:
    azureSubscription: <Azure_Resource_Manager_Service_Connection>
    scriptType: 'ps'
    scriptLocation: 'scriptPath'
    arguments: '$(AZURE_STORAGE_ACCOUNT) $(AZURE_STORAGE_KEY)'
    scriptPath: './scripts/publish.ps1'

```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.0.0 or greater
Task category	Deploy

See also

- [Azure Resource Group Deployment](#)
- [Azure Cloud Service Deployment](#)
- [Azure Web App Deployment](#)

AzureCLI@1 - Azure CLI v1 task

Article • 09/26/2023

Run Azure CLI commands against an Azure subscription in a shell script when running on Linux agent or batch script when running on Windows agent.

Syntax

YAML

```
# Azure CLI v1
# Run Azure CLI commands against an Azure subscription in a Shell script
when running on Linux agent or Batch script when running on Windows agent.
- task: AzureCLI@1
  inputs:
    azureSubscription: # string. Alias: connectedServiceNameARM. Required.
    Azure subscription.
    scriptLocation: 'scriptPath' # 'inlineScript' | 'scriptPath'. Required.
    Script Location. Default: scriptPath.
    scriptPath: # string. Required when scriptLocation = scriptPath. Script
    Path.
    #inlineScript: # string. Required when scriptLocation = inlineScript.
    Inline Script.
    #arguments: # string. Alias: args. Arguments.
    # Advanced
    #addSpnToEnvironment: false # boolean. Access service principal details
    in script. Default: false.
    #useGlobalConfig: false # boolean. Use global Azure CLI configuration.
    Default: false.
    #workingDirectory: # string. Alias: cwd. Working Directory.
    #failOnStandardError: false # boolean. Fail on Standard Error. Default:
    false.
```

Inputs

`azureSubscription` - Azure subscription

Input alias: `connectedServiceNameARM`. `string`. Required.

Selects an Azure Resource Manager subscription for the deployment.

`scriptLocation` - Script Location

`string`. Required. Allowed values: `inlineScript` (Inline script), `scriptPath` (Script path). Default value: `scriptPath`.

Selects the script location.

scriptPath - Script Path

`string`. Required when `scriptLocation = scriptPath`.

Fully qualified path of the script or a path relative to the the default working directory.

inlineScript - Inline Script

`string`. Required when `scriptLocation = inlineScript`.

You can write your scripts inline here. When using Windows agent, use batch scripting. Use shell scripting when using Linux-based agents. For batch files, use the prefix `call` before every Azure command. You can also pass predefined and custom variables to this script using arguments

See the following examples. The first is a shell example and the second is a batch example:

```
azure --version || azure account show
```

```
call azure --version || call azure account show
```

arguments - Arguments

Input alias: `args`. `string`.

Arguments passed to the script.

addSpnToEnvironment - Access service principal details in script

`boolean`. Default value: `false`.

Adds the service principal ID and key of the Azure endpoint that you chose to the script's execution environment. You can use the `$servicePrincipalId` and `$servicePrincipalKey` variables in your script.

This is honored only when the Azure endpoint has Service Principal authentication scheme.

`useGlobalConfig` - Use global Azure CLI configuration

`boolean`. Default value: `false`.

If this is false, this task will use its own separate [Azure CLI configuration directory](#). This can be used to run Azure CLI tasks in *parallel* releases.

`workingDirectory` - Working Directory

Input alias: `cwd`. `string`.

Current working directory where the script is run. If left blank, this input is the root of the repo (build) or artifacts (release), which is `$(System.DefaultWorkingDirectory)`.

`failOnStandardError` - Fail on Standard Error

`boolean`. Default value: `false`.

If this input is true, this task will fail when any errors are written to the StandardError stream. Clear the checkbox to ignore standard errors and instead rely on exit codes to determine the status.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

What's new in Version 1.0:

- Supports the new Azure CLI 2.0 which is Python based
- Works with cross-platform agents (Linux, macOS, or Windows)
- For working with Azure CLI 1.0 (node.js-based), switch to task version 0.0

- Limitations: - No support for Azure Classic subscriptions. Azure CLI 2.0 supports only Azure Resource Manager (ARM) subscriptions.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.0.0 or greater
Task category	Deploy

AzureCLI@0 - Azure CLI Preview v0 task

Article • 09/26/2023

Run a shell or batch script with Azure CLI commands against an Azure subscription.

Syntax

YAML

```
# Azure CLI Preview v0
# Run a Shell or Batch script with Azure CLI commands against an azure
subscription.
- task: AzureCLI@0
  inputs:
    connectedServiceNameSelector: 'connectedServiceNameARM' #
'connectedServiceName' | 'connectedServiceNameARM'. Required. Azure
Connection Type. Default: connectedServiceNameARM.
    connectedServiceNameARM: # string. Required when
connectedServiceNameSelector = connectedServiceNameARM. AzureRM
Subscription.
    #connectedServiceName: # string. Required when
connectedServiceNameSelector = connectedServiceName. Azure Classic
Subscription.
    scriptLocation: 'scriptPath' # 'inlineScript' | 'scriptPath'. Required.
Script Location. Default: scriptPath.
    scriptPath: # string. Required when scriptLocation = scriptPath. Script
Path.
    #inlineScript: # string. Required when scriptLocation = inlineScript.
Inline Script.
    #args: # string. Arguments.
    # Advanced
    #cwd: # string. Working Directory.
    #failOnStandardError: true # boolean. Fail on Standard Error. Default:
true.
```

Inputs

`connectedServiceNameSelector` - Azure Connection Type

`string`. Required. Allowed values: `connectedServiceName` (Azure Classic),

`connectedServiceNameARM` (Azure Resource Manager). Default value:

`connectedServiceNameARM`.

Selects the Azure connection type for the deployment.

connectedServiceNameARM - AzureRM Subscription

`string`. Required when `connectedServiceNameSelector = connectedServiceNameARM`.

Selects the Azure Resource Manager subscription for the deployment.

connectedServiceName - Azure Classic Subscription

`string`. Required when `connectedServiceNameSelector = connectedServiceName`.

Selects the Azure Classic subscription for the deployment.

scriptLocation - Script Location

`string`. Required. Allowed values: `inlineScript` (Inline Script), `scriptPath` (Script Path).

Default value: `scriptPath`.

Selects the script location.

scriptPath - Script Path

`string`. Required when `scriptLocation = scriptPath`.

Fully qualified path of the script or a path relative to the the default working directory.

inlineScript - Inline Script

`string`. Required when `scriptLocation = inlineScript`.

You can write your scripts inline here. For batch files, use the prefix `call` before every Azure command. You can also pass predefined and custom variables to this script using arguments.

See the following examples. The first is a shell example and the second is a batch example:

```
azure --version || azure account show
```

```
call azure --version || call azure account show
```

`args` - Arguments

`string`.

Arguments passed to the script.

`cwd` - Working Directory

`string`.

Current working directory where the script is run. If left blank, this input is the root of the repo (build) or artifacts (release), which is `$(System.DefaultWorkingDirectory)`.

`failOnStandardError` - Fail on Standard Error

`boolean`. Default value: `true`.

If this is true, this task will fail when any errors are written to the StandardError stream.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any

Requirement	Description
Agent version	1.95.0 or greater
Task category	Deploy

AzureCloudPowerShellDeployment@2 - Azure Cloud Service deployment v2 task

Article • 09/26/2023

Deploy an Azure Cloud Service.

Syntax

YAML

```
# Azure Cloud Service deployment v2
# Deploy an Azure Cloud Service.
- task: AzureCloudPowerShellDeployment@2
  inputs:
    ARMConnectedServiceName: # string. Required. Azure subscription (ARM).
    ResourceGroupName: # string. Required. Resource group.
    ARMStorageAccount: # string. Required. Storage account (ARM).
    ServiceName: # string. Required. Service name.
    ServiceLocation: # string. Required. Service location.
    CsCfg: # string. Required. CsCfg.
    CsDef: # string. Required. CsDef.
    CsPkg: # string. Required. CsPkg.
    #KeyVault: # string. Azure KeyVault.
    #DeploymentLabel: '${Build.BuildNumber}' # string. Deployment label.
    Default: ${Build.BuildNumber}.
    #AppendDateTimeToLabel: false # boolean. Append current date and time.
    Default: false.
    #UpgradeMode: 'Auto' # string. Update mode for the cloud service.
    Default: Auto.
    #AllowUpgrade: true # boolean. Allow upgrade. Default: true.
    #VerifyRoleInstanceStatus: false # boolean. Verify role instance status.
    Default: false.
    # Advanced Options For Creating New Service
    #DiagnosticStorageAccountKeys: # string. Diagnostic storage account
    keys.
```

Inputs

`ARMConnectedServiceName` - Azure subscription (ARM)

`string`. Required.

Azure Resource Manager subscription.

ResourceGroupName - Resource group

`string`. Required.

Enter or Select the Azure Resource Group that contains the Azure App Service specified above.

ARMStorageAccount - Storage account (ARM)

`string`. Required.

A pre-existing ARM storage account.

ServiceName - Service name

`string`. Required.

An existing cloud service name.

ServiceLocation - Service location

`string`. Required.

A region for new service deployment. Options include: East US, East US 2, Central US, South Central US, West US, North Europe, West Europe, and others.

CsCfg - CsCfg

`string`. Required.

The CsCfg path in the default artifact directory.

CsDef - CsDef

`string`. Required.

Path of CsDef under the default artifact directory.

CsPkg - CsPkg

`string`. Required.

Path to the CsPkg in the default artifact directory.

KeyVault - Azure KeyVault`string`.

Choose a pre-existing Azure KeyVault with certificates.

DeploymentLabel - Deployment label`string`. Default value: `$(Build.BuildNumber)`.

Specifies the label name for the new deployment. If not specified, defaults to a Globally Unique Identifier (GUID).

AppendDateTimeToLabel - Append current date and time`boolean`. Default value: `false`.

Appends current date and time to the deployment label.

UpgradeMode - Update mode for the cloud service`string`. Default value: `Auto`.

Auto, Manual or Simultaneous.

AllowUpgrade - Allow upgrade`boolean`. Default value: `true`.

Allows an upgrade to the Microsoft Azure deployment.

VerifyRoleInstanceStatus - Verify role instance status`boolean`. Default value: `false`.

Causes the task to wait until role instances are in the ready state.

DiagnosticStorageAccountKeys - Diagnostic storage account keys`string`.

Format storage key string as `Role:Storagekey`. The diagnostics storage account name for each role is retrieved from the diagnostic config file (.wadcfgx).

- If the .wadcfgx file for a role is not found: The diagnostic extension isn't set for that role.

- If the storage account name is not found in the .wadcfgx file: The default storage account is used for storing diagnostic results, and storage key parameters from the deployment task is ignored.

If there is sensitive information in the diagnostic results for your environment, save the `storage_account_key` as a secret variable. For example:

- WebRole: `WebRole_storage_account_key`
- WorkerRole: `WorkerRole_stoarge_account_key`

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: azureps
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.103.0 or greater
Task category	Deploy

AzureCloudPowerShellDeployment@1 - Azure Cloud Service deployment v1 task

Article • 09/26/2023

Deploy an Azure Cloud Service.

ⓘ Note

This task does not support Azure Resource Manager authentication with workflow identity federation.

Syntax

```
# Azure Cloud Service deployment v1
# Deploy an Azure Cloud Service.
- task: AzureCloudPowerShellDeployment@1
  inputs:
    azureClassicSubscription: # string. Alias: ConnectedServiceName.
    Required. Azure subscription (Classic).
    #EnableAdvancedStorageOptions: false # boolean. Enable ARM storage
    support. Default: false.
    StorageAccount: # string. Required when EnableAdvancedStorageOptions =
    false. Storage account (Classic).
    #ARMConnectedServiceName: # string. Required when
    EnableAdvancedStorageOptions = true. Azure subscription (ARM).
    #ARMStorageAccount: # string. Required when EnableAdvancedStorageOptions =
    true. Storage account (ARM).
    ServiceName: # string. Required. Service name.
    ServiceLocation: # string. Required. Service location.
    CsPkg: # string. Required. CsPkg.
    CsCfg: # string. Required. CsCfg.
    slotName: 'Production' # string. Alias: Slot. Required. Environment
    (Slot). Default: Production.
    #DeploymentLabel: '$(Build.BuildNumber)' # string. Deployment label.
    Default: $(Build.BuildNumber).
    #AppendDateTimeToLabel: false # boolean. Append current date and time.
    Default: false.
    #AllowUpgrade: true # boolean. Allow upgrade. Default: true.
    #SimultaneousUpgrade: false # boolean. Optional. Use when AllowUpgrade
    == true. Simultaneous upgrade. Default: false.
    #ForceUpgrade: false # boolean. Optional. Use when AllowUpgrade == true.
    Force upgrade. Default: false.
    #VerifyRoleInstanceStatus: false # boolean. Verify role instance status.
    Default: false.
    # Advanced Options For Creating New Service
    #DiagnosticStorageAccountKeys: # string. Diagnostic storage account
```

```
keys.  
#NewServiceCustomCertificates: # string. Custom certificates to import.  
#NewServiceAdditionalArguments: # string. Additional arguments.  
#NewServiceAffinityGroup: # string. Affinity group.
```

Inputs

azureClassicSubscription - Azure subscription (Classic)

Input alias: `ConnectedServiceName`. `string`. Required.

The Azure subscription to target for deployment.

EnableAdvancedStorageOptions - Enable ARM storage support

`boolean`. Default value: `false`.

Enables or disables ARM storage support.

StorageAccount - Storage account (Classic)

`string`. Required when `EnableAdvancedStorageOptions = false`.

The storage account must exist prior to deployment.

ARMConnectedServiceName - Azure subscription (ARM)

`string`. Required when `EnableAdvancedStorageOptions = true`.

The ARM subscription.

ARMStorageAccount - Storage account (ARM)

`string`. Required when `EnableAdvancedStorageOptions = true`.

A pre-existing ARM storage account.

ServiceName - Service name

`string`. Required.

An existing cloud service name.

ServiceLocation - Service location

`string`. Required.

A region for new service deployment. Options include: East US, East US 2, Central US, South Central US, West US, North Europe, West Europe, and others.

CsPkg - CsPkg

`string`. Required.

Path to the CsPkg in the default artifact directory.

CsCfg - CsCfg

`string`. Required.

The CsCfg path in the default artifact directory.

slotName - Environment (Slot)

Input alias: `Slot`. `string`. Required. Default value: `Production`.

Set this value to 'Staging' or use the default.

DeploymentLabel - Deployment label

`string`. Default value: `$(Build.BuildNumber)`.

Specifies the label name for the new deployment. If not specified, defaults to a Globally Unique Identifier (GUID).

AppendDateTimeToLabel - Append current date and time

`boolean`. Default value: `false`.

Appends current date and time to the deployment label.

AllowUpgrade - Allow upgrade

`boolean`. Default value: `true`.

Allows an upgrade to the Microsoft Azure deployment.

SimultaneousUpgrade - Simultaneous upgrade

`boolean`. Optional. Use when `AllowUpgrade == true`. Default value: `false`.

Upgrades all instances at once. Your cloud service is unavailable during this time.

ForceUpgrade - Force upgrade

`boolean`. Optional. Use when `AllowUpgrade == true`. Default value: `false`.

Sets a forced upgrade. Forcing an upgrade can cause loss of local data.

VerifyRoleInstanceStatus - Verify role instance status

`boolean`. Default value: `false`.

Causes the task to wait until role instances are in the ready state.

DiagnosticStorageAccountKeys - Diagnostic storage account keys

`string`.

Format storage key string as `Role:Storagekey`. The diagnostics storage account name for each role is retrieved from the diagnostic config file (.wadcfgx).

- If the .wadcfgx file for a role is not found: The diagnostic extension isn't set for that role.
- If the storage account name is not found in the .wadcfgx file: The default storage account is used for storing diagnostic results, and storage key parameters from the deployment task is ignored.

NOTE: If there is sensitive information in the diagnostic results for your environment, save the `storage_account_key` as a secret variable. For example:

- WebRole: `WebRole_storage_account_key`
- WorkerRole: `WorkerRole_stoarge_account_key`

NewServiceCustomCertificates - Custom certificates to import

`string`.

Format the custom certificate string as `CertificatePfxBase64:CertificatePassword`. Save the `certificate_password` as a secret variable. For example:

- Certificate1: `Certificate1_password`
- Certificate2: `Certificate2_password`

NewServiceAdditionalArguments - Additional arguments

`string`.

Passes additional arguments when creating a new service. Arguments are passed to the `New-AzureService` cmdlet. For example, `-Label 'MyTestService'`.

NewServiceAffinityGroup - Affinity group

`string`.

The affinity group used instead of service location when creating a new service.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to deploy an Azure Cloud Service.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: azureps

Requirement	Description
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.103.0 or greater
Task category	Deploy

AzureContainerApps@1 - Azure Container Apps Deploy v1 task

Article • 09/26/2023

An Azure DevOps Task to build and deploy Azure Container Apps.

Syntax

YAML

```
# Azure Container Apps Deploy v1
# An Azure DevOps Task to build and deploy Azure Container Apps.
- task: AzureContainerApps@1
  inputs:
    # advanced
    #workingDirectory: # string. Alias: cwd. Working Directory.
    #appSourcePath: # string. Application source path.
    azureSubscription: # string. Alias: connectedServiceNameARM. Required.
      Azure Resource Manager connection.
    #acrName: # string. Azure Container Registry name.
    #acrUsername: # string. Azure Container Registry username.
    #acrPassword: # string. Azure Container Registry password.
    #dockerfilePath: # string. Dockerfile path.
    #imageToBuild: # string. Docker image to build.
    #imageToDeploy: # string. Docker image to deploy.
    #containerAppName: # string. Azure Container App name.
    #resourceGroup: # string. Azure resource group name.
    #containerAppEnvironment: # string. Azure Container App environment.
    #runtimeStack: # string. Application runtime stack.
    #targetPort: # string. Application target port.
    #location: # string. Location of the Container App.
    #environmentVariables: # string. Environment variables.
    #ingress: # string. Ingress setting.
    #yamlConfigPath: # string. YAML configuration file path.
    #disableTelemetry: # boolean. Disable telemetry.
```

Inputs

`workingDirectory` - Working Directory

Input alias: `cwd`. `string`.

Current working directory where the script is run. Empty is the root of the repo (build) or artifacts (release), which is `$(System.DefaultWorkingDirectory)`.

appSourcePath - Application source path`string.`

Absolute path on the runner of the source application code to be built. If not provided, the 'imageToDeploy' argument must be provided to ensure the Container App has an image to reference.

When pushing a new image to ACR, the `acrName` and `appSourcePath` task inputs are required.

azureSubscription - Azure Resource Manager connection

Input alias: `connectedServiceNameARM`. `string`. Required.

Specify an Azure Resource Manager service connection for the deployment. This service connection must be linked to the user's Azure Subscription where the Container App will be created/updated. This service connection *must* have proper permissions to make these changes within the subscription, for example Contributor role.

acrName - Azure Container Registry name`string.`

The name of the Azure Container Registry that the runnable application image will be pushed to.

When pushing a new image to ACR, the `acrName` and `appSourcePath` task inputs are required.

acrUsername - Azure Container Registry username`string.`

The username used to authenticate push requests to the provided Azure Container Registry. If not provided, an access token will be generated via 'az acr login' and provided to 'docker login' to authenticate the requests.

acrPassword - Azure Container Registry password`string.`

The password used to authenticate push requests to the provided Azure Container Registry. If not provided, an access token will be generated via 'az acr login' and provided to 'docker login' to authenticate the requests.

dockerfilePath - Dockerfile path

`string.`

Relative path (_without file prefixes (see the following [Examples](#)) to the Dockerfile in the provided application source that should be used to build the image that is then pushed to ACR and deployed to the Container App. If not provided, this task will check if there is a file named 'Dockerfile' at the root of the provided application source and use that to build the image. Otherwise, the Oryx++ Builder will be used to create the image.

imageToBuild - Docker image to build

`string.`

The custom name of the image that is to be built, pushed to ACR and deployed to the Container App by this task. Note: this image name should include the ACR server; e.g., `<acr-name>.azurecr.io/<repo>:<tag>`. If this argument is not provided, a default image name will be constructed in the form of `<acr-name>.azurecr.io/ado-task/container-app:<build-id>.<build-number>`.

imageToDeploy - Docker image to deploy

`string.`

The name of the image that has already been pushed to ACR and will be deployed to the Container App by this task. Note: the image name should include the ACR server; e.g., `<acr-name>.azurecr.io/<repo>:<tag>`. If this argument is not provided, the value provided (or determined) for the 'imageToBuild' argument will be used. If this image is found in an ACR instance that requires authentication to pull, the `acrName` argument, or the `acrUsername` and `acrPassword` arguments, can be provided to authenticate requests to the ACR instance.

containerAppName - Azure Container App name

`string.`

The name of the Azure Container App that will be created or updated. If not provided, this value will be in the form of `ado-task-app-<build-id>-<build-number>`.

resourceGroupName - Azure resource group name

`string.`

The existing resource group that the Azure Container App will be created in (or currently exists in). If not provided, this value will be in the form of <container-app-name>-rg.

containerAppEnvironment - Azure Container App environment

`string`.

The name of the Azure Container App environment to use with the application. If not provided, an existing environment in the resource group of the Container App will be used, otherwise, an environment will be created in the format of <container-app-name>-env.

runtimeStack - Application runtime stack

`string`.

The platform version stack used in the final runnable application image that is deployed to the Container App. The value should be provided in the formation <platform>:<version>. If not provided, this value is determined by Oryx based on the contents of the provided application. Please refer to [this document](#) for more information on supported runtime stacks for Oryx.

targetPort - Application target port

`string`.

The target port that the Container App will listen on. If not provided, this value will be "80" for Python applications and "8080" for all other supported platforms.

location - Location of the Container App

`string`.

The location that the Container App (and other created resources) will be deployed to.

environmentVariables - Environment variables

`string`.

A list of environment variable(s) for the container. Space-separated values in 'key=value' format. Empty string to clear existing values. Prefix value with 'secretref:' to reference a secret.

`ingress` - Ingress setting

`string`.

Possible options: external, internal, disabled. If set to `external` (default value if not provided when creating a Container App), the Container App will be visible from the internet or a VNET, depending on the app environment endpoint configured. If set to `internal`, the Container App will be visible from within the app environment only. If set to `disabled`, ingress will be disabled for this Container App and will not have an HTTP or TCP endpoint.

`yamlConfigPath` - YAML configuration file path

`string`.

Full path (on the executing Azure Pipelines agent) to the YAML file detailing the configuration of the Container App.

The `resourceGroup` property in the YAML configuration file *will not* be used; the value for this either comes from the `resourceGroup` argument provided to the task, or the default resource group name generated by the task. All other properties provided in the YAML configuration file will override the values provided as arguments to this task; for example, if the `containerAppName` argument is provided to the task, and the `name` property is set in the YAML configuration file, the `name` property in the YAML file will be used when creating or updating the Container App.

Image and application source arguments (*e.g.*, `appSourcePath`, `imageToDeploy`) will still be used to first build and/or push an image that is used by the Container App; in this case, the provided YAML configuration file will need to reference the image specified by `imageToDeploy` (or `imageToBuild`, depending on your scenario).

When creating a new Container App, all properties listed in the YAML configuration file (except `resourceGroup` as mentioned above) will be set when the Container App is created. When updating an existing Container App, only the properties listed in the file will be updated on the Container App.

Currently, the YAML file does not support setting up managed identity authentication for the container registry used; for more information on this issue, please see [this GitHub issue](#).

In cases where the `yamlConfigPath` argument is provided, the YAML file will be passed through to the corresponding `az containerapp` command, either `create` or `update` depending on your scenario. For more information on the intended behavior when the

YAML configuration file is provided, please see the documents linked for the corresponding commands.

For more information on the structure of the YAML configuration file, please visit [this site](#).

`disableTelemetry` - Disable telemetry

`boolean`.

If set to 'true', no telemetry will be collected by this Azure DevOps Task. If set to 'false', or if this argument is not provided, telemetry will be sent to Microsoft about the Container App build and deploy scenario targeted by this Azure DevOps Task.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

This Azure Pipelines Task allows users to easily deploy their application source to an [Azure Container App](#) in their Azure Pipelines workflow by either providing a previously built image, a Dockerfile that an image can be built from, or using a builder to create a runnable application image for the user.

The task has the following two usage patterns.

- **Pushing an image to ACR** - when pushing a new image to ACR, the `acrName` and `appSourcePath` task inputs are required.
- **Deploying a previously pushed image** - when deploying a previously pushed image, the `imageToDeploy` task input is required. If this image is found in an ACR instance that requires authentication to pull, the `acrName` argument, or the `acrUsername` and `acrPassword` arguments, can be provided to authenticate requests to the ACR instance.

Note

Although no task input is officially marked as "required" in the metadata of this task, some inputs will need to be provided in order for this task to successfully run using one of the two main usage patterns.

If no Dockerfile is found or provided in the provided application source, the following steps are performed by this task:

- Uses the Oryx++ Builder to build the application source using [Oryx](#) to produce a runnable application image
- Pushes this runnable application image to the provided Azure Container Registry
- Creates or updates a Container App based on this image

If a Dockerfile is found or discovered in the application source, the builder won't be used and the image will be built with a call to `docker build` and the Container App will be created or updated based on this image.

If a previously built image has already been pushed to the ACR instance and is provided to this task, no application source is required and the image will be used when creating or updating the Container App.

Running this task on Microsoft-hosted agents

If you are running this task on a [Microsoft-hosted agent](#), you may find that this task is *not* able to run successfully with the following operating systems:

- macOS
 - The [macOS runners](#) provided by Microsoft do not come installed with Docker (more information [here](#)); as a result, this task is not able to run any `docker` commands, such as pushing the built runnable application images to ACR.
- Windows
 - The [Windows runners](#) provided by Microsoft comes with Docker installed, but by default, Linux-based images are unable to be pulled down; as a result, this task is not able to pull down the Oryx builder to create runnable application images from provided application source.

Please see the below Docker prerequisite section for more information.

Data/Telemetry Collection Notice

By default, this Azure DevOps Task collects the following pieces of data for Microsoft:

- The Container App build and deploy scenario targeted by the user
 - *i.e.*, used the Oryx++ Builder, used a provided/found Dockerfile, or provided a previously built image
 - *Note:* the image name is *not* collected
- The processing time of the task, in milliseconds
- The result of the task
 - *i.e.*, succeeded or failed
- If the Oryx++ Builder is used, events and metrics relating to building the provided application using Oryx

If you want to disable data collection, please set the `disableTelemetry` argument to `true`.

Prerequisites

Prior to running this task, Azure resources and an Azure DevOps service connection are either required or optional depending on the arguments provided to this task.

Azure DevOps Service Connection

To deploy to Azure, an Azure subscription has to be linked to Team Foundation Server or to Azure Pipelines using the Services tab in the settings section. Add the Azure subscription to use in the Build or Release Management definition by opening the Account Administration screen (gear icon on the top-right of the screen) and then click on the Services Tab.

Create the [ARM](#) service endpoint and use the 'Azure Resource Manager' endpoint type; for more information on creating service connections, please follow [this document](#).

Azure CLI

This task requires that the Azure CLI is installed on the Azure Pipelines agent to execute a variety of commands throughout the execution of the task. For more information on how to install the Azure CLI on the agent, please see [this document](#). If an agent is already running on the machine on which the Azure CLI is installed, ensure that you restart the agent so that all relevant environment variables are updated.

Docker

This task requires that Docker is installed on the Azure Pipelines agent to push images to the provided Azure Container Registry. For more information on how to install Docker on the agent, please see [this document](#).

In addition, users running this task with a Windows agent may encounter an issue with not being able to pull down Linux-based images; to resolve this, please visit [this site](#) or located the `DockerCli.exe` file on your agent (typically in the `Program Files\ Docker\ Docker` folder) and run

```
& `.\DockerCli.exe` -SwitchDaemon
```

If Docker is not installed on the agent running this task, the following scenario(s) are still enabled:

- Providing a *previously built* image to the `imageToDeploy` argument that the Container App deploys with

If Docker is on the agent, but unable to work with Linux-based images, the following scenario(s) are still enabled:

- Providing a *previously built* image to the `imageToDeploy` argument that the Container App deploys with
- Providing a `Dockerfile` as a part of your application source that will be built and deployed with the Container App
 - *Note:* the `Dockerfile` cannot have any Linux-based image layers

pack CLI

The [pack CLI](#) is maintained by the Cloud Native Buildpacks project and is used by this task to create runnable application images for the user when the application source code is provided and no additional Dockerfile is provided or found. A [builder](#) was created by Oryx to take in the application source code provided to this task and produce an image that could then be pushed to an image registry and used within a Container App to build and run the application.

A stable version of the pack CLI is installed on the Azure Pipelines agent executing the task, and depending on the base OS of this agent, different tools will be leverage to assist with the installation:

- On Windows runners:
 - A set of PowerShell commands are executed to do the following:

- Creates a `pack` folder in the agent's temporary folder, if the `pack` folder doesn't already exist
- Downloads the pack CLI `.zip` into this `pack` folder
- Unzips the content from this `.zip` and places them in the `pack` folder
- Deletes the `.zip`
- On non-Windows runners:
 - `curl` will be used to pull down the `.tgz` containing the `pack` executable
 - `tar` will be used to unzip the `.tgz` and place the `pack` executable in `/usr/local/bin`

Azure Container Registry

An [Azure Container Registry](#) must exist that the user is able to push container images to. This task will leverage the Azure Container Registry to either push a built runnable application image to and/or deploy a Container App from.

The name of the Azure Container Registry is required via the `acrName` argument.

The user can also provide values for the `acrUsername` and `acrPassword` arguments that will authenticate calls to the Azure Container Registry instance; if not provided, an access token will be generated via the Azure CLI that will authenticate the calls instead.

Azure Container App environment

An [Azure Container App environment](#) is recommended to have been previously created by the user to improve the performance of the task. If no environment has been created before, or if an environment cannot be found in the resource group that is being used to host the created Container App, then an environment will be created by as a part of the `az containerapp up` command, which may take additional time.

Examples

The following examples outline how to use the `AzureContainerApps` in different scenarios.

Minimal - Build application image for Container App

```
yml
```

```
steps:
- task: AzureContainerApps@1
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    appSourcePath: '$(System.DefaultWorkingDirectory)'
    acrName: 'mytestacr'
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a new resource group named `<container-app-name>-rg`. The Container App will be based off of an image that was built from the provided `appSourcePath` and pushed to the provided ACR instance. An access token will be generated to authenticate the push to the provided ACR instance.

Minimal - Use previously published image for Container App

yml

```
steps:
- task: AzureContainerApps@1
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    imageToDeploy: mcr.microsoft.com/azuredocs/containerapps-
      helloworld:latest
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a new resource group named `<container-app-name>-rg` where **no new image is built**, but an existing image named `mcr.microsoft.com/azuredocs/containerapps-
 helloworld:latest` will be used for the Container App.

Minimal - Use YAML configuration file with previously published image for Container App

yml

```
steps:
- task: AzureContainerApps@1
  displayName: Build and deploy Container App
  inputs:
    yamlConfigPath: simple-image-container-app.yaml
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a new resource group named `<container-app-name>-rg` where **no new image is built**, but an existing image named `mcr.microsoft.com/azuredocs/containerapps-helloworld:latest` will be used for the Container App. Additional properties about the Container App will be pulled from the `simple-image-container-app.yaml` file and will override any additional values that would've been provided to the task as arguments **excluding `resourceGroup`**.

The `simple-image-container-app.yaml` file has the following structure:

```
yml

properties:
  managedEnvironmentId:
    /subscriptions/SUBSCRIPTION_ID/resourceGroup/RESOURCE_GROUP/providers/Microsoft.App/managedEnvironments/CONTAINER_APP_ENVIRONMENT
  configuration:
    ingress:
      external: true
      allowInsecure: false
      targetPort: 80
    template:
      containers:
        - image: mcr.microsoft.com/azuredocs/containerapps-helloworld:latest
          name: mysampleimagecontainer
```

The values for `SUBSCRIPTION_ID`, `RESOURCE_GROUP` and `CONTAINER_APP_ENVIRONMENT` must be updated to point to the full resource ID of the **existing** Container App environment that the Container App will use.

Using ACR credentials to authenticate

```
yml

steps:
- task: AzureContainerApps@1
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    appSourcePath: '$(System.DefaultWorkingDirectory)'
    acrName: 'mytestacr'
    acrUsername: $(ACR_USERNAME_SECRET)
    acrPassword: $(ACR_PASSWORD_SECRET)
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a new resource group named `<container-app-name>-rg`. The Container App will be

based off of an image that was built from the provided `appSourcePath` and pushed to the provided ACR instance. The provided ACR credentials will be used to authenticate calls to the ACR instance.

Container App name provided

```
yml
```

```
steps:
- task: AzureContainerApps@1
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    appSourcePath: '$(System.DefaultWorkingDirectory)'
    acrName: 'mytestacr'
    containerAppName: 'my-test-container-app'
```

This will create a new Container App named `my-test-container-app` in a new resource group name `my-test-container-app-rg`.

Resource group provided

```
yml
```

```
steps:
- task: AzureContainerApps@1
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    appSourcePath: '$(System.DefaultWorkingDirectory)'
    acrName: 'mytestacr'
    resourceGroup: 'my-test-rg'
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a resource group named `my-test-rg`. If the `my-test-rg` resource group does not exist, it will be created as a part of this task.

Container App name and resource group provided

```
yml
```

```
steps:
- task: AzureContainerApps@1
  displayName: Build and deploy Container App
  inputs:
```

```
connectedServiceNameARM: 'azure-subscription-service-connection'
appSourcePath: '$(System.DefaultWorkingDirectory)'
acrName: 'mytestacr'
containerAppName: 'my-test-container-app'
resourceGroup: 'my-test-rg'
```

This will create a new Container App named `my-test-container-app` in a resource group named `my-test-rg`. If the `my-test-rg` resource group does not exist, it will be created as a part of this task.

Container App environment provided

yml

```
steps:
- task: AzureContainerApps@1
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    appSourcePath: '$(System.DefaultWorkingDirectory)'
    acrName: 'mytestacr'
    containerAppEnvironment: 'my-test-container-app-env'
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a new resource group named `<container-app-name>-rg` with a new Container App environment named `my-test-container-app-env`.

Runtime stack provided

yml

```
steps:
- task: AzureContainerApps@1
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    appSourcePath: '$(System.DefaultWorkingDirectory)'
    acrName: 'mytestacr'
    runtimeStack: 'dotnetcore:7.0'
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a new resource group named `<container-app-name>-rg` where the runnable application image is using the .NET 7 runtime stack.

Dockerfile provided

```
yml

steps:
- task: AzureContainerApps@1
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    appSourcePath: '$(System.DefaultWorkingDirectory)'
    acrName: 'mytestacr'
    dockerfilePath: 'test.Dockerfile'
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a new resource group named `<container-app-name>-rg` where the runnable application image was created from the `test.Dockerfile` file found in the provided application source path directory.

Note: for values provided to `dockerfilePath`, no file prefixes should be included (e.g., `./test.Dockerfile` should be passed as just `test.Dockerfile`). The provided `appSourcePath` and `dockerfilePath` arguments will be concatenated inside of the task.

Image to build provided

```
yml

steps:
- task: AzureContainerApps@1
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    appSourcePath: '$(System.DefaultWorkingDirectory)'
    acrName: 'mytestacr'
    imageToBuild: 'mytestacr.azurecr.io/app:latest'
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a new resource group named `<container-app-name>-rg` where the image built and pushed to ACR is named `mytestacr.azurecr.io/app:latest`.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release

Requirement	Description
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Deploy

See also

- [Deploy to Azure Container Apps from Azure Pipelines](#)

AzureContainerApps@0 - Azure Container Apps Deploy v0 task

Article • 09/26/2023

An Azure DevOps Task to build and deploy Azure Container Apps.

Syntax

YAML

```
# Azure Container Apps Deploy v0
# An Azure DevOps Task to build and deploy Azure Container Apps.
- task: AzureContainerApps@0
  inputs:
    # advanced
    #workingDirectory: # string. Alias: cwd. Working Directory.
    #appSourcePath: # string. Application source path.
    azureSubscription: # string. Alias: connectedServiceNameARM. Required.
      Azure Resource Manager connection.
    #acrName: # string. Azure Container Registry name.
    #acrUsername: # string. Azure Container Registry username.
    #acrPassword: # string. Azure Container Registry password.
    #dockerfilePath: # string. Dockerfile path.
    #imageToBuild: # string. Docker image to build.
    #imageToDeploy: # string. Docker image to deploy.
    #containerAppName: # string. Azure Container App name.
    #resourceGroup: # string. Azure resource group name.
    #containerAppEnvironment: # string. Azure Container App environment.
    #runtimeStack: # string. Application runtime stack.
    #targetPort: # string. Application target port.
    #location: # string. Location of the Container App.
    #environmentVariables: # string. Environment variables.
    #ingress: # string. Ingress setting.
    #yamlConfigPath: # string. YAML configuration file path.
    #disableTelemetry: # boolean. Disable telemetry.
```

Inputs

`workingDirectory` - Working Directory

Input alias: `cwd`. `string`.

Current working directory where the script is run. Empty is the root of the repo (build) or artifacts (release), which is `$(System.DefaultWorkingDirectory)`.

appSourcePath - Application source path`string.`

Absolute path on the runner of the source application code to be built. If not provided, the 'imageToDeploy' argument must be provided to ensure the Container App has an image to reference.

When pushing a new image to ACR, the `acrName` and `appSourcePath` task inputs are required.

azureSubscription - Azure Resource Manager connection

Input alias: `connectedServiceNameARM`. `string`. Required.

Specify an Azure Resource Manager service connection for the deployment. This service connection must be linked to the user's Azure Subscription where the Container App will be created/updated. This service connection *must* have proper permissions to make these changes within the subscription, for example Contributor role.

acrName - Azure Container Registry name`string.`

The name of the Azure Container Registry that the runnable application image will be pushed to.

When pushing a new image to ACR, the `acrName` and `appSourcePath` task inputs are required.

acrUsername - Azure Container Registry username`string.`

The username used to authenticate push requests to the provided Azure Container Registry. If not provided, an access token will be generated via 'az acr login' and provided to 'docker login' to authenticate the requests.

acrPassword - Azure Container Registry password`string.`

The password used to authenticate push requests to the provided Azure Container Registry. If not provided, an access token will be generated via 'az acr login' and provided to 'docker login' to authenticate the requests.

dockerfilePath - Dockerfile path

`string.`

Relative path (_without file prefixes (see the following [Examples](#)) to the Dockerfile in the provided application source that should be used to build the image that is then pushed to ACR and deployed to the Container App. If not provided, this task will check if there is a file named 'Dockerfile' at the root of the provided application source and use that to build the image. Otherwise, the Oryx++ Builder will be used to create the image.

imageToBuild - Docker image to build

`string.`

The custom name of the image that is to be built, pushed to ACR and deployed to the Container App by this task. Note: this image name should include the ACR server; e.g., `<acr-name>.azurecr.io/<repo>:<tag>`. If this argument is not provided, a default image name will be constructed in the form of `<acr-name>.azurecr.io/ado-task/container-app:<build-id>.<build-number>`.

imageToDeploy - Docker image to deploy

`string.`

The name of the image that has already been pushed to ACR and will be deployed to the Container App by this task. Note: the image name should include the ACR server; e.g., `<acr-name>.azurecr.io/<repo>:<tag>`. If this argument is not provided, the value provided (or determined) for the 'imageToBuild' argument will be used. If this image is found in an ACR instance that requires authentication to pull, the `acrName` argument, or the `acrUsername` and `acrPassword` arguments, can be provided to authenticate requests to the ACR instance.

containerAppName - Azure Container App name

`string.`

The name of the Azure Container App that will be created or updated. If not provided, this value will be in the form of `ado-task-app-<build-id>-<build-number>`.

resourceGroupName - Azure resource group name

`string.`

The resource group that the Azure Container App will be created in (or currently exists in). If not provided, this value will be in the form of <container-app-name>-rg.

containerAppEnvironment - Azure Container App environment

`string`.

The name of the Azure Container App environment to use with the application. If not provided, an existing environment in the resource group of the Container App will be used, otherwise, an environment will be created in the format of <container-app-name>-env.

runtimeStack - Application runtime stack

`string`.

The platform version stack used in the final runnable application image that is deployed to the Container App. The value should be provided in the formation <platform>:<version>. If not provided, this value is determined by Oryx based on the contents of the provided application. Please refer to [this document](#) for more information on supported runtime stacks for Oryx.

targetPort - Application target port

`string`.

The target port that the Container App will listen on. If not provided, this value will be "80" for Python applications and "8080" for all other supported platforms.

location - Location of the Container App

`string`.

The location that the Container App (and other created resources) will be deployed to.

environmentVariables - Environment variables

`string`.

A list of environment variable(s) for the container. Space-separated values in 'key=value' format. Empty string to clear existing values. Prefix value with 'secretref:' to reference a secret.

`ingress` - Ingress setting

`string`.

Possible options: external, internal, disabled. If set to `external` (default value if not provided when creating a Container App), the Container App will be visible from the internet or a VNET, depending on the app environment endpoint configured. If set to `internal`, the Container App will be visible from within the app environment only. If set to `disabled`, ingress will be disabled for this Container App and will not have an HTTP or TCP endpoint.

`yamlConfigPath` - YAML configuration file path

`string`.

Full path (on the executing Azure Pipelines agent) to the YAML file detailing the configuration of the Container App.

The `resourceGroup` property in the YAML configuration file *will not* be used; the value for this either comes from the `resourceGroup` argument provided to the task, or the default resource group name generated by the task. All other properties provided in the YAML configuration file will override the values provided as arguments to this task; for example, if the `containerAppName` argument is provided to the task, and the `name` property is set in the YAML configuration file, the `name` property in the YAML file will be used when creating or updating the Container App.

Image and application source arguments (e.g., `appSourcePath`, `imageToDeploy`) will still be used to first build and/or push an image that is used by the Container App; in this case, the provided YAML configuration file will need to reference the image specified by `imageToDeploy` (or `imageToBuild`, depending on your scenario).

When creating a new Container App, all properties listed in the YAML configuration file (except `resourceGroup` as mentioned above) will be set when the Container App is created. When updating an existing Container App, only the properties listed in the file will be updated on the Container App.

Currently, the YAML file does not support setting up managed identity authentication for the container registry used; for more information on this issue, please see [this GitHub issue](#).

In cases where the `yamlConfigPath` argument is provided, the YAML file will be passed through to the corresponding `az containerapp` command, either `create` or `update` depending on your scenario. For more information on the intended behavior when the

YAML configuration file is provided, please see the documents linked for the corresponding commands.

For more information on the structure of the YAML configuration file, please visit [this site](#).

`disableTelemetry` - Disable telemetry

`boolean`.

If set to 'true', no telemetry will be collected by this Azure DevOps Task. If set to 'false', or if this argument is not provided, telemetry will be sent to Microsoft about the Container App build and deploy scenario targeted by this Azure DevOps Task.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

This Azure Pipelines Task allows users to easily deploy their application source to an [Azure Container App](#) in their Azure Pipelines workflow by either providing a previously built image, a Dockerfile that an image can be built from, or using a builder to create a runnable application image for the user.

The task has the following two usage patterns.

- **Pushing an image to ACR** - when pushing a new image to ACR, the `acrName` and `appSourcePath` task inputs are required.
- **Deploying a previously pushed image** - when deploying a previously pushed image, the `imageToDeploy` task input is required. If this image is found in an ACR instance that requires authentication to pull, the `acrName` argument, or the `acrUsername` and `acrPassword` arguments, can be provided to authenticate requests to the ACR instance.

Note

Although no task input is officially marked as "required" in the metadata of this task, some inputs will need to be provided in order for this task to successfully run using one of the two main usage patterns.

If no Dockerfile is found or provided in the provided application source, the following steps are performed by this task:

- Uses the Oryx++ Builder to build the application source using [Oryx](#) to produce a runnable application image
- Pushes this runnable application image to the provided Azure Container Registry
- Creates or updates a Container App based on this image

If a Dockerfile is found or discovered in the application source, the builder won't be used and the image will be built with a call to `docker build` and the Container App will be created or updated based on this image.

If a previously built image has already been pushed to the ACR instance and is provided to this task, no application source is required and the image will be used when creating or updating the Container App.

Running this task on Microsoft-hosted agents

If you are running this task on a [Microsoft-hosted agent](#), you may find that this task is *not* able to run successfully with the following operating systems:

- macOS
 - The [macOS runners](#) provided by Microsoft do not come installed with Docker (more information [here](#)); as a result, this task is not able to run any `docker` commands, such as pushing the built runnable application images to ACR.
- Windows
 - The [Windows runners](#) provided by Microsoft comes with Docker installed, but by default, Linux-based images are unable to be pulled down; as a result, this task is not able to pull down the Oryx builder to create runnable application images from provided application source.

Please see the below Docker prerequisite section for more information.

Data/Telemetry Collection Notice

By default, this Azure DevOps Task collects the following pieces of data for Microsoft:

- The Container App build and deploy scenario targeted by the user
 - *i.e.*, used the Oryx++ Builder, used a provided/found Dockerfile, or provided a previously built image
 - *Note:* the image name is *not* collected
- The processing time of the task, in milliseconds
- The result of the task
 - *i.e.*, succeeded or failed
- If the Oryx++ Builder is used, events and metrics relating to building the provided application using Oryx

If you want to disable data collection, please set the `disableTelemetry` argument to `true`.

Prerequisites

Prior to running this task, Azure resources and an Azure DevOps service connection are either required or optional depending on the arguments provided to this task.

Azure DevOps Service Connection

To deploy to Azure, an Azure subscription has to be linked to Team Foundation Server or to Azure Pipelines using the Services tab in the settings section. Add the Azure subscription to use in the Build or Release Management definition by opening the Account Administration screen (gear icon on the top-right of the screen) and then click on the Services Tab.

Create the [ARM](#) service endpoint and use the 'Azure Resource Manager' endpoint type; for more information on creating service connections, please follow [this document](#).

Azure CLI

This task requires that the Azure CLI is installed on the Azure Pipelines agent to execute a variety of commands throughout the execution of the task. For more information on how to install the Azure CLI on the agent, please see [this document](#). If an agent is already running on the machine on which the Azure CLI is installed, ensure that you restart the agent so that all relevant environment variables are updated.

Docker

This task requires that Docker is installed on the Azure Pipelines agent to push images to the provided Azure Container Registry. For more information on how to install Docker on the agent, please see [this document](#).

In addition, users running this task with a Windows agent may encounter an issue with not being able to pull down Linux-based images; to resolve this, please visit [this site](#) or located the `DockerCli.exe` file on your agent (typically in the `Program Files\ Docker\ Docker` folder) and run

```
& `.\DockerCli.exe` -SwitchDaemon
```

If Docker is not installed on the agent running this task, the following scenario(s) are still enabled:

- Providing a *previously built* image to the `imageToDeploy` argument that the Container App deploys with

If Docker is on the agent, but unable to work with Linux-based images, the following scenario(s) are still enabled:

- Providing a *previously built* image to the `imageToDeploy` argument that the Container App deploys with
- Providing a `Dockerfile` as a part of your application source that will be built and deployed with the Container App
 - *Note:* the `Dockerfile` cannot have any Linux-based image layers

pack CLI

The [pack CLI](#) is maintained by the Cloud Native Buildpacks project and is used by this task to create runnable application images for the user when the application source code is provided and no additional Dockerfile is provided or found. A [builder](#) was created by Oryx to take in the application source code provided to this task and produce an image that could then be pushed to an image registry and used within a Container App to build and run the application.

A stable version of the pack CLI is installed on the Azure Pipelines agent executing the task, and depending on the base OS of this agent, different tools will be leverage to assist with the installation:

- On Windows runners:
 - A set of PowerShell commands are executed to do the following:

- Creates a `pack` folder in the agent's temporary folder, if the `pack` folder doesn't already exist
- Downloads the pack CLI `.zip` into this `pack` folder
- Unzips the content from this `.zip` and places them in the `pack` folder
- Deletes the `.zip`
- On non-Windows runners:
 - `curl` will be used to pull down the `.tgz` containing the `pack` executable
 - `tar` will be used to unzip the `.tgz` and place the `pack` executable in `/usr/local/bin`

Azure Container Registry

An [Azure Container Registry](#) must exist that the user is able to push container images to. This task will leverage the Azure Container Registry to either push a built runnable application image to and/or deploy a Container App from.

The name of the Azure Container Registry is required via the `acrName` argument.

The user can also provide values for the `acrUsername` and `acrPassword` arguments that will authenticate calls to the Azure Container Registry instance; if not provided, an access token will be generated via the Azure CLI that will authenticate the calls instead.

Azure Container App environment

An [Azure Container App environment](#) is recommended to have been previously created by the user to improve the performance of the task. If no environment has been created before, or if an environment cannot be found in the resource group that is being used to host the created Container App, then an environment will be created by as a part of the `az containerapp up` command, which may take additional time.

Examples

The following examples outline how to use the `AzureContainerApps` in different scenarios.

Minimal - Build application image for Container App

```
yml
```

```
steps:
- task: AzureContainerApps@0
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    appSourcePath: '$(System.DefaultWorkingDirectory)'
    acrName: 'mytestacr'
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a new resource group named `<container-app-name>-rg`. The Container App will be based off of an image that was built from the provided `appSourcePath` and pushed to the provided ACR instance. An access token will be generated to authenticate the push to the provided ACR instance.

Minimal - Use previously published image for Container App

yml

```
steps:
- task: AzureContainerApps@0
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    imageToDeploy: mcr.microsoft.com/azuredocs/containerapps-
      helloworld:latest
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a new resource group named `<container-app-name>-rg` where **no new image is built**, but an existing image named `mcr.microsoft.com/azuredocs/containerapps-
 helloworld:latest` will be used for the Container App.

Minimal - Use YAML configuration file with previously published image for Container App

yml

```
steps:
- task: AzureContainerApps@0
  displayName: Build and deploy Container App
  inputs:
    yamlConfigPath: simple-image-container-app.yaml
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a new resource group named `<container-app-name>-rg` where **no new image is built**, but an existing image named `mcr.microsoft.com/azuredocs/containerapps-helloworld:latest` will be used for the Container App. Additional properties about the Container App will be pulled from the `simple-image-container-app.yaml` file and will override any additional values that would've been provided to the task as arguments **excluding `resourceGroup`**.

The `simple-image-container-app.yaml` file has the following structure:

```
yml

properties:
  managedEnvironmentId:
    /subscriptions/SUBSCRIPTION_ID/resourceGroup/RESOURCE_GROUP/providers/Microsoft.App/managedEnvironments/CONTAINER_APP_ENVIRONMENT
  configuration:
    ingress:
      external: true
      allowInsecure: false
      targetPort: 80
    template:
      containers:
        - image: mcr.microsoft.com/azuredocs/containerapps-helloworld:latest
          name: mysampleimagecontainer
```

The values for `SUBSCRIPTION_ID`, `RESOURCE_GROUP` and `CONTAINER_APP_ENVIRONMENT` must be updated to point to the full resource ID of the **existing** Container App environment that the Container App will use.

Using ACR credentials to authenticate

```
yml

steps:
- task: AzureContainerApps@0
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    appSourcePath: '$(System.DefaultWorkingDirectory)'
    acrName: 'mytestacr'
    acrUsername: $(ACR_USERNAME_SECRET)
    acrPassword: $(ACR_PASSWORD_SECRET)
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a new resource group named `<container-app-name>-rg`. The Container App will be

based off of an image that was built from the provided `appSourcePath` and pushed to the provided ACR instance. The provided ACR credentials will be used to authenticate calls to the ACR instance.

Container App name provided

```
yml
```

```
steps:
- task: AzureContainerApps@0
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    appSourcePath: '$(System.DefaultWorkingDirectory)'
    acrName: 'mytestacr'
    containerAppName: 'my-test-container-app'
```

This will create a new Container App named `my-test-container-app` in a new resource group name `my-test-container-app-rg`.

Resource group provided

```
yml
```

```
steps:
- task: AzureContainerApps@0
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    appSourcePath: '$(System.DefaultWorkingDirectory)'
    acrName: 'mytestacr'
    resourceGroup: 'my-test-rg'
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a resource group named `my-test-rg`. If the `my-test-rg` resource group does not exist, it will be created as a part of this task.

Container App name and resource group provided

```
yml
```

```
steps:
- task: AzureContainerApps@0
  displayName: Build and deploy Container App
  inputs:
```

```
connectedServiceNameARM: 'azure-subscription-service-connection'
appSourcePath: '$(System.DefaultWorkingDirectory)'
acrName: 'mytestacr'
containerAppName: 'my-test-container-app'
resourceGroup: 'my-test-rg'
```

This will create a new Container App named `my-test-container-app` in a resource group named `my-test-rg`. If the `my-test-rg` resource group does not exist, it will be created as a part of this task.

Container App environment provided

yml

```
steps:
- task: AzureContainerApps@0
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    appSourcePath: '$(System.DefaultWorkingDirectory)'
    acrName: 'mytestacr'
    containerAppEnvironment: 'my-test-container-app-env'
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a new resource group named `<container-app-name>-rg` with a new Container App environment named `my-test-container-app-env`.

Runtime stack provided

yml

```
steps:
- task: AzureContainerApps@0
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    appSourcePath: '$(System.DefaultWorkingDirectory)'
    acrName: 'mytestacr'
    runtimeStack: 'dotnetcore:7.0'
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a new resource group named `<container-app-name>-rg` where the runnable application image is using the .NET 7 runtime stack.

Dockerfile provided

```
yml

steps:
- task: AzureContainerApps@0
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    appSourcePath: '$(System.DefaultWorkingDirectory)'
    acrName: 'mytestacr'
    dockerfilePath: 'test.Dockerfile'
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a new resource group named `<container-app-name>-rg` where the runnable application image was created from the `test.Dockerfile` file found in the provided application source path directory.

Note: for values provided to `dockerfilePath`, no file prefixes should be included (e.g., `./test.Dockerfile` should be passed as just `test.Dockerfile`). The provided `appSourcePath` and `dockerfilePath` arguments will be concatenated inside of the task.

Image to build provided

```
yml

steps:
- task: AzureContainerApps@0
  displayName: Build and deploy Container App
  inputs:
    connectedServiceNameARM: 'azure-subscription-service-connection'
    appSourcePath: '$(System.DefaultWorkingDirectory)'
    acrName: 'mytestacr'
    imageToBuild: 'mytestacr.azurecr.io/app:latest'
```

This will create a new Container App named `ado-task-app-<build-id>-<build-number>` in a new resource group named `<container-app-name>-rg` where the image built and pushed to ACR is named `mytestacr.azurecr.io/app:latest`.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release

Requirement	Description
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Deploy

See also

- [Deploy to Azure Container Apps from Azure Pipelines](#)

AzureMysqlDeployment@1 - Azure Database for MySQL deployment v1 task

Article • 09/26/2023

Use this task to run your scripts and make changes to your database in Azure Database for MySQL. The Azure Database for MySQL Deployment task only works with [Azure Database for MySQL Single Server](#).

Syntax

YAML

```
# Azure Database for MySQL deployment v1
# Run your scripts and make changes to your Azure Database for MySQL.
- task: AzureMysqlDeployment@1
  inputs:
    azureSubscription: # string. Alias: ConnectedServiceName. Required.
    Azure Subscription.
    # DB Details
    ServerName: # string. Required. Host Name.
    #DatabaseName: # string. Database Name.
    SqlUsername: # string. Required. Server Admin Login.
    SqlPassword: # string. Required. Password.
    # Deployment Package
    #TaskNameSelector: 'SqlTaskFile' # 'SqlTaskFile' | 'InlineSqlTask'.
    Type. Default: SqlTaskFile.
    SqlFile: # string. Required when TaskNameSelector = SqlTaskFile. MySQL Script.
    #SqlInline: # string. Required when TaskNameSelector = InlineSqlTask.
    Inline MySQL Script.
    #SqlAdditionalArguments: # string. Additional MySQL Arguments.
    # Firewall
    IpDetectionMethod: 'AutoDetect' # 'AutoDetect' | 'IPAddressRange'.
    Required. Specify Firewall Rules Using. Default: AutoDetect.
    #StartIpAddress: # string. Required when IpDetectionMethod =
    IPAddressRange. Start IP Address.
    #EndIpAddress: # string. Required when IpDetectionMethod =
    IPAddressRange. End IP Address.
    #DeleteFirewallRule: true # boolean. Delete Rule After Task Ends.
    Default: true.
```

Inputs

`azureSubscription` - Azure Subscription

Input alias: `ConnectedServiceName`. `string`. Required.

This is needed to connect to your Azure account.

To configure a new service connection, select the Azure subscription from the list and click **Authorize**.

If your subscription is not listed or if you want to use an existing Service Principal, you can setup an Azure service connection using the **Add** or **Manage** buttons.

ServerName - Host Name

string. Required.

The name of your Azure Database for MySQL server.

Example: `fabrikam.mysql.database.azure.com`

The server name is provided in the Azure portal on the 'Overview' blade of your Azure Database for MySQL server resource.

When you connect using MySQL Workbench, this is the same value that is used for **Hostname** in **Parameters**.

DatabaseName - Database Name

string.

Optional. The name of the database. The script will create a database name if one does not exist.

If not specified, ensure that the database is referenced in the supplied SQL file or inline SQL, where needed.

Note: MySQL database names are case-sensitive.

SqlUsername - Server Admin Login

string. Required.

The Azure Database for MySQL server supports native MySQL authentication. You can connect and authenticate to a server with the server's admin login. Example:

`bbo1@fabrikam`.

When you connect using MySQL Workbench, this is the same value that is used for **Username** in **Parameters**.

SqlPassword - Password

`string`. Required.

The administrator password for Azure Database for MySQL. In case you don't recall the password, you can change the password from [Azure portal](#).

This string can be defined with a variable in the pipeline. Example: `$(password)`.

Also, you may mark the variable type as `secret` to secure it.

TaskNameSelector - Type

`string`. Allowed values: `SqlTaskFile` (MySQL Script File), `InlineSqlTask` (Inline MySQL Script). Default value: `SqlTaskFile`.

Optional. Selects one of the options between Script File & Inline Script.

- `SqlTaskFile` (default), for use with the `SqlFile` argument
- `InlineSqlTask`, for use with the `SqlInline` argument.

Note: these values are case-sensitive.

SqlFile - MySQL Script

`string`. Required when `TaskNameSelector = SqlTaskFile`.

The full path of the script file on the automation agent or on a UNC path accessible to the automation agent. For example: `\BudgetIT\DeployBuilds\script.sql`.

Predefined system variables, such as `$(agent.releaseDirectory)`, and files containing SQL statements can be used here.

Note: The MySQL client prefers Unix style paths, so from version 1.183.0 on, the task will convert Windows style paths to Unix style paths. Example: from `c:\foo\bar\myscript.sql` to `c:/foo/bar/myscript.sql`.

When the task is used on Linux platforms, paths remain unchanged. There is no need to escape special characters in paths.

SqlInline - Inline MySQL Script

`string`. Required when `TaskNameSelector = InlineSqlTask`.

Enters the MySQL script to execute on the database selected above.

SqlAdditionalArguments - Additional MySQL Arguments

`string`.

Optional. The additional options supported by the MySQL client. These options are applied when executing the given file on the Azure Database for MySQL.

Example: You can change to the default tab separated output format, to HTML, or even to the XML format. Other examples include:

- `--comments` to strip comments sent from the client to the server.
- `--quick` to prevent result caching.
- `--xml` to output results as XML.

All available options are described in the MySQL client documentation.

IpDetectionMethod - Specify Firewall Rules Using

`string`. Required. Allowed values: `AutoDetect`, `IPAddressRange`. Default value: `AutoDetect`.

For the successful execution of the task, we need to enable administrators to access the Azure Database for MySQL Server from the IP Address of the automation agent.

By selecting auto-detect, you can automatically add a firewall exception for the range of possible IP addresses of automation agents, or you can explicitly specify the range.

Accepted values:

- `AutoDetect` to auto-detect the automation agent's public IP address.
- `IPAddressRange` to explicitly specify the IP address range to configure. Set the IP address range using the `StartIpAddress` and `EndIpAddress` parameters.

Note: These values are case-sensitive.

StartIpAddress - Start IP Address

`string`. Required when `IpDetectionMethod = IPAddressRange`.

The starting IP Address of the automation agent machine pool. For example:

`196.21.30.50`.

EndIpAddress - End IP Address

`string`. Required when `IpDetectionMethod = IPAddressRange`.

The ending IP Address of the automation agent machine pool. For example:

`196.21.30.65`.

DeleteFirewallRule - Delete Rule After Task Ends

`boolean`. Default value: `true`.

Optional. If selected, the added exception for the IP addresses of the automation agent will be removed for the corresponding Azure Database for MySQL.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to run your scripts and make changes to your database in Azure Database for MySQL. Note that this is a preview version. The Azure Database for MySQL Deployment task only works with [Azure Database for MySQL Single Server](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any

Requirement	Description
Settable variables	Any
Agent version	1.100.0 or greater
Task category	Deploy

AzureFileCopy@5 - Azure file copy v5 task

Article • 09/26/2023

Copy files to Azure Blob Storage or virtual machines.

ⓘ Note

This task does not support Azure Resource Manager authentication with workflow identity federation.

Syntax

YAML

```
# Azure file copy v5
# Copy files to Azure Blob Storage or virtual machines.
- task: AzureFileCopy@5
  inputs:
    SourcePath: # string. Required. Source.
    azureSubscription: # string. Alias: ConnectedServiceNameARM. Required.
    Azure Subscription.
    Destination: # 'AzureBlob' | 'AzureVMs'. Required. Destination Type.
    storage: # string. Alias: StorageAccountRM. Required. RM Storage
    Account.
    #ContainerName: # string. Required when Destination = AzureBlob.
    Container Name.
    #BlobPrefix: # string. Optional. Use when Destination = AzureBlob. Blob
    Prefix.
    #resourceGroup: # string. Alias: EnvironmentNameRM. Required when
    Destination = AzureVMs. Resource Group.
    #ResourceFilteringMethod: 'machineNames' # 'machineNames' | 'tags'.
    Optional. Use when Destination = AzureVMs. Select Machines By. Default:
    machineNames.
    #MachineNames: # string. Optional. Use when Destination = AzureVMs.
    Filter Criteria.
    #vmsAdminUserName: # string. Required when Destination = AzureVMs. Admin
    Login.
    #vmsAdminPassword: # string. Required when Destination = AzureVMs.
    Password.
    #TargetPath: # string. Required when Destination = AzureVMs. Destination
    Folder.
    #AdditionalArgumentsForBlobCopy: # string. Optional Arguments (for
    uploading files to blob).
    #AdditionalArgumentsForVMCopy: # string. Optional. Use when Destination
    = AzureVMs. Optional Arguments (for downloading files to VM).
```

```
#sasTokenTimeOutInMinutes: '240' # string. Optional. Use when
Destination = AzureBlob. SAS Token Expiration Period In Minutes. Default:
240.
#enableCopyPrerequisites: false # boolean. Optional. Use when
Destination = AzureVMs. Enable Copy Prerequisites. Default: false.
#CopyFilesInParallel: true # boolean. Optional. Use when Destination =
AzureVMs. Copy in Parallel. Default: true.
#CleanTargetBeforeCopy: false # boolean. Clean Target. Default: false.
#skipCACheck: true # boolean. Optional. Use when Destination = AzureVMs.
Test Certificate. Default: true.
```

Inputs

`SourcePath` - Source

`string`. Required.

The location of source files. Supported values include YAML Pipelines and Classic Release support [predefined system variables](#) like `Build.Repository.LocalPath`.

[Release variables](#) are supported only in classic releases. The wild card symbol (*) is supported anywhere in the file path or file name.

`azureSubscription` - Azure Subscription

Input alias: `ConnectedServiceNameARM`. `string`. Required.

Specify the name of an [Azure Resource Manager service connection](#) configured for the subscription where the target Azure service, virtual machine, or storage account is located. See [Azure Resource Manager overview](#) for more details.

`Destination` - Destination Type

`string`. Required. Allowed values: `AzureBlob` (Azure Blob), `AzureVMs` (Azure VMs).

Specify the destination type.

`storage` - RM Storage Account

Input alias: `StorageAccountRM`. `string`. Required.

Specify a pre-existing ARM storage account. This is the storage account used as an intermediary for copying files to Azure VMs.

ContainerName - Container Name

`string`. Required when `Destination = AzureBlob`.

The name of the container into which files are copied. If the specified container does not exist in the storage account, it will be created.

To create a virtual directory inside the container, use the blob prefix input. For example, for the target location `https://myaccount.blob.core.windows.net/mycontainer/vd1/vd2/`, specify container name `mycontainer` and blob prefix: `vd1/vd2`.

BlobPrefix - Blob Prefix

`string`. Optional. Use when `Destination = AzureBlob`.

Specify a prefix for the destination virtual directory within the Azure Blob container. This applies when the `SourcePath` contains a wildcard that may match multiple items.

Example: You can append a build number to prefix the files from all blobs with the same build number.

Example: If you specify a blob prefix `myvd1`, a virtual directory is created inside the container. Files are copied from the source to

`https://myaccount.blob.core.windows.net/mycontainer/myvd1/`.

In the case that the `SourcePath` is a single item with no wildcard, this blob prefix will function as the destination blob name.

resourceGroup - Resource Group

Input alias: `EnvironmentNameRM`. `string`. Required when `Destination = AzureVMs`.

Specify the name of the target Resource Group into which the files will be copied.

ResourceFilteringMethod - Select Machines By

`string`. Optional. Use when `Destination = AzureVMs`. Allowed values: `machineNames` (Machine Names), `tags`. Default value: `machineNames`.

Specify a VM host name or tag that identifies a subset of VMs in a resource group. `Tags` are supported for resources created via the Azure Resource Manager only.

MachineNames - Filter Criteria

`string`. Optional. Use when `Destination = AzureVMs`.

Provide a list of VM names or tag names that identify the VMs the task will target. Valid filter criteria includes:

- The name of an [Azure Resource Group](#).
- An output variable from a previous task.
- A comma-delimited list of tag names or VM names.
- Format VM names using a comma-separated list of FQDNs or IP addresses.
- Format tag names for a filter as `{TagName}:{Value}` Example: `Role:DB;OS:Win8.1`

vmsAdminUserName - Admin Login

`string`. Required when `Destination = AzureVMs`.

Provide the user name of an account with administrative permissions on all of the target VMs.

- Supported formats include: `username`, `domain\username`, `machine-name\username`, and `.\username`.
- UPN formats including `username@domain.com` and built-in system accounts such as `NT Authority\System` are not supported.

vmsAdminPassword - Password

`string`. Required when `Destination = AzureVMs`.

Provide the password for the `Admin Login` parameter.

To find the variable, locate the `Admin Login` parameter. Select the padlock icon for a variable defined in the `Variables` tab to protect the value and insert the variable name here.

TargetPath - Destination Folder

`string`. Required when `Destination = AzureVMs`.

Specify the path to the folder in the Azure VMs into which files will be copied.

Environment variables such as `$env:windir` and `$env:systemroot` are supported.

Examples: `$env:windir\FabrikamFiber\Web` and `c:\FabrikamFiber`

`AdditionalArgumentsForBlobCopy` - Optional Arguments (for uploading files to blob)

`string`.

Provide additional arguments to `AzCopy.exe` for use when uploading to the Blob and downloading to the VMs. See [Transfer data with the AzCopy Command-Line Utility](#) for details.

For Premium storage accounts that support only Azure page Blobs use `--blob-type=PageBlob` as an additional argument.

Default arguments include `--log-level=INFO` (default) and `--recursive` (if the container name is not `$root`).

`AdditionalArgumentsForVMCopy` - Optional Arguments (for downloading files to VM)

`string`. Optional. Use when `Destination = AzureVMs`.

Provide additional arguments to `AzCopy.exe` that will be applied when downloading to VMs such as, `--check-length=true`.

If no optional arguments are specified, the following are added by default:

- `--log-level=INFO`
- `--log-level=DEBUG` (If the pipeline is running in debug mode set)
- `--recursive`

`sasTokenTimeOutInMinutes` - SAS Token Expiration Period In Minutes

`string`. Optional. Use when `Destination = AzureBlob`. Default value: `240`.

Specify the time in minutes after which the SAS token for the container will expire. By default, this token expires after 4 hours.

`enableCopyPrerequisites` - Enable Copy Prerequisites

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `false`.

When enabled, this option uses a self-signed certificate to configure the Windows Remote Management (WinRM) listener over the HTTPS protocol on port 5986. This configuration is required for performing copy operations on Azure VMs. Applicable only for ARM VMs.

- If the target VMs are accessed through a load balancer, configure an inbound NAT rule to allow access on port 5986.
- If the target VMs are associated with a Network Security Group (NSG), configure an inbound security rule to allow access on port 5986.

CopyFilesInParallel - Copy in Parallel

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `true`.

Specify `true` to copy files in parallel to the target VMs.

CleanTargetBeforeCopy - Clean Target

`boolean`. Default value: `false`.

Specify `true` to clean-up the destination folder before copying files.

skipCACheck - Test Certificate

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `true`.

WinRM requires a certificate for the HTTPS transfer when copying files from the intermediate storage Blob into the Azure VMs.

If you use a self-signed certificate, specify `true` to prevent the process from validating the certificate with a trusted CA.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

StorageContainerUri

Uri of the container where the files were copied to. Valid only when the selected destination is Azure Blob.

StorageContainerSasToken

SasToken for the container where the files were copied to. Valid only when the selected destination is Azure Blob.

Remarks

AzureFileCopy@5 supports AzCopy.exe version 10.12.2.

ⓘ Note

This task is written in PowerShell and works **only** when run on Windows agents. If your pipelines require Linux agents and need to copy files to an Azure Storage Account, consider running `az storage blob` commands in the **Azure CLI task** as an alternative.

The task is used to copy application files and other artifacts that are required in order to install the app; such as PowerShell scripts, PowerShell-DSC modules, and more.

When the target is Azure VMs, the files are first copied to an automatically generated Azure blob container and then downloaded into the VMs. The container is deleted after the files are successfully copied to the VMs.

The task uses **AzCopy**, the command-line utility built for fast copying data from and to Azure storage accounts. Version 5 of the Azure File Copy task uses [AzCopy V10](#).

To dynamically deploy Azure Resource Groups that contain virtual machines, use the [Azure Resource Group Deployment](#) task. This task has a sample template that can perform the required operations to set up the WinRM HTTPS protocol on the VMs, open port 5986 in the firewall, and install the test certificate.

ⓘ Note

If you are deploying to Azure Static Websites as a container in Blob storage, use [Version 2](#) or higher of the task in order to preserve the `$web` container name.

The task supports authentication based on Azure Active Directory. Authentication using a service principal and managed identity are available. For managed identities, only system-wide managed identity is supported.

ⓘ Note

For authorization, provide access to the Security Principal. The level of authorization required is shown in [Option 1: Use Active Directory](#).

What are the Azure PowerShell prerequisites for using this task?

The task requires that Azure PowerShell is installed on the machine running the automation agent. The recommended version is 1.0.2, but the task will work with version 0.9.8 and higher. You can use the [Azure PowerShell Installer v1.0.2](#) to obtain this.

What are the WinRM prerequisites for this task?

The task uses Windows Remote Management (WinRM) HTTPS protocol to copy the files from the storage Blob container to the Azure VMs. This requires that the WinRM HTTPS service is configured on the VMs, and a suitable certificate is installed.

Configure WinRM after virtual machine creation

If the VMs were created without opening the WinRM HTTPS ports, perform the following:

1. Configure an inbound access rule to allow HTTPS on port 5986 of each VM.
2. Disable [UAC remote restrictions](#).
3. Specify the credentials for the task to access the VMs using an administrator-level login in the simple form **username** without any domain part.
4. Install a certificate on the machine that runs the automation agent.
5. If you are using a self-signed certificate, set the **Test Certificate** parameter of the task.

What type of service connection should I choose?

- For Azure Resource Manager storage accounts and Azure Resource Manager VMs, use an **Azure Resource Manager** service connection type. See [Automating Azure Resource Group deployment using a Service Principal](#).
- While using an **Azure Resource Manager** service connection type, the task automatically filters appropriate newer Azure Resource Manager storage accounts, and other fields. For example, the Resource Group or cloud service, and the VMs.

How do I create a school or work account for use with this task?

A suitable account can be created for use in a service connection:

1. Use the Azure portal to create a new user account in Azure Active Directory.
2. Add the Azure Active Directory user account to the co-administrators group in your Azure subscription.
3. Sign into the Azure portal with this user account and change the password.
4. Use the credentials of this account in the service connection. Deployments are then processed using this account.

If the task fails, will the copy resume?

Since AzCopy V10 does not support journal files, the task cannot resume the copy. You must run the task again to copy all the files.

Are the log files and plan files cleaned after the copy?

The log and plan files are not deleted by the task. To explicitly clean-up the files, add a CLI step in the workflow using [azcopy jobs clean](#).

How do I use the Azure file copy task to copy a file to an Azure virtual machine that doesn't have a public IP address?

Make sure that you're using version 5 of the Azure file copy task. If the task fails, you can add a build step to run the command `azcopy cp "source-file-path" "destination-file-path"` to substitute the source and destination values.

Forbidden error: 'AzCopy.exe exited with non-zero exit code while uploading files to blob storage' while using Azure File Copy task

The hosted agents are assigned randomly every time a build is triggered, the [agent IP addresses](#) will be different on every run. If these IP addresses are not in your allowed list of IPs, the communication between Azure DevOps and the storage account fails. In such scenarios, follow the steps outlined:

1. Add a build step using Azure CLI to identify the IP address of the Microsoft Hosted Build agent at runtime. It will add the IP address to the Network rule on the Azure Storage Account.
2. Run the build step for your Azure Storage Account.
3. Add another build step using Azure CLI to remove the IP address of the build agent from the Azure Storage Account network rule.

Examples

```
yml

trigger:
- main

pool:
  vmImage: windows-latest

steps:
- task: AzureFileCopy@5
  inputs:
    SourcePath: 'Readme.md'
    azureSubscription: 'MyAzureSubscription'
    Destination: 'AzureBlob'
    storage: 'MyStorage'
    ContainerName: 'MyContainerName'
    name: AzureFileCopy

- script: |
  echo $(AzureFileCopy.StorageContainerUri)
  echo $(AzureFileCopy.StorageContainerSasToken)
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: azureps
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any

Requirement	Description
Settable variables	Any
Agent version	1.103.0 or greater
Task category	Deploy

AzureFileCopy@4 - Azure file copy v4 task

Article • 09/26/2023

Copy files to Azure Blob Storage or virtual machines.

ⓘ Note

This task does not support Azure Resource Manager authentication with workflow identity federation.

Syntax

YAML

```
# Azure file copy v4
# Copy files to Azure Blob Storage or virtual machines.
- task: AzureFileCopy@4
  inputs:
    SourcePath: # string. Required. Source.
    azureSubscription: # string. Alias: ConnectedServiceNameARM. Required.
    Azure Subscription.
    Destination: # 'AzureBlob' | 'AzureVMs'. Required. Destination Type.
    storage: # string. Alias: StorageAccountRM. Required. RM Storage
    Account.
    #ContainerName: # string. Required when Destination = AzureBlob.
    Container Name.
    #BlobPrefix: # string. Optional. Use when Destination = AzureBlob. Blob
    Prefix.
    #resourceGroup: # string. Alias: EnvironmentNameRM. Required when
    Destination = AzureVMs. Resource Group.
    #ResourceFilteringMethod: 'machineNames' # 'machineNames' | 'tags'.
    Optional. Use when Destination = AzureVMs. Select Machines By. Default:
    machineNames.
    #MachineNames: # string. Optional. Use when Destination = AzureVMs.
    Filter Criteria.
    #vmsAdminUserName: # string. Required when Destination = AzureVMs. Admin
    Login.
    #vmsAdminPassword: # string. Required when Destination = AzureVMs.
    Password.
    #TargetPath: # string. Required when Destination = AzureVMs. Destination
    Folder.
    #AdditionalArgumentsForBlobCopy: # string. Optional Arguments (for
    uploading files to blob).
    #AdditionalArgumentsForVMCopy: # string. Optional. Use when Destination
    = AzureVMs. Optional Arguments (for downloading files to VM).
```

```
#sasTokenTimeOutInMinutes: '240' # string. Optional. Use when  
Destination = AzureBlob. SAS Token Expiration Period In Minutes. Default:  
240.  
#enableCopyPrerequisites: false # boolean. Optional. Use when  
Destination = AzureVMs. Enable Copy Prerequisites. Default: false.  
#CopyFilesInParallel: true # boolean. Optional. Use when Destination =  
AzureVMs. Copy in Parallel. Default: true.  
#CleanTargetBeforeCopy: false # boolean. Optional. Use when Destination =  
= AzureVMs. Clean Target. Default: false.  
#skipCACheck: true # boolean. Optional. Use when Destination = AzureVMs.  
Test Certificate. Default: true.
```

Inputs

SourcePath - Source

`string`. Required.

The location of source files. Supported values include YAML Pipelines and Classic Release support [predefined system variables](#) like `Build.Repository.LocalPath`.

[Release variables](#) are supported only in classic releases. The wild card symbol (*) is supported anywhere in the file path or file name.

azureSubscription - Azure Subscription

Input alias: `ConnectedServiceNameARM`. `string`. Required.

Specify the name of an [Azure Resource Manager service connection](#) configured for the subscription where the target Azure service, virtual machine, or storage account is located. See [Azure Resource Manager overview](#) for more details.

Destination - Destination Type

`string`. Required. Allowed values: `AzureBlob` (Azure Blob), `AzureVMs` (Azure VMs).

Specify the destination type.

storage - RM Storage Account

Input alias: `StorageAccountRM`. `string`. Required.

Specify a pre-existing ARM storage account. This is the storage account used as an intermediary for copying files to Azure VMs.

ContainerName - Container Name

`string`. Required when `Destination = AzureBlob`.

The name of the container into which files are copied. If the specified container does not exist in the storage account, it will be created.

To create a virtual directory inside the container, use the blob prefix input. For example, for the target location `https://myaccount.blob.core.windows.net/mycontainer/vd1/vd2/`, specify container name `mycontainer` and blob prefix: `vd1/vd2`.

BlobPrefix - Blob Prefix

`string`. Optional. Use when `Destination = AzureBlob`.

Specify a prefix that can be used to filter files.

Example: You can append a build number to filter the files from all blobs with the same build number.

Example: If you specify a blob prefix `myvd1`, a virtual directory is created inside the container. Files are copied from the source to

`https://myaccount.blob.core.windows.net/mycontainer/myvd1/`.

resourceGroup - Resource Group

Input alias: `EnvironmentNameRM`. `string`. Required when `Destination = AzureVMs`.

Specify the name of the target Resource Group into which the files will be copied.

ResourceFilteringMethod - Select Machines By

`string`. Optional. Use when `Destination = AzureVMs`. Allowed values: `machineNames` (Machine Names), `tags`. Default value: `machineNames`.

Specify a VM host name or tag that identifies a subset of VMs in a resource group. `Tags` are supported for resources created via the Azure Resource Manager only.

MachineNames - Filter Criteria

`string`. Optional. Use when `Destination = AzureVMs`.

Provide a list of VM names or tag names that identify the VMs the task will target. Valid filter criteria includes:

- The name of an [Azure Resource Group](#).
- An output variable from a previous task.
- A comma-delimited list of tag names or VM names.
- Format VM names using a comma-separated list of FQDNs or IP addresses.
- Format tag names for a filter as `{TagName}:{Value}` Example: `Role:DB;OS:Win8.1`

vmsAdminUserName - Admin Login

`string`. Required when `Destination = AzureVMs`.

Provide the user name of an account with administrative permissions on all of the target VMs.

- Supported formats include: `username`, `domain\username`, `machine-name\username`, and `.\username`.
- UPN formats including `username@domain.com` and built-in system accounts such as `NT Authority\System` are not supported.

vmsAdminPassword - Password

`string`. Required when `Destination = AzureVMs`.

Provide the password for the `Admin Login` parameter.

To find the variable, locate the `Admin Login` parameter. Select the padlock icon for a variable defined in the `Variables` tab to protect the value and insert the variable name here.

TargetPath - Destination Folder

`string`. Required when `Destination = AzureVMs`.

Specify the path to the folder in the Azure VMs into which files will be copied.

Environment variables such as `$env:windir` and `$env:systemroot` are supported.

Examples: `$env:windir\FabrikamFiber\Web` and `c:\FabrikamFiber`

AdditionalArgumentsForBlobCopy - Optional Arguments (for uploading files to blob)

`string`.

Provide additional arguments to `AzCopy.exe` for use when uploading to the Blob and downloading to the VMs. See [Transfer data with the AzCopy Command-Line Utility](#) for details.

For Premium storage accounts that support only Azure page Blobs use `--blob-type=PageBlob` as an additional argument.

Default arguments include `--log-level=INFO` (default) and `--recursive` (if the container name is not `$root`).

AdditionalArgumentsForVMCopy - Optional Arguments (for downloading files to VM)

`string`. Optional. Use when `Destination = AzureVMs`.

Provide additional arguments to `AzCopy.exe` that will be applied when downloading to VMs such as, `--check-length=true`.

If no optional arguments are specified, the following are added by default:

- `--log-level=INFO`
- `--log-level=DEBUG` (If the pipeline is running in debug mode set)
- `--recursive`

sasTokenTimeOutInMinutes - SAS Token Expiration Period In Minutes

`string`. Optional. Use when `Destination = AzureBlob`. Default value: `240`.

Specify the time in minutes after which the SAS token for the container will expire. By default, this token expires after 4 hours.

enableCopyPrerequisites - Enable Copy Prerequisites

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `false`.

When enabled, this option uses a self-signed certificate to configure the Windows Remote Management (WinRM) listener over the HTTPS protocol on port 5986. This configuration is required for performing copy operations on Azure VMs.

- If the target VMs are accessed through a load balancer, configure an inbound NAT rule to allow access on port 5986.
- If the target VMs are associated with a Network Security Group (NSG), configure an inbound security rule to allow access on port 5986.

CopyFilesInParallel - Copy in Parallel

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `true`.

Specify `true` to copy files in parallel to the target VMs.

CleanTargetBeforeCopy - Clean Target

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `false`.

Specify `true` to clean-up the destination folder before copying files.

skipCACheck - Test Certificate

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `true`.

WinRM requires a certificate for the HTTPS transfer when copying files from the intermediate storage Blob into the Azure VMs.

If you use a self-signed certificate, specify `true` to prevent the process from validating the certificate with a trusted CA.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

StorageContainerUri

URI of the container to which the files were copied. Valid only when the selected destination is an Azure Blob.

StorageContainerSasToken

SasToken for the container to which the files were copied. Valid only when the selected destination is an Azure Blob.

Remarks

AzureFileCopy@4 supports AzCopy.exe version 10.8.0.

ⓘ Note

This task is written in PowerShell and works **only** when run on Windows agents. If your pipelines require Linux agents and need to copy files to an Azure Storage Account, consider running `az storage blob` commands in the **Azure CLI task** as an alternative.

The task is used to copy application files and other artifacts that are required in order to install the app; such as PowerShell scripts, PowerShell-DSC modules, and more.

When the target is Azure VMs, the files are first copied to an automatically generated Azure blob container and then downloaded into the VMs. The container is deleted after the files are successfully copied to the VMs.

The task uses **AzCopy**, the command-line utility built for fast copying data from and to Azure storage accounts. Version 4 of the Azure File Copy task uses [AzCopy V10](#).

To dynamically deploy Azure Resource Groups that contain virtual machines, use the [Azure Resource Group Deployment](#) task. This task has a sample template that can perform the required operations to set up the WinRM HTTPS protocol on the VMs, open port 5986 in the firewall, and install the test certificate.

ⓘ Note

If you are deploying to Azure Static Websites as a container in Blob storage, use [Version 2 or higher](#) of the task in order to preserve the `$web` container name.

The task supports authentication based on Azure Active Directory. Authentication using a service principal and managed identity are available. For managed identities, only system-wide managed identity is supported.

ⓘ Note

For authorization, provide access to the Security Principal. The level of authorization required is shown in [Option 1: Use Active Directory](#).

What are the Azure PowerShell prerequisites for using this task?

The task requires that Azure PowerShell is installed on the machine running the automation agent. The recommended version is 1.0.2, but the task will work with version 0.9.8 and higher. You can use the [Azure PowerShell Installer v1.0.2](#) to obtain this.

What are the WinRM prerequisites for this task?

The task uses Windows Remote Management (WinRM) HTTPS protocol to copy the files from the storage Blob container to the Azure VMs. This requires that the WinRM HTTPS service is configured on the VMs, and a suitable certificate is installed.

Configure WinRM after virtual machine creation

If the VMs were created without opening the WinRM HTTPS ports, perform the following:

1. Configure an inbound access rule to allow HTTPS on port 5986 of each VM.
2. Disable [UAC remote restrictions](#).
3. Specify the credentials for the task to access the VMs using an administrator-level login in the simple form **username** without any domain part.
4. Install a certificate on the machine that runs the automation agent.
5. If you are using a self-signed certificate, set the **Test Certificate** parameter of the task.

What type of service connection should I choose?

- For Azure Resource Manager storage accounts and Azure Resource Manager VMs, use an **Azure Resource Manager** service connection type. See [Automating Azure Resource Group deployment using a Service Principal](#).
- While using an **Azure Resource Manager** service connection type, the task automatically filters appropriate newer Azure Resource Manager storage accounts, and other fields. For example, the Resource Group or cloud service, and the VMs.

How do I create a school or work account for use with this task?

A suitable account can be created for use in a service connection:

1. Use the Azure portal to create a new user account in Azure Active Directory.
2. Add the Azure Active Directory user account to the co-administrators group in your Azure subscription.
3. Sign into the Azure portal with this user account and change the password.
4. Use the credentials of this account in the service connection. Deployments are then processed using this account.

If the task fails, will the copy resume?

Since AzCopy V10 does not support journal files, the task cannot resume the copy. You must run the task again to copy all the files.

Are the log files and plan files cleaned after the copy?

The log and plan files are not deleted by the task. To explicitly clean-up the files, add a CLI step in the workflow using [azcopy jobs clean](#).

How do I use the Azure file copy task to copy a file to an Azure virtual machine that doesn't have a public IP address?

Make sure that you're using version 4 of the Azure file copy task. If the task fails, you can add a build step to run the command `azcopy cp "source-file-path" "destination-file-path"` to substitute the source and destination values.

Forbidden error: 'AzCopy.exe exited with non-zero exit code while uploading files to blob storage' while using Azure File Copy task

The hosted agents are assigned randomly every time a build is triggered, the [agent IP addresses](#) will be different on every run. If these IP addresses are not in your allowed list of IPs, the communication between Azure DevOps and the storage account fails. In such scenarios, follow the steps outlined:

1. Add a build step using Azure CLI to identify the IP address of the Microsoft Hosted Build agent at runtime. It will add the IP address to the Network rule on the Azure Storage Account.
2. Run the build step for your Azure Storage Account.
3. Add another build step using Azure CLI to remove the IP address of the build agent from the Azure Storage Account network rule.

Examples

yml

```
- task: AzureFileCopy@4
  inputs:
    SourcePath: 'Readme.md'
    azureSubscription: 'Azure'
    Destination: 'AzureBlob'
    storage: 'storageAccount'
    ContainerName: 'containerName'
    BlobPrefix: ''
    name: AzureFileCopy

- script: |
  echo $(AzureFileCopy.StorageContainerUri)
  echo $(AzureFileCopy.StorageContainerSasToken)
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: azureps
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.103.0 or greater
Task category	Deploy

AzureFileCopy@3 - Azure file copy v3 task

Article • 09/26/2023

Copy files to Azure Blob Storage or virtual machines.

ⓘ Note

This task does not support Azure Resource Manager authentication with workflow identity federation.

Syntax

YAML

```
# Azure file copy v3
# Copy files to Azure Blob Storage or virtual machines.
- task: AzureFileCopy@3
  inputs:
    SourcePath: # string. Required. Source.
    azureSubscription: # string. Alias: ConnectedServiceNameARM. Required.
    Azure Subscription.
    Destination: # 'AzureBlob' | 'AzureVMs'. Required. Destination Type.
    storage: # string. Alias: StorageAccountRM. Required. RM Storage
    Account.
    #ContainerName: # string. Required when Destination = AzureBlob.
    Container Name.
    #BlobPrefix: # string. Optional. Use when Destination = AzureBlob. Blob
    Prefix.
    #resourceGroup: # string. Alias: EnvironmentNameRM. Required when
    Destination = AzureVMs. Resource Group.
    #ResourceFilteringMethod: 'machineNames' # 'machineNames' | 'tags'.
    Optional. Use when Destination = AzureVMs. Select Machines By. Default:
    machineNames.
    #MachineNames: # string. Optional. Use when Destination = AzureVMs.
    Filter Criteria.
    #vmsAdminUserName: # string. Required when Destination = AzureVMs. Admin
    Login.
    #vmsAdminPassword: # string. Required when Destination = AzureVMs.
    Password.
    #TargetPath: # string. Required when Destination = AzureVMs. Destination
    Folder.
    #AdditionalArgumentsForBlobCopy: # string. Optional Arguments (for
    uploading files to blob).
    #AdditionalArgumentsForVMCopy: # string. Optional. Use when Destination
    = AzureVMs. Optional Arguments (for downloading files to VM).
```

```
#enableCopyPrerequisites: false # boolean. Optional. Use when Destination = AzureVMs. Enable Copy Prerequisites. Default: false.  
#CopyFilesInParallel: true # boolean. Optional. Use when Destination = AzureVMs. Copy in Parallel. Default: true.  
#CleanTargetBeforeCopy: false # boolean. Optional. Use when Destination = AzureVMs. Clean Target. Default: false.  
#skipCACheck: true # boolean. Optional. Use when Destination = AzureVMs. Test Certificate. Default: true.  
# Output  
#outputStorageUri: # string. Storage Container URI.  
#outputStorageContainerSasToken: # string. Storage Container SAS Token.  
#sasTokenTimeOutInMinutes: # string. SAS Token Expiration Period In Minutes.
```

Inputs

SourcePath - Source

`string`. Required.

Specify the absolute path of the source folder, or file on the local machine, or a UNC share. You can use pre-defined system variables such as `$(Build.Repository.LocalPath)`. Names containing wildcards such as `*.zip` are not supported. The value or expression you specify should return a single folder or a file name.

azureSubscription - Azure Subscription

Input alias: `ConnectedServiceNameARM`. `string`. Required.

Specify the name of an [Azure Resource Manager service connection](#) configured for the subscription where the target Azure service, virtual machine, or storage account is located. See [Azure Resource Manager overview](#) for more details.

Destination - Destination Type

`string`. Required. Allowed values: `AzureBlob` (Azure Blob), `AzureVMs` (Azure VMs).

Specify the destination type.

storage - RM Storage Account

Input alias: `StorageAccountRM`. `string`. Required.

Specify a pre-existing ARM storage account. This is the storage account used as an intermediary for copying files to Azure VMs.

ContainerName - Container Name

`string`. Required when `Destination = AzureBlob`.

The name of the container into which files are copied. If the specified container does not exist in the storage account, it will be created.

To create a virtual directory inside the container, use the blob prefix input. For example, for the target location `https://myaccount.blob.core.windows.net/mycontainer/vd1/vd2/`, specify container name `mycontainer` and blob prefix: `vd1/vd2`.

BlobPrefix - Blob Prefix

`string`. Optional. Use when `Destination = AzureBlob`.

Specify a prefix that can be used to filter files.

Example: You can append a build number to filter the files from all blobs with the same build number.

Example: If you specify a blob prefix `myvd1`, a virtual directory is created inside the container. Files are copied from the source to

`https://myaccount.blob.core.windows.net/mycontainer/myvd1/`.

resourceGroup - Resource Group

Input alias: `EnvironmentNameRM`. `string`. Required when `Destination = AzureVMs`.

Specify the name of the target Resource Group into which files will be copied.

ResourceFilteringMethod - Select Machines By

`string`. Optional. Use when `Destination = AzureVMs`. Allowed values: `machineNames` (Machine Names), `tags`. Default value: `machineNames`.

Specify a VM host name or tag that identifies a subset of VMs in a resource group. `Tags` are supported for resources created via the Azure Resource Manager only.

MachineNames - Filter Criteria

`string`. Optional. Use when `Destination = AzureVMs`.

Provide a list of VM names or tag names that identify the VMs the task will target. Valid filter criteria includes:

- The name of an [Azure Resource Group](#).
- An output variable from a previous task.
- A comma-delimited list of tag names or VM names.
- Format VM names using a comma-separated list of FQDNs or IP addresses.
- Format tag names for a filter as `{TagName}:{Value}`. Example: `Role:DB;OS:Win8.1, ffweb, ffdb`, or tags like `Role:DB, Web, OS:Win8.1`.

Note: Valid delimiters for tags include ,(comma), :(colon) and ;(semicolon). When providing multiple tags, the task will run only in the VMs that contain the specified tags. By default, the task runs in all VMs.

vmsAdminUserName - Admin Login

`string`. Required when `Destination = AzureVMs`.

Provide the user name of an account with administrative permissions on all of the target VMs.

- Supported formats include: `username`, `domain\username`, `machine-name\username`, and `.\username`.
- UPN formats including `username@domain.com` and built-in system accounts such as `NT Authority\System` are not supported.

vmsAdminPassword - Password

`string`. Required when `Destination = AzureVMs`.

Provide the administrator password of the VMs.

Valid input includes variables defined in build or release pipelines such as `$(passwordVariable)`. To secure a password, mark it as `secret`.

TargetPath - Destination Folder

`string`. Required when `Destination = AzureVMs`.

Specify the path to the folder in the Azure VMs into which files will be copied.

Environment variables such as `$env:windir` and `$env:systemroot` are supported.

Examples: `$env:windir\FabrikamFiber\Web` and `c:\FabrikamFiber`

AdditionalArgumentsForBlobCopy - Optional Arguments (for uploading files to blob)

`string`.

Provide additional arguments to `AzCopy.exe` that can be applied when uploading to Blobs such as `/NC:10`.

If no optional arguments are specified, the following arguments are added by default.

- `/Y`
- `/SetContentType`
- `/Z`
- `/V`
- `/S` - Added when the container name is not `$root`.
- `/BlobType:page` -Added when the specified storage account is a premium account.
- `/Pattern` - Added when the source path is a file. Included with any other specified optional arguments.

AdditionalArgumentsForVMCopy - Optional Arguments (for downloading files to VM)

`string`. Optional. Use when `Destination = AzureVMs`.

Provide additional arguments to `AzCopy.exe` that can be applied when downloading to VMs such as `/NC:10`.

If no optional arguments are specified, the following are added by default.

- `/Y`
- `/S`
- `/Z`
- `/V`

enableCopyPrerequisites - Enable Copy Prerequisites

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `false`.

When enabled, uses a self-signed certificate to configure a Windows Remote Management (WinRM) listener on port 5986 instead of the HTTPS protocol. Required for performing copy operation on Azure VMs. If the target VMs use a load balancer, configure inbound NAT rules for the target port (5986). Applies only for ARM VMs. On target VMs associated with a Network Security Group (NSG), configure an inbound security rule to allow access on port 5986.

CopyFilesInParallel - Copy in Parallel

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `true`.

Specify `true` to copy files in parallel to the target VMs. Using this value can reduce the overall time taken to perform the action.

CleanTargetBeforeCopy - Clean Target

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `false`.

Setting this value to `true` cleans the destination folder before performing the copy action.

skipCACheck - Test Certificate

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `true`.

The default value will not validate if the server certificate was signed by a trusted CA before connecting over HTTPS.

outputStorageUri - Storage Container URI

`string`.

Specify the name of the variable used for the storage container URI to which files were copied. Valid only when the selected destination is an Azure Blob.

outputStorageContainerSasToken - Storage Container SAS Token

`string`.

Specify the name of the variable used for the storage container SAS token that accesses the files that were copied. Use this variable as an input to subsequent tasks. By default, the SAS token expires after 4 hours.

sasTokenTimeOutInMinutes - SAS Token Expiration Period In Minutes

`string`.

Specify the time in minutes after which the SAS token will expire. Valid only when the selected destination is Azure Blob.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

What's new in Version AzureFileCopy@3

- AzureFileCopy@3 supports Az Module and stopped supporting the Azure classic service endpoint.
- The task is used to copy application files and other artifacts that are required to install the app such as PowerShell scripts, PowerShell-DSC modules, and more.
- When the target is Azure VMs, the files are first copied to an automatically generated Azure Blob container and then downloaded into the VMs. The container is deleted after the files are successfully copied to the VMs.
- The task uses **AzCopy**, the command-line utility built to quickly copy of data from and into Azure storage accounts. The task version 3 or below uses AzCopy V7.
- To dynamically deploy Azure Resource Groups that contain virtual machines, use the [Azure Resource Group Deployment](#) task. This task has a sample template that can perform the required operations to set up the WinRM HTTPS protocol on VMs, open port 5986 in the firewall, and install the test certificate.

Note

If you're deploying to Azure Static Websites as a container in Blob storage, use [Version 2 or higher](#) in order to preserve the **\$web** container name.

FAQ

What are the Azure PowerShell prerequisites for using this task?

The task requires that Azure PowerShell is installed on the machine running the automation agent. The recommended version is 1.0.2, but the task works with version 0.9.8 and higher. Use [Azure PowerShell Installer v1.0.2](#) to get the recommended version.

What are the WinRM prerequisites for this task?

The task uses the WinRM HTTPS protocol to copy the files from the storage Blob container to the Azure VMs. The WinRM HTTPS service must be configured on the VMs, and a suitable certificate installed.

If the VMs are created without opening the WinRM HTTPS ports, follow these steps:

1. Configure an inbound access rule to allow HTTPS on port 5986 of each VM.
2. Disable [UAC remote restrictions](#).
3. Specify the credentials for the task to access the VMs using an administrator-level login formatted as **username** without any domain reference.
4. Install a certificate on the machine that runs the automation agent.
5. Set the **Test Certificate** parameter of the task for a self-signed certificate.

What type of service connection should I choose?

The following table lists storage account types and the associated service connections. To identify whether a storage account is based on the classic APIs or the Resource Manager APIs, log into the [Azure portal](#) and search for **Storage accounts (Classic)** or **Storage accounts**.

Storage account type	Azure Service Connections in TFS/TS
Resource Manager	Azure Resource Manager service connection
Classic	Azure service connection with certificate-based or credentials-based authentication using a school or work account

- For Azure classic resources, use an **Azure** service connection type with certificate or credentials-based authentication. If you're using credentials-based authentication, ensure that the credentials are for a [school or work account](#). Microsoft accounts such as `joe@live.com` and `joe@hotmail.com` are not supported.
- For Azure Resource Manager VMs, use an **Azure Resource Manager** service connection type. For more details, see [Automating Azure Resource Group](#)

[deployment using a Service Principal ↗](#).

- If using an **Azure Resource Manager** service connection type, or an **Azure** service connection type with certificate-based authentication, the task automatically filters the appropriate classic storage accounts, newer Azure Resource Manager storage accounts, and other fields. For example, the Resource Group or cloud service, and the virtual machines.

 **Note**

Currently an **Azure** service connection type with credentials-based authentication does not filter the storage, Resource Group or cloud service, and virtual machine fields.

How do I fix failure '403: This request is not authorized to perform this operation using this permission'?

When Azure DevOps creates and authorizes the service connection to Azure, it creates an App Registration in your subscription's Active Directory. This identity is automatically added with a `Contributor` role to all resources in the Resource Group you chose to authorize. In order to upload Blobs to a storage account, being a `Contributor` is *not enough*. You must manually assign the [Storage Blob Data Contributor role to the app registration identity](#).

Copy the app identity from the existing inherited entry as `Contributor` that you'll see in the IAM pane and search explicitly for it in the `Add role assignment` UI. The identity is not listed in the dropdown, you must search for its identifier.

What happens if my Resource Group contains both Classic and Resource Manager VMs?

If the specified Resource Group contains both Azure Resource Manager and Classic VMs, the set of VMs that are targeted depends on the connection type.

- For certificate-based connections and credentials-based connections, the copy operation is only performed on Classic VMs.
- For Service Principal Name based connections, the copy operation is performed only on Resource Manager VMs.

How do I create a school or work account for use with this task?

A suitable account can be easily created for use in a service connection:

1. Use the Azure portal to create a new user account in Azure Active Directory.
2. Add the Azure Active Directory user account to the co-administrators group in your Azure subscription.
3. Sign into the Azure portal with this user account and change the password.
4. Use the new credentials for this account in the service connection. Deployments will be processed using this account.

Examples

yml

```
# Example: Upload files from Pipeline staging directory to blob storage.
- task: AzureFileCopy@3
  displayName: 'Example Step Name'
  inputs:
    sourcePath: '$(Build.ArtifactStagingDirectory)/BlobsToUpload'
    additionalArgumentsForBlobCopy: |
      '/Y' # Suppresses all AZCopy Confirmations. Used here to allow
    overwrites
      '/Pattern:/*' # Pattern of files to copy.
      '/S' # Recursive Copy
    azureSubscription: 'Subscription Name'
    destination: AzureBlob
    storage: storageaccountname
    containerName: storagecontainername
    blobPrefix: targetdirectoryincontainer
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: azureps
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any

Requirement	Description
Agent version	1.103.0 or greater
Task category	Deploy

AzureFileCopy@2 - Azure file copy v2 task

Article • 09/26/2023

Copy files to Azure Blob Storage or virtual machines.

ⓘ Note

This task does not support Azure Resource Manager authentication with workflow identity federation.

Syntax

YAML

```
# Azure file copy v2
# Copy files to Azure Blob Storage or virtual machines.
- task: AzureFileCopy@2
  inputs:
    SourcePath: # string. Required. Source.
    #azureConnectionType: 'ConnectedServiceNameARM' # 'ConnectedServiceName'
    | 'ConnectedServiceNameARM'. Alias: ConnectedServiceNameSelector. Azure
    Connection Type. Default: ConnectedServiceNameARM.
    #azureClassicSubscription: # string. Alias: ConnectedServiceName.
    Required when ConnectedServiceNameSelector = ConnectedServiceName. Azure
    Classic Subscription.
    azureSubscription: # string. Alias: ConnectedServiceNameARM. Required
    when ConnectedServiceNameSelector = ConnectedServiceNameARM. Azure
    Subscription.
    Destination: # 'AzureBlob' | 'AzureVMs'. Required. Destination Type.
    #classicStorage: # string. Alias: StorageAccount. Required when
    ConnectedServiceNameSelector = ConnectedServiceName. Classic Storage
    Account.
    storage: # string. Alias: StorageAccountRM. Required when
    ConnectedServiceNameSelector = ConnectedServiceNameARM. RM Storage Account.
    #ContainerName: # string. Required when Destination = AzureBlob.
    Container Name.
    #BlobPrefix: # string. Optional. Use when Destination = AzureBlob. Blob
    Prefix.
    #cloudService: # string. Alias: EnvironmentName. Required when
    ConnectedServiceNameSelector = ConnectedServiceName && Destination =
    AzureVMs. Cloud Service.
    #resourceGroup: # string. Alias: EnvironmentNameRM. Required when
    ConnectedServiceNameSelector = ConnectedServiceNameARM && Destination =
    AzureVMs. Resource Group.
    #ResourceFilteringMethod: 'machineNames' # 'machineNames' | 'tags'.
```

```
Optional. Use when Destination = AzureVMs. Select Machines By. Default:  
machineNames.  
    #MachineNames: # string. Optional. Use when Destination = AzureVMs.  
    Filter Criteria.  
        #vmsAdminUserName: # string. Required when Destination = AzureVMs. Admin  
        Login.  
        #vmsAdminPassword: # string. Required when Destination = AzureVMs.  
        Password.  
        #TargetPath: # string. Required when Destination = AzureVMs. Destination  
        Folder.  
        #AdditionalArgumentsForBlobCopy: # string. Optional Arguments (for  
        uploading files to blob).  
        #AdditionalArgumentsForVMCopy: # string. Optional. Use when Destination  
        = AzureVMs. Optional Arguments (for downloading files to VM).  
        #enableCopyPrerequisites: false # boolean. Optional. Use when  
        ConnectedServiceNameSelector = ConnectedServiceNameARM && Destination =  
        AzureVMs. Enable Copy Prerequisites. Default: false.  
        #CopyFilesInParallel: true # boolean. Optional. Use when Destination =  
        AzureVMs. Copy in Parallel. Default: true.  
        #CleanTargetBeforeCopy: false # boolean. Optional. Use when Destination  
        = AzureVMs. Clean Target. Default: false.  
        #skipCACheck: true # boolean. Optional. Use when Destination = AzureVMs.  
        Test Certificate. Default: true.  
    # Output  
    #outputStorageUri: # string. Storage Container URI.  
    #outputStorageContainerSasToken: # string. Storage Container SAS Token.
```

Inputs

`SourcePath` - Source

`string`. Required.

Specify the absolute path to the source folder, file on the local machine, or a UNC share. The specified value or expression should return either a single folder name or a file name.

`azureConnectionType` - Azure Connection Type

Input alias: `ConnectedServiceNameSelector`. `string`. Allowed values:

`ConnectedServiceName` (Azure Classic), `ConnectedServiceNameARM` (Azure Resource Manager). Default value: `ConnectedServiceNameARM`.

Specify the Azure connection type.

`azureClassicSubscription` - Azure Classic Subscription

Input alias: `ConnectedServiceName`. `string`. Required when `ConnectedServiceNameSelector`

= ConnectedServiceName.

Specify the target Azure Classic subscription.

azureSubscription - Azure Subscription

Input alias: ConnectedServiceNameARM. string. Required when

ConnectedServiceNameSelector = ConnectedServiceNameARM.

Specify the target Azure Resource Manager subscription.

Destination - Destination Type

string. Required. Allowed values: AzureBlob (Azure Blob), AzureVMs (Azure VMs).

Specify the destination type to use for copying the files.

classicStorage - Classic Storage Account

Input alias: StorageAccount. string. Required when ConnectedServiceNameSelector = ConnectedServiceName.

Specify a pre-existing classic storage account. This is the storage account used as an intermediary for copying files to Azure VMs.

storage - RM Storage Account

Input alias: StorageAccountRM. string. Required when ConnectedServiceNameSelector = ConnectedServiceNameARM.

Specify a pre-existing ARM storage account. This is the storage account used as an intermediary for copying files to Azure VMs.

ContainerName - Container Name

string. Required when Destination = AzureBlob.

Specify the name of the container into which files are copied. If the specified container does not exist in the storage account, it will be created.

To create a virtual directory inside the container, use the Blob prefix input. For example, for target location <https://myaccount.blob.core.windows.net/mycontainer/vd1/vd2/>, specify the container name mycontainer and Blob prefix vd1/vd2.

BlobPrefix - Blob Prefix

`string`. Optional. Use when `Destination = AzureBlob`.

Specify a prefix that can be used to filter files.

Example: You can append a build number to filter the files from all Blobs with the same build number.

Example: If you specify a Blob prefix `myvd1`, a virtual directory is created inside the container. Files are copied from the source to

`https://myaccount.blob.core.windows.net/mycontainer/myvd1/`.

cloudService - Cloud Service

Input alias: `EnvironmentName`. `string`. Required when `ConnectedServiceNameSelector = ConnectedServiceName && Destination = AzureVMs`.

Specify the name of the target Cloud Service.

resourceGroup - Resource Group

Input alias: `EnvironmentNameRM`. `string`. Required when `ConnectedServiceNameSelector = ConnectedServiceNameARM && Destination = AzureVMs`.

Specify the name of the target Resource Group.

ResourceFilteringMethod - Select Machines By

`string`. Optional. Use when `Destination = AzureVMs`. Allowed values: `machineNames` (Machine Names), `tags`. Default value: `machineNames`.

Specify the VM host name or tag that identifies a subset of VMs in a resource group.

[Tags](#) are supported for resources created via the Azure Resource Manager only.

MachineNames - Filter Criteria

`string`. Optional. Use when `Destination = AzureVMs`.

Provide a list of Azure VM host names such as `ffweb`, `ffdb`, or tags such as `Role:DB`, `Web`, `OS:Win8.1`.

Note: Valid delimiters for tags include ,(comma), :(colon) and ;(semicolon). When providing multiple tags, the task will run in all the VMs that contains the specified tags. By default, the task runs in all the VMs.

vmsAdminUserName - Admin Login

`string`. Required when `Destination = AzureVMs`.

Provide the user name of the Azure VM administrator account.

vmsAdminPassword - Password

`string`. Required when `Destination = AzureVMs`.

Provide the password for the Azure VM administrator account.

Valid input includes variables defined in build or release pipelines such as `$(passwordVariable)`. To secure a password, mark it as `secret`.

TargetPath - Destination Folder

`string`. Required when `Destination = AzureVMs`.

Specify the local path on the target VMs.

Valid input includes environment variables such as `$env:windir\BudgetIT\Web`.

AdditionalArgumentsForBlobCopy - Optional Arguments (for uploading files to blob)

`string`.

Provide additional arguments to `AzCopy.exe` that can be applied when uploading to Blobs such as `/NC:10`.

If no optional arguments are specified, the following arguments are added by default.

- `/Y`
- `/SetContentType`
- `/Z`
- `/V`
- `/S` - Added when the container name is not `$root`.
- `/BlobType:page` - Added when the specified storage account is a premium account.
- `/Pattern` - Added when the source path is a file. Included with any other specified optional arguments.

AdditionalArgumentsForVMCopy - Optional Arguments (for downloading files to VM)

`string`. Optional. Use when `Destination = AzureVMs`.

Provide additional arguments to `AzCopy.exe` that can be applied when downloading to VMs such as `/NC:10`.

If no optional arguments are specified, the following are added by default.

- `/Y`
- `/S`
- `/Z`
- `/V`

enableCopyPrerequisites - Enable Copy Prerequisites

`boolean`. Optional. Use when `ConnectedServiceNameSelector = ConnectedServiceNameARM` && `Destination = AzureVMs`. Default value: `false`.

When enabled, uses a self-signed certificate to configure a Windows Remote Management (WinRM) listener on port 5986 instead of the HTTPS protocol. Required for performing copy operations on Azure VMs. If the target VMs use a load balancer, configure inbound NAT rules for the target port (5986). Applies only for ARM VMs.

CopyFilesInParallel - Copy in Parallel

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `true`.

Accepting the default setting copies files in parallel to the target VMs.

CleanTargetBeforeCopy - Clean Target

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `false`.

Setting this value to `true` cleans the destination folder before performing the copy action.

skipCACheck - Test Certificate

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `true`.

The default value will not validate that the server certificate was signed by a trusted CA before connecting over HTTPS.

`outputStorageUri` - Storage Container URI

`string`.

Specify the name of the variable used for the storage container URI to which files were copied. Valid only when the selected destination is an Azure Blob.

`outputStorageContainerSasToken` - Storage Container SAS Token

`string`.

Specify the name of the variable used for the storage container SAS token that accesses the files that were copied. Valid only when the selected destination is an Azure Blob.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

What's new in Version 2.0: Using newer version of AzCopy.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: azureps
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any

Requirement	Description
Settable variables	Any
Agent version	1.103.0 or greater
Task category	Deploy

AzureFileCopy@1 - Azure file copy v1 task

Article • 09/26/2023

Copy files to Azure Blob Storage or virtual machines.

ⓘ Note

This task does not support Azure Resource Manager authentication with workflow identity federation.

Syntax

YAML

```
# Azure file copy v1
# Copy files to Azure Blob Storage or virtual machines.
- task: AzureFileCopy@1
  inputs:
    SourcePath: # string. Required. Source.
    #azureConnectionType: 'ConnectedServiceNameARM' # 'ConnectedServiceName'
    | 'ConnectedServiceNameARM'. Alias: ConnectedServiceNameSelector. Azure
    Connection Type. Default: ConnectedServiceNameARM.
    #azureClassicSubscription: # string. Alias: ConnectedServiceName.
    Required when ConnectedServiceNameSelector = ConnectedServiceName. Azure
    Classic Subscription.
    azureSubscription: # string. Alias: ConnectedServiceNameARM. Required
    when ConnectedServiceNameSelector = ConnectedServiceNameARM. Azure
    Subscription.
    Destination: # 'AzureBlob' | 'AzureVMs'. Required. Destination Type.
    #classicStorage: # string. Alias: StorageAccount. Required when
    ConnectedServiceNameSelector = ConnectedServiceName. Classic Storage
    Account.
    storage: # string. Alias: StorageAccountRM. Required when
    ConnectedServiceNameSelector = ConnectedServiceNameARM. RM Storage Account.
    #ContainerName: # string. Required when Destination = AzureBlob.
    Container Name.
    #BlobPrefix: # string. Optional. Use when Destination = AzureBlob. Blob
    Prefix.
    #cloudService: # string. Alias: EnvironmentName. Required when
    ConnectedServiceNameSelector = ConnectedServiceName && Destination =
    AzureVMs. Cloud Service.
    #resourceGroup: # string. Alias: EnvironmentNameRM. Required when
    ConnectedServiceNameSelector = ConnectedServiceNameARM && Destination =
    AzureVMs. Resource Group.
    #ResourceFilteringMethod: 'machineNames' # 'machineNames' | 'tags'.
```

```
Optional. Use when Destination = AzureVMs. Select Machines By. Default:  
machineNames.  
    #MachineNames: # string. Optional. Use when Destination = AzureVMs.  
    Filter Criteria.  
        #vmsAdminUserName: # string. Required when Destination = AzureVMs. Admin  
        Login.  
        #vmsAdminPassword: # string. Required when Destination = AzureVMs.  
        Password.  
        #TargetPath: # string. Required when Destination = AzureVMs. Destination  
        Folder.  
        #AdditionalArguments: # string. Additional Arguments.  
        #enableCopyPrerequisites: false # boolean. Optional. Use when  
        ConnectedServiceNameSelector = ConnectedServiceNameARM && Destination =  
        AzureVMs. Enable Copy Prerequisites. Default: false.  
        #CopyFilesInParallel: true # boolean. Optional. Use when Destination =  
        AzureVMs. Copy in Parallel. Default: true.  
        #CleanTargetBeforeCopy: false # boolean. Optional. Use when Destination  
        = AzureVMs. Clean Target. Default: false.  
        #skipCACheck: true # boolean. Optional. Use when Destination = AzureVMs.  
        Test Certificate. Default: true.  
    # Output  
    #outputStorageUri: # string. Storage Container URI.  
    #outputStorageContainerSasToken: # string. Storage Container SAS Token.
```

Inputs

`SourcePath` - Source

`string`. Required.

Specify the absolute path of the source folder, file on the local machine, or a UNC share. The specified value or expression should return either a single folder name or a file name.

`azureConnectionType` - Azure Connection Type

Input alias: `ConnectedServiceNameSelector`. `string`. Allowed values:

`ConnectedServiceName` (Azure Classic), `ConnectedServiceNameARM` (Azure Resource Manager). Default value: `ConnectedServiceNameARM`.

Specify the Azure connection type.

`azureClassicSubscription` - Azure Classic Subscription

Input alias: `ConnectedServiceName`. `string`. Required when `ConnectedServiceNameSelector` = `ConnectedServiceName`.

Specify the target Azure Classic subscription.

azureSubscription - Azure Subscription

Input alias: `ConnectedServiceNameARM`. `string`. Required when

`ConnectedServiceNameSelector = ConnectedServiceNameARM`.

Specify the target Azure Resource Manager subscription.

Destination - Destination Type

`string`. Required. Allowed values: `AzureBlob` (Azure Blob), `AzureVMs` (Azure VMs).

Specify the destination type to use for copying the files.

classicStorage - Classic Storage Account

Input alias: `StorageAccount`. `string`. Required when `ConnectedServiceNameSelector = ConnectedServiceName`.

Specify a pre-existing classic storage account. This is the storage account used as an intermediary for copying files to Azure VMs.

storage - RM Storage Account

Input alias: `StorageAccountRM`. `string`. Required when `ConnectedServiceNameSelector = ConnectedServiceNameARM`.

Specify a pre-existing ARM storage account. This is the storage account used as an intermediary for copying files to Azure VMs.

ContainerName - Container Name

`string`. Required when `Destination = AzureBlob`.

Specify the name of the container into which files are copied. If the specified container does not exist in the storage account, it will be created.

To create a virtual directory inside the container use the Blob prefix input.

Example: For target location

`https://myaccount.blob.core.windows.net/mycontainer/vd1/vd2/`, specify container name `mycontainer` and Blob prefix: `vd1/vd2`.

BlobPrefix - Blob Prefix

`string`. Optional. Use when `Destination = AzureBlob`.

Specify a prefix that can be used to filter files.

Example: You can append a build number to filter the files from all Blobs with the same build number.

Example: If you specify a Blob prefix `myvd1`, a virtual directory is created inside the container. Files are copied from the source to

`https://myaccount.blob.core.windows.net/mycontainer/myvd1/`.

cloudService - Cloud Service

Input alias: `EnvironmentName`. `string`. Required when `ConnectedServiceNameSelector = ConnectedServiceName && Destination = AzureVMs`.

Specify the name of the target Cloud Service.

resourceGroup - Resource Group

Input alias: `EnvironmentNameRM`. `string`. Required when `ConnectedServiceNameSelector = ConnectedServiceNameARM && Destination = AzureVMs`.

Specify the name of the target Resource Group.

ResourceFilteringMethod - Select Machines By

`string`. Optional. Use when `Destination = AzureVMs`. Allowed values: `machineNames` (Machine Names), `tags`. Default value: `machineNames`.

Specify a VM host name or tag that identifies a subset of VMs in a resource group. [Tags](#) are supported for resources created via the Azure Resource Manager only.

MachineNames - Filter Criteria

`string`. Optional. Use when `Destination = AzureVMs`.

Provide a list of Azure VM host names such as `ffweb`, `ffdb`, or tags such as `Role:DB`, `Web`, `OS:Win8.1`.

Note: Valid delimiters for tags include ,(comma), :(colon) and ;(semicolon). When providing multiple tags, the task will run only in the VMs that contain the specified tags. By default, the task runs in all VMs.

vmsAdminUserName - Admin Login

`string`. Required when `Destination = AzureVMs`.

Specify the user name of the Azure VM administrator account.

vmsAdminPassword - Password

`string`. Required when `Destination = AzureVMs`.

Specify the password for the Azure VM administrator account.

Valid input includes variables defined in build or release pipelines such as `$(passwordVariable)`. To secure a password, mark it as `secret`.

TargetPath - Destination Folder

`string`. Required when `Destination = AzureVMs`.

Specify the local path on the target VMs for copying files from the source.

Valid input includes environment variables such as `$env:windir\BudgetIT\Web`.

AdditionalArguments - Additional Arguments

`string`.

Provide additional arguments to `AzCopy.exe` that will be applied when uploading to Blobs or VMs such as, `/NC:10`.

enableCopyPrerequisites - Enable Copy Prerequisites

`boolean`. Optional. Use when `ConnectedServiceNameSelector = ConnectedServiceNameARM` `&& Destination = AzureVMs`. Default value: `false`.

When enabled, uses a self-signed certificate to configure a Windows Remote Management (WinRM) listener on port 5986 instead of the HTTPS protocol. Required for performing copy operations on Azure VMs. If the target VMs use a load balancer, configure inbound NAT rules for the target port (5986). Applies only for ARM VMs.

CopyFilesInParallel - Copy in Parallel

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `true`.

Accepting the default setting copies files in parallel to the target VMs.

CleanTargetBeforeCopy - Clean Target

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `false`.

Setting to `true` cleans the destination folder before performing the copy action.

skipCACheck - Test Certificate

`boolean`. Optional. Use when `Destination = AzureVMs`. Default value: `true`.

The default value will not validate if the server certificate was signed by a trusted CA before connecting over HTTPS.

outputStorageUri - Storage Container URI

`string`.

Specify the name of the variable used for the storage container URI to which files were copied. Valid only when the selected destination is an Azure Blob.

outputStorageContainerSasToken - Storage Container SAS Token

`string`.

Specify the name of the variable used for the storage container SAS token that accesses the files that were copied. Valid only when the selected destination is an Azure Blob.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: azureps
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.103.0 or greater
Task category	Deploy

AzureFunctionOnKubernetes@1 - Azure Function on Kubernetes v1 task

Article • 09/26/2023

Deploy Azure function to Kubernetes cluster.

Syntax

YAML

```
# Azure Function on Kubernetes v1
# Deploy Azure function to Kubernetes cluster.
- task: AzureFunctionOnKubernetes@1
  inputs:
    # Service Connections
    connectionType: 'Kubernetes Service Connection' # 'Azure Resource Manager' | 'Kubernetes Service Connection'. Required. Service connection type. Default: Kubernetes Service Connection.
    dockerRegistryServiceConnection: # string. Required. Docker registry service connection.
    #kubernetesServiceConnection: # string. Alias: kubernetesServiceEndpoint. Required when connectionType = Kubernetes Service Connection. Kubernetes service connection.
    #azureSubscriptionConnection: # string. Alias: azureSubscriptionEndpoint. Required when connectionType = Azure Resource Manager. Azure subscription.
    #azureResourceGroup: # string. Required when connectionType = Azure Resource Manager. Resource group.
    #kubernetesCluster: # string. Required when connectionType = Azure Resource Manager. Kubernetes cluster.
    # Commands
    #namespace: # string. Kubernetes namespace.
    #secretName: # string. Secret Name.
    #dockerHubNamespace: # string. Docker Hub namespace.
    appName: # string. Required. Application Name.
    #functionRootDirectory: # string. Function root directory.
    #waitForStability: true # boolean. Wait for stability. Default: true.
    #arguments: # string. Arguments.
```

Inputs

`connectionType` - Service connection type

`string`. Required. Allowed values: `Azure Resource Manager`, `Kubernetes Service Connection`. Default value: `Kubernetes Service Connection`.

Select a Kubernetes service connection type.

- **Kubernetes Service Connection** - Allows you to provide a KubeConfig file, specify a Service Account, or import an AKS instance with the **Azure Subscription** option. Importing an AKS instance with the **Azure Subscription** option requires Kubernetes cluster access at Service Connection configuration time.
- **Azure Resource Manager** - Lets you select an AKS instance. Does not access Kubernetes cluster at Service Connection configuration time.

For more information, see [Remarks](#).

dockerRegistryServiceConnection - Docker registry service connection

`string`. Required.

Select a Docker registry service connection.

kubernetesServiceConnection - Kubernetes service connection

Input alias: `kubernetesServiceEndpoint`. `string`. Required when `connectionType = Kubernetes Service Connection`.

Select a Kubernetes service connection.

azureSubscriptionConnection - Azure subscription

Input alias: `azureSubscriptionEndpoint`. `string`. Required when `connectionType = Azure Resource Manager`.

Select the Azure Resource Manager subscription, which contains Azure Container Registry. Note: To configure new service connection, select the Azure subscription from the list and click 'Authorize'. If your subscription is not listed or if you want to use an existing Service Principal, you can setup an Azure service connection using 'Add' or 'Manage' button.

azureResourceGroup - Resource group

`string`. Required when `connectionType = Azure Resource Manager`.

Select an Azure resource group.

kubernetesCluster - Kubernetes cluster

`string`. Required when `connectionType = Azure Resource Manager`.

Select an Azure managed cluster.

namespace - Kubernetes namespace

`string`.

Kubernetes namespace.

secretName - Secret Name

`string`.

Kubernetes secret containing function config data (for ex. AzureWebJobsStorage: `Azure storage connection string`).

dockerHubNamespace - Docker Hub namespace

`string`.

Docker Hub namespace. Required for private Docker Hub repository.

appName - Application Name

`string`. Required.

Application Name. The Kubernetes objects created use this name. This should follow Kubernetes naming conventions for resource names.

functionRootDirectory - Function root directory

`string`.

Function root directory. Should contain host.json. Docker build and push is performed from this directory.

waitForStability - Wait for stability

`boolean`. Default value: `true`.

Wait for the Kubernetes objects to reach the desired state.

arguments - Arguments

`string`.

Pass arguments to command. Ex:
--no-docker --service-type NodePort.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Kubernetes Service Connection considerations when accessing AKS

You can create a Kubernetes service connection with any of the following options.

- KubeConfig
- Service Account
- Azure Subscription

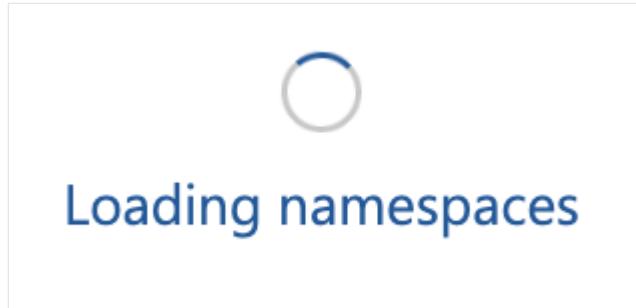
New Kubernetes service connection

Authentication method

- KubeConfig
- Service Account
- Azure Subscription

When selecting the **Azure Subscription** option, Kubernetes needs to be accessible to Azure DevOps at service connection configuration time. There may be various reasons a service connection cannot be created, for example you created a private cluster or the cluster has local accounts disabled. In these cases, Azure DevOps is unable to connect to

your cluster at service connection configuration time and you will observe the dialog to be stuck at **Loading namespaces**.



Starting with Kubernetes 1.24, long-lived tokens are [no longer created by default](#). Kubernetes recommends not to use long-lived tokens. As a result, tasks using a Kubernetes service connection created using the Azure Subscription option do not have access to the permanent token required to authenticate and can't access your Kubernetes cluster. This also results in the **Loading namespaces** dialog to be frozen.

Use the Azure Resource Manager Service Connection to access AKS

For AKS customers, the Azure Resource Manager service connection type provides the best method to connect to a private cluster, or a cluster that has local accounts disabled. This method is not dependent on cluster connectivity at the time you create a service connection. Access to AKS is deferred to pipeline runtime, which has the following advantages:

- Access to a (private) AKS cluster can be performed from a self-hosted or scale set agent with line of sight to the cluster.
- A token is created for every task that uses an Azure Resource Manager service connection. This ensures you are connecting to Kubernetes with a short-lived token, which is the [Kubernetes recommendation](#).
- AKS can be accessed even when local accounts are disabled.

Service connection FAQ

I receive the following error message: Could not find any secret associated with the service account. What is happening?

You are using the Kubernetes service connection with Azure Subscription option. We are updating this method to create long-lived tokens. This is expected to be available mid-May. However, it is recommended to start using the Azure service connection type and not to use long-lived tokens as per [Kubernetes guidance](#).

I'm using AKS and don't want to change anything, can I continue to use tasks with the Kubernetes service connection?

We are updating this method to create long-lived tokens. This is expected to be available mid-May. However, please be aware that this approach is against [Kubernetes guidance](#).

I'm using the Kubernetes tasks and Kubernetes service connection but not AKS. Should I be concerned?

Your tasks will continue to work as before.

Will the Kubernetes service connection type be removed?

Our Kubernetes tasks work with any Kubernetes cluster, regardless where they are running. The Kubernetes service connection will continue to exist.

I'm an AKS customer and everything is running fine, should I act?

There is no need to change anything. If you are using the Kubernetes service connection and selected Azure Subscription during creation, you should be aware of the [Kubernetes guidance on using long-lived tokens](#).

I'm creating a Kubernetes Environment, and have no option to use service connections

In case you can't access your AKS during environment creation time, you can use an empty environment and set the `connectionType` input to an Azure Resource Manager service connection.

I have AKS configured with Azure Active Directory RBAC, and my pipeline doesn't work. Will these updates resolve that?

Accessing Kubernetes when AAD RBAC is enabled is unrelated to token creation. To prevent an interactive prompt, we will support [kubelogin](#) in a future update.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release

Requirement	Description
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Deploy

AzureFunctionOnKubernetes@0 - Azure Function on Kubernetes v0 task

Article • 09/26/2023

Deploy Azure function to Kubernetes cluster.

Syntax

YAML

```
# Azure Function on Kubernetes v0
# Deploy Azure function to Kubernetes cluster.
- task: AzureFunctionOnKubernetes@0
  inputs:
    # Service Connections
    dockerRegistryServiceConnection: # string. Required. Docker registry service connection.
    kubernetesServiceConnection: # string. Required. Kubernetes service connection.
    # Commands
    #namespace: # string. Kubernetes namespace.
    #secretName: # string. Secret Name.
    #dockerHubNamespace: # string. Docker Hub namespace.
    appName: # string. Required. Application Name.
    #functionRootDirectory: # string. Function root directory.
    #waitForStability: true # boolean. Wait for stability. Default: true.
    #arguments: # string. Arguments.
```

Inputs

`dockerRegistryServiceConnection` - Docker registry service connection

`string`. Required.

Select a Docker registry service connection.

`kubernetesServiceConnection` - Kubernetes service connection

`string`. Required.

Select a Kubernetes service connection.

namespace - Kubernetes namespace

`string.`

Kubernetes namespace.

secretName - Secret Name

`string.`

Kubernetes secret containing function config data (for ex. AzureWebJobsStorage: `Azure storage connection string`).

dockerHubNamespace - Docker Hub namespace

`string.`

Docker Hub namespace. Required for private Docker Hub repository.

appName - Application Name

`string.` Required.

Application Name. The Kubernetes objects created use this name. This should follow Kubernetes naming conventions for resource names.

functionRootDirectory - Function root directory

`string.`

Function root directory. Should contain host.json. Docker build and push is performed from this directory.

waitForStability - Wait for stability

`boolean.` Default value: `true`.

Wait for the Kubernetes objects to reach the desired state.

arguments - Arguments

`string.`

Pass arguments to command. Ex:
`--no-docker --service-type NodePort.`

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Note

There is a newer version of this task available that provides additional support for targetting a Kubernetes cluster in different ways, using the `connectionType` property. For more information, see [AzureFunctionOnKubernetes@1](#) and [AzureFunctionOnKubernetes@1 remarks](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Deploy

AzureFunctionApp@2 - Azure Functions Deploy v2 task

Article • 09/26/2023

Update a function app with .NET, Python, JavaScript, PowerShell, Java based web applications.

Syntax

YAML

```
# Azure Functions Deploy v2
# Update a function app with .NET, Python, JavaScript, PowerShell, Java
based web applications.
- task: AzureFunctionApp@2
  inputs:
    connectedServiceNameARM: # string. Alias: azureSubscription. Required.
    Azure Resource Manager connection.
    appType: # 'functionApp' | 'functionAppLinux'. Required. App type.
    appName: # string. Required. Azure Functions App name.
    #deployToSlotOrASE: false # boolean. Optional. Use when appType != "".
    Deploy to Slot or App Service Environment. Default: false.
    #resourceGroupName: # string. Required when deployToSlotOrASE = true.
    Resource group.
    #slotName: 'production' # string. Required when deployToSlotOrASE =
    true. Slot. Default: production.
    package: '$(System.DefaultWorkingDirectory)/**/*.zip' # string.
    Required. Package or folder. Default:
    $(System.DefaultWorkingDirectory)/**/*.zip.
    #runtimeStack: # 'DOTNET|2.2' | 'DOTNET|3.1' | 'DOTNET|6.0' | 'DOTNET-
    ISOLATED|7.0' | 'JAVA|8' | 'JAVA|11' | 'NODE|8' | 'NODE|10' | 'NODE|12' |
    'NODE|14' | 'NODE|16' | 'NODE|18' | 'PYTHON|3.6' | 'PYTHON|3.7' |
    'PYTHON|3.8' | 'PYTHON|3.9' | 'PYTHON|3.10'. Optional. Use when appType =
    functionAppLinux. Runtime stack.
    # Application and Configuration Settings
    #appSettings: # string. App settings.
    # Additional Deployment Options
    #deploymentMethod: 'auto' # 'auto' | 'zipDeploy' | 'runFromPackage'.
    Required when appType != "" && package NotEndsWith .war && Package
    NotEndsWith .jar. Deployment method. Default: auto.
```

Inputs

`connectedServiceNameARM` - Azure Resource Manager connection

Input alias: `azureSubscription`. `string`. Required.

Select the Azure Resource Manager subscription for the deployment.

appType - App type

`string`. Required. Allowed values: `functionApp` (Function App on Windows), `functionAppLinux` (Function App on Linux).

Select the Azure Function App type for the deployment.

appName - Azure Functions App name

`string`. Required.

Specify the name of an existing Azure Functions App. The Function Apps listed will be based on the selected app type.

deployToSlotOrASE - Deploy to Slot or App Service Environment

`boolean`. Optional. Use when `appType != ""`. Default value: `false`.

Deploys to an existing deployment slot or Azure App Service Environment. For both targets, the task needs a Resource group name.

If the deployment target is a slot, it will default to the **production** slot. Any other existing slot name can also be provided.

If the deployment target is an Azure App Service Environment, leave the slot name as **production** and specify the Resource group name.

resourceGroupName - Resource group

`string`. Required when `deployToSlotOrASE = true`.

The Resource group name is required when the deployment target is either a deployment slot or an App Service Environment.

Enters or selects the Azure Resource group that contains the Azure App Service specified above.

slotName - Slot

`string`. Required when `deployToSlotOrASE = true`. Default value: `production`.

Enters or selects an existing slot, excluding the Production slot.

package - Package or folder

`string`. Required. Default value: `$(System.DefaultWorkingDirectory)/**/*.zip`.

The file path to the package or folder that contains App Service content generated by MSBuild or a compressed zip file. Variables ([Build | Release](#)) and wildcards are supported. For example, `$(System.DefaultWorkingDirectory)/**/*.zip`.

runtimeStack - Runtime stack

`string`. Optional. Use when `appType = functionAppLinux`. Allowed values: `DOTNET|2.2` (`DOTNET|2.2` (functionapp v2)), `DOTNET|3.1` (`DOTNET|3.1` (functionapp v3)), `DOTNET|6.0` (`DOTNET|6.0` (functionapp v4)), `DOTNET-ISOLATED|7.0` (`DOTNET-ISOLATED|7.0` (functionapp v4)), `JAVA|8` (`JAVA|8` (functionapp v2/v3/v4)), `JAVA|11` (`JAVA|11` (functionapp v3/v4)), `NODE|8` (`NODE|8` (functionapp v2)), `NODE|10` (`NODE|10` (functionapp v2/v3)), `NODE|12` (`NODE|12` (functionapp v3)), `NODE|14` (`NODE|14` (functionapp v3/v4)), `NODE|16` (`NODE|16` (functionapp v4)), `NODE|18` (`NODE|18` (functionapp v4)), `PYTHON|3.6` (`PYTHON|3.6` (functionapp v2/v3)), `PYTHON|3.7` (`PYTHON|3.7` (functionapp v2/v3/v4)), `PYTHON|3.8` (`PYTHON|3.8` (functionapp v3/v4)), `PYTHON|3.9` (`PYTHON|3.9` (functionapp v3/v4)), `PYTHON|3.10` (`PYTHON|3.10` (functionapp v3/v4)).

Learn about [supported runtime versions](#). Old values like `DOCKER|microsoft/azure-functions-*` are deprecated. New values are listed in the dropdown menu.

appSettings - App settings

`string`.

Enter the application settings using the syntax `-key value` (for example: `-Port 5000` `-RequestTimeout 5000` `-WEBSITE_TIME_ZONE`). Enclose values that contain spaces in double quotes (for example: `"Eastern Standard Time"`).

deploymentMethod - Deployment method

`string`. Required when `appType != "" && package NotEndsWith .war && Package NotEndsWith .jar`. Allowed values: `auto` (Auto-detect), `zipDeploy` (Zip Deploy), `runFromPackage` (Zip Deploy with Run From Package). Default value: `auto`.

Chooses the [deployment method](#) for the app. Linux Consumption apps do not support this configuration.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

`AppServiceApplicationUrl`

Application URL of the selected Azure Function App.

Remarks

The Azure Function Deployment task is used to update Azure Functions to deploy [Functions](#) to Azure. The task works on cross platform Azure Pipelines agents running Windows, Linux or Mac and uses the underlying deployment technologies of RunFromPackage, Zip Deploy and [Kudu REST APIs](#).

The task works for the Azure Functions [supported languages](#).

Pre-requisites for the task

The following pre-requisites need to be setup in the target machine(s) for the task to work properly.

Azure Function

The task is used to deploy an Azure Functions project to an existing Azure Function. The Azure Function app should exist prior to running the task. The Azure Function App can be created from the [Azure portal](#). Alternatively, the [Azure PowerShell task](#) can be used to run [AzureRM PowerShell scripts](#) to provision and configure the Azure Function app.

The task can be used to deploy [Azure Functions](#) (Windows/Linux).

Azure Subscription

To deploy to Azure, an Azure subscription has to be linked to Azure Pipelines using the Services tab in the Account Administration section. Add the Azure subscription to use in

the Build or Release Management definition by opening the Account Administration screen (gear icon on the top-right of the screen) and then click on the Services Tab.

Create the [ARM](#) service endpoint and use **Azure Resource Manager** endpoint type. For more details, follow the steps listed in the link [here](#).

The task does not work with the Azure Classic service endpoint and it will not list these connections in the parameters in the task.

Deployment methods

Several deployment methods are available in this task.

To change the package-based deployment option in a designer task, expand **Additional Deployment Options** and enable **Select Deployment Method**.

Based on the type of Azure App Service and Azure Pipelines agent, the task uses a suitable deployment technology. The deployment technologies used by tasks are as follows:

- [Kudu REST API](#)
- [Zip Deploy](#)
- [Run From Package](#)

By default, the task attempts to select the appropriate deployment technology based on the input package, App Service type, and agent OS.

- If a post-deployment script is provided, use Zip Deploy.
- If the App Service type is Web App on Linux, use Zip Deploy.
- If a .war file is provided, use War Deploy.
- If a .jar file is provided, use Run-From-Zip.
- For all other tasks, use Run From Package (via Zip Deploy).

On a non-Windows agent (for any App Service type), the task relies on the [Kudu REST API](#) to deploy the web app.

Kudu REST API

The [Kudu REST API](#) works on both Windows and Linux automation agents when the target is a Web App on Windows, a Web App on Linux (built-in source), or a function app. The task uses Kudu to copy files to the Azure App Service.

Zip Deploy

Zip Deploy creates a .zip deployment package from the chosen package or folder. It then deploys the file contents to the wwwroot folder of the App Service name function app in Azure. This option overwrites all existing content in the wwwroot folder. For more information, see [Zip deployment for Azure Functions](#).

Run From Package

Run From Package creates the same deployment package as Zip Deploy. Instead of deploying files to the wwwroot folder, the Functions runtime mounts the entire package. When you use this option, files in the wwwroot folder become read-only. For more information, see [Run your Azure Functions from a package file](#).

Troubleshooting

Error: Could not fetch access token for Azure. Verify if the Service Principal used is valid and not expired.

The task uses the service principal in the service connection to authenticate with Azure. If the service principal has expired or doesn't have permissions to the App Service, the task fails with this error. Verify the validity of the service principal used and that it's present in the app registration. For more information, see [Use role-based access control to manage access to your Azure subscription resources](#). [This blog post](#) also contains more information about using service principal authentication.

SSL error

If you want to use a certificate in App Service, the certificate must be signed by a trusted certificate authority. If your web app gives you certificate validation errors, you're probably using a self-signed certificate. Set a variable named `VSTS_ARM_REST_IGNORE_SSL_ERRORS` to the value `true` in the build or release pipeline to resolve the error.

A release hangs for long time and then fails

This problem could be the result of insufficient capacity in your App Service plan. To resolve this problem, you can scale up the App Service instance to increase available CPU, RAM, and disk space or try with a different App Service plan.

5xx error codes

If you're seeing a 5xx error, check the status of your Azure service [\[?\]](#).

Azure Function suddenly stopped working

Azure Functions may suddenly stop working if more than one year has passed since the last deployment. If you deploy with "RunFromPackage" in "deploymentMethod", a SAS with an expiration date of 1 year is generated and set as the value of "WEBSITE_RUN_FROM_PACKAGE" in the application configuration. Azure Functions uses this SAS to reference the package file for function execution, so if the SAS has expired, the function will not be executed. To resolve this issue, deploy again to generate a SAS with an expiration date of one year.

Error: No package found with specified pattern

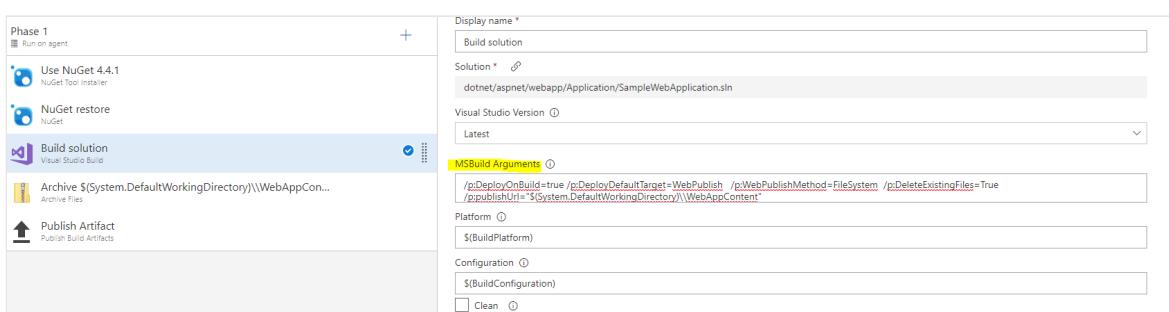
Check if the package mentioned in the task is published as an artifact in the build or a previous stage and downloaded in the current job.

Error: Publish using zip deploy option is not supported for msBuild package type

Web packages created via the MSBuild task (with default arguments) have a nested folder structure that can be deployed correctly only by Web Deploy. The publish-to-zip deployment option can't be used to deploy those packages. To convert the packaging structure, take these steps:

1. In the Build solution task, change the **MSBuild Arguments** to

```
/p:DeployOnBuild=true /p:DeployDefaultTarget=WebPublish  
/p:WebPublishMethod=FileSystem /p:DeleteExistingFiles=True  
/p:publishUrl="$(System.DefaultWorkingDirectory)\\WebAppContent":
```

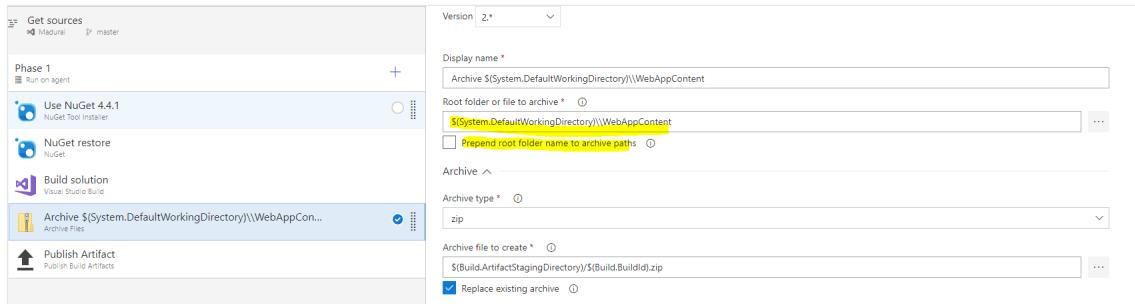


2. Add an Archive task and change the values as follows:

- a. Change **Root folder or file to archive** to

```
$(System.DefaultWorkingDirectory)\\WebAppContent.
```

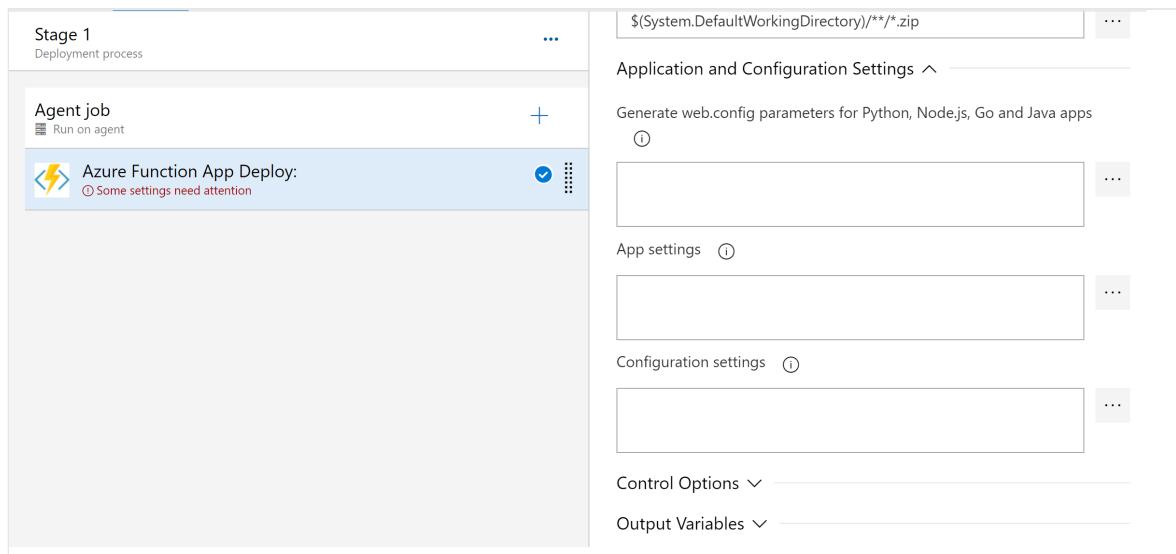
- b. Clear the Prepend root folder name to archive paths check box:



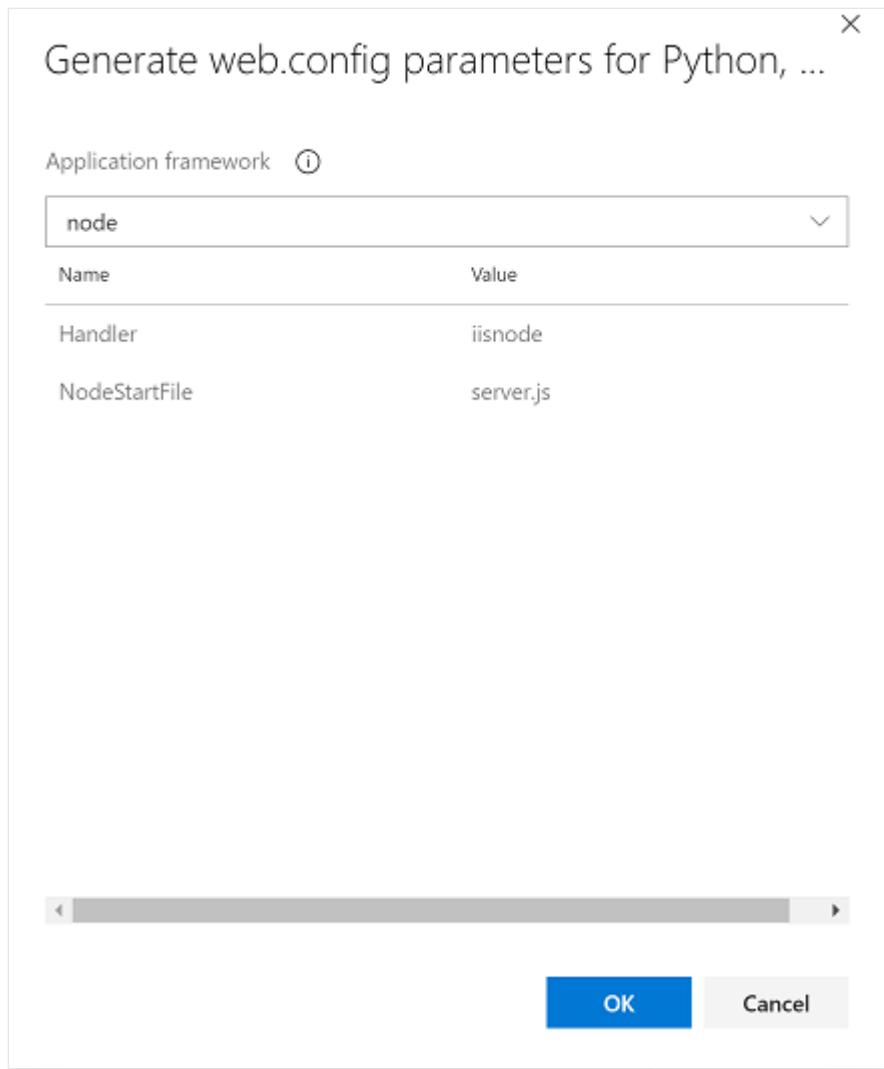
Function app deployment on Windows succeeds but the app doesn't work

This problem could occur if a web.config file isn't present in your app. You can either add a web.config file to your source or automatically generate one by using the **Application and Configuration Settings** of the task.

1. Select the task and go to **Generate web.config parameters for Python, Node.js, Go and Java apps**:



2. Select the More button (...) under **Generate web.config parameters for Python, Node.js, Go and Java apps** to edit the parameters:



3. Select your application type in the **Application framework** list.
4. Select **OK**. Doing so will populate the web.config parameters required to generate the web.config file.

FAQs

How should I configure my service connection?

This task requires an [Azure Resource Manager service connection](#).

How should I configure web job deployment with Application Insights?

When you're deploying to an App Service, if you have [Application Insights](#) configured and you've enabled `Remove additional files at destination`, you also need to enable `Exclude files from the App_Data folder`. Enabling this option keeps the Application

Insights extension in a safe state. This step is required because the Application Insights continuous WebJob is installed into the App_Data folder.

How should I configure my agent if it's behind a proxy while I'm deploying to App Service?

If your self-hosted agent requires a web proxy, you can inform the agent about the proxy during configuration. Doing so allows your agent to connect to Azure Pipelines or Azure DevOps Server through the proxy. [Learn more about running a self-hosted agent behind a web proxy](#).

I can't deploy to an internal App Service Environment by using an Azure Resource Manager service connection and a Microsoft-hosted agent

By design, a Microsoft-hosted agent won't work with an App Service Environment. Instead, you need to configure a private agent on a virtual machine that's in the same virtual network as the App Service Environment. Also, set a private DNS zone to enable communication between the resources.

Examples

Here's a sample YAML snippet that deploys Azure functions on Windows:

```
YAML

variables:
  azureSubscription: Contoso
  # To ignore SSL error, uncomment the below variable
  # VSTS_ARM_REST_IGNORE_SSL_ERRORS: true

steps:
- task: AzureFunctionApp@2
  displayName: Azure Function App Deploy
  inputs:
    azureSubscription: $(azureSubscription)
    appName: samplefunctionapp
    appType: functionApp
    package: $(System.DefaultWorkingDirectory)/**/*.zip
```

To deploy a function on Linux, add the `appType` parameter and set it to `appType: functionAppLinux`. If you don't specify a value, `functionApp` is the default.

To explicitly specify the deployment method as Zip Deploy, add the parameter `deploymentMethod: zipDeploy`. Another supported value for this parameter is `runFromPackage`. If you don't specify a value, `auto` is the default.

For a walkthrough that shows how to create a CI/CD pipeline, see [Build and deploy Java to Azure Functions](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.104.1 or greater
Task category	Deploy

AzureFunctionApp@1 - Azure Functions v1 task

Article • 09/26/2023

Update a function app with .NET, Python, JavaScript, PowerShell, Java based web applications.

Syntax

YAML

```
# Azure Functions Deploy v1
# Update a function app with .NET, Python, JavaScript, PowerShell, Java
based web applications.
- task: AzureFunctionApp@1
  inputs:
    azureSubscription: # string. Required. Azure subscription.
    appType: # 'functionApp' | 'functionAppLinux'. Required. App type.
    appName: # string. Required. Azure Functions App name.
    #deployToSlotOrASE: false # boolean. Deploy to Slot or App Service
    Environment. Default: false.
    #resourceGroupName: # string. Required when deployToSlotOrASE = true.
    Resource group.
    #slotName: 'production' # string. Required when deployToSlotOrASE =
    true. Slot. Default: production.
    package: '$(System.DefaultWorkingDirectory)/**/*.zip' # string.
    Required. Package or folder. Default:
    $(System.DefaultWorkingDirectory)/**/*.zip.
    #runtimeStack: # 'DOTNET|2.2' | 'DOTNET|3.1' | 'DOTNET|6.0' | 'JAVA|8' |
    'JAVA|11' | 'NODE|8' | 'NODE|10' | 'NODE|12' | 'NODE|14' | 'NODE|16' |
    'PYTHON|3.6' | 'PYTHON|3.7' | 'PYTHON|3.8' | 'PYTHON|3.9'. Optional. Use
    when appType = functionAppLinux. Runtime stack.
    #startUpCommand: # string. Optional. Use when appType =
    functionAppLinux. Startup command.
    # Application and Configuration Settings
    #customWebConfig: # string. Optional. Use when appType !=
    functionAppLinux && package NotEndsWith .war. Generate web.config parameters
    for Python, Node.js, Go and Java apps.
    #appSettings: # string. App settings.
    #configurationStrings: # string. Configuration settings.
    # Additional Deployment Options
    #deploymentMethod: 'auto' # 'auto' | 'zipDeploy' | 'runFromPackage'.
    Required when appType != functionAppLinux && appType != "" && package
    NotEndsWith .war && Package NotEndsWith .jar. Deployment method. Default:
    auto.
```

Inputs

`azureSubscription` - Azure subscription

`string`. Required.

Selects the Azure Resource Manager subscription for the deployment.

`appType` - App type

`string`. Required. Allowed values: `functionApp` (Function App on Windows),
`functionAppLinux` (Function App on Linux).

`appName` - Azure Functions App name

`string`. Required.

Enters or selects the name of an existing Azure Functions App. The Function Apps listed will be based on the selected app type.

`deployToSlotOrASE` - Deploy to Slot or App Service Environment

`boolean`. Default value: `false`.

Deploys to an existing deployment slot or Azure App Service Environment. For both targets, the task needs a Resource group name.

If the deployment target is a slot, it will default to the **production** slot. Any other existing slot name can also be provided.

If the deployment target is an Azure App Service Environment, leave the slot name as **production** and specify the Resource group name.

`resourceGroupName` - Resource group

`string`. Required when `deployToSlotOrASE = true`.

The Resource group name is required when the deployment target is either a deployment slot or an App Service Environment.

Enters or selects the Azure Resource group that contains the Azure App Service specified above.

slotName - Slot

`string`. Required when `deployToSlotOrASE = true`. Default value: `production`.

Enters or selects an existing slot, excluding the Production slot.

package - Package or folder

`string`. Required. Default value: `$(System.DefaultWorkingDirectory)/**/*.zip`.

The file path to the package or folder that contains App Service content generated by MSBuild, a compressed zip file, or a war file. Variables ([Build | Release](#)) and wildcards are supported. For example, `$(System.DefaultWorkingDirectory)/**/*.zip` or `$(System.DefaultWorkingDirectory)/**/*.war`.

runtimeStack - Runtime stack

`string`. Optional. Use when `appType = functionAppLinux`. Allowed values: `DOTNET|2.2` (`DOTNET|2.2` (functionapp v2)), `DOTNET|3.1` (`DOTNET|3.1` (functionapp v3)), `DOTNET|6.0` (`DOTNET|6.0` (functionapp v4)), `JAVA|8` (`JAVA|8` (functionapp v2/v3/v4)), `JAVA|11` (`JAVA|11` (functionapp v3/v4)), `NODE|8` (`NODE|8` (functionapp v2)), `NODE|10` (`NODE|10` (functionapp v2/v3)), `NODE|12` (`NODE|12` (functionapp v3)), `NODE|14` (`NODE|14` (functionapp v3/v4)), `NODE|16` (`NODE|16` (functionapp v4)), `PYTHON|3.6` (`PYTHON|3.6` (functionapp v2/v3)), `PYTHON|3.7` (`PYTHON|3.7` (functionapp v2/v3/v4)), `PYTHON|3.8` (`PYTHON|3.8` (functionapp v3/v4)), `PYTHON|3.9` (`PYTHON|3.9` (functionapp v3/v4)).

Learn about [supported runtime versions](#). Old values like `DOCKER|microsoft/azure-functions-*` are deprecated. New values are listed in the dropdown menu.

startUpCommand - Startup command

`string`. Optional. Use when `appType = functionAppLinux`.

Enters the start up command. For example:

```
dotnet run
```

```
dotnet filename.dll
```

customWebConfig - Generate web.config parameters for Python, Node.js, Go and Java apps

`string`. Optional. Use when `appType != functionAppLinux && package NotEndsWith .war`.

A standard Web.config will be generated and deployed to Azure App Service if the application does not have one. The values in web.config vary based on the application framework, and they can be edited. For example, for the node.js application, web.config will have a startup file and iis_node module values. This edit feature is only for the [generated web.config](#).

`appSettings` - App settings

`string`.

Enter the application settings using the syntax `-key value` (for example: `-Port 5000` `-RequestTimeout 5000` `-WEBSITE_TIME_ZONE`). Enclose values that contain spaces in double quotes (for example: `"Eastern Standard Time"`).

`configurationStrings` - Configuration settings

`string`.

Enter the configuration strings using the syntax `-key value` (for example: `-phpVersion 5.6` `-linuxFxVersion: node|6.11`). Enclose values that contain spaces in double quotes.

`deploymentMethod` - Deployment method

`string`. Required when `appType != functionAppLinux && appType != "" && packageNotEndsWith .war && PackageNotEndsWith .jar`. Allowed values: `auto` (Auto-detect), `zipDeploy` (Zip Deploy), `runFromPackage` (Zip Deploy with Run From Package). Default value: `auto`.

Chooses the [deployment method](#) for the app.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

AppServiceApplicationUrl

The application URL of the selected App Service.

Remarks

Use the Azure Function App task to deploy [functions](#) to Azure.

Deployment methods

Several deployment methods are available in this task. The default value is `auto`.

To change the package-based deployment option in a designer task, expand **Additional Deployment Options** and enable **Select Deployment Method**.

Based on the type of Azure App Service and Azure Pipelines agent, the task uses a suitable deployment technology. The deployment technologies used by tasks are as follows:

- [Kudu REST API](#)
- [Zip Deploy](#)
- [Run From Package](#)

By default, the task attempts to select the appropriate deployment technology based on the input package, App Service type, and agent OS.

- If a post-deployment script is provided, use Zip Deploy.
- If the App Service type is Web App on Linux, use Zip Deploy.
- If a .war file is provided, use War Deploy.
- If a .jar file is provided, use Run-From-Zip.
- For all other tasks, use Run From Package (via Zip Deploy).

On a non-Windows agent (for any App Service type), the task relies on the [Kudu REST API](#) to deploy the web app.

Kudu REST API

The [Kudu REST API](#) works on both Windows and Linux automation agents when the target is a Web App on Windows, a Web App on Linux (built-in source), or a function app. The task uses Kudu to copy files to the Azure App Service.

Zip Deploy

Zip Deploy creates a .zip deployment package from the chosen package or folder. It then deploys the file contents to the wwwroot folder of the App Service name function app in Azure. This option overwrites all existing content in the wwwroot folder. For more information, see [Zip deployment for Azure Functions](#).

Run From Package

Run From Package creates the same deployment package as Zip Deploy. Instead of deploying files to the wwwroot folder, the Functions runtime mounts the entire package. When you use this option, files in the wwwroot folder become read-only. For more information, see [Run your Azure Functions from a package file](#).

Troubleshooting

Error: Could not fetch access token for Azure. Verify if the Service Principal used is valid and not expired.

The task uses the service principal in the service connection to authenticate with Azure. If the service principal has expired or doesn't have permissions to the App Service, the task fails with this error. Verify the validity of the service principal used and that it's present in the app registration. For more information, see [Use role-based access control to manage access to your Azure subscription resources](#). [This blog post](#) also contains more information about using service principal authentication.

SSL error

If you want to use a certificate in App Service, the certificate must be signed by a trusted certificate authority. If your web app gives you certificate validation errors, you're probably using a self-signed certificate. Set a variable named `VSTS_ARM_REST_IGNORE_SSL_ERRORS` to the value `true` in the build or release pipeline to resolve the error.

A release hangs for long time and then fails

This problem could be the result of insufficient capacity in your App Service plan. To resolve this problem, you can scale up the App Service instance to increase available CPU, RAM, and disk space or try with a different App Service plan.

5xx error codes

If you're seeing a 5xx error, check the status of your Azure service [\[?\]](#).

Azure Function suddenly stopped working

Azure Functions may suddenly stop working if more than one year has passed since the last deployment. If you deploy with "RunFromPackage" in "deploymentMethod", a SAS with an expiration date of 1 year is generated and set as the value of "WEBSITE_RUN_FROM_PACKAGE" in the application configuration. Azure Functions uses this SAS to reference the package file for function execution, so if the SAS has expired, the function will not be executed. To resolve this issue, deploy again to generate a SAS with an expiration date of one year.

Error: No package found with specified pattern

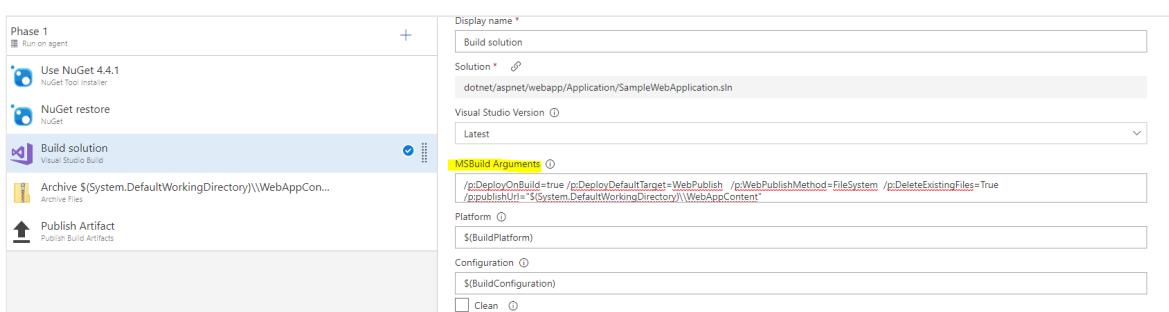
Check if the package mentioned in the task is published as an artifact in the build or a previous stage and downloaded in the current job.

Error: Publish using zip deploy option is not supported for msBuild package type

Web packages created via the MSBuild task (with default arguments) have a nested folder structure that can be deployed correctly only by Web Deploy. The publish-to-zip deployment option can't be used to deploy those packages. To convert the packaging structure, take these steps:

1. In the Build solution task, change the **MSBuild Arguments** to

```
/p:DeployOnBuild=true /p:DeployDefaultTarget=WebPublish  
/p:WebPublishMethod=FileSystem /p:DeleteExistingFiles=True  
/p:publishUrl="$(System.DefaultWorkingDirectory)\\WebAppContent":
```

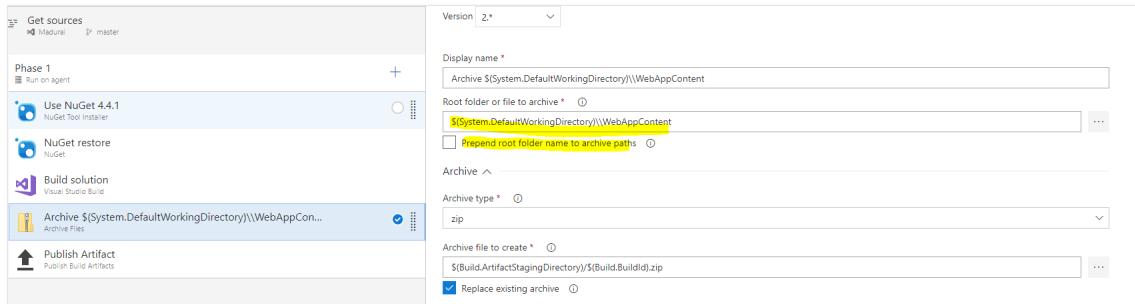


2. Add an Archive task and change the values as follows:

- a. Change **Root folder or file to archive** to

```
$(System.DefaultWorkingDirectory)\\WebAppContent.
```

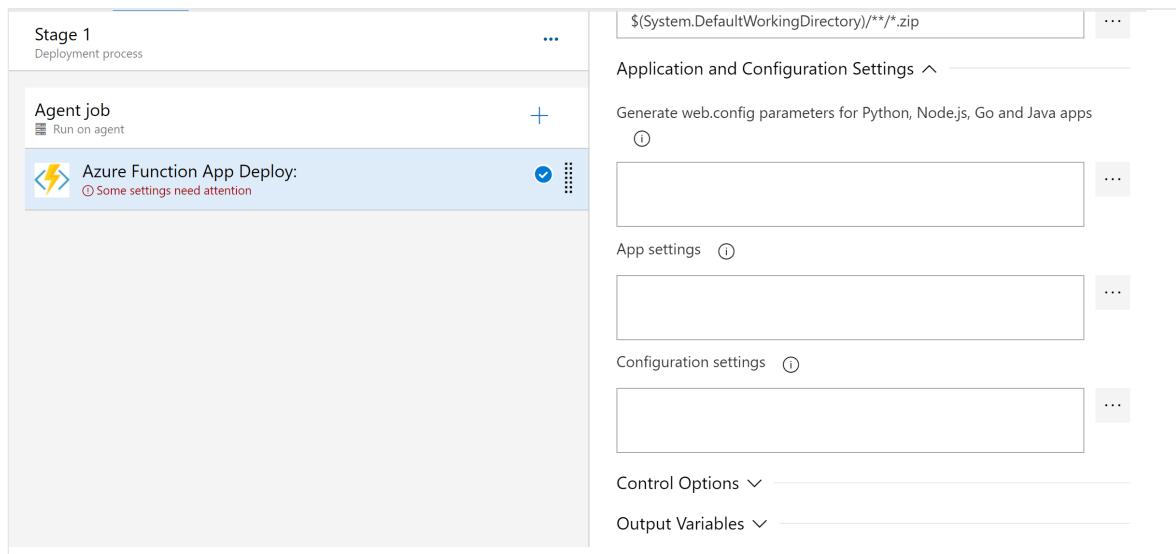
- b. Clear the Prepend root folder name to archive paths check box:



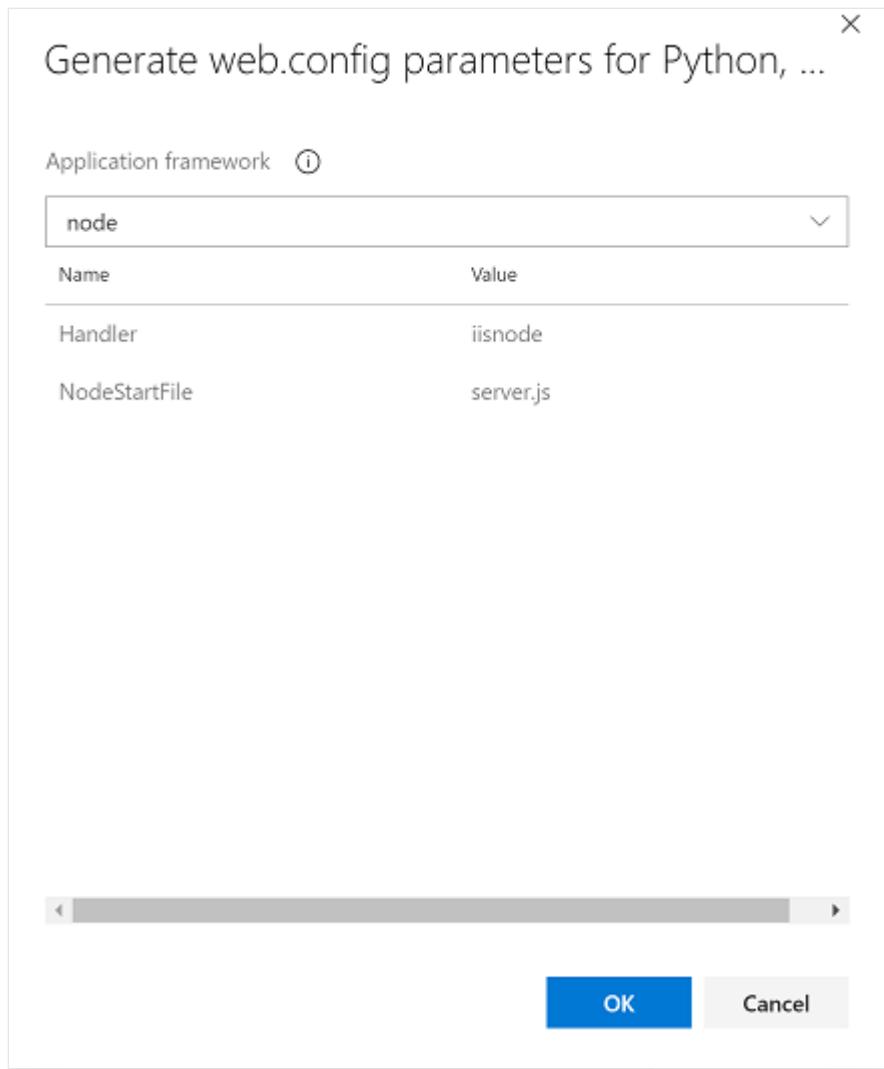
Function app deployment on Windows succeeds but the app doesn't work

This problem could occur if a web.config file isn't present in your app. You can either add a web.config file to your source or automatically generate one by using the **Application and Configuration Settings** of the task.

1. Select the task and go to **Generate web.config parameters for Python, Node.js, Go and Java apps**:



2. Select the More button (...) under **Generate web.config parameters for Python, Node.js, Go and Java apps** to edit the parameters:



3. Select your application type in the **Application framework** list.
4. Select **OK**. Doing so will populate the web.config parameters required to generate the web.config file.

FAQs

How should I configure my service connection?

This task requires an [Azure Resource Manager service connection](#).

How should I configure web job deployment with Application Insights?

When you're deploying to an App Service, if you have [Application Insights](#) configured and you've enabled `Remove additional files at destination`, you also need to enable `Exclude files from the App_Data folder`. Enabling this option keeps the Application

Insights extension in a safe state. This step is required because the Application Insights continuous WebJob is installed into the App_Data folder.

How should I configure my agent if it's behind a proxy while I'm deploying to App Service?

If your self-hosted agent requires a web proxy, you can inform the agent about the proxy during configuration. Doing so allows your agent to connect to Azure Pipelines or Azure DevOps Server through the proxy. [Learn more about running a self-hosted agent behind a web proxy](#).

I can't deploy to an internal App Service Environment by using an Azure Resource Manager service connection and a Microsoft-hosted agent

By design, a Microsoft-hosted agent won't work with an App Service Environment. Instead, you need to configure a private agent on a virtual machine that's in the same virtual network as the App Service Environment. Also, set a private DNS zone to enable communication between the resources.

Examples

Here's a sample YAML snippet that deploys Azure functions on Windows:

```
YAML

variables:
  azureSubscription: Contoso
  # To ignore SSL error, uncomment the below variable
  # VSTS_ARM_REST_IGNORE_SSL_ERRORS: true

steps:
- task: AzureFunctionApp@1
  displayName: Azure Function App Deploy
  inputs:
    azureSubscription: $(azureSubscription)
    appName: samplefunctionapp
    appType: functionApp
    package: $(System.DefaultWorkingDirectory)/**/*.zip
```

To deploy a function on Linux, add the `appType` parameter and set it to `appType: functionAppLinux`. If you don't specify a value, `functionApp` is the default.

To explicitly specify the deployment method as Zip Deploy, add the parameter `deploymentMethod: zipDeploy`. Another supported value for this parameter is `runFromPackage`. If you don't specify a value, `auto` is the default.

For a walkthrough that shows how to create a CI/CD pipeline, see [Build and deploy Java to Azure Functions](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.104.1 or greater
Task category	Deploy

AzureFunctionAppContainer@1 - Azure Functions for container v1 task

Article • 09/26/2023

Update a function app with a Docker container.

Syntax

YAML

```
# Azure Functions for container v1
# Update a function app with a Docker container.
- task: AzureFunctionAppContainer@1
  inputs:
    azureSubscription: # string. Required. Azure subscription.
    appName: # string. Required. App name.
    #deployToSlotOrASE: false # boolean. Deploy to Slot or App Service Environment. Default: false.
    #resourceGroupName: # string. Required when deployToSlotOrASE = true. Resource group.
    #slotName: 'production' # string. Required when deployToSlotOrASE = true. Slot. Default: production.
    imageName: # string. Required. Image name.
    #containerCommand: # string. Startup command.
    # Application and Configuration Settings
    #appSettings: # string. App settings.
    #configurationStrings: # string. Configuration settings.
```

Inputs

`azureSubscription` - Azure subscription

`string`. Required.

Selects the [Azure Resource Manager subscription](#) for the deployment.

`appName` - App name

`string`. Required.

The name of the Function App for Containers.

`deployToSlotOrASE` - Deploy to Slot or App Service Environment

`boolean`. Default value: `false`.

Set this input to `true` to deploy to an existing deployment slot or Azure App Service Environment. The task needs a Resource Group name for both targets. For the deployment slot option, the default deploys to the **production** slot, or you can specify any other existing slot name. If the deployment target is an Azure App Service Environment, leave the slot name as **production** and specify the Resource Group name.

`resourceGroupName` - Resource group

`string`. Required when `deployToSlotOrASE = true`.

The name of the Resource Group that contains the Function App for Containers.

`slotName` - Slot

`string`. Required when `deployToSlotOrASE = true`. Default value: `production`.

Enters or selects an existing slot, excluding the **production** slot.

`imageName` - Image name

`string`. Required.

A globally unique top-level domain name for your specific registry or namespace.

Note: A fully qualified image name will be of the format: `<registry or namespace><repository> <tag>`. For example, `myregistry.azurecr.io/nginx:latest`.

`containerCommand` - Startup command

`string`.

The startup command that executes after deployment. For example, `dotnet run dotnet filename.dll`.

`appSettings` - App settings

`string`.

Enter the application settings using the syntax `-key value` (for example: `-Port 5000 -RequestTimeout 5000 -WEBSITE_TIME_ZONE`). Enclose values that contain spaces in double

quotes (for example: "Eastern Standard Time").

`configurationStrings` - Configuration settings

`string`.

Enter the configuration strings using the syntax `-key value` (for example: `-phpVersion 5.6 -linuxFxVersion: node|6.11`). Enclose values that contain spaces in double quotes.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

`AppServiceApplicationUrl`

The application URL of the selected App Service.

Remarks

Use this task to deploy an Azure Function on Linux using a [custom image](#).

Error: Could not fetch access token for Azure. Verify if the Service Principal used is valid and not expired.

The task uses the service principal in the service connection to authenticate with Azure. If the service principal has expired or doesn't have permissions to the App Service, the task fails with this error. Verify the validity of the service principal used and that it's present in the app registration. For more information, see [Use role-based access control to manage access to your Azure subscription resources](#). This blog post [also](#) contains more information about using service principal authentication.

SSL error

If you want to use a certificate in App Service, the certificate must be signed by a trusted certificate authority. If your web app gives you certificate validation errors, you're probably using a self-signed certificate. Set a variable named `VSTS_ARM_REST_IGNORE_SSL_ERRORS` to the value `true` in the build or release pipeline to resolve the error.

A release hangs for long time and then fails

This problem could be the result of insufficient capacity in your App Service plan. To resolve this problem, you can scale up the App Service instance to increase available CPU, RAM, and disk space or try with a different App Service plan.

5xx error codes

If you're seeing a 5xx error, [check the status of your Azure service](#).

Azure Function suddenly stopped working

Azure Functions may suddenly stop working if more than one year has passed since the last deployment. If you deploy with "RunFromPackage" in "deploymentMethod", a SAS with an expiration date of 1 year is generated and set as the value of "WEBSITE_RUN_FROM_PACKAGE" in the application configuration. Azure Functions uses this SAS to reference the package file for function execution, so if the SAS has expired, the function will not be executed. To resolve this issue, deploy again to generate a SAS with an expiration date of one year.

How should I configure my service connection?

This task requires an [Azure Resource Manager service connection](#).

How should I configure web job deployment with Application Insights?

When you're deploying to an App Service, if you have [Application Insights](#) configured and you've enabled `Remove additional files at destination`, you also need to enable `Exclude files from the App_Data folder`. Enabling this option keeps the Application Insights extension in a safe state. This step is required because the Application Insights continuous WebJob is installed into the App_Data folder.

How should I configure my agent if it's behind a proxy while I'm deploying to App Service?

If your self-hosted agent requires a web proxy, you can inform the agent about the proxy during configuration. Doing so allows your agent to connect to Azure Pipelines or Azure DevOps Server through the proxy. [Learn more about running a self-hosted agent behind a web proxy](#).

Examples

This example deploys Azure Functions on Linux using containers:

YAML

```
variables:
  imageName: contoso.azurecr.io/azurefunctions-containers:${{build.buildId}}
  azureSubscription: Contoso
  # To ignore SSL error uncomment the following variable
  # VSTS_ARM_REST_IGNORE_SSL_ERRORS: true

steps:
- task: AzureFunctionAppContainer@1
  displayName: Azure Function App on Container deploy
  inputs:
    azureSubscription: $(azureSubscription)
    appName: functionappcontainers
    imageName: $(imageName)
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.104.1 or greater

Requirement	Description
Task category	Deploy

AzureIoTEdge@2 - Azure IoT Edge v2 task

Article • 09/26/2023

Use this task to build and deploy images quickly and efficiently to Azure IoT Edge.

This task supports custom variables. If you're not familiar with how to use variables in Pipelines, see [define variables](#).

ⓘ Note

This task does not support Azure Resource Manager authentication with workflow identity federation.

Syntax

YAML

```
# Azure IoT Edge v2
# Build and deploy an Azure IoT Edge image.
- task: AzureIoTEdge@2
  inputs:
    action: 'Build module images' # 'Build module images' | 'Push module images' | 'Generate deployment manifest' | 'Deploy to IoT Edge devices'.
    Required. Action. Default: Build module images.
    #deploymentFilePath:
    #'$(System.DefaultWorkingDirectory)/config/deployment.json' # string.
    Required when action == Deploy to IoT Edge devices. Deployment file.
    Default: $(System.DefaultWorkingDirectory)/config/deployment.json.
    #azureSubscription: # string. Alias: connectedServiceNameARM. Required
    when action == Deploy to IoT Edge devices. Azure subscription contains IoT Hub.
    #iothubname: # string. Required when action == Deploy to IoT Edge devices. IoT Hub name.
    #deviceOption: # 'Single Device' | 'Multiple Devices'. Required when
    action == Deploy to IoT Edge devices. Choose single/multiple device.
    #deviceId: # string. Required when deviceOption == Single Device. IoT
    Edge device ID.
    #targetcondition: # string. Required when deviceOption == Multiple
    Devices. IoT Edge device target condition.
    #containerregistrytype: 'Azure Container Registry' # 'Azure Container
    Registry' | 'Generic Container Registry'. Required when action = Push module
    images. Container registry type. Default: Azure Container Registry.
    #dockerRegistryConnection: # string. Alias: dockerRegistryEndpoint.
    Required when containerregistrytype = Generic Container Registry. Docker
    Registry Connection.
```

```

    #azureSubscriptionEndpoint: # string. Optional. Use when
    containerregistrytype = Azure Container Registry. Azure subscription.
    #azureContainerRegistry: # string. Required when containerregistrytype =
    Azure Container Registry. Azure Container Registry.
    #templateFilePath: 'deployment.template.json' # string. Required when
    action = Build module images || action = Push module images || action =
    Generate deployment manifest. .template.json file. Default:
    deployment.template.json.
    #defaultPlatform: 'amd64' # 'amd64' | 'windows-amd64' | 'arm32v7' |
    'arm64v8'. Required when action = Build module images || action = Push
    module images || action = Generate deployment manifest. Default platform.
    Default: amd64.
    #fillRegistryCredential: 'true' # 'true' | 'false'. Required when action
    = Push module images. Add registry credential to deployment manifest.
    Default: true.
    #deploymentManifestOutputPath:
    '$(System.DefaultWorkingDirectory)/config/deployment.json' # string.
    Required when action == Generate deployment manifest. Output path. Default:
    $(System.DefaultWorkingDirectory)/config/deployment.json.
    #validateGeneratedDeploymentManifest: 'false' # 'true' | 'false'.
    Required when action = Generate deployment manifest. Validate the schema of
    generated deployment manifest. Default: false.
    # Advanced
    #deploymentid: '$(System.TeamProject)-devops-deployment' # string.
    Required when action = Deploy to IoT Edge devices. IoT Edge deployment ID.
    Default: $(System.TeamProject)-devops-deployment.
    #priority: '0' # string. Required when action = Deploy to IoT Edge
    devices. IoT Edge deployment priority. Default: 0.
    # Advanced
    #bypassModules: # string. Optional. Use when action = Push module
    images. Bypass module(s).

```

Inputs

`action` - Action

`string`. Required. Allowed values: `Build module images`, `Push module images`, `Generate deployment manifest`, `Deploy to IoT Edge devices`. Default value: `Build module images`.

Selects an Azure IoT Edge action.

`Build module images` only builds modules (you can use it to check compilation errors).

`Push module images` pushes modules to the container registry.

`Deploy to IoT Edge devices` deploys the generated deployment file to IoT Hub. (We recommend putting the `Deploy` task in the release pipeline.)

deploymentFilePath - Deployment file

`string`. Required when `action == Deploy to IoT Edge devices`. Default value:

`$(System.DefaultWorkingDirectory)/config/deployment.json`.

Selects the deployment json file. If this task is in `release pipeline`, you need to set the location of the deployment file in artifact. (The default value works for most conditions.) If this task is in a build pipeline, you must specify the deployment manifest output path.

azureSubscription - Azure subscription contains IoT Hub

Input alias: `connectedServiceNameARM`. `string`. Required when `action == Deploy to IoT Edge devices`.

Selects an Azure subscription that contains IoT Hub.

iothubname - IoT Hub name

`string`. Required when `action == Deploy to IoT Edge devices`.

Selects the IoT Hub.

deploymentid - IoT Edge deployment ID

`string`. Required when `action = Deploy to IoT Edge devices`. Default value:

`$(System.TeamProject)-devops-deployment`.

Inputs the IoT Edge Deployment ID. If the ID already exists, it will be overridden. This has up to 128 lowercase letters and numbers, and the following characters are allowed: `-:+%_#*?!(),=@;'`. For more information, see [Azure IoT Edge deployment](#).

priority - IoT Edge deployment priority

`string`. Required when `action = Deploy to IoT Edge devices`. Default value: `0`.

Sets the `priority` to a positive integer to resolve deployment conflicts. When this task is targeted by multiple deployments, a device will use the one with the highest priority or, in the case of two deployments with the same priority, the latest creation time. For more information, see [Azure IoT Edge deployment](#).

deviceOption - Choose single/multiple device

`string`. Required when `action == Deploy to IoT Edge devices`. Allowed values: `Single Device`, `Multiple Devices`.

According to tags, chooses to deploy to single or multiple devices.

deviceId - IoT Edge device ID

`string`. Required when `deviceOption == Single Device`.

Inputs the IoT Edge `device ID`.

targetcondition - IoT Edge device target condition

`string`. Required when `deviceOption == Multiple Devices`.

Inputs the `target condition` of devices you would like to deploy. Do not use double quotes. Example: `tags.building=9` and `tags.environment='test'`. For more information, see [Azure IoT Edge deployment](#).

containerregistrytype - Container registry type

`string`. Required when `action = Push module images`. Allowed values: `Azure Container Registry`, `Generic Container Registry`. Default value: `Azure Container Registry`.

Selects a `Container Registry Type`. `Azure Container Registry` is for ACR, and `Generic Container Registry` is for generic registries including docker hub.

dockerRegistryConnection - Docker Registry Connection

Input alias: `dockerRegistryEndpoint`. `string`. Required when `containerregistrytype = Generic Container Registry`.

Selects a generic Docker registry connection. This is required for build and push.

azureSubscriptionEndpoint - Azure subscription

`string`. Optional. Use when `containerregistrytype = Azure Container Registry`.

Selects an Azure subscription.

azureContainerRegistry - Azure Container Registry

`string`. Required when `containerregistrytype = Azure Container Registry`.

Selects an Azure Container Registry.

templateFilePath - .template.json file
string. Required when action = Build module images || action = Push module images
|| action = Generate deployment manifest. Default value: deployment.template.json.

The path of Azure IoT Edge solution .template.json. This file defines the modules and routes in Azure IoT Edge solution. The file name must end with .template.json.

defaultPlatform - Default platform

string. Required when action = Build module images || action = Push module images
|| action = Generate deployment manifest. Allowed values: amd64, windows-amd64,
arm32v7, arm64v8. Default value: amd64.

In your .template.json, you can leave the modules platform unspecified. For these modules, the default platform will be used.

fillRegistryCredential - Add registry credential to deployment manifest

string. Required when action = Push module images. Allowed values: true, false.
Default value: true.

Adds the registry credential for pushing docker images to the deployment manifest.

deploymentManifestOutputPath - Output path

string. Required when action == Generate deployment manifest. Default value:
\$(System.DefaultWorkingDirectory)/config/deployment.json.

The output path of the generated deployment manifest.

validateGeneratedDeploymentManifest - Validate the schema of generated deployment manifest

string. Required when action = Generate deployment manifest. Allowed values: true,
false. Default value: false.

Fail this step if the generated deployment manifest does not pass schema validation.
Search Azure IoT Edge deployment in [JSON Schema Store](#) to find latest schema.

bypassModules - Bypass module(s)

string. Optional. Use when action = Push module images.

Selects the module(s) that you do not need to build or push in `.template.json`, specifies the module names, and separates them with commas. Example: if you have `SampleModule1` and `SampleModule2` in your `.template.json` and you want to only build or push `SampleModule1`, then you set the bypass modules as `SampleModule2`. Leave this empty if you would like to build all the modules in `.template.json`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

`DEPLOYMENT_FILE_PATH`

This is the path of generated deployment file.

Remarks

Use this task to build, test, and deploy applications quickly and efficiently to Azure IoT Edge.

This task supports custom variables. If you're not familiar with how to use variables in Pipelines, see [Define variables](#).

Examples

Build module images

The following YAML example builds module images:

YAML

```
- task: AzureIoTEdge@2
  displayName: AzureIoTEdge - Build module images
  inputs:
    action: Build module images
```

```
templateFilePath: deployment.template.json
defaultPlatform: amd64
```

Push module images

The following YAML example pushes module images:

YAML

```
variables:
  azureSubscriptionEndpoint: Contoso
  azureContainerRegistry: contoso.azurecr.io

steps:
- task: AzureIoTEdge@2
  displayName: AzureIoTEdge - Push module images
  inputs:
    action: Push module images
    containerregistrytype: Azure Container Registry
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureContainerRegistry: {"loginServer": "$(azureContainerRegistry)"}
    templateFilePath: deployment.template.json
    defaultPlatform: amd64
    fillRegistryCredential: true
```

Generate deployment manifest

The following YAML example creates a deployment manifest based on the template file:

YAML

```
steps:
- task: AzureIoTEdge@2
  displayName: AzureIoTEdge - Generate deployment manifest
  inputs:
    action: Generate deployment manifest
    templateFilePath: deployment.template.json
    defaultPlatform: amd64
    deploymentManifestOutputPath:
      $(System.DefaultWorkingDirectory)/config/deployment.json
    validateGeneratedDeploymentManifest: false
```

Deploy to IoT Edge devices

The following YAML example deploys module images:

YAML

```
steps:
- task: AzureIoTEdge@2
  displayName: 'Azure IoT Edge - Deploy to IoT Edge devices'
  inputs:
    action: 'Deploy to IoT Edge devices'
    deploymentFilePath:
      $(System.DefaultWorkingDirectory)/config/deployment.json
    azureSubscription: $(azureSubscriptionEndpoint)
    iothubname: iothubname
    deploymentid: '$(System.TeamProject)-devops-deployment'
    priority: '0'
    deviceOption: 'Single Device'
    deviceId: deviceId
```

More examples

For step-by-step examples of how to use these actions in Azure Pipelines, see the following articles:

- [Continuous integration and continuous deployment to Azure IoT Edge devices \(YAML\)](#)
- [Continuous integration and continuous deployment to Azure IoT Edge devices \(classic editor\)](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Build

AzureKeyVault@2 - Azure Key Vault v2 task

Article • 09/26/2023

Use this task to download secrets, such as authentication keys, storage account keys, data encryption keys, .PFX files, and passwords from an [Azure Key Vault](#) instance. The task can be used to fetch the latest values of all or a subset of secrets from the vault and set them as variables that can be used in subsequent tasks of a pipeline. The task is Node-based and works with agents on Linux, macOS, and Windows.

Syntax

YAML

```
# Azure Key Vault v2
# Download Azure Key Vault secrets.
- task: AzureKeyVault@2
  inputs:
    azureSubscription: # string. Alias: ConnectedServiceName. Required.
    Azure subscription.
    KeyVaultName: # string. Required. Key vault.
    SecretsFilter: '*' # string. Required. Secrets filter. Default: *.
    #RunAsPreJob: false # boolean. Make secrets available to whole job.
    Default: false.
```

Inputs

azureSubscription - Azure subscription

Input alias: `ConnectedServiceName`. `string`. Required.

Select the service connection for the Azure subscription containing the Azure Key Vault instance, or create a new connection. [Learn more](#).

KeyVaultName - Key vault

`string`. Required.

The name of the Azure Key Vault that contains the secrets to download.

SecretsFilter - Secrets filter

`string`. Required. Default value: `*`.

Downloads secret names according to the entered value. The value can be the default value to download all secrets from the selected key vault, or a comma-separated list of secret names.

RunAsPreJob - Make secrets available to whole job

`boolean`. Default value: `false`.

Runs the task before the job execution begins. Exposes secrets to all tasks in the job, not just tasks that follow this one.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

What's new in Version 2.0: Added support for %3B, %5D in secrets.

Use this task to download secrets, such as authentication keys, storage account keys, data encryption keys, .PFX files, and passwords from an [Azure Key Vault](#) instance. The task can be used to fetch the latest values of all or a subset of secrets from the vault and set them as variables that can be used in subsequent tasks of a pipeline. The task is Node-based and works with agents on Linux, macOS, and Windows.

I get a `forbidden` error on pipelines at the point of getting credentials from Azure Key Vault

This occurs if the required permissions are missing in the Azure key vault. To resolve the issue, [add an access policy with the correct permissions](#).

Prerequisites

The task has the following Prerequisites:

- An Azure subscription linked to Azure Pipelines or Team Foundation Server using the [Azure Resource Manager service connection](#).
- An [Azure Key Vault](#) containing the secrets.

You can create a key vault:

- In the [Azure portal](#)
- By using [Azure PowerShell](#)
- By using the [Azure CLI](#)

Add secrets to a key vault:

- By using the PowerShell cmdlet [Set-AzureKeyVaultSecret](#). If the secret does not exist, this cmdlet creates it. If the secret already exists, this cmdlet creates a new version of that secret.
- By using the Azure CLI. To add a secret to a key vault, for example a secret named **SQLPassword** with the value **PlaceholderPassword**, type:

Azure CLI

```
az keyvault secret set --vault-name 'ContosoKeyVault' --name  
'SQLPassword' --value 'PlaceholderPassword'
```

When you want to access secrets:

- Ensure the Azure service connection has at least **Get** and **List** permissions on the vault. You can set these permissions in the [Azure portal](#):
 - Open the **Settings** blade for the vault, choose **Access policies**, then **Add new**.
 - In the **Add access policy** blade, choose **Select principal** and select the service principal for your client account.
 - In the **Add access policy** blade, choose **Secret permissions** and ensure that **Get** and **List** are checked (ticked).
 - Choose **OK** to save the changes.

ⓘ Note

If you're using a Microsoft-hosted agent, you must add the IP range of the Microsoft-hosted agent to your firewall. Get the weekly list of IP ranges from the [weekly JSON file](#), which is published every Wednesday. The new IP ranges

become effective the following Monday. For more information, see [Microsoft-hosted agents](#). To find the IP ranges that are required for your Azure DevOps organization, learn how to [identify the possible IP ranges for Microsoft-hosted agents](#).

Note

Values are retrieved as strings. For example, if there is a secret named `connectionString`, a task variable `connectionString` is created with the latest value of the respective secret fetched from Azure key vault. This variable is then available in subsequent tasks.

If the value fetched from the vault is a certificate (for example, a PFX file), the task variable will contain the contents of the PFX in string format. You can use the following PowerShell code to retrieve the PFX file from the task variable:

PowerShell

```
$kvSecretBytes = [System.Convert]::FromBase64String($"$(PfxSecret)")  
$certCollection = New-Object  
System.Security.Cryptography.X509Certificates.X509Certificate2Collection  
$certCollection.Import($kvSecretBytes,$null,  
[System.Security.Cryptography.X509Certificates.X509KeyStorageFlags]::Exportable)
```

If the certificate file will be stored locally on the machine, it is good practice to encrypt it with a password:

PowerShell

```
#Get the file created  
$password = 'your password'  
$protectedCertificateBytes =  
$certCollection.Export([System.Security.Cryptography.X509Certificates.X509Contentype]::Pkcs12, $password)  
$pfxPath = [Environment]::GetFolderPath("Desktop") + "\MyCert.pfx"  
[System.IO.File]::WriteAllBytes($pfxPath, $protectedCertificateBytes)
```

For more information, see [Get started with Azure Key Vault certificates](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.182.1 or greater
Task category	Deploy

AzureKeyVault@1 - Azure Key Vault v1 task

Article • 09/26/2023

Use this task to download secrets, such as authentication keys, storage account keys, data encryption keys, .PFX files, and passwords from an [Azure Key Vault](#) instance. The task can be used to fetch the latest values of all or a subset of secrets from the vault and set them as variables that can be used in subsequent tasks of a pipeline. The task is Node-based and works with agents on Linux, macOS, and Windows.

Syntax

YAML

```
# Azure Key Vault v1
# Download Azure Key Vault secrets.
- task: AzureKeyVault@1
  inputs:
    azureSubscription: # string. Alias: ConnectedServiceName. Required.
    Azure subscription.
    KeyVaultName: # string. Required. Key vault.
    SecretsFilter: '*' # string. Required. Secrets filter. Default: *.
    #RunAsPreJob: false # boolean. Make secrets available to whole job.
    Default: false.
```

Inputs

azureSubscription - Azure subscription

Input alias: `ConnectedServiceName`. `string`. Required.

The service connection for the Azure subscription that either contains the Azure Key Vault instance or creates a new connection. Learn more about [connecting to Azure](#).

KeyVaultName - Key vault

`string`. Required.

The name of the Azure Key Vault that contains the secrets to download.

SecretsFilter - Secrets filter

`string`. Required. Default value: `*`.

Downloads secret names according to the entered value. The value can be the default value to download all secrets from the selected key vault, or a comma-separated list of secret names.

RunAsPreJob - Make secrets available to whole job

`boolean`. Default value: `false`.

Runs the task before the job execution begins. Exposes secrets to all tasks in the job, not just tasks that follow this one.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Works with cross-platform agents (Linux, macOS, or Windows).

There is a newer version of the Archive Files task available.

- [Azure Key Vault v2](#)

I get a `forbidden` error on pipelines at the point of getting credentials from Azure Key Vault

This occurs if the required permissions are missing in the Azure key vault. To resolve the issue, [add an access policy with the correct permissions](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.0.0 or greater
Task category	Deploy

See also

- [Azure Key Vault v2](#)

AzureLoadTest@1 - Azure Load Testing v1 task

Article • 09/26/2023

Automate performance regression testing with Azure Load Testing.

Syntax

YAML

```
# Azure Load Testing v1
# Automate performance regression testing with Azure Load Testing.
- task: AzureLoadTest@1
  inputs:
    azureSubscription: # string. Alias: connectedServiceNameARM. Required.
    Azure subscription.
    loadTestConfigFile: # string. Required. Load Test File.
    resourceGroup: # string. Required. Load Test Resource Group.
    loadTestResource: # string. Required. Load Test Resource Name.
    #loadTestRunName: # string. Load Test Run Name.
    #loadTestRunDescription: # string. Load Test Run Description.
    #secrets: # string. Secrets.
    #env: # string. env.
```

Inputs

`azureSubscription` - Azure subscription

Input alias: `connectedServiceNameARM`. `string`. Required.

Selects an Azure Resource Manager subscription to run the load test.

`loadTestConfigFile` - Load Test File

`string`. Required.

The path to the load test YAML configuration file relative from the repo root. See [Test configuration YAML reference](#). The path must be fully qualified or relative to the default working directory.

`resourceGroup` - Load Test Resource Group

`string`. Required.

Enters or selects the Azure Resource Group that contains the Load test resource.

loadTestResource - Load Test Resource Name

`string`. Required.

Enters or selects the name of an existing Azure Load Testing resource.

loadTestRunName - Load Test Run Name

`string`.

Custom name for the load test run.

loadTestRunDescription - Load Test Run Description

`string`.

Custom description for the load test run.

secrets - Secrets

`string`.

An array of JSON objects that consist of the name and value for each secret. The name should match the secret name used in the Apache JMeter test script. Add or update the secret parameters using the json syntax as shown in the following example.

JSON

```
[  
  {  
    "name": "key1",  
    "value": ${secret1}  
  },  
  {  
    "name": "key2",  
    "value": ${secret2}  
  }]
```

env - env

`string`.

An array of JSON objects that consist of the name and value for each environment variable. The name should match the variable name used in the Apache JMeter test script. Add or update the environment variables using the JSON syntax as shown in the following example.

```
JSON

[  
  {  
    "name": "env1",  
    "value": "value1"  
  },  
  {  
    "name": "env2",  
    "value": "value2"  
  }  
]
```

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to run an Apache JMeter script by using Azure Load Testing. Azure Load Testing is a fully managed load testing service that enables you to generate high-scale load.

The task succeeds if the load test finishes successfully and all [test criteria](#) pass.

Although Azure PowerShell isn't listed in the demands for `AzureLoadTest@1`, the agent must have Azure PowerShell installed. Azure PowerShell is installed on [Windows and Linux hosted agent images](#).

Note

AzureLoadTest@1 is part of the Azure Load Testing marketplace extension. For more information on installing and using this task, see [Identify performance regressions with Azure Load Testing and Azure Pipelines](#).

Examples

For an example using this task, see the Azure Load Testing documentation article [Continuous regression testing with Azure Pipelines](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Azure Pipelines

See also

For more information about using this task, see the Azure Load Testing documentation article [Continuous regression testing with Azure Pipelines](#).

AzureMonitorAlerts@0 - Azure Monitor alerts (Deprecated) v0 task

Article • 09/26/2023

Configure alerts on available metrics for an Azure resource (Deprecated).

ⓘ Note

This task does not support Azure Resource Manager authentication with workflow identity federation.

Syntax

YAML

```
# Azure Monitor alerts (Deprecated) v0
# Configure alerts on available metrics for an Azure resource (Deprecated).
- task: AzureMonitorAlerts@0
  inputs:
    azureSubscription: # string. Alias: ConnectedServiceName. Required.
    Azure Subscription.
    ResourceGroupName: # string. Required. Resource Group.
    ResourceType: 'Microsoft.Insights/components' #
    'Microsoft.Insights/components' | 'Microsoft.Web/sites' |
    'Microsoft.Storage/storageAccounts' | 'Microsoft.Compute/virtualMachines'.
    Required. Resource Type. Default: Microsoft.Insights/components.
    resourceName: # string. Required. Resource name.
    AlertRules: # string. Required. Alert rules.
    # Notify via email
    #NotifyServiceOwners: # boolean. Subscription owners, contributors and
    readers.
    #NotifyEmails: # string. Additional administrator emails.
```

Inputs

azureSubscription - Azure Subscription

Input alias: `ConnectedServiceName`. `string`. Required.

Selects the Azure Resource Manager subscription.

Note: To configure new service connection, select the Azure subscription from the list and click `Authorize`.

If your subscription is not listed or if you want to use an existing service principal, you can setup an Azure service connection using the `Add` or `Manage` button.

ResourceGroupName - Resource Group

`string`. Required.

Selects the Azure Resource Group that contains the Azure resource where you want to configure an alert.

ResourceType - Resource Type

`string`. Required. Allowed values: `Microsoft.Insights/components` (Application Insights), `Microsoft.Web/sites` (App Services), `Microsoft.Storage/storageAccounts` (Storage Account), `Microsoft.Compute/virtualMachines` (Virtual Machines). Default value: `Microsoft.Insights/components`.

Selects the Azure resource type.

ResourceName - Resource name

`string`. Required.

Selects the name of the Azure resource where you want to configure an alert.

AlertRules - Alert rules

`string`. Required.

The list of Azure monitor alerts that are configured on the selected Azure resource.

To add or modify alerts, click the `...` button.

NotifyServiceOwners - Subscription owners, contributors and readers

`boolean`.

Optional. Sends an email notification to everyone who has access to the specified resource group.

NotifyEmails - Additional administrator emails

`string`.

Optional. Add email addresses separated by semicolons (;) if you want to include additional email notification recipients. This feature can be incorporated whether or not the "subscription owners..." box is checked.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to configure alerts on available metrics for an Azure resource.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.111.0 or greater
Task category	Deploy

AzureNLBManagement@1 - Azure Network Load Balancer v1 task

Article • 09/26/2023

Use this task to connect or disconnect an Azure virtual machine's network interface to a load balancer's back-end address pool.

ⓘ Note

This task does not support Azure Resource Manager authentication with workflow identity federation.

Syntax

YAML

```
# Azure Network Load Balancer v1
# Connect or disconnect an Azure virtual machine's network interface to a
Load Balancer's back end address pool.
- task: AzureNLBManagement@1
  inputs:
    azureSubscription: # string. Alias: ConnectedServiceName. Required.
    Azure Subscription.
    ResourceGroupName: # string. Required. Resource Group.
    LoadBalancer: # string. Required. Load Balancer Name.
    Action: # 'Disconnect' | 'Connect'. Required. Action.
```

Inputs

azureSubscription - Azure Subscription

Input alias: `ConnectedServiceName`. `string`. Required.

Specifies the Azure Resource Manager subscription for the deployment.

ResourceGroupName - Resource Group

`string`. Required.

Specifies the resource group name.

LoadBalancer - Load Balancer Name

`string`. Required.

Specifies or enters the load balancer's name.

Action - Action

`string`. Required. Allowed values: `Disconnect` (Disconnect Primary Network Interface), `Connect` (Connect Primary Network Interface).

The action you'd like to perform.

Disconnect: Removes the virtual machine's primary network interface from the load balancer's back-end pool so it stops receiving network traffic.

Connect: Adds the virtual machine's primary network interface to load balancer back-end pool so it starts receiving network traffic based on the load balancing rules for the load balancer resource.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to connect or disconnect an Azure virtual machine's network interface to a load balancer's address pool.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	DeploymentGroup

Requirement	Description
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.95.0 or greater
Task category	Utility

AzurePowerShell@5 - Azure PowerShell v5 task

Article • 09/26/2023

Use this task to run a PowerShell script within an Azure environment. The Azure context is authenticated with the provided Azure Resource Manager service connection.

ⓘ Note

By default, Azure PowerShell v5 uses PowerShell Core for Linux agents and Windows PowerShell for Windows agents. To use the latest version of PowerShell on Windows agents, set the `pwsh` parameter to `true`. PowerShell Core will then be used instead.

Syntax

YAML

```
# Azure PowerShell v5
# Run a PowerShell script within an Azure environment.
- task: AzurePowerShell@5
  inputs:
    azureSubscription: # string. Alias: ConnectedServiceNameARM. Required.
    Azure Subscription.
    #ScriptType: 'FilePath' # 'FilePath' | 'InlineScript'. Script Type.
    Default: FilePath.
    #ScriptPath: # string. Optional. Use when ScriptType = FilePath. Script
    Path.
    #Inline: # string. Optional. Use when ScriptType = InlineScript. Inline
    Script.
    #ScriptArguments: # string. Optional. Use when ScriptType = FilePath.
    Script Arguments.
    #errorActionPreference: 'stop' # 'stop' | 'continue' |
    'silentlyContinue'. ErrorActionPreference. Default: stop.
    #FailOnStandardError: false # boolean. Fail on Standard Error. Default:
    false.
    # Azure PowerShell version options
    #azurePowerShellVersion: 'OtherVersion' # 'LatestVersion' |
    'OtherVersion'. Alias: TargetAzurePs. Azure PowerShell Version. Default:
    OtherVersion.
    preferredAzurePowerShellVersion: # string. Alias: CustomTargetAzurePs.
    Required when TargetAzurePs = OtherVersion. Preferred Azure PowerShell
    Version.
    # Advanced
    #pwsh: false # boolean. Use PowerShell Core. Default: false.
```

```
#validateScriptSignature: false # boolean. Optional. Use when ScriptType  
= FilePath. Validate script signature. Default: false.  
#workingDirectory: # string. Working Directory.
```

Inputs

azureSubscription - Azure Subscription

Input alias: `ConnectedServiceNameARM`. `string`. Required.

The Azure Resource Manager subscription to configure before running PowerShell.

ScriptType - Script Type

`string`. Allowed values: `FilePath` (Script File Path), `InlineScript` (Inline Script). Default value: `FilePath`.

The type of the script: file path or inline.

ScriptPath - Script Path

`string`. Optional. Use when `ScriptType = FilePath`.

The path of the script. This should be a fully qualified path or one relative to the default working directory.

Inline - Inline Script

`string`. Optional. Use when `ScriptType = InlineScript`. Default value: `# You can write your azure powershell scripts inline here. \n# You can also pass predefined and custom variables to this script using arguments.`

Specifies the script to execute. The maximum supported inline script length is 5000 characters. Use a script from a file if you want to use a longer script.

ScriptArguments - Script Arguments

`string`. Optional. Use when `ScriptType = FilePath`.

The additional parameters to pass to PowerShell. These can be either ordinal or named parameters. Not applicable for an inline script option.

errorActionPreference - ErrorActionPreference

`string`. Allowed values: `stop`, `continue`, `silentlyContinue`. Default value: `stop`.

Selects the value of the `ErrorActionPreference` variable for executing the script.

FailOnStandardError - Fail on Standard Error

`boolean`. Default value: `false`.

When this is true, this task will fail if any errors are written to the error pipeline or if any data is written to the standard error stream.

azurePowerShellVersion - Azure PowerShell Version

Input alias: `TargetAzurePs`. `string`. Allowed values: `LatestVersion` (Latest installed version), `OtherVersion` (Specify other version). Default value: `OtherVersion`.

In case of hosted agents, the supported Azure PowerShell Versions are `1.0.0`, `1.6.0`, `2.3.2`, `2.6.0`, and `3.1.0` (Hosted VS2017 Queue). To pick the latest version available on the agent, select `LatestVersion` (Latest installed version).

For private agents you can specify a preferred version of Azure PowerShell using `OtherVersion` (Specify other version).

preferredAzurePowerShellVersion - Preferred Azure PowerShell Version

Input alias: `CustomTargetAzurePs`. `string`. Required when `TargetAzurePs = OtherVersion`.

The preferred Azure PowerShell Version needs to be a proper semantic version eg. `1.2.3`. Regex like `2.*,2.3.*` is not supported. The Hosted VS2017 Pool currently supports Az module versions `1.0.0`, `1.6.0`, `2.3.2`, `2.6.0`, and `3.1.0`.

pwsh - Use PowerShell Core

`boolean`. Default value: `false`.

If this is true, then tasks running on Windows agents will use `pwsh.exe` from your path instead of `powershell.exe`.

validateScriptSignature - Validate script signature

`boolean`. Optional. Use when `ScriptType = FilePath`. Default value: `false`.

If this is true, then the task will first check to make sure specified script is signed and valid before executing it.

`workingDirectory` - Working Directory

`string`.

The working directory where the script is run.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Troubleshooting

Script worked locally, but failed in the pipeline

This typically occurs when the service connection used in the pipeline has insufficient permissions to run the script. Locally, the script runs with your credentials and would succeed as you may have the required access.

To resolve this issue, ensure the service principle/ authentication credentials have the required permissions. For more information, see [Use Role-Based Access Control to manage access to your Azure subscription resources](#).

Error: Could not find the modules: '<module name>' with Version: '<version>'. If the module was recently installed, retry after restarting the Azure Pipelines task agent

Azure PowerShell task uses Azure/AzureRM/Az PowerShell Module to interact with Azure Subscription. This issue occurs when the PowerShell module is not available on the Hosted Agent. Hence, for a particular task version, *Preferred Azure PowerShell*

version must be specified in the [Azure PowerShell version options](#) from the list of available versions. The installed software can be found in the [Software](#) table in [Microsoft-hosted agents](#).

Service Connection Issues

To troubleshoot issues related to service connections, see [Service Connection troubleshooting](#).

Examples

The following example shows how to invoke a script from a file and pass script arguments to it.

```
yml

- task: AzurePowerShell@5
  inputs:
    azureSubscription: my-arm-service-connection
    scriptType: filePath
    scriptPath: $(Build.SourcesDirectory)\myscript.ps1
    scriptArguments:
      -Arg1 val1 `
      -Arg2 val2 `
      -Arg3 val3
    azurePowerShellVersion: latestVersion
    pwsh: true
```

The following arguments shows how to invoke an inline script.

```
yml

- task: AzurePowerShell@5
  inputs:
    azureSubscription: 'Azure subscription connection placeholder'
    azurePowerShellVersion: LatestVersion
    ScriptType: 'InlineScript'
    Inline: |
      # You can write your azure powershell scripts inline here.
      # You can also pass predefined and custom variables to this script
      using arguments
      Write-Host 'Hello'
      Write-Host 'World!'
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.115.0 or greater
Task category	Deploy

AzurePowerShell@4 - Azure PowerShell v4 task

Article • 09/26/2023

Use this task to run a PowerShell script within an Azure environment. The Azure context is authenticated with the provided Azure Resource Manager service connection.

Syntax

YAML

```
# Azure PowerShell v4
# Run a PowerShell script within an Azure environment.
- task: AzurePowerShell@4
  inputs:
    azureSubscription: # string. Alias: ConnectedServiceNameARM. Required.
    Azure Subscription.
    #ScriptType: 'FilePath' # 'FilePath' | 'InlineScript'. Script Type.
    Default: FilePath.
    #ScriptPath: # string. Optional. Use when ScriptType = FilePath. Script Path.
    #Inline: # string. Optional. Use when ScriptType = InlineScript. Inline Script.
    #ScriptArguments: # string. Optional. Use when ScriptType = FilePath.
    Script Arguments.
    #errorActionPreference: 'stop' # 'stop' | 'continue' |
    'silentlyContinue'. ErrorActionPreference. Default: stop.
    #FailOnStandardError: false # boolean. Fail on Standard Error. Default:
    false.
    #RestrictContextToCurrentTask: false # boolean. Restrict scope of
    context to current task. Default: false.
    # Azure PowerShell version options
    #azurePowerShellVersion: 'OtherVersion' # 'LatestVersion' |
    'OtherVersion'. Alias: TargetAzurePs. Azure PowerShell Version. Default:
    OtherVersion.
    preferredAzurePowerShellVersion: # string. Alias: CustomTargetAzurePs.
    Required when TargetAzurePs = OtherVersion. Preferred Azure PowerShell
    Version.
    # Advanced
    #pwsh: false # boolean. Use PowerShell Core. Default: false.
    #validateScriptSignature: false # boolean. Optional. Use when ScriptType
    = FilePath. Validate script signature. Default: false.
    #workingDirectory: # string. Working Directory.
```

Inputs

azureSubscription - Azure Subscription

Input alias: `ConnectedServiceNameARM`. `string`. Required.

The Azure Resource Manager subscription to configure before running PowerShell.

ScriptType - Script Type

`string`. Allowed values: `FilePath` (Script File Path), `InlineScript` (Inline Script). Default value: `FilePath`.

The type of the script: file path or inline.

ScriptPath - Script Path

`string`. Optional. Use when `ScriptType = FilePath`.

The path of the script. This should be a fully qualified path or one relative to the default working directory.

Inline - Inline Script

`string`. Optional. Use when `ScriptType = InlineScript`. Default value: `# You can write your azure powershell scripts inline here. \n# You can also pass predefined and custom variables to this script using arguments.`

Specifies the script to execute. The maximum supported inline script length is 5000 characters. Use a script from a file if you want to use a longer script.

ScriptArguments - Script Arguments

`string`. Optional. Use when `ScriptType = FilePath`.

The additional parameters to pass to PowerShell. Can be either ordinal or named parameters.

errorActionPreference - ErrorActionPreference

`string`. Allowed values: `stop`, `continue`, `silentlyContinue`. Default value: `stop`.

Selects the value of the `ErrorActionPreference` variable for executing the script.

FailOnStandardError - Fail on Standard Error

`boolean`. Default value: `false`.

When this is true, this task will fail if any errors are written to the error pipeline or if any data is written to the standard error stream.

`RestrictContextToCurrentTask` - Restrict scope of context to current task

`boolean`. Default value: `false`.

When this is true, this task will restrict the scope of context to the current task only, and the context will not be available to other tasks in the pipeline when using a private agent.

`azurePowerShellVersion` - Azure PowerShell Version

Input alias: `TargetAzurePs`. `string`. Allowed values: `LatestVersion` (Latest installed version), `OtherVersion` (Specify other version). Default value: `OtherVersion`.

In case of hosted agents, the supported Azure PowerShell Version is: `1.0.0` (Hosted VS2017 Queue). To pick the latest version available on the agent, select `LatestVersion` (Latest installed version).

For private agents you can specify a preferred version of Azure PowerShell using `OtherVersion` (Specify other version).

`preferredAzurePowerShellVersion` - Preferred Azure PowerShell Version

Input alias: `CustomTargetAzurePs`. `string`. Required when `TargetAzurePs = OtherVersion`.

The preferred Azure PowerShell Version needs to be a proper semantic version eg. `1.2.3`. Regex like `2.*,2.3.*` is not supported. The Hosted VS2017 Pool currently supports Az module version `1.0.0`.

`pwsh` - Use PowerShell Core

`boolean`. Default value: `false`.

If this is true, then on Windows the task will use `pwsh.exe` from your path instead of `powershell.exe`.

`validateScriptSignature` - Validate script signature

`boolean`. Optional. Use when `ScriptType = FilePath`. Default value: `false`.

If this is true, then the task will first check to make sure specified script is signed and valid before executing it.

`workingDirectory` - Working Directory

`string`.

The working directory where the script is run.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Added support for Az Module and cross platform agents.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.115.0 or greater
Task category	Deploy

AzurePowerShell@3 - Azure PowerShell v3 task

Article • 09/26/2023

Use this task to run a PowerShell script within an Azure environment. The Azure context is authenticated with the provided Azure Resource Manager service connection.

Syntax

YAML

```
# Azure PowerShell v3
# Run a PowerShell script within an Azure environment.
- task: AzurePowerShell@3
  inputs:
    #azureConnectionType: 'ConnectedServiceNameARM' # 'ConnectedServiceName'
    | 'ConnectedServiceNameARM'. Alias: ConnectedServiceNameSelector. Azure
    Connection Type. Default: ConnectedServiceNameARM.
    #azureClassicSubscription: # string. Alias: ConnectedServiceName.
    Required when ConnectedServiceNameSelector = ConnectedServiceName. Azure
    Classic Subscription.
    azureSubscription: # string. Alias: ConnectedServiceNameARM. Required
    when ConnectedServiceNameSelector = ConnectedServiceNameARM. Azure
    Subscription.
    #ScriptType: 'FilePath' # 'FilePath' | 'InlineScript'. Script Type.
    Default: FilePath.
    #ScriptPath: # string. Optional. Use when ScriptType = FilePath. Script
    Path.
    #Inline: # string. Optional. Use when ScriptType = InlineScript. Inline
    Script.
    #ScriptArguments: # string. Optional. Use when ScriptType = FilePath.
    Script Arguments.
    #errorActionPreference: 'stop' # 'stop' | 'continue' |
    'silentlyContinue'. ErrorActionPreference. Default: stop.
    #FailOnStandardError: false # boolean. Fail on Standard Error. Default:
    false.
    # Azure PowerShell version options
    #azurePowerShellVersion: 'OtherVersion' # 'LatestVersion' |
    'OtherVersion'. Alias: TargetAzurePs. Azure PowerShell Version. Default:
    OtherVersion.
    preferredAzurePowerShellVersion: # string. Alias: CustomTargetAzurePs.
    Required when TargetAzurePs = OtherVersion. Preferred Azure PowerShell
    Version.
    # Advanced
    #validateScriptSignature: false # boolean. Optional. Use when ScriptType
    = FilePath. Validate script signature. Default: false.
```

Inputs

`azureConnectionType` - Azure Connection Type

Input alias: `ConnectedServiceNameSelector`. `string`. Allowed values: `ConnectedServiceName` (Azure Classic), `ConnectedServiceNameARM` (Azure Resource Manager). Default value: `ConnectedServiceNameARM`.

`azureClassicSubscription` - Azure Classic Subscription

Input alias: `ConnectedServiceName`. `string`. Required when `ConnectedServiceNameSelector = ConnectedServiceName`.

The Azure Classic subscription to configure before running PowerShell.

`azureSubscription` - Azure Subscription

Input alias: `ConnectedServiceNameARM`. `string`. Required when `ConnectedServiceNameSelector = ConnectedServiceNameARM`.

The Azure Resource Manager subscription to configure before running PowerShell.

`ScriptType` - Script Type

`string`. Allowed values: `FilePath` (Script File Path), `InlineScript` (Inline Script). Default value: `FilePath`.

The type of the script: file path or inline.

`ScriptPath` - Script Path

`string`. Optional. Use when `ScriptType = FilePath`.

The path of the script. This should be a fully qualified path or one relative to the default working directory.

`Inline` - Inline Script

`string`. Optional. Use when `ScriptType = InlineScript`. Default value: `# You can write your azure powershell scripts inline here. \n# You can also pass predefined and custom variables to this script using arguments.`

Specifies the script to execute. The maximum supported inline script length is 5000 characters. Use a script from a file if you want to use a longer script.

ScriptArguments - Script Arguments

`string`. Optional. Use when `ScriptType = FilePath`.

The additional parameters to pass to PowerShell. These can be either ordinal or named parameters.

errorActionPreference - ErrorActionPreference

`string`. Allowed values: `stop`, `continue`, `silentlyContinue`. Default value: `stop`.

Selects the value of the `ErrorActionPreference` variable for executing the script.

FailOnStandardError - Fail on Standard Error

`boolean`. Default value: `false`.

When this is true, this task will fail if any errors are written to the error pipeline or if any data is written to the standard error stream.

azurePowerShellVersion - Azure PowerShell Version

Input alias: `TargetAzurePs`. `string`. Allowed values: `LatestVersion` (Latest installed version), `OtherVersion` (Specify other version). Default value: `OtherVersion`.

In case of hosted agents, the supported Azure PowerShell Versions are `2.1.0`, `3.8.0`, `4.2.1`, `5.1.1` and `6.7.0`. To pick the latest version available on the agent, select `LatestVersion` (Latest installed version).

For private agents, you can specify a preferred version of Azure PowerShell using `OtherVersion` (Specify other version).

preferredAzurePowerShellVersion - Preferred Azure PowerShell Version

Input alias: `CustomTargetAzurePs`. `string`. Required when `TargetAzurePs = OtherVersion`.

The preferred Azure PowerShell Version needs to be a proper semantic version eg. `1.2.3..`. Regex like `2.*,2.3.*` is not supported. Hosted agents currently supports Azure module versions `2.1.0`, `3.8.0`, `4.2.1`, `5.1.1` and AzureRM module versions `2.1.0`, `3.8.0`, `4.2.1`, `5.1.1`, `6.7.0`.

`validateScriptSignature` - Validate script signature

`boolean`. Optional. Use when `ScriptType = FilePath`. Default value: `false`.

If this is true, then the task will first check to make sure specified script is signed and valid before executing it.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Added support for Fail on standard error and ErrorActionPreference.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: azureps
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.0.0 or greater
Task category	Deploy

AzurePowerShell@2 - Azure PowerShell v2 task

Article • 09/26/2023

Use this task to run a PowerShell script within an Azure environment. The Azure context is authenticated with the provided Azure Resource Manager service connection.

Syntax

YAML

```
# Azure PowerShell v2
# Run a PowerShell script within an Azure environment.
- task: AzurePowerShell@2
  inputs:
    #azureConnectionType: 'ConnectedServiceNameARM' # 'ConnectedServiceName'
    | 'ConnectedServiceNameARM'. Alias: ConnectedServiceNameSelector. Azure
    Connection Type. Default: ConnectedServiceNameARM.
    #azureClassicSubscription: # string. Alias: ConnectedServiceName.
    Required when ConnectedServiceNameSelector = ConnectedServiceName. Azure
    Classic Subscription.
    azureSubscription: # string. Alias: ConnectedServiceNameARM. Required
    when ConnectedServiceNameSelector = ConnectedServiceNameARM. Azure
    Subscription.
    ScriptType: 'FilePath' # 'FilePath' | 'InlineScript'. Required. Script
    Type. Default: FilePath.
    #ScriptPath: # string. Optional. Use when ScriptType = FilePath. Script
    Path.
    #Inline: # string. Optional. Use when ScriptType = InlineScript. Inline
    Script.
    #ScriptArguments: # string. Script Arguments.
    #azurePowerShellVersion: 'OtherVersion' # 'LatestVersion' |
    'OtherVersion'. Alias: TargetAzurePs. Azure PowerShell Version. Default:
    OtherVersion.
    preferredAzurePowerShellVersion: # string. Alias: CustomTargetAzurePs.
    Required when TargetAzurePs = OtherVersion. Preferred Azure PowerShell
    Version.
```

Inputs

`azureConnectionType` - Azure Connection Type

Input alias: `ConnectedServiceNameSelector`. `string`. Allowed values:

`ConnectedServiceName` (Azure Classic), `ConnectedServiceNameARM` (Azure Resource Manager). Default value: `ConnectedServiceNameARM`.

azureClassicSubscription - Azure Classic Subscription

Input alias: `ConnectedServiceName`. `string`. Required when `ConnectedServiceNameSelector = ConnectedServiceName`.

The Azure Classic subscription to configure before running PowerShell.

azureSubscription - Azure Subscription

Input alias: `ConnectedServiceNameARM`. `string`. Required when `ConnectedServiceNameSelector = ConnectedServiceNameARM`.

The Azure Resource Manager subscription to configure before running PowerShell.

ScriptType - Script Type

`string`. Required. Allowed values: `FilePath` (Script File Path), `InlineScript` (Inline Script). Default value: `FilePath`.

The type of script: file path or inline.

ScriptPath - Script Path

`string`. Optional. Use when `ScriptType = FilePath`.

The path of the script. This should be a fully qualified path or one relative to the default working directory.

Inline - Inline Script

`string`. Optional. Use when `ScriptType = InlineScript`. Default value: `# You can write your azure powershell scripts inline here. \n# You can also pass predefined and custom variables to this script using arguments.`

Specifies the script to execute. The maximum supported inline script length is 5000 characters. Use a script from a file if you want to use a longer script.

ScriptArguments - Script Arguments

`string`.

The additional parameters to pass to PowerShell. These can be either ordinal or named parameters.

`azurePowerShellVersion` - Azure PowerShell Version

Input alias: `TargetAzurePs`. `string`. Allowed values: `LatestVersion` (Latest installed version), `OtherVersion` (Specify other version). Default value: `OtherVersion`.

In case of hosted agents, the supported Azure PowerShell Versions are `2.1.0`, `3.8.0`, `4.2.1`, `5.1.1` and `6.7.0`. To pick the latest version available on the agent, select `LatestVersion` (Latest installed version).

For private agents you can specify a preferred version of Azure PowerShell using `OtherVersion` (Specify other version).

`preferredAzurePowerShellVersion` - Preferred Azure PowerShell Version

Input alias: `CustomTargetAzurePs`. `string`. Required when `TargetAzurePs = OtherVersion`.

The preferred Azure PowerShell Version needs to be a proper semantic version eg. `1.2.3..`. Regex like `2.*,2.3.*` is not supported. Hosted agents currently supports Azure module versions `2.1.0`, `3.8.0`, `4.2.1`, `5.1.1` and AzureRM module versions `2.1.0`, `3.8.0`, `4.2.1`, `5.1.1`, `6.7.0`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: azureps

Requirement	Description
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.95.0 or greater
Task category	Deploy

AzurePowerShell@1 - Azure PowerShell v1 task

Article • 09/26/2023

Use this task to run a PowerShell script within an Azure environment. The Azure context is authenticated with the provided Azure Resource Manager service connection.

Syntax

YAML

```
# Azure PowerShell v1
# Run a PowerShell script within an Azure environment.
- task: AzurePowerShell@1
  inputs:
    #ConnectedServiceNameSelector: 'ConnectedServiceName' #
    'ConnectedServiceName' | 'ConnectedServiceNameARM'. Azure Connection Type.
    Default: ConnectedServiceName.
    ConnectedServiceName: # string. Required when
    ConnectedServiceNameSelector = ConnectedServiceName. Azure Classic
    Subscription.
    #ConnectedServiceNameARM: # string. Required when
    ConnectedServiceNameSelector = ConnectedServiceNameARM. Azure Subscription.
    ScriptType: 'FilePath' # 'FilePath' | 'InlineScript'. Required. Script
    Type. Default: FilePath.
    #ScriptPath: # string. Optional. Use when ScriptType = FilePath. Script
    Path.
    #Inline: # string. Optional. Use when ScriptType = InlineScript. Inline
    Script.
    #ScriptArguments: # string. Script Arguments.
```

Inputs

`ConnectedServiceNameSelector` - Azure Connection Type

`string`. Allowed values: `ConnectedServiceName` (Azure Classic), `ConnectedServiceNameARM` (Azure Resource Manager). Default value: `ConnectedServiceName`.

`ConnectedServiceName` - Azure Classic Subscription

`string`. Required when `ConnectedServiceNameSelector = ConnectedServiceName`.

The Azure Classic subscription to configure before running PowerShell.

ConnectedServiceNameARM - Azure Subscription

`string`. Required when `ConnectedServiceNameSelector = ConnectedServiceNameARM`.

The Azure Resource Manager subscription to configure before running PowerShell.

ScriptType - Script Type

`string`. Required. Allowed values: `FilePath` (Script File Path), `InlineScript` (Inline Script). Default value: `FilePath`.

The type of the script: file path or inline.

ScriptPath - Script Path

`string`. Optional. Use when `ScriptType = FilePath`.

The path of the script. This should be a fully qualified path or one relative to the default working directory.

Inline - Inline Script

`string`. Optional. Use when `ScriptType = InlineScript`. Default value: `# You can write your azure powershell scripts inline here. \n# You can also pass predefined and custom variables to this script using arguments.`

Specifies the script to execute. The maximum supported inline script length is 500 characters. Use a script from a file if you want to use a longer script.

ScriptArguments - Script Arguments

`string`.

The additional parameters to pass to PowerShell. These can be either ordinal or named parameters.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: azureps
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.95.0 or greater
Task category	Deploy

AzureResourceGroupDeployment@2 - Azure resource group deployment v2 task

Article • 09/26/2023

Deploy an Azure Resource Manager (ARM) template to a resource group and manage virtual machines.

Syntax

YAML

```
# Azure resource group deployment v2
# Deploy an Azure Resource Manager (ARM) template to a resource group and
# manage virtual machines.
- task: AzureResourceGroupDeployment@2
  inputs:
    # Azure Details
    azureSubscription: # string. Alias: ConnectedServiceName. Required.
    Azure subscription.
    action: 'Create Or Update Resource Group' # 'Create Or Update Resource
    Group' | 'Select Resource Group' | 'Start' | 'Stop' | 'StopWithDeallocate' |
    'Restart' | 'Delete' | 'DeleteRG'. Required. Action. Default: Create Or
    Update Resource Group.
    resourceGroupName: # string. Required. Resource group.
    #location: # string. Required when action = Create Or Update Resource
    Group. Location.
    # Template
    #templateLocation: 'Linked artifact' # 'Linked artifact' | 'URL of the
    file'. Required when action = Create Or Update Resource Group. Template
    location. Default: Linked artifact.
    #csmFileLink: # string. Required when templateLocation = URL of the file
    && action = Create Or Update Resource Group. Template link.
    #csmParametersFileLink: # string. Optional. Use when templateLocation =
    URL of the file && action = Create Or Update Resource Group. Template
    parameters link.
    #csmFile: # string. Required when templateLocation = Linked artifact &&
    action = Create Or Update Resource Group. Template.
    #csmParametersFile: # string. Optional. Use when templateLocation =
    Linked artifact && action = Create Or Update Resource Group. Template
    parameters.
    #overrideParameters: # string. Optional. Use when action = Create Or
    Update Resource Group. Override template parameters.
    #deploymentMode: 'Incremental' # 'Incremental' | 'Complete' |
    'Validation'. Required when action = Create Or Update Resource Group.
    Deployment mode. Default: Incremental.
    # Advanced deployment options for virtual machines
```

```

    #enableDeploymentPrerequisites: 'None' # 'None' | 'ConfigureVMwithWinRM'
    | 'ConfigureVMWithDGAgent'. Optional. Use when action = Create Or Update
    Resource Group || action = Select Resource Group. Enable prerequisites.
    Default: None.

    #teamServicesConnection: # string. Alias: deploymentGroupEndpoint.
    Required when enableDeploymentPrerequisites = ConfigureVMWithDGAgent &&
    action = Create Or Update Resource Group || action = Select Resource Group.
    Azure Pipelines service connection.

    #teamProject: # string. Alias: project. Required when
    enableDeploymentPrerequisites = ConfigureVMWithDGAgent && action = Create Or
    Update Resource Group || action = Select Resource Group. Team project.

    #deploymentGroupName: # string. Required when
    enableDeploymentPrerequisites = ConfigureVMWithDGAgent && action = Create Or
    Update Resource Group || action = Select Resource Group. Deployment Group.

    #copyAzureVMTags: true # boolean. Optional. Use when
    enableDeploymentPrerequisites = ConfigureVMWithDGAgent && action = Create Or
    Update Resource Group || action = Select Resource Group. Copy Azure VM tags
    to agents. Default: true.

    #runAgentServiceAsUser: false # boolean. Optional. Use when
    enableDeploymentPrerequisites = ConfigureVMWithDGAgent && action = Create Or
    Update Resource Group || action = Select Resource Group. Run agent service
    as a user. Default: false.

    #userName: # string. Required when enableDeploymentPrerequisites =
    ConfigureVMWithDGAgent && runAgentServiceAsUser = true && action = Create Or
    Update Resource Group || action = Select Resource Group. User name.

    #password: # string. Optional. Use when enableDeploymentPrerequisites =
    ConfigureVMWithDGAgent && runAgentServiceAsUser = true && action = Create Or
    Update Resource Group || action = Select Resource Group. Password.

    #outputVariable: # string. Optional. Use when
    enableDeploymentPrerequisites = ConfigureVMwithWinRM ||
    enableDeploymentPrerequisites = None && action = Create Or Update Resource
    Group || action = Select Resource Group. VM details for WinRM.

    # Advanced

    #deploymentName: # string. Optional. Use when action = Create Or Update
    Resource Group. Deployment name.

    #deploymentOutputs: # string. Optional. Use when action = Create Or
    Update Resource Group. Deployment outputs.

    #addSpnToEnvironment: false # boolean. Optional. Use when action =
    Create Or Update Resource Group. Access service principal details in
    override parameters. Default: false.

    #useWithoutJSON: false # boolean. Optional. Use when action = Create Or
    Update Resource Group. Use individual output values without JSON.Stringify
    applied. Default: false.

```

Inputs

`azureSubscription` - Azure subscription

Input alias: `ConnectedServiceName`. `string`. Required.

Selects the service connection that contains an Azure Subscription for the deployment.

action - Action

`string`. Required. Allowed values: `Create Or Update Resource Group`, `Select Resource Group` (Configure virtual machine deployment options), `Start` (Start virtual machines), `Stop` (Stop virtual machines), `StopWithDeallocate` (Stop and deallocate virtual machines), `Restart` (Restart virtual machines), `Delete` (Delete virtual machines), `DeleteRG` (Delete resource group). Default value: `Create Or Update Resource Group`.

The action to be performed on the Azure resources or resource group.

resourceGroupName - Resource group

`string`. Required.

Provides the name of the resource group.

location - Location

`string`. Required when `action = Create Or Update Resource Group`.

The location to deploy the resource group. If the resource group already exists in the subscription, then this value will be ignored.

templateLocation - Template location

`string`. Required when `action = Create Or Update Resource Group`. Allowed values: `Linked artifact`, `URL of the file`. Default value: `Linked artifact`.

Select either **Linked artifact** or **URL of the file**.

csmFileLink - Template link

`string`. Required when `templateLocation = URL of the file && action = Create Or Update Resource Group`.

Specifies the URL of the template file. An example URL:

`https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-vm-simple-windows/azuredeploy.json`

To deploy a template stored in a private storage account, retrieve and include the shared access signature (SAS) token in the URL of the template. Example:

`<blob_storage_url>/template.json?<SAStoken>`

To upload a template file (or a linked template) to a storage account and generate a SAS token, use the [Azure file copy](#) task or follow the steps using [PowerShell](#) or [Azure CLI](#).

To view the template parameters in a grid, click on `...` next to the override template parameters text box. This feature requires that CORS rules are enabled at the source. If the templates are in an Azure storage blob, see [Understanding CORS requests](#) to enable CORS.

`csmParametersFileLink` - Template parameters link

`string`. Optional. Use when `templateLocation = URL of the file && action = Create Or Update Resource Group`.

Specifies the URL of the parameters file. Example:

[https://raw.githubusercontent.com/Azure/...!\[\]\(dc212d66f80874ee3a57f78987af37b9_img.jpg\)](https://raw.githubusercontent.com/Azure/...)

To use a file stored in a private storage account, retrieve and include the shared access signature (SAS) token in the URL of the template. Example:

`<blob_storage_url>/template.json?<SASToken>` To upload a parameters file to a storage account and generate a SAS token, you could use [Azure file copy](#) task or follow the steps using [PowerShell](#) or [Azure CLI](#).

To view the template parameters in a grid, click on `...` next to the override template parameters text box. This feature requires that CORS rules are enabled at the source. If the templates are in an Azure storage blob, see [Understanding CORS requests](#) to enable CORS.

`csmFile` - Template

`string`. Required when `templateLocation = Linked artifact && action = Create Or Update Resource Group`.

Specifies the path or a pattern pointing to the Azure Resource Manager template. Learn more about [Azure Resource Manager templates](#). To get started immediately, use [this sample template](#).

`csmParametersFile` - Template parameters

`string`. Optional. Use when `templateLocation = Linked artifact && action = Create Or Update Resource Group`.

Specifies the URL of the parameters file. An example URL:

```
https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-vm-simple-windows/azuredeploy.parameters.json
```

To use a file stored in a private storage account, retrieve and include the shared access signature (SAS) token in the URL of the template. Example:

```
<blob_storage_url>/template.json?<SASToken>
```

To upload a parameters file to a storage account and generate a SAS token, use the [Azure file copy](#) task or follow the steps using [PowerShell](#) or [Azure CLI](#).

To view the template parameters in a grid, click on `...` next to the override template parameters text box. This feature requires that CORS rules are enabled at the source. If the templates are in an Azure storage blob, see [Understanding CORS requests](#) to enable CORS.

overrideParameters - Override template parameters

```
string. Optional. Use when action = Create Or Update Resource Group.
```

Specifies the template parameters to override.

To view the template parameters in a grid, click on `...` next to the override parameters textbox. This feature requires that CORS rules are enabled at the source. If the templates are in the Azure storage blob, reference this string to enable CORS, or type the template parameters to override in the textbox.

```
Example: -storageName fabrikam -adminUsername $(vmusername) -adminPassword  
(ConvertTo-SecureString -String '$(password)' -AsPlainText -Force) -  
azureKeyVaultName $(fabrikamFibre).
```

If the parameter value has multiple words, enclose the words in quotes, even if you're passing the value by using variables. For example, `-name "parameter value" -name2 "$(var)"`. To override object type parameters, use stringified JSON objects. For example, `-options ["option1"] -map {"key1": "value1" }`.

deploymentMode - Deployment mode

```
string. Required when action = Create Or Update Resource Group. Allowed values:  
Incremental, Complete, Validation (Validation only). Default value: Incremental.
```

The `Incremental` mode handles deployments as incremental updates to the resource group. It leaves unchanged resources that exist in the resource group but are not specified in the template.

`Complete` mode deletes resources that are not in your template. Complete mode takes relatively more time than incremental mode. If the task times out, consider increasing the timeout or changing to the `Incremental` mode.

⚠ Warning

Complete mode will delete all the existing resources in the resource group that are not specified in the template. Do review if the resource group you're deploying to doesn't contain any necessary resources that are not specified in the template.

`Validate` mode enables you to find problems with the template before creating actual resources.

ⓘ Note

The `Validate` mode always creates a resource group, even if no resources are deployed.

Learn more about [deployment modes](#).

`enableDeploymentPrerequisites` - Enable prerequisites

`string`. Optional. Use when `action = Create Or Update Resource Group || action = Select Resource Group`. Allowed values: `None`, `ConfigureVMwithWinRM` (Configure with WinRM agent), `ConfigureVMWithDAGAgent` (Configure with Deployment Group agent). Default value: `None`.

Applicable only when the resource group contains virtual machines.

Choosing the Deployment Group option configures the Deployment Group agent on each of the virtual machines.

Selecting the WinRM option configures the Windows Remote Management (WinRM) listener over HTTPS protocol on port 5986 using a self-signed certificate. This configuration is required for performing deployment operation on Azure machines. If the target virtual machines are backed by a load balancer, ensure the Inbound NAT rules are configured for target port (5986).

`teamServicesConnection` - Azure Pipelines service connection

Input alias: `deploymentGroupEndpoint`. `string`. Required when

```
enableDeploymentPrerequisites = ConfigureVMWithDGAgent && action = Create Or Update Resource Group || action = Select Resource Group.
```

Specifies the service connection to connect to an Azure DevOps organization or collection for agent registration.

You can create a service connection using [+New](#) and then selecting [Token-based authentication](#). You need a [personal access token\(PAT\)](#) to setup a service connection. Click [Manage](#) to update the service connection details.

teamProject - Team project

Input alias: `project`. `string`. Required when `enableDeploymentPrerequisites = ConfigureVMWithDGAgent && action = Create Or Update Resource Group || action = Select Resource Group`.

Specifies the Team Project which defines the deployment group.

deploymentGroupName - Deployment Group

`string`. Required when `enableDeploymentPrerequisites = ConfigureVMWithDGAgent && action = Create Or Update Resource Group || action = Select Resource Group`.

Specifies the deployment group against which the agent(s) will be registered. Learn more about [deployment groups](#).

copyAzureVMTags - Copy Azure VM tags to agents

`boolean`. Optional. Use when `enableDeploymentPrerequisites = ConfigureVMWithDGAgent && action = Create Or Update Resource Group || action = Select Resource Group`.

Default value: `true`.

Chooses if the configured tags on the Azure VM need to be copied to the corresponding deployment group agent.

By default, all Azure tags are copied following the format: `Key: Value`. Example: A `Role: Web` Azure tag would be copied as-is to the agent machine.

Learn more about [using tags for Azure resources](#).

runAgentServiceAsUser - Run agent service as a user

`boolean`. Optional. Use when `enableDeploymentPrerequisites = ConfigureVMWithDGAgent`

```
&& action = Create Or Update Resource Group || action = Select Resource Group.
```

Default value: `false`.

Runs the agent service as a user other than the default user if the value is set to `true`.

The default user is `NT AUTHORITY\SYSTEM` in Windows and `root` in Linux.

`userName` - User name

```
string. Required when enableDeploymentPrerequisites = ConfigureVMWithDGAgent &&
runAgentServiceAsUser = true && action = Create Or Update Resource Group || action
= Select Resource Group.
```

The username to run the agent service on the virtual machines.

For domain users, specify values as `domain\username` or `username@domain.com`. For local users, specify `username`.

It is assumed that the same domain user or a local user with the same name, respectively, is present on all the virtual machines in the resource group.

`password` - Password

```
string. Optional. Use when enableDeploymentPrerequisites = ConfigureVMWithDGAgent
&& runAgentServiceAsUser = true && action = Create Or Update Resource Group || action
= Select Resource Group.
```

The password for the user to run the agent service on the Windows VMs.

It is assumed that the password is the same for the specified user on all the VMs.

It can accept variables defined in build or release pipelines as `$(passwordVariable)`. You may mark the variable as `secret` to secure it.

For Linux VMs, a password is not required and will be ignored.

`outputVariable` - VM details for WinRM

```
string. Optional. Use when enableDeploymentPrerequisites = ConfigureVMwithWinRM || enableDeploymentPrerequisites = None && action = Create Or Update Resource Group || action = Select Resource Group.
```

Required when an existing resource group is selected. Provides a name for the resource group variable. The variable can be used as `$(variableName)` to refer to the resource

group in subsequent tasks, such as in PowerShell on Target Machines task for deploying applications.

Valid only when the selected action is `Create`, `Update`, or `Select`.

deploymentName - Deployment name

`string`. Optional. Use when `action = Create Or Update Resource Group`.

Specifies the name of the resource group deployment to create.

deploymentOutputs - Deployment outputs

`string`. Optional. Use when `action = Create Or Update Resource Group`.

Provides a name for the output variable, which contains the outputs section of the current deployment object in string format. Use the `ConvertFrom-Json` PowerShell cmdlet to parse the JSON object and access the individual output values.

addSpnToEnvironment - Access service principal details in override parameters

`boolean`. Optional. Use when `action = Create Or Update Resource Group`. Default value: `false`.

Adds the service principal ID and key of the Azure endpoint chosen to be the script's execution environment. The variables `$servicePrincipalId` and `$servicePrincipalKey` can be in override parameters, such as `-key $servicePrincipalKey`.

useWithoutJSON - Use individual output values without `JSON.Stringify` applied

`boolean`. Optional. Use when `action = Create Or Update Resource Group`. Default value: `false`.

Individual output values are being converted via `JSON.Stringify` by default. If you want to use the output values as it is without converting them via `JSON.Stringify`, enable this option. For more details refer to [this ↗](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

There is a new version of this task available at
[AzureResourceManagerTemplateDeployment@3 - ARM template deployment v3 task.](#)

What's new in task version 2

- Works with cross-platform agents (Linux, macOS, or Windows)
- Supports Template JSONs located at any publicly accessible http/https URLs.
- Enhanced UX for Override parameters which can now be viewed/edited in a grid.
- NAT rule mapping for VMs which are backed by an Load balancer.
- "Resource group" field is now renamed as "VM details for WinRM" and is included in the section "Advanced deployment options for virtual machines".
- Limitations:
 - No support for Classic subscriptions. Only ARM subscriptions are supported.
 - No support for PowerShell syntax as the task is now node.js based. Ensure the case sensitivity of the parameter names match, when you override the template parameters. Also, remove the PowerShell cmdlets like "ConvertTo-SecureString" when you migrate from version 1.0 to version 2.0.

Troubleshooting

Error: Internal Server Error

These issues are mostly transient in nature. There are multiple reasons why it could be happening:

- One of the Azure services you're trying to deploy is undergoing maintenance in the region you're trying to deploy to. Keep an eye out on <https://status.azure.com/> to check downtimes of Azure Services.
- Azure Pipelines service itself is going through maintenance. Keep an eye out on <https://status.dev.azure.com/> for downtimes.

However, we've seen some instances where this is due to an error in the ARM template, such as the Azure service you're trying to deploy doesn't support the region you've chosen for the resource.

Error: Timeout

Timeout issues could be coming from two places:

- Azure Pipelines Agent
- Portal Deployment

You can identify if the timeout is from portal, by checking for the portal deployment link that'll be in the task logs. If there's no link, this is likely due to Azure Pipelines agent. If there's a link, follow the link to see if there's a timeout that has happened in the portal deployment.

Error: CORS rules to be enabled while overriding parameters

If the template file is being referred from a BLOB, while overriding parameters in the pipeline, you might see the following warning message:

`Warning: Failed to download the file from template path.`

This feature requires the CORS rules to be enabled at the source. If templates are in Azure storage blob, see [Cross-origin resource sharing support](#) to enable CORS.

Besides enabling CORS, ensure that the SAS token specified in the link of the template is "srt-sco". This token is required for you to download the file and proceed.

Azure Pipelines Agent

If the issue is coming from Azure Pipelines agent, you can increase the timeout by setting `timeoutInMinutes` as key in the YAML to 0. For more information, see [Specify jobs in your pipeline](#).

Portal Deployment

Check out this doc on how to identify if the error came from the Azure portal: [View deployment history with Azure Resource Manager](#).

In case of portal deployment, try setting "`timeoutInMinutes`" in the ARM template to "0". If not specified, the value assumed is 60 minutes. 0 makes sure the deployment will run for as long as it can to succeed.

This could also be happening because of transient issues in the system. Keep an eye on <https://status.dev.azure.com/> to check if there's a downtime in Azure Pipelines service.

Error: Azure Resource Manager (ARM) template failed validation

This issue happens mostly because of an invalid parameter in the ARM template, such as an unsupported SKU or region. If the validation fails, check the error message. It should point you to the resource and parameter that's invalid.

This issue also might occur because of multiline strings. Currently, the Azure Resource Group Deployment task doesn't support multiline strings in an ARM template or parameter JSON file.

In addition, refer to this article regarding structure and syntax of ARM Templates: [Understand the structure and syntax of ARM templates](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.119.1 or greater
Task category	Deploy

AzureResourceGroupDeployment@1 - Azure Resource Group Deployment v1 task

Article • 09/26/2023

Use this task to deploy, start, stop, and delete Azure Resource Groups.

This task is deprecated; use [AzureResourceGroupDeployment@2](#).

Syntax

YAML

```
# Azure Resource Group Deployment v1
# Deploy, start, stop, delete Azure Resource Groups.
- task: AzureResourceGroupDeployment@1
  inputs:
    #ConnectedServiceNameSelector: 'ConnectedServiceName' #
    'ConnectedServiceName' | 'ConnectedServiceNameClassic'. Azure Connection
    Type. Default: ConnectedServiceName.
    ConnectedServiceName: # string. Required when
    ConnectedServiceNameSelector = ConnectedServiceName. Azure Subscription.
    #ConnectedServiceNameClassic: # string. Required when
    ConnectedServiceNameSelector = ConnectedServiceNameClassic. Azure Classic
    Subscription.
    action: 'Create Or Update Resource Group' # 'Create Or Update Resource
    Group' | 'Select Resource Group' | 'Start' | 'Stop' | 'Restart' | 'Delete' |
    'DeleteRG'. Required when ConnectedServiceNameSelector =
    ConnectedServiceName. Action. Default: Create Or Update Resource Group.
    #actionClassic: 'Select Resource Group' # 'Select Resource Group'.
    Required when ConnectedServiceNameSelector = ConnectedServiceNameClassic.
    Action. Default: Select Resource Group.
    resourceGroupName: # string. Required when ConnectedServiceNameSelector
    = ConnectedServiceName. Resource Group.
    #cloudService: # string. Required when ConnectedServiceNameSelector =
    ConnectedServiceNameClassic. Cloud Service.
    #location: 'East US' # 'Australia East' | 'Australia Southeast' |
    'Brazil South' | 'Canada Central' | 'Canada East' | 'Central India' |
    'Central US' | 'East Asia' | 'East US' | 'East US 2' | 'Japan East' |
    'Japan West' | 'North Central US' | 'North Europe' | 'South Central US' |
    'South India' | 'Southeast Asia' | 'UK South' | 'UK West' | 'West Central
    US' | 'West Europe' | 'West India' | 'West US' | 'West US 2'. Required when
    action = Create Or Update Resource Group. Location. Default: East US.
    #csmFile: # string. Required when action = Create Or Update Resource
    Group. Template.
    #csmParametersFile: # string. Optional. Use when action = Create Or
    Update Resource Group. Template Parameters.
```

```
#overrideParameters: # string. Optional. Use when action = Create Or
Update Resource Group. Override Template Parameters.
#deploymentMode: 'Incremental' # 'Validation' | 'Incremental' |
'Complete'. Required when action = Create Or Update Resource Group.
Deployment Mode. Default: Incremental.
#enableDeploymentPrerequisitesForCreate: false # boolean. Optional. Use
when action = Create Or Update Resource Group. Enable Deployment
Prerequisites. Default: false.
#enableDeploymentPrerequisitesForSelect: false # boolean. Optional. Use
when action = Select Resource Group. Enable Deployment Prerequisites.
Default: false.
# Output
#outputVariable: # string. Resource Group.
```

Inputs

`ConnectedServiceNameSelector` - Azure Connection Type

`string`. Allowed values: `ConnectedServiceName` (Azure Resource Manager),
`ConnectedServiceNameClassic` (Azure Classic). Default value: `ConnectedServiceName`.

Required. Selects the service connection that contains an Azure Subscription for the deployment.

`ConnectedServiceName` - Azure Subscription

`string`. Required when `ConnectedServiceNameSelector = ConnectedServiceName`.

Required. Selects the service connection that contains an Azure Subscription for the deployment.

`ConnectedServiceNameClassic` - Azure Classic Subscription

`string`. Required when `ConnectedServiceNameSelector = ConnectedServiceNameClassic`.

Selects an Azure Classic subscription for the deployment.

`action` - Action

`string`. Required when `ConnectedServiceNameSelector = ConnectedServiceName`. Allowed values: `Create Or Update Resource Group`, `Select Resource Group`, `Start` (Start Virtual Machines), `Stop` (Stop Virtual Machines), `Restart` (Restart Virtual Machines), `Delete` (Delete Virtual Machines), `DeleteRG` (Delete Resource Group). Default value: `Create Or Update Resource Group`.

The action to be performed on the Azure resources or resource group.

actionClassic - Action

`string`. Required when `ConnectedServiceNameSelector = ConnectedServiceNameClassic`.

Allowed values: `Select Resource Group` (Select Cloud Service). Default value: `Select Resource Group`.

The action to be performed on the Azure resources or cloud service.

resourceGroupName - Resource Group

`string`. Required when `ConnectedServiceNameSelector = ConnectedServiceName`.

Provides the name of the resource group.

cloudService - Cloud Service

`string`. Required when `ConnectedServiceNameSelector = ConnectedServiceNameClassic`.

Provides the name of the cloud service.

location - Location

`string`. Required when `action = Create Or Update Resource Group`. Allowed values: `Australia East`, `Australia Southeast`, `Brazil South`, `Canada Central`, `Canada East`, `Central India`, `Central US`, `East Asia`, `East US`, `East US 2`, `Japan East`, `Japan West`, `North Central US`, `North Europe`, `South Central US`, `South India`, `Southeast Asia`, `UK South`, `UK West`, `West Central US`, `West Europe`, `West India`, `West US`, `West US 2`. Default value: `East US`.

The location to deploy the resource group. If the resource group already exists in the subscription, then this value will be ignored.

csmFile - Template

`string`. Required when `action = Create Or Update Resource Group`.

Specifies the path or a pattern pointing to the Azure Resource Manager template. Learn more about [Azure Resource Manager templates](#). To get started immediately, use [this sample template](#).

csmParametersFile - Template Parameters

`string`. Optional. Use when `action = Create Or Update Resource Group`.

Specifies the URL of the parameters file. An example URL:

`https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-vm-simple-windows/azuredeploy.parameters.json`

To use a file stored in a private storage account, retrieve and include the shared access signature (SAS) token in the URL of the template. Example:

`<blob_storage_url>/template.json?<SAStoken>` To upload a parameters file to a storage account and generate a SAS token, use the [Azure file copy](#) task or follow the steps using [PowerShell](#) or [Azure CLI](#).

To view the template parameters in a grid, click on `...` next to the override template parameters text box. This feature requires that CORS rules are enabled at the source. If templates are in Azure storage blob, refer to [Cross-Origin Resource Sharing](#) to enable CORS.

overrideParameters - Override Template Parameters

`string`. Optional. Use when `action = Create Or Update Resource Group`.

Specifies the template parameters to override.

To view the template parameters in a grid, click on `...` next to the Override Parameters textbox. This feature requires that CORS rules are enabled at the source. If the templates are in the Azure storage blob, reference this string to enable CORS, or type the template parameters to override in the textbox.

Example: `-storageName fabrikam -adminUsername $(vmusername) -adminPassword (ConvertTo-SecureString -String '$(password)' -AsPlainText -Force) -azureKeyVaultName $(fabrikamFibre)`.

If the parameter value has multiple words, enclose the words in quotes, even if you're passing the value by using variables. For example, `-name "parameter value" -name2 "$(var)"`. To override object type parameters, use stringified JSON objects. For example, `-options ["option1"] -map {"key1": "value1" }`.

deploymentMode - Deployment Mode

`string`. Required when `action = Create Or Update Resource Group`. Allowed values:

`Validation` (`Validation Only`), `Incremental`, `Complete`. Default value: `Incremental`.

The `Incremental` mode handles deployments as incremental updates to the resource group. It leaves unchanged resources that exist in the resource group but are not specified in the template.

`Complete` mode deletes resources that are not in your template. Complete mode takes relatively more time than incremental mode. If the task times out, consider increasing the timeout or changing to the `Incremental` mode.

⚠ Warning

Complete mode will delete all the existing resources in the resource group that are not specified in the template. Do review if the resource group you're deploying to doesn't contain any necessary resources that are not specified in the template.

`Validate` mode enables you to find problems with the template before creating actual resources.

ⓘ Note

The `Validate` mode always creates a resource group, even if no resources are deployed.

Learn more about [deployment modes](#).

`enableDeploymentPrerequisitesForCreate` - Enable Deployment Prerequisites

`boolean`. Optional. Use when `action = Create Or Update Resource Group`. Default value: `false`.

Applicable only when the resource group contains virtual machines.

Choosing the Deployment Group option configures the Deployment Group agent on each of the virtual machines.

Selecting the WinRM option configures the Windows Remote Management (WinRM) listener over HTTPS protocol on port 5986 using a self-signed certificate. This configuration is required for performing deployment operation on Azure machines. If the target virtual machines are backed by a load balancer, ensure the Inbound NAT rules are configured for the target port (5986).

`enableDeploymentPrerequisitesForSelect` - Enable Deployment Prerequisites

`boolean`. Optional. Use when `action = Select Resource Group`. Default value: `false`.

Applicable only when the resource group contains virtual machines.

Choosing the Deployment Group option configures the Deployment Group agent on each of the virtual machines.

Selecting the WinRM option configures the Windows Remote Management (WinRM) listener over HTTPS protocol on port 5986 using a self-signed certificate. This configuration is required for performing deployment operation on Azure machines. If the target virtual machines are backed by a load balancer, ensure the Inbound NAT rules are configured for the target port (5986).

`outputVariable` - Resource Group

`string`.

Required when an existing resource group is selected. Provides a name for the resource group variable. The variable can be used as `$(variableName)` to refer to the resource group in subsequent tasks, such as in PowerShell on Target Machines task for deploying applications.

Valid only when the selected action is `Create`, `Update`, or `Select`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup

Requirement	Description
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: azureps
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.103.0 or greater
Task category	Deploy

AzureSpringCloud@0 - Azure Spring Apps v0 task

Article • 09/26/2023

This task deploys applications to Azure Spring Apps and manages those deployments.

Syntax

YAML

```
# Azure Spring Apps v0
# Deploy applications to Azure Spring Apps and manage deployments.
- task: AzureSpringCloud@0
  inputs:
    azureSubscription: # string. Alias: ConnectedServiceName. Required.
    Azure subscription.
    Action: 'Deploy' # 'Deploy' | 'Set Production' | 'Delete Staging
    Deployment'. Required. Action. Default: Deploy.
    AzureSpringCloud: # string. Required. Azure Spring Apps Name.
    AppName: # string. Required. App.
    #DeploymentType: 'Artifacts' # 'Artifacts' | 'CustomContainer'.
    Optional. Use when Action = Deploy. Deployment Type. Default: Artifacts.
    #UseStagingDeployment: true # boolean. Optional. Use when Action =
    Deploy || Action = Set Production. Use Staging Deployment. Default: true.
    #CreateNewDeployment: false # boolean. Optional. Use when Action =
    Deploy && UseStagingDeployment = false. Create a new staging deployment if
    one does not exist. Default: false.
    #DeploymentName: # string. Optional. Use when UseStagingDeployment =
    false && Action != Delete Staging Deployment. Deployment.
    #Package: '$(System.DefaultWorkingDirectory)/**/*.jar' # string.
    Optional. Use when Action = Deploy && DeploymentType = Artifacts. Package or
    folder. Default: $(System.DefaultWorkingDirectory)/**/*.jar.
    #RegistryServer: 'docker.io' # string. Optional. Use when Action =
    Deploy && DeploymentType = CustomContainer. Registry Server. Default:
    docker.io.
    #RegistryUsername: # string. Optional. Use when Action = Deploy &&
    DeploymentType = CustomContainer. Registry Username.
    #RegistryPassword: # string. Optional. Use when Action = Deploy &&
    DeploymentType = CustomContainer. Registry Password.
    #ImageName: 'hello-world:v1' # string. Optional. Use when Action =
    Deploy && DeploymentType = CustomContainer. Image Name and Tag. Default:
    hello-world:v1.
    #ImageCommand: # string. Optional. Use when Action = Deploy &&
    DeploymentType = CustomContainer. Image Command.
    #ImageArgs: # string. Optional. Use when Action = Deploy &&
    DeploymentType = CustomContainer. Image Arguments.
    #ImageLanguageFramework: # 'springboot'. Optional. Use when Action =
    Deploy && DeploymentType = CustomContainer. Language Framework.
    # Application and Configuration Settings
```

```
#Builder: # string. Optional. Use when Action = Deploy && DeploymentType = Artifacts. Builder.  
#EnvironmentVariables: # string. Optional. Use when Action = Deploy. Environment Variables.  
#JvmOptions: # string. Optional. Use when Action = Deploy && DeploymentType = Artifacts. JVM Options.  
#RuntimeVersion: 'Java_11' # 'Java_8' | 'Java_11' | 'NetCore_31'. Optional. Use when Action = Deploy && DeploymentType = Artifacts. Runtime Version. Default: Java_11.  
#DotNetCoreMainEntryPath: # string. Optional. Use when RuntimeVersion = NetCore_31. Main Entry Path.  
#Version: # string. Optional. Use when Action = Deploy. Version.
```

Inputs

`azureSubscription` - Azure subscription

Input alias: `ConnectedServiceName`. `string`. Required.

Specifies the [Azure Resource Manager](#) subscription for the deployment.

`Action` - Action

`string`. Required. Allowed values: `Deploy`, `Set Production` (Set Production Deployment), `Delete Staging Deployment`. Default value: `Deploy`.

The action to be performed on Azure Spring Apps.

`AzureSpringCloud` - Azure Spring Apps Name

`string`. Required.

The name or resource ID of the Azure Spring Apps instance to deploy.

`AppName` - App

`string`. Required.

The name of the Azure Spring Apps app to deploy. The app must exist prior to the task execution.

`DeploymentType` - Deployment Type

`string`. Optional. Use when `Action = Deploy`. Allowed values: `Artifacts`, `CustomContainer` (Custom Container). Default value: `Artifacts`.

To deploy with source code or Java package, choose "Artifacts"; To deploy with container image, choose "Custom Container".

UseStagingDeployment - Use Staging Deployment

`boolean`. Optional. Use when `Action = Deploy || Action = Set Production`. Default value: `true`.

At the time the task runs, this input automatically selects the deployment that's set as `staging`.

If set to `true`, apply the task to the [deployment](#) that is set as the staging deployment at time of execution. If omitted, the `DeploymentName` parameter must be set.

CreateNewDeployment - Create a new staging deployment if one does not exist.

`boolean`. Optional. Use when `Action = Deploy && UseStagingDeployment = false`. Default value: `false`.

If set to `true`, and the deployment specified by `DeploymentName` does not exist at execution time, it will be created. If omitted, the `DeploymentName` parameter must be set.

DeploymentName - Deployment

`string`. Optional. Use when `UseStagingDeployment = false && Action != Delete Staging Deployment`.

The [deployment](#) to which this task will apply. If not using blue-green deployments, set this field to `default`. The value must start with a letter and consist of lowercase letters and numbers only.

Package - Package or folder

`string`. Optional. Use when `Action = Deploy && DeploymentType = Artifacts`. Default value: `$(System.DefaultWorkingDirectory)/**/*.jar`.

The file path to the package or folder containing the Azure Spring Apps app contents (`.jar` file for Java, `.zip` for .NET Core).

Variables ([Build](#) | [Release](#)) and wildcards are supported.

For example, `$(System.DefaultWorkingDirectory)/**/*.jar`

Builder - Builder

`string`. Optional. Use when `Action = Deploy && DeploymentType = Artifacts`.

Select a builder of VMware Tanzu® Build Service™, this can be used in enterprise tier.
For detailed description, please check [Use Tanzu Build Service](#).

RegistryServer - Registry Server

`string`. Optional. Use when `Action = Deploy && DeploymentType = CustomContainer`.

Default value: `docker.io`.

The registry of the container image. Default: docker.io.

RegistryUsername - Registry Username

`string`. Optional. Use when `Action = Deploy && DeploymentType = CustomContainer`.

The username of the container registry.

RegistryPassword - Registry Password

`string`. Optional. Use when `Action = Deploy && DeploymentType = CustomContainer`.

The password of the container registry.

ImageName - Image Name and Tag

`string`. Optional. Use when `Action = Deploy && DeploymentType = CustomContainer`.

Default value: `hello-world:v1`.

The container image tag.

ImageCommand - Image Command

`string`. Optional. Use when `Action = Deploy && DeploymentType = CustomContainer`.

The command of the container image.

ImageArgs - Image Arguments

`string`. Optional. Use when `Action = Deploy && DeploymentType = CustomContainer`.

The arguments of the container image.

ImageLanguageFramework - Language Framework

`string`. Optional. Use when `Action = Deploy && DeploymentType = CustomContainer`.

Allowed values: `springboot`.

EnvironmentVariables - Environment Variables

`string`. Optional. Use when `Action = Deploy`.

The environment variables to be entered using the syntax `-key value` (for example: `-CUSTOMER_NAME Contoso -WEBSITE_TIME_ZONE`). Values containing spaces should be enclosed in double quotes (for example: `"Eastern Standard Time"`).

JvmOptions - JVM Options

`string`. Optional. Use when `Action = Deploy && DeploymentType = Artifacts`.

Edits the app's JVM options. A string containing JVM options, such as `-Xms1024m -Xmx2048m`.

RuntimeVersion - Runtime Version

`string`. Optional. Use when `Action = Deploy && DeploymentType = Artifacts`. Allowed values: `Java_8` (Java 8), `Java_11` (Java 11), `NetCore_31` (.Net Core 3.1). Default value: `Java_11`.

The runtime version on which the app will run.

DotNetCoreMainEntryPath - Main Entry Path

`string`. Optional. Use when `RuntimeVersion = NetCore_31`.

The path to the .NET executable relative to the zip root.

Version - Version

`string`. Optional. Use when `Action = Deploy`.

The deployment version. If not set, the version is left unchanged.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

testEndpoint

After the 'Deploy' action only. Contains private URL for accessing the updated deployment.

Remarks

Use this task to deploy applications to [Azure Spring Apps](#) and to manage Azure Spring Cloud [deployments](#).

Examples

The following examples demonstrate common usage scenarios. For more information, see [Automate application deployments to Azure Spring Apps](#).

Deleting a staging deployment

The "Delete Staging Deployment" action allows you to delete the deployment not receiving production traffic. This frees up resources used by that deployment and makes room for a new staging deployment:

```
yml

variables:
    azureSubscription: Contoso

steps:
- task: AzureSpringCloud@0
  continueOnError: true # Don't fail the pipeline if a staging deployment
  doesn't already exist.
  inputs:
    continueOnError: true
    inputs:
      azureSubscription: $(azureSubscription)
      Action: 'Delete Staging Deployment'
```

```
AppName: customer-api
AzureSpringCloud: contoso-dev-az-spr-cld
```

Deploying

To production

The following example deploys to the default production deployment in Azure Spring Apps. This is the only possible deployment scenario when using the Basic SKU:

ⓘ Note

The package search pattern should only return exactly one package. If the build task produces multiple JAR packages such as *sources.jar* and *javadoc.jar*, you need to refine the search pattern so that it only matches the application binary artifact.

```
yml
```

```
variables:
  azureSubscription: Contoso

steps:
- task: AzureSpringCloud@0
  inputs:
    azureSubscription: $(azureSubscription)
    Action: 'Deploy'
    AzureSpringCloud: contoso-dev-az-spr-cld
    AppName: customer-api
    UseStagingDeployment: false
    DeploymentName: default
    Package: '$(System.DefaultWorkingDirectory)/**/*customer-api*.jar'
```

Blue-green

The following example deploys to a pre-existing staging deployment. This deployment will not receive production traffic until it is set as a production deployment.

```
yml
```

```
variables:
  azureSubscription: Contoso

steps:
- task: AzureSpringCloud@0
```

```

inputs:
  azureSubscription: $(azureSubscription)
  Action: 'Deploy'
  AzureSpringCloud: contoso-dev-az-spr-cld
  AppName: customer-api
  UseStagingDeployment: true
  Package: '$(System.DefaultWorkingDirectory)/**/*customer-api*.jar'

```

For more on blue-green deployments, including an alternative approach, see [Blue-green deployment strategies](#).

Setting production deployment

The following example sets the current staging deployment as production, effectively swapping which deployment receives production traffic.

```

yml

variables:
  azureSubscription: Contoso

steps:
- task: AzureSpringCloud@0
  inputs:
    azureSubscription: $(azureSubscription)
    Action: 'Set Production'
    AzureSpringCloud: contoso-dev-az-spr-cld
    AppName: customer-api
    UseStagingDeployment: true

```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.104.1 or greater

Requirement	Description
Task category	Deploy

SqlAzureDacpacDeployment@1 - Azure SQL Database deployment v1 task

Article • 09/26/2023

Use this task to deploy an Azure SQL Database using DACPAC, or run scripts using SQLCMD.

Syntax

YAML

```
# Azure SQL Database deployment v1
# Deploy an Azure SQL Database using DACPAC or run scripts using SQLCMD.
- task: SqlAzureDacpacDeployment@1
  inputs:
    #azureConnectionType: 'ConnectedServiceNameARM' # 'ConnectedServiceName'
    | 'ConnectedServiceNameARM'. Alias: ConnectedServiceNameSelector. Azure
    Service Connection Type. Default: ConnectedServiceNameARM.
    #azureClassicSubscription: # string. Alias: ConnectedServiceName.
    Required when ConnectedServiceNameSelector = ConnectedServiceName. Azure
    Classic Subscription.
    azureSubscription: # string. Alias: ConnectedServiceNameARM. Required
    when ConnectedServiceNameSelector = ConnectedServiceNameARM. Azure
    Subscription.
    # SQL Database
    AuthenticationType: 'server' # 'server' | 'aadAuthenticationPassword' |
    'aadAuthenticationIntegrated' | 'connectionString' | 'servicePrincipal'.
    Required. Authentication Type. Default: server.
    #ServerName: # string. Required when AuthenticationType = server ||
    AuthenticationType = aadAuthenticationPassword || AuthenticationType =
    aadAuthenticationIntegrated || AuthenticationType = servicePrincipal. Azure
    SQL Server.
    #DatabaseName: # string. Required when AuthenticationType = server ||
    AuthenticationType = aadAuthenticationPassword || AuthenticationType =
    aadAuthenticationIntegrated || AuthenticationType = servicePrincipal.
    Database.
    SqlUsername: # string. Required when AuthenticationType = server. Login.
    SqlPassword: # string. Required when AuthenticationType = server.
    Password.
    #aadSqlUsername: # string. Required when AuthenticationType =
    aadAuthenticationPassword. Login.
    #aadSqlPassword: # string. Required when AuthenticationType =
    aadAuthenticationPassword. Password.
    #ConnectionString: # string. Required when AuthenticationType =
    connectionString. Connection String.
    # Deployment Package
    deployType: 'DacpacTask' # 'DacpacTask' | 'SqlTask' | 'InlineSqlTask'.
    Alias: TaskNameSelector. Required. Deploy type. Default: DacpacTask.
```

```

    DeploymentAction: 'Publish' # 'Publish' | 'Extract' | 'Export' |
    'Import' | 'Script' | 'DriftReport' | 'DeployReport'. Required when
    TaskNameSelector = DacpacTask. Action. Default: Publish.
        #DacpacFile: # string. Required when DeploymentAction = Publish ||
    DeploymentAction = Script || DeploymentAction = DeployReport. DACPAC File.
        #BACPACFile: # string. Required when DeploymentAction = Import. BACPAC
    File.
        #SqlFile: # string. Required when TaskNameSelector = SqlTask. SQL
    Script.
        #SqlInline: # string. Required when TaskNameSelector = InlineSqlTask.
    Inline SQL Script.
        #PublishProfile: # string. Optional. Use when TaskNameSelector =
    DacpacTask || DeploymentAction = Script || DeploymentAction = DeployReport.
    Publish Profile.
        #AdditionalArguments: # string. Optional. Use when TaskNameSelector =
    DacpacTask || DeploymentAction = Extract || DeploymentAction = Export ||
    DeploymentAction = Import || DeploymentAction = Script || DeploymentAction =
    DeployReport || DeploymentAction = DriftReport. Additional SqlPackage.exe
    Arguments.
        #SqlAdditionalArguments: # string. Optional. Use when TaskNameSelector =
    SqlTask. Additional Invoke-Sqlcmd Arguments.
        #InlineAdditionalArguments: # string. Optional. Use when
    TaskNameSelector = InlineSqlTask. Additional Invoke-Sqlcmd Arguments.
    # Firewall
        IpDetectionMethod: 'AutoDetect' # 'AutoDetect' | 'IPAddressRange'.
    Required. Specify Firewall Rules Using. Default: AutoDetect.
        #StartIpAddress: # string. Required when IpDetectionMethod =
    IPAddressRange. Start IP Address.
        #EndIpAddress: # string. Required when IpDetectionMethod =
    IPAddressRange. End IP Address.
        #DeleteFirewallRule: true # boolean. Delete Rule After Task Ends.
    Default: true.

```

Inputs

`azureConnectionType` - Azure Service Connection Type

Input alias: `ConnectedServiceNameSelector`. `string`. Allowed values:

`ConnectedServiceName` (Azure Classic), `ConnectedServiceNameARM` (Azure Resource Manager). Default value: `ConnectedServiceNameARM`.

`azureClassicSubscription` - Azure Classic Subscription

Input alias: `ConnectedServiceName`. `string`. Required when `ConnectedServiceNameSelector` = `ConnectedServiceName`.

Specifies the target Azure classic subscription for deploying SQL files.

azureSubscription - Azure Subscription

Input alias: `ConnectedServiceNameARM`. `string`. Required when

```
ConnectedServiceNameSelector = ConnectedServiceNameARM.
```

Specifies the target Azure Resource Manager subscription for deploying SQL files.

AuthenticationType - Authentication Type

`string`. Required. Allowed values: `server` (SQL Server Authentication), `aadAuthenticationPassword` (Active Directory - Password), `aadAuthenticationIntegrated` (Active Directory - Integrated), `connectionString` (Connection String), `servicePrincipal` (Service Principal). Default value: `server`.

Specifies the type of database authentication. It can be an SQL Server, Active Directory (integrated), Active Directory (password), connection string, or service principal authentication. Integrated authentication means that the agent accesses the database using its current Active Directory account context.

Specify the option to connect to the Azure SQL Server database. You can provide the Azure SQL Server database details, the SQL Server connection string, AD Authentication (password or integrated), or use a service principal. For SQL Server authentication, use the SQL Server's user credentials. For AD authentication, use the credentials for the AD user configured to the SQL Server.

ServerName - Azure SQL Server

`string`. Required when `AuthenticationType = server` || `AuthenticationType = aadAuthenticationPassword` || `AuthenticationType = aadAuthenticationIntegrated` || `AuthenticationType = servicePrincipal`.

Specifies the Azure SQL Server name, like `Fabrikam.database.windows.net,1433` or `Fabrikam.database.windows.net`.

DatabaseName - Database

`string`. Required when `AuthenticationType = server` || `AuthenticationType = aadAuthenticationPassword` || `AuthenticationType = aadAuthenticationIntegrated` || `AuthenticationType = servicePrincipal`.

Specifies the name of the Azure SQL database where the files are deployed.

SqlUsername - Login

`string`. Required when `AuthenticationType = server`.

Specifies the Azure SQL Server administrator login.

SqlPassword - Password

`string`. Required when `AuthenticationType = server`.

Specifies the password for the Azure SQL Server administrator. Variables defined in the build or release pipelines as `$(passwordVariable)` are accepted. You can mark the variable type as `secret` to secure it.

aadSqlUsername - Login

`string`. Required when `AuthenticationType = aadAuthenticationPassword`.

Specifies the Active Directory user name.

aadSqlPassword - Password

`string`. Required when `AuthenticationType = aadAuthenticationPassword`.

Specifies the password for the Active Directory user. Variables defined in the build or release pipelines as `$(passwordVariable)` are accepted. You can mark the variable type as `secret` to secure it.

ConnectionString - Connection String

`string`. Required when `AuthenticationType = connectionString`.

Specifies the Azure SQL Server connection string, like

```
Server=testServer.database.windows.net;Database=testdb;User  
ID=AccountPlaceholder;Password=$(securePassword);
```

deployType - Deploy type

Input alias: `TaskNameSelector`. `string`. Required. Allowed values: `DacpacTask` (SQL DACPAC File), `SqlTask` (SQL Script File), `InlineSqlTask` (Inline SQL Script). Default value: `DacpacTask`.

DeploymentAction - Action

`string`. Required when `TaskNameSelector = DacpacTask`. Allowed values: `Publish`, `Extract`, `Export`, `Import`, `Script`, `DriftReport` (Drift Report), `DeployReport` (Deploy Report). Default value: `Publish`.

Specifies one of the SQL actions from the list. Learn more about the [SQL actions list](#).

DacpacFile - DACPAC File

`string`. Required when `DeploymentAction = Publish || DeploymentAction = Script || DeploymentAction = DeployReport`.

Specifies the location of the DACPAC file on the automation agent or on a UNC path that's accessible to the automation agent, like

`\BudgetIT\Web\Deploy\FabrikamDB.dacpac`. Predefined system variables, like `$(agent.releaseDirectory)`, can also be used.

BacpacFile - BACPAC File

`string`. Required when `DeploymentAction = Import`.

Specifies the location of the BACPAC file on the automation agent or on a UNC path that's accessible to the automation agent, like

`\BudgetIT\Web\Deploy\FabrikamDB.bacpac`. Predefined system variables, like `$(agent.releaseDirectory)`, can also be used.

SqlFile - SQL Script

`string`. Required when `TaskNameSelector = SqlTask`.

Specifies the location of the SQL script file on the automation agent or on a UNC path that's accessible to the automation agent, like `\BudgetIT\Web\Deploy\FabrikamDB.sql`. Predefined system variables, like `$(agent.releaseDirectory)`, can also be used.

SqlInline - Inline SQL Script

`string`. Required when `TaskNameSelector = InlineSqlTask`.

Specifies the SQL script to execute on the previously selected database.

PublishProfile - Publish Profile

`string`. Optional. Use when `TaskNameSelector = DacpacTask || DeploymentAction =`

```
Script || DeploymentAction = DeployReport.
```

Provides fine-grained control over Azure SQL database creation or upgrades. Specifies the path to the publish profile XML file on the automation agent machine or on a UNC share. If the publish profile contains secrets, like credentials, upload it to the [secure files](#) library where it is securely stored with encryption. Next, use the [Download secure file](#) task at the start of your pipeline to download it to the agent machine when the pipeline runs. Delete it when the pipeline is complete. Predefined system variables, like `$(agent.buildDirectory)` or `$(agent.releaseDirectory)`, can also be used.

AdditionalArguments - Additional SqlPackage.exe Arguments

```
string. Optional. Use when TaskNameSelector = DacpacTask || DeploymentAction =
Extract || DeploymentAction = Export || DeploymentAction = Import ||
DeploymentAction = Script || DeploymentAction = DeployReport || DeploymentAction =
DriftReport.
```

Specifies the additional `SqlPackage.exe` arguments that will be applied when deploying the Azure SQL database if the DACPAC option is selected, like `/p:IgnoreAnsiNulls=True` `/p:IgnoreComments=True`. These arguments will override the settings in the publish profile XML file (if provided).

SqlAdditionalArguments - Additional Invoke-Sqlcmd Arguments

```
string. Optional. Use when TaskNameSelector = SqlTask.
```

Specifies the additional Invoke-Sqlcmd arguments that are applied when executing the given SQL query on the Azure SQL database, like `-ConnectionTimeout 100` `-OutputSqlErrors`.

InlineAdditionalArguments - Additional Invoke-Sqlcmd Arguments

```
string. Optional. Use when TaskNameSelector = InlineSqlTask.
```

Specifies the additional Invoke-Sqlcmd arguments that are applied when executing the given SQL query on the Azure SQL Database, like `-ConnectionTimeout 100` `-OutputSqlErrors`.

IpDetectionMethod - Specify Firewall Rules Using

```
string. Required. Allowed values: AutoDetect, IPAddressRange. Default value:
AutoDetect.
```

For the task to run, the IP address of the automation agent must be added to the **Allowed IP Addresses** in the Azure SQL Server's firewall. Select auto-detect to automatically add the firewall exception for the range of the possible IP address of the automation agent, or specify the range explicitly.

StartIpAddress - Start IP Address

`string`. Required when `IpDetectionMethod = IPAddressRange`.

Specifies the starting IP address of the automation agent machine pool, like `196.21.30.50`.

EndIpAddress - End IP Address

`string`. Required when `IpDetectionMethod = IPAddressRange`.

Specifies the ending IP address of the automation agent machine pool, like `196.21.30.65`.

DeleteFirewallRule - Delete Rule After Task Ends

`boolean`. Default value: `true`.

If selected, after the task ends, the IP addresses specified here are deleted from the **Allowed IP Addresses** list in the Azure SQL Server's firewall.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

SqlDeploymentOutputFile

The generated output file path when the deployment package action is `Extract`, `Export`, `Script`, `DriftReport`, or `DeployReport`.

Remarks

Use this task to deploy an Azure SQL database using a DACPAC, or run scripts using SQLCMD.

ⓘ Important

This task is only supported in a Windows environment. If you are trying to use Azure Active Directory (Azure AD) integrated authentication, you must create a private agent. Azure AD integrated authentication is not supported for hosted agents.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: sqlpackage
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.103.0 or greater
Task category	Deploy

AzureVmssDeployment@0 - Azure VM scale set deployment v0 task

Article • 09/26/2023

This task deploys a Virtual Machine scale set image.

Syntax

YAML

```
# Azure VM scale set deployment v0
# Deploy a virtual machine scale set image.
- task: AzureVmssDeployment@0
  inputs:
    # Azure Details
    azureSubscription: # string. Alias: ConnectedServiceName. Required.
    Azure subscription.
    action: 'Update image' # 'Update image' | 'Configure application
    startup'. Required. Action. Default: Update image.
    vmssName: # string. Required. Virtual Machine scale set name.
    vmssOsType: # 'Windows' | 'Linux'. Required. OS type.
    # Image Details
    #imageUrl: # string. Required when action = Update image || action =
    UpdateImage. Image URL.
    # Configure start-up
    #customScriptsDirectory: # string. Optional. Use when action = Configure
    application startup || action = Update image || action = UpdateImage. Custom
    script directory.
    #customScript: # string. Optional. Use when action = Configure
    application startup || action = Update image || action = UpdateImage.
    Command.
    #customScriptArguments: # string. Optional. Use when action = Configure
    application startup || action = Update image || action = UpdateImage.
    Arguments.
    #customScriptsStorageAccount: # string. Optional. Use when action =
    Configure application startup || action = Update image || action =
    UpdateImage. Azure storage account where custom scripts will be uploaded.
    # Advanced
    #skipArchivingCustomScripts: false # boolean. Skip Archiving custom
    scripts. Default: false.
```

Inputs

`azureSubscription` - Azure subscription

Input alias: `ConnectedServiceName`. `string`. Required.

Specifies the Azure Resource Manager subscription for the scale set.

action - Action

`string`. Required. Allowed values: `Update image` (Update VM Scale set by using an image), `Configure application startup` (Run Custom Script VM extension on VM scale set). Default value: `Update image`.

Updates a VM scale set by the chosen method, using a VHD image and/or by running deployment/install scripts using the Custom Script VM Extension.

The VHD image approach is better for scaling quickly and doing rollback. When a VM scale set is created by using a custom image, it can be updated by a VHD image. The update will fail if the VM scale set was created by using a platform/gallery image available in Azure.

The Custom Script VM Extension approach is useful for post deployment configuration, software installation, or any other configuration/management task. The Custom Script VM Extension approach can be used for a VM scale set created by using either a custom image or a platform/gallery image.

vmssName - Virtual Machine scale set name

`string`. Required.

Specifies the name of the VM scale setting to update. Use either a VHD image or a Custom Script VM Extension.

vmssOsType - OS type

`string`. Required. Allowed values: `Windows`, `Linux`.

Specifies the operating system type of the VM scale set.

imageUrl - Image URL

`string`. Required when `action = Update image || action = UpdateImage`.

Specifies the URL of the VHD image. If it's an Azure storage blob URL, the storage account location is the same as the scale set location.

customScriptsDirectory - Custom script directory

`string`. Optional. Use when `action = Configure application startup || action =`

```
Update image || action = UpdateImage.
```

Optional. The path to the directory containing the custom script(s) that are run by using the Custom Script VM Extension. The extension approach is useful for post deployment configuration, application/software installation, or any other application configuration/management task. For example, the script can set a machine level environment variable which the application uses, like database connection strings.

customScript - Command

```
string. Optional. Use when action = Configure application startup || action =  
Update image || action = UpdateImage.
```

Optional. The script that is run by using the Custom Script VM Extension. This script can invoke other scripts in the directory and is invoked with the arguments passed below. In conjunction with such arguments, this script can be used to execute commands.

For example:

1. `Update-Database.ConnectionStrings.ps1 -clusterType dev -user $(dbUser) -password $(dbUserPwd)` updates the connection string in `web.config` of the web application.
2. `install-secrets.sh --key-vault-type prod -key serviceprincipalkey` creates an encrypted file containing a service principal key.

customScriptArguments - Arguments

```
string. Optional. Use when action = Configure application startup || action =  
Update image || action = UpdateImage.
```

Optional. The custom script will be invoked with arguments passed. Build/release variables can be used, which makes it easy to use secrets.

customScriptsStorageAccount - Azure storage account where custom scripts will be uploaded

```
string. Optional. Use when action = Configure application startup || action =  
Update image || action = UpdateImage.
```

Optional. The Custom Script Extension downloads and executes the provided scripts on each virtual machine in the VM scale set. These scripts will be stored in the pre-existing ARM storage account specified here.

`skipArchivingCustomScripts` - Skip Archiving custom scripts

`boolean`. Default value: `false`.

Optional. By default, this task creates a compressed archive of the directory containing the custom scripts. This improves performance and reliability while uploading to Azure storage. If not selected, archiving will not be done and all files will be individually uploaded.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to deploy a Virtual Machine scale set image.

The script execution is reported as successful, however the VMSS instances are not updated

Scale sets have an upgrade policy that determine how VMs are brought up-to-date with the latest scale set model, and if the upgrade policy is set to manual you must manually upgrade each VM. For more information, see [How to bring VMs up-to-date with the latest scale set model](#). You can change the update policy or manually upgrade each VM. For example, to upgrade the policy to `Automatic`, use the following Az CLI command:

```
az vmss update --set upgradePolicy.mode=Automatic -g <resource group name> -n <vmss name>
```

Error: 'Permission denied: Script is not executable'

This issue occurs if you try to run a custom script, but the script isn't executable.

To resolve the issue, first make sure that the `customScript` input doesn't have `./` or anything else before the script name `'test.sh'`:

```
yml  
  customScript: 'test.sh'
```

Next, try adding a command line task before the virtual machine scale set task:

```
yml  
  - task: CmdLine@2  
    inputs:  
      script: 'chmod 777 $(System.DefaultWorkingDirectory)/test.sh'
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.0.0 or greater
Task category	Deploy

AzureWebApp@1 - Azure Web App v1 task

Article • 09/26/2023

This task deploys an Azure Web App for Linux or Windows.

Syntax

YAML

```
# Azure Web App v1
# Deploy an Azure Web App for Linux or Windows.
- task: AzureWebApp@1
  inputs:
    azureSubscription: # string. Required. Azure subscription.
    appType: # 'webApp' | 'webAppLinux'. Required. App type.
    appName: # string. Required. App name.
    #deployToSlotOrASE: false # boolean. Optional. Use when appType != "".
    Deploy to Slot or App Service Environment. Default: false.
    #resourceGroupName: # string. Required when deployToSlotOrASE = true.
    Resource group.
    #slotName: 'production' # string. Required when deployToSlotOrASE =
    true. Slot. Default: production.
    package: '$(System.DefaultWorkingDirectory)/**/*.zip' # string.
    Required. Package or folder. Default:
    $(System.DefaultWorkingDirectory)/**/*.zip.
    #customDeployFolder: # string. Optional. Use when package EndsWith .war.
    Custom Deploy Folder.
    #runtimeStack: # string. Optional. Use when appType = webAppLinux.
    Runtime stack.
    #startUpCommand: # string. Optional. Use when appType = webAppLinux.
    Startup command.
    # Application and Configuration Settings
    #customWebConfig: # string. Optional. Use when appType != webAppLinux &&
    package NotEndsWith .war. Generate web.config parameters for Python,
    Node.js, Go and Java apps.
    #appSettings: # string. App settings.
    #configurationStrings: # string. Configuration settings.
    # Additional Deployment Options
    #deploymentMethod: 'auto' # 'auto' | 'zipDeploy' | 'runFromPackage'.
    Required when appType != webAppLinux && appType != "" && package NotEndsWith
    .war && package NotEndsWith .jar. Deployment method. Default: auto.
```

Inputs

azureSubscription - Azure subscription

`string`. Required.

Specifies the [Azure Resource Manager subscription connection](#) for the deployment.

appType - App type

`string`. Required. Allowed values: `webApp` (Web App on Windows), `webAppLinux` (Web App on Linux).

Specifies the Azure Web App type.

appName - App name

`string`. Required.

Specifies the name of an existing Azure App Service. Only app services that are based on the selected app type will be listed.

deployToSlotOrASE - Deploy to Slot or App Service Environment

`boolean`. Optional. Use when `appType != ""`. Default value: `false`.

Selects the option to deploy to an existing deployment slot or an Azure App Service Environment.

For both targets, the task needs a resource group name.

If the deployment target is a slot, the default is the production slot. Any other existing slot name can also be provided.

If the deployment target is an Azure App Service Environment, leave the slot name as 'production', and specify the resource group name.

resourceGroupName - Resource group

`string`. Required when `deployToSlotOrASE = true`.

The resource group name is required when the deployment target is either a deployment slot or an Azure App Service Environment.

Specifies the Azure resource group that contains the Azure App Service indicated above.

slotName - Slot

`string`. Required when `deployToSlotOrASE = true`. Default value: `production`.

Specifies an existing slot, excluding the production slot.

package - Package or folder

`string`. Required. Default value: `$(System.DefaultWorkingDirectory)/**/*.zip`.

The file path to the package or folder that contains App Service content generated by MSBuild, a compressed zip file, or a war file. Variables ([Build | Release](#)) and wildcards are supported. For example, `$(System.DefaultWorkingDirectory)/**/*.zip` or `$(System.DefaultWorkingDirectory)/**/*.war`.

customDeployFolder - Custom Deploy Folder

`string`. Optional. Use when `package` `EndsWith` `.war`.

Specifies the custom folder name you want to deploy to.

If the field is empty, the package is deployed to

`<appname>.azurewebsites.net/<warpackagename>`.

If ROOT is entered, the package is deployed to `<appname>.azurewebsites.net`.

In all other instances, it is deployed to `<appname>.azurewebsites.net/<customWarName>`.

runtimeStack - Runtime stack

`string`. Optional. Use when `appType = webAppLinux`.

Web App on Linux offers two different options to publish your application: custom image deployment (Web App for Containers) and app deployment with a built-in platform image (Web App on Linux). This parameter is only available when [Linux Web App](#) is selected as an app type in the task.

startUpCommand - Startup command

`string`. Optional. Use when `appType = webAppLinux`.

Specifies the start up command.

For example:

`dotnet run`

`dotnet filename.dll`.

customWebConfig - Generate web.config parameters for Python, Node.js, Go and Java apps

`string`. Optional. Use when `appType != webAppLinux && package NotEndsWith .war`.

A standard web.config will be generated and deployed to Azure App Service if the application does not have one. The values in web.config vary based on the application framework, and they can be edited. For example, for the node.js application, web.config will have a startup file and iis_node module values. This edit feature is only for the [generated web.config](#).

`appSettings` - App settings

`string`.

Specify the web app application settings using the syntax `-key value` (for example: `-Port 5000 -RequestTimeout 5000 -WEBSITE_TIME_ZONE`). Enclose values that contain spaces in double quotes (for example: `"Eastern Standard Time"`).

`configurationStrings` - Configuration settings

`string`.

Specify the web app configuration settings using the syntax `-key value` (for example: `-phpVersion 5.6 -linuxFxVersion: node|6.11`). Enclose values that contain spaces in double quotes.

`deploymentMethod` - Deployment method

`string`. Required when `appType != webAppLinux && appType != "" && package NotEndsWith .war && package NotEndsWith .jar`. Allowed values: `auto` (Auto-detect), `zipDeploy` (Zip Deploy), `runFromPackage` (Run From Package). Default value: `auto`.

Choose the [deployment method](#) for the app. Acceptable values are `auto`, `zipDeploy`, and `runFromPackage`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

AppServiceApplicationUrl

The application URL of the selected Azure App Service.

Remarks

Use this task to deploy web applications to Azure App Service.

Deployment methods

Several deployment methods are available in this task. **Auto** is the default option.

To change the package-based deployment option in designer task, expand **Additional Deployment Options** and enable **Select Deployment Method**.

Based on the type of Azure App Service and Azure Pipelines agent, the task chooses a suitable deployment technology. The different deployment technologies used by the task are:

- Kudu REST APIs
- Zip Deploy
- RunFromPackage

By default, the task tries to select the appropriate deployment technology given the input package, app service type, and agent OS.

- When the app service type is Web App on Linux App, use **Zip Deploy**
- If a War file is provided, use **War Deploy**
- If a Jar file is provided, use **Run From Package**
- For all others, use **Run From Zip** (via Zip Deploy)

On a non-Windows agent (for any app service type), the task relies on [Kudu REST APIs](#) to deploy the web app.

Kudu REST APIs

[Kudu REST APIs](#) work on Windows or Linux automation agents when the target is Web App on Windows, Web App on Linux (built-in source), or Function App. The task uses Kudu to copy files to the Azure App Service.

Zip Deploy

Creates a .zip deployment package of the chosen package or folder. The file contents are then deployed to the wwwroot folder of the function app in Azure App Service. This option overwrites all existing contents in the wwwroot folder. For more information, see [Zip deployment for Azure Functions](#).

RunFromPackage

Creates the same deployment package as Zip Deploy. However, instead of deploying files to the wwwroot folder, the entire package is mounted by the Azure Functions runtime. With this option, files in the wwwroot folder become read-only. For more information, see [Run your Azure Functions from a package file](#).

Error: Could not fetch access token for Azure. Verify if the Service Principal used is valid and not expired.

The task uses the service principal in the service connection to authenticate with Azure. If the service principal has expired or doesn't have permissions to the App Service, the task fails with this error. Verify the validity of the service principal used and that it's present in the app registration. For more information, see [Use role-based access control to manage access to your Azure subscription resources](#). This blog post [also](#) contains more information about using service principal authentication.

SSL error

If you want to use a certificate in App Service, the certificate must be signed by a trusted certificate authority. If your web app gives you certificate validation errors, you're probably using a self-signed certificate. Set a variable named `VSTS_ARM_REST_IGNORE_SSL_ERRORS` to the value `true` in the build or release pipeline to resolve the error.

A release hangs for long time and then fails

This problem could be the result of insufficient capacity in your App Service plan. To resolve this problem, you can scale up the App Service instance to increase available CPU, RAM, and disk space or try with a different App Service plan.

5xx error codes

If you're seeing a 5xx error, check the status of your Azure service [↗](#).

Azure Function suddenly stopped working

Azure Functions may suddenly stop working if more than one year has passed since the last deployment. If you deploy with "RunFromPackage" in "deploymentMethod", a SAS with an expiration date of 1 year is generated and set as the value of "WEBSITE_RUN_FROM_PACKAGE" in the application configuration. Azure Functions uses this SAS to reference the package file for function execution, so if the SAS has expired, the function will not be executed. To resolve this issue, deploy again to generate a SAS with an expiration date of one year.

Error: No package found with specified pattern

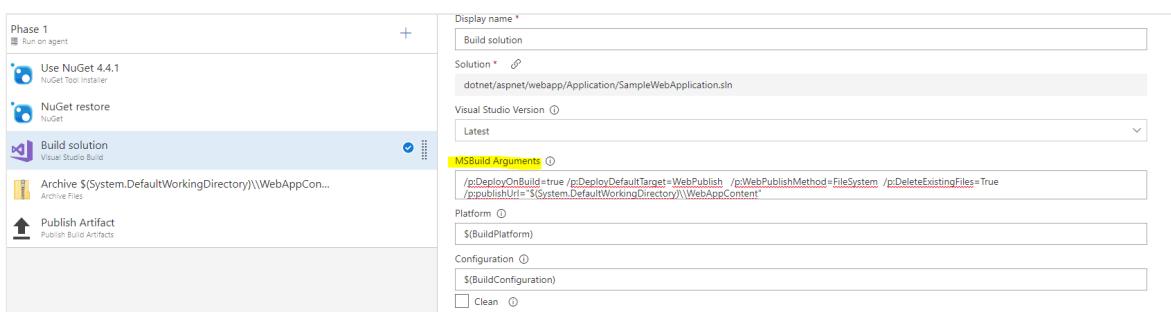
Check if the package mentioned in the task is published as an artifact in the build or a previous stage and downloaded in the current job.

Error: Publish using zip deploy option is not supported for msBuild package type

Web packages created via the MSBuild task (with default arguments) have a nested folder structure that can be deployed correctly only by Web Deploy. The publish-to-zip deployment option can't be used to deploy those packages. To convert the packaging structure, take these steps:

1. In the Build solution task, change the **MSBuild Arguments** to

```
/p:DeployOnBuild=true /p:DeployDefaultTarget=WebPublish  
/p:WebPublishMethod=FileSystem /p:DeleteExistingFiles=True  
/p:publishUrl="$(System.DefaultWorkingDirectory)\\WebAppContent":
```

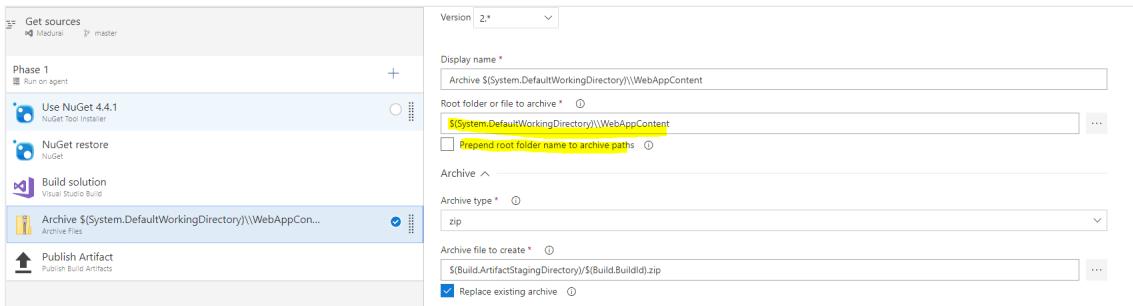


2. Add an Archive task and change the values as follows:

- a. Change **Root folder or file to archive** to

```
$(System.DefaultWorkingDirectory)\\WebAppContent .
```

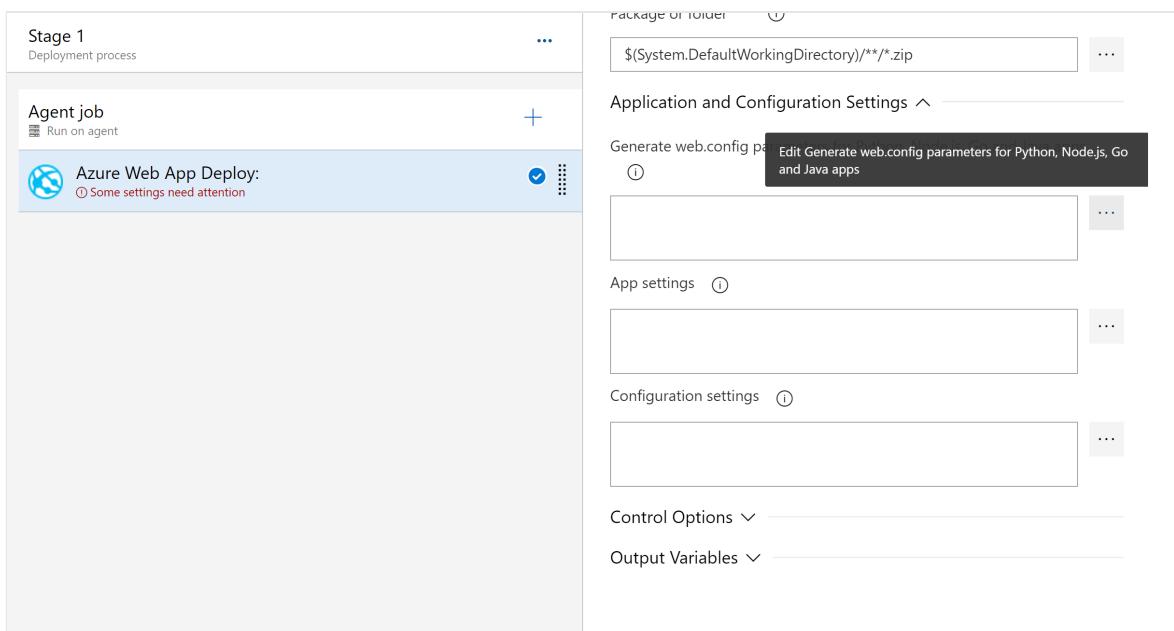
- b. Clear the **Prepend root folder name to archive paths** check box:



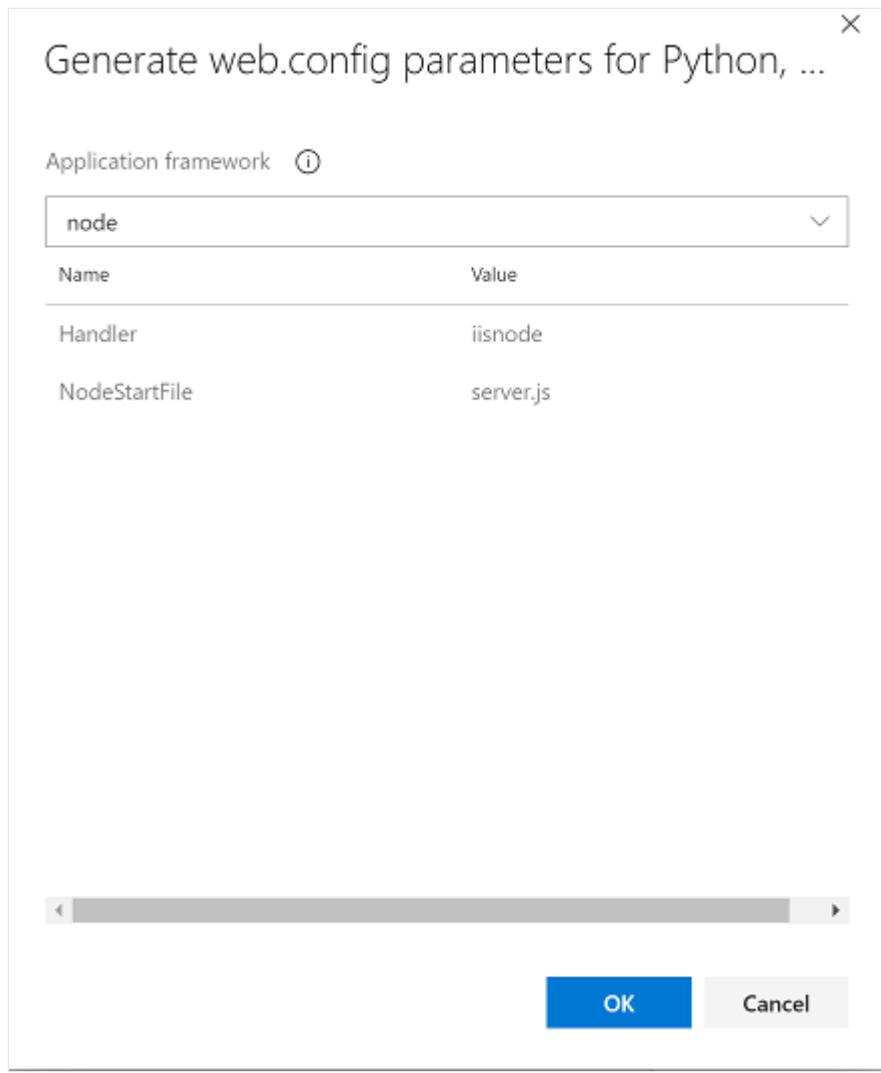
Web app deployment on Windows is successful but the app is not working

This may be because web.config is not present in your app. You can either add a web.config file to your source or auto-generate one using **Application and Configuration Settings**.

- Click on the task and go to **Generate web.config parameters for Python, Node.js, Go and Java apps**.



- Click on the more button ... to edit the parameters.



- Select your application type from the drop down.
- Click OK. This will populate the web.config parameters required to generate web.config.

Web app deployment on App Service Environment (ASE) is not working

- Ensure that the Azure DevOps build agent is on the same VNET (subnet can be different) as the Internal Load Balancer (ILB) of ASE. This will enable the agent to pull code from Azure DevOps and deploy to ASE.
- If you are using Azure DevOps, the agent doesn't need to be accessible from the internet but needs only outbound access to connect to Azure DevOps Service.
- If you are using TFS/Azure DevOps Server deployed in a Virtual Network, the agent can be completely isolated.
- The build agent must be configured with the DNS configuration of the Web App it needs to deploy to. The private resources in the Virtual Network don't have entries in Azure DNS, so this needs to be added to the host's file on the agent machine.

- If a self-signed certificate is used for the ASE configuration, the `-allowUntrusted` option needs to be set in the deploy task for MSDeploy. It is also recommended to set the variable `VSTS_ARM_REST_IGNORE_SSL_ERRORS` to `true`. If a certificate from a certificate authority is used for ASE configuration, this should not be necessary.

How should I configure my service connection?

This task requires an [Azure Resource Manager service connection](#).

How should I configure web job deployment with Application Insights?

When you're deploying to an App Service, if you have [Application Insights](#) configured and you've enabled `Remove additional files at destination`, you also need to enable `Exclude files from the App_Data folder`. Enabling this option keeps the Application Insights extension in a safe state. This step is required because the Application Insights continuous WebJob is installed into the App_Data folder.

How should I configure my agent if it's behind a proxy while I'm deploying to App Service?

If your self-hosted agent requires a web proxy, you can inform the agent about the proxy during configuration. Doing so allows your agent to connect to Azure Pipelines or Azure DevOps Server through the proxy. [Learn more about running a self-hosted agent behind a web proxy](#).

Examples

Following is an example YAML snippet to deploy web application to the Azure Web App Service running on Windows.

YAML

```
variables:  
  azureSubscription: Contoso  
  # To ignore SSL error uncomment the below variable  
  # VSTS_ARM_REST_IGNORE_SSL_ERRORS: true  
  
steps:  
  - task: AzureWebApp@1  
    displayName: Azure Web App Deploy
```

```
inputs:  
  azureSubscription: $(azureSubscription)  
  appName: samplewebapp  
  package: $(System.DefaultWorkingDirectory)/**/*.zip
```

To deploy Web App on Linux, add the `appType` parameter and set it to `appType: webAppLinux`.

To specify the deployment method as Zip Deploy, add the parameter `deploymentMethod: zipDeploy`. Another supported value for this parameter is `runFromPackage`.

If not specified, `auto` is the default value.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.104.1 or greater
Task category	Deploy

AzureWebAppContainer@1 - Azure Web App for Containers v1 task

Article • 09/26/2023

This task deploys containers to Azure App Service.

Syntax

YAML

```
# Azure Web App for Containers v1
# Deploy containers to Azure App Service.
- task: AzureWebAppContainer@1
  inputs:
    azureSubscription: # string. Required. Azure subscription.
    appName: # string. Required. App name.
    #deployToSlotOrASE: false # boolean. Deploy to Slot or App Service Environment. Default: false.
    #resourceGroupName: # string. Required when deployToSlotOrASE = true. Resource group.
    #slotName: 'production' # string. Required when deployToSlotOrASE = true. Slot. Default: production.
    #containers: # string. Alias: imageName. Image name.
    #multicontainerConfigFile: # string. Configuration File.
    #containerCommand: # string. Startup command.
    # Application and Configuration Settings
    #appSettings: # string. App settings.
    #configurationStrings: # string. Configuration settings.
```

Inputs

`azureSubscription` - Azure subscription

`string`. Required.

The name of the [Azure Resource Manager subscription](#) for the deployment.

`appName` - App name

`string`. Required.

Specifies the name of an existing Azure App Service. Only app services based on the selected app type will be listed.

`deployToSlotOrASE` - Deploy to Slot or App Service Environment

`boolean`. Default value: `false`.

Selects the option to deploy to an existing deployment slot or an Azure App Service Environment.

For both targets, the task needs a resource group name.

If the deployment target is a slot, the default is the production slot. Any other existing slot name can also be provided.

If the deployment target is an Azure App Service Environment, leave the slot name as `production`, and specify the resource group name.

`resourceGroupName` - Resource group

`string`. Required when `deployToSlotOrASE = true`.

The resource group name is required when the deployment target is either a deployment slot or an Azure App Service Environment.

Specifies the Azure resource group that contains the Azure App Service indicated above.

`slotName` - Slot

`string`. Required when `deployToSlotOrASE = true`. Default value: `production`.

Specifies an existing slot, excluding the production slot.

`containers` - Image name

Input alias: `imageName`. `string`.

Specifies the fully qualified container image name. For example, `myregistry.azurecr.io/nginx:latest` or `python:3.7.2-alpine/`.

For a multi-container scenario, multiple container image names can be provided.

`multicontainerConfigFile` - Configuration File

`string`.

The path of the Docker-Compose file. Must be a fully qualified path or a path relative to the default working directory.

`containerCommand` - Startup command

`string`.

Specifies the start up command.

For example:

```
dotnet run
```

```
dotnet filename.dll
```

appSettings - App settings

```
string.
```

Edits the web app application settings using the syntax -key value (for example: `-Port 5000 -RequestTimeout 5000 -WEBSITE_TIME_ZONE`). A value containing spaces should be enclosed in double quotes (for example: `"Eastern Standard Time"`).

configurationStrings - Configuration settings

```
string.
```

Edits the web app application settings using the syntax -key value (for example: `-phpVersion 5.6 -linuxFxVersion: node|6.11`). A value containing spaces should be enclosed in double quotes.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

AppServiceApplicationUrl

The application URL of the selected Azure App Service.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release

Requirement	Description
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.104.1 or greater
Task category	Deploy

Bash@3 - Bash v3 task

Article • 09/26/2023

Use this task to run a Bash script on macOS, Linux, or Windows.

Syntax

YAML

```
# Bash v3
# Run a Bash script on macOS, Linux, or Windows.
- task: Bash@3
  inputs:
    targetType: 'filePath' # 'filePath' | 'inline'. Type. Default:
    filePath.
    filePath: # string. Required when targetType = filePath. Script Path.
    #arguments: # string. Optional. Use when targetType = filePath.
    Arguments.
    #script: # string. Required when targetType = inline. Script.
    # Advanced
    #workingDirectory: # string. Working Directory.
    #failOnStderr: false # boolean. Fail on Standard Error. Default: false.
    #bashEnvValue: # string. Set value for BASH_ENV environment variable.
```

Inputs

`targetType` - Type

`string`. Allowed values: `filePath` (File Path), `inline`. Default value: `filePath`.

Targets script type: file path or inline.

`filePath` - Script Path

`string`. Required when `targetType = filePath`.

The path of the script to execute. This must be a fully qualified path or relative to `$(System.DefaultWorkingDirectory)`.

`arguments` - Arguments

`string`. Optional. Use when `targetType = filePath`.

The arguments passed to the shell script. Either ordinal parameters or named parameters.

script - Script

`string`. Required when `targetType = inline`. Default value: `# Write your commands here\n\necho 'Hello world'`.

The contents of the script.

workingDirectory - Working Directory

`string`.

Specifies the working directory in which you want to run the command. If you leave it empty, the working directory is [\\$\(Build.SourcesDirectory\)](#).

failOnStderr - Fail on Standard Error

`boolean`. Default value: `false`.

If this is true, this task will fail if any errors are written to the `StandardError` stream.

bashEnvValue - Set value for BASH_ENV environment variable

`string`.

If the input is specified, its value is expanded and used as the path of a startup file to execute before running the script. If the environment variable `BASH_ENV` has already been defined, the task will override this variable only for the current task. Learn more about [Bash Startup Files](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

The bash task has a shortcut in YAML: `steps.bash`.

yml

```
steps:  
- bash: string # Required as first property. An inline script.  
## Other task inputs
```

The Bash task will find the first Bash implementation on your system. Running `which bash` on Linux/macOS or `where bash` on Windows will give you an idea of which one it will select.

Info about Bash startup files

The Bash task invokes the Bash as a non-interactive, non-login shell. When Bash is started non-interactively, to run a shell script, the Bash looks for the variable `BASH_ENV` in the environment, unfolds its value if it appears there, and uses the value as the name of a file to read and execute.

There are several options for defining the `BASH_ENV` environment variable in a pipeline. Firstly, it's possible to set the `BASH_ENV` environment variable as a pipeline variable. In this case, each instance of the Bash task will try to unfold the value of the `BASH_ENV` variable and use its value.

YAML

```
variables:  
  BASH_ENV: "~/.profile"  
  
steps:  
- task: Bash@3  
  inputs:  
    targetType: 'inline'  
    script: env
```

Another option is to set `BASH_ENV` for one particular instance of the Bash task, there are two ways how this can be done:

The first way is to use the `bashEnvValue` task input, see an example for reference:

YAML

```
steps:
- task: Bash@3
  inputs:
    targetType: 'inline'
    script: env
    bashEnvValue: '~/.profile'
```

Another way is to set the `BASH_ENV` variable as an environment variable for the pipeline task via the `env` keyword, for example:

YAML

```
- task: Bash@3
  inputs:
    targetType: 'inline'
    script: env
  env:
    BASH_ENV: '~/.profile'
```

ⓘ Note

Note that if the `bashEnvValue` input is defined in the Bash task, the pipeline task will override the value of the `BASH_ENV` variable with the value from the `bashEnvValue` input in a case when the `BASH_ENV` environment variable was already defined in the environment.

Bash scripts checked into the repo should be set executable (`chmod +x`). Otherwise, the task will show a warning and `source` the file instead.

Examples

You can map in variables using the `env` parameter which is [common across all tasks](#), and is list of additional items to map into the process's environment. For example, secret variables are not automatically mapped. If you have a secret variable called `Foo`, you can map it in like this:

YAML

```
steps:
- task: Bash@3
  inputs:
    targetType: 'inline'
    script: echo $MYSECRET
```

```
env:  
  MYSECRET: $(Foo)
```

On macOS or Linux, the example above is equivalent to:

YAML

```
steps:  
- script: echo $MYSECRET  
env:  
  MYSECRET: $(Foo)
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.115.0 or greater
Task category	Utility

BatchScript@1 - Batch script v1 task

Article • 09/26/2023

Use this task to run a Windows `.bat` or `.cmd` script. Optionally, the `.bat` or `.cmd` script can permanently modify environment variables.

Syntax

YAML

```
# Batch script v1
# Run a Windows command or batch script and optionally allow it to change
# the environment.
- task: BatchScript@1
  inputs:
    filename: # string. Required. Path.
    #arguments: # string. Arguments.
    #modifyEnvironment: False # boolean. Modify Environment. Default: False.
    # Advanced
    #workingFolder: # string. Working folder.
    #failOnStandardError: false # boolean. Fail on Standard Error. Default:
    false.
```

Inputs

`filename` - Path

`string`. Required.

The path of the `.cmd` or `.bat` script to execute. This should be a fully qualified path or one relative to the default working directory. (Please note that the working directory could differ from `workingFolder`, which could be specified for this task.)

`arguments` - Arguments

`string`.

The arguments passed to the `.cmd` or `.bat` script.

`modifyEnvironment` - Modify Environment

`boolean`. Default value: `False`.

Determines whether environment variable modifications will affect subsequent tasks.

`workingFolder` - Working folder

`string`.

The current working directory when a script is run. This defaults to the folder where the script is located.

`failOnStandardError` - Fail on Standard Error

`boolean`. Default value: `false`.

If this is true, this task will fail if any errors are written to the `StandardError` stream.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to run a Windows .bat or .cmd script. Optionally, allow it to permanently modify environment variables.

ⓘ Note

This task is not compatible with Windows containers. If you need to run a batch script on a Windows container, use the [command line task](#) instead.

For information on supporting multiple platforms, see [cross platform scripting](#).

Examples

Create `test.bat` at the root of your repo:

bat

```
@echo off
echo Hello World from %AGENT_NAME%.
echo My ID is %AGENT_ID%.
echo AGENT_WORKFOLDER contents:
@dir %AGENT_WORKFOLDER%
echo AGENT_BUILDDIRECTORY contents:
@dir %AGENT_BUILDDIRECTORY%
echo BUILD_SOURCESDIRECTORY contents:
@dir %BUILD_SOURCESDIRECTORY%
echo Over and out.
```

To run this script, add the following task to your pipeline.

yml

```
- task: BatchScript@1
  inputs:
    filename: 'test.bat'
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: Cmd
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.83.0 or greater
Task category	Utility

PackerBuild@1 - Build machine image v1 task

Article • 09/26/2023

Use this task to build a machine image using Packer, which may be used for Azure Virtual machine scale set deployment.

ⓘ Note

This task does not support Azure Resource Manager authentication with workflow identity federation.

Syntax

YAML

```
# Build machine image v1
# Build a machine image using Packer, which may be used for Azure Virtual
machine scale set deployment.
- task: PackerBuild@1
  inputs:
    templateType: 'builtin' # 'builtin' | 'custom'. Required. Packer
    template. Default: builtin.
    #customTemplateLocation: # string. Required when templateType = custom.
    Packer template location.
    #customTemplateParameters: '{}' # string. Optional. Use when
    templateType = custom. Template parameters. Default: {}.
    # Azure Details
    ConnectedServiceName: # string. Required when templateType = builtin.
    Azure subscription.
    #isManagedImage: true # boolean. Optional. Use when templateType =
    builtin. Managed VM disk image. Default: true.
    #managedImageName: # string. Required when isManagedImage = true &&
    templateType = builtin. Managed VM Disk Image Name.
    location: # string. Required when templateType = builtin. Storage
    location.
    storageAccountName: # string. Required when templateType = builtin.
    Storage account.
    azureResourceGroup: # string. Required when templateType = builtin.
    Resource group.
    # Deployment Inputs
    baseImageSource: 'default' # 'default' | 'customVhd'. Required when
    templateType = builtin. Base image source. Default: default.
    #baseImage: 'MicrosoftWindowsServer:WindowsServer:2012-R2-
    Datacenter:windows' # 'MicrosoftWindowsServer:WindowsServer:2012-R2-
    Datacenter:windows' | 'MicrosoftWindowsServer:WindowsServer:2016-
```

```

Datacenter:windows' | 'MicrosoftWindowsServer:WindowsServer:2012-
Datacenter:windows' | 'MicrosoftWindowsServer:WindowsServer:2008-R2-
SP1:windows' | 'Canonical:UbuntuServer:14.04.4-LTS:linux' |
'Canonical:UbuntuServer:16.04-LTS:linux' | 'Canonical:UbuntuServer:18.04-
LTS:linux' | 'RedHat:RHEL:7.2:linux' | 'RedHat:RHEL:6.8:linux' |
'OpenLogic:CentOS:7.2:linux' | 'OpenLogic:CentOS:6.8:linux' |
'credativ:Debian:8:linux' | 'credativ:Debian:7:linux' | 'SUSE:openSUSE-
Leap:42.2:linux' | 'SUSE:SLES:12-SP2:linux' | 'SUSE:SLES:11-SP4:linux'.
Required when baseImageSource = default && templateType = builtin. Base
image. Default: MicrosoftWindowsServer:WindowsServer:2012-R2-
Datacenter:windows.

#customImageUrl: # string. Required when baseImageSource = customVhd &&
templateType = builtin. Base image URL.

#customImageOSType: 'windows' # 'windows' | 'linux'. Required when
baseImageSource = customVhd && templateType = builtin. Base image OS.
Default: windows.

packagePath: # string. Required when templateType = builtin. Deployment
Package.

deployScriptPath: # string. Required when templateType = builtin.
Deployment script.

#deployScriptArguments: # string. Optional. Use when templateType =
builtin. Deployment script arguments.

# Advanced

#additionalBuilderParameters: '{"vm_size":"Standard_D3_v2"}' # string.
Optional. Use when templateType = builtin. Additional Builder parameters.
Default: {"vm_size":"Standard_D3_v2"}.

#skipTempFileCleanupDuringVMDeprovision: true # boolean. Optional. Use
when templateType = builtin. Skip temporary file cleanup during deprovision.
Default: true.

#packerVersion: # string. Optional. Use when templateType = custom.
Packer Version.

# Output

#imageUri: # string. Image URL or Name.
#imageId: # string. Azure Resource Id.

```

Inputs

`templateType` - Packer template

`string`. Required. Allowed values: `builtin` (Auto generated), `custom` (User provided).
 Default value: `builtin`.

Specifies whether the task auto generates a Packer template or uses a custom template provided by you.

`customTemplateLocation` - Packer template location

`string`. Required when `templateType = custom`.

Specifies the path to a custom user-provided template.

customTemplateParameters - Template parameters

`string`. Optional. Use when `templateType = custom`. Default value: `{}`.

Specifies the parameters which will be passed to the Packer for building a custom template. This should map to a `variables` section in your custom template. For example, if the template has a variable named `drop-location`, then add a parameter here with the name `drop-location` and a value which you want to use. You can link the value to a release variable as well. To view/edit the additional parameters in a grid, click on `...` next to text box.

ConnectedServiceName - Azure subscription

`string`. Required when `templateType = builtin`.

Specifies the Azure Resource Manager subscription for baking and storing the machine image.

isManagedImage - Managed VM disk image

`boolean`. Optional. Use when `templateType = builtin`. Default value: `true`.

Checks if the generated image should be a managed image.

managedImageName - Managed VM Disk Image Name

`string`. Required when `isManagedImage = true && templateType = builtin`.

Specifies the name of the Managed disk image for auto-generated templates.

location - Storage location

`string`. Required when `templateType = builtin`.

Specifies the location for storing the built machine image. This location will also be used to create a temporary VM for the purpose of building an image.

storageAccountName - Storage account

`string`. Required when `templateType = builtin`.

Specifies the storage account for storing the built machine image. This storage account must be pre-existing in the location selected.

`azureResourceGroup` - Resource group

`string`. Required when `templateType = builtin`.

Specifies the Azure Resource group that contains the selected storage account.

`baseImageSource` - Base image source

`string`. Required when `templateType = builtin`. Allowed values: `default` (Gallery), `customVhd` (Custom). Default value: `default`.

Specifies the source of the base image. You can either choose from a curated gallery of OS images or provide a URL of your custom VHD image.

Note

If you have selected the option to create a Managed image by checking the `Managed VM disk image` option, then you should only choose the `Gallery` option here. `Custom` source is not supported to create a managed image.

`baseImage` - Base image

`string`. Required when `baseImageSource = default && templateType = builtin`. Allowed values: `MicrosoftWindowsServer:WindowsServer:2012-R2-Datacenter:windows` (Windows 2012-R2-Datacenter), `MicrosoftWindowsServer:WindowsServer:2016-Datacenter:windows` (Windows 2016-Datacenter), `MicrosoftWindowsServer:WindowsServer:2012-Datacenter:windows` (Windows 2012-Datacenter), `MicrosoftWindowsServer:WindowsServer:2008-R2-SP1:windows` (Windows 2008-R2-SP1), `Canonical:UbuntuServer:14.04.4-LTS:linux` (Ubuntu 14.04.4-LTS), `Canonical:UbuntuServer:16.04-LTS:linux` (Ubuntu 16.04-LTS), `Canonical:UbuntuServer:18.04-LTS:linux` (Ubuntu 18.04-LTS), `RedHat:RHEL:7.2:linux` (RHEL 7.2), `RedHat:RHEL:6.8:linux` (RHEL 6.8), `OpenLogic:CentOS:7.2:linux` (CentOS 7.2), `OpenLogic:CentOS:6.8:linux` (CentOS 6.8), `credativ:Debian:8:linux` (Debian 8), `credativ:Debian:7:linux` (Debian 7), `SUSE:openSUSE-Leap:42.2:linux` (openSUSE-Leap 42.2), `SUSE:SLES:12-SP2:linux` (SLES 12-SP2), `SUSE:SLES:11-SP4:linux` (SLES 11-SP4). Default value: `MicrosoftWindowsServer:WindowsServer:2012-R2-Datacenter:windows`.

Chooses from a curated list of OS images. This will be used for installing pre-requisite(s) and application(s) before capturing a machine image.

`customImageUrl` - Base image URL

`string`. Required when `baseImageSource = customVhd && templateType = builtin`.

Specifies the URL of a base image. This will be used for installing pre-requisite(s) and application(s) before capturing a machine image.

`customImageOSType` - Base image OS

`string`. Required when `baseImageSource = customVhd && templateType = builtin`.

Allowed values: `windows`, `linux`. Default value: `windows`.

`packagePath` - Deployment Package

`string`. Required when `templateType = builtin`.

Specifies the path for a deployment package directory relative to `$(System.DefaultWorkingDirectory)`. Supports a minimatch pattern. Example path:
`FrontendWebApp/**/GalleryApp`

Note

This package will be copied to a temporary virtual machine which Packer creates. If the package contains a large number of files and/or the files are very large in size, the upload can take a long time (possibly running for a few hours). To optimize the upload time, please see if the size of the package can be meaningfully reduced. Another alternative is to use an intermediary Azure storage account. Upload the package to a storage account prior to running this task. For this task, use a package containing a script which will download the required package from the storage account.

`deployScriptPath` - Deployment script

`string`. Required when `templateType = builtin`.

Specifies the relative path to the powershell script (for Windows) or the shell script (for Linux) which deploys the package. This script should be contained in the package path selected above. Supports a minimatch pattern. Example path:

`deploy/**/scripts/windows/deploy.ps1`.

deployScriptArguments - Deployment script arguments

`string`. Optional. Use when `templateType = builtin`.

Specifies the arguments to be passed to the deployment script.

additionalBuilderParameters - Additional Builder parameters

`string`. Optional. Use when `templateType = builtin`. Default value:

`{"vm_size": "Standard_D3_v2"}`.

In the auto-generated Packer template mode, the task creates a Packer template with an Azure builder. This builder is used to generate a machine image. You can add keys to the Azure builder to customize the generated Packer template. For example: Setting `ssh_tty=true` if you are using a CentOS base image and you need to have a tty to run `sudo`. To view/edit the additional parameters in a grid, click on `...` next to text box.

skipTempFileCleanupDuringVMDeprovision - Skip temporary file cleanup during deprovision

`boolean`. Optional. Use when `templateType = builtin`. Default value: `true`.

During the deprovisioning of a VM, skips the cleanup of temporary files uploaded to the VM. Learn more about [Azure Virtual Machine Image Builders in Packer](#).

packerVersion - Packer Version

`string`. Optional. Use when `templateType = custom`.

Specifies the version of Packer to install. This will work only with custom templates.

imageUri - Image URL or Name

`string`.

Specifies a name for the output variable which will store the generated machine image VHD URL for an un-managed VM image or the image name for a managed VM image.

imageId - Azure Resource Id

`string`.

Specifies a name for the output variable which will store the Azure Resource ID for the newly created image. This is for managed images only.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to build a machine image using Packer. This image can be used for Azure Virtual machine scale set deployment.

ⓘ Note

If you want to enable detailed logs, navigate to **Pipelines > Edit > Variables**, and then add a new variable *PACKER_LOG* and set its value to 1.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.0.0 or greater
Task category	Deploy

PackerBuild@0 - Build machine image v0 task

Article • 09/26/2023

Use this task to build a machine image using Packer, which may be used for Azure Virtual machine scale set deployment.

ⓘ Note

This task does not support Azure Resource Manager authentication with workflow identity federation.

Syntax

YAML

```
# Build machine image v0
# Build a machine image using Packer, which may be used for Azure Virtual
machine scale set deployment.
- task: PackerBuild@0
  inputs:
    templateType: 'builtin' # 'builtin' | 'custom'. Required. Packer
template. Default: builtin.
    #customTemplateLocation: # string. Required when templateType = custom.
Packer template location.
    #customTemplateParameters: '{}' # string. Optional. Use when
templateType = custom. Template parameters. Default: {}.
  # Azure Details
    ConnectedServiceName: # string. Required when templateType = builtin.
Azure subscription.
    location: # string. Required when templateType = builtin. Storage
location.
    storageAccountName: # string. Required when templateType = builtin.
Storage account.
    azureResourceGroup: # string. Required when templateType = builtin.
Resource group.
  # Deployment Inputs
    baseImageSource: 'default' # 'default' | 'customVhd'. Required when
templateType = builtin. Base image source. Default: default.
    #baseImage: 'MicrosoftWindowsServer:WindowsServer:2012-R2-
Datacenter:windows' # 'MicrosoftWindowsServer:WindowsServer:2012-R2-
Datacenter:windows' | 'MicrosoftWindowsServer:WindowsServer:2016-
Datacenter:windows' | 'MicrosoftWindowsServer:WindowsServer:2012-
Datacenter:windows' | 'MicrosoftWindowsServer:WindowsServer:2008-R2-
SP1:windows' | 'Canonical:UbuntuServer:14.04.4-LTS:linux' |
'Canonical:UbuntuServer:16.04-LTS:linux' | 'RedHat:RHEL:7.2:linux' |
```

```

'RedHat:RHEL:6.8:linux' | 'OpenLogic:CentOS:7.2:linux' |
'OpenLogic:CentOS:6.8:linux' | 'credativ:Debian:8:linux' |
'credativ:Debian:7:linux' | 'SUSE:openSUSE-Leap:42.2:linux' | 'SUSE:SLES:12-
SP2:linux' | 'SUSE:SLES:11-SP4:linux'. Required when baseImageSource =
default && templateType = builtin. Base image. Default:
MicrosoftWindowsServer:WindowsServer:2012-R2-Datacenter:windows.

#customImageUrl: # string. Required when baseImageSource = customVhd &&
templateType = builtin. Base image URL.

#customImageOSType: 'windows' # 'windows' | 'linux'. Required when
baseImageSource = customVhd && templateType = builtin. Base image OS.
Default: windows.

packagePath: # string. Required when templateType = builtin. Deployment
Package.

deployScriptPath: # string. Required when templateType = builtin.
Deployment script.

#deployScriptArguments: # string. Optional. Use when templateType =
builtin. Deployment script arguments.

# Advanced

#additionalBuilderParameters: '{}' # string. Optional. Use when
templateType = builtin. Additional Builder parameters. Default: {}.

#skipTempFileCleanupDuringVMDeprovision: true # boolean. Optional. Use
when templateType = builtin. Skip temporary file cleanup during deprovision.
Default: true.

# Output

#imageUri: # string. Image URL.

```

Inputs

`templateType` - Packer template

`string`. Required. Allowed values: `builtin` (Auto generated), `custom` (User provided). Default value: `builtin`.

Specifies whether you want the task to auto generate a Packer template or use a custom template provided by you.

`customTemplateLocation` - Packer template location

`string`. Required when `templateType = custom`.

Specifies the path to a custom user-provided template.

`customTemplateParameters` - Template parameters

`string`. Optional. Use when `templateType = custom`. Default value: `{}`.

Specifies parameters which will be passed to Packer for building a custom template. This should map to the `variables` section in your custom template. For example, if the

template has a variable named `drop-location`, then add a parameter here with the name `drop-location` and a value which you want to use. You can link the value to a release variable as well. To view/edit the additional parameters in a grid, click on `...` next to the text box.

ConnectedServiceName - Azure subscription

`string`. Required when `templateType = builtin`.

Specifies the Azure Resource Manager subscription for baking and storing the machine image.

location - Storage location

`string`. Required when `templateType = builtin`.

Specifies the location for storing the built machine image. This location will also be used to create a temporary VM for the purpose of building an image.

storageAccountName - Storage account

`string`. Required when `templateType = builtin`.

Specifies the storage account for storing the built machine image. This storage account must be pre-existing in the location selected.

azureResourceGroup - Resource group

`string`. Required when `templateType = builtin`.

Specifies the Azure Resource group that contains the selected storage account.

baseImageSource - Base image source

`string`. Required when `templateType = builtin`. Allowed values: `default` (Gallery), `customVhd` (Custom). Default value: `default`.

Specifies the source of a base image. You can either choose from a curated gallery of OS images or provide a URL of your custom image.

baseImage - Base image

`string`. Required when `baseImageSource = default && templateType = builtin`. Allowed

values: MicrosoftWindowsServer:WindowsServer:2012-R2-Datacenter:windows (Windows 2012-R2-Datacenter), MicrosoftWindowsServer:WindowsServer:2016-Datacenter:windows (Windows 2016-Datacenter), MicrosoftWindowsServer:WindowsServer:2012-Datacenter:windows (Windows 2012-Datacenter), MicrosoftWindowsServer:WindowsServer:2008-R2-SP1:windows (Windows 2008-R2-SP1), Canonical:UbuntuServer:14.04.4-LTS:linux (Ubuntu 14.04.4-LTS), Canonical:UbuntuServer:16.04-LTS:linux (Ubuntu 16.04-LTS), RedHat:RHEL:7.2:linux (RHEL 7.2), RedHat:RHEL:6.8:linux (RHEL 6.8), OpenLogic:CentOS:7.2:linux (CentOS 7.2), OpenLogic:CentOS:6.8:linux (CentOS 6.8), credativ:Debian:8:linux (Debian 8), credativ:Debian:7:linux (Debian 7), SUSE:openSUSE-Leap:42.2:linux (openSUSE-Leap 42.2), SUSE:SLES:12-SP2:linux (SLES 12-SP2), SUSE:SLES:11-SP4:linux (SLES 11-SP4). Default value: MicrosoftWindowsServer:WindowsServer:2012-R2-Datacenter:windows.

Chooses from a curated list of OS images. This is used for installing pre-requisite(s) and application(s) before capturing a machine image.

customImageUrl - Base image URL

string. Required when baseImageSource = customVhd && templateType = builtin.

Specifies the URL of a base image. This is used for installing pre-requisite(s) and application(s) before capturing a machine image.

customImageOSType - Base image OS

string. Required when baseImageSource = customVhd && templateType = builtin.

Allowed values: windows, linux. Default value: windows.

packagePath - Deployment Package

string. Required when templateType = builtin.

Specifies the path for the deployment package directory relative to \$(System.DefaultWorkingDirectory). Supports a minimatch pattern. Example path: FrontendWebApp/**/GalleryApp.

deployScriptPath - Deployment script

string. Required when templateType = builtin.

Specifies the relative path to a powershell script (for Windows) or a shell script (for Linux) which deploys the package. This script should be contained in the package path

selected above. Supports a minimatch pattern. Example path:

```
deploy/**/scripts/windows/deploy.ps1.
```

deployScriptArguments - Deployment script arguments

`string`. Optional. Use when `templateType = builtin`.

Specifies the arguments to be passed to the deployment script.

additionalBuilderParameters - Additional Builder parameters

`string`. Optional. Use when `templateType = builtin`. Default value: `{}`.

In an auto-generated Packer template mode, the task creates a Packer template with an Azure builder. This builder is used to generate a machine image. You can add keys to the Azure builder to customize the generated Packer template. For example, setting `ssh_tty=true` in case you are using a CentOS base image, and you need to have a tty to run sudo.

To view or edit the additional parameters in a grid, click on `...` next to text box.

skipTempFileCleanupDuringVMDeprovision - Skip temporary file cleanup during deprovision

`boolean`. Optional. Use when `templateType = builtin`. Default value: `true`.

During the deprovisioning of a VM, skips the cleanup of temporary files uploaded to the VM. For more information, refer to [Azure Virtual Machine Image Builders](#).

imageUri - Image URL

`string`.

Specifies a name for the output variable which stores the generated machine image URL.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to build a machine image using Packer. This image can be used for Azure Virtual machine scale set deployment.

 Note

If you want to enable detailed logs, navigate to **Pipelines > Edit > Variables**, and then add a new variable *PACKER_LOG* and set its value to 1.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.0.0 or greater
Task category	Deploy

Cache@2 - Cache v2 task

Article • 09/26/2023

Improve build performance by using this task to cache files, like dependencies, between pipeline runs. See [Cache task: how it works](#) and [Reduce build time using caching](#) for specific examples and more details.

Syntax

YAML

```
# Cache v2
# Cache files between runs.
- task: Cache@2
  inputs:
    key: # string. Required. Key.
    path: # string. Required. Path.
    #cacheHitVar: # string. Cache hit variable.
    #restoreKeys: # string. Additional restore key prefixes.
```

Inputs

key - Key

`string`. Required.

The key (unique identifier) for the cache. This should be a string that can be segmented using `|`. File paths can be absolute or relative to `$(System.DefaultWorkingDirectory)`.

While there is no defined maximum number of segments for a key, if you are getting cache misses, try using a shorter key with fewer segments, for example by creating a new key that is a hash of your segments.

path - Path

`string`. Required.

The path of the folder to cache. Can be fully qualified or relative to `$(System.DefaultWorkingDirectory)`. Wildcards are not supported. [Variables](#) are supported.

`cacheHitVar` - Cache hit variable

`string`.

The variable to set to `true` when the cache is restored (i.e. a cache hit). Otherwise, sets the variable to `false`.

`restoreKeys` - Additional restore key prefixes

`string`.

The additional restore key prefixes that the task uses if the primary key misses. This can be a newline-delimited list of key prefixes.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Improve build performance by caching files, like dependencies, between pipeline runs.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any

Requirement	Description
Agent version	2.160.0 or greater
Task category	Utility

See also

- [Pipeline Caching](#)
- [Cache NuGet packages](#)

CacheBeta@1 - Cache (Beta) v1 task

Article • 09/26/2023

Improve build performance by using this task to cache files, like dependencies, between pipeline runs.

Syntax

YAML

```
# Cache (Beta) v1
# Cache files between runs.
- task: CacheBeta@1
  inputs:
    key: # string. Required. Key.
    path: # string. Required. Path.
    #cacheHitVar: # string. Cache hit variable.
    #restoreKeys: # string. Additional restore key prefixes.
```

Inputs

key - Key

`string`. Required.

The key (unique identifier) for the cache. This should be a string that can be segmented using `|`. File paths can be absolute or relative to `$(System.DefaultWorkingDirectory)`.

path - Path

`string`. Required.

The path of the folder to cache. Can be fully qualified or relative to

`$(System.DefaultWorkingDirectory)`. Wildcards are not supported. [Variables](#) are supported.

cacheHitVar - Cache hit variable

`string`.

The variable to set to `true` when the cache is restored (i.e. a cache hit). Otherwise, sets the variable to `false`.

`restoreKeys` - Additional restore key prefixes

`string`.

The additional restore key prefixes that the task uses if the primary key misses. This can be a newline-delimited list of key prefixes.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.159.2 or greater
Task category	Utility

CacheBeta@0 - Cache (Beta) v0 task

Article • 09/26/2023

Improve build performance by using this task to cache files, like dependencies, between pipeline runs.

Syntax

YAML

```
# Cache (Beta) v0
# Cache files between runs.
- task: CacheBeta@0
  inputs:
    key: # string. Required. Key.
    path: # string. Required. Path.
    #cacheHitVar: # string. Cache hit variable.
```

Inputs

key - Key

`string`. Required.

The key (unique identifier) for the cache. This should be a newline-delimited list of strings or file paths. File paths can be absolute or relative to `$(System.DefaultWorkingDirectory)`.

path - Path

`string`. Required.

The path of the folder to cache. Can be fully qualified or relative to `$(System.DefaultWorkingDirectory)`. Wildcards are not supported. [Variables](#) are supported.

cacheHitVar - Cache hit variable

`string`.

The variable to set to `true` when the cache is restored (i.e. a cache hit). Otherwise, sets the variable to `false`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.159.2 or greater
Task category	Utility

CargoAuthenticate@0 - Cargo authenticate (for task runners) v0 task

Article • 09/26/2023

Authentication task for the cargo client used for installing Cargo crates.

ⓘ Note

Cargo support in Azure Artifacts is currently in preview.

Syntax

YAML

```
# Cargo authenticate (for task runners) v0
# Authentication task for the cargo client used for installing Cargo crates
distribution.
- task: CargoAuthenticate@0
  inputs:
    configFile: # string. Required. config.toml file to authenticate.
    cargoServiceConnections: # string. Credentials for registries outside
this organization/collection.
```

Inputs

`configFile` - config.toml file to authenticate

`string`. Required.

Path to the config.toml file that specifies the registries you want to work with. Select the file, not the folder e.g. "./cargo/config.toml".

`cargoServiceConnections` - Credentials for registries outside this organization/collection

`string`.

Credentials to use for external registries located in the project's config.toml. For registries in this organization/collection, don't specify this attribute; the build's credentials are used automatically.

Use the [task assistant](#) to select the desired service connections.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Package

AzurePolicyCheckGate@0 - Check Azure Policy compliance v0 task

Article • 09/26/2023

Use this task to check the security and compliance assessment for Azure Policy.

Syntax

YAML

```
# Check Azure Policy compliance v0
# Security and compliance assessment for Azure Policy.
- task: AzurePolicyCheckGate@0
  inputs:
    azureSubscription: # string. Alias: ConnectedServiceName. Required.
    Azure subscription.
    #ResourceGroupName: # string. Resource group.
    #Resources: # string. Resource name.
    # Advanced
    #RetryDuration: '00:02:00' # string. Retry duration. Default: 00:02:00.
```

Inputs

azureSubscription - Azure subscription

Input alias: `ConnectedServiceName`. `string`. Required.

Selects the Azure Resource Manager subscription you want to use to enforce the policies.

ResourceGroupName - Resource group

`string`.

Provides the resource group name.

Resources - Resource name

`string`.

Selects the name of Azure resources for which you want to check the policy compliance.

`RetryDuration` - Retry duration

`string`. Default value: `00:02:00`.

The Check Azure Policy compliance task performs an asynchronous [On-demand evaluation scan](#) of your [compliance data of Azure resources](#). The call returns a [202 Accepted](#) status while the evaluation is ongoing. The `RetryDuration` input configures the intervals in which the task retries the REST API call to check for the completion of the policy evaluation. The format is `hours:minutes:seconds` in the following format: `hh:mm:ss`.

The default is `00:02:00` (two minutes), which is the minimum interval that may be configured.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Azure Policy allows you to assess and enforce resource compliance against defined IT policies. Use this task in a gate to identify, analyze and evaluate the security risks, and determine the mitigation measures required to reduce the risks.

Note

Can be used only as a [gate](#). This task is not supported in a build or release pipeline.

The screenshot shows the 'Pre-deployment app' configuration in the Azure DevOps pipeline editor. The 'Check Azure Policy compliance' task is highlighted with a red box. Other tasks listed include 'Invoke Azure Function', 'Invoke REST API', 'Query Azure Monitor alerts', and 'Query work items'. A 'Gates' section is also visible.

Pre-deployment app

Select the users who can approve this deployment

Gates

The delay before evaluation

Deployment gates

Add

Requirements

Requirement	Description
Pipeline types	Classic release
Runs on	ServerGate
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Deploy

Chef@1 - Chef v1 task

Article • 09/26/2023

Deploy to Chef environments by editing environment attributes.

This task is deprecated.

Syntax

YAML

```
# Chef v1
# Deploy to Chef environments by editing environment attributes.
- task: Chef@1
  inputs:
    connectedServiceName: # string. Required. Chef Service Connection.
    Environment: # string. Required. Environment.
    Attributes: # string. Required. Environment Attributes.
    chefWaitTime: '30' # string. Required. Wait Time. Default: 30.
```

Inputs

connectedServiceName - Chef Service Connection

`string`. Required.

The name of the Chef subscription service connection.

Environment - Environment

`string`. Required.

The name of the Chef environment to be used for deployment. The attributes of that environment will be edited.

Attributes - Environment Attributes

`string`. Required.

Specifies the value of the leaf node attribute(s) to be updated. Example: {

```
"default_attributes.connectionString" : "$(connectionString)",
"override_attributes.buildLocation" : "https://sample.blob.core.windows.net/build"
}
```

The task fails if the leaf node does not exist.

chefWaitTime - Wait Time

`string`. Required. Default value: `30`.

The amount of time (in minutes) to wait for this task to complete. Default value: 30 minutes.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to deploy to Chef environments by editing environment attributes.

ⓘ Note

This task is deprecated.

Requirements

Requirement	Description
Pipeline types	YAML, Preview, Classic build, Classic release
Runs on	Agent
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: Chef, KnifeReporting, DotNetFramework
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any

Requirement	Description
Settable variables	Any
Agent version	1.83.0 or greater
Task category	Deploy

ChefKnife@1 - Chef Knife v1 task

Article • 09/26/2023

Run scripts with Knife commands on your Chef workstation.

This task is deprecated.

Syntax

YAML

```
# Chef Knife v1
# Run scripts with Knife commands on your Chef workstation.
- task: ChefKnife@1
  inputs:
    ConnectedServiceName: # string. Required. Chef Subscription.
    ScriptPath: # string. Required. Script Path.
    #ScriptArguments: # string. Script Arguments.
```

Inputs

ConnectedServiceName - Chef Subscription

`string`. Required.

The Chef subscription to configure before running knife commands.

ScriptPath - Script Path

`string`. Required.

The path of the script. This should be fully qualified path or a relative to the default working directory.

ScriptArguments - Script Arguments

`string`.

Optional. The additional parameters to pass to the script. Can be either ordinal or named parameters.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to run scripts with Knife commands on your Chef workstation.

Note

This task is deprecated.

Requirements

Requirement	Description
Pipeline types	YAML, Preview, Classic build
Runs on	Agent
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: Chef, DotNetFramework
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.83.0 or greater
Task category	Deploy

CMake@1 - CMake v1 task

Article • 09/26/2023

Use this task to build with the CMake cross-platform build system.

Syntax

YAML

```
# CMake v1
# Build with the CMake cross-platform build system.
- task: CMake@1
  inputs:
    #workingDirectory: 'build' # string. Alias: cwd. Working Directory.
  Default: build.
    #cmakeArgs: # string. Arguments.
    # Advanced
    #runInsideShell: false # boolean. Run cmake command inside shell.
  Default: false.
```

Inputs

`workingDirectory` - Working Directory

Input alias: `cwd`. `string`. Default value: `build`.

The current working directory when CMake is run.

If you specify a relative path, then it is relative to your repo. For example, if you specify `build`, the result is the same as if you specified `$(Build.SourcesDirectory)\build`.

You can also specify a full path outside the repo, and you can use [variables](#). For example:
`$(Build.ArtifactStagingDirectory)\build`

If the path you specify does not exist, CMake creates it.

`cmakeArgs` - Arguments

`string`.

The arguments passed to CMake.

`runInsideShell` - Run cmake command inside shell

boolean. Default value: `false`.

CMake arguments are handled the same way as they are handled inside of an OS specific shell. This input can be used to handle environment variables inside of argument strings.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to build with the CMake cross-platform build system.

How do I enable CMake for Microsoft-hosted agents?

The [Microsoft-hosted agents](#) have CMake installed already, so you don't need to do anything. You do not need to add a demand for CMake in your `azure-pipelines.yml` file.

How do I enable CMake for my on-premises agent?

1. [Deploy an agent](#).
2. On your agent machine, [install CMake](#) and make sure to add it to the user's path that the agent is running as.
3. In your web browser, go to [Agent pools](#) and [add a capability](#) named `cmake`. Set its value to `yes`.

How does CMake work? What arguments can I use?

- [About CMake](#)
- [CMake documentation](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: cmake
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.91.0 or greater
Task category	Build

CocoaPods@0 - CocoaPods v0 task

Article • 09/26/2023

Use this task to run [CocoaPods pod install](#).

[CocoaPods](#) is the dependency manager for Swift and Objective-C Cocoa projects. This task optionally runs `pod repo update` and then runs `pod install`.

Syntax

YAML

```
# CocoaPods v0
# Install CocoaPods dependencies for Swift and Objective-C Cocoa projects.
- task: CocoaPods@0
  inputs:
    #workingDirectory: # string. Alias: cwd. Working directory.
  # Advanced
    #forceRepoUpdate: false # boolean. Force repo update. Default: false.
    #projectDirectory: # string. Project directory.
```

Inputs

`workingDirectory` - Working directory

Input alias: `cwd`. `string`.

Specifies the working directory in which to execute this task. If left empty, the repository directory will be used.

`forceRepoUpdate` - Force repo update

`boolean`. Default value: `false`.

Selecting this option will force running `pod repo update` before installation.

`projectDirectory` - Project directory

`string`.

Optional. Specifies the path to the root of the project directory. If left empty, the task uses the specified project in the podfile. If no project is specified, then the task searches for an Xcode project. If the task finds more than one Xcode project, an error will occur.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to run CocoaPods [pod install](#).

CocoaPods is the dependency manager for Swift and Objective-C Cocoa projects. This task optionally runs `pod repo update` and then runs `pod install`.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : Setting variables is disabled
Agent version	2.182.1 or greater
Task category	Package

CmdLine@2 - Command line v2 task

Article • 09/26/2023

Use this task to run a command line script using Bash on Linux, macOS, and cmd.exe on Windows.

Syntax

YAML

```
# Command Line v2
# Run a command line script using Bash on Linux and macOS and cmd.exe on
# Windows.
- task: CmdLine@2
  inputs:
    script: # string. Required. Script.
    # Advanced
    #workingDirectory: # string. Working Directory.
    #failOnStderr: false # boolean. Fail on Standard Error. Default: false.
```

Inputs

script - Script

`string`. Required. Default value: `echo Write your commands here\n\necho Hello world.`

The contents of the script you want to run.

workingDirectory - Working Directory

`string`.

Specifies the working directory to run commands. If you leave it empty, the working directory is [\\$\(Build.SourcesDirectory\)](#).

failOnStderr - Fail on Standard Error

`boolean`. Default value: `false`.

If the value is set to true, the task fails if any errors are written to the Standard Error stream.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

The command line has a shortcut in YAML: `steps.script`.

```
yml
```

```
- script: # inline script
  workingDirectory: #
  displayName: #
  failOnStderr: #
  env: { string: string } # mapping of environment variables to add
```

Running batch and .CMD files

Azure Pipelines puts your inline script contents into a temporary batch file (.cmd) in order to run it. When you want to run a batch file from another batch file in Windows CMD, you must use the `call` command, otherwise the first batch file is terminated. This will result in Azure Pipelines running your intended script up until the first batch file, then running the batch file, then ending the step. Additional lines in the first script aren't run. You should always prepend `call` before executing a batch file in an Azure Pipelines script step.

Important

You may not realize you're running a batch file. For example, `npm` on Windows, along with any tools that you install using `npm install -g`, are actually batch files. Always use `call npm <command>` to run NPM commands in a Command Line task on Windows.

Examples

YAML

```
steps:
- script: date /t
  displayName: Get the date
- script: dir
  workingDirectory: $(Agent.BuildDirectory)
  displayName: List contents of a folder
- script: |
  set MYVAR=foo
  set
  displayName: Set a variable and then display all
env:
  aVarFromYaml: someValue
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Utility

See also

- Learn how to use [verbose logs for troubleshooting](#).

CmdLine@1 - Command Line v1 task

Article • 09/26/2023

Use this task to run a program from the command prompt.

Syntax

YAML

```
# Command Line v1
# Run a command line with arguments.
- task: CmdLine@1
  inputs:
    filename: # string. Required. Tool.
    #arguments: # string. Arguments.
    # Advanced
    #workingFolder: # string. Working folder.
    #failOnStandardError: false # boolean. Fail on Standard Error. Default:
false.
```

Inputs

filename - Tool

`string`. Required.

The name of the tool to run. The tool should be found in your path. Optionally, a fully qualified path can be supplied, but the path must be present on the agent.

Note: You can use `$(Build.SourcesDirectory)\` if you want the path relative to repo.

arguments - Arguments

`string`.

The arguments that are passed to the tool. Use double quotes to escape spaces.

workingFolder - Working folder

`string`.

Specifies the working directory to run commands. If you leave it empty, the working directory is `$(Build.SourcesDirectory)`.

`failOnStandardError` - Fail on Standard Error

`boolean`. Default value: `false`.

If the value is set to true, the task fails if any errors are written to the Standard Error stream.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

There is a newer version of the Command line task at [CommandLine@2](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Utility

See also

- [CommandLine@2](#)

CondaEnvironment@1 - Conda environment v1 task

Article • 09/26/2023

Use this task to create and activate a Conda environment.

ⓘ Note

This task has been deprecated. Use `conda` directly in the bash task or batch script task as an alternative.

Syntax

YAML

```
# Conda environment v1
# This task is deprecated. Use `conda` directly in script to work with
Anaconda environments.
- task: CondaEnvironment@1
  inputs:
    #createCustomEnvironment: false # boolean. Create a custom environment.
    Default: false.
    #environmentName: # string. Required when createCustomEnvironment ==
true. Environment name.
    #packageSpecs: 'python=3' # string. Package specs. Default: python=3.
    #updateConda: true # boolean. Update to the latest Conda. Default: true.
    #installOptions: # string. Optional. Use when createCustomEnvironment ==
false. Other options for `conda install`.
    #createOptions: # string. Optional. Use when createCustomEnvironment ==
true. Other options for `conda create`.
    #cleanEnvironment: false # boolean. Optional. Use when
createCustomEnvironment == true. Clean the environment. Default: false.
```

Inputs

`createCustomEnvironment` - Create a custom environment

`boolean`. Default value: `false`.

If the value for this boolean is set to `true`, the task creates ↗ or reactivates a Conda environment instead of using the `base` environment. Setting the value to `true` is recommended for self-hosted agents.

environmentName - Environment name

`string`. Required when `createCustomEnvironment == true`.

The name of the Conda environment to create and activate, or reactivate if it already exists.

packageSpecs - Package specs

`string`. Default value: `python=3`.

The space-delimited list of packages to install in the environment.

updateConda - Update to the latest Conda

`boolean`. Default value: `true`.

Updates Conda to the latest version. This applies to the Conda installation found in `PATH` or to the path specified by the `CONDA` environment variable.

installOptions - Other options for `conda install`

`string`. Optional. Use when `createCustomEnvironment == false`.

The space-delimited list of additional arguments to pass to the `conda install` command.

createOptions - Other options for `conda create`

`string`. Optional. Use when `createCustomEnvironment == true`.

The space-delimited list of additional options to pass to the `conda create` command.

cleanEnvironment - Clean the environment

`boolean`. Optional. Use when `createCustomEnvironment == true`. Default value: `false`.

Deletes the environment and recreates it if it already exists. If this boolean is not selected, the task will reactivate an existing environment.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to create and activate a Conda environment.

ⓘ Note

This task has been deprecated. Use `conda` directly in the **bash task or batch script task** as an alternative.

This task will create a Conda environment and activate it for subsequent build tasks.

If the task finds an existing environment with the same name, the task will simply reactivate it. This is possible on self-hosted agents. To recreate the environment and reinstall any of its packages, set the "Clean the environment" option.

Running with the "Update to the latest Conda" option will attempt to update Conda before creating or activating the environment. If you are running a self-hosted agent and have [configured a Conda installation to work with the task](#), this may result in your Conda installation being updated.

ⓘ Note

Microsoft-hosted agents won't have Conda in their `PATH` by default. You will need to run this task in order to use Conda.

After running this task, `PATH` will contain the binary directory for the activated environment, followed by the binary directories for the Conda installation itself. You can run scripts as subsequent build tasks that run Python, Conda, or the command-line utilities from other packages you install. For example, you can run tests with [pytest](#) or upload a package to Anaconda Cloud with the [Anaconda client](#).

ⓘ Tip

After running this task, the environment will be "activated," and packages you install by calling `conda install` will get installed to this environment.

Prerequisites

- A Microsoft-hosted agent, or a self-hosted agent with Anaconda or Miniconda installed.
- If using a self-hosted agent, you must either add the `conda` executable to `PATH` or set the `CONDA` environment variable to the root of the Conda installation.

How can I configure a self-hosted agent to use this task?

You can use this task either with a full Anaconda installation or a Miniconda installation. If using a self-hosted agent, you must add the `conda` executable to `PATH`. Alternatively, you can set the `CONDA` environment variable to the root of the Conda installation -- that is, the directory you specify as the "prefix" when installing Conda.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Package

CondaEnvironment@0 - Conda environment v0 task

Article • 09/26/2023

Use this task to create and activate a Conda environment.

ⓘ Note

This task has been deprecated. Use `conda` directly in the **bash task or batch script task** as an alternative.

Syntax

YAML

```
# Conda environment v0
# Create and activate a Conda environment.
- task: CondaEnvironment@0
  inputs:
    environmentName: # string. Required. Environment name.
    packageSpecs: 'python=3' # string. Package specs. Default: python=3.
    updateConda: true # boolean. Update to the latest Conda. Default: true.
    # Advanced
    #createOptions: # string. Environment creation options.
    #cleanEnvironment: false # boolean. Clean the environment. Default:
    false.
```

Inputs

`environmentName` - Environment name

`string`. Required.

The name of the Conda environment to create and activate, or reactivate if it already exists.

`packageSpecs` - Package specs

`string`. Default value: `python=3`.

The space-delimited list of packages to install in the environment.

`updateConda` - Update to the latest Conda

`boolean`. Default value: `true`.

Updates Conda to the latest version. This applies to the Conda installation found in `PATH` or to the path specified by the `CONDA` environment variable.

`createOptions` - Environment creation options

`string`.

The space-delimited list of additional options to pass to the `conda create` command.

`cleanEnvironment` - Clean the environment

`boolean`. Default value: `false`.

Deletes the environment and recreates it if it already exists. If this boolean is not selected, the task will reactivate an existing environment.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to create and activate a Conda environment.

Note

This task has been deprecated. Use `conda` directly in the `bash task` or `batch script task` as an alternative.

This task will create a Conda environment and activate it for subsequent build tasks.

If the task finds an existing environment with the same name, the task will simply reactivate it. This is possible on self-hosted agents. To recreate the environment and reinstall any of its packages, set the "Clean the environment" option.

Running with the "Update to the latest Conda" option will attempt to update Conda before creating or activating the environment. If you are running a self-hosted agent and have [configured a Conda installation to work with the task](#), this may result in your Conda installation being updated.

Note

Microsoft-hosted agents won't have Conda in their `PATH` by default. You will need to run this task in order to use Conda.

After running this task, `PATH` will contain the binary directory for the activated environment, followed by the binary directories for the Conda installation itself. You can run scripts as subsequent build tasks that run Python, Conda, or the command-line utilities from other packages you install. For example, you can run tests with [pytest](#) or upload a package to Anaconda Cloud with the [Anaconda client](#).

Tip

After running this task, the environment will be "activated", and packages you install by calling `conda install` will get installed to this environment.

Prerequisites

- A Microsoft-hosted agent, or a self-hosted agent with Anaconda or Miniconda installed.
- If using a self-hosted agent, you must either add the `conda` executable to `PATH` or set the `CONDA` environment variable to the root of the Conda installation.

How can I configure a self-hosted agent to use this task?

You can use this task either with a full Anaconda installation or a Miniconda installation. If using a self-hosted agent, you must add the `conda` executable to `PATH`. Alternatively, you can set the `CONDA` environment variable to the root of the Conda installation -- that is, the directory you specify as the "prefix" when installing Conda.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Package

ContainerBuild@0 - Container Build v0 task

Article • 09/26/2023

Container Build Task.

Syntax

YAML

```
# Container Build v0
# Container Build Task.
- task: ContainerBuild@0
  inputs:
    dockerRegistryServiceConnection: # string. Docker registry service connection.
    repository: # string. Container repository.
    Dockerfile: 'Dockerfile' # string. Required. Dockerfile. Default: Dockerfile.
    buildContext: '.' # string. Build context. Default: ..
    tags: '$(Build.BuildId)' # string. Tags. Default: $(Build.BuildId).
```

Inputs

`dockerRegistryServiceConnection` - Docker registry service connection

`string`.

Specifies a Docker registry service connection.

`repository` - Container repository

`string`.

The name of the repository within the container registry.

`Dockerfile` - Dockerfile

`string`. Required. Default value: `Dockerfile`.

The path to the Dockerfile.

`buildContext` - Build context

`string`. Default value: `..`

The path to the build context.

`tags` - Tags

`string`. Default value: `$(Build.BuildId)`.

The list of tags in separate lines. Tags are used while building and pushing the image to container registry.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: Buildctl
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Build

ContainerStructureTest@0 - Container Structure Test v0 task

Article • 09/26/2023

Uses container-structure-test (<https://github.com/GoogleContainerTools/container-structure-test>) to validate the structure of an image based on four categories of tests - command tests, file existence tests, file content tests and metadata tests.

Syntax

YAML

```
# Container Structure Test v0
# Uses container-structure-test
#(https://github.com/GoogleContainerTools/container-structure-test) to
# validate the structure of an image based on four categories of tests -
# command tests, file existence tests, file content tests and metadata tests.
- task: ContainerStructureTest@0
  inputs:
    # Container Repository
    dockerRegistryServiceConnection: # string. Required. Docker registry
    service connection.
      repository: # string. Required. Container repository.
      #tag: '$(Build.BuildId)' # string. Tag. Default: $(Build.BuildId).
      configFile: # string. Required. Config file path.
      #testRunTitle: # string. Test run title.
      #failTaskOnFailedTests: false # boolean. Fail task if there are test
      failures. Default: false.
```

Inputs

`dockerRegistryServiceConnection` - Docker registry service connection

`string`. Required.

Specify a Docker registry service connection. Required for commands that need to authenticate with a registry.

`repository` - Container repository

`string`. Required.

The name of the repository.

`tag` - Tag

`string`. Default value: `$(Build.BuildId)`.

The tag is used in pulling the image from docker registry service connection.

`configFile` - Config file path

`string`. Required.

The config file path that contains container structure tests, either in .yaml or .json file formats.

`testRunTitle` - Test run title

`string`.

Specify a name for the Test Run.

`failTaskOnFailedTests` - Fail task if there are test failures

`boolean`. Default value: `false`.

Fails the task if there are any test failures. Check this option to fail the task if test failures are detected.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

This task helps you run container structure tests and publish test results to Azure Pipelines and provides a comprehensive test reporting and analytics experience.

Note

This is an early preview feature. More upcoming features will be rolled out in upcoming sprints.

The Container Structure Tests provide a powerful framework to validate the structure of a container image. These tests can be used to check the output of commands in an image, as well as verify metadata and contents of the filesystem. Tests can be run either through a standalone binary, or through a Docker image.

Tests within this framework are specified through a YAML or JSON config file. Multiple config files may be specified in a single test run. The config file will be loaded in by the test runner, which will execute the tests in order. Within this config file, four types of tests can be written:

- Command Tests (testing output/error of a specific command issued)
- File Existence Tests (making sure a file is, or isn't, present in the file system of the image)
- File Content Tests (making sure files in the file system of the image contain, or do not contain, specific contents)
- Metadata Test, singular (making sure certain container metadata is correct)

Build, Test and Publish Test

The container structure test task can be added in the classic pipeline as well as in unified pipeline (multi-stage) & YAML based pipelines.

YAML

In the new YAML based unified pipeline, you can search for task in the window.

```

48 - task: PublishTestResults@2
  displayName: 'Publish Test Results **/ResultsFileSamples/VsTsResults/*.trx'
  inputs:
    testResultsFormat: 'VSTest'
    testResultsFiles: '**/ResultsFileSamples/VsTsResults/*.trx'
    continueOnError: true
    enabled: true
  ...
  Settings
  - task: PublishBuildArtifacts@1
    displayName: 'Publish Artifact: drop'
    inputs:
      PathtoPublish: '$(build.artifactstagingdirectory)'
      condition: succeededOrFailed()

```

Once the task is added, you need to set the config file path, docker registry service connection, container repository and tag, if required. Task input in the yaml based pipeline is created.

```

48 - task: PublishTestResults@2
  displayName: 'Publish Test Results **/ResultsFileSamples/VsTsResults/*.trx'
  inputs:
    testResultsFormat: 'VSTest'
    testResultsFiles: '**/ResultsFileSamples/VsTsResults/*.trx'
    continueOnError: true
    enabled: true
  ...
  Settings
  - task: PublishBuildArtifacts@1
    displayName: 'Publish Artifact: drop'
    inputs:
      PathtoPublish: '$(build.artifactstagingdirectory)'
      condition: succeededOrFailed()

```

YAML file

```

❖ HalaRM / azure-pipelines.yml *

41   - platform: '$(BuildPlatform)'
42   - configuration: '$(BuildConfiguration)'
43   - continueOnError: true
44   - enabled: false
45
46   Settings
47   - task: PublishTestResults@2
48   - displayName: 'Publish Test Results **/ResultsFileSamples/VsTsResults/*.trx'
49   - inputs:
50     - testResultsFormat: VSTest
51     - testResultsFiles: '**/ResultsFileSamples/VsTsResults/*.trx'
52     - continueOnError: true
53     - enabled: true
54
55   Settings
56   - task: PublishBuildArtifacts@1
57   - displayName: 'Publish Artifact: drop'
58   - inputs:
59     - PathToPublish: '$(build.artifactstagingdirectory)'
60     - condition: succeededOrFailed()
61
62   Settings
63   - task: ContainerStructureTest@0
64   - inputs:
65     - dockerRegistryServiceConnection: 'navin_dockerHub'
66     - repository: 'navb/helldocker'
67     - tag: 'v1'
68     - configFile: '/home/navin/cstfiles/fileexisttest.yaml'
69     - failTaskOnFailedTests: true
70

```

yml

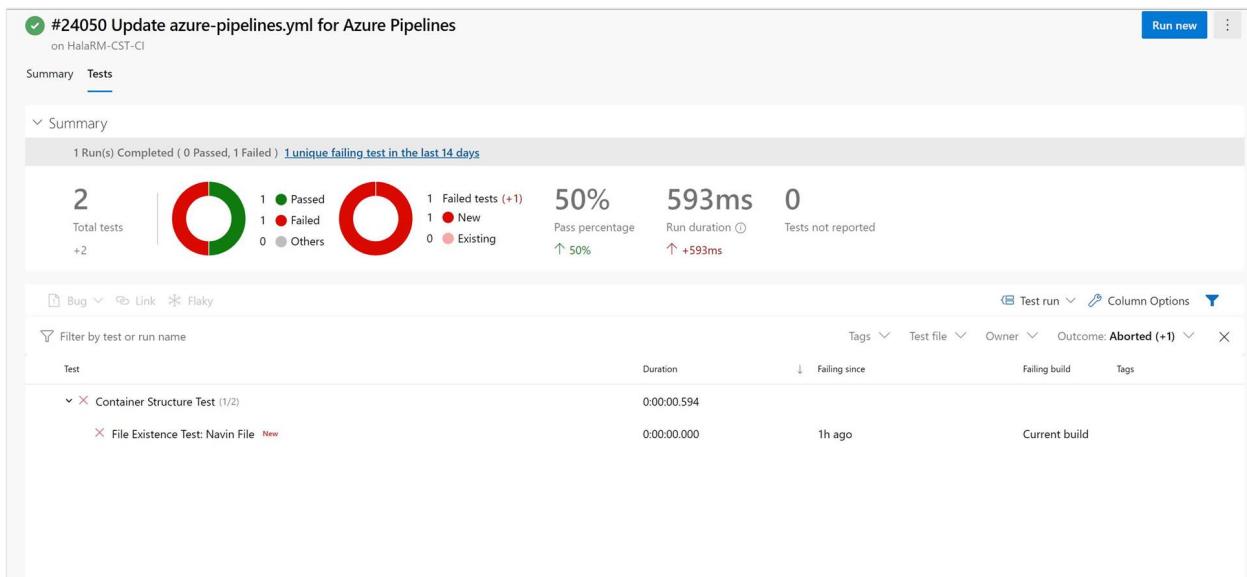
```

steps:
- task: ContainerStructureTest@0
  displayName: 'Container Structure Test '
  inputs:
    dockerRegistryServiceConnection: 'Container_dockerHub'
    repository: adma/helldocker
    tag: v1
    configFile: /home/user/cstfiles/fileexisttest.yaml

```

View test report

Once the task is executed, you can directly go to test tab to view the full report. The published test results are displayed in the [Tests tab](#) in the pipeline summary and help you to measure pipeline quality, review traceability, troubleshoot failures, and drive failure ownership.



Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Test

CopyPublishBuildArtifacts@1 - Copy and Publish Build Artifacts v1 task

Article • 09/26/2023

Use this task to copy build artifacts to a staging folder and then publish them to the server or a file share. Files are copied to the `$(Build.ArtifactStagingDirectory)` staging folder and then published.

ⓘ Important

This task is deprecated. We recommend that you use [Pipeline Artifacts](#) and the [Copy Files task](#) and the [Publish Build Artifacts](#) task instead.

Syntax

YAML

```
# Copy and Publish Build Artifacts v1
# CopyPublishBuildArtifacts@1 is deprecated. Use the Copy Files task and the
# Publish Build Artifacts task instead.
- task: CopyPublishBuildArtifacts@1
  inputs:
    #CopyRoot: # string. Copy Root.
    Contents: # string. Required. Contents.
    ArtifactName: # string. Required. Artifact Name.
    ArtifactType: # 'Container' | 'FilePath'. Required. Artifact Type.
    #TargetPath: '\\my\share\$(Build.DefinitionName)\$(Build.BuildNumber)' #
    # string. Optional. Use when ArtifactType = FilePath. Path. Default:
    # \\my\share\$(Build.DefinitionName)\$(Build.BuildNumber).
```

Inputs

CopyRoot - Copy Root

`string.`

The folder that contains the files you want to copy. If the folder is empty, the task copies files from the root folder of the repo as though `$(Build.SourcesDirectory)` was specified.

If your build produces artifacts outside of the sources directory, specify `$(Agent.BuildDirectory)` to copy files from the build agent working directory.

Contents - Contents

`string`. Required.

Specifies pattern filters (one on each line) that you want to apply to the list of files to be copied. For example:

- `**` copies all files in the root folder.
- `***` copies all files in the root folder and all files in all sub-folders.
- `**\bin` copies files in any sub-folder named `bin`.

ArtifactName - Artifact Name

`string`. Required.

Specifies the name of the artifact to create.

ArtifactType - Artifact Type

`string`. Required. Allowed values: `Container` (Server), `FilePath` (File share).

Specifies whether to store the artifact on TFS/Team Services or to copy it to a file share that must be accessible from the build agent.

TargetPath - Path

`string`. Optional. Use when `ArtifactType = FilePath`. Default value:

`\my\share\$(Build.DefinitionName)\$(Build.BuildNumber)`.

The UNC file path location where the artifact is copied. It must be accessible from the build agent.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Important

This task is deprecated. We recommend that you use [Pipeline Artifacts](#) and the [Copy Files task](#) and the [Publish Build Artifacts](#) task instead.

This step didn't produce the outcome I was expecting. How can I fix it?

This task has a couple of known issues:

- Some minimatch patterns don't work.
- It eliminates the most common root path for all paths matched.

You can avoid these issues by instead using the [Copy Files task](#) and the [Publish Build Artifacts task](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.83.0 or greater
Task category	Utility

CopyFiles@2 - Copy files v2 task

Article • 09/26/2023

Use this task to copy files from a source folder to a target folder using match patterns. (The match patterns will only match file paths, not folder paths).

Syntax

YAML

```
# Copy files v2
# Copy files from a source folder to a target folder using patterns matching
# file paths (not folder paths).
- task: CopyFiles@2
  inputs:
    #SourceFolder: # string. Source Folder.
    Contents: '**' # string. Required. Contents. Default: **.
    TargetFolder: # string. Required. Target Folder.
    # Advanced
    #CleanTargetFolder: false # boolean. Clean Target Folder. Default:
    false.
    #OverWrite: false # boolean. Overwrite. Default: false.
    #flattenFolders: false # boolean. Flatten Folders. Default: false.
    #preserveTimestamp: false # boolean. Preserve Target Timestamp. Default:
    false.
    #retryCount: '0' # string. Retry count to copy the file. Default: 0.
    #delayBetweenRetries: '1000' # string. Delay between two retries.
    Default: 1000.
    #ignoreMakeDirErrors: false # boolean. Ignore errors during creation of
    target folder. Default: false.
```

Inputs

SourceFolder - Source Folder

string.

Optional. The folder that contains the files you want to copy. If the folder is empty, then the task copies files from the root folder of the repo as though **\$(Build.SourcesDirectory)** was specified.

If your build produces artifacts outside of the sources directory, specify **\$(Agent.BuildDirectory)** to copy files from the directory created for the pipeline.

Contents - Contents

`string`. Required. Default value: `**`.

The file paths to include as part of the copy. This string supports multiple lines of match patterns.

For example:

- `*` copies all files in the specified source folder.
- `**` copies all files in the specified source folder and all files in all sub-folders.
- `**\bin**` copies all files recursively from any bin folder.

The pattern is used to match only file paths, not folder paths. Specify patterns, such as `**\bin**` instead of `**\bin`.

Use the path separator that matches your build agent type. For example, `/` must be used for Linux agents. More examples are shown below.

TargetFolder - Target Folder

`string`. Required.

The target folder or UNC path that will contain the copied files. You can use [variables](#).

Example: `$(build.artifactstagingdirectory)`.

CleanTargetFolder - Clean Target Folder

`boolean`. Default value: `false`.

Optional. Deletes all existing files in the target folder before the copy process.

OverWrite - Overwrite

`boolean`. Default value: `false`.

Optional. Replaces the existing files in the target folder.

flattenFolders - Flatten Folders

`boolean`. Default value: `false`.

Optional. Flattens the folder structure and copies all files into the specified target folder.

`preserveTimestamp` - Preserve Target Timestamp

`boolean`. Default value: `false`.

Preserves the target file timestamp by using the original source file.

`retryCount` - Retry count to copy the file

`string`. Default value: `0`.

Specifies the retry count to copy the file. This string is useful for intermittent issues, such as UNC target paths on a remote host.

`delayBetweenRetries` - Delay between two retries.

`string`. Default value: `1000`.

Specifies the delay between two retries. This string is useful for intermittent issues, such as UNC target paths on a remote host.

`ignoreMakeDirErrors` - Ignore errors during creation of target folder.

`boolean`. Default value: `false`.

Ignores errors that occur during the creation of the target folder. This string is useful for avoiding issues with the parallel execution of tasks by several agents within one target folder.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

If no files match, the task will still report success.

- If `Overwrite` is `false` and a matched file already exists in the target folder, the task will not report failure but log that the file already exists and skip it.
- If `Overwrite` is `true` and a matched file already exists in the target folder, the matched file will be overwritten.

Examples

Copy file to artifacts staging directory and publish

YAML

```
steps:
- task: CopyFiles@2
  inputs:
    contents: '_buildOutput/**'
    targetFolder: $(Build.ArtifactStagingDirectory)
- task: PublishBuildArtifacts@1
  inputs:
    pathToPublish: $(Build.ArtifactStagingDirectory)
    artifactName: MyBuildOutputs
```

Copy executables and a readme file

Goal

You want to copy just the readme and the files needed to run this C# console app:

```
`-- ConsoleApplication1
  |-- ConsoleApplication1.sln
  |-- readme.txt
  `-- ClassLibrary1
      |-- ClassLibrary1.csproj
  `-- ClassLibrary2
      |-- ClassLibrary2.csproj
`-- ConsoleApplication1
    |-- ConsoleApplication1.csproj
```

 Note

ConsoleApplication1.sln contains a *bin* folder with .dll and .exe files, see the Results below to see what gets moved!

On the Variables tab, `$(BuildConfiguration)` is set to `release`.

YAML

Example with multiple match patterns:

YAML

```
steps:
- task: CopyFiles@2
  displayName: 'Copy Files to: $(Build.ArtifactStagingDirectory)'
  inputs:
    Contents: |
      ConsoleApplication1\ConsoleApplication1\bin\**\*.exe
      ConsoleApplication1\ConsoleApplication1\bin\**\*.dll
      ConsoleApplication1\readme.txt
    TargetFolder: '$(Build.ArtifactStagingDirectory)'
```

Example with OR condition:

YAML

```
steps:
- task: CopyFiles@2
  displayName: 'Copy Files to: $(Build.ArtifactStagingDirectory)'
  inputs:
    Contents: |
      ConsoleApplication1\ConsoleApplication1\bin\**\?(*.exe|*.dll)
      ConsoleApplication1\readme.txt
    TargetFolder: '$(Build.ArtifactStagingDirectory)'
```

Example with NOT condition:

YAML

```
steps:
- task: CopyFiles@2
  displayName: 'Copy Files to: $(Build.ArtifactStagingDirectory)'
  inputs:
    Contents: |
      !ConsoleApplication1\**\bin\**\!(*.pdb|*.config)
      !ConsoleApplication1\**\ClassLibrary\**
      ConsoleApplication1\readme.txt
    TargetFolder: '$(Build.ArtifactStagingDirectory)'
```

Example with variables in content section

YAML

```
- task: CopyFiles@2
  inputs:
    Contents: '$(Build.Repository.LocalPath)/**'
    TargetFolder: '$(Build.ArtifactStagingDirectory)'
```

Results

These files are copied to the staging directory:

```
`-- ConsoleApplication1
  |-- readme.txt
  '-- ConsoleApplication1
    '-- bin
      '-- Release
        | -- ClassLibrary1.dll
        | -- ClassLibrary2.dll
        | -- ConsoleApplication1.exe
```

Copy everything from the source directory except the .git folder

YAML

Example with multiple match patterns:

YAML

```
steps:
- task: CopyFiles@2
  displayName: 'Copy Files to: $(Build.ArtifactStagingDirectory)'
  inputs:
    SourceFolder: '$(Build.SourcesDirectory)'
    Contents: |
      **/*
      !.git/**/*
    TargetFolder: '$(Build.ArtifactStagingDirectory)'
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : Setting variables is disabled
Agent version	2.182.1 or greater
Task category	Utility

See also

- [File matching patterns reference](#)
- [How do I use this task to publish artifacts](#)
- Learn how to use [verbose logs for troubleshooting](#).

CopyFiles@1 - Copy Files v1 task

Article • 09/26/2023

Use this task to copy files from a source folder to a target folder using match patterns.
(The match patterns will only match file paths, not folder paths).

Syntax

YAML

```
# Copy Files v1
# Copy files from source folder to target folder using minimatch patterns
(The minimatch patterns will only match file paths, not folder paths).
- task: CopyFiles@1
  inputs:
    #SourceFolder: # string. Source Folder.
    Contents: '**' # string. Required. Contents. Default: **.
    TargetFolder: # string. Required. Target Folder.
  # Advanced
  #CleanTargetFolder: false # boolean. Clean Target Folder. Default:
false.
  #OverWrite: false # boolean. Overwrite. Default: false.
  #flattenFolders: false # boolean. Flatten Folders. Default: false.
```

Inputs

SourceFolder - Source Folder

`string`.

Optional. The folder that contains the files you want to copy. If the folder is empty, then the task copies files from the root folder of the repo as though [\\$\(Build.SourcesDirectory\)](#) was specified.

If your build produces artifacts outside of the sources directory, specify [\\$\(Agent.BuildDirectory\)](#) to copy files from the directory created for the pipeline.

Contents - Contents

`string`. Required. Default value: `**`.

The file paths to include as part of the copy. This string supports multiple lines of match patterns.

For example:

- `*` copies all files in the specified source folder.
- `**` copies all files in the specified source folder and all files in all sub-folders.
- `**\bin**` copies all files recursively from any bin folder.

The pattern is used to match only file paths, not folder paths. Specify patterns, such as `**\bin**` instead of `**\bin`.

Use the path separator that matches your build agent type. For example, `/` must be used for Linux agents. More examples are shown below.

TargetFolder - Target Folder

`string`. Required.

The target folder or UNC path that will contain the copied files. You can use [variables](#).

Example: `$(build.artifactstagingdirectory)`.

CleanTargetFolder - Clean Target Folder

`boolean`. Default value: `false`.

Optional. Deletes all existing files in the target folder before the copy process.

OverWrite - Overwrite

`boolean`. Default value: `false`.

Optional. Replaces the existing files in the target folder.

flattenFolders - Flatten Folders

`boolean`. Default value: `false`.

Optional. Flattens the folder structure and copies all files into the specified target folder.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

There is a newer version of this task available at [CopyFiles@2](#).

If no files match, the task will still report success. If a matched file already exists in the target folder, the task will report failure unless `Overwrite` is set to true.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.91.0 or greater
Task category	Utility

See also

- [CopyFiles@2](#)

CopyFilesOverSSH@0 - Copy files over SSH v0 task

Article • 09/26/2023

Copy files or build artifacts to a remote machine over SSH.

Syntax

YAML

```
# Copy files over SSH v0
# Copy files or build artifacts to a remote machine over SSH.
- task: CopyFilesOverSSH@0
  inputs:
    sshEndpoint: # string. Required. SSH service connection.
    #sourceFolder: # string. Source folder.
    contents: '**' # string. Required. Contents. Default: **.
    #targetFolder: # string. Target folder.
    # Advanced
    #isWindowsOnTarget: false # boolean. Target machine running Windows.
    Default: false.
    #cleanTargetFolder: false # boolean. Clean target folder. Default:
    false.
    #cleanHiddenFilesInTarget: false # boolean. Optional. Use when
    cleanTargetFolder = true. Remove hidden files in target folder. Default:
    false.
    readyTimeout: '20000' # string. Required. SSH handshake timeout.
    Default: 20000.
    #overwrite: true # boolean. Overwrite. Default: true.
    #failOnEmptySource: false # boolean. Fail if no files found to copy.
    Default: false.
    #flattenFolders: false # boolean. Flatten folders. Default: false.
```

Inputs

`sshEndpoint` - SSH service connection

`string`. Required.

The name of an [SSH service connection](#) containing connection details for the remote machine.

- The hostname or IP address of the remote machine, the port number, and the user name are required to create an SSH service connection.
- The private key and the passphrase must be specified for authentication.

sourceFolder - Source folder`string.`

The source folder of the files to copy to the remote machine. When empty, the root of the repository (build) or artifacts directory (release) is used, which is

`$(System.DefaultWorkingDirectory)`. Use [variables](#) if files are not in the repository.

Example: `$(Agent.BuildDirectory)`.

contents - Contents`string`. Required. Default value: `**`.

The file paths to include as part of the copy. Supports multiple lines of [minimatch patterns](#). The default value is `**`, which includes all files (including sub-folders) under the source folder.

- Example: `**/*.*(jar|war)` includes all .jar and .war files (including sub-folders) under the source folder.
- Example: `** \n !**/*.xml` includes all files (including sub-folders) under the source folder, but excludes xml files.

targetFolder - Target folder`string.`

The target folder on the remote machine, where files will be copied. Example:

`/home/user/MySite`. Preface with a tilde (`~`) to specify the user's home directory.

isWindowsOnTarget - Target machine running Windows`boolean`. Default value: `false`.

Checks if the target machine is running Windows.

cleanTargetFolder - Clean target folder`boolean`. Default value: `false`.

Deletes all existing files and sub-folders in the target folder before copying.

`cleanHiddenFilesInTarget` - Remove hidden files in target folder

`boolean`. Optional. Use when `cleanTargetFolder = true`. Default value: `false`.

When set to `true`, removes hidden files in the target folder.

`readyTimeout` - SSH handshake timeout

`string`. Required. Default value: `20000`.

How long (in milliseconds) to wait for the SSH handshake to complete.

`overwrite` - Overwrite

`boolean`. Default value: `true`.

Replaces existing files in and beneath the target folder.

`failOnEmptySource` - Fail if no files found to copy

`boolean`. Default value: `false`.

Fails if no matching files to be copied are found under the source folder.

`flattenFolders` - Flatten folders

`boolean`. Default value: `false`.

Flattens the folder structure and copies all files into the specified target folder on the remote machine.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : Setting variables is disabled
Agent version	2.182.1 or greater
Task category	Deploy

cURLUploader@2 - cURL Upload Files v2 task

Article • 09/26/2023

Use this task with [cURL](#) to upload files. Supported data transfer protocols include FTP, FTPS, SFTP, HTTP, and others.

Syntax

YAML

```
# cURL Upload Files v2
# Use cURL's supported protocols to upload files.
- task: cURLUploader@2
  inputs:
    files: # string. Required. Files.
    #authType: 'ServiceEndpoint' # 'ServiceEndpoint' | 'UserAndPass'.
    Authentication Method. Default: ServiceEndpoint.
    serviceEndpoint: # string. Required when authType = ServiceEndpoint.
    Service Connection.
    #username: # string. Optional. Use when authType = UserAndPass.
    Username.
    #password: # string. Optional. Use when authType = UserAndPass.
    Password.
    #url: # string. Required when authType = UserAndPass. URL.
    #remotePath: 'upload/${Build.BuildId}/' # string. Remote Directory.
    Default: upload/${Build.BuildId}/.
    #options: # string. Optional Arguments.
    # Advanced
    #redirectStderr: true # boolean. Redirect Standard Error to Standard
    Out. Default: true.
```

Inputs

files - Files

string. Required.

File(s) to be uploaded. Wildcards can be used. For example, `**/*.zip` for all ZIP files in all subfolders.

authType - Authentication Method

string. Allowed values: `ServiceEndpoint` (Service connection), `UserAndPass` (Username

and password). Default value: `ServiceEndpoint`.

Specifies the authentication method for server authentication.

serviceEndpoint - Service Connection

`string`. Required when `authType = ServiceEndpoint`.

Specifies the service connection with the credentials for the server authentication. Use the Generic service connection type for the service connection.

username - Username

`string`. Optional. Use when `authType = UserAndPass`.

Specifies the username for server authentication.

password - Password

`string`. Optional. Use when `authType = UserAndPass`.

Specifies the password for server authentication. Use a new build variable with its lock enabled on the Variables tab to encrypt this value. Use a [secret variable](#) to avoid exposing this value.

url - URL

`string`. Required when `authType = UserAndPass`.

Specifies the URL to where the file(s) will be uploaded. The directory should end with a trailing slash. Possible URL protocols include `DICT://`, `FILE://`, `FTP://`, `FTPS://`, `GOPHER://`, `HTTP://`, `HTTPS://`, `IMAP://`, `IMAPS://`, `LDAP://`, `LDAPS://`, `POP3://`, `POP3S://`, `RTMP://`, `RTSP://`, `SCP://`, `SFTP://`, `SMTP://`, `SMTPS://`, `TELNET://` and `TFTP://`.

remotePath - Remote Directory

`string`. Default value: `upload/$(Build.BuildId)/`.

Optional. Specifies the sub-folder on the remote server for the URL supplied in the credentials.

`options` - Optional Arguments

`string`.

Optional. The additional arguments that will be passed to cURL.

`redirectStderr` - Redirect Standard Error to Standard Out

`boolean`. Default value: `true`.

Adds `--stderr` as an argument to cURL. By default, cURL writes its progress bar to `stderr`, which is interpreted by the build as error output. Enabling this checkbox suppresses that behavior.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to use [cURL](#) to upload files with supported protocols such as FTP, FTPS, SFTP, HTTP, and more.

Where can I learn more about file matching patterns?

- [File matching patterns reference](#)

Where can I learn FTP commands?

- [List of raw FTP commands](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : Setting variables is disabled
Agent version	2.182.1 or greater
Task category	Utility

cURLUploader@1 - cURL Upload Files v1 task

Article • 09/26/2023

Use this task with [cURL](#) to upload files. Supported data transfer protocols include FTP, FTPS, SFTP, HTTP, and others.

Syntax

YAML

```
# cURL Upload Files v1
# Use cURL to upload files with FTP, FTPS, SFTP, HTTP, and more.
- task: cURLUploader@1
  inputs:
    files: # string. Required. Files.
    #username: # string. Username.
    #password: # string. Password.
    url: # string. Required. URL.
    #options: # string. Optional Arguments.
  # Advanced
  #redirectStderr: true # boolean. Redirect Standard Error to Standard
  Out. Default: true.
```

Inputs

`files` - Files

`string`. Required.

The file(s) to be uploaded. Wildcards can be used. For example, `***.zip` for all ZIP files in all subfolders.

`username` - Username

`string`.

Optional. Specifies the username for server authentication.

`password` - Password

`string`.

Optional. Specifies the password for server authentication. Use a new build variable with its lock enabled on the Variables tab to encrypt this value. Use a [secret variable](#) to avoid exposing this value.

`url` - URL

`string`. Required.

Specifies the URL to where the file(s) will be uploaded. The directory should end with a trailing slash. Possible URL protocols include `DICT://`, `FILE://`, `FTP://`, `FTPS://`, `GOPHER://`, `HTTP://`, `HTTPS://`, `IMAP://`, `IMAPS://`, `LDAP://`, `LDAPS://`, `POP3://`, `POP3S://`, `RTMP://`, `RTSP://`, `SCP://`, `SFTP://`, `SMTP://`, `SMTPS://`, `TELNET://` and `TFTP://`.

`options` - Optional Arguments

`string`.

Optional. The additional arguments that will be passed to cURL.

`redirectStderr` - Redirect Standard Error to Standard Out

`boolean`. Default value: `true`.

Adds `--stderr` as an argument to cURL. By default, cURL writes its progress bar to `stderr`, which is interpreted by the build as error output. Enabling this checkbox suppresses that behavior.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Where can I learn FTP commands?

See the [list of raw FTP commands ↗](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: curl
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Utility

DecryptFile@1 - Decrypt file (OpenSSL)

v1 task

Article • 09/26/2023

Use this task to decrypt files using OpenSSL.

Syntax

YAML

```
# Decrypt file (OpenSSL) v1
# Decrypt a file using OpenSSL.
- task: DecryptFile@1
  inputs:
    cipher: 'des3' # string. Required. Cypher. Default: des3.
    inFile: # string. Required. Encrypted file.
    passphrase: # string. Required. Passphrase.
    #outFile: # string. Decrypted file path.
    # Advanced
    #workingDirectory: # string. Alias: cwd. Working directory.
```

Inputs

cipher - Cypher

`string`. Required. Default value: `des3`.

The encryption cypher to use. See [cypher suite names](#) for a complete list of possible values.

inFile - Encrypted file

`string`. Required.

The relative path of the file to decrypt.

passphrase - Passphrase

`string`. Required.

The passphrase to use for decryption. **Use a variable to encrypt the passphrase.**

`outfile` - Decrypted file path

`string`.

The optional filename for the decrypted file. Defaults to the encrypted file with an `.out` extension.

`workingDirectory` - Working directory

Input alias: `cwd`. `string`.

The optional working directory for decryption. Defaults to the root of the repository.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to decrypt files using OpenSSL.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : Setting variables is disabled

Requirement	Description
Agent version	2.182.1 or greater
Task category	Utility

Delay@1 - Delay v1 task

Article • 09/26/2023

Delays further execution of a workflow by a fixed time.

Syntax

YAML

```
# Delay v1
# Delay further execution of a workflow by a fixed time.
- task: Delay@1
  inputs:
    delayForMinutes: '0' # string. Required. Delay Time (minutes). Default:
    0.
```

Inputs

`delayForMinutes` - Delay Time (minutes)

`string`. Required. Default value: `0`.

Delays the execution of the workflow by specified time in minutes. A `0` value means that workflow execution will start without delay. The maximum value is `86400` (60 days).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task in an [agentless job](#) of a release pipeline to pause the execution of the pipeline for a fixed delay time.

 **Note**

Can be used in only an **agentless job** of a release pipeline.

The maximum value for a delay is 60 days (86400 minutes).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Server
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Utility

DeleteFiles@1 - Delete files v1 task

Article • 09/26/2023

Delete folders, or files matching a pattern.

Syntax

YAML

```
# Delete files v1
# Delete folders, or files matching a pattern.
- task: DeleteFiles@1
  inputs:
    #SourceFolder: # string. Source Folder.
    Contents: 'myFileShare' # string. Required. Contents. Default:
    myFileShare.
    #RemoveSourceFolder: false # boolean. Remove SourceFolder. Default:
    false.
    # Advanced
    #RemoveDotFiles: false # boolean. Remove files starting with a dot.
    Default: false.
```

Inputs

SourceFolder - Source Folder

string.

If the source folder is empty, the task deletes files from the root folder of the repository as though `$(Build.SourcesDirectory)` was specified. If your build produces artifacts outside of the sources directory, specify `$(Agent.BuildDirectory)` to delete files from the build agent working directory.

Contents - Contents

string. Required. Default value: `myFileShare`.

The file/folder paths to delete. Supports multiple lines of minimatch patterns; each one is processed before moving onto the next line. Learn more about [File matching patterns](#). For example:

- `**/*` deletes all files and folders in the root folder.
- `temp` deletes the *temp* folder in the root folder.

- `temp*` deletes any file or folder in the root folder with a name that begins with *temp*.
- `**/temp/*` deletes all files and folders in any sub-folder named *temp*.
- `**/temp*` deletes any file or folder with a name that begins with *temp*.
- `!(*.vsix)` deletes all files in the root folder that do not have a `.vsix` extension.

`RemoveSourceFolder` - Remove SourceFolder

`boolean`. Default value: `false`.

Attempts to remove the source folder after attempting to remove `Contents`. If you want to remove the whole folder, set this to `true` and set `Contents` to `*`.

`RemoveDotFiles` - Remove files starting with a dot

`boolean`. Default value: `false`.

Deletes files starting with a dot. For example: `.git` and `.dockerfile`. Omits these files if it's not specified explicitly. For example: `/.*`. Learn more about [minimatch](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to delete files or folders from the agent working directory.

Examples

Delete several patterns

This example will delete `some/file`, all files beginning with `test`, and all files in all subdirectories called `bin`.

YAML

```
steps:
- task: DeleteFiles@1
  displayName: 'Remove unneeded files'
  inputs:
    contents: |
      some/file
      test*
      **/bin/*
```

Delete all but one subdirectory

This example will delete `some/one`, `some/three` and `some/four` but will leave `some/two`.

YAML

```
steps:
- task: DeleteFiles@1
  displayName: 'Remove unneeded files'
  inputs:
    contents: |
      some/!(two)
```

Delete using brace expansion

This example will delete `some/one` and `some/four` but will leave `some/two` and `some/three`.

YAML

```
steps:
- task: DeleteFiles@1
  displayName: 'Remove unneeded files'
  inputs:
    contents: |
      some/{one,four}
```

Delete files starting with a dot

This example will delete all `.txt` files. Files starting with a dot will be deleted as well.

YAML

```
steps:
- task: DeleteFiles@1
  displayName: 'Remove unneeded files'
  inputs:
    contents: |
      /some/*.txt
    removeDotFiles: true
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : Setting variables is disabled
Agent version	2.182.1 or greater
Task category	Utility

AzureStaticWebApp@0 - Deploy Azure Static Web App v0 task

Article • 09/26/2023

This task builds and deploys an Azure Static Web app.

Syntax

YAML

```
# Deploy Azure Static Web App v0
# Build and deploy an Azure Static Web App.
- task: AzureStaticWebApp@0
  inputs:
    #workingDirectory: '$(System.DefaultWorkingDirectory)' # string. Alias:
    cwd | rootDirectory. Working directory. Default:
    $(System.DefaultWorkingDirectory).
    #app_location: # string. App location.
    #app_build_command: # string. App build command.
    #output_location: # string. Output location.
    #api_location: # string. Api location.
    #api_build_command: # string. Api build command.
    #routes_location: # string. Routes location.
    #config_file_location: # string. Config file location.
    #skip_app_build: # boolean. Skip app build.
    #skip_api_build: # boolean. Skip api build.
    #is_static_export: # boolean. Set static export.
    #verbose: # boolean. Verbose.
    #build_timeout_in_minutes: # string. Build timeout in minutes.
    #azure_static_web_apps_api_token: # string. Azure Static Web Apps api
token.
    #deployment_environment: # string. Deployment Environment.
    #production_branch: # string. Production Branch.
    #data_api_location: # string. Data api location.
```

Inputs

workingDirectory - Working directory

Input alias: cwd | rootDirectory. string. Default value:
\$(System.DefaultWorkingDirectory).

Specifies the absolute working directory in which to execute this task. If left empty, the default working directory is used.

app_location - App location`string.`

The directory location of the application source code, relative to the working directory.

app_build_command - App build command`string.`

The custom command used to run Oryx when building application source code.

output_location - Output location`string.`

The directory location of the compiled application code after building is complete, relative to the working directory.

api_location - Api location`string.`

The directory location of the Azure Functions source code, relative to the working directory.

api_build_command - Api build command`string.`

The custom command used to run Oryx when building Azure Functions source code.

routes_location - Routes location`string.`

The directory location of the routes.json file, relative to the working directory.

Note: Routes.json is deprecated. Use staticwebapp.config.json.

config_file_location - Config file location`string.`

The directory location of the staticwebapp.config.json file, relative to the working directory.

`skip_app_build` - Skip app build

`boolean`.

Skips Oryx build for the app folder.

`skip_api_build` - Skip api build

`boolean`.

Skips Oryx build for the API folder.

`is_static_export` - Set static export

`boolean`.

Set this flag to `true` when your application is configured to export to static HTML, like when using `next export`.

When this flag is set to `true`

`verbose` - Verbose

`boolean`.

Enables verbose logging.

`build_timeout_in_minutes` - Build timeout in minutes

`string`.

Specifies the time limit of the Oryx app folder build in minutes.

`azure_static_web_apps_api_token` - Azure Static Web Apps api token

`string`.

Specifies the API token for deployment.

Note: Not required if passed as an environment variable.

`deployment_environment` - Deployment Environment

`string`.

Specifies the environment to deploy to. Leave blank for the production environment. This input takes precedence over the production branch.

production_branch - Production Branch

`string`.

Specifies the production branch. When defined, and the deployment environment is empty, deployments from other branches will be preview environments.

data_api_location - Data api location

`string`.

Directory location of the Data API source files relative to working directory.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

ⓘ Note

This task only runs on Linux agents.

Examples

YAML

```
trigger:
  - main

pool:
  vmImage: ubuntu-latest

steps:
  - checkout: self
```

```
submodules: true
- task: AzureStaticWebApp@0
  inputs:
    app_location: '/build'
    api_location: 'api'
    output_location: '/output'
    azure_static_web_apps_api_token: $(deployment_token)
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.95.0 or greater
Task category	Utility

KubernetesManifest@1 - Deploy to Kubernetes v1 task

Article • 09/26/2023

Use Kubernetes manifest files to deploy to clusters or even bake the manifest files to be used for deployments using Helm charts.

Syntax

YAML

```
# Deploy to Kubernetes v1
# Use Kubernetes manifest files to deploy to clusters or even bake the
manifest files to be used for deployments using Helm charts.
- task: KubernetesManifest@1
  inputs:
    #action: 'deploy' # 'bake' | 'createSecret' | 'delete' | 'deploy' |
    'patch' | 'promote' | 'scale' | 'reject'. Action. Default: deploy.
    #connectionType: 'kubernetesServiceConnection' # 'azureResourceManager'
    | 'kubernetesServiceConnection'. Required when action != bake. Service
    connection type. Default: kubernetesServiceConnection.
    #kubernetesServiceConnection: # string. Alias:
    kubernetesServiceEndpoint. Required when action != bake && connectionType =
    kubernetesServiceConnection. Kubernetes service connection.
    #azureSubscriptionConnection: # string. Alias:
    azureSubscriptionEndpoint. Required when action != bake && connectionType =
    azureResourceManager. Azure subscription.
    #azureResourceGroup: # string. Required when action != bake &&
    connectionType = azureResourceManager. Resource group.
    #kubernetesCluster: # string. Required when action != bake &&
    connectionType = azureResourceManager. Kubernetes cluster.
    #useClusterAdmin: false # boolean. Optional. Use when connectionType =
    azureResourceManager. Use cluster admin credentials. Default: false.
    #namespace: # string. Namespace.
    #strategy: 'none' # 'canary' | 'none'. Optional. Use when action =
    deploy || action = promote || action = reject. Strategy. Default: none.
    #trafficSplitMethod: 'pod' # 'pod' | 'smi'. Optional. Use when strategy
    = canary. Traffic split method. Default: pod.
    #percentage: '0' # string. Required when strategy = Canary && action =
    deploy. Percentage. Default: 0.
    #baselineAndCanaryReplicas: '1' # string. Required when strategy =
    Canary && action = deploy && trafficSplitMethod = SMI. Baseline and canary
    replicas. Default: 1.
    #manifests: # string. Required when action = deploy || action = promote
    || action = reject. Manifests.
    #containers: # string. Optional. Use when action = deploy || action =
    promote || action = bake. Containers.
    #imagePullSecrets: # string. Optional. Use when action = deploy ||
    #imagePullSecrets: # string. Optional. Use when action = deploy ||
```

```

action = promote. ImagePullSecrets.
    #renderType: 'helm' # 'helm' | 'kompose' | 'kustomize'. Optional. Use
when action = bake. Render Engine. Default: helm.
    #dockerComposeFile: # string. Required when action = bake && renderType
= kompose. Path to docker compose file.
    #helmChart: # string. Required when action = bake && renderType = helm.
Helm Chart.
    #releaseName: # string. Optional. Use when action = bake && renderType = helm.
Helm Release Name.
    #overrideFiles: # string. Optional. Use when action = bake && renderType
= helm. Override Files.
    #overrides: # string. Optional. Use when action = bake && renderType = helm.
Overrides.
    #kustomizationPath: # string. Optional. Use when action = bake &&
renderType = kustomize. Kustomization Path.
    #resourceToPatch: 'file' # 'file' | 'name'. Required when action =
patch. Resource to patch. Default: file.
    #resourceFileToPatch: # string. Required when action = patch &&
resourceToPatch = file. File path.
    #kind: # 'deployment' | 'replicaset' | 'statefulset'. Required when
action = scale || resourceToPatch = name. Kind.
    #name: # string. Required when action = scale || resourceToPatch = name.
Name.
    #replicas: # string. Required when action = scale. Replica count.
    #mergeStrategy: 'strategic' # 'json' | 'merge' | 'strategic'. Required
when action = patch. Merge Strategy. Default: strategic.
    #arguments: # string. Optional. Use when action = delete. Arguments.
    #patch: # string. Required when action = patch. Patch.
    #secretType: 'dockerRegistry' # 'dockerRegistry' | 'generic'. Required
when action = createSecret. Type of secret. Default: dockerRegistry.
    #secretName: # string. Optional. Use when action = createSecret. Secret
name.
    #secretArguments: # string. Optional. Use when action = createSecret &&
secretType = generic. Arguments.
    #dockerRegistryEndpoint: # string. Optional. Use when action =
createSecret && secretType = dockerRegistry. Docker registry service
connection.
    #rolloutStatusTimeout: '0' # string. Optional. Use when action = deploy
|| action = patch || action = scale || action = promote. Timeout for rollout
status. Default: 0.

```

Inputs

`action` - Action

`string`. Allowed values: `bake`, `createSecret` (create secret), `delete`, `deploy`, `patch`, `promote`, `scale`, `reject`. Default value: `deploy`.

Specifies the action to be performed.

`connectionType` - Service connection type

`string`. Required when `action != bake`. Allowed values: `azureResourceManager` (Azure Resource Manager), `kubernetesServiceConnection` (Kubernetes Service Connection). Default value: `kubernetesServiceConnection`.

Select a Kubernetes service connection type.

- `kubernetesServiceConnection` (Kubernetes Service Connection) - Allows you to provide a KubeConfig file, specify a Service Account, or import an AKS instance with the **Azure Subscription** option. Importing an AKS instance with the **Azure Subscription** option requires Kubernetes cluster access at Service Connection configuration time.
- `azureResourceManager` (Azure Resource Manager) - Lets you select an AKS instance. Does not access Kubernetes cluster at Service Connection configuration time.

For more information, see [Remarks](#).

`kubernetesServiceConnection` - Kubernetes service connection

Input alias: `kubernetesServiceEndpoint`. `string`. Required when `action != bake && connectionType = kubernetesServiceConnection`.

Specifies a [Kubernetes service connection](#).

`azureSubscriptionConnection` - Azure subscription

Input alias: `azureSubscriptionEndpoint`. `string`. Required when `action != bake && connectionType = azureResourceManager`.

Select the Azure Resource Manager subscription, which contains Azure Container Registry. Note: To configure new service connection, select the Azure subscription from the list and click 'Authorize'. If your subscription is not listed or if you want to use an existing Service Principal, you can setup an Azure service connection using 'Add' or 'Manage' button.

`azureResourceGroup` - Resource group

`string`. Required when `action != bake && connectionType = azureResourceManager`.

Select an Azure resource group.

kubernetesCluster - Kubernetes cluster

`string`. Required when `action != bake && connectionType = azureResourceManager`.

Select an Azure managed cluster.

useClusterAdmin - Use cluster admin credentials

`boolean`. Optional. Use when `connectionType = azureResourceManager`. Default value: `false`.

Use cluster administrator credentials instead of default cluster user credentials.

namespace - Namespace

`string`.

Specifies the namespace for the commands by using the `-namespace` flag. If the namespace is not provided, the commands will run in the default namespace.

strategy - Strategy

`string`. Optional. Use when `action = deploy || action = promote || action = reject`.

Allowed values: `canary`, `none`. Default value: `none`.

Specifies the deployment strategy used in the `deploy` action before a `promote` action or `reject` action. Currently, `canary` is the only acceptable deployment strategy.

trafficSplitMethod - Traffic split method

`string`. Optional. Use when `strategy = canary`. Allowed values: `pod`, `smi`. Default value: `pod`.

For the value `smi`, the percentage traffic split is done at the request level by using a service mesh. A service mesh must be set up by a cluster admin. This task handles orchestration of SMI [TrafficSplit](#) objects.

For the value `pod`, the percentage split isn't possible at the request level in the absence of a service mesh. Instead, the percentage input is used to calculate the replicas for baseline and canary. The calculation is a percentage of replicas that are specified in the input manifests for the stable variant.

percentage - Percentage

`string`. Required when `strategy = Canary && action = deploy`. Default value: `0`.

The percentage that is used to compute the number of baseline-variant and canary-variant replicas of the workloads that are contained in manifest files.

For the specified percentage input, calculate:

$$(percentage \times number\ of\ replicas) / 100$$

If the result isn't an integer, the mathematical floor of the result is used when baseline and canary variants are created.

For example, assume the deployment `hello-world` is in the input manifest file and that the following lines are in the task input:

```
replicas: 4
strategy: canary
percentage: 25
```

In this case, the deployments `hello-world-baseline` and `hello-world-canary` are created with one replica each. The baseline variant is created with the same image and tag as the stable version, which is the four-replica variant before deployment. The canary variant is created with the image and tag corresponding to the newly deployed changes.

baselineAndCanaryReplicas - Baseline and canary replicas

`string`. Required when `strategy = Canary && action = deploy && trafficSplitMethod = SMI`. Default value: `1`.

When you set `trafficSplitMethod` to `smi`, the percentage traffic split is controlled in the service mesh plane. You can control the actual number of replicas for canary and baseline variants independently of the traffic split.

For example, assume that the input deployment manifest specifies 30 replicas for the stable variant. Also assume that you specify the following input for the task:

```
strategy: canary
trafficSplitMethod: smi
percentage: 20
baselineAndCanaryReplicas: 1
```

In this case, the stable variant receives 80% of the traffic, while the baseline and canary variants each receive half of the specified 20%. Baseline and canary variants don't receive three replicas each. They instead receive the specified number of replicas, which means they each receive one replica.

manifests - Manifests

`string`. Required when `action = deploy || action = promote || action = reject`.

Specifies the path to the manifest files to be used for deployment. Each line represents a single path. A [file-matching pattern](#) is an acceptable value for each line.

containers - Containers

`string`. Optional. Use when `action = deploy || action = promote || action = bake`.

Specifies the fully qualified resource URL of the image to be used for substitutions on the manifest files. The URL `contosodemo.azurecr.io/helloworld:test` is an example.

imagePullSecrets - ImagePullSecrets

`string`. Optional. Use when `action = deploy || action = promote`.

Specifies a multiline input where each line contains the name of a Docker registry secret that has already been set up within the cluster. Each secret name is added under `imagePullSecrets` for the workloads that are found in the input manifest files.

renderType - Render Engine

`string`. Optional. Use when `action = bake`. Allowed values: `helm`, `kompose`, `kustomize`.

Default value: `helm`.

Specifies the render type used to produce the manifest files.

dockerComposeFile - Path to docker compose file

`string`. Required when `action = bake && renderType = kompose`.

Specifies a docker-compose file path.

helmChart - Helm Chart

`string`. Required when `action = bake && renderType = helm`.

Specifies the Helm chart path to bake.

releaseName - Helm Release Name

`string`. Optional. Use when `action = bake && renderType = helm`.

Specifies the Helm release name to use.

overrideFiles - Override Files

`string`. Optional. Use when `action = bake && renderType = helm`.

Specifies a multiline input that accepts the path to the override files. The files are used when manifest files from Helm charts are baked.

overrides - Overrides

`string`. Optional. Use when `action = bake && renderType = helm`.

Specifies the override values to set.

kustomizationPath - Kustomization Path

`string`. Optional. Use when `action = bake && renderType = kustomize`.

Specifies the argument that must be the path to the directory containing the file, or a git repository URL with a path suffix specifying `same` with respect to the repository root.

resourceToPatch - Resource to patch

`string`. Required when `action = patch`. Allowed values: `file`, `name`. Default value: `file`.

Indicates one of the following patch methods:

- A manifest file identifies the objects to be patched.
- An individual object is identified by kind and name as the patch target.

Acceptable values are `file` and `name`.

resourceFileToPatch - File path

`string`. Required when `action = patch && resourceToPatch = file`.

Specifies the path to the file used for a patch.

kind - Kind

`string`. Required when `action = scale || resourceToPatch = name`. Allowed values: `deployment`, `replicaset`, `statefulset`.

Specifies the kind of K8s object, such as `deployment`, `replicaSet` and more.

name - Name

`string`. Required when `action = scale || resourceToPatch = name`.

Specifies the name of the K8s object.

replicas - Replica count

`string`. Required when `action = scale`.

Specifies the number of replicas to scale to.

mergeStrategy - Merge Strategy

`string`. Required when `action = patch`. Allowed values: `json`, `merge`, `strategic`. Default value: `strategic`.

Specifies the type of patch being provided.

arguments - Arguments

`string`. Optional. Use when `action = delete`.

Specifies the arguments for the `kubectl delete` command. An example is: `arguments: deployment hello-world foo-bar`

patch - Patch

`string`. Required when `action = patch`.

Specifies the contents of the patch.

secretType - Type of secret

`string`. Required when `action = createSecret`. Allowed values: `dockerRegistry`, `generic`. Default value: `dockerRegistry`.

Creates or updates a generic or docker `imagepullsecret`. Specify `dockerRegistry` to create or update the `imagepullsecret` of the selected registry. An `imagePullSecret` is a way to pass a secret that contains a container registry password to the Kubelet, so it can pull a private image on behalf of your Pod.

`secretName` - Secret name

`string`. Optional. Use when `action = createSecret`.

Specifies the name of the secret. You can use this secret name in the Kubernetes YAML configuration file.

`secretArguments` - Arguments

`string`. Optional. Use when `action = createSecret && secretType = generic`.

Specifies keys and literal values to insert in secret. For example, `--from-literal=key1=value1 --from-literal=key2="top secret"`.

`dockerRegistryEndpoint` - Docker registry service connection

`string`. Optional. Use when `action = createSecret && secretType = dockerRegistry`.

Specifies the credentials of the specified service connection that are used to create a Docker registry secret within the cluster. Manifest files under the `imagePullSecrets` field can then refer to this secret's name.

`rolloutStatusTimeout` - Timeout for rollout status

`string`. Optional. Use when `action = deploy || action = patch || action = scale || action = promote`. Default value: `0`.

Specifies the length of time (in seconds) to wait before ending `watch on rollout` status.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

manifestsBundle

The location of the manifest bundles created by bake action

Remarks

Kubernetes Service Connection considerations when accessing AKS

You can create a Kubernetes service connection with any of the following options.

- KubeConfig
- Service Account
- Azure Subscription

New Kubernetes service connection

Authentication method

- KubeConfig
- Service Account
- Azure Subscription

When selecting the **Azure Subscription** option, Kubernetes needs to be accessible to Azure DevOps at service connection configuration time. There may be various reasons a service connection cannot be created, for example you created a private cluster or the cluster has local accounts disabled. In these cases, Azure DevOps is unable to connect to your cluster at service connection configuration time and you will observe the dialog to be stuck at **Loading namespaces**.



Loading namespaces

Starting with Kubernetes 1.24, long-lived tokens are [no longer created by default](#).

Kubernetes recommends not to use long-lived tokens. As a result, tasks using a Kubernetes service connection created using the Azure Subscription option do not have access to the permanent token required to authenticate and can't access your Kubernetes cluster. This also results in the **Loading namespaces** dialog to be frozen.

Use the Azure Resource Manager Service Connection to access AKS

For AKS customers, the Azure Resource Manager service connection type provides the best method to connect to a private cluster, or a cluster that has local accounts disabled. This method is not dependent on cluster connectivity at the time you create a service connection. Access to AKS is deferred to pipeline runtime, which has the following advantages:

- Access to a (private) AKS cluster can be performed from a self-hosted or scale set agent with line of sight to the cluster.
- A token is created for every task that uses an Azure Resource Manager service connection. This ensures you are connecting to Kubernetes with a short-lived token, which is the [Kubernetes recommendation](#).
- AKS can be accessed even when local accounts are disabled.

Service connection FAQ

I receive the following error message: Could not find any secret associated with the service account. What is happening?

You are using the Kubernetes service connection with Azure Subscription option. We are updating this method to create long-lived tokens. This is expected to be available mid-May. However, it is recommended to start using the Azure service connection type and not use long-lived tokens as per [Kubernetes guidance](#).

I'm using AKS and don't want to change anything, can I continue to use tasks with the Kubernetes service connection?

We are updating this method to create long-lived tokens. This is expected to be available mid-May. However, please be aware that this approach is against [Kubernetes guidance](#).

I'm using the Kubernetes tasks and Kubernetes service connection but not AKS. Should I be concerned?

Your tasks will continue to work as before.

Will the Kubernetes service connection type be removed?

Our Kubernetes tasks work with any Kubernetes cluster, regardless where they are running. The Kubernetes service connection will continue to exist.

I'm an AKS customer and everything is running fine, should I act?

There is no need to change anything. If you are using the Kubernetes service connection and selected Azure Subscription during creation, you should be aware of the [Kubernetes guidance on using long-lived tokens](#).

I'm creating a Kubernetes Environment, and have no option to use service connections

In case you can't access your AKS during environment creation time, you can use an empty environment and set the `connectionType` input to an Azure Resource Manager service connection.

I have AKS configured with Azure Active Directory RBAC, and my pipeline doesn't work. Will these updates resolve that?

Accessing Kubernetes when AAD RBAC is enabled is unrelated to token creation. To prevent an interactive prompt, we will support `kubelogin` in a future update.

Use a Kubernetes manifest task in a build or release pipeline to bake and deploy manifests to Kubernetes clusters.

This task supports the following:

- **Artifact substitution:** The deployment action takes as input a list of container images that you can specify along with their tags and digests. The same input is substituted into the nontemplatized manifest files before application to the cluster. This substitution ensures that the cluster nodes pull the right version of the image.

- **Manifest stability:** The rollout status of the deployed Kubernetes objects is checked. The stability checks are incorporated to determine whether the task status is a success or a failure.
- **Traceability annotations:** Annotations are added to the deployed Kubernetes objects to superimpose traceability information. The following annotations are supported:
 - azure-pipelines/org
 - azure-pipelines/project
 - azure-pipelines/pipeline
 - azure-pipelines/pipelineld
 - azure-pipelines/execution
 - azure-pipelines/executionuri
 - azure-pipelines/jobName
- **Secret handling:** The `createSecret` action lets Docker registry secrets be created using Docker registry service connections. It also lets generic secrets be created using either plain-text variables or secret variables. Before deployment to the cluster, you can use the `secrets` input along with the `deploy` action to augment the input manifest files with the appropriate `imagePullSecrets` value.
- **Bake manifest:** The `bake` action of the task allows for baking templates into Kubernetes manifest files. The action uses tools such as Helm, Compose, and Kustomize. With baking, these Kubernetes manifest files are usable for deployments to the cluster.
- **Deployment strategy:** Choosing the `canary` strategy with the `deploy` action leads to the creation of workload names suffixed with `-baseline` and `-canary`. The task supports two methods of traffic splitting:
 - **Service Mesh Interface:** [Service Mesh Interface](#) (SMI) abstraction allows configuration with service mesh providers like `Linkerd` and `Istio`. The Kubernetes Manifest task maps SMI `TrafficSplit` objects to the stable, baseline, and canary services during the life cycle of the deployment strategy.

Canary deployments that are based on a service mesh and use this task are more accurate. This accuracy is due to how service mesh providers enable the granular percentage-based split of traffic. The service mesh uses the service registry and sidecar containers that are injected into pods. This injection occurs alongside application containers to achieve the granular traffic split.

- **Kubernetes with no service mesh:** In the absence of a service mesh, you might not get the exact percentage split you want at the request level. However, you can do canary deployments by using baseline and canary variants next to the stable variant.

The service sends requests to pods of all three workload variants as the selector-label constraints are met. Kubernetes Manifest honors these requests when creating baseline and canary variants. This routing behavior achieves the intended effect of routing only a portion of total requests to the canary.

Compare the baseline and canary workloads by using either a [Manual Intervention task](#) in release pipelines or a [Delay task](#) in YAML pipelines. Do the comparison before using the promote or reject action of the task.

Deploy action

The following YAML code is an example of deploying to a Kubernetes namespace by using manifest files:

YAML

```
steps:
- task: KubernetesManifest@0
  displayName: Deploy
  inputs:
    kubernetesServiceConnection: someK8sSC1
    namespace: default
    manifests: |
      manifests/deployment.yml
      manifests/service.yml
    containers: |
      foo/demo:${tagVariable1}
      bar/demo:${tagVariable2}
    imagePullSecrets: |
      some-secret
      some-other-secret
```

In the above example, the task tries to find matches for the images `foo/demo` and `bar/demo` in the image fields of manifest files. For each match found, the value of either `tagVariable1` or `tagVariable2` is appended as a tag to the image name. You can also specify digests in the containers input for artifact substitution.

 **Note**

While you can author `deploy`, `promote`, and `reject` actions with YAML input related to deployment strategy, support for a Manual Intervention task is currently unavailable for build pipelines.

For release pipelines, we advise you to use actions and input related to deployment strategy in the following sequence:

1. A deploy action specified with `strategy: canary` and `percentage: $(someValue)`.
2. A Manual Intervention task so that you can pause the pipeline and compare the baseline variant with the canary variant.
3. A promote action that runs if a Manual Intervention task is resumed and a reject action that runs if a Manual Intervention task is rejected.

Create secret action

The following YAML code shows a sample creation of Docker registry secrets by using [Docker Registry service connection](#):

YAML

```
steps:  
- task: KubernetesManifest@0  
  displayName: Create secret  
  inputs:  
    action: createSecret  
    secretType: dockerRegistry  
    secretName: foobar  
    dockerRegistryEndpoint: demoACR  
    kubernetesServiceConnection: someK8sSC  
    namespace: default
```

This YAML code shows a sample creation of generic secrets:

YAML

```
steps:  
- task: KubernetesManifest@0  
  displayName: Create secret  
  inputs:  
    action: createSecret  
    secretType: generic  
    secretName: some-secret  
    secretArguments: --from-literal=key1=value1
```

```
kubernetesServiceConnection: someK8sSC
namespace: default
```

Bake action

The following YAML code is an example of baking manifest files from Helm charts. Note the usage of a name input in the first task. This name is later referenced from the deploy step for specifying the path to the manifests that were produced by the bake step.

YAML

```
steps:
- task: KubernetesManifest@0
  name: bake
  displayName: Bake K8s manifests from Helm chart
  inputs:
    action: bake
    helmChart: charts/sample
    overrides: 'image.repository:nginx'

- task: KubernetesManifest@0
  displayName: Deploy K8s manifests
  inputs:
    kubernetesServiceConnection: someK8sSC
    namespace: default
    manifests: $(bake.manifestsBundle)
    containers: |
      nginx: 1.7.9
```

ⓘ Note

To use Helm directly for managing releases and rollbacks, see the [Package and deploy Helm charts task](#).

Kustomize example

The following YAML code is an example of baking manifest files generated with Kustomize that contain a `kustomization.yaml` file.

YAML

```
steps:
- task: KubernetesManifest@0
  name: bake
  displayName: Bake K8s manifests from kustomization path
```

```
inputs:
  action: bake
  renderType: kustomize
  kustomizationPath: folderContainingKustomizationFile

- task: KubernetesManifest@0
  displayName: Deploy K8s manifests
  inputs:
    kubernetesServiceConnection: k8sSC1
    manifests: $(bake.manifestsBundle)
```

Kompose example

The following YAML code is an example of baking manifest files generated with Kompose, a conversion tool for Docker Compose.

YAML

```
steps:
- task: KubernetesManifest@0
  name: bake
  displayName: Bake K8s manifests from Docker Compose
  inputs:
    action: bake
    renderType: kompose
    dockerComposeFile: docker-compose.yaml

- task: KubernetesManifest@0
  displayName: Deploy K8s manifests
  inputs:
    kubernetesServiceConnection: k8sSC1
    manifests: $(bake.manifestsBundle)
```

Scale action

The following YAML code shows an example of scaling objects:

YAML

```
steps:
- task: KubernetesManifest@0
  displayName: Scale
  inputs:
    action: scale
    kind: deployment
    name: bootcamp-demo
    replicas: 5
```

```
kubernetesServiceConnection: someK8sSC
namespace: default
```

Patch action

The following YAML code shows an example of object patching:

YAML

```
steps:
- task: KubernetesManifest@0
  displayName: Patch
  inputs:
    action: patch
    kind: pod
    name: demo-5fbc4d6cd9-pgxn4
    mergeStrategy: strategic
    patch: '{"spec":{"containers":'
[{"name":"demo","image":"foobar/demo:2239"}]}]}
    kubernetesServiceConnection: someK8sSC
    namespace: default
```

Delete action

This YAML code shows a sample object deletion:

YAML

```
steps:
- task: KubernetesManifest@0
  displayName: Delete
  inputs:
    action: delete
    arguments: deployment expressapp
    kubernetesServiceConnection: someK8sSC
    namespace: default
```

Troubleshooting

My Kubernetes cluster is behind a firewall and I am using hosted agents. How can I deploy to this cluster?

You can grant hosted agents access through your firewall by allowing the IP addresses for the hosted agents. For more details, see [Agent IP ranges](#).

How do requests work to stable and variant service routes with canary deployments?

The label selector relationship between pods and services in Kubernetes allows for setting up deployments so that a single service routes requests to both the stable and the canary variants. The Kubernetes manifest task uses this for canary deployments.

If the task includes the inputs of `action: deploy` and `strategy: canary`, for each workload (Deployment, ReplicaSet, Pod, ...) defined in the input manifest files, a `-baseline` and `-canary` variant of the deployment are created. In this example, there's a deployment `sampleapp` in the input manifest file and that after completion of run number 22 of the pipeline, the stable variant of this deployment named `sampleapp` is deployed in the cluster. In the subsequent run (in this case run number 23), Kubernetes manifest task with `action: deploy` and `strategy: canary` would result in creation of `sampleapp-baseline` and `sampleapp-canary` deployments whose number of replicas are determined by the product of `percentage` task input with the value of the desired number of replicas for the final stable variant of `sampleapp` as per the input manifest files.

Excluding the number of replicas, the baseline version has the same configuration as the stable variant while the canary version has the new changes that are being introduced by the current run (in this case, run number 23). If a manual intervention is set up in the pipeline after the above mentioned step, it would allow for an opportunity to pause the pipeline so that the pipeline admin can evaluate key metrics for the baseline and canary versions and take the decision on whether the canary changes are safe and good enough for a complete rollout.

The `action: promote` and `strategy: canary` or `action: reject` and `strategy: canary` inputs of the Kubernetes manifest tasks can be used to promote or reject the canary changes respectively. Note that in either cases, at the end of this step, only the stable variant of the workloads declared in the input manifest files will be remain deployed in the cluster, while the ephemeral baseline and canary versions are cleaned up.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None

Requirement	Description
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Deploy

KubernetesManifest@0 - Deploy to Kubernetes v0 task

Article • 09/26/2023

Use a Kubernetes manifest task in a build or release pipeline to bake and deploy manifests to Kubernetes clusters using Helm charts.

Syntax

YAML

```
# Deploy to Kubernetes v0
# Use Kubernetes manifest files to deploy to clusters or even bake the
manifest files to be used for deployments using Helm charts.
- task: KubernetesManifest@0
  inputs:
    #action: 'deploy' # 'bake' | 'createSecret' | 'delete' | 'deploy' |
    'patch' | 'promote' | 'scale' | 'reject'. Action. Default: deploy.
    #kubernetesServiceConnection: # string. Required when action != bake.
    Kubernetes service connection.
    #namespace: # string. Namespace.
    #strategy: 'none' # 'canary' | 'none'. Optional. Use when action =
    deploy || action = promote || action = reject. Strategy. Default: none.
    #trafficSplitMethod: 'pod' # 'pod' | 'smi'. Optional. Use when strategy
    = canary. Traffic split method. Default: pod.
    #percentage: '0' # string. Required when strategy = Canary && action =
    deploy. Percentage. Default: 0.
    #baselineAndCanaryReplicas: '1' # string. Required when strategy =
    Canary && action = deploy && trafficSplitMethod = SMI. Baseline and canary
    replicas. Default: 1.
    #manifests: # string. Required when action = deploy || action = promote
    || action = reject. Manifests.
    #containers: # string. Optional. Use when action = deploy || action =
    promote || action = bake. Containers.
    #imagePullSecrets: # string. Optional. Use when action = deploy ||
    action = promote. ImagePullSecrets.
    #renderType: 'helm' # 'helm' | 'kompose' | 'kustomize'. Optional. Use
    when action = bake. Render Engine. Default: helm.
    #dockerComposeFile: # string. Required when action = bake && renderType
    = kompose. Path to docker compose file.
    #helmChart: # string. Required when action = bake && renderType = helm.
    Helm Chart.
    #releaseName: # string. Optional. Use when action = bake && renderType =
    helm. Helm Release Name.
    #overrideFiles: # string. Optional. Use when action = bake && renderType
    = helm. Override Files.
    #overrides: # string. Optional. Use when action = bake && renderType =
    helm. Overrides.
```

```
#kustomizationPath: # string. Optional. Use when action = bake &&
renderType = kustomize. Kustomization Path.
#resourceToPatch: 'file' # 'file' | 'name'. Required when action =
patch. Resource to patch. Default: file.
#resourceFileToPatch: # string. Required when action = patch &&
resourceToPatch = file. File path.
#kind: # 'deployment' | 'replicaset' | 'statefulset'. Required when
action = scale || resourceToPatch = name. Kind.
#name: # string. Required when action = scale || resourceToPatch = name.
Name.
#replicas: # string. Required when action = scale. Replica count.
#mergeStrategy: 'strategic' # 'json' | 'merge' | 'strategic'. Required
when action = patch. Merge Strategy. Default: strategic.
#arguments: # string. Optional. Use when action = delete. Arguments.
#patch: # string. Required when action = patch. Patch.
#secretType: 'dockerRegistry' # 'dockerRegistry' | 'generic'. Required
when action = createSecret. Type of secret. Default: dockerRegistry.
#secretName: # string. Optional. Use when action = createSecret. Secret
name.
#secretArguments: # string. Optional. Use when action = createSecret &&
secretType = generic. Arguments.
#dockerRegistryEndpoint: # string. Optional. Use when action =
createSecret && secretType = dockerRegistry. Docker registry service
connection.
#rolloutStatusTimeout: '0' # string. Optional. Use when action = deploy
|| action = patch || action = scale || action = promote. Timeout for rollout
status. Default: 0.
```

Inputs

`action` - Action

`string`. Allowed values: `bake`, `createSecret` (create secret), `delete`, `deploy`, `patch`, `promote`, `scale`, `reject`. Default value: `deploy`.

Specifies the action to be performed.

`kubernetesServiceConnection` - Kubernetes service connection

`string`. Required when `action != bake`.

Specifies a [Kubernetes service connection](#).

`namespace` - Namespace

`string`.

Specifies the namespace for the commands by using the `-namespace` flag. If the namespace is not provided, the commands will run in the default namespace.

strategy - Strategy

`string`. Optional. Use when `action = deploy || action = promote || action = reject`.

Allowed values: `canary`, `none`. Default value: `none`.

Specifies the deployment strategy used in the `deploy` action before a `promote` action or `reject` action. Currently, `canary` is the only acceptable deployment strategy.

trafficSplitMethod - Traffic split method

`string`. Optional. Use when `strategy = canary`. Allowed values: `pod`, `smi`. Default value: `pod`.

For the value `smi`, the percentage traffic split is done at the request level by using a service mesh. A service mesh must be set up by a cluster admin. This task handles orchestration of SMI [TrafficSplit](#) objects.

For the value `pod`, the percentage split isn't possible at the request level in the absence of a service mesh. Instead, the percentage input is used to calculate the replicas for baseline and canary. The calculation is a percentage of replicas that are specified in the input manifests for the stable variant.

percentage - Percentage

`string`. Required when `strategy = Canary && action = deploy`. Default value: `0`.

The percentage that is used to compute the number of baseline-variant and canary-variant replicas of the workloads that are contained in manifest files.

For the specified percentage input, calculate:

$$(percentage \times number\ of\ replicas) / 100$$

If the result isn't an integer, the mathematical floor of the result is used when baseline and canary variants are created.

For example, assume the deployment `hello-world` is in the input manifest file and that the following lines are in the task input:

```
replicas: 4
strategy: canary
```

```
percentage: 25
```

In this case, the deployments `hello-world-baseline` and `hello-world-canary` are created with one replica each. The baseline variant is created with the same image and tag as the stable version, which is the four-replica variant before deployment. The canary variant is created with the image and tag corresponding to the newly deployed changes.

baselineAndCanaryReplicas - Baseline and canary replicas

`string`. Required when `strategy = Canary && action = deploy && trafficSplitMethod = SMI`. Default value: `1`.

When you set `trafficSplitMethod` to `smi`, the percentage traffic split is controlled in the service mesh plane. You can control the actual number of replicas for canary and baseline variants independently of the traffic split.

For example, assume that the input deployment manifest specifies 30 replicas for the stable variant. Also assume that you specify the following input for the task:

```
strategy: canary
trafficSplitMethod: smi
percentage: 20
baselineAndCanaryReplicas: 1
```

In this case, the stable variant receives 80% of the traffic, while the baseline and canary variants each receive half of the specified 20%. Baseline and canary variants don't receive three replicas each. They instead receive the specified number of replicas, which means they each receive one replica.

manifests - Manifests

`string`. Required when `action = deploy || action = promote || action = reject`.

Specifies the path to the manifest files to be used for deployment. Each line represents a single path. A [file-matching pattern](#) is an acceptable value for each line.

containers - Containers

`string`. Optional. Use when `action = deploy || action = promote || action = bake`.

Specifies the fully qualified resource URL of the image to be used for substitutions on the manifest files. The URL `contosodemo.azurecr.io/helloworld:test` is an example.

`imagePullSecrets` - ImagePullSecrets

`string`. Optional. Use when `action = deploy || action = promote`.

Specifies a multiline input where each line contains the name of a Docker registry secret that has already been set up within the cluster. Each secret name is added under `imagePullSecrets` for the workloads that are found in the input manifest files.

`renderType` - Render Engine

`string`. Optional. Use when `action = bake`. Allowed values: `helm`, `kompose`, `kustomize`.

Default value: `helm`.

Specifies the render type used to produce the manifest files.

`dockerComposeFile` - Path to docker compose file

`string`. Required when `action = bake && renderType = kompose`.

Specifies a docker-compose file path.

`helmChart` - Helm Chart

`string`. Required when `action = bake && renderType = helm`.

Specifies the Helm chart path to bake.

`releaseName` - Helm Release Name

`string`. Optional. Use when `action = bake && renderType = helm`.

Specifies the Helm release name to use.

`overrideFiles` - Override Files

`string`. Optional. Use when `action = bake && renderType = helm`.

Specifies a multiline input that accepts the path to the override files. The files are used when manifest files from Helm charts are baked.

overrides - Overrides

`string`. Optional. Use when `action = bake && renderType = helm`.

Specifies the override values to set.

kustomizationPath - Kustomization Path

`string`. Optional. Use when `action = bake && renderType = kustomize`.

Specifies the argument that must be the path to the directory containing the file, or a git repository URL with a path suffix specifying `same` with respect to the repository root.

resourceToPatch - Resource to patch

`string`. Required when `action = patch`. Allowed values: `file`, `name`. Default value: `file`.

Indicates one of the following patch methods:

- A manifest file identifies the objects to be patched.
- An individual object is identified by kind and name as the patch target.

Acceptable values are `file` and `name`.

resourceFileToPatch - File path

`string`. Required when `action = patch && resourceToPatch = file`.

Specifies the path to the file used for a patch.

kind - Kind

`string`. Required when `action = scale || resourceToPatch = name`. Allowed values: `deployment`, `replicaset`, `statefulset`.

Specifies the kind of K8s object, such as `deployment`, `replicaSet` and more.

name - Name

`string`. Required when `action = scale || resourceToPatch = name`.

Specifies the name of the K8s object.

replicas - Replica count

`string`. Required when `action = scale`.

Specifies the number of replicas to scale to.

mergeStrategy - Merge Strategy

`string`. Required when `action = patch`. Allowed values: `json`, `merge`, `strategic`.

Default value: `strategic`.

Specifies the type of patch being provided.

arguments - Arguments

`string`. Optional. Use when `action = delete`.

Specifies the arguments for the `kubectl delete` command. An example is: `arguments:`

```
deployment hello-world foo-bar
```

patch - Patch

`string`. Required when `action = patch`.

Specifies the contents of the patch.

secretType - Type of secret

`string`. Required when `action = createSecret`. Allowed values: `dockerRegistry`,

`generic`. Default value: `dockerRegistry`.

Creates or updates a generic or docker `imagepullsecret`. Specify `dockerRegistry` to create or update the `imagepullsecret` of the selected registry. An `imagePullSecret` is a way to pass a secret that contains a container registry password to the Kubelet, so it can pull a private image on behalf of your Pod.

secretName - Secret name

`string`. Optional. Use when `action = createSecret`.

Specifies the name of the secret. You can use this secret name in the Kubernetes YAML configuration file.

`secretArguments` - Arguments

`string`. Optional. Use when `action = createSecret && secretType = generic`.

Specifies keys and literal values to insert in secret. For example, `--from-literal=key1=value1 --from-literal=key2="top secret"`.

`dockerRegistryEndpoint` - Docker registry service connection

`string`. Optional. Use when `action = createSecret && secretType = dockerRegistry`.

Specifies the credentials of the specified service connection that are used to create a Docker registry secret within the cluster. Manifest files under the `imagePullSecrets` field can then refer to this secret's name.

`rolloutStatusTimeout` - Timeout for rollout status

`string`. Optional. Use when `action = deploy || action = patch || action = scale || action = promote`. Default value: `0`.

Specifies the length of time (in seconds) to wait before ending `watch` on `rollout` status.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

`manifestsBundle`

Specifies the location of the manifest bundles created by bake action.

Remarks

Note

There is a newer version of this task available that provides additional support for targetting a Kubernetes cluster in different ways, using the `connectionType`

property. For more information, see [KubernetesManifest@1](#) and [KubernetesManifest@1 service connection remarks](#)

Use a Kubernetes manifest task in a build or release pipeline to bake and deploy manifests to Kubernetes clusters.

This task supports the following:

- **Artifact substitution:** The deployment action takes as input a list of container images that you can specify along with their tags and digests. The same input is substituted into the nontemplatized manifest files before application to the cluster. This substitution ensures that the cluster nodes pull the right version of the image.
- **Manifest stability:** The rollout status of the deployed Kubernetes objects is checked. The stability checks are incorporated to determine whether the task status is a success or a failure.
- **Traceability annotations:** Annotations are added to the deployed Kubernetes objects to superimpose traceability information. The following annotations are supported:
 - azure-pipelines/org
 - azure-pipelines/project
 - azure-pipelines/pipeline
 - azure-pipelines/pipelinelid
 - azure-pipelines/execution
 - azure-pipelines/executionuri
 - azure-pipelines/jobName
- **Secret handling:** The `createSecret` action lets Docker registry secrets be created using Docker registry service connections. It also lets generic secrets be created using either plain-text variables or secret variables. Before deployment to the cluster, you can use the `secrets` input along with the `deploy` action to augment the input manifest files with the appropriate `imagePullSecrets` value.
- **Bake manifest:** The `bake` action of the task allows for baking templates into Kubernetes manifest files. The action uses tools such as Helm, Compose, and Kustomize. With baking, these Kubernetes manifest files are usable for deployments to the cluster.
- **Deployment strategy:** Choosing the `canary` strategy with the `deploy` action leads to the creation of workload names suffixed with `-baseline` and `-canary`. The task supports two methods of traffic splitting:

- **Service Mesh Interface:** [Service Mesh Interface](#) (SMI) abstraction allows configuration with service mesh providers like [Linkerd](#) and [Istio](#). The Kubernetes Manifest task maps SMI [TrafficSplit](#) objects to the stable, baseline, and canary services during the life cycle of the deployment strategy.

Canary deployments that are based on a service mesh and use this task are more accurate. This accuracy is due to how service mesh providers enable the granular percentage-based split of traffic. The service mesh uses the service registry and sidecar containers that are injected into pods. This injection occurs alongside application containers to achieve the granular traffic split.

- **Kubernetes with no service mesh:** In the absence of a service mesh, you might not get the exact percentage split you want at the request level. However, you can do canary deployments by using baseline and canary variants next to the stable variant.

The service sends requests to pods of all three workload variants as the selector-label constraints are met. Kubernetes Manifest honors these requests when creating baseline and canary variants. This routing behavior achieves the intended effect of routing only a portion of total requests to the canary.

Compare the baseline and canary workloads by using either a [Manual Intervention task](#) in release pipelines or a [Delay task](#) in YAML pipelines. Do the comparison before using the promote or reject action of the task.

Deploy action

The following YAML code is an example of deploying to a Kubernetes namespace by using manifest files:

YAML

```
steps:
- task: KubernetesManifest@0
  displayName: Deploy
  inputs:
    kubernetesServiceConnection: someK8sSC1
    namespace: default
    manifests: |
      manifests/deployment.yml
      manifests/service.yml
    containers: |
      foo/demo:${tagVariable1}
      bar/demo:${tagVariable2}
    imagePullSecrets: |
```

```
some-secret  
some-other-secret
```

In the above example, the task tries to find matches for the images `foo/demo` and `bar/demo` in the image fields of manifest files. For each match found, the value of either `tagVariable1` or `tagVariable2` is appended as a tag to the image name. You can also specify digests in the containers input for artifact substitution.

ⓘ Note

While you can author `deploy`, `promote`, and `reject` actions with YAML input related to deployment strategy, support for a Manual Intervention task is currently unavailable for build pipelines.

For release pipelines, we advise you to use actions and input related to deployment strategy in the following sequence:

1. A deploy action specified with `strategy: canary` and `percentage: $(someValue)`.
2. A Manual Intervention task so that you can pause the pipeline and compare the baseline variant with the canary variant.
3. A promote action that runs if a Manual Intervention task is resumed and a reject action that runs if a Manual Intervention task is rejected.

Create secret action

The following YAML code shows a sample creation of Docker registry secrets by using [Docker Registry service connection](#):

YAML

```
steps:  
- task: KubernetesManifest@0  
  displayName: Create secret  
  inputs:  
    action: createSecret  
    secretType: dockerRegistry  
    secretName: foobar  
    dockerRegistryEndpoint: demoACR  
    kubernetesServiceConnection: someK8sSC  
    namespace: default
```

This YAML code shows a sample creation of generic secrets:

YAML

```
steps:
- task: KubernetesManifest@0
  displayName: Create secret
  inputs:
    action: createSecret
    secretType: generic
    secretName: some-secret
    secretArguments: --from-literal=key1=value1
    kubernetesServiceConnection: someK8sSC
    namespace: default
```

Bake action

The following YAML code is an example of baking manifest files from Helm charts. Note the usage of a name input in the first task. This name is later referenced from the deploy step for specifying the path to the manifests that were produced by the bake step.

YAML

```
steps:
- task: KubernetesManifest@0
  name: bake
  displayName: Bake K8s manifests from Helm chart
  inputs:
    action: bake
    helmChart: charts/sample
    overrides: 'image.repository:nginx'

- task: KubernetesManifest@0
  displayName: Deploy K8s manifests
  inputs:
    kubernetesServiceConnection: someK8sSC
    namespace: default
    manifests: $(bake.manifestsBundle)
    containers: |
      nginx: 1.7.9
```

ⓘ Note

To use Helm directly for managing releases and rollbacks, see the [Package and deploy Helm charts task](#).

Kustomize example

The following YAML code is an example of baking manifest files generated with Kustomize that contain a `kustomization.yaml` file.

YAML

```
steps:
- task: KubernetesManifest@0
  name: bake
  displayName: Bake K8s manifests from kustomization path
  inputs:
    action: bake
    renderType: kustomize
    kustomizationPath: folderContainingKustomizationFile

- task: KubernetesManifest@0
  displayName: Deploy K8s manifests
  inputs:
    kubernetesServiceConnection: k8sSC1
    manifests: $(bake.manifestsBundle)
```

Kompose example

The following YAML code is an example of baking manifest files generated with Kompose, a conversion tool for Docker Compose.

YAML

```
steps:
- task: KubernetesManifest@0
  name: bake
  displayName: Bake K8s manifests from Docker Compose
  inputs:
    action: bake
    renderType: kompose
    dockerComposeFile: docker-compose.yaml

- task: KubernetesManifest@0
  displayName: Deploy K8s manifests
  inputs:
    kubernetesServiceConnection: k8sSC1
    manifests: $(bake.manifestsBundle)
```

Scale action

The following YAML code shows an example of scaling objects:

YAML

```
steps:
- task: KubernetesManifest@0
  displayName: Scale
  inputs:
    action: scale
    kind: deployment
    name: bootcamp-demo
    replicas: 5
    kubernetesServiceConnection: someK8sSC
    namespace: default
```

Patch action

The following YAML code shows an example of object patching:

YAML

```
steps:
- task: KubernetesManifest@0
  displayName: Patch
  inputs:
    action: patch
    kind: pod
    name: demo-5fbc4d6cd9-pgxn4
    mergeStrategy: strategic
    patch: '{"spec":{"containers": [{"name":"demo","image":"foobar/demo:2239"}]}}'
    kubernetesServiceConnection: someK8sSC
    namespace: default
```

Delete action

This YAML code shows a sample object deletion:

YAML

```
steps:
- task: KubernetesManifest@0
  displayName: Delete
  inputs:
    action: delete
    arguments: deployment expressapp
    kubernetesServiceConnection: someK8sSC
    namespace: default
```

Troubleshooting

My Kubernetes cluster is behind a firewall and I am using hosted agents. How can I deploy to this cluster?

You can grant hosted agents access through your firewall by allowing the IP addresses for the hosted agents. For more details, see [Agent IP ranges](#).

How do requests work to stable and variant service routes with canary deployments?

The label selector relationship between pods and services in Kubernetes allows for setting up deployments so that a single service routes requests to both the stable and the canary variants. The Kubernetes manifest task uses this for canary deployments.

If the task includes the inputs of `action: deploy` and `strategy: canary`, for each workload (Deployment, ReplicaSet, Pod, ...) defined in the input manifest files, a `-baseline` and `-canary` variant of the deployment are created. In this example, there's a deployment `sampleapp` in the input manifest file and that after completion of run number 22 of the pipeline, the stable variant of this deployment named `sampleapp` is deployed in the cluster. In the subsequent run (in this case run number 23), Kubernetes manifest task with `action: deploy` and `strategy: canary` would result in creation of `sampleapp-baseline` and `sampleapp-canary` deployments whose number of replicas are determined by the product of `percentage` task input with the value of the desired number of replicas for the final stable variant of `sampleapp` as per the input manifest files.

Excluding the number of replicas, the baseline version has the same configuration as the stable variant while the canary version has the new changes that are being introduced by the current run (in this case, run number 23). If a manual intervention is set up in the pipeline after the above mentioned step, it would allow for an opportunity to pause the pipeline so that the pipeline admin can evaluate key metrics for the baseline and canary versions and take the decision on whether the canary changes are safe and good enough for a complete rollout.

The `action: promote` and `strategy: canary` or `action: reject` and `strategy: canary` inputs of the Kubernetes manifest tasks can be used to promote or reject the canary changes respectively. Note that in either cases, at the end of this step, only the stable variant of the workloads declared in the input manifest files will be remain deployed in the cluster, while the ephemeral baseline and canary versions are cleaned up.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Deploy

Docker@2 - Docker v2 task

Article • 09/26/2023

Build or push Docker images, log in or log out, start or stop containers, or run a Docker command.

Syntax

YAML

```
# Docker v2
# Build or push Docker images, login or logout, start or stop containers, or
run a Docker command.
- task: Docker@2
  inputs:
    # Container Repository
    #containerRegistry: # string. Container registry.
    #repository: # string. Optional. Use when command != login && command !=
logout && command != start && command != stop. Container repository.
    # Commands
    command: 'buildAndPush' # 'buildAndPush' | 'build' | 'push' | 'login' |
'logout' | 'start' | 'stop'. Required. Command. Default: buildAndPush.
    Dockerfile: '**/Dockerfile' # string. Required when command = build ||
command = buildAndPush. Dockerfile. Default: **/Dockerfile.
    #buildContext: '**' # string. Optional. Use when command = build ||
command = buildAndPush. Build context. Default: **.
    #tags: '$(Build.BuildId)' # string. Optional. Use when command = build ||
command = push || command = buildAndPush. Tags. Default:
$(Build.BuildId).
    #arguments: # string. Optional. Use when command != login && command !=
logout && command != buildAndPush. Arguments.
    #addPipelineData: true # boolean. Add Pipeline metadata to image(s).
Default: true.
    #addBaseImageData: true # boolean. Add base image metadata to image(s).
Default: true.
    #container: # string. Optional. Use when command = start || command =
stop. Container.
```

Inputs

`containerRegistry` - Container registry

`string`.

Name of the [Docker registry service connection](#). Required for commands that perform authentication with a registry.

repository - Container repository

`string`. Optional. Use when `command != login && command != logout && command != start && command != stop`.

Specifies the name of the repository.

command - Command

`string`. Required. Allowed values: `buildAndPush`, `build`, `push`, `login`, `logout`, `start`, `stop`. Default value: `buildAndPush`.

Specifies the Docker command to run.

Dockerfile - Dockerfile

`string`. Required when `command = build || command = buildAndPush`. Default value: `**/Dockerfile`.

Specifies the path to the Docker file. The task uses the first Docker file it finds to build the image.

buildContext - Build context

`string`. Optional. Use when `command = build || command = buildAndPush`. Default value: `**`.

Specifies the path to the build context. Pass `**` to indicate the directory that contains the Docker file.

tags - Tags

`string`. Optional. Use when `command = build || command = push || command = buildAndPush`. Default value: `$(Build.BuildId)`.

Specifies a list of tags on separate lines. These tags are used in `build`, `push` and `buildAndPush` commands.

arguments - Arguments

`string`. Optional. Use when `command != login && command != logout && command != buildAndPush`.

Specifies additional arguments to pass to the Docker client. If using the value `buildAndPush` for the command parameter, the arguments property is ignored.

Example: Using the build command, `--build-arg HTTP_PROXY=http://10.20.30.2:1234 --quiet`.

`addPipelineData` - Add Pipeline metadata to image(s)

`boolean`. Default value: `true`.

By default, pipeline data like source branch name, or build ID are added and help with traceability. For example, you can inspect an image to find out which pipeline built the image. You can opt out of this default behavior.

`addBaseImageData` - Add base image metadata to image(s)

`boolean`. Default value: `true`.

By default, base image data like base image name, or digest are added and help with traceability. You can opt out of this default behavior.

`container` - Container

`string`. Optional. Use when `command = start || command = stop`.

Specifies the name of the container resource to start or stop. Use this command with `start` and `stop` commands.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

`DockerOutput`

Specifies the path to the files that contain the command output. You can list two file

paths on separate lines for the `buildAndPush` command, and one file path for any other command.

Remarks

The following are the key benefits of using the Docker task instead of directly using Docker client binary in a script.

- **Integration with Docker registry service connection** - The task makes it easy to use a Docker registry service connection for connecting to any container registry. Once signed in, you can add follow up tasks that execute other tasks or scripts by leveraging the sign on used by the Docker task. For example, use the Docker task to sign in to any Azure Container Registry, and then use another task or script to build and push an image to the registry.
- **Metadata added as labels** - The task adds traceability-related metadata to the image in the following labels -
 - com.azure.dev.image.build.buildnumber
 - com.azure.dev.image.build.builduri
 - com.azure.dev.image.build.definitionname
 - com.azure.dev.image.build.repository.name
 - com.azure.dev.image.build.repository.uri
 - com.azure.dev.image.build.sourcebranchname
 - com.azure.dev.image.build.sourceversion
 - com.azure.dev.image.release.definitionname
 - com.azure.dev.image.release.releaseid
 - com.azure.dev.image.release.releaseweburl
 - com.azure.dev.image.system.teamfoundationcollectionuri
 - com.azure.dev.image.system.teamproject

Troubleshooting

Why does the Docker task ignore arguments passed to the `buildAndPush` command?

A Docker task configured using the `buildAndPush` command ignores the arguments passed because they become ambiguous to the internal build and push commands. You can split your command into separate build and push steps and pass the suitable arguments. For example, see this [stackoverflow post](#).

DockerV2 only supports Docker registry service connection and not support ARM service connection. How can I use an existing Azure service principal (SPN) for authentication in Docker task?

You can create a Docker registry service connection using your Azure SPN credentials. Choose the others from Registry type and provide the details as follows:

```
Docker Registry: Your container registry URL (eg.  
https://myacr.azurecr.io)  
Docker ID: Service principal client ID  
Password: Service principal key
```

Examples

Login

YAML

The following YAML snippet shows a container registry sign on using a Docker registry service connection.

YAML

```
- task: Docker@2  
displayName: Login to ACR  
inputs:  
  command: login  
  containerRegistry: dockerRegistryServiceConnection1
```

Build and Push

A convenience command called `buildAndPush` allows the build and push of images to a container registry in a single command.

YAML

The following YAML snippet is an example of building and pushing multiple tags of an image to multiple registries.

YAML

```
steps:
- task: Docker@2
  displayName: Login to ACR
  inputs:
    command: login
    containerRegistry: dockerRegistryServiceConnection1
- task: Docker@2
  displayName: Login to Docker Hub
  inputs:
    command: login
    containerRegistry: dockerRegistryServiceConnection2
- task: Docker@2
  displayName: Build and Push
  inputs:
    command: buildAndPush
    repository: contosoRepository # username/contosoRepository for
DockerHub
  tags: |
    tag1
    tag2
```

In the above snippet, the images `contosoRepository:tag1` and `contosoRepository:tag2` are built and pushed to the container registries corresponding to `dockerRegistryServiceConnection1` and `dockerRegistryServiceConnection2`.

If you want to build and push to a specific authenticated container registry instead of building and pushing to all authenticated container registries at once, explicitly specify the `containerRegistry` input with `command: buildAndPush` as shown:

YAML

```
steps:
- task: Docker@2
  displayName: Build and Push
  inputs:
    command: buildAndPush
    containerRegistry: dockerRegistryServiceConnection1
    repository: contosoRepository
  tags: |
    tag1
    tag2
```

Logout

YAML

The following YAML snippet shows how to log out from a container registry using a Docker registry service connection.

YAML

```
- task: Docker@2
  displayName: Logout of ACR
  inputs:
    command: logout
    containerRegistry: dockerRegistryServiceConnection1
```

Start/stop

Use this task to control job and service containers. This usage is uncommon, but occasionally used in unique circumstances.

YAML

```
resources:
  containers:
    - container: builder
      image: ubuntu:18.04
steps:
  - script: echo "I can run inside the container (it starts by default)"
    target:
      container: builder
  - task: Docker@2
    inputs:
      command: stop
      container: builder
# any task beyond this point would not be able to target the builder
# container
# because it's been stopped
```

Other commands and arguments

The command and argument inputs are used to pass additional arguments for build or push commands using Docker client binary as shown in the example.

YAML

```
steps:
  - task: Docker@2
```

```

displayName: Login to ACR
inputs:
  command: login
  containerRegistry: dockerRegistryServiceConnection1
- task: Docker@2
  displayName: Build
  inputs:
    command: build
    repository: contosoRepository # username/contosoRepository for DockerHub
    tags: tag1
  arguments: --secret id=mysecret,src=mysecret.txt

```

ⓘ Note

The arguments input is evaluated for all commands except `buildAndPush`.

`buildAndPush` is a convenience command (`build` followed by `push`), arguments input is ignored when it is used.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.172.0 or greater
Task category	Build

Docker@1 - Docker v1 task

Article • 09/26/2023

Build, tag, push, or run Docker images, or run a Docker command.

Syntax

YAML

```
# Docker v1
# Build, tag, push, or run Docker images, or run a Docker command.
- task: Docker@1
  inputs:
    # Container Registry
    #containerregistrytype: 'Azure Container Registry' # 'Azure Container Registry' | 'Container Registry'. Required when command != logout. Container registry type. Default: Azure Container Registry.
    #dockerRegistryEndpoint: # string. Optional. Use when containerregistrytype = Container Registry && command != logout. Docker registry service connection.
    #azureSubscriptionEndpoint: # string. Optional. Use when containerregistrytype = Azure Container Registry && command != logout. Azure subscription.
    #azureContainerRegistry: # string. Optional. Use when containerregistrytype = Azure Container Registry && command != logout. Azure container registry.
    # Commands
    #addBaseImageData: true # boolean. Add base image metadata to image(s). Default: true.
    command: 'Build an image' # 'Build an image' | 'Tag image' | 'Push an image' | 'Run an image' | 'login' | 'logout'. Required. Command. Default: Build an image.
    #dockerFile: '**/Dockerfile' # string. Required when command = Build an image || command = build. Dockerfile. Default: **/Dockerfile.
    #arguments: # string. Optional. Use when command != login && command != logout. Arguments.
    #pushMultipleImages: false # boolean. Optional. Use when command = Push an image || command = push. Push multiple images. Default: false.
    #tagMultipleImages: false # boolean. Optional. Use when command = Tag image || command = tag. Tag multiple images. Default: false.
    #imageName: '$(Build.Repository.Name):$(Build.BuildId)' # string. Required when command = Build an image || command = build || command = Run an image || command = run || pushMultipleImages = false || tagMultipleImages = false. Image name. Default: $(Build.Repository.Name):$(Build.BuildId).
    #imageNamesPath: # string. Required when tagMultipleImages = true || pushMultipleImages = true. Image names path.
    #qualify imageName: true # boolean. Optional. Use when command = Build an image || command = build || command = Tag image || command = tag || command = Push an image || command = push || command = Run an image || command = run. Qualify image name. Default: true.
```

```

    #qualifySourceImageName: false # boolean. Optional. Use when command = Tag image || command = tag. Qualify source image name. Default: false.
    #includeSourceTags: false # boolean. Optional. Use when command = Build an image || command = build || command = Tag image || command = tag || command = Push an image || command = push. Include source tags. Default: false.
    #includeLatestTag: false # boolean. Optional. Use when command = Build an image || command = build. Include latest tag. Default: false.
    #addDefaultLabels: true # boolean. Optional. Use when addDefaultLabels = false. Add default labels. Default: true.
    #useDefaultContext: true # boolean. Optional. Use when command = Build an image || command = build. Use default build context. Default: true.
    #buildContext: # string. Optional. Use when useDefaultContext = false. Build context.
    #imageDigestFile: # string. Optional. Use when command = Push an image || command = push. Image digest file.
    #containerName: # string. Optional. Use when command = Run an image || command = run. Container name.
    #ports: # string. Optional. Use when command = Run an image || command = run. Ports.
    #volumes: # string. Optional. Use when command = Run an image || command = run. Volumes.
    #envVars: # string. Optional. Use when command = Run an image || command = run. Environment variables.
    #workingDirectory: # string. Optional. Use when command = Run an image || command = run. Working directory.
    #entrypointOverride: # string. Optional. Use when command = Run an image || command = run. Entry point override.
    #containerCommand: # string. Optional. Use when command = Run an image || command = run. Container command.
    #runInBackground: true # boolean. Optional. Use when command = Run an image || command = run. Run in background. Default: true.
    restartPolicy: 'no' # 'no' | 'onFailure' | 'always' | 'unlessStopped'. Required when runInBackground = true. Restart policy. Default: no.
    #maxRestartRetries: # string. Optional. Use when runInBackground = true && restartPolicy = onFailure. Maximum restart retries.
    # Advanced Options
    #dockerHostEndpoint: # string. Optional. Use when command != login && command != logout. Docker host service connection.
    #enforceDockerNamingConvention: true # boolean. Optional. Use when command != login && command != logout. Force image name to follow Docker naming convention. Default: true.
    #memoryLimit: # string. Optional. Use when command != login && command != logout. Memory limit.

```

Inputs

`containerregistrytype` - Container registry type

`string`. Required when `command != logout`. Allowed values: `Azure Container Registry`, `Container Registry`. Default value: `Azure Container Registry`.

Specifies the Azure Container Registry to connect using an Azure Service Connection. Select an Azure Container Registry to connect to a Docker Hub or any other private container registry.

addBaseImageData - Add base image metadata to image(s)

`boolean`. Default value: `true`.

The default value adds base image data, such as the base image name and digest, to help with traceability. You can opt out of this default behavior by setting this value to `false`.

dockerRegistryEndpoint - Docker registry service connection

`string`. Optional. Use when `containerregistrytype = Container Registry && command != logout`.

Specifies a Docker registry service connection. Required for commands that authenticate using a registry.

azureSubscriptionEndpoint - Azure subscription

`string`. Optional. Use when `containerregistrytype = Azure Container Registry && command != logout`.

Specifies an Azure subscription.

azureContainerRegistry - Azure container registry

`string`. Optional. Use when `containerregistrytype = Azure Container Registry && command != logout`.

Specifies an Azure Container Registry in the selected Azure Subscription. The container image is built and pushed to this container registry.

command - Command

`string`. Required. Allowed values: `Build an image` (`build`), `Tag image` (`tag`), `Push an image` (`push`), `Run an image` (`run`), `login`, `logout`. Default value: `Build an image`.

Specifies the docker command to run.

dockerFile - Dockerfile

`string`. Required when `command = Build an image || command = build`. Default value: `**/Dockerfile`.

Specifies the path to the Docker file. The task uses the first docker file it finds to build the image.

arguments - Arguments

`string`. Optional. Use when `command != login && command != logout`.

Specifies additional arguments to pass to the docker client. Using the value `buildAndPush` in the command parameter ignores the arguments property.

pushMultipleImages - Push multiple images

`boolean`. Optional. Use when `command = Push an image || command = push`. Default value: `false`.

Specifies a list in a text file of Docker images to push. List each image name in the format `Imagename1:tag1` on a separate line. Listing an image name without tags, for example `Imagename2`, pushes all tags in the `Imagename2` container.

tagMultipleImages - Tag multiple images

`boolean`. Optional. Use when `command = Tag image || command = tag`. Default value: `false`.

Specifies a list of multiple image tags and Docker images to tag in a text file. List each image name in the format `Imagename1:tag1` on a separate line. Images listed without a tag as `Imagename2` are tagged as *latest* by default.

imageName - Image name

`string`. Required when `command = Build an image || command = build || command = Run an image || command = run || pushMultipleImages = false || tagMultipleImages = false`. Default value: `$(Build.Repository.Name):$(Build.BuildId)`.

Specifies the name of the Docker image to build, push, or run.

imageNamesPath - Image names path

`string`. Required when `tagMultipleImages = true || pushMultipleImages = true`.

Specifies the path to a text file that contains the names of the Docker images to tag or push. List each image name on a separate line.

qualifyImageName - Qualify image name

`boolean`. Optional. Use when `command = Build an image || command = build || command = Tag image || command = tag || command = Push an image || command = push || command = Run an image || command = run`. Default value: `true`.

Specifies a qualify image name with the Docker registry service connection's hostname.

qualifySourceImageName - Qualify source image name

`boolean`. Optional. Use when `command = Tag image || command = tag`. Default value: `false`.

Specifies a qualify image name with the Docker registry service connection's hostname.

includeSourceTags - Include source tags

`boolean`. Optional. Use when `command = Build an image || command = build || command = Tag image || command = tag || command = Push an image || command = push`. Default value: `false`.

Specifies Git tags to include when building or pushing the Docker image.

includeLatestTag - Include latest tag

`boolean`. Optional. Use when `command = Build an image || command = build`. Default value: `false`.

Specifies whether to use the *latest* tag when building the Docker image.

addDefaultLabels - Add default labels

`boolean`. Optional. Use when `addDefaultLabels = false`. Default value: `true`.

Specifies whether to add CI/CD metadata to the container image by using Docker labels, like repository, commit, build and release information.

useDefaultContext - Use default build context

`boolean`. Optional. Use when `command = Build an image || command = build`. Default

`value: true`.

Specifies adding or removing build context to the directory that contains the Docker file.

`buildContext` - Build context

`string`. Optional. Use when `useDefaultContext = false`.

Specifies the path to the build context.

`imageDigestFile` - Image digest file

`string`. Optional. Use when `command = Push an image || command = push`.

Specifies the path to a file that is created and populated with the full image repository digest of the Docker image that was pushed.

`containerName` - Container name

`string`. Optional. Use when `command = Run an image || command = run`.

Specifies the name of the Docker container to run.

`ports` - Ports

`string`. Optional. Use when `command = Run an image || command = run`.

Specifies the ports in the Docker container to publish to the host. List each `host-port:container-port` binding on a separate line.

`volumes` - Volumes

`string`. Optional. Use when `command = Run an image || command = run`.

Specifies the volumes to mount from the host. List each `host-dir:container-dir` on a separate line.

`envVars` - Environment variables

`string`. Optional. Use when `command = Run an image || command = run`.

Specifies environment variables for the Docker container. List each `name=value` pair on a separate line.

workingDirectory - Working directory

`string`. Optional. Use when `command = Run an image || command = run`.

Specifies the working directory for the Docker container.

entrypointOverride - Entry point override

`string`. Optional. Use when `command = Run an image || command = run`.

Specifies whether to override the default entry point for the Docker container.

containerCommand - Container command

`string`. Optional. Use when `command = Run an image || command = run`.

Specifies a Docker run command. The docker run command first creates a writeable container layer over the specified image, and then starts it by using the specified run command. For example, if the image contains a simple Python Flask web application you can specify `python app.py` to launch the web application.

runInBackground - Run in background

`boolean`. Optional. Use when `command = Run an image || command = run`. Default value: `true`.

Specifies whether to run the Docker container in the background.

restartPolicy - Restart policy

`string`. Required when `runInBackground = true`. Allowed values: `no`, `onFailure` (On failure), `always`, `unlessStopped` (Unless stopped). Default value: `no`.

Specifies when to run a restart policy.

maxRestartRetries - Maximum restart retries

`string`. Optional. Use when `runInBackground = true && restartPolicy = onFailure`.

Specifies the maximum number of restart retries the Docker daemon attempts.

dockerHostEndpoint - Docker host service connection

`string`. Optional. Use when `command != login && command != logout`.

Specifies a Docker host service connection. Defaults to the agent's host.

enforceDockerNamingConvention - Force image name to follow Docker naming convention

`boolean`. Optional. Use when `command != login && command != logout`. Default value: `true`.

The default value modifies the Docker image name according to Docker naming conventions. For example, convert upper case characters to lower case and remove spaces.

memoryLimit - Memory limit

`string`. Optional. Use when `command != login && command != logout`.

Specifies the maximum amount of memory available to the container as an integer with optional suffixes like `2GB`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

DockerOutput

Stores the output of the docker command

DockerOutputPath

The path of the file which contains the output of the build command.

Remarks

[Docker@2](#) is a newer version of this task that simplifies the task by removing inputs that can be passed as arguments to the command.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Build

See also

- [Docker@2](#)

Docker@0 - Docker v0 task

Article • 09/26/2023

Build, tag, push, run Docker images, or run a Docker command.

Syntax

YAML

```
# Docker v0
# Build, tag, push, or run Docker images, or run a Docker command.
- task: Docker@0
  inputs:
    containerregistrytype: 'Azure Container Registry' # 'Azure Container Registry' | 'Container Registry'. Required. Container Registry Type. Default: Azure Container Registry.
    #dockerRegistryConnection: # string. Alias: dockerRegistryEndpoint. Optional. Use when containerregistrytype = Container Registry. Docker Registry Service Connection.
    #azureSubscription: # string. Alias: azureSubscriptionEndpoint. Optional. Use when containerregistrytype = Azure Container Registry. Azure subscription.
    #azureContainerRegistry: # string. Optional. Use when containerregistrytype = Azure Container Registry. Azure Container Registry.
    action: 'Build an image' # 'Build an image' | 'Tag images' | 'Push an image' | 'Push images' | 'Run an image' | 'Run a Docker command'. Required. Action. Default: Build an image.
    dockerFile: '**/Dockerfile' # string. Required when action = Build an image. Docker File. Default: **/Dockerfile.
    buildArguments: # string. Optional. Use when action = Build an image. Build Arguments.
    defaultContext: true # boolean. Optional. Use when action = Build an image. Use Default Build Context. Default: true.
    context: # string. Optional. Use when action = Build an image && defaultContext = false. Build Context.
    imageName: '$(Build.Repository.Name):$(Build.BuildId)' # string. Required when action == Build an image || action == Push an image || action == Run an image. Image Name. Default: $(Build.Repository.Name):$(Build.BuildId).
    imageNamesPath: # string. Required when action == Tag images || action == Push images. Image Names Path.
    qualify imageName: true # boolean. Optional. Use when action = Build an image || action = Tag images || action = Push an image || action = Push images || action = Run an image. Qualify Image Name. Default: true.
    additionalImageTags: # string. Optional. Use when action = Build an image || action = Tag images || action = Push an image || action = Push images. Additional Image Tags.
    includeSourceTags: false # boolean. Optional. Use when action = Build an image || action = Tag image || action = Push an image || action = Push images. Include Source Tags. Default: false.
```

```

    #includeLatestTag: false # boolean. Optional. Use when action = Build an
    image || action = Push an image || action = Push images. Include Latest Tag.
    Default: false.
    #imageDigestFile: # string. Optional. Use when action = Push an image || action = Push images. Image Digest File.
    #containerName: # string. Optional. Use when action = Run an image.
    Container Name.
    #ports: # string. Optional. Use when action = Run an image. Ports.
    #volumes: # string. Optional. Use when action = Run an image. Volumes.
    #envVars: # string. Optional. Use when action = Run an image.
    Environment Variables.
    #workDir: # string. Optional. Use when action = Run an image. Working
    Directory.
    #entrypoint: # string. Optional. Use when action = Run an image. Entry
    Point Override.
    #containerCommand: # string. Optional. Use when action = Run an image.
    Command.
    #detached: true # boolean. Optional. Use when action = Run an image. Run
    In Background. Default: true.
    #restartPolicy: 'no' # 'no' | 'onFailure' | 'always' | 'unlessStopped'.
    Required when action = Run an image && detached = true. Restart Policy.
    Default: no.
    #restartMaxRetries: # string. Optional. Use when action = Run an image
    && detached = true && restartPolicy = onFailure. Maximum Restart Retries.
    #customCommand: # string. Required when action = Run a Docker command.
    Command.
    # commands
    #addBaseImageData: true # boolean. Add base image metadata to image(s).
    Default: true.
    # Advanced Options
    #dockerHostEndpoint: # string. Docker Host Service Connection.
    #enforceDockerNamingConvention: true # boolean. Force image name to
    follow Docker naming convention. Default: true.
    #workingDirectory: '$(System.DefaultWorkingDirectory)' # string. Alias:
    cwd. Working Directory. Default: $(System.DefaultWorkingDirectory).
    #memory: # string. Memory limit.

```

Inputs

containerregistrytype - Container Registry Type

`string`. Required. Allowed values: `Azure Container Registry`, `Container Registry`.

Default value: `Azure Container Registry`.

Select 'Azure Container Registry' to connect to it by using an Azure Service Connection. Select 'Container registry' to connect to Docker Hub or any other private container registry.

dockerRegistryConnection - Docker Registry Service Connection

Input alias: `dockerRegistryEndpoint`. `string`. Optional. Use when `containerregistrytype = Container Registry`.

Specifies a Docker registry service connection. Required for commands that need to authenticate with a registry.

azureSubscription - Azure subscription

Input alias: `azureSubscriptionEndpoint`. `string`. Optional. Use when `containerregistrytype = Azure Container Registry`.

Specifies an Azure subscription.

azureContainerRegistry - Azure Container Registry

`string`. Optional. Use when `containerregistrytype = Azure Container Registry`.

Specifies an Azure Container Registry in the selected Azure Subscription. The container image is built and then pushed to this container registry.

action - Action

`string`. Required. Allowed values: `Build an image`, `Tag images`, `Push an image`, `Push images`, `Run an image`, `Run a Docker command`. Default value: `Build an image`.

Specifies a Docker action.

dockerFile - Docker File

`string`. Required when `action = Build an image`. Default value: `*/Dockerfile`.

Specifies the path to the Docker file. The task uses the first Docker file it finds to build the image.

addBaseImageData - Add base image metadata to image(s)

`boolean`. Default value: `true`.

The default value adds base image data such as, the base image name and digest to help with traceability. You can opt out by setting the value to `false`.

buildArguments - Build Arguments

`string`. Optional. Use when `action = Build an image`.

Specifies build-time variables for the Docker file. Format each `name=value` pair on a new line.

defaultContext - Use Default Build Context

`boolean`. Optional. Use when `action = Build an image`. Default value: `true`.

Specifies the build context of the directory that contains the Docker file.

context - Build Context

`string`. Optional. Use when `action = Build an image && defaultContext = false`.

Specifies the path to the build context.

imageName - Image Name

`string`. Required when `action == Build an image || action == Push an image || action == Run an image`. Default value: `$(Build.Repository.Name):$(Build.BuildId)`.

Specifies the name of the Docker image to build, push, or run.

imageNamesPath - Image Names Path

`string`. Required when `action == Tag images || action == Push images`.

Specifies the path to a text file that contains the names of the Docker images to tag or push. List each image name on a separate line.

qualifyImageName - Qualify Image Name

`boolean`. Optional. Use when `action = Build an image || action = Tag images || action = Push an image || action = Push images || action = Run an image`. Default value: `true`.

Specifies a qualify image name with the Docker registry service connection's hostname.

additionalImageTags - Additional Image Tags

`string`. Optional. Use when `action = Build an image || action = Tag images || action = Push an image || action = Push images`.

Specifies additional tags for the Docker image being built or pushed.

includeSourceTags - Include Source Tags

`boolean`. Optional. Use when `action = Build an image || action = Tag image || action = Push an image || action = Push images`. Default value: `false`.

Specifies whether to include Git tags when building or pushing the Docker image.

includeLatestTag - Include Latest Tag

`boolean`. Optional. Use when `action = Build an image || action = Push an image || action = Push images`. Default value: `false`.

Specifies whether to include the *latest* tag when building or pushing the Docker image.

imageDigestFile - Image Digest File

`string`. Optional. Use when `action = Push an image || action = Push images`.

Specifies the path to a file that is created and populated with the full image repository digest of the Docker image that was pushed.

containerName - Container Name

`string`. Optional. Use when `action = Run an image`.

Specifies the name of the Docker container to run.

ports - Ports

`string`. Optional. Use when `action = Run an image`.

Specifies ports in the Docker container to publish to the host. List each `host-port:container-port` binding on a new line.

volumes - Volumes

`string`. Optional. Use when `action = Run an image`.

Specifies the volumes to mount from the host. List each `host-dir:container-dir` on a new line.

envVars - Environment Variables

`string`. Optional. Use when `action = Run an image`.

Specifies environment variables for the Docker container. List each `name=value` pair on a new line.

workDir - Working Directory

`string`. Optional. Use when `action = Run an image`.

Specifies the working directory for the Docker container.

entrypoint - Entry Point Override

`string`. Optional. Use when `action = Run an image`.

Specifies an override of the default entry point for the Docker container.

containerCommand - Command

`string`. Optional. Use when `action = Run an image`.

Specifies a Docker run command. The docker run command first creates a writeable container layer over the specified image, and then starts it by using the specified run command. For example, if the image contains a simple Python Flask web application you can specify `python app.py` to launch the web application.

detached - Run In Background

`boolean`. Optional. Use when `action = Run an image`. Default value: `true`.

Specifies whether to run the Docker container in the background.

restartPolicy - Restart Policy

`string`. Required when `action = Run an image && detached = true`. Allowed values: `no`, `onFailure` (On failure), `always`, `unlessStopped` (Unless stopped). Default value: `no`.

Specifies a restart policy.

restartMaxRetries - Maximum Restart Retries

`string`. Optional. Use when `action = Run an image && detached = true && restartPolicy = onFailure`.

Specifies the maximum number of restart retries that the Docker daemon attempts.

customCommand - Command

`string`. Required when `action = Run a Docker command`.

Specifies the Docker command and arguments to execute. For example, `rmi -f imageName` removes an image.

dockerHostEndpoint - Docker Host Service Connection

`string`.

Specifies a Docker host service connection. Defaults to the agent's host.

enforceDockerNamingConvention - Force image name to follow Docker naming convention

`boolean`. Default value: `true`.

If enabled, modifies the Docker image name according to Docker naming conventions. For example, convert upper case characters to lower case and remove spaces.

workingDirectory - Working Directory

Input alias: `cwd`. `string`. Default value: `$(System.DefaultWorkingDirectory)`.

Specifies the working directory for the Docker command.

memory - Memory limit

`string`.

Specifies the maximum amount of memory available to the container as a integer with optional suffixes, for example `2GB`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

DockerOutput

Stores the output of the docker command.

DockerOutputPath

The path of the file which contains the output of the build command.

Remarks

[Docker@2](#) is a newer version of this task that simplifies the task by removing inputs that can be passed as arguments to the command.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Build

See also

- [Docker@2](#)

DockerInstaller@0 - Docker CLI installer v0 task

Article • 09/26/2023

Install the Docker CLI on an agent machine.

Syntax

YAML

```
# Docker CLI installer v0
# Install Docker CLI on agent machine.
- task: DockerInstaller@0
  inputs:
    dockerVersion: '17.09.0-ce' # string. Required. Docker Version. Default: 17.09.0-ce.
    #releaseType: 'stable' # 'stable' | 'edge' | 'test' | 'nightly'. Release type. Default: stable.
```

Inputs

`dockerVersion` - Docker Version

`string`. Required. Default value: `17.09.0-ce`.

Specifies the version of the Docker CLI to install.

`releaseType` - Release type

`string`. Allowed values: `stable`, `edge`, `test`, `nightly`. Default value: `stable`.

Specifies the release type to install. The value `nightly` is not supported on Windows.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to install a specific version of the Docker CLI on the agent machine.

Examples

This YAML example installs the Docker CLI on the agent machine:

YAML

```
- task: DockerInstaller@0
  displayName: Docker Installer
  inputs:
    dockerVersion: 17.09.0-ce
    releaseType: stable
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: Docker
Command restrictions	Any
Settable variables	Any
Agent version	2.142.1 or greater
Task category	Tool

DockerCompose@0 - Docker Compose v0 task

Article • 09/26/2023

Build, push or run multi-container Docker applications. Use this task with Docker or the Azure Container registry.

Syntax

YAML

```
# Docker Compose v0
# Build, push or run multi-container Docker applications. Task can be used
with Docker or Azure Container registry.
- task: DockerCompose@0
  inputs:
    containerregistrytype: 'Azure Container Registry' # 'Azure Container
Registry' | 'Container Registry'. Required. Container Registry Type.
    Default: Azure Container Registry.
    dockerRegistryEndpoint: # string. Optional. Use when
    containerregistrytype = Container Registry. Docker Registry Service
    Connection.
    azureSubscription: # string. Alias: azureSubscriptionEndpoint.
    Optional. Use when containerregistrytype = Azure Container Registry. Azure
    subscription.
    azureContainerRegistry: # string. Optional. Use when
    containerregistrytype = Azure Container Registry. Azure Container Registry.
    dockerComposeFile: '**/docker-compose.yml' # string. Required. Docker
    Compose File. Default: **/docker-compose.yml.
    additionalDockerComposeFiles: # string. Additional Docker Compose
    Files.
    dockerComposeFileArgs: # string. Environment Variables.
    projectName: '$(Build.Repository.Name)' # string. Project Name.
    Default: $(Build.Repository.Name).
    qualifyImageNames: true # boolean. Qualify Image Names. Default: true.
    action: 'Run a Docker Compose command' # 'Build services' | 'Push
    services' | 'Run services' | 'Run a specific service' | 'Lock services' |
    'Write service image digests' | 'Combine configuration' | 'Run a Docker
    Compose command'. Required. Action. Default: Run a Docker Compose command.
    additionalImageTags: # string. Optional. Use when action = Build
    services || action = Push services. Additional Image Tags.
    includeSourceTags: false # boolean. Optional. Use when action = Build
    services || action = Push services. Include Source Tags. Default: false.
    includeLatestTag: false # boolean. Optional. Use when action = Build
    services || action = Push services. Include Latest Tag. Default: false.
    buildImages: true # boolean. Optional. Use when action = Run services.
    Build Images. Default: true.
    serviceName: # string. Required when action = Run a specific service.
```

```

Service Name.
  #containerName: # string. Optional. Use when action = Run a specific
service. Container Name.
  #ports: # string. Optional. Use when action = Run a specific service.
Ports.
  #workingDirectory: # string. Alias: workDir. Optional. Use when action =
Run a specific service. Working Directory.
  #entrypoint: # string. Optional. Use when action = Run a specific
service. Entry Point Override.
  #containerCommand: # string. Optional. Use when action = Run a specific
service. Command.
  #detached: true # boolean. Optional. Use when action = Run services |||
action = Run a specific service. Run in Background. Default: true.
  #abortOnContainerExit: true # boolean. Optional. Use when action = Run
services && detached == false. Abort on Container Exit. Default: true.
  #imageDigestComposeFile: '$(Build.StagingDirectory)/docker-
compose.images.yml' # string. Required when action = Write service image
digests. Image Digest Compose File. Default:
$(Build.StagingDirectory)/docker-compose.images.yml.
  #removeBuildOptions: false # boolean. Optional. Use when action = Lock
services ||| action = Combine configuration. Remove Build Options. Default:
false.
  #baseResolveDirectory: # string. Optional. Use when action = Lock
services ||| action = Combine configuration. Base Resolve Directory.
  #outputDockerComposeFile: '$(Build.StagingDirectory)/docker-compose.yml'
# string. Required when action = Lock services ||| action = Combine
configuration. Output Docker Compose File. Default:
$(Build.StagingDirectory)/docker-compose.yml.
  #dockerComposeCommand: # string. Required when action = Run a Docker
Compose command. Command.
  #arguments: # string. Optional. Use when action != Lock services &&
action != Combine configuration && action != Write service image digests.
Arguments.
  # Advanced Options
  #dockerHostEndpoint: # string. Docker Host Service Connection.
  #nopIfNoDockerComposeFile: false # boolean. No-op if no Docker Compose
File. Default: false.
  #requireAdditionalDockerComposeFiles: false # boolean. Require
Additional Docker Compose Files. Default: false.
  #currentWorkingDirectory: '$(System.DefaultWorkingDirectory)' # string.
Alias: cwd. Working Directory. Default: $(System.DefaultWorkingDirectory).
  #dockerComposePath: # string. Docker Compose executable Path.

```

Inputs

`containerregistrytype` - Container Registry Type

`string`. Required. Allowed values: Azure Container Registry, Container Registry.

Default value: Azure Container Registry.

Specifies an Azure Container Registry type if using ACR. Specify a Container Registry type if using any other container registry.

dockerRegistryEndpoint - Docker Registry Service Connection

`string`. Optional. Use when `containerregistrytype = Container Registry`.

Specifies a Docker registry service connection. Required when commands need to authenticate using a registry.

azureSubscription - Azure subscription

Input alias: `azureSubscriptionEndpoint`. `string`. Optional. Use when `containerregistrytype = Azure Container Registry`.

Specifies an Azure subscription. Name of the Azure Service Connection. To manually set up the connection, see [Azure Resource Manager service connection](#).

azureContainerRegistry - Azure Container Registry

`string`. Optional. Use when `containerregistrytype = Azure Container Registry`.

Specifies an Azure Container Registry.

dockerComposeFile - Docker Compose File

`string`. Required. Default value: `**/docker-compose.yml`.

Specifies the file path to the primary Docker Compose file.

additionalDockerComposeFiles - Additional Docker Compose Files

`string`.

Specifies additional Docker Compose files that are combined with the primary Docker Compose file. Relative paths are resolved relative to the directory containing the primary Docker Compose file. If a specified file is not found, it is ignored. Specify each file path on a new line.

dockerComposeFileArgs - Environment Variables

`string`.

Specifies any environment variables that are set.

Format as follows:

- List each `name=value` pair on a new line.
- Use the `|` operator in YAML to preserve new lines.

`projectName` - Project Name

`string`. Default value: `$(Build.Repository.Name)`.

Specifies the project name to use by default to name images and containers.

`qualifyImageNames` - Qualify Image Names

`boolean`. Default value: `true`.

By default, specifies the Docker registry service connection's hostname.

`action` - Action

`string`. Required. Allowed values: `Build services` (Build service images), `Push services` (Push service images), `Run services` (Run service images), `Run a specific service` (Run a specific service image), `Lock services` (Lock service images), `Write service image digests`, `Combine configuration`, `Run a Docker Compose command`. Default value: `Run a Docker Compose command`.

Specifies a Docker Compose action from the list of allowed values.

`additionalImageTags` - Additional Image Tags

`string`. Optional. Use when `action = Build services || action = Push services`.

Specifies additional tags for the Docker images being built or pushed. You can specify multiple tags separating each with a line feed `\n`.

`includeSourceTags` - Include Source Tags

`boolean`. Optional. Use when `action = Build services || action = Push services`.

Default value: `false`.

Specifies Git tags when building or pushing Docker images.

includeLatestTag - Include Latest Tag

`boolean`. Optional. Use when `action = Build services || action = Push services`.

Default value: `false`.

Specifies the *latest* tag when building or pushing Docker images.

buildImages - Build Images

`boolean`. Optional. Use when `action = Run services`. Default value: `true`.

Specifies build images before starting service containers.

serviceName - Service Name

`string`. Required when `action = Run a specific service`.

Specifies the name of the service you want to run.

containerName - Container Name

`string`. Optional. Use when `action = Run a specific service`.

Specifies the name of the service container you want to use.

ports - Ports

`string`. Optional. Use when `action = Run a specific service`.

Specifies ports in the service container to publish to the host. Add each `host-port:container-port` binding on a new line.

workingDirectory - Working Directory

Input alias: `workDir`. `string`. Optional. Use when `action = Run a specific service`.

Specifies the working directory for the service container.

entrypoint - Entry Point Override

`string`. Optional. Use when `action = Run a specific service`.

Specifies an override value for the default entry point of the service container.

`containerCommand` - Command

`string`. Optional. Use when `action = Run a specific service`.

Specifies the command to run in the service container. For example, if the image contains a simple Python Flask web application you can specify `python app.py` to launch the web application.

`detached` - Run in Background

`boolean`. Optional. Use when `action = Run services || action = Run a specific service`. Default value: `true`.

Specifies the service containers to run in the background.

`abortOnContainerExit` - Abort on Container Exit

`boolean`. Optional. Use when `action = Run services && detached == false`. Default value: `true`.

Specifies all containers that should stop when any container exits.

`imageDigestComposeFile` - Image Digest Compose File

`string`. Required when `action = Write service image digests`. Default value: `$(Build.StagingDirectory)/docker-compose.images.yml`.

Specifies the path to a Docker Compose file that is created and populated with the full image repository digests of each service's Docker image.

`removeBuildOptions` - Remove Build Options

`boolean`. Optional. Use when `action = Lock services || action = Combine configuration`. Default value: `false`.

Specifies if build options should be removed from the output Docker Compose file.

`baseResolveDirectory` - Base Resolve Directory

`string`. Optional. Use when `action = Lock services || action = Combine configuration`.

Specifies the base directory from which relative paths in the output Docker Compose file should be resolved.

outputDockerComposeFile - Output Docker Compose File

`string`. Required when `action = Lock services || action = Combine configuration`.

Default value: `$(Build.StagingDirectory)/docker-compose.yml`.

Specifies the path to an output Docker Compose file.

dockerComposeCommand - Command

`string`. Required when `action = Run a Docker Compose command`.

Specifies the Docker Compose command to execute with arguments. For example, `rm -a1` to remove all stopped service containers.

arguments - Arguments

`string`. Optional. Use when `action != Lock services && action != Combine configuration && action != Write service image digests`.

Specifies Docker Compose command options.

Example: For the build command, `--pull --compress --parallel`.

dockerHostEndpoint - Docker Host Service Connection

`string`.

Specifies a Docker host service connection. Defaults to the agent's host.

nopIfNoDockerComposeFile - No-op if no Docker Compose File

`boolean`. Default value: `false`.

Specifies a value to skip the task if the Docker Compose file does not exist. This option is useful when the task offers optional behavior based on the existence of a Docker Compose file in the repository.

requireAdditionalDockerComposeFiles - Require Additional Docker Compose Files

`boolean`. Default value: `false`.

Specifies a value to produce an error if the additional Docker Compose files do not exist. This option overrides the default behavior that would ignore a file if it does not exist.

currentWorkingDirectory - Working Directory

Input alias: `cwd`. `string`. Default value: `$(System.DefaultWorkingDirectory)`.

Specifies the working directory for the Docker Compose command.

dockerComposePath - Docker Compose executable Path

`string`.

Specifies a path to determine if the docker-compose executable is used.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

DockerComposeOutput

The path to the files which contain the output of the command. This can contain multiple file paths (separated by newline characters) such as, dockerComposeRun command (one for running and one for down), dockerPush (one for each image pushed), dockerBuild (the build itself and all the tag commands) and dockerDigest (one for each image pulled). The other commands only output one file.

Remarks

Use this task to build, push or run multi-container Docker applications. Use this task with a Docker registry or an Azure Container Registry.

Examples

- [Azure Container Registry](#)
- [Other container registries](#)
- [Build service images](#)

- Push service images
- Run service images
- Run a specific service image
- Lock service images
- Write service image digests
- Combine configuration
- Run a Docker Compose command

Azure Container Registry

This YAML example specifies the inputs for Azure Container Registry:

YAML

```
variables:
  azureContainerRegistry: Contoso.azurecr.io
  azureSubscriptionEndpoint: Contoso
steps:
- task: DockerCompose@0
  displayName: Container registry login
  inputs:
    containerregistrytype: Azure Container Registry
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureContainerRegistry: $(azureContainerRegistry)
```

Other container registries

The `containerregistrytype` value is required when using any container registry other than ACR. Use `containerregistrytype: Container Registry` in this case.

This YAML example specifies a container registry other than ACR where **Contoso** is the name of the Docker registry service connection for the container registry:

YAML

```
- task: DockerCompose@0
  displayName: Container registry login
  inputs:
    containerregistrytype: Container Registry
    dockerRegistryEndpoint: Contoso
```

Build service images

This YAML example builds the image where the image name is qualified on the basis of the inputs related to Azure Container Registry:

YAML

```
- task: DockerCompose@0
  displayName: Build services
  inputs:
    action: Build services
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureContainerRegistry: $(azureContainerRegistry)
    dockerComposeFile: docker-compose.yml
    projectName: $(Build.Repository.Name)
    qualifyImageNames: true
    additionalImageTags: $(Build.BuildId)
    dockerComposeFileArgs: |
      firstArg=$(firstArg)
      secondArg=$(secondArg)
```

Push service images

This YAML example pushes an image to a container registry:

YAML

```
- task: DockerCompose@0
  displayName: Push services
  inputs:
    action: Push services
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureContainerRegistry: $(azureContainerRegistry)
    dockerComposeFile: docker-compose.yml
    projectName: $(Build.Repository.Name)
    qualifyImageNames: true
    additionalImageTags: $(Build.BuildId)
```

Run service images

This YAML example runs services:

YAML

```
- task: DockerCompose@0
  displayName: Run services
  inputs:
    action: Run services
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
```

```
azureContainerRegistry: $(azureContainerRegistry)
dockerComposeFile: docker-compose.ci.build.yml
 projectName: $(Build.Repository.Name)
qualifyImageNames: true
buildImages: true
abortOnContainerExit: true
detached: true
```

Run a specific service image

This YAML example runs a specific service:

YAML

```
- task: DockerCompose@0
displayName: Run a specific service
inputs:
  action: Run a specific service
  azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
  azureContainerRegistry: $(azureContainerRegistry)
  dockerComposeFile: docker-compose.yml
  projectName: $(Build.Repository.Name)
  qualifyImageNames: true
  serviceName: myhealth.web
  ports: 80:80
  detached: true
```

Lock service images

This YAML example locks services:

YAML

```
- task: DockerCompose@0
displayName: Lock services
inputs:
  action: Lock services
  azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
  azureContainerRegistry: $(azureContainerRegistry)
  dockerComposeFile: docker-compose.yml
  projectName: $(Build.Repository.Name)
  qualifyImageNames: true
  outputDockerComposeFile: $(Build.StagingDirectory)/docker-compose.yml
```

Write service image digests

This YAML example writes service image digests:

YAML

```
- task: DockerCompose@0
  displayName: Write service image digests
  inputs:
    action: Write service image digests
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureContainerRegistry: $(azureContainerRegistry)
    dockerComposeFile: docker-compose.yml
    projectName: $(Build.Repository.Name)
    qualifyImageNames: true
    imageDigestComposeFile: $(Build.StagingDirectory)/docker-
      compose.images.yml
```

Combine configuration

This YAML example combines configurations:

YAML

```
- task: DockerCompose@0
  displayName: Combine configuration
  inputs:
    action: Combine configuration
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureContainerRegistry: $(azureContainerRegistry)
    dockerComposeFile: docker-compose.yml
    additionalDockerComposeFiles: docker-compose.override.yml
    projectName: $(Build.Repository.Name)
    qualifyImageNames: true
    outputDockerComposeFile: $(Build.StagingDirectory)/docker-compose.yml
```

Run a Docker Compose command

This YAML example runs a docker Compose command:

YAML

```
- task: DockerCompose@0
  displayName: Run a Docker Compose command
  inputs:
    action: Run a Docker Compose command
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureContainerRegistry: $(azureContainerRegistry)
    dockerComposeFile: docker-compose.yml
    projectName: $(Build.Repository.Name)
```

```
qualifyImageNames: true  
dockerComposeCommand: rm
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Build

DownloadFileshareArtifacts@1 - Download artifacts from file share v1 task

Article • 09/26/2023

Use this task to download artifacts from a file share, like `\share\drop`.

Syntax

YAML

```
# Download artifacts from file share v1
# Download artifacts from a file share, like \share\drop.
- task: DownloadFileshareArtifacts@1
  inputs:
    filesharePath: # string. Required. File share path.
    artifactName: # string. Required. Artifact name.
    #itemPattern: '**' # string. Matching pattern. Default: **.
    downloadPath: '$(System.ArtifactsDirectory)' # string. Required.
    Download path. Default: $(System.ArtifactsDirectory).
    # Advanced
    #parallelizationLimit: '8' # string. Parallelization limit. Default: 8.
```

Inputs

`filesharePath` - File share path

`string`. Required.

Specifies the file share path (for example: `\server\folder`).

`artifactName` - Artifact name

`string`. Required.

Specifies the name of the artifact to download (for example: `drop`).

`itemPattern` - Matching pattern

`string`. Default value: `**`.

Specifies the files to be downloaded as a multi line minimatch pattern. Learn more about [file matching patterns](#).

The default pattern (**) downloads all files within the artifact.

downloadPath - Download path

`string`. Required. Default value: `$(System.ArtifactsDirectory)`.

Specifies the path on the agent machine where the artifacts are downloaded.

parallelizationLimit - Parallelization limit

`string`. Default value: `8`.

Specifies the number of files to download simultaneously.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to download file share artifacts.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.

Requirement	Description
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Utility

DownloadBuildArtifacts@1 - Download build artifacts v1 task

Article • 09/26/2023

Use this task to download files that were saved as artifacts of a completed build.

Syntax

YAML

```
# Download build artifacts v1
# Download files that were saved as artifacts of a completed build.
- task: DownloadBuildArtifacts@1
  inputs:
    buildType: 'current' # 'current' | 'specific'. Required. Download artifacts produced by. Default: current.
    #project: # string. Required when buildType == specific. Project.
    #pipeline: # string. Alias: definition. Required when buildType == specific. Build pipeline.
    #specificBuildWithTriggering: false # boolean. Optional. Use when buildType == specific. When appropriate, download artifacts from the triggering build. Default: false.
    #buildVersionToDownload: 'latest' # 'latest' | 'latestFromBranch' | 'specific'. Required when buildType == specific. Build version to download. Default: latest.
    #allowPartiallySucceededBuilds: false # boolean. Optional. Use when buildType == specific && buildVersionToDownload != specific. Download artifacts even from partially succeeded builds. Default: false.
    #branchName: 'refs/heads/master' # string. Required when buildType == specific && buildVersionToDownload == latestFromBranch. Branch name. Default: refs/heads/master.
    #buildId: # string. Required when buildType == specific && buildVersionToDownload == specific. Build.
    #tags: # string. Optional. Use when buildType == specific && buildVersionToDownload != specific. Build Tags.
    downloadType: 'single' # 'single' | 'specific'. Required. Download type. Default: single.
    artifactName: # string. Required when downloadType == single. Artifact name.
    #itemPattern: '**' # string. Matching pattern. Default: **.
    downloadPath: '$(System.ArtifactsDirectory)' # string. Required. Destination directory. Default: $(System.ArtifactsDirectory).
    #cleanDestinationFolder: false # boolean. Clean destination folder. Default: false.
    # Advanced
    #parallelizationLimit: '8' # string. Parallelization limit. Default: 8.
    #checkDownloadedFiles: false # boolean. Check downloaded files. Default: false.
    #retryDownloadCount: '4' # string. Retry count. Default: 4.
```

```
#extractTars: # boolean. Extract all files that are stored inside tar archives.
```

Inputs

buildType - Download artifacts produced by

`string`. Required. Allowed values: `current` (Current build), `specific` (Specific build).

Default value: `current`.

Whether to download artifacts produced by the current build or from a specific build.

project - Project

`string`. Required when `buildType == specific`.

The project from which to download the build artifacts.

pipeline - Build pipeline

Input alias: `definition.string`. Required when `buildType == specific`.

Specifies the build pipeline name.

specificBuildWithTriggering - When appropriate, download artifacts from the triggering build.

`boolean`. Optional. Use when `buildType == specific`. Default value: `false`.

If `true`, this build task tries to download artifacts from the triggering build. If there is no triggering build from the specified pipeline, it downloads artifacts from the build specified in the options below.

buildVersionToDownload - Build version to download

`string`. Required when `buildType == specific`. Allowed values: `latest`, `latestFromBranch` (Latest from specific branch and specified Build Tags), `specific` (Specific version). Default value: `latest`.

Specifies which version of the build to download.

- Choose `latest` to download the latest available build version.

- Choose `latestFromBranch` to download the latest available build version of the branch specified by `branchName` and tags specified by `tags`.
- Choose `specific` to download the build version specified by `buildId`.

allowPartiallySucceededBuilds - Download artifacts even from partially succeeded builds.

`boolean`. Optional. Use when `buildType == specific && buildVersionToDownload != specific`. Default value: `false`.

If `true`, this build task tries to download artifacts whether the build succeeds or partially succeeds.

branchName - Branch name

`string`. Required when `buildType == specific && buildVersionToDownload == latestFromBranch`. Default value: `refs/heads/master`.

Specifies whether to filter on branch/ref name, for example: `refs/heads/develop`.

buildId - Build

`string`. Required when `buildType == specific && buildVersionToDownload == specific`.

The build you want to download the artifacts from.

tags - Build Tags

`string`. Optional. Use when `buildType == specific && buildVersionToDownload != specific`.

A comma-delimited list of tags. Only builds with these tags are returned.

downloadType - Download type

`string`. Required. Allowed values: `single` (Specific artifact), `specific` (Specific files).

Default value: `single`.

Downloads a specific artifact or specific files from the build.

- Choose `single` (Specific artifact) when you want only one specific artifact specified by `artifactName`

- Choose `specific` (Specific files) when you want all artifacts of the selected build

artifactName - Artifact name

`string`. Required when `downloadType == single`.

The name of the artifact to download.

itemPattern - Matching pattern

`string`. Default value: `**`.

Specifies the files to download as a multi-line minimatch pattern. For more information, see [File matching patterns reference](#).

The default pattern `**` downloads all files across all artifacts in the build if you choose the **Specific files** option. To download all files within the artifact drop, use `drop/**`.

downloadPath - Destination directory

`string`. Required. Default value: `$(System.ArtifactsDirectory)`.

The path on the agent machine where the artifacts are downloaded.

cleanDestinationFolder - Clean destination folder

`boolean`. Default value: `false`.

Delete all existing files in destination folder before artifact download.

parallelizationLimit - Parallelization limit

`string`. Default value: `8`.

The number of files to download simultaneously.

checkDownloadedFiles - Check downloaded files

`boolean`. Default value: `false`.

If `true`, this build task checks that all files are fully downloaded.

`retryDownloadCount` - Retry count

`string`. Default value: `4`.

The number of times to retry downloading a build artifact if the download fails.

`extractTars` - Extract all files that are stored inside tar archives

`boolean`.

Set to `true` to extract all downloaded files that have the `.tar` extension. This is helpful because you need to pack your artifact files into tar if you want to preserve Unix file permissions. Enabling the `StoreAsTar` option in the [Publish build artifacts](#) task will store artifacts as `.tar` files automatically.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

`BuildNumber`

Stores the build number of the build artifact source.

Please note that this input returns `BuildId` due to backward compatibility. For more information, see [Variables](#).

Remarks

Note

The Azure Pipelines team recommends upgrading from **build artifacts** to **Pipeline Artifacts** for faster performance.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.191.1 or greater
Task category	Utility

DownloadBuildArtifacts@0 - Download build artifacts v0 task

Article • 09/26/2023

Use this task to download files that were saved as artifacts of a completed build.

ⓘ Note

Disable IIS Basic Authentication if you are using Azure DevOps Server to allow authentication with your Personal Access Token. See [IIS Basic Authentication and PATs](#) for more details.

Syntax

YAML

```
# Download build artifacts v0
# Download files that were saved as artifacts of a completed build.
- task: DownloadBuildArtifacts@0
  inputs:
    buildType: 'current' # 'current' | 'specific'. Required. Download artifacts produced by. Default: current.
    #project: # string. Required when buildType == specific. Project.
    #pipeline: # string. Alias: definition. Required when buildType == specific. Build pipeline.
    #specificBuildWithTriggering: false # boolean. Optional. Use when buildType == specific. When appropriate, download artifacts from the triggering build. Default: false.
    #buildVersionToDownload: 'latest' # 'latest' | 'latestFromBranch' | 'specific'. Required when buildType == specific. Build version to download. Default: latest.
    #allowPartiallySucceededBuilds: false # boolean. Optional. Use when buildType == specific && buildVersionToDownload != specific. Download artifacts even from partially succeeded builds. Default: false.
    #branchName: 'refs/heads/master' # string. Required when buildType == specific && buildVersionToDownload == latestFromBranch. Branch name. Default: refs/heads/master.
    #buildId: # string. Required when buildType == specific && buildVersionToDownload == specific. Build.
    #tags: # string. Optional. Use when buildType == specific && buildVersionToDownload != specific. Build Tags.
    downloadType: 'single' # 'single' | 'specific'. Required. Download type. Default: single.
    artifactName: # string. Required when downloadType == single. Artifact name.
    #itemPattern: '**' # string. Matching pattern. Default: **.
```

```
    downloadPath: '$(System.ArtifactsDirectory)' # string. Required.  
Destination directory. Default: $(System.ArtifactsDirectory).  
    #cleanDestinationFolder: false # boolean. Clean destination folder.  
Default: false.  
    # Advanced  
    #parallelizationLimit: '8' # string. Parallelization limit. Default: 8.  
    #checkDownloadedFiles: false # boolean. Check downloaded files. Default:  
false.  
    #retryDownloadCount: '4' # string. Retry count. Default: 4.  
    #extractTars: # boolean. Extract all files that are stored inside tar  
archives.
```

Inputs

buildType - Download artifacts produced by

`string`. Required. Allowed values: `current` (Current build), `specific` (Specific build).

Default value: `current`.

Whether to download artifacts produced by the current build or from a specific build.

project - Project

`string`. Required when `buildType == specific`.

The project from which you want to download the build artifacts.

pipeline - Build pipeline

Input alias: `definition`. `string`. Required when `buildType == specific`.

Specifies the build pipeline name.

specificBuildWithTriggering - When appropriate, download artifacts from the triggering build.

`boolean`. Optional. Use when `buildType == specific`. Default value: `false`.

If `true`, this build task tries to download artifacts from the triggering build. If there is no triggering build from the specified pipeline, it downloads artifacts from the build specified in the options below.

buildVersionToDownload - Build version to download

`string`. Required when `buildType == specific`. Allowed values: `latest`,

`latestFromBranch` (Latest from specific branch and specified Build Tags), `specific` (Specific version). Default value: `latest`.

`allowPartiallySucceededBuilds` - Download artifacts even from partially succeeded builds.

`boolean`. Optional. Use when `buildType == specific && buildVersionToDownload != specific`. Default value: `false`.

If `true`, this build task tries to download artifacts whether the build succeeds or partially succeeds.

`branchName` - Branch name

`string`. Required when `buildType == specific && buildVersionToDownload == latestFromBranch`. Default value: `refs/heads/master`.

Specifies whether to filter on branch/ref name, for example: `refs/heads/develop`.

`buildId` - Build

`string`. Required when `buildType == specific && buildVersionToDownload == specific`.

The build you want to download the artifacts from.

`tags` - Build Tags

`string`. Optional. Use when `buildType == specific && buildVersionToDownload != specific`.

A comma-delimited list of tags. Only builds with these tags are returned.

`downloadType` - Download type

`string`. Required. Allowed values: `single` (Specific artifact), `specific` (Specific files). Default value: `single`.

Downloads a specific artifact or specific files from the build.

`artifactName` - Artifact name

`string`. Required when `downloadType == single`.

The name of the artifact to download.

itemPattern - Matching pattern

`string`. Default value: `**`.

Specifies the files to download as a multi-line minimatch pattern. For more information, see [File matching patterns reference](#).

The default pattern `**` downloads all files across all artifacts in the build if you choose the **Specific files** option. To download all the files within the artifact drop, use `drop/**`.

downloadPath - Destination directory

`string`. Required. Default value: `$(System.ArtifactsDirectory)`.

The path on the agent machine where the artifacts are downloaded.

cleanDestinationFolder - Clean destination folder

`boolean`. Default value: `false`.

Deletes all the existing files in the destination folder before the artifact is downloaded.

parallelizationLimit - Parallelization limit

`string`. Default value: `8`.

The number of files to download simultaneously.

checkDownloadedFiles - Check downloaded files

`boolean`. Default value: `false`.

If `true`, this build task checks that all files are fully downloaded.

retryDownloadCount - Retry count

`string`. Default value: `4`.

The number of times to retry downloading a build artifact if the download fails.

`extractTars` - Extract all files that are stored inside tar archives

`boolean`.

Extracts all downloaded files that have a `.tar` extension. This is helpful because you need to pack your artifact files into a `.tar` file if you want to preserve Unix file permissions. Enabling the `StoreAsTar` option in the PublishBuildArtifacts task stores artifacts as `.tar` files automatically.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

`BuildNumber`

Stores the build artifact source's build number.

Please note that this input returns `BuildId` due to backward compatibility. For more information, see [Variables](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Utility

DownloadGithubNpmPackage@1 - Download Github Npm Package v1 task

Article • 09/26/2023

Use this task to install npm packages from GitHub.

Syntax

YAML

```
# Download Github Npm Package v1
# Install npm packages from GitHub.
- task: DownloadGithubNpmPackage@1
  inputs:
    packageName: # string. Required. Package Name.
    version: # string. Required. Package Version.
    externalRegistryCredentials: # string. Alias: externalEndpoints.
    Required. Credentials for registry from GitHub.
    #installDirectory: # string. Alias: packagesDirectory. Destination
    directory.
```

Inputs

packageName - Package Name

`string`. Required.

Specifies the name of the package to download from GitHub.

version - Package Version

`string`. Required.

Specifies the version of the package to download from GitHub.

externalRegistryCredentials - Credentials for registry from GitHub

Input alias: `externalEndpoints`. `string`. Required.

Specifies the credentials to use for external registry from GitHub.

`installDirectory` - Destination directory

Input alias: `packagesDirectory`. `string`.

Specifies the folder where packages are installed. If no folder is specified, packages are restored into the default system working directory.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: npm
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Package

DownloadGitHubNugetPackage@1 - Download GitHub Nuget Packages v1 task

Article • 09/26/2023

Use this task to restore your NuGet packages using dotnet CLI.

Syntax

YAML

```
# Download GitHub Nuget Packages v1
# Restore your nuget packages using dotnet CLI.
- task: DownloadGitHubNugetPackage@1
  inputs:
    packageName: # string. Required. Package Name.
    version: # string. Required. Package Version.
    # Feeds and authentication
    #externalFeedCredentials: # string. Alias: externalEndpoints. Required
    when selectOrConfig = config && command = restore. Credentials for feed from
    GitHub.
    # Advanced
    #restoreDirectory: # string. Alias: packagesDirectory. Optional. Use
    when command = restore. Destination directory.
```

Inputs

packageName - Package Name

string. Required.

Specifies the name of the package to download from GitHub.

version - Package Version

string. Required.

Specifies the version of the package to download from GitHub.

externalFeedCredentials - Credentials for feed from GitHub

Input alias: externalEndpoints. string. Required when selectOrConfig = config &&

```
command = restore.
```

Specifies the credentials to use for external registry from GitHub.

restoreDirectory - Destination directory

Input alias: `packagesDirectory`. `string`. Optional. Use when `command = restore`.

Specifies the folder where packages are installed. If no folder is specified, packages are restored into the default system working directory.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Build

DownloadGitHubRelease@0 - Download GitHub Release v0 task

Article • 09/26/2023

Use this task to download a GitHub release from a repository.

Syntax

YAML

```
# Download GitHub Release v0
# Downloads a GitHub Release from a repository.
- task: DownloadGitHubRelease@0
  inputs:
    connection: # string. Required. GitHub Connection.
    userRepository: # string. Required. Repository.
    defaultVersionType: 'latest' # 'latest' | 'specificVersion' |
'specificTag'. Required. Default version. Default: latest.
    version: # string. Required when defaultVersionType != latest. Release.
    #itemPattern: '**' # string. Item Pattern. Default: **.
    downloadPath: '$(System.ArtifactsDirectory)' # string. Required.
Destination directory. Default: $(System.ArtifactsDirectory).
```

Inputs

`connection` - GitHub Connection

`string`. Required.

Specifies the GitHub service connection name. Learn more about [service connections](#).

`userRepository` - Repository

`string`. Required.

Specifies the name of the GitHub repository that GitHub releases are downloaded from.

`defaultVersionType` - Default version

`string`. Required. Allowed values: `latest` (Latest Release), `specificVersion` (Specific Version), `specificTag` (Specific Tag). Default value: `latest`.

Downloads assets from the latest GitHub release or a specific GitHub release version/tag.

version - Release

`string`. Required when `defaultVersionType != latest`.

Defines the GitHub release version/tag to download. This option appears if `specificVersion` or `specificTag` is selected as the value for `defaultVersionType`.

itemPattern - Item Pattern

`string`. Default value: `**`.

The minimatch pattern that filters files to be downloaded. To download all files within a release, use the default value `**`.

downloadPath - Destination directory

`string`. Required. Default value: `$(System.ArtifactsDirectory)`.

The path on the agent machine where the release assets are downloaded.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task in your pipeline to download assets from your [GitHub release](#) as part of your CI/CD pipeline.

GitHub service connection

This task requires a [GitHub service connection](#) with **Read** permission to the GitHub repository. You can create a GitHub service connection in your Azure Pipelines project. Once created, use the name of the service connection in this task's settings.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Utility

DownloadPackage@1 - Download package v1 task

Article • 09/26/2023

Use this task to download a package from a package management feed in Azure Artifacts.

Syntax

YAML

```
# Download package v1
# Download a package from a package management feed in Azure Artifacts.
- task: DownloadPackage@1
  inputs:
    packageType: 'nuget' # 'maven' | 'npm' | 'nuget' | 'pypi' | 'upack' |
    'cargo'. Required. Package Type. Default: nuget.
    feed: # string. Required. Feed.
    #view: # string. View.
    definition: # string. Required. Package.
    version: # string. Required. Version.
    downloadPath: '$(System.ArtifactsDirectory)' # string. Required.
    Destination directory. Default: $(System.ArtifactsDirectory).
    # Advanced
    #files: '**' # string. Optional. Use when packageType = maven ||
    packageType = pypi || packageType = upack. Files. Default: **.
    #extract: true # boolean. Optional. Use when packageType = nuget ||
    packageType = npm. Extract package contents. Default: true.
```

Inputs

packageType - Package Type

`string`. Required. Allowed values: `maven`, `npm`, `nuget`, `pypi` (Python), `upack` (Universal), `cargo`. Default value: `nuget`.

feed - Feed

`string`. Required.

For project-scoped feeds, the format is `projectID/feedID`. See the following [remarks](#) to learn how to get a feed or project ID, or learn how to use a project and feed name instead.

`view` - View

`string`.

Specifies a view that only uses versions promoted to that specific view.

`definition` - Package

`string`. Required.

If you don't find the package in the list, you can provide the package ID, which you can find using the instructions [here ↗](#).

`version` - Version

`string`. Required.

Specifies the version of the package. Use `latest` to download the latest version of the package at runtime.

`files` - Files

`string`. Optional. Use when `packageType = maven || packageType = pypi || packageType = upack`. Default value: `**`.

Specifies which files to download using [file matching patterns ↗](#).

`extract` - Extract package contents

`boolean`. Optional. Use when `packageType = nuget || packageType = npm`. Default value: `true`.

Extracts the package contents and contains the package archive in the artifact folder.

`downloadPath` - Destination directory

`string`. Required. Default value: `$(System.ArtifactsDirectory)`.

Specifies the path on the agent machine where the package is downloaded.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to download a package from a package management feed in Azure Artifacts or TFS.

ⓘ Note

Requires the [Package Management extension](#).

How do I find the ID of the feed (or project) I want to download my artifact from

The get feed API can be used to retrieve the feed and project ID for your feed. The API is documented [here](#).

Can I use the project or feed name instead of IDs

Yes, you can use the project or feed name in your definition. However, if your project or feed is renamed in the future, your task will also have to be updated, or it might fail.

Examples

Download a NuGet package from an organization-scoped feed and extract to destination directory

YAML

```
# Download an artifact with id 'cfe01b64-ded4-47b7-a569-2ac17cbcedbd' to
$(System.ArtifactsDirectory)
- task: DownloadPackage@1
  inputs:
    packageType: 'nuget'
```

```

feed: '6a60ef3b-e29f-41b6-9885-7874278baac7'
definition: 'cfe01b64-ded4-47b7-a569-2ac17cbcedbd' # Can also be package
name
version: '1.0.0'
extract: true
downloadPath: '$(System.ArtifactsDirectory)'

```

Download a maven package from a project-scoped feed and download only pom files

YAML

```

# Download an artifact with name 'com.test:testpackage' to
$(System.ArtifactsDirectory)
- task: DownloadPackage@1
  inputs:
    packageType: 'maven'
    feed: '132f5c2c-2aa0-475a-8b47-02c79617954b/c85e5de9-7b12-4cf8-9293-
1b33cdff540e' # <projectId>/<feedId>
    definition: 'com.test:testpackage'
    version: '1.0.0-snapshot' # Should be normalized version
    files: '*.pom'
    downloadPath: '$(System.ArtifactsDirectory)'

```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Utility

DownloadPackage@0 - Download package v0 task

Article • 09/26/2023

Use this task to download a package from a package management feed in Azure Artifacts.

This task is deprecated; use [DownloadPackage@1](#).

Syntax

YAML

```
# Download package v0
# Download a package from a package management feed in Azure Artifacts.
- task: DownloadPackage@0
  inputs:
    feed: # string. Required. Feed.
    definition: # string. Required. Package.
    version: # string. Required. Version.
    downloadPath: '$(System.ArtifactsDirectory)' # string. Required.
    Destination directory. Default: $(System.ArtifactsDirectory).
```

Inputs

feed - Feed

`string`. Required.

Specifies the package source.

definition - Package

`string`. Required.

Specifies the package to download. Only NuGet packages are currently supported.

version - Version

`string`. Required.

Specifies the version of the package.

`downloadPath` - Destination directory

`string`. Required. Default value: `$(System.ArtifactsDirectory)`.

Specifies the path on the agent machine where the package is downloaded.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Utility

DownloadPipelineArtifact@2 - Download Pipeline Artifacts v2 task

Article • 09/26/2023

Use this task to download pipeline artifacts from earlier stages in this pipeline, or from another pipeline.

ⓘ Note

For more information, including Azure CLI commands, see [downloading artifacts](#).

Syntax

YAML

```
# Download Pipeline Artifacts v2
# Download build and pipeline artifacts.
- task: DownloadPipelineArtifact@2
  inputs:
    buildType: 'current' # 'current' | 'specific'. Alias: source. Required.
    Download artifacts produced by. Default: current.
    #project: # string. Required when source == specific. Project.
    #definition: # string. Alias: pipeline. Required when source ==
    specific. Build pipeline.
    #specificBuildWithTriggering: false # boolean. Alias:
    preferTriggeringPipeline. Optional. Use when source == specific. When
    appropriate, download artifacts from the triggering build. Default: false.
    #buildVersionToDownload: 'latest' # 'latest' | 'latestFromBranch' |
    'specific'. Alias: runVersion. Required when source == specific. Build
    version to download. Default: latest.
    #branchName: 'refs/heads/master' # string. Alias: runBranch. Required
    when source == specific && runVersion == latestFromBranch. Branch name.
    Default: refs/heads/master.
    #pipelineId: # string. Alias: runId | buildId. Required when source ==
    specific && runVersion == specific. Build.
    #tags: # string. Optional. Use when source == specific && runVersion !==
    specific. Build Tags.
    #allowPartiallySucceededBuilds: false # boolean. Optional. Use when
    source == specific && runVersion != specific. Download artifacts from
    partially succeeded builds. Default: false.
    #allowFailedBuilds: false # boolean. Optional. Use when source ==
    specific && runVersion != specific. Download artifacts from failed builds.
    Default: false.
    #artifactName: # string. Alias: artifact. Artifact name.
    #itemPattern: '**' # string. Alias: patterns. Matching patterns.
    Default: **.
```

```
targetPath: '$(Pipeline.Workspace)' # string. Alias: path |
downloadPath. Required. Destination directory. Default:
$(Pipeline.Workspace).
```

Inputs

buildType - Download artifacts produced by

Input alias: `source`. `string`. Required. Allowed values: `current` (Current run), `specific` (Specific run). Default value: `current`.

Downloads artifacts produced by the current pipeline run or from a specific pipeline run.

project - Project

`string`. Required when `source == specific`.

Specifies the project name or GUID from which to download the pipeline artifacts.

definition - Build pipeline

Input alias: `pipeline`. `string`. Required when `source == specific`.

The definition ID of the pipeline. In a running pipeline the `definitionId` can be found in the `System.DefinitionId` variable. The `definitionId` can also be retrieved from the URL on the pipeline overview page in the Azure DevOps portal. In the following URL example, the `definitionId` is 78: https://dev.azure.com/fabrikam-inc/FabrikamFiber/_build?definitionId=78&a=summary. To download artifacts from a specific pipeline definition, capture the `definitionId` from that pipeline, and specify it as the `pipeline` parameter.

specificBuildWithTriggering - When appropriate, download artifacts from the triggering build.

Input alias: `preferTriggeringPipeline`. `boolean`. Optional. Use when `source == specific`. Default value: `false`.

If checked, the task downloads artifacts from the triggering build. If there is no triggering build from the specified pipeline, the task downloads artifacts from the build specified in the options below.

`buildVersionToDownload` - Build version to download

Input alias: `runVersion`. `string`. Required when `source == specific`. Allowed values: `latest`, `latestFromBranch` (Latest from specific branch and specified Build Tags), `specific` (Specific version). Default value: `latest`.

Specifies the build version to download.

`branchName` - Branch name

Input alias: `runBranch`. `string`. Required when `source == specific && runVersion == latestFromBranch`. Default value: `refs/heads/master`.

Specifies the filter on the branch/ref name. For example: `refs/heads/develop`.

`pipelineId` - Build

Input alias: `runId | buildId`. `string`. Required when `source == specific && runVersion == specific`.

The identifier of the pipeline run from which to download the artifacts. In a running pipeline the `buildId` can be found in the `Build.BuildId` variable. The `buildId` can also be retrieved from the URL on the pipeline run summary page in the Azure DevOps portal. In the following URL example, the `buildId` is 1088: https://dev.azure.com/fabrikam-inc/FabrikamFiber/_build/results?buildId=1088&view=results. To download artifacts from a specific pipeline run, capture the `buildId` from that run, and specify it as the `buildId` parameter.

`tags` - Build Tags

`string`. Optional. Use when `source == specific && runVersion != specific`.

The comma-delimited list of tags that the task uses to return tagged builds. Untagged builds are not returned.

`allowPartiallySucceededBuilds` - Download artifacts from partially succeeded builds.

`boolean`. Optional. Use when `source == specific && runVersion != specific`. Default value: `false`.

Specifies if the build task downloads artifacts whether the build succeeds or partially succeeds.

`allowFailedBuilds` - Download artifacts from failed builds.

`boolean`. Optional. Use when `source == specific && runVersion != specific`. Default value: `false`.

If checked, the build task downloads artifacts whether the build succeeds or fails.

`artifactName` - Artifact name

Input alias: `artifact`. `string`.

Specifies the name of the artifact to download. If the value is left empty, the task downloads all artifacts associated with the pipeline run.

`itemPattern` - Matching patterns

Input alias: `patterns`. `string`. Default value: `**`.

The file matching patterns that limit downloaded files. The value can be one or more file matching patterns that are new line delimited. Learn more about [file matching patterns](#).

`targetPath` - Destination directory

Input alias: `path` | `downloadPath`. `string`. Required. Default value: `$(Pipeline.Workspace)`.

Specifies either a relative or absolute path on the agent machine where the artifacts will download. If the multi-download option is applied (by leaving an empty artifact name), a sub-directory will be created for each download. Learn more about [Artifacts in Azure Pipelines](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

BuildNumber

Stores the build number of the pipeline artifact source.

Due to backwards compatibility, this variable returns `BuildId`.

Learn more about [build variables](#).

Remarks

Use this task to download pipeline artifacts from earlier stages in this pipeline, or from another pipeline. By default, artifacts are downloaded to `$(Pipeline.Workspace)`. If you don't specify an artifact name, a subdirectory will be created for each downloaded artifact. You can use [file matching patterns](#) to limit the files you want to download.

The [publish](#) and [download](#) keywords are tasks shortcuts to [publish](#) and [download](#) your pipeline artifacts.

How can I find the ID of the Pipeline I want to download an artifact from?

To find the `definitionId` for a specific pipeline definition

In a running pipeline, the `definitionId` can be found in the [System.DefinitionId](#) variable. The `definitionId` can also be retrieved from the URL on the pipeline overview page in the Azure DevOps portal. In the following URL example, the `definitionId` is 78:

https://dev.azure.com/fabrikam-inc/FabrikamFiber/_build?definitionId=78&a=summary.

To download artifacts from a specific pipeline definition, capture the `definitionId` from that pipeline, and specify it as the `pipeline` parameter.

To find the `buildId` for a specific pipeline run

The identifier of the pipeline run from which to download the artifacts. In a running pipeline the `buildId` can be found in the [Build.BuildId](#) variable. The `buildId` can also be retrieved from the URL on the pipeline run summary page in the Azure DevOps portal. In the following URL example, the `buildId` is 1088: https://dev.azure.com/fabrikam-inc/FabrikamFiber/_build/results?buildId=1088&view=results. To download artifacts from a specific pipeline run, capture the `buildId` from that run, and specify it as the `buildId` parameter.

Examples

Download a specific artifact

YAML

```
# Download an artifact named 'WebApp' to 'bin' in $(Build.SourcesDirectory)
- task: DownloadPipelineArtifact@2
  inputs:
    artifactName: 'WebApp'
    targetPath: $(Build.SourcesDirectory)/bin
```

Download artifacts from a specific project/pipeline

YAML

```
# Download artifacts from a specific pipeline.
- task: DownloadPipelineArtifact@2
  inputs:
    buildType: 'specific'
    project: 'FabrikamFiber'
    definition: 12
    buildVersionToDownload: 'latest'
```

Download artifacts from a specific branch

YAML

```
# Download artifacts from a specific branch with a tag
- task: DownloadPipelineArtifact@2
  inputs:
    buildType: 'specific'
    project: 'FabrikamFiber'
    definition: 12
    buildVersionToDownload: 'latestFromBranch'
    branchName: 'refs/heads/master'
    tags: 'testTag'
```

Download an artifact from a specific build run

YAML

```
# Download an artifact named 'WebApp' from a specific build run to 'bin' in
$(Build.SourcesDirectory)
```

```
- task: DownloadPipelineArtifact@2
  inputs:
    buildType: 'specific'
    artifactName: 'WebApp'
    targetPath: $(Build.SourcesDirectory)/bin
    project: 'FabrikamFiber'
    definition: 12
    buildVersionToDownload: 'specific'
    pipelineId: 40
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.164.1 or greater
Task category	Utility

DownloadPipelineArtifact@1 - Download pipeline artifact v1 task

Article • 09/26/2023

Use this task to download pipeline artifacts from earlier stages in this pipeline, or from another pipeline.

This task is deprecated; use [DownloadPipelineArtifact@2](#).

ⓘ Note

For more information, including Azure CLI commands, see [downloading artifacts](#).

Syntax

```
# Download Pipeline Artifacts v1
# Download a named artifact from a pipeline to a local path.
- task: DownloadPipelineArtifact@1
  inputs:
    buildType: 'current' # 'current' | 'specific'. Required. Download
  artifacts produced by. Default: current.
    #project: # string. Required when buildType == specific. Project.
    #pipeline: # string. Alias: definition. Required when buildType ==
  specific. Build pipeline.
    #specificBuildWithTriggering: false # boolean. Optional. Use when
  buildType == specific. When appropriate, download artifacts from the
  triggering build. Default: false.
    #buildVersionToDownload: 'latest' # 'latest' | 'latestFromBranch' |
  'specific'. Required when buildType == specific. Build version to download.
  Default: latest.
    #branchName: 'refs/heads/master' # string. Required when buildType ==
  specific && buildVersionToDownload == latestFromBranch. Branch name.
  Default: refs/heads/master.
    #pipelineId: # string. Alias: buildId. Required when buildType ==
  specific && buildVersionToDownload == specific. Build.
    #tags: # string. Optional. Use when buildType == specific &&
  buildVersionToDownload != specific. Build Tags.
    #artifactName: # string. Artifact name.
    #itemPattern: '**' # string. Matching pattern. Default: **.
    targetPath: '$(System.ArtifactsDirectory)' # string. Alias:
  downloadPath. Required. Destination directory. Default:
  $(System.ArtifactsDirectory).
```

Inputs

buildType - Download artifacts produced by

`string`. Required. Allowed values: `current` (Current build), `specific` (Specific build).

Default value: `current`.

Downloads artifacts produced by the current pipeline run or from a specific pipeline run.

project - Project

`string`. Required when `buildType == specific`.

Specifies the project name or GUID from which to download the pipeline artifacts.

pipeline - Build pipeline

Input alias: `definition`. `string`. Required when `buildType == specific`.

The definition ID of the pipeline. In a running pipeline the `definitionId` can be found in the [System.DefinitionId](#) variable. The `definitionId` can also be retrieved from the URL on the pipeline overview page in the Azure DevOps portal. In the following URL example, the `definitionId` is 78: https://dev.azure.com/fabrikam-inc/FabrikamFiber/_build?definitionId=78&_a=summary. To download artifacts from a specific pipeline definition, capture the `definitionId` from that pipeline, and specify it as the `pipeline` parameter.

specificBuildWithTriggering - When appropriate, download artifacts from the triggering build.

`boolean`. Optional. Use when `buildType == specific`. Default value: `false`.

If checked, the task downloads artifacts from the triggering build. If there is no triggering build from the specified pipeline, the task downloads artifacts from the build specified in the options below.

buildVersionToDownload - Build version to download

`string`. Required when `buildType == specific`. Allowed values: `latest`,

`latestFromBranch` (Latest from specific branch and specified Build Tags), `specific`

(Specific version). Default value: `latest`.

Specifies the build version to download.

branchName - Branch name

`string`. Required when `buildType == specific && buildVersionToDownload == latestFromBranch`. Default value: `refs/heads/master`.

Specifies the filter on the branch/ref name. For example: `refs/heads/develop`.

pipelineId - Build

Input alias: `buildId`. `string`. Required when `buildType == specific && buildVersionToDownload == specific`.

The identifier of the pipeline run from which to download the artifacts. In a running pipeline the `buildId` can be found in the `Build.BuildId` variable. The `buildId` can also be retrieved from the URL on the pipeline run summary page in the Azure DevOps portal. In the following URL example, the `buildId` is 1088: https://dev.azure.com/fabrikam-inc/FabrikamFiber/_build/results?buildId=1088&view=results. To download artifacts from a specific pipeline run, capture the `buildId` from that run, and specify it as the `buildId` parameter.

tags - Build Tags

`string`. Optional. Use when `buildType == specific && buildVersionToDownload != specific`.

The comma-delimited list of tags that the task uses to return tagged builds. Untagged builds are not returned.

artifactName - Artifact name

`string`.

Specifies the name of the artifact to download. If the value is left empty, the task downloads all artifacts associated with the pipeline run.

itemPattern - Matching pattern

`string`. Default value: `**`.

The file matching patterns that limit downloaded files. The value can be one or more file matching patterns that are new line delimited. Learn more about [file matching patterns](#).

`targetPath` - Destination directory

Input alias: `downloadPath`. `string`. Required. Default value:

`$(System.ArtifactsDirectory)`.

The path on the agent machine where the artifacts will be downloaded.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

By default, artifacts are downloaded to `$(Pipeline.Workspace)`. If you don't specify an artifact name, a subdirectory will be created for each downloaded artifact. You can use [file matching patterns](#) to limit the files you want to download.

How can I find the ID of the Pipeline I want to download an artifact from?

To find the `definitionId` for a specific pipeline definition

In a running pipeline, the `definitionId` can be found in the [System.DefinitionId](#) variable. The `definitionId` can also be retrieved from the URL on the pipeline overview page in the Azure DevOps portal. In the following URL example, the `definitionId` is 78:

`https://dev.azure.com/fabrikam-inc/FabrikamFiber/_build?definitionId=78&a=summary`.

To download artifacts from a specific pipeline definition, capture the `definitionId` from that pipeline, and specify it as the `pipeline` parameter.

To find the `buildId` for a specific pipeline run

The identifier of the pipeline run from which to download the artifacts. In a running pipeline the `buildId` can be found in the [Build.BuildId](#) variable. The `buildId` can also be retrieved from the URL on the pipeline run summary page in the Azure DevOps portal.

In the following URL example, the `buildId` is 1088: https://dev.azure.com/fabrikam-inc/FabrikamFiber/_build/results?buildId=1088&view=results. To download artifacts from a specific pipeline run, capture the `buildId` from that run, and specify it as the `buildId` parameter.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.155.1 or greater
Task category	Utility

DownloadPipelineArtifact@0 - Download pipeline artifact v0 task

Article • 09/26/2023

Use this task to download pipeline artifacts from earlier stages in this pipeline, or from another pipeline.

This task is deprecated; use [DownloadPipelineArtifact@2](#).

ⓘ Note

For more information, including Azure CLI commands, see [downloading artifacts](#).

Syntax

YAML

```
# Download Pipeline Artifacts v0
# Downloads an artifact associated with a pipeline.
- task: DownloadPipelineArtifact@0
  inputs:
    #pipelineId: # string. The specific pipeline to download from.
    artifactName: 'drop' # string. Required. The name of artifact to
download. Default: drop.
    targetPath: # string. Required. Path to download to.
```

Inputs

`pipelineId` - The specific pipeline to download from

`string`.

The build from which to download the artifacts. For example: `1764`. If missing, target the current pipeline.

`artifactName` - The name of artifact to download.

`string`. Required. Default value: `drop`.

Specifies the name of the artifact to download. If the value is left empty, the task downloads all artifacts associated with the pipeline run.

`targetPath` - Path to download to

`string`. Required.

The folder path to download the artifact to. This can be a fully-qualified path or a path relative to the root of the repository. Wildcards are not supported. [Variables ↗](#) are supported. If the folder doesn't exist it will be created.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

By default, artifacts are downloaded to `$(Pipeline.Workspace)`. If you don't specify an artifact name, a subdirectory will be created for each downloaded artifact. You can use [file matching patterns](#) to limit the files you want to download.

How can I find the ID of the Pipeline I want to download an artifact from?

You can find the ID of the pipeline in the 'Pipeline variables'. The pipeline ID is the [system.definitionId](#) variable. You can also find it in the URL path.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None

Requirement	Description
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.155.1 or greater
Task category	Utility

DownloadSecureFile@1 - Download secure file v1 task

Article • 09/26/2023

Use this task to download a secure file to the agent machine.

Syntax

YAML

```
# Download secure file v1
# Download a secure file to the agent machine.
- task: DownloadSecureFile@1
  inputs:
    secureFile: # string. Required. Secure File.
    #retryCount: '8' # string. Retry Count. Default: 8.
    #socketTimeout: # string. Socket Timeout.
```

Inputs

secureFile - Secure File

`string`. Required.

Specifies the name or unique identifier (GUID) of the secure file that is downloaded to the agent machine. The file is deleted when the pipeline job completes.

retryCount - Retry Count

`string`. Default value: `8`.

Optional. Specifies the number of times to retry downloading a secure file if the download fails.

socketTimeout - Socket Timeout

`string`.

Optional. When downloading a secure file request in Microsoft, this input specifies the timeout for a socket.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

`secureFilePath`

Specifies the location of the secure file that was downloaded.

Remarks

Use this task in a pipeline to download a [secure file](#) to the agent machine. When specifying the name of the file (using the `secureFile` input), use the name you specified when uploading it, rather than the actual file name.

Once downloaded, use the `name` value that is set on the task (or "Reference name" in the classic editor) to reference the path to the secure file on the agent machine. For example, if the task is given the name `mySecureFile`, its path can be referenced in the pipeline as `$(mySecureFile.secureFilePath)`. Alternatively, downloaded secure files can be found in the directory given by `$(Agent.TempDirectory)`. See a full example [below](#).

When the pipeline job completes, whether it succeeds, fails, or is canceled, the secure file is deleted from its download location.

It is unnecessary to use this task with the [Install Apple Certificate](#) or [Install Apple Provisioning Profile](#) tasks because they automatically download, install, and delete (at the end of the pipeline job) the secure file.

This task currently supports only one file task per instance.

Examples

This example downloads a secure certificate file and installs it to a trusted certificate authority (CA) directory on Linux:

YAML

```

- task: DownloadSecureFile@1
  name: caCertificate
  displayName: 'Download CA certificate'
  inputs:
    secureFile: 'myCACertificate.pem'

- script: |
  echo Installing $(caCertificate.secureFilePath) to the trusted CA
  directory...
  sudo chown root:root $(caCertificate.secureFilePath)
  sudo chmod a+r $(caCertificate.secureFilePath)
  sudo ln -s $(caCertificate.secureFilePath) /etc/ssl/certs/

```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : secureFilePath
Agent version	2.182.1 or greater
Task category	Utility

DuffleInstaller@0 - Duffle tool installer v0 task

Article • 09/26/2023

Use this task to install a specified version of Duffle, which is used for installing and managing CNAB bundles.

Syntax

YAML

```
# Duffle tool installer v0
# Install a specified version of Duffle for installing and managing CNAB
bundles.
- task: DuffleInstaller@0
  inputs:
    version: '0.1.0-ralpha.4+dramallamabuie' # string. Required. Version.
  Default: 0.1.0-ralpha.4+dramallamabuie.
    #checkLatestVersion: false # boolean. Check for latest version. Default:
  false.
```

Inputs

`version` - Version

`string`. Required. Default value: `0.1.0-ralpha.4+dramallamabuie`.

Specifies the version of Duffle to install.

`checkLatestVersion` - Check for latest version

`boolean`. Default value: `false`.

Always checks online for the latest available version (stable.txt) that satisfies the version spec. Typically, the default value, `false`, is used unless you have a specific scenario where the latest update is always needed. This may incur download costs when potentially not necessary, especially with the hosted build pool.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Tool

ExtractFiles@1 - Extract files v1 task

Article • 09/26/2023

Use this task to extract a variety of archive and compression files, such as .7z, .rar, .tar.gz, and .zip.

Syntax

YAML

```
# Extract files v1
# Extract a variety of archive and compression files such as .7z, .rar,
.tar.gz, and .zip.
- task: ExtractFiles@1
  inputs:
    archiveFilePatterns: '**/*.zip' # string. Required. Archive file
patterns. Default: **/*.zip.
    destinationFolder: # string. Required. Destination folder.
    #cleanDestinationFolder: true # boolean. Clean destination folder before
extracting. Default: true.
    #overwriteExistingFiles: false # boolean. Overwrite existing files.
Default: false.
    #pathToSevenZipTool: # string. Path to 7z utility.
```

Inputs

`archiveFilePatterns` - Archive file patterns

`string`. Required. Default value: `**/*.zip`.

Specifies the file paths or patterns of the archive files to extract. Supports multiple lines of minimatch patterns. Learn more about the [Extract Files task](#).

Specifies the patterns to match the archives you want to extract. By default, patterns start in the root folder of the repo (same as if you had specified

`$(Build.SourcesDirectory)`.

Specifies the pattern filters, one per line, that match the archives to extract. For example:

- `test.zip` extracts the test.zip file in the root folder.
- `test/*.zip` extracts all .zip files in the test folder.
- `**/*.tar` extracts all .tar files in the root folder and sub-folders.
- `**/bin/**.7z` extracts all .7z files in any sub-folder named "bin".

The pattern is used to match only archive file paths, not folder paths, and not

archive contents to be extracted. So, you should specify patterns, such as `**/bin/**` instead of `**/bin`.

`destinationFolder` - Destination folder

`string`. Required.

Specifies the destination folder into which archive files should be extracted. Use [variables](#) if files are not in the repo. For example: `$(agent.builddirectory)`.

`cleanDestinationFolder` - Clean destination folder before extracting

`boolean`. Default value: `true`.

Specifies the option to clean the destination directory before archive contents are extracted into it.

`overwriteExistingFiles` - Overwrite existing files

`boolean`. Default value: `false`.

Specifies the option to overwrite existing files in the destination directory if they already exist. If the option is `false`, the script prompts on existing files, asking whether you want to overwrite them.

`pathToSevenZipTool` - Path to 7z utility

`string`.

Specifies the custom path to 7z utility. For example, `c:\7z\7z.exe` on Windows and `/usr/local/bin/7z` on MacOS/Ubuntu. If it's not specified on Windows, the default 7zip version supplied with a task will be used.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to extract files from archives to a target folder using match patterns. A range of standard archive formats is supported, including .zip, .jar, .war, .ear, .tar, .7z, and more.

For more information about file matching patterns, see the [File matching patterns reference](#).

Examples

Extract all .zip files recursively

This example will extract all .zip files recursively, including both root files and files from sub-folders.

YAML

```
steps:
- task: ExtractFiles@1
  inputs:
    archiveFilePatterns: '**/*.zip'
    cleanDestinationFolder: true
    overwriteExistingFiles: false
```

Extract all .zip files from subfolder

This example will extract `test/one.zip` and `test/two.zip`, but will leave `test/nested/three.zip`.

YAML

```
steps:
- task: ExtractFiles@1
  inputs:
    archiveFilePatterns: 'test/*.zip'
    cleanDestinationFolder: true
    overwriteExistingFiles: false
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : Setting variables is disabled
Agent version	2.182.1 or greater
Task category	Utility

See also

- [File matching patterns reference](#)

FileTransform@2 - File transform v2 task

Article • 09/26/2023

Use this task to replace tokens with variable values in XML or JSON configuration files.

Syntax

YAML

```
# File transform v2
# Replace tokens with variable values in XML or JSON configuration files.
- task: FileTransform@2
  inputs:
    filePath: '$(System.DefaultWorkingDirectory)/**/*.zip' # string.
    Required. Package or folder. Default:
    $(System.DefaultWorkingDirectory)/**/*.zip.
    #xmlTransformationRules: '-transform **\*.Release.config -xml
    **\*.config' # string. XML Transformation rules. Default: -transform
    **\*.Release.config -xml **\*.config.
    # Variable Substitution
    #jsonTargetFiles: # string. JSON target files.
    #xmlTargetFiles: # string. XML target files.
```

Inputs

filePath - Package or folder

`string`. Required. Default value: `$(System.DefaultWorkingDirectory)/**/*.zip`.

File path to the package or a folder.

Variables are [Build](#) and [Release](#). Wildcards are supported.

For example, `$(System.DefaultWorkingDirectory)/**/*.zip`. For zipped folders, the contents are extracted to the TEMP location, transformations executed, and the results zipped in original artifact location.

xmlTransformationRules - XML Transformation rules

`string`. Default value: `-transform ***.Release.config -xml ***.config`.

Provides a newline-separated list of transformation file rules using the syntax: `-transform <pathToTransformFile> -xml <pathToSourceConfigurationFile>`. The result file

path is optional, and if not specified, the source configuration file will be replaced with the transformed result file.

`jsonTargetFiles` - JSON target files

`string`.

Provides a newline-separated list of files to substitute the variable values. File names are to be provided relative to the root folder.

For example, to replace the value of `ConnectionString` in the sample below, you need to define a variable as `Data.DefaultConnection.ConnectionString` in the build or release pipeline (or release pipeline's environment).

JSON

```
{  
  "Data": {  
    "DefaultConnection": {  
      "ConnectionString": "Server=  
      (localdb)\SQLEXPRESS;Database=MyDB;Trusted_Connection=True"  
    }  
  }  
}
```

Variable Substitution is run after configuration transforms.

Note: Only custom variables that are defined in build/release pipelines are used in substitution. Default/system defined pipeline variables are excluded. If the same variables are defined in the release pipeline and in the stage, then the stage variables will supersede the release pipeline variables.

`xmlTargetFiles` - XML target files

`string`.

Provides a newline-separated list of files to substitute the variable values. File names are to be provided relative to the root folder.

For XML, Variables defined in the build or release pipelines will be matched against the `key` or `name` entries in the `appSettings`, `applicationSettings`, and `connectionStrings` sections of any config file and `parameters.xml`.

Variable Substitution is run after configuration transforms.

Note: Only custom variables defined in build/release pipelines are used in substitution. Default/system defined pipeline variables are excluded. If the same variables are defined in the release pipeline and in the stage, then the stage variables will supersede the release pipeline variables.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

What's new in File Transform version 2:

- More optimized task fields that allow users to enable any/all of the transformation (XML), variable substitution (JSON and XML) features in a single task instance.
- Task fails when any of the configured transformation/substitution is NOT applied or when the task is no-op.

Use this task to apply file transformations and variable substitutions on configuration and parameters files. For details of how translations are processed, see [File transforms and variable substitution reference](#).

Important

This task is intended for web packages and requires a web package file. It does not work on standalone JSON files.

File transformations

- At present, file transformations are supported for only XML files.
- To apply an XML transformation to configuration files (*.config) you must specify a newline-separated list of transformation file rules using the syntax:
`-t transform
<path to the transform file> -xml <path to the source file> -result <path to
the result file>`

- File transformations are useful in many scenarios, particularly when you are deploying to an App service and want to add, remove or modify configurations for different environments (such as Dev, Test, or Prod) by following the standard [Web.config Transformation Syntax](#).
- You can also use this functionality to transform other files, including Console or Windows service application configuration files (for example, `FabrikamService.exe.config`).
- Config file transformations are run before variable substitutions.

Variable substitution

- At present only XML and JSON file formats are supported for variable substitution.
- Tokens defined in the target configuration files are updated and then replaced with variable values.
- Variable substitutions are run after config file transformations.
- Variable substitution is applied for only the JSON keys predefined in the object hierarchy. It does not create new keys.

Note

Only custom variables defined in build and release pipelines are used in substitution. Default and system pipeline variables are excluded.

Here's a list of currently excluded prefixes:

- `agent.`
- `azure_http_user_agent`
- `build.`
- `common.`
- `release.`
- `system.`
- `tf_`

If the same variables are defined in both the release pipeline and in a stage, the stage-defined variables supersede the pipeline-defined variables.

See also: [File transforms and variable substitution reference](#).

Examples

If you need XML transformation to run on all the configuration files named with pattern `.Production.config`, the transformation rule should be specified as:

```
-transform **\*.Production.config -xml **\*.config
```

If you have a configuration file named based on the stage name in your pipeline, you can use:

```
-transform **\*.$(Release.EnvironmentName).config -xml **\*.config
```

To substitute JSON variables that are nested or hierarchical, specify them using JSONPath expressions. For example, to replace the value of **ConnectionString** in the sample below, you must define a variable as `Data.DefaultConnection.ConnectionString` in the build or release pipeline (or in a stage within the release pipeline).

```
{
  "Data": {
    "DefaultConnection": {
      "ConnectionString": "Server=(localdb)\SQLEXPRESS;Database=MyDB;Trusted_Connection=True"
    }
  }
}
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Utility

See also

- [File transforms and variable substitution reference](#)

FileTransform@1 - File transform v1 task

Article • 09/26/2023

Use this task to replace tokens with variable values in XML or JSON configuration files.

Syntax

YAML

```
# File transform v1
# Replace tokens with variable values in XML or JSON configuration files.
- task: FileTransform@1
  inputs:
    filePath: '$(System.DefaultWorkingDirectory)/**/*.zip' # string.
    Required. Package or folder. Default:
    $(System.DefaultWorkingDirectory)/**/*.zip.
    enableXmlTransform: false # boolean. XML transformation. Default:
    false.
    xmlTransformationRules: '-transform **\*.Release.config -xml
    **\*.config' # string. Optional. Use when enableXmlTransform == true.
    Transformation rules. Default: -transform **\*.Release.config -xml
    **\*.config.
    # Variable Substitution
    fileType: # 'xml' | 'json'. File format.
    targetFiles: # string. Optional. Use when fileType = xml || fileType =
    json. Target files.
```

Inputs

`filePath` - Package or folder

`string`. Required. Default value: `$(System.DefaultWorkingDirectory)/**/*.zip`.

The file path to the package or a folder.

Variables are [Build](#) and [Release](#). Wildcards are supported.

For example, `$(System.DefaultWorkingDirectory)/**/*.zip`.

`enableXmlTransform` - XML transformation

`boolean`. Default value: `false`.

Config transforms will be run prior to the Variable Substitution.

XML transformations are supported only for Windows platform.

xmlTransformationRules - Transformation rules

`string`. Optional. Use when `enableXmlTransform == true`. Default value: `-transform ***.Release.config -xml ***.config`.

Provides a new line separated list of transformation file rules using the syntax:

```
-transform <pathToTransformFile> -xml <pathToSourceConfigurationFile>.
```

fileType - File format

`string`. Allowed values: `xml`, `json`.

Provides the file format on which the substitution is performed.

For XML, variables defined in the build or release pipelines will be matched against the `key` or `name` entries in the `appSettings`, `applicationSettings`, and `connectionStrings` sections of any config file and `parameters.xml`. `Variable Substitution` is run after config transforms.

To substitute JSON variables that are nested or hierarchical, specify them using JSONPath expressions.

For example, to replace the value of `ConnectionString` in the sample below, you need to define a variable as `Data.DefaultConnection.ConnectionString` in the build or release pipeline (or release pipeline's environment).

```
JSON

{
  "Data": {
    "DefaultConnection": {
      "ConnectionString": "Server=(localdb)\SQLEXPRESS;Database=MyDB;Trusted_Connection=True"
    }
  }
}
```

Variable Substitution is run after configuration transforms.

Note: Only custom variables that are defined in build/release pipelines are used in substitution. Default/system defined pipeline variables are excluded. If the same

variables are defined in the release pipeline and in the stage, then the stage variables will supersede the release pipeline variables.

targetFiles - Target files

`string`. Optional. Use when `fileType = xml || fileType = json`.

Provides a newline-separated list of files to substitute the variable values. File names are to be provided relative to the root folder.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Utility

FtpUpload@2 - FTP upload v2 task

Article • 09/26/2023

Use this task to upload files to a remote machine using FTP or securely with FTPS.

Syntax

YAML

```
# FTP upload v2
# Upload files using FTP.
- task: FtpUpload@2
  inputs:
    credentialsOption: 'serviceEndpoint' # 'serviceEndpoint' | 'inputs'.
    Alias: credsType. Required. Authentication Method. Default: serviceEndpoint.
    serverEndpoint: # string. Required when credsType = serviceEndpoint. FTP
    Service Connection.
    #serverUrl: # string. Required when credsType = inputs. Server URL.
    #username: # string. Required when credsType = inputs. Username.
    #password: # string. Required when credsType = inputs. Password.
    rootDirectory: # string. Alias: rootFolder. Required. Root folder.
    filePatterns: '**' # string. Required. File patterns. Default: **.
    remoteDirectory: '/upload/${Build.BuildId}/' # string. Alias:
    remotePath. Required. Remote directory. Default: /upload/${Build.BuildId}/.
    # Advanced
    #enableUtf8: false # boolean. Enable UTF8 support. Default: false.
    #clean: false # boolean. Delete remote directory. Default: false.
    #cleanContents: false # boolean. Optional. Use when clean = false. Clear
    remote directory contents. Default: false.
    #preservePaths: false # boolean. Preserve file paths. Default: false.
    #trustSSL: false # boolean. Trust server certificate. Default: false.
    #customCmds: # string. FTP Commands.
```

Inputs

`credentialsOption` - Authentication Method

Input alias: `credsType`. `string`. Required. Allowed values: `serviceEndpoint` (FTP service connection), `inputs` (Enter credentials). Default value: `serviceEndpoint`.

Specifies the authentication method. Use an FTP service connection or enter the connection credentials.

`serverEndpoint` - FTP Service Connection

`string`. Required when `credsType = serviceEndpoint`.

Specifies the service connection for the FTP server. To create one, click the Manage link and create a new generic service connection, and then enter the FTP server URL for the server URL, e.g. `ftp://server.example.com`, and the required credentials.

Secure connections will always be made regardless of the specified protocol (`ftp://` or `ftps://`) if the target server supports FTPS. To allow only secure connections, use the `ftps://` protocol, e.g. `ftps://server.example.com`. Connections to servers not supporting FTPS will fail if `ftps://` is specified.

serverUrl - Server URL

`string`. Required when `credsType = inputs`.

Specifies the URL for the FTP server.

username - Username

`string`. Required when `credsType = inputs`.

Specifies the user name for the FTP connection.

password - Password

`string`. Required when `credsType = inputs`.

Specifies the password for the FTP connection.

rootDirectory - Root folder

Input alias: `rootFolder`. `string`. Required.

Specifies the source folder to upload files from.

filePatterns - File patterns

`string`. Required. Default value: `**`.

Specifies the file paths or patterns of the files to upload. The string supports multiple lines of minimatch patterns. Learn more about [file matching patterns](#).

remoteDirectory - Remote directory

Input alias: `remotePath`. `string`. Required. Default value: `/upload/$(Build.BuildId)/`.

Specifies the directory on the remote FTP server where the task uploads files.

enableUtf8 - Enable UTF8 support

`boolean`. Default value: `false`.

Enables UTF-8 support for the FTP connection (`OPTS UTF8 ON`).

clean - Delete remote directory

`boolean`. Default value: `false`.

Deletes the remote directory, including its contents, before uploading.

cleanContents - Clear remote directory contents

`boolean`. Optional. Use when `clean = false`. Default value: `false`.

Recursively deletes all content in the remote directory before uploading. The existing directory will not be deleted. For better performance, use `clean` instead.

preservePaths - Preserve file paths

`boolean`. Default value: `false`.

If selected, the relative local directory structure is recreated under the remote directory where files are uploaded. Otherwise, files are uploaded directly to the remote directory without creating additional subdirectories.

For example, suppose your source folder is `/home/user/source/`, which contains the file `foo/bar/foobar.txt`, and your remote directory is: `/uploads/`. If this boolean is selected, the file is uploaded to `/uploads/foo/bar/foobar.txt`. If this boolean is not selected, the file is uploaded to `/uploads/foobar.txt`.

trustSSL - Trust server certificate

`boolean`. Default value: `false`.

Trusts the FTP server's SSL certificate with `ftps://`, even if it is self-signed or cannot be validated by a certificate authority (CA).

customCmds - FTP Commands

`string`.

The optional FTP commands that will be sent to the remote FTP server upon connection.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to upload files to a remote machine using FTP or securely with FTPS.

Where can I learn more about file matching patterns?

- [File matching patterns reference](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : Setting variables is disabled
Agent version	2.182.1 or greater
Task category	Utility

See also

- File matching patterns reference

FtpUpload@1 - FTP upload v1 task

Article • 09/26/2023

Use this task to upload files to a remote machine using FTP or securely with FTPS.

Syntax

YAML

```
# FTP upload v1
# Upload files using FTP.
- task: FtpUpload@1
  inputs:
    credentialsOption: 'serviceEndpoint' # 'serviceEndpoint' | 'inputs'.
    Alias: credsType. Required. Authentication Method. Default: serviceEndpoint.
    serverEndpoint: # string. Required when credsType = serviceEndpoint. FTP
    Service Connection.
    #serverUrl: # string. Required when credsType = inputs. Server URL.
    #username: # string. Required when credsType = inputs. Username.
    #password: # string. Required when credsType = inputs. Password.
    rootDirectory: # string. Alias: rootFolder. Required. Root folder.
    filePatterns: '**' # string. Required. File patterns. Default: **.
    remoteDirectory: '/upload/${Build.BuildId}/' # string. Alias:
    remotePath. Required. Remote directory. Default: /upload/${Build.BuildId}/.
    # Advanced
    #clean: false # boolean. Delete remote directory. Default: false.
    #cleanContents: false # boolean. Optional. Use when clean = false. Clear
    remote directory contents. Default: false.
    #overwrite: true # boolean. Overwrite. Default: true.
    #preservePaths: false # boolean. Preserve file paths. Default: false.
    #trustSSL: false # boolean. Trust server certificate. Default: false.
```

Inputs

`credentialsOption` - Authentication Method

Input alias: `credsType`. `string`. Required. Allowed values: `serviceEndpoint` (FTP service connection), `inputs` (Enter credentials). Default value: `serviceEndpoint`.

Specifies the authentication method. Use an FTP service connection or enter the connection credentials.

`serverEndpoint` - FTP Service Connection

`string`. Required when `credsType = serviceEndpoint`.

Specifies the service connection for the FTP server. To create one, click the Manage link and create a new generic service connection, and then enter the FTP server URL for the server URL, e.g. `ftp://server.example.com`, and the required credentials.

Secure connections will always be made regardless of the specified protocol (`ftp://` or `ftps://`) if the target server supports FTPS. To allow only secure connections, use the `ftps://` protocol, e.g. `ftps://server.example.com`. Connections to servers not supporting FTPS will fail if `ftps://` is specified.

serverUrl - Server URL

`string`. Required when `credsType = inputs`.

Specifies the URL for the FTP server.

username - Username

`string`. Required when `credsType = inputs`.

Specifies the user name for the FTP connection.

password - Password

`string`. Required when `credsType = inputs`.

Specifies the password for the FTP connection.

rootDirectory - Root folder

Input alias: `rootFolder`. `string`. Required.

Specifies the source folder to upload files from.

filePatterns - File patterns

`string`. Required. Default value: `**`.

Specifies the file paths or patterns of the files to upload. The string supports multiple lines of minimatch patterns. Learn more about [file matching patterns](#).

remoteDirectory - Remote directory

Input alias: `remotePath`. `string`. Required. Default value: `/upload/$(Build.BuildId)/`.

Specifies the directory on the remote FTP server where the task uploads files.

clean - Delete remote directory

`boolean`. Default value: `false`.

Deletes the remote directory, including its contents, before uploading.

cleanContents - Clear remote directory contents

`boolean`. Optional. Use when `clean = false`. Default value: `false`.

Recursively deletes all content in the remote directory before uploading. The existing directory will not be deleted. For better performance, use `clean` instead.

overwrite - Overwrite

`boolean`. Default value: `true`.

Overwrites existing files in the remote directory.

preservePaths - Preserve file paths

`boolean`. Default value: `false`.

If selected, the relative local directory structure is recreated under the remote directory where files are uploaded. Otherwise, files are uploaded directly to the remote directory without creating additional subdirectories.

For example, suppose your source folder is `/home/user/source/`, which contains the file `foo/bar/foobar.txt`, and your remote directory is: `/uploads/`. If this boolean is selected, the file is uploaded to `/uploads/foo/bar/foobar.txt`. If this boolean is not selected, the file is uploaded to `/uploads/foobar.txt`.

trustSSL - Trust server certificate

`boolean`. Default value: `false`.

Trusts the FTP server's SSL certificate with `ftps://`, even if it is self-signed or cannot be validated by a certificate authority (CA).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to upload files to a remote machine using FTP or securely with FTPS.

Where can I learn more about file matching patterns?

- [File matching patterns reference](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : Setting variables is disabled
Agent version	2.182.1 or greater
Task category	Utility

GitHubComment@0 - GitHub Comment v0 task

Article • 09/26/2023

Use this task to write a comment to your GitHub entity, for example an issue or a Pull Request (PR).

Syntax

YAML

```
# GitHub Comment v0
# Write a comment to your GitHub entity i.e. issue or a pull request (PR).
- task: GitHubComment@0
  inputs:
    gitHubConnection: # string. Required. GitHub connection (OAuth or PAT).
    repositoryName: '${Build.Repository.Name}' # string. Required.
    Repository. Default: ${Build.Repository.Name}.
    #id: # string. ID of the github pr/issue.
    #comment: # string. Comment.
```

Inputs

`gitHubConnection` - GitHub connection (OAuth or PAT)

`string`. Required.

Specifies the name of the GitHub service connection to use to connect to the GitHub repository. The connection must be based on a GitHub user's OAuth or a GitHub personal access token. For more information about service connections, see [Manage service connections](#).

`repositoryName` - Repository

`string`. Required. Default value: `$(Build.Repository.Name)`.

Specifies the name of the GitHub repository where the GitHub comment will be created.

`id` - ID of the github pr/issue

`string`.

Specifies the issue or PR number. If used in a PR pipeline, leave this field empty to dynamically figure out the ID.

comment - Comment

`string`.

The contents of the comment to be written.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.0.0 or greater
Task category	Utility

GitHubRelease@1 - GitHub Release v1 task

Article • 09/26/2023

Use this task to create, edit, or delete a GitHub release.

Syntax

YAML

```
# GitHub Release v1
# Create, edit, or delete a GitHub release.
- task: GitHubRelease@1
  inputs:
    gitHubConnection: # string. Required. GitHub connection (OAuth or PAT).
    repositoryName: '$(Build.Repository.Name)' # string. Required.
    Repository. Default: $(Build.Repository.Name).
    action: 'create' # 'create' | 'edit' | 'delete'. Required. Action.
    Default: create.
    #target: '$(Build.SourceVersion)' # string. Required when action = create || action = edit. Target. Default: $(Build.SourceVersion).
    tagSource: 'gitTag' # 'gitTag' | 'userSpecifiedTag'. Required when action = create. Tag source. Default: gitTag.
    #tagPattern: # string. Optional. Use when tagSource = gitTag. Tag Pattern.
    #tag: # string. Required when action = edit || action = delete || tagSource = userSpecifiedTag. Tag.
    #title: # string. Optional. Use when action = create || action = edit. Release title.
    #releaseNotesSource: 'filePath' # 'filePath' | 'inline'. Optional. Use when action = create || action = edit. Release notes source. Default: filePath.
    #releaseNotesFilePath: # string. Optional. Use when releaseNotesSource = filePath. Release notes file path.
    #releaseNotesInline: # string. Optional. Use when releaseNotesSource = inline. Release notes.
    #assets: '$(Build.ArtifactStagingDirectory)/*' # string. Optional. Use when action = create || action = edit. Assets. Default: $(Build.ArtifactStagingDirectory)/*.
    #assetUploadMode: 'delete' # 'delete' | 'replace'. Optional. Use when action = edit. Asset upload mode. Default: delete.
    #isDraft: false # boolean. Optional. Use when action = create || action = edit. Draft release. Default: false.
    #isPreRelease: false # boolean. Optional. Use when action = create || action = edit. Pre-release. Default: false.
    #addChangeLog: true # boolean. Optional. Use when action = create || action = edit. Add changelog. Default: true.
    # Changelog configuration
    changeLogCompareToRelease: 'lastFullRelease' # 'lastFullRelease' |
```

```
'lastNonDraftRelease' | 'lastNonDraftReleaseByTag'. Required when
addChangeLog = true. Compare to. Default: lastFullRelease.
  #changeLogCompareToReleaseTag: # string. Required when
  changeLogCompareToRelease = lastNonDraftReleaseByTag && addChangeLog = true.
  Release Tag.
    changeLogType: 'commitBased' # 'commitBased' | 'issueBased'. Required
    when addChangeLog = true. Changelog type. Default: commitBased.
    #changeLogLabels: '[{ "label" : "bug", "displayName" : "Bugs", "state" :
    "closed" }]' # string. Optional. Use when changeLogType = issueBased &&
    addChangeLog = true. Categories. Default: [{ "label" : "bug", "displayName"
    : "Bugs", "state" : "closed" }].
```

Inputs

gitHubConnection - GitHub connection (OAuth or PAT)

`string`. Required.

Specifies the name of the GitHub service connection to use to connect to the GitHub repository. The connection must be based on a GitHub user's OAuth or a GitHub personal access token. For more information about service connections, see [Manage service connections](#).

repositoryName - Repository

`string`. Required. Default value: `$(Build.Repository.Name)`.

Specifies the name of the GitHub repository where you will create, edit, or delete the GitHub release.

action - Action

`string`. Required. Allowed values: `create`, `edit`, `delete`. Default value: `create`.

Specifies the type of release operation to perform. This task can create, edit, or delete a GitHub release.

target - Target

`string`. Required when `action = create || action = edit`. Default value: `$(Build.SourceVersion)`.

Specifies the commit SHA you want to use to create the GitHub release, for example `48b11d8d6e92a22e3e9563a3f643699c16fd6e27`. You can also use a variable, like `$(myCommitSHA)`, in this field.

tagSource - Tag source

`string`. Required when `action = create`. Allowed values: `gitTag` (Git tag), `userSpecifiedTag` (User specified tag). Default value: `gitTag`.

Specifies the tag you want to use for release creation. The `gitTag` option automatically uses the tag that is associated with the Git commit. Use the `userSpecifiedTag` option to manually provide a tag.

tagPattern - Tag Pattern

`string`. Optional. Use when `tagSource = gitTag`.

Specifies the Git tag pattern by using regex, for example `release-v1.*`. A GitHub release will be created only for commits that have matching Git tag.

tag - Tag

`string`. Required when `action = edit || action = delete || tagSource = userSpecifiedTag`.

Specifies the tag you want to use when you create, edit, or delete a release. You can also use a variable, like `$(myTagName)`, in this field.

title - Release title

`string`. Optional. Use when `action = create || action = edit`.

Specifies the title of the GitHub release. If left empty, the tag will be used as the release title.

releaseNotesSource - Release notes source

`string`. Optional. Use when `action = create || action = edit`. Allowed values: `filePath` (Release notes file), `inline` (Inline release notes). Default value: `filePath`.

Specifies the description of the GitHub release. Use the `filePath` (Release notes file) option to use file contents as release notes. Use the `inline` (Inline release notes) option to manually enter release notes

releaseNotesFilePath - Release notes file path

`string`. Optional. Use when `releaseNotesSource = filePath`.

Specifies the file that contains the release notes.

releaseNotesInline - Release notes

`string`. Optional. Use when `releaseNotesSource = inline`.

Specifies the release notes. Markdown is supported.

assets - Assets

`string`. Optional. Use when `action = create || action = edit`. Default value:

`$(Build.ArtifactStagingDirectory)/*`.

Specifies the files you want to upload as assets of the release. You can use wildcard characters to specify multiple files. For example, use

`$(Build.ArtifactStagingDirectory)/*.zip` or use

`$(System.DefaultWorkingDirectory)/*.zip` for release pipelines.

You can also specify multiple patterns, one per line. By default, all files in the `$(Build.ArtifactStagingDirectory)` directory will be uploaded. For more information about the list of pre-defined variables that are available, see [build variables](#) and [release variables](#).

assetUploadMode - Asset upload mode

`string`. Optional. Use when `action = edit`. Allowed values: `delete` (Delete existing assets), `replace` (Replace existing assets). Default value: `delete`.

Specifies the asset upload mode you want to use. Use the `delete` (Delete existing assets) option to first delete any existing assets in the release and then upload all assets. Use the `replace` (Replace existing assets) option to replace any assets that have the same name.

isDraft - Draft release

`boolean`. Optional. Use when `action = create || action = edit`. Default value: `false`.

Indicates whether the release should be saved as a draft (unpublished). If `false`, the release will be published.

`isPreRelease` - Pre-release

`boolean`. Optional. Use when `action = create || action = edit`. Default value: `false`.

Indicates whether the release should be marked as a pre-release.

`addChangeLog` - Add changelog

`boolean`. Optional. Use when `action = create || action = edit`. Default value: `true`.

Specifies if you want to include a changelog. If set to `true`, a list of changes (commits and issues) between the current release and the last published release will be generated and appended to the release notes.

`changeLogCompareToRelease` - Compare to

`string`. Required when `addChangeLog = true`. Allowed values: `lastFullRelease` (Last full release), `lastNonDraftRelease` (Last non-draft release), `lastNonDraftReleaseByTag` (Last non-draft release by tag). Default value: `lastFullRelease`.

Indicates which release to compare with to generate the changelog:

- `lastFullRelease` (Last full release): Compares the current release with the most recent non-draft release that is not marked as pre-release.
- `lastNonDraftRelease` (Last non-draft release): Compares the current release with the most recent non-draft release.
- `lastNonDraftReleaseByTag` (Last non-draft release by tag): Compares the current release with the last non-draft release matching the specified tag. You can also specify a regex instead of an exact tag.

`changeLogCompareToReleaseTag` - Release Tag

`string`. Required when `changeLogCompareToRelease = lastNonDraftReleaseByTag & addChangeLog = true`.

Specifies the regex for release tag. Release matching this tag will be used as base for changelog computation.

`changeLogType` - Changelog type

`string`. Required when `addChangeLog = true`. Allowed values: `commitBased` (Commit based), `issueBased` (Issue based). Default value: `commitBased`.

Specifies the changelog type. A changelog can be commit-based or issue-based. A commit-based changelog lists all commits included in a release. An issue-based changelog lists all the issues or pull requests (PRs) included in the release.

`changeLogLabels` - Categories

`string`. Optional. Use when `changeLogType = issueBased && addChangeLog = true`.

Default value: `[{ "label" : "bug", "displayName" : "Bugs", "state" : "closed" }]`.

Categorizes changes based on the label associated with the issue or PR. For a label, you can mention the display name for the category and the state of issue. Examples of labels include: `"[{ "label" : "bug", "displayName" : "Bugs", "state" : "closed" }]"`. In cases where a change has multiple labels on it, the first specified label takes priority. Leave this field empty to see a flat list of issues or PRs.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task in your pipeline to create, edit, or discard a [GitHub release](#).

GitHub service connection

This task requires a [GitHub service connection](#) with **Write** permission to the GitHub repository. You can create a GitHub service connection in your Azure Pipelines project. Once created, use the name of the service connection in this task's settings.

Examples

Create a GitHub release

The following YAML creates a GitHub release every time the task runs. The build number is used as the tag version for the release. All .exe files and README.txt files in the \$(Build.ArtifactStagingDirectory) folder are uploaded as assets. By default, the task also generates a change log (a list of commits and issues that are part of this release) and publishes it as release notes.

YAML

```
- task: GithubRelease@1
  displayName: 'Create GitHub Release'
  inputs:
    gitHubConnection: zenithworks
    repositoryName: zenithworks/javaAppWithMaven
    tagSource: manual
    tag: $(Build.BuildNumber)
    assets: |
      $(Build.ArtifactStagingDirectory)/*.exe
      $(Build.ArtifactStagingDirectory)/README.txt
```

You can also control the creation of the release based on repository tags. The following YAML creates a GitHub release only when the commit that triggers the pipeline has a Git tag associated with it. The GitHub release is created with the same tag version as the associated Git tag.

YAML

```
- task: GithubRelease@1
  displayName: 'Create GitHub Release'
  inputs:
    gitHubConnection: zenithworks
    repositoryName: zenithworks/javaAppWithMaven
    assets: $(Build.ArtifactStagingDirectory)/*.exe
```

You may also want to use the task in conjunction with task conditions to get even finer control over when the task runs, thereby restricting the creation of releases. For example, in the following YAML the task runs only when the pipeline is triggered by a Git tag matching the pattern 'refs/tags/release-v*'.

YAML

```
- task: GithubRelease@1
  displayName: 'Create GitHub Release'
  condition: startsWith(variables['Build.SourceBranch'], 'refs/tags/release-v')
  inputs:
    gitHubConnection: zenithworks
```

```
repositoryName: zenithworks/javaAppWithMaven
assets: $(Build.ArtifactStagingDirectory)/*.exe
```

Edit a GitHub release

The following YAML updates the status of a GitHub release from 'draft' to 'published'. The release to be edited is determined by the specified tag.

YAML

```
- task: GithubRelease@1
displayName: 'Edit GitHub Release'
inputs:
  gitHubConnection: zenithworks
  repositoryName: zenithworks/javaAppWithMaven
  action: edit
  tag: $(myDraftReleaseVersion)
  isDraft: false
```

Delete a GitHub release

The following YAML deletes a GitHub release. The release to be deleted is determined by the specified tag.

YAML

```
- task: GithubRelease@1
displayName: 'Delete GitHub Release'
inputs:
  gitHubConnection: zenithworks
  repositoryName: zenithworks/javaAppWithMaven
  action: delete
  tag: $(myDraftReleaseVersion)
```

Inline release notes

The following YAML create a GitHub release and add inline release notes.

YAML

```
- task: GitHubRelease@1
inputs:
  gitHubConnection: <GITHUB_SERVICE_CONNECTION>
  repositoryName: '$(Build.Repository.Name)'
  action: 'create'
```

```
target: '$(Build.SourceVersion)'  
tagSource: 'userSpecifiedTag'  
tag: <YOUR_TAG>  
title: <YOUR_TITLE>  
releaseNotesSource: 'inline'  
releaseNotesInline: <YOUR_RELEASE_NOTES>
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.0.0 or greater
Task category	Utility

GitHubRelease@0 - GitHub Release v0 task

Article • 09/26/2023

Use this task to create, edit, or delete a GitHub release.

Syntax

YAML

```
# GitHub Release v0
# Create, edit, or delete a GitHub release.
- task: GitHubRelease@0
  inputs:
    gitHubConnection: # string. Required. GitHub connection (OAuth or PAT).
    repositoryName: '$(Build.Repository.Name)' # string. Required.
    Repository. Default: $(Build.Repository.Name).
    action: 'create' # 'create' | 'edit' | 'delete'. Required. Action.
    Default: create.
    #target: '$(Build.SourceVersion)' # string. Required when action = create || action = edit. Target. Default: $(Build.SourceVersion).
    tagSource: 'auto' # 'auto' | 'manual'. Required when action = create.
    Tag source. Default: auto.
    #tagPattern: # string. Optional. Use when tagSource = auto. Tag Pattern.
    #tag: # string. Required when action = edit || action = delete || tagSource = manual. Tag.
    #title: # string. Optional. Use when action = create || action = edit.
    Release title.
    #releaseNotesSource: 'file' # 'file' | 'input'. Optional. Use when action = create || action = edit. Release notes source. Default: file.
    #releaseNotesFile: # string. Optional. Use when releaseNotesSource = file. Release notes file path.
    #releaseNotes: # string. Optional. Use when releaseNotesSource = input.
    Release notes.
    #assets: '$(Build.ArtifactStagingDirectory)/*' # string. Optional. Use when action = create || action = edit. Assets. Default: $(Build.ArtifactStagingDirectory)/*.
    #assetUploadMode: 'delete' # 'delete' | 'replace'. Optional. Use when action = edit. Asset upload mode. Default: delete.
    #isDraft: false # boolean. Optional. Use when action = create || action = edit. Draft release. Default: false.
    #isPreRelease: false # boolean. Optional. Use when action = create || action = edit. Pre-release. Default: false.
    #addChangeLog: true # boolean. Optional. Use when action = create || action = edit. Add changelog. Default: true.
    # Changelog configuration
    changeLogCompareToRelease: 'lastFullRelease' # 'lastFullRelease' | 'lastNonDraftRelease' | 'lastNonDraftReleaseByTag'. Required when addChangeLog = true. Compare to. Default: lastFullRelease.
```

```
#changeLogCompareToReleaseTag: # string. Required when
changeLogCompareToRelease = lastNonDraftReleaseByTag && addChangeLog = true.
Release Tag.
  changeLogType: 'commitBased' # 'commitBased' | 'issueBased'. Required
when addChangeLog = true. Changelog type. Default: commitBased.
  #changeLogLabels: '[{ "label" : "bug", "displayName" : "Bugs", "state" :
"closed" }]' # string. Optional. Use when changeLogType = issueBased &&
addChangeLog = true. Categories. Default: [{ "label" : "bug", "displayName"
: "Bugs", "state" : "closed" }].
```

Inputs

gitHubConnection - GitHub connection (OAuth or PAT)
`string`. Required.

Specifies the name of the GitHub service connection to use to connect to the GitHub repository. The connection must be based on a GitHub user's OAuth or a GitHub personal access token. For more information about service connections, see [Manage service connections](#).

repositoryName - Repository
`string`. Required. Default value: `$(Build.Repository.Name)`.

Specifies the name of the GitHub repository where you will create, edit, or delete the GitHub release.

action - Action
`string`. Required. Allowed values: `create`, `edit`, `delete`. Default value: `create`.

Specifies the type of release operation to perform. This task can create, edit, or delete a GitHub release.

target - Target
`string`. Required when `action = create || action = edit`. Default value:
`$(Build.SourceVersion)`.

Specifies the commit SHA you want to use to create the GitHub release, for example `48b11d8d6e92a22e3e9563a3f643699c16fd6e27`. You can also use a variable, like `$(myCommitSHA)`, in this field.

tagSource - Tag source

`string`. Required when `action = create`. Allowed values: `auto` (Git tag), `manual` (User specified tag). Default value: `auto`.

Specifies the tag you want to use for release creation. The `auto` (Git tag) option automatically uses the tag that is associated with the Git commit. Use the `manual` (User specified tag) option to manually provide a tag.

tagPattern - Tag Pattern

`string`. Optional. Use when `tagSource = auto`.

Specifies the Git tag pattern by using regex, for example `release-v1.*`. GitHub release will be created only for commits that have matching Git tag.

tag - Tag

`string`. Required when `action = edit || action = delete || tagSource = manual`.

Specifies the tag you want to use when you create, edit, or delete a release. You can also use a variable, like `$(myTagName)`, in this field.

title - Release title

`string`. Optional. Use when `action = create || action = edit`.

Specifies the title of the GitHub release. If left empty, the tag will be used as the release title.

releaseNotesSource - Release notes source

`string`. Optional. Use when `action = create || action = edit`. Allowed values: `file` (Release notes file), `input` (Inline release notes). Default value: `file`.

Specifies the description of the GitHub release. Use the `file` (Release notes file) option to use file contents as release notes. Use the `input` (Inline release notes) option to manually enter release notes.

releaseNotesFile - Release notes file path

`string`. Optional. Use when `releaseNotesSource = file`.

Specifies the file that contains the release notes.

releaseNotes - Release notes

`string`. Optional. Use when `releaseNotesSource = input`.

Specifies the release notes. Markdown is supported.

assets - Assets

`string`. Optional. Use when `action = create || action = edit`. Default value: `$(Build.ArtifactStagingDirectory)/*`.

Specifies the files you want to upload as assets of the release. You can use wildcard characters to specify multiple files. For example, use

`$(Build.ArtifactStagingDirectory)/*.zip` for build pipelines or use
`$(System.DefaultWorkingDirectory)/*.zip` for release pipelines.

You can also specify multiple patterns, one per line. By default, all files in the `$(Build.ArtifactStagingDirectory)` directory will be uploaded. For more information about the list of pre-defined variables that are available, see [build variables](#) and [release variables](#).

assetUploadMode - Asset upload mode

`string`. Optional. Use when `action = edit`. Allowed values: `delete` (Delete existing assets), `replace` (Replace existing assets). Default value: `delete`.

Specifies the asset upload mode you want to use. Use the `delete` (Delete existing assets) option to first delete any existing assets in the release and then upload all assets. Use the `replace` (Replace existing assets) option to replace any assets that have the same name.

isDraft - Draft release

`boolean`. Optional. Use when `action = create || action = edit`. Default value: `false`.

Indicates whether you want to save the release as a draft (unpublished). If `false`, the release will be published.

isPreRelease - Pre-release

`boolean`. Optional. Use when `action = create || action = edit`. Default value: `false`.

Indicates whether you want to mark the release as a pre-release.

addChangeLog - Add changelog

`boolean`. Optional. Use when `action = create || action = edit`. Default value: `true`.

Specifies if you want to include a changelog. If set to `true`, a list of changes (commits and issues) between the current release and the last published release will be generated and appended to the release notes.

changeLogCompareToRelease - Compare to

`string`. Required when `addChangeLog = true`. Allowed values: `lastFullRelease` (Last full release), `lastNonDraftRelease` (Last non-draft release), `lastNonDraftReleaseByTag` (Last non-draft release by tag). Default value: `lastFullRelease`.

Indicates which release to compare with to generate the changelog:

- `lastFullRelease` (Last full release): Compares the current release with the most recent non-draft release that is not marked as pre-release.
- `lastNonDraftRelease` (Last non-draft release): Compares the current release with the most recent non-draft release.
- `lastNonDraftReleaseByTag` (Last non-draft release by tag): Compares the current release with the last non-draft release matching the specified tag. You can also specify a regex instead of an exact tag.

changeLogCompareToReleaseTag - Release Tag

`string`. Required when `changeLogCompareToRelease = lastNonDraftReleaseByTag && addChangeLog = true`.

Specifies the regex for the release tag. A release matching this tag will be used as the base for changelog computation.

changeLogType - Changelog type

`string`. Required when `addChangeLog = true`. Allowed values: `commitBased` (Commit based), `issueBased` (Issue based). Default value: `commitBased`.

Specifies the changelog type. A changelog can be commit-based or issue-based. A commit-based changelog lists all commits included in a release. An issue-based changelog lists all the issues or pull requests (PRs) included in the release.

`changeLogLabels` - Categories

`string`. Optional. Use when `changeLogType = issueBased && addChangeLog = true`.

Default value: `[{ "label" : "bug", "displayName" : "Bugs", "state" : "closed" }]`.

Categorizes changes based on the label associated with the issue or PR. For a label, you can mention the display name for the category and the state of issue. Examples of labels include: `"[{ "label" : "bug", "displayName" : "Bugs", "state" : "closed" }]`. In cases where a change has multiple labels on it, the first specified label takes priority. Leave this field empty to see a flat list of issues or PRs.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.0.0 or greater
Task category	Utility

Go@0 - Go v0 task

Article • 09/26/2023

Use this task to get, build, or test a Go application, or to run a custom Go command.

Syntax

YAML

```
# Go v0
# Get, build, or test a Go application, or run a custom Go command.
- task: Go@0
  inputs:
    command: 'get' # 'get' | 'build' | 'test' | 'custom'. Required. Command.
    Default: get.
    #customCommand: # string. Required when command == custom. Custom
    #command.
    #arguments: # string. Arguments.
    # Advanced
    #workingDirectory: # string. Working directory.
```

Inputs

`command` - Command

`string`. Required. Allowed values: `get`, `build`, `test`, `custom`. Default value: `get`.

Specifies a Go command to run. Use `Custom` to run a command not listed here.

`customCommand` - Custom command

`string`. Required when `command == custom`.

A custom Go command to execute. For example, to execute `go version`, use `version`.

`arguments` - Arguments

`string`.

The optional arguments to the selected command. For example, use build-time arguments for the `go build` command.

`workingDirectory` - Working directory

`string`.

The working directory where you want the command to run. When empty, the root of the repository (for builds) or artifacts (for releases) is used, which is the value of `$(System.DefaultWorkingDirectory)`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to get, build, or test a Go application, or to run a custom Go command.

Examples

yml

```
variables:
  GOBIN: '$(GOPATH)/bin' # Go binaries path
  GOROOT: '/usr/local/go1.11' # Go installation path
  GOPATH: '$(system.defaultWorkingDirectory)/gopath' # Go workspace path
  modulePath: '$(GOPATH)/src/github.com/${build.repository.name}' # Path to
the module's code

steps:
- task: GoTool@0
  displayName: 'Use Go 1.10'

- task: Go@0
  displayName: 'go get'
  inputs:
    arguments: '-d'

- task: Go@0
  displayName: 'go build'
  inputs:
    command: build
```

```

arguments: '-o "$(System.TeamProject).exe"'

- task: ArchiveFiles@2
  displayName: 'Archive files'
  inputs:
    rootFolderOrFile: '$(Build.Repository.LocalPath)'
    includeRootFolder: False

- task: PublishBuildArtifacts@1
  displayName: 'Publish artifact'
  condition: succeededOrFailed()

```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Build

GoTool@0 - Go tool installer v0 task

Article • 09/26/2023

Use this task to find in the tools cache or download a specific version of Go and add it to the PATH.

Syntax

YAML

```
# Go tool installer v0
# Find in cache or download a specific version of Go and add it to the PATH.
- task: GoTool@0
  inputs:
    version: '1.10' # string. Required. Version. Default: 1.10.
  # Advanced
  #goPath: # string. GOPATH.
  #goBin: # string. GOBIN.
```

Inputs

`version` - Version

`string`. Required. Default value: `1.10`.

The Go version to download (if necessary) and use, for example `1.9.3`.

`goPath` - GOPATH

`string`.

A custom value for the GOPATH environment variable.

`goBin` - GOBIN

`string`.

A custom value for the GOBIN environment variable.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to find or download a specific version of the Go tool into the tools cache and add it to the PATH. Use the task to change the version of Go Lang used in subsequent tasks.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: GO
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Tool

Gradle@3 - Gradle v3 task

Article • 09/26/2023

Use this task to build using a Gradle wrapper script.

Syntax

YAML

```
# Gradle v3
# Build using a Gradle wrapper script.
- task: Gradle@3
  inputs:
    gradleWrapperFile: 'gradlew' # string. Alias: wrapperScript. Required.
    Gradle wrapper. Default: gradlew.
    #workingDirectory: # string. Alias: cwd. Working directory.
    #options: # string. Options.
    tasks: 'build' # string. Required. Tasks. Default: build.
  # JUnit Test Results
  #publishJUnitResults: true # boolean. Publish to Azure Pipelines.
  Default: true.
  testResultsFiles: '**/TEST-*.xml' # string. Required when
  publishJUnitResults = true. Test results files. Default: **/TEST-*.
  #testRunTitle: # string. Optional. Use when publishJUnitResults = true.
  Test run title.
  # Code Coverage
  #codeCoverageToolOption: 'None' # 'None' | 'Cobertura' | 'JaCoCo'.
  Alias: codeCoverageTool. Code coverage tool. Default: None.
  codeCoverageClassFilesDirectories: 'build/classes/main/' # string.
  Alias: classFilesDirectories. Required when codeCoverageTool != None. Class
  files directories. Default: build/classes/main/.
  #codeCoverageClassFilter: # string. Alias: classFilter. Optional. Use
  when codeCoverageTool != None. Class inclusion/exclusion filters.
  #codeCoverageFailIfEmpty: false # boolean. Alias: failIfCoverageEmpty.
  Optional. Use when codeCoverageTool != None. Fail when code coverage results
  are missing. Default: false.
  #codeCoverageGradle5xOrHigher: true # boolean. Alias: gradle5xOrHigher.
  Optional. Use when codeCoverageTool = JaCoCo. Gradle version >= 5.x.
  Default: true.
  # Advanced
  javaHomeOption: 'JDKVersion' # 'JDKVersion' | 'Path'. Alias:
  javaHomeSelection. Required. Set JAVA_HOME by. Default: JDKVersion.
  #jdkVersionOption: 'default' # 'default' | '1.17' | '1.11' | '1.10' |
  '1.9' | '1.8' | '1.7' | '1.6'. Alias: jdkVersion. Optional. Use when
  javaHomeSelection = JDKVersion. JDK version. Default: default.
  #jdkDirectory: # string. Alias: jdkUserInputPath. Required when
  javaHomeSelection = Path. JDK path.
  #jdkArchitectureOption: 'x64' # 'x86' | 'x64'. Alias: jdkArchitecture.
  Optional. Use when jdkVersion != default. JDK architecture. Default: x64.
  #gradleOptions: '-Xmx1024m' # string. Alias: gradleOpts. Set
```

```
GRADLE_OPTS. Default: -Xmx1024m.  
# Code Analysis  
#sonarQubeRunAnalysis: false # boolean. Alias: sqAnalysisEnabled. Run  
SonarQube or SonarCloud Analysis. Default: false.  
#sqGradlePluginVersionChoice: 'specify' # 'specify' | 'build'. Required  
when sqAnalysisEnabled = true. SonarQube scanner for Gradle version.  
Default: specify.  
#sonarQubeGradlePluginVersion: '2.6.1' # string. Alias:  
sqGradlePluginVersion. Required when sqAnalysisEnabled = true &&  
sqGradlePluginVersionChoice = specify. SonarQube scanner for Gradle plugin  
version. Default: 2.6.1.  
#checkStyleRunAnalysis: false # boolean. Alias:  
checkstyleAnalysisEnabled. Run Checkstyle. Default: false.  
#findBugsRunAnalysis: false # boolean. Alias: findbugsAnalysisEnabled.  
Run FindBugs. Default: false.  
#pmdRunAnalysis: false # boolean. Alias: pmdAnalysisEnabled. Run PMD.  
Default: false.  
#spotBugsAnalysis: false # boolean. Alias: spotBugsAnalysisEnabled. Run  
SpotBugs. Default: false.  
#spotBugsGradlePluginVersionChoice: 'specify' # 'specify' | 'build'.  
Required when spotBugsAnalysisEnabled = true. Spotbugs plugin version.  
Default: specify.  
#spotbugsGradlePluginVersion: '4.7.0' # string. Required when  
spotBugsAnalysisEnabled = true && spotBugsGradlePluginVersionChoice =  
specify. Version number. Default: 4.7.0.
```

Inputs

`gradleWrapperFile` - Gradle wrapper

Input alias: `wrapperScript`. `string`. Required. Default value: `gradlew`.

Specifies the `gradlew` wrapper's location within the repository that will be used for the build. Agents on Windows (including Microsoft-hosted agents) must use the `gradlew.bat` wrapper. Agents on Linux or macOS can use the `gradlew` shell script. Learn more about the [Gradle Wrapper](#).

`workingDirectory` - Working directory

Input alias: `cwd`. `string`.

Specifies the working directory to run the Gradle build. The task uses the repository root directory if the working directory is not specified.

`options` - Options

`string`.

Specifies the command line options that will be passed to the Gradle wrapper. See [Gradle Command Line](#) for more information.

`tasks` - Tasks

`string`. Required. Default value: `build`.

The task(s) for Gradle to execute. A list of task names should be separated by spaces and can be taken from `gradlew tasks` issued from a command prompt.

See [Gradle Build Script Basics](#) for more information.

`publishJUnitResults` - Publish to Azure Pipelines

`boolean`. Default value: `true`.

Publishes JUnit test results produced by the Gradle build to Azure Pipelines. The task publishes each test results file matching `Test Results Files` as a test run in Azure Pipelines.

`testResultsFiles` - Test results files

`string`. Required when `publishJUnitResults = true`. Default value: `**/TEST-*.xml`.

The file path for test results. [Wildcards](#) can be used. For example, `**/TEST-*.xml` for all XML files whose name starts with `TEST-`.

`testRunTitle` - Test run title

`string`. Optional. Use when `publishJUnitResults = true`.

Provides a name for the JUnit test case results for this build.

`codeCoverageToolOption` - Code coverage tool

Input alias: `codeCoverageTool`. `string`. Allowed values: `None`, `Cobertura`, `JaCoCo`. Default value: `None`.

Specifies a code coverage tool to determine the code that is covered by the test cases for the build.

`codeCoverageClassFilesDirectories` - Class files directories

Input alias: `classFilesDirectories`. `string`. Required when `codeCoverageTool != None`.

Default value: `build/classes/main/`.

The comma-separated list of directories containing class files and archive files (.jar, .war, and more). Code coverage is reported for class files in these directories. Normally, the task searches classes under `build/classes/java/main` (for Gradle 4+), which is the default class directory for Gradle builds.

`codeCoverageClassFilter` - Class inclusion/exclusion filters

Input alias: `classFilter`. `string`. Optional. Use when `codeCoverageTool != None`.

The comma-separated list of filters to include or exclude classes from collecting code coverage. For example: `+:com.*`, `+:org.*`, `-:my.app*.*`.

`codeCoverageFailIfEmpty` - Fail when code coverage results are missing

Input alias: `failIfCoverageEmpty`. `boolean`. Optional. Use when `codeCoverageTool != None`. Default value: `false`.

Fails the build if code coverage did not produce any results to publish.

`codeCoverageGradle5xOrHigher` - Gradle version >= 5.x

Input alias: `gradle5xOrHigher`. `boolean`. Optional. Use when `codeCoverageTool = JaCoCo`. Default value: `true`.

Set this to 'true' if gradle version is >= 5.x.

`javaHomeOption` - Set JAVA_HOME by

Input alias: `javaHomeSelection`. `string`. Required. Allowed values: `JDKVersion` (JDK Version), `Path`. Default value: `JDKVersion`.

Sets JAVA_HOME by selecting a JDK version that the task discovers during builds or by manually entering a JDK path.

`jdkVersionOption` - JDK version

Input alias: `jdkVersion`. `string`. Optional. Use when `javaHomeSelection = JDKVersion`. Allowed values: `default`, `1.17` (JDK 17), `1.11` (JDK 11), `1.10` (JDK 10 (out of support)), `1.9` (JDK 9 (out of support)), `1.8` (JDK 8), `1.7` (JDK 7), `1.6` (JDK 6 (out of support)). Default value: `default`.

Attempts to discover the path to the selected JDK version and set JAVA_HOME accordingly.

`jdkDirectory` - JDK path

Input alias: `jdkUserInputPath`. `string`. Required when `javaHomeSelection = Path`.

Sets JAVA_HOME to the given path.

`jdkArchitectureOption` - JDK architecture

Input alias: `jdkArchitecture`. `string`. Optional. Use when `jdkVersion != default`.

Allowed values: `x86`, `x64`. Default value: `x64`.

Supplies the JDK architecture (x86 or x64).

`gradleOptions` - Set GRADLE_OPTS

Input alias: `gradleOpts`. `string`. Default value: `-Xmx1024m`.

Sets the GRADLE_OPTS environment variable, which is used to send command-line arguments to start the JVM. The `xmx` flag specifies the maximum memory available to the JVM.

`sonarQubeRunAnalysis` - Run SonarQube or SonarCloud Analysis

Input alias: `sqAnalysisEnabled`. `boolean`. Default value: `false`.

This option has changed from version 1 of the **Gradle** task to use the [SonarQube](#) and [SonarCloud](#) marketplace extensions. Enable this option to run [SonarQube or SonarCloud analysis](#) after executing tasks in the **Tasks** field. You must also add a **Prepare Analysis Configuration** task from one of the extensions to the build pipeline before this Gradle task.

`sqGradlePluginVersionChoice` - SonarQube scanner for Gradle version

`string`. Required when `sqAnalysisEnabled = true`. Allowed values: `specify` (Specify version number), `build` (Use plugin applied in your build.gradle). Default value: `specify`.

Specifies the SonarQube Gradle plugin version to use. Declare the version in the Gradle configuration file, or specify a version with this string.

`sonarQubeGradlePluginVersion` - SonarQube scanner for Gradle plugin version

Input alias: `sqGradlePluginVersion`. `string`. Required when `sqAnalysisEnabled = true` & `sqGradlePluginVersionChoice = specify`. Default value: `2.6.1`.

Contains the version number of the [SonarQube Gradle plugin](#).

`checkStyleRunAnalysis` - Run Checkstyle

Input alias: `checkstyleAnalysisEnabled`. `boolean`. Default value: `false`.

Runs the Checkstyle tool with the default Sun checks. Results are uploaded as build artifacts.

`findBugsRunAnalysis` - Run FindBugs

Input alias: `findbugsAnalysisEnabled`. `boolean`. Default value: `false`.

Uses the FindBugs static analysis tool to look for bugs in the code. Results are uploaded as build artifacts. In Gradle 6.0, [this plugin was removed](#). Use the SpotBugs plugin instead.

`pmdRunAnalysis` - Run PMD

Input alias: `pmdAnalysisEnabled`. `boolean`. Default value: `false`.

Uses the PMD Java static analysis tool to look for bugs in the code. The results are uploaded as build artifacts.

`spotBugsAnalysis` - Run SpotBugs

Input alias: `spotBugsAnalysisEnabled`. `boolean`. Default value: `false`.

Runs `spotBugs` when `true`. This plugin works with Gradle v5.6 or later. Learn more about [using the SpotBugs Gradle plugin](#). The plugin may work in an unexpected way or may not work at all with an earlier Gradle version.

`spotBugsGradlePluginVersionChoice` - Spotbugs plugin version

`string`. Required when `spotBugsAnalysisEnabled = true`. Allowed values: `specify` (Specify version number), `build` (Use plugin applied in your build.gradle). Default value: `specify`.

Specifies the SpotBugs Gradle plugin version to use. The version can be declared in the Gradle configuration file, or the version can be specified in this string.

`spotbugsGradlePluginVersion` - Version number
string. Required when `spotBugsAnalysisEnabled = true && spotBugsGradlePluginVersionChoice = specify`. Default value: `4.7.0`.

Contains the version number of the [SpotBugs Gradle plugin](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Configuration of the SonarQube analysis was moved to the [SonarQube](#) or [SonarCloud](#) extensions in the task `Prepare Analysis Configuration`.

Use this task to build using a Gradle wrapper script.

How do I generate a wrapper from my Gradle project?

The Gradle wrapper allows the build agent to download and configure the exact Gradle environment that is checked into the repository without having any software configuration on the build agent itself other than the JVM.

1. Create the Gradle wrapper by issuing the following command from the root project directory where your `build.gradle` resides:

```
jamal@fabrikam> gradle wrapper
```

2. Upload your Gradle wrapper to your remote repository.

There is a binary artifact that is generated by the gradle wrapper (located at `gradle/wrapper/gradle-wrapper.jar`). This binary file is small and doesn't require

updating. If you need to change the Gradle configuration run on the build agent, you update the `gradle-wrapper.properties`.

The repository should look something like this:

```
|-- gradle/
  '-- wrapper/
    '-- gradle-wrapper.jar
    '-- gradle-wrapper.properties
|-- src/
|-- .gitignore
|-- build.gradle
|-- gradlew
|-- gradlew.bat
```

How do I fix timeouts when downloading dependencies?

To fix errors such as `Read timed out` when downloading dependencies, users of Gradle 4.3+ can change the timeout by adding `-Dhttp.socketTimeout=60000 -Dhttp.connectionTimeout=60000` to `Options`. This increases the timeout from 10 seconds to 1 minute.

Examples

[Build your Java app with Gradle](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.91.0 or greater

Requirement	Description
Task category	Build

Gradle@2 - Gradle v2 task

Article • 09/26/2023

Build using a Gradle wrapper script.

Syntax

YAML

```
# Gradle v2
# Build using a Gradle wrapper script.
- task: Gradle@2
  inputs:
    gradleWrapperFile: 'gradlew' # string. Alias: wrapperScript. Required.
    Gradle wrapper. Default: gradlew.
    #workingDirectory: # string. Alias: cwd. Working directory.
    #options: # string. Options.
    tasks: 'build' # string. Required. Tasks. Default: build.
  # JUnit Test Results
  #publishJUnitResults: true # boolean. Publish to Azure Pipelines.
  Default: true.
  testResultsFiles: '**/TEST-*.xml' # string. Required when
  publishJUnitResults = true. Test results files. Default: **/TEST-*.
  #testRunTitle: # string. Optional. Use when publishJUnitResults = true.
  Test run title.
  # Code Coverage
  #codeCoverageToolOption: 'None' # 'None' | 'Cobertura' | 'JaCoCo'.
  Alias: codeCoverageTool. Code coverage tool. Default: None.
  codeCoverageClassFilesDirectories: 'build/classes/main/' # string.
  Alias: classFilesDirectories. Required when codeCoverageTool != None. Class
  files directories. Default: build/classes/main/.
  #codeCoverageClassFilter: # string. Alias: classFilter. Optional. Use
  when codeCoverageTool != None. Class inclusion/exclusion filters.
  #codeCoverageFailIfEmpty: false # boolean. Alias: failIfCoverageEmpty.
  Optional. Use when codeCoverageTool != None. Fail when code coverage results
  are missing. Default: false.
  #codeCoverageGradle5xOrHigher: true # boolean. Alias: gradle5xOrHigher.
  Optional. Use when codeCoverageTool = JaCoCo. Gradle version >= 5.x.
  Default: true.
  # Advanced
  javaHomeOption: 'JDKVersion' # 'JDKVersion' | 'Path'. Alias:
  javaHomeSelection. Required. Set JAVA_HOME by. Default: JDKVersion.
  #jdkVersionOption: 'default' # 'default' | '1.17' | '1.11' | '1.10' |
  '1.9' | '1.8' | '1.7' | '1.6'. Alias: jdkVersion. Optional. Use when
  javaHomeSelection = JDKVersion. JDK version. Default: default.
  #jdkDirectory: # string. Alias: jdkUserInputPath. Required when
  javaHomeSelection = Path. JDK path.
  #jdkArchitectureOption: 'x64' # 'x86' | 'x64'. Alias: jdkArchitecture.
  Optional. Use when jdkVersion != default. JDK architecture. Default: x64.
  #gradleOptions: '-Xmx1024m' # string. Alias: gradleOpts. Set
```

```
GRADLE_OPTS. Default: -Xmx1024m.  
# Code Analysis  
#sonarQubeRunAnalysis: false # boolean. Alias: sqAnalysisEnabled. Run  
SonarQube or SonarCloud Analysis. Default: false.  
#sqGradlePluginVersionChoice: 'specify' # 'specify' | 'build'. Required  
when sqAnalysisEnabled = true. SonarQube scanner for Gradle version.  
Default: specify.  
#sonarQubeGradlePluginVersion: '2.6.1' # string. Alias:  
sqGradlePluginVersion. Required when sqAnalysisEnabled = true &&  
sqGradlePluginVersionChoice = specify. SonarQube scanner for Gradle plugin  
version. Default: 2.6.1.  
#checkStyleRunAnalysis: false # boolean. Alias:  
checkstyleAnalysisEnabled. Run Checkstyle. Default: false.  
#findBugsRunAnalysis: false # boolean. Alias: findbugsAnalysisEnabled.  
Run FindBugs. Default: false.  
#pmdRunAnalysis: false # boolean. Alias: pmdAnalysisEnabled. Run PMD.  
Default: false.  
#spotBugsAnalysis: false # boolean. Alias: spotBugsAnalysisEnabled. Run  
SpotBugs. Default: false.  
#spotBugsGradlePluginVersionChoice: 'specify' # 'specify' | 'build'.  
Required when spotBugsAnalysisEnabled = true. Spotbugs plugin version.  
Default: specify.  
#spotbugsGradlePluginVersion: '4.7.0' # string. Required when  
spotBugsAnalysisEnabled = true && spotBugsGradlePluginVersionChoice =  
specify. Version number. Default: 4.7.0.
```

Inputs

`gradleWrapperFile` - Gradle wrapper

Input alias: `wrapperScript`. `string`. Required. Default value: `gradlew`.

Specifies the `gradlew` wrapper's location within the repository that will be used for the build. Agents on Windows (including Microsoft-hosted agents) must use the `gradlew.bat` wrapper. Agents on Linux or macOS can use the `gradlew` shell script. Learn more about the [Gradle Wrapper](#).

`workingDirectory` - Working directory

Input alias: `cwd`. `string`.

Specifies the working directory to run the Gradle build. The task uses the repository root directory if the working directory is not specified.

`options` - Options

`string`.

Specifies the command line options that will be passed to the Gradle wrapper. See [Gradle Command Line](#) for more information.

`tasks` - Tasks

`string`. Required. Default value: `build`.

The task(s) for Gradle to execute. A list of task names should be separated by spaces and can be taken from `gradlew tasks` issued from a command prompt.

See [Gradle Build Script Basics](#) for more information.

`publishJUnitResults` - Publish to Azure Pipelines

`boolean`. Default value: `true`.

Publishes JUnit test results produced by the Gradle build to Azure Pipelines. The task publishes each test results file matching `Test Results Files` as a test run in Azure Pipelines.

`testResultsFiles` - Test results files

`string`. Required when `publishJUnitResults = true`. Default value: `**/TEST-*.xml`.

The file path for test results. [Wildcards](#) can be used. For example, `**/TEST-*.xml` for all XML files whose name starts with `TEST-`.

`testRunTitle` - Test run title

`string`. Optional. Use when `publishJUnitResults = true`.

Provides a name for the JUnit test case results for this build.

`codeCoverageToolOption` - Code coverage tool

Input alias: `codeCoverageTool`. `string`. Allowed values: `None`, `Cobertura`, `JaCoCo`. Default value: `None`.

Specifies a code coverage tool to determine the code that is covered by the test cases for the build.

`codeCoverageClassFilesDirectories` - Class files directories

Input alias: `classFilesDirectories`. `string`. Required when `codeCoverageTool != None`.

Default value: `build/classes/main/`.

The comma-separated list of directories containing class files and archive files (.jar, .war, and more). Code coverage is reported for class files in these directories. Normally, the task searches classes under `build/classes/java/main` (for Gradle 4+), which is the default class directory for Gradle builds.

`codeCoverageClassFilter` - Class inclusion/exclusion filters

Input alias: `classFilter`. `string`. Optional. Use when `codeCoverageTool != None`.

The comma-separated list of filters to include or exclude classes from collecting code coverage. For example: `+:com.*`, `+:org.*`, `-:my.app*.*`.

`codeCoverageFailIfEmpty` - Fail when code coverage results are missing

Input alias: `failIfCoverageEmpty`. `boolean`. Optional. Use when `codeCoverageTool != None`. Default value: `false`.

Fails the build if code coverage did not produce any results to publish.

`codeCoverageGradle5xOrHigher` - Gradle version >= 5.x

Input alias: `gradle5xOrHigher`. `boolean`. Optional. Use when `codeCoverageTool = JaCoCo`. Default value: `true`.

Set this to 'true' if gradle version is >= 5.x.

`javaHomeOption` - Set JAVA_HOME by

Input alias: `javaHomeSelection`. `string`. Required. Allowed values: `JDKVersion` (JDK Version), `Path`. Default value: `JDKVersion`.

Sets JAVA_HOME by selecting a JDK version that the task discovers during builds or by manually entering a JDK path.

`jdkVersionOption` - JDK version

Input alias: `jdkVersion`. `string`. Optional. Use when `javaHomeSelection = JDKVersion`. Allowed values: `default`, `1.17` (JDK 17), `1.11` (JDK 11), `1.10` (JDK 10 (out of support)), `1.9` (JDK 9 (out of support)), `1.8` (JDK 8), `1.7` (JDK 7), `1.6` (JDK 6 (out of support)). Default value: `default`.

Attempts to discover the path to the selected JDK version and set JAVA_HOME accordingly.

`jdkDirectory` - JDK path

Input alias: `jdkUserInputPath`. `string`. Required when `javaHomeSelection = Path`.

Sets JAVA_HOME to the given path.

`jdkArchitectureOption` - JDK architecture

Input alias: `jdkArchitecture`. `string`. Optional. Use when `jdkVersion != default`.

Allowed values: `x86`, `x64`. Default value: `x64`.

Supplies the JDK architecture (x86 or x64).

`gradleOptions` - Set GRADLE_OPTS

Input alias: `gradleOpts`. `string`. Default value: `-Xmx1024m`.

Sets the GRADLE_OPTS environment variable, which is used to send command-line arguments to start the JVM. The `xmx` flag specifies the maximum memory available to the JVM.

`sonarQubeRunAnalysis` - Run SonarQube or SonarCloud Analysis

Input alias: `sqAnalysisEnabled`. `boolean`. Default value: `false`.

This option has changed from version 1 of the **Gradle** task to use the [SonarQube](#) and [SonarCloud](#) marketplace extensions. Enable this option to run [SonarQube or SonarCloud analysis](#) after executing tasks in the **Tasks** field. You must also add a **Prepare Analysis Configuration** task from one of the extensions to the build pipeline before this Gradle task.

`sqGradlePluginVersionChoice` - SonarQube scanner for Gradle version

`string`. Required when `sqAnalysisEnabled = true`. Allowed values: `specify` (Specify version number), `build` (Use plugin applied in your build.gradle). Default value: `specify`.

Specifies the SonarQube Gradle plugin version to use. Declare the version in the Gradle configuration file, or specify a version with this string.

`sonarQubeGradlePluginVersion` - SonarQube scanner for Gradle plugin version

Input alias: `sqGradlePluginVersion`. `string`. Required when `sqAnalysisEnabled = true` & `sqGradlePluginVersionChoice = specify`. Default value: `2.6.1`.

Contains the version number of the [SonarQube Gradle plugin](#).

`checkStyleRunAnalysis` - Run Checkstyle

Input alias: `checkstyleAnalysisEnabled`. `boolean`. Default value: `false`.

Runs the Checkstyle tool with the default Sun checks. Results are uploaded as build artifacts.

`findBugsRunAnalysis` - Run FindBugs

Input alias: `findbugsAnalysisEnabled`. `boolean`. Default value: `false`.

Uses the FindBugs static analysis tool to look for bugs in the code. Results are uploaded as build artifacts. In Gradle 6.0, [this plugin was removed](#). Use the SpotBugs plugin instead.

`pmdRunAnalysis` - Run PMD

Input alias: `pmdAnalysisEnabled`. `boolean`. Default value: `false`.

Uses the PMD Java static analysis tool to look for bugs in the code. The results are uploaded as build artifacts.

`spotBugsAnalysis` - Run SpotBugs

Input alias: `spotBugsAnalysisEnabled`. `boolean`. Default value: `false`.

Runs `spotBugs` when `true`. This plugin works with Gradle v5.6 or later. Learn more about [using the SpotBugs Gradle plugin](#). The plugin may work in an unexpected way or may not work at all with an earlier Gradle version.

`spotBugsGradlePluginVersionChoice` - Spotbugs plugin version

`string`. Required when `spotBugsAnalysisEnabled = true`. Allowed values: `specify` (Specify version number), `build` (Use plugin applied in your build.gradle). Default value: `specify`.

Specifies the SpotBugs Gradle plugin version to use. The version can be declared in the Gradle configuration file, or the version can be specified in this string.

`spotbugsGradlePluginVersion` - Version number
string. Required when `spotBugsAnalysisEnabled = true && spotBugsGradlePluginVersionChoice = specify`. Default value: `4.7.0`.

Contains the version number of the [SpotBugs Gradle plugin](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Configuration of the SonarQube analysis was moved to the [SonarQube](#) or [SonarCloud](#) extensions in the task `Prepare Analysis Configuration`.

Use this task to build using a Gradle wrapper script.

How do I generate a wrapper from my Gradle project?

The Gradle wrapper allows the build agent to download and configure the exact Gradle environment that is checked into the repository without having any software configuration on the build agent itself other than the JVM.

1. Create the Gradle wrapper by issuing the following command from the root project directory where your `build.gradle` resides:

```
jamal@fabrikam> gradle wrapper
```

2. Upload your Gradle wrapper to your remote repository.

There is a binary artifact that is generated by the gradle wrapper (located at `gradle/wrapper/gradle-wrapper.jar`). This binary file is small and doesn't require

updating. If you need to change the Gradle configuration run on the build agent, you update the `gradle-wrapper.properties`.

The repository should look something like this:

```
|-- gradle/
  '-- wrapper/
    '-- gradle-wrapper.jar
    '-- gradle-wrapper.properties
|-- src/
|-- .gitignore
|-- build.gradle
|-- gradlew
|-- gradlew.bat
```

How do I fix timeouts when downloading dependencies?

To fix errors such as `Read timed out` when downloading dependencies, users of Gradle 4.3+ can change the timeout by adding `-Dhttp.socketTimeout=60000 -Dhttp.connectionTimeout=60000` to `Options`. This increases the timeout from 10 seconds to 1 minute.

Examples

[Build your Java app with Gradle](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: java
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any

Requirement	Description
Agent version	1.91.0 or greater
Task category	Build

Gradle@1 - Gradle v1 task

Article • 09/26/2023

Build using a Gradle wrapper script.

Syntax

YAML

```
# Gradle v1
# Build using a Gradle wrapper script.
- task: Gradle@1
  inputs:
    gradleWrapperFile: 'gradlew' # string. Alias: wrapperScript. Required.
    Gradle Wrapper. Default: gradlew.
    #options: # string. Options.
    tasks: 'build' # string. Required. Tasks. Default: build.
  # Advanced
    #workingDirectory: # string. Alias: cwd. Working Directory.
    javaHomeOption: 'JDKVersion' # 'JDKVersion' | 'Path'. Alias:
    javaHomeSelection. Required. Set JAVA_HOME by. Default: JDKVersion.
    #jdkVersionOption: 'default' # 'default' | '1.9' | '1.8' | '1.7' |
    '1.6'. Alias: jdkVersion. Optional. Use when javaHomeSelection = JDKVersion.
    JDK Version. Default: default.
    #jdkDirectory: # string. Alias: jdkUserInputPath. Required when
    javaHomeSelection = Path. JDK Path.
    #jdkArchitectureOption: 'x64' # 'x86' | 'x64'. Alias: jdkArchitecture.
    Optional. Use when jdkVersion != default. JDK Architecture. Default: x64.
    #gradleOptions: '-Xmx1024m' # string. Alias: gradleOpts. Set
    GRADLE_OPTS. Default: -Xmx1024m.
  # JUnit Test Results
  #publishJUnitResults: true # boolean. Publish to TFS/Team Services.
  Default: true.
  testResultsFiles: '**/build/test-results/TEST-*.xml' # string. Required
  when publishJUnitResults = true. Test Results Files. Default: **/build/test-
  results/TEST-*.xml.
  #testRunTitle: # string. Optional. Use when publishJUnitResults = true.
  Test Run Title.
  # Code Coverage
  #codeCoverageToolOption: 'None' # 'None' | 'Cobertura' | 'JaCoCo'.
  Alias: codeCoverageTool. Code Coverage Tool. Default: None.
  #codeCoverageClassDirectories: 'build/classes/main/' # string.
  Alias: classDirectories. Required when codeCoverageTool = false. Class
  Files Directories. Default: build/classes/main/.
  #codeCoverageClassFilter: # string. Alias: classFilter. Optional. Use
  when codeCoverageTool != None. Class Inclusion/Exclusion Filters.
  #codeCoverageFailIfEmpty: false # boolean. Alias: failIfCoverageEmpty.
  Optional. Use when codeCoverageTool != None. Fail When Code Coverage Results
  Are Missing. Default: false.
  # Code Analysis
```

```
#sonarQubeRunAnalysis: false # boolean. Alias: sqAnalysisEnabled. Run SonarQube Analysis. Default: false.
#sonarQubeServiceEndpoint: # string. Alias: sqConnectedServiceName. Required when sqAnalysisEnabled = true. SonarQube Endpoint.
#sonarQubeProjectName: # string. Alias: sqProjectName. Required when sqAnalysisEnabled = true. SonarQube Project Name.
#sonarQubeProjectKey: # string. Alias: sqProjectKey. Required when sqAnalysisEnabled = true. SonarQube Project Key.
#sonarQubeProjectVersion: # string. Alias: sqProjectVersion. Required when sqAnalysisEnabled = true. SonarQube Project Version.
#sonarQubeGradlePluginVersion: '2.0.1' # string. Alias: sqGradlePluginVersion. Required when sqAnalysisEnabled = true. SonarQube Gradle Plugin Version. Default: 2.0.1.
#sonarQubeSpecifyDB: false # boolean. Alias: sqDbDetailsRequired. Optional. Use when sqAnalysisEnabled = true. The SonarQube server version is lower than 5.2. Default: false.
#sonarQubeDBUrl: # string. Alias: sqDbUrl. Optional. Use when sqDbDetailsRequired = true. Db Connection String.
#sonarQubeDBUsername: # string. Alias: sqDbUsername. Optional. Use when sqDbDetailsRequired = true. Db Username.
#sonarQubeDBPassword: # string. Alias: sqDbPassword. Optional. Use when sqDbDetailsRequired = true. Db User Password.
#sonarQubeIncludeFullReport: true # boolean. Alias: sqAnalysisIncludeFullReport. Optional. Use when sqAnalysisEnabled = true. Include full analysis report in the build summary (SQ 5.3+). Default: true.
#sonarQubeFailWhenQualityGateFails: # boolean. Alias: sqAnalysisBreakBuildIfQualityGateFailed. Optional. Use when sqAnalysisEnabled = true. Fail the build on quality gate failure (SQ 5.3+).
#checkStyleRunAnalysis: false # boolean. Alias: checkstyleAnalysisEnabled. Run Checkstyle. Default: false.
#findBugsRunAnalysis: false # boolean. Alias: findbugsAnalysisEnabled. Run FindBugs. Default: false.
#pmdRunAnalysis: false # boolean. Alias: pmdAnalysisEnabled. Run PMD. Default: false.
```

Inputs

`gradleWrapperFile` - Gradle Wrapper

Input alias: `wrapperScript`. `string`. Required. Default value: `gradlew`.

Specifies the `gradlew` wrapper's location within the repository that will be used for the build. Agents on Windows (including Microsoft-hosted agents) must use the `gradlew.bat` wrapper. Agents on Linux or macOS can use the `gradlew` shell script. Learn more about the [Gradle Wrapper ↗](#).

`options` - Options

`string`.

Specifies the command line options that will be passed to the Gradle wrapper. See [Gradle Command Line](#) for more information.

`tasks` - Tasks

`string`. Required. Default value: `build`.

The task(s) for Gradle to execute. A list of task names should be separated by spaces and can be taken from `gradlew tasks` issued from a command prompt.

See [Gradle Build Script Basics](#) for more information.

`workingDirectory` - Working Directory

Input alias: `cwd`. `string`.

Specifies the working directory to run the Gradle build. The task uses the repository root directory if the working directory is not specified.

`publishJUnitResults` - Publish to TFS/Team Services

`boolean`. Default value: `true`.

Publishes JUnit test results produced by the Gradle build to Azure Pipelines. The task publishes each test results file matching `Test Results Files` as a test run in Azure Pipelines.

`testResultsFiles` - Test Results Files

`string`. Required when `publishJUnitResults = true`. Default value: `**/build/test-results/TEST-*.xml`.

The file path for test results. [Wildcards](#) can be used. For example, `**/TEST-*.xml` for all XML files whose name starts with `TEST-`.

`testRunTitle` - Test Run Title

`string`. Optional. Use when `publishJUnitResults = true`.

Provides a name for the JUnit test case results for this build.

`codeCoverageToolOption` - Code Coverage Tool

Input alias: `codeCoverageTool`. `string`. Allowed values: `None`, `Cobertura`, `Jacoco`. Default

`value: None`.

Specifies a code coverage tool to determine the code that is covered by the test cases for the build.

`codeCoverageClassFilesDirectories` - Class Files Directories

Input alias: `classFilesDirectories`. `string`. Required when `codeCoverageTool = false`.

Default value: `build/classes/main/`.

The comma-separated list of directories containing class files and archive files (.jar, .war, and more). Code coverage is reported for class files in these directories. Normally, the task searches classes under `build/classes/java/main` (for Gradle 4+), which is the default class directory for Gradle builds.

`codeCoverageClassFilter` - Class Inclusion/Exclusion Filters

Input alias: `classFilter`. `string`. Optional. Use when `codeCoverageTool != None`.

The comma-separated list of filters to include or exclude classes from collecting code coverage. For example: `+:com.*`, `+:org.*`, `-:my.app*.*`.

`codeCoverageFailIfEmpty` - Fail When Code Coverage Results Are Missing

Input alias: `failIfCoverageEmpty`. `boolean`. Optional. Use when `codeCoverageTool != None`. Default value: `false`.

Fails the build if code coverage did not produce any results to publish.

`javaHomeOption` - Set JAVA_HOME by

Input alias: `javaHomeSelection`. `string`. Required. Allowed values: `JDKVersion` (JDK Version), `Path`. Default value: `JDKVersion`.

Sets JAVA_HOME by selecting a JDK version that the task discovers during builds or by manually entering a JDK path.

`jdkVersionOption` - JDK Version

Input alias: `jdkVersion`. `string`. Optional. Use when `javaHomeSelection = JDKVersion`. Allowed values: `default`, `1.9` (JDK 9), `1.8` (JDK 8), `1.7` (JDK 7), `1.6` (JDK 6). Default value: `default`.

Attempts to discover the path to the selected JDK version and set JAVA_HOME accordingly.

jdkDirectory - JDK Path

Input alias: `jdkUserInputPath`. `string`. Required when `javaHomeSelection = Path`.

Sets JAVA_HOME to the given path.

jdkArchitectureOption - JDK Architecture

Input alias: `jdkArchitecture`. `string`. Optional. Use when `jdkVersion != default`.

Allowed values: `x86`, `x64`. Default value: `x64`.

Supplies the JDK architecture (x86 or x64).

gradleOptions - Set GRADLE_OPTS

Input alias: `gradleOpts`. `string`. Default value: `-Xmx1024m`.

Sets the GRADLE_OPTS environment variable, which is used to send command-line arguments to start the JVM. The `xmx` flag specifies the maximum memory available to the JVM.

sonarQubeRunAnalysis - Run SonarQube Analysis

Input alias: `sqAnalysisEnabled`. `boolean`. Default value: `false`.

Runs a SonarQube analysis after executing the current goals. `install` or `package` goals should be executed first.

sonarQubeServiceEndpoint - SonarQube Endpoint

Input alias: `sqConnectedServiceName`. `string`. Required when `sqAnalysisEnabled = true`.

The endpoint that specifies the SonarQube server to use.

sonarQubeProjectName - SonarQube Project Name

Input alias: `sqProjectName`. `string`. Required when `sqAnalysisEnabled = true`.

The SonarQube project name, that is `sonar.projectName`.

sonarQubeProjectKey - SonarQube Project Key

Input alias: `sqProjectKey`. `string`. Required when `sqAnalysisEnabled = true`.

The SonarQube project unique key, that is `sonar.projectKey`.

sonarQubeProjectVersion - SonarQube Project Version

Input alias: `sqProjectVersion`. `string`. Required when `sqAnalysisEnabled = true`.

The SonarQube project version, that is `sonar.projectVersion`.

sonarQubeGradlePluginVersion - SonarQube Gradle Plugin Version

Input alias: `sqGradlePluginVersion`. `string`. Required when `sqAnalysisEnabled = true`.

Default value: `2.0.1`.

Contains the version number of the [SpotBugs Gradle plugin](#).

sonarQubeSpecifyDB - The SonarQube server version is lower than 5.2

Input alias: `sqDbDetailsRequired`. `boolean`. Optional. Use when `sqAnalysisEnabled = true`. Default value: `false`.

SonarQube server 5.1 and lower only. Specifies the database connection details.

sonarQubeDBUrl - Db Connection String

Input alias: `sqDbUrl`. `string`. Optional. Use when `sqDbDetailsRequired = true`.

SonarQube server version 5.1 and lower only. Enters the database connection setting, that is `sonar.jdbc.url`. For example:

`jdbc:jtds:sqlserver://localhost/sonar;SelectMethod=Cursor`.

sonarQubeDBUsername - Db Username

Input alias: `sqDbUsername`. `string`. Optional. Use when `sqDbDetailsRequired = true`.

SonarQube server 5.1 and lower only. Enters the username for the database user, that is `sonar.jdbc.username`.

sonarQubeDBPassword - Db User Password

Input alias: `sqDbPassword`. `string`. Optional. Use when `sqDbDetailsRequired = true`.

SonarQube server 5.1 and lower only. Enter the password for the database user, that is `sonar.jdbc.password`.

`sonarQubeIncludeFullReport` - Include full analysis report in the build summary (SQ 5.3+)

Input alias: `sqAnalysisIncludeFullReport`. `boolean`. Optional. Use when `sqAnalysisEnabled = true`. Default value: `true`.

Delays the build until the SonarQube analysis is completed.

`sonarQubeFailWhenQualityGateFails` - Fail the build on quality gate failure (SQ 5.3+)

Input alias: `sqAnalysisBreakBuildIfQualityGateFailed`. `boolean`. Optional. Use when `sqAnalysisEnabled = true`.

SonarQube server version 5.3 or above only. Introduces delays as the build must wait for SonarQube to complete the analysis. Learn more about [using SonarQube for builds](#).

`checkStyleRunAnalysis` - Run Checkstyle

Input alias: `checkstyleAnalysisEnabled`. `boolean`. Default value: `false`.

Runs the Checkstyle tool with the default Sun checks. Results are uploaded as build artifacts.

`findBugsRunAnalysis` - Run FindBugs

Input alias: `findbugsAnalysisEnabled`. `boolean`. Default value: `false`.

Uses the FindBugs static analysis tool to look for bugs in the code. Results are uploaded as build artifacts. In Gradle 6.0, [this plugin was removed](#). Use the SpotBugs plugin instead.

`pmdRunAnalysis` - Run PMD

Input alias: `pmdAnalysisEnabled`. `boolean`. Default value: `false`.

Uses the PMD Java static analysis tool to look for bugs in the code. The results are uploaded as build artifacts.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Configuration of the SonarQube analysis was moved to the [SonarQube](#) or [SonarCloud](#) extensions in the task `Prepare Analysis Configuration`.

Use this task to build using a Gradle wrapper script.

How do I generate a wrapper from my Gradle project?

The Gradle wrapper allows the build agent to download and configure the exact Gradle environment that is checked into the repository without having any software configuration on the build agent itself other than the JVM.

1. Create the Gradle wrapper by issuing the following command from the root project directory where your `build.gradle` resides:

```
jamal@fabrikam> gradle wrapper
```

2. Upload your Gradle wrapper to your remote repository.

There is a binary artifact that is generated by the gradle wrapper (located at `gradle/wrapper/gradle-wrapper.jar`). This binary file is small and doesn't require updating. If you need to change the Gradle configuration run on the build agent, you update the `gradle-wrapper.properties`.

The repository should look something like this:

```
|-- gradle/
|   '-- wrapper/
|       '-- gradle-wrapper.jar
|       '-- gradle-wrapper.properties
|-- src/
|-- .gitignore
|-- build.gradle
```

```
|-- gradlew  
|-- gradlew.bat
```

How do I fix timeouts when downloading dependencies?

To fix errors such as `Read timed out` when downloading dependencies, users of Gradle 4.3+ can change the timeout by adding `-Dhttp.socketTimeout=60000 -Dhttp.connectionTimeout=60000` to `Options`. This increases the timeout from 10 seconds to 1 minute.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: java
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.91.0 or greater
Task category	Build

Grunt@0 - Grunt v0 task

Article • 09/26/2023

Use this task to run the Grunt JavaScript task runner.

Syntax

YAML

```
# Grunt v0
# Run the Grunt JavaScript task runner.
- task: Grunt@0
  inputs:
    gruntFile: 'gruntfile.js' # string. Required. Grunt File Path. Default: gruntfile.js.
    #targets: # string. Grunt Task(s).
    #arguments: # string. Arguments.
  # Advanced
    #workingDirectory: # string. Alias: cwd. Working Directory.
    gruntCli: 'node_modules/grunt-cli/bin/grunt' # string. Required. grunt-cli location. Default: node_modules/grunt-cli/bin/grunt.
    # JUnit Test Results
    #publishJUnitResults: false # boolean. Publish to Azure Pipelines.
  Default: false.
    #testResultsFiles: '**/TEST-*.xml' # string. Required when publishJUnitResults = true. Test Results Files. Default: **/TEST-*.xml.
    #testRunTitle: # string. Optional. Use when publishJUnitResults = true. Test Run Title.
    # Code Coverage
    #enableCodeCoverage: false # boolean. Enable Code Coverage. Default: false.
    #testFramework: 'Mocha' # 'Mocha' | 'Jasmine'. Optional. Use when enableCodeCoverage = true. Test Framework. Default: Mocha.
    #srcFiles: # string. Optional. Use when enableCodeCoverage = true. Source Files.
    #testFiles: 'test/*.js' # string. Required when enableCodeCoverage = true. Test Script Files. Default: test/*.js.
```

Inputs

`gruntFile` - Grunt File Path

`string`. Required. Default value: `gruntfile.js`.

Specifies the relative path from the repo root to the Grunt script.

targets - Grunt Task(s)`string.`

Optional. Specifies the space-delimited list of tasks to run. If not specified, the default task will run.

arguments - Arguments`string.`

Specifies the additional arguments passed to Grunt. See [Using the CLI](#) for more information.

Note: `--gruntfile` is not needed because it was already added via the `gruntFile` input above.

workingDirectory - Working Directory

Input alias: `cwd`. `string.`

Optional. Specifies the current working directory when the script is run. If not specified, the working directory defaults to the folder where the script is located.

gruntCli - grunt-cli location

`string`. Required. Default value: `node_modules/grunt-cli/bin/grunt`.

Specifies the grunt-cli to run when the agent can't find the globally installed grunt-cli. Defaults to the grunt-cli under the `node_modules` folder of the working directory.

publishJUnitResults - Publish to Azure Pipelines

`boolean`. Default value: `false`.

Select this option to publish the JUnit test results produced by the Grunt build to Azure Pipelines/TFS.

testResultsFiles - Test Results Files

`string`. Required when `publishJUnitResults = true`. Default value: `**/TEST-*.xml`.

Specifies the test results files path. Wildcards can be used.

For example, `**/TEST-*.xml` for all XML file names that start with `TEST-`.

testRunTitle - Test Run Title

`string`. Optional. Use when `publishJUnitResults = true`.

Specifies a name for the test run.

enableCodeCoverage - Enable Code Coverage

`boolean`. Default value: `false`.

Select this option to enable code coverage using Istanbul.

testFramework - Test Framework

`string`. Optional. Use when `enableCodeCoverage = true`. Allowed values: `Mocha`, `Jasmine`. Default value: `Mocha`.

Specifies your test framework.

srcFiles - Source Files

`string`. Optional. Use when `enableCodeCoverage = true`.

Specifies the path to your source files which you want to `hookRequire()`.

testFiles - Test Script Files

`string`. Required when `enableCodeCoverage = true`. Default value: `test/*.js`.

Specifies the path to your test script files.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to run Grunt tasks using the JavaScript Task Runner.

Examples

See a [Sample Gruntfile](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: node.js
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.91.0 or greater
Task category	Build

gulp@1 - gulp v1 task

Article • 09/26/2023

Use this task to run the gulp Node.js streaming task-based build system.

Syntax

YAML

```
# gulp v1
# Run the gulp Node.js streaming task-based build system.
- task: gulp@1
  inputs:
    #gulpFile: 'gulpfile.js' # string. gulp File Path. Default: gulpfile.js.
    #targets: # string. gulp Task(s).
    #arguments: # string. Arguments.
  # Advanced
  #workingDirectory: # string. Alias: cwd. Working Directory.
  #gulpjs: # string. gulp.js location.
  # JUnit Test Results
  #publishJUnitResults: false # boolean. Publish to Azure Pipelines.
  Default: false.
  #testResultsFiles: '**/TEST-*.xml' # string. Required when
  publishJUnitResults = true. Test Results Files. Default: **/TEST-*.xml.
  #testRunTitle: # string. Optional. Use when publishJUnitResults = true.
  Test Run Title.
  # Code Coverage
  #enableCodeCoverage: false # boolean. Enable code Coverage. Default:
  false.
  #testFramework: 'Mocha' # 'Mocha' | 'Jasmine'. Optional. Use when
  enableCodeCoverage = true. Test Framework. Default: Mocha.
  #srcFiles: # string. Optional. Use when enableCodeCoverage = true.
  Source Files.
  #testFiles: 'test/*.js' # string. Required when enableCodeCoverage =
  true. Test Script Files. Default: test/*.js.
```

Inputs

`gulpFile` - gulp File Path

`string`. Default value: `gulpfile.js`.

The relative path from the repo root of the gulp file script file you want to run.

targets - gulp Task(s)`string.`

Optional space-delimited list of tasks to run. If this input isn't specified, the default task will run.

arguments - Arguments`string.`

Additional arguments passed to gulp. `--gulpfile` is not needed since it's already added via `gulpFile` input above.

workingDirectory - Working Directory

Input alias: `cwd`. `string.`

The current working directory to use when the script is run. This input defaults to the folder where the script is located.

gulpjs - gulp.js location`string.`

Path to an alternative `gulp.js`, relative to the working directory.

publishJUnitResults - Publish to Azure Pipelines`boolean`. Default value: `false`.

Publishes JUnit test results produced by the gulp build to Azure Pipelines/TFS.

testResultsFiles - Test Results Files`string`. Required when `publishJUnitResults = true`. Default value: `**/TEST-*.xml`.

Test results files path. You can use wildcards. For example, you can use `**/TEST-*.xml` for all XML files whose name starts with `TEST-`.

testRunTitle - Test Run Title`string`. Optional. Use when `publishJUnitResults = true`.

Provides a name for the test run.

`enableCodeCoverage` - Enable code Coverage

`boolean`. Default value: `false`.

Enables Code Coverage using Istanbul.

`testFramework` - Test Framework

`string`. Optional. Use when `enableCodeCoverage = true`. Allowed values: `Mocha`, `Jasmine`. Default value: `Mocha`.

Specifies your test framework.

`srcFiles` - Source Files

`string`. Optional. Use when `enableCodeCoverage = true`.

Provides the path to the source files you want to hookRequire().

`testFiles` - Test Script Files

`string`. Required when `enableCodeCoverage = true`. Default value: `test/*.js`.

Provides the path to your test script files.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to run gulp tasks using the Node.js streaming task-based build system.

Note

Gulp is not preinstalled on all hosted agents. See [installed software on virtual machine images](#).

Examples

Run gulp.js

```
yml  
  
- task: Npm@1  
  inputs:  
    command: 'install'  
  
- task: gulp@1  
  inputs:  
    gulpFile: 'gulpfile.js'  
    gulpjs: 'node_modules/gulp/bin/gulp.js'
```

Build a Node.js app

- Build your Node.js app with gulp

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: node.js
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.91.0 or greater
Task category	Build

See also

- [Build your Node.js app with gulp](#)

gulp@0 - gulp v0 task

Article • 09/26/2023

Use this task to run the gulp Node.js streaming task-based build system.

Syntax

YAML

```
# gulp v0
# Run the gulp Node.js streaming task-based build system.
- task: gulp@0
  inputs:
    gulpfile: 'gulpfile.js' # string. Required. gulp File Path. Default:
    gulpfile.js.
    #targets: # string. gulp Task(s).
    #arguments: # string. Arguments.
  # Advanced
    #workingDirectory: # string. Alias: cwd. Working Directory.
    gulpjs: 'node_modules/gulp/bin/gulp.js' # string. Required. gulp.js
    location. Default: node_modules/gulp/bin/gulp.js.
    # JUnit Test Results
    #publishJUnitResults: false # boolean. Publish to Azure Pipelines.
    Default: false.
    #testResultsFiles: '**/TEST-*.xml' # string. Required when
    publishJUnitResults = true. Test Results Files. Default: **/TEST-*.xml.
    #testRunTitle: # string. Optional. Use when publishJUnitResults = true.
    Test Run Title.
    # Code Coverage
    #enableCodeCoverage: false # boolean. Enable code Coverage. Default:
    false.
    #testFramework: 'Mocha' # 'Mocha' | 'Jasmine'. Optional. Use when
    enableCodeCoverage = true. Test Framework. Default: Mocha.
    #srcFiles: # string. Optional. Use when enableCodeCoverage = true.
    Source Files.
    #testFiles: 'test/*.js' # string. Required when enableCodeCoverage =
    true. Test Script Files. Default: test/*.js.
```

Inputs

`gulpfile` - gulp File Path

`string`. Required. Default value: `gulpfile.js`.

The relative path from the repo root of the gulp file script file you want to run.

targets - gulp Task(s)`string.`

Optional space-delimited list of tasks to run. If this input isn't specified, the default task will run.

arguments - Arguments`string.`

Additional arguments passed to gulp. `--gulpfile` is not needed since it's already added via `gulpFile` input above.

workingDirectory - Working Directory

Input alias: `cwd`. `string.`

The current working directory to use when the script is run. This input defaults to the folder where the script is located.

gulpjs - gulp.js location

`string`. Required. Default value: `node_modules/gulp/bin/gulp.js`.

Runs `gulp.js` when the agent can't find global-installed gulp. This input defaults to the `gulp.js` installed under the `node_modules` folder of the working directory.

publishJUnitResults - Publish to Azure Pipelines

`boolean`. Default value: `false`.

Publishes JUnit test results from the gulp build to Azure Pipelines/TFS.

testResultsFiles - Test Results Files

`string`. Required when `publishJUnitResults = true`. Default value: `**/TEST-*.xml`.

The test results files path. You can use wildcards. For example, you can use `**/TEST-*.xml` for all XML files whose name starts with `TEST-`.

testRunTitle - Test Run Title

`string`. Optional. Use when `publishJUnitResults = true`.

Provides a name for the test run.

`enableCodeCoverage` - Enable code Coverage

`boolean`. Default value: `false`.

Enables Code Coverage using Istanbul.

`testFramework` - Test Framework

`string`. Optional. Use when `enableCodeCoverage = true`. Allowed values: `Mocha`, `Jasmine`. Default value: `Mocha`.

Specifies your test framework.

`srcFiles` - Source Files

`string`. Optional. Use when `enableCodeCoverage = true`.

Provides the path to the source files that you want to hookRequire().

`testFiles` - Test Script Files

`string`. Required when `enableCodeCoverage = true`. Default value: `test/*.js`.

Provides the path to your test script files.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to run gulp tasks using the Node.js streaming task-based build system.

Note

Gulp is not preinstalled on all hosted agents. See [installed software on virtual machine images](#).

Examples

Run gulp.js

```
yml  
  
- task: Npm@1  
  inputs:  
    command: 'install'  
  
- task: gulp@0  
  inputs:  
    gulpFile: 'gulpfile.js'  
    gulpjs: 'node_modules/gulp/bin/gulp.js'
```

Build a Node.js app

- Build your Node.js app with gulp

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: node.js
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.91.0 or greater
Task category	Build

HelmInstaller@1 - Helm tool installer v1 task

Article • 09/26/2023

Use this task to install Helm on an agent machine.

Syntax

YAML

```
# Helm tool installer v1
# Install Helm on an agent machine.
- task: HelmInstaller@1
  inputs:
    helmVersionToInstall: 'latest' # string. Helm Version Spec. Default:
latest.
```

Inputs

`helmVersionToInstall` - Helm Version Spec

`string`. Default value: `latest`.

Specifies the version of Helm to install. Acceptable values include any semantic version string, like `2.14.1`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

This task can be used for installing a specific version of helm binary on agents.

Troubleshooting

HelmInstaller task running on a private agent behind a proxy fails to download helm package

The HelmInstaller task does not use the proxy settings to download the file <https://get.helm.sh/helm-v3.1.0-linux-amd64.zip>. You can work around this by pre-installing Helm on your private agents.

Examples

The following YAML example showcases the installation of latest version of helm binary on the agent -

```
YAML  
  
- task: HelmInstaller@1  
  displayName: Helm installer  
  inputs:  
    helmVersionToInstall: latest
```

The following YAML example demonstrates the use of an explicit version string rather than installing the latest version available at the time of task execution -

```
YAML  
  
- task: HelmInstaller@1  
  displayName: Helm installer  
  inputs:  
    helmVersionToInstall: 2.14.1
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: Helm

Requirement	Description
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Tool

HelmInstaller@0 - Helm tool installer v0 task

Article • 09/26/2023

Use this task to install Helm and Kubernetes on an agent machine.

Syntax

YAML

```
# Helm tool installer v0
# Install Helm and Kubernetes on an agent machine.
- task: HelmInstaller@0
  inputs:
    helmVersion: '2.14.1' # string. Required. Helm Version Spec. Default: 2.14.1.
    #checkLatestHelmVersion: true # boolean. Check for latest version of Helm. Default: true.
    # Prerequisite
    #installKubeCtl: true # boolean. Install Kubectl. Default: true.
    #kubectlVersion: '1.8.9' # string. Optional. Use when installKubeCtl == true. Kubectl Version Spec. Default: 1.8.9.
    #checkLatestKubeCtl: true # boolean. Optional. Use when installKubeCtl == true. Check for latest version of kubectl. Default: true.
```

Inputs

`helmVersion` - Helm Version Spec

`string`. Required. Default value: `2.14.1`.

Specifies the version of Helm to install.

`checkLatestHelmVersion` - Check for latest version of Helm

`boolean`. Default value: `true`.

Checks for the latest version of Helm.

`installKubeCtl` - Install Kubectl

`boolean`. Default value: `true`.

Installs Kubectl.

`kubectlVersion` - Kubectl Version Spec

`string`. Optional. Use when `installKubeCtl == true`. Default value: `1.8.9`.

Specifies the version of Kubectl to install.

`checkLatestKubeCtl` - Check for latest version of kubectl

`boolean`. Optional. Use when `installKubeCtl == true`. Default value: `true`.

Checks for the latest version of Kubectl.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: Helm
Command restrictions	Any
Settable variables	Any
Agent version	2.115.0 or greater
Task category	Tool

IISWebAppDeploymentOnMachineGroup@0 - IIS web app deploy v0 task

Article • 09/26/2023

Use this task to deploy a website or web application using Web Deploy.

Syntax

YAML

```
# IIS web app deploy v0
# Deploy a website or web application using Web Deploy.
- task: IISWebAppDeploymentOnMachineGroup@0
  inputs:
    WebSiteName: # string. Required. Website Name.
    #VirtualApplication: # string. Virtual Application.
    Package: '$(System.DefaultWorkingDirectory)\**\*.zip' # string.
    Required. Package or Folder. Default:
    $(System.DefaultWorkingDirectory)\**\*.zip.
    # Advanced Deployment Options
    #SetParametersFile: # string. SetParameters File.
    #RemoveAdditionalFilesFlag: false # boolean. Remove Additional Files at
    Destination. Default: false.
    #ExcludeFilesFromAppDataFlag: false # boolean. Exclude Files from the
    App_Data Folder. Default: false.
    #TakeAppOfflineFlag: false # boolean. Take App Offline. Default: false.
    #AdditionalArguments: # string. Additional Arguments.
    # File Transforms & Variable Substitution Options
    #XmlTransformation: false # boolean. XML transformation. Default: false.
    #XmlVariableSubstitution: false # boolean. XML variable substitution.
    Default: false.
    #JSONFiles: # string. JSON variable substitution.
```

Inputs

`WebSiteName` - Website Name

`string`. Required.

Specifies the name of an existing website on the machine group machines.

`VirtualApplication` - Virtual Application

`string`.

Specifies the name of an already existing Azure Virtual application on the target machines.

Package - Package or Folder

`string`. Required. Default value: `$(System.DefaultWorkingDirectory)***.zip`.

Specifies the file path to the package or folder generated by MSBuild or a compressed archive file. Variables ([Build](#) | [Release](#)) and wildcards are supported. For example, `$(System.DefaultWorkingDirectory)***.zip`.

SetParametersFile - SetParameters File

`string`.

Optional. Specifies the location of the `SetParameters.xml` file to use.

RemoveAdditionalFilesFlag - Remove Additional Files at Destination

`boolean`. Default value: `false`.

Selects the option to delete files on the Web App that have no matching files in the Web App zip package.

ExcludeFilesFromAppDataFlag - Exclude Files from the App_Data Folder

`boolean`. Default value: `false`.

Selects the option to prevent files in the `App_Data` folder from being deployed to the Web App.

TakeAppOfflineFlag - Take App Offline

`boolean`. Default value: `false`.

Selects the option to take the Web App offline by placing an `app_offline.htm` file in the root directory of the Web App before the sync operation begins. The file will be removed after the sync operation completes successfully.

AdditionalArguments - Additional Arguments

`string`.

Specifies additional Web Deploy arguments that are applied when deploying the Azure Web App. For example, `-disableLink:AppPoolExtension` or `- disableLink:ContentExtension`.

For a list of Web Deploy arguments, see [Web Deploy Operation Settings](#).

XmlTransformation - XML transformation

`boolean`. Default value: `false`.

Specifies the config transforms that are run for `*.Release.config` and `*`.

`<EnvironmentName>.config` on the `*.config` file. Config transforms are run prior to the Variable Substitution. XML transformations are only supported on Windows.

XmlVariableSubstitution - XML variable substitution

`boolean`. Default value: `false`.

Specifies the variables defined in the build or release pipeline. These variables are matched against the `key` or `name` entries in the `appSettings`, `applicationSettings`, and `connectionStrings` sections of any config file and `parameters.xml`. Variable Substitution is run after config transforms.

Note: If the same variables are defined in the release pipeline and in the environment, then the environment variables will supersede the release pipeline variables.

JSONFiles - JSON variable substitution

`string`.

Specifies a new line separated list of JSON files to substitute the variable values. File names must be relative to the root folder.

To substitute JSON variables that are nested or hierarchical, specify them using JSONPath expressions. For example, to replace the value of `ConnectionString` in the sample below, you must define a variable as `Data.DefaultConnection.ConnectionString` in the build or release pipeline (or in the release pipeline's stage).

```
{  
  "Data": {  
    "DefaultConnection": {  
      "ConnectionString": "Server= (localdb)\\SQLEXPRESS;Database=MyDB;Trusted_Connection=True"  
    }  
  }  
}
```

```
    }  
}  
}
```

Variable Substitution is run after configuration transforms.

Note: Pipeline variables are excluded in substitution.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to deploy a website or web app using WebDeploy.

Requirements

Requirement	Description
Pipeline types	Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.104.1 or greater
Task category	Deploy

IISWebAppDeployment@1 - IIS Web App deployment (Deprecated) v1 task

Article • 09/26/2023

Use this task to deploy IIS Web App using MSDeploy, then create or update websites and app pools.

This task is deprecated.

Syntax

YAML

```
# IIS Web App deployment (Deprecated) v1
# Deploy using MSDeploy, then create/update websites and app pools.
- task: IISWebAppDeployment@1
  inputs:
    EnvironmentName: # string. Required. Machines.
    #AdminUserName: # string. Admin Login.
    #AdminPassword: # string. Password.
    #WinRMProtocol: # 'Http' | 'Https'. Protocol.
    #TestCertificate: true # boolean. Optional. Use when WinRMProtocol =
    Https. Test Certificate. Default: true.
    # Deployment
    WebDeployPackage: # string. Required. Web Deploy Package.
    #WebDeployParamFile: # string. Web Deploy Parameter File.
    #OverRideParams: # string. Override Parameters.
    # Website
    #CreateWebSite: false # boolean. Create or Update Website. Default:
    false.
    #WebSiteName: # string. Required when CreateWebSite = true. Website
    Name.
    #WebSitePhysicalPath: '%SystemDrive%\inetpub\wwwroot' # string. Required
    when CreateWebSite = true. Physical Path. Default:
    %SystemDrive%\inetpub\wwwroot.
    #WebSitePhysicalPathAuth: 'Application User (Pass-through)' #
    'WebSiteUserPassThrough' | 'WebSiteWindowsAuth'. Required when CreateWebSite
    = true. Physical Path Authentication. Default: Application User (Pass-
    through).
    #WebSiteAuthUserName: # string. Required when WebSitePhysicalPathAuth =
    WebSiteWindowsAuth. User Name.
    #WebSiteAuthUserPassword: # string. Optional. Use when
    WebSitePhysicalPathAuth = WebSiteWindowsAuth. Password.
    #AddBinding: true # boolean. Optional. Use when CreateWebSite = true.
    Add Binding. Default: true.
    #AssignDuplicateBinding: false # boolean. Optional. Use when AddBinding
    = true. Assign Duplicate Binding. Default: false.
    Protocol: 'http' # 'https' | 'http'. Required when AddBinding = true.
```

```

Protocol. Default: http.
  IPAddress: 'All Unassigned' # string. Required when AddBinding = true.
IP Address. Default: All Unassigned.
  Port: '80' # string. Required when AddBinding = true. Port. Default: 80.
  #ServerNameIndication: false # boolean. Optional. Use when Protocol =
https. Server Name Indication Required. Default: false.
  #HostNameWithOutSNI: # string. Optional. Use when ServerNameIndication =
false. Host Name.
  #HostNameWithHttp: # string. Optional. Use when Protocol = http. Host
Name.
  #HostNameWithSNI: # string. Required when ServerNameIndication = true.
Host Name.
  #SSLCertThumbPrint: # string. Required when Protocol = https. SSL
Certificate Thumb Print.
# Application Pool
  #CreateAppPool: false # boolean. Create or Update Application Pool.
Default: false.
  #AppPoolName: # string. Required when CreateAppPool = true. Name.
  #DotNetVersion: 'v4.0' # 'v4.0' | 'v2.0' | 'No Managed Code'. Required
when CreateAppPool = true. .NET Version. Default: v4.0.
  #PipeLineMode: 'Integrated' # 'Integrated' | 'Classic'. Required when
CreateAppPool = true. Managed Pipeline Mode. Default: Integrated.
  #AppPoolIdentity: 'ApplicationPoolIdentity' # 'ApplicationPoolIdentity'
| 'LocalService' | 'LocalSystem' | 'NetworkService' | 'SpecificUser'.
Required when CreateAppPool = true. Identity. Default:
ApplicationPoolIdentity.
  #AppPoolUsername: # string. Required when AppPoolIdentity =
SpecificUser. Username.
  #AppPoolPassword: # string. Optional. Use when AppPoolIdentity =
SpecificUser. Password.
# Advanced
  #AppCmdCommands: # string. Additional AppCmd.exe Commands.
  #DeployInParallel: true # boolean. Deploy in Parallel. Default: true.
  #ResourceFilteringMethod: 'machineNames' # 'machineNames' | 'tags'.
Select Machines By. Default: machineNames.
  #MachineFilter: # string. Deploy to Machines.

```

Inputs

`EnvironmentName` - Machines

`string`. Required.

Specifies a comma-separated list of machine IP addresses or FQDNs, along with ports. The default port is based on the selected protocol, for example `dbserver.fabrikam.com` or `dbserver_int.fabrikam.com:5986,192.168.12.34:5986`. You can also provide the output variable of other tasks, for example `$(variableName)`.

AdminUserName - Admin Login`string.`

Specifies the administrator login for the target machines.

AdminPassword - Password`string.`

Specifies the administrator password for the target machines. It can accept variables defined in build/release definitions, such as `$(passwordVariable)`. You may mark the variable type as `secret` to secure it.

WinRMProtocol - Protocol`string`. Allowed values: `Http`, `Https`.

Specifies the protocol used for the WinRM connection with the machine(s). The default is `HTTPS`.

TestCertificate - Test Certificate`boolean`. Optional. Use when `WinRMProtocol = Https`. Default value: `true`.

Selects the option to skip validating the authenticity of the machine's certificate by a trusted certification authority. The parameter is required for the WinRM `HTTPS` protocol.

WebDeployPackage - Web Deploy Package`string`. Required.

Specifies the location of the Web Deploy (MSDeploy) zip file on the target machines or on a UNC path like, `\BudgetIT\WebDeploy\WebDeployPackage.zip`. The UNC path should be accessible to the machine's administrator account. Environment variables are also supported, like `$env:windir`, `$env:systemroot`, and `$env:windir\FabrikamFibre\Web`.

WebDeployParamFile - Web Deploy Parameter File`string`.

Specifies the location of the parameter file on the target machines or on a UNC path. The parameter file is used to override Web application configuration settings, like the IIS Web application name or database connection string.

OverRideParams - Override Parameters

`string`.

Parameters specified here will override the parameters in the MSDeploy zip file and the parameter file. To override more than one parameter, use a line separator.

For example, `"IIS Web Application Name"="Fabrikam"` or
`"ConnectionString"="Server=localhost;Database=Fabrikam;"`.

CreateWebSite - Create or Update Website

`boolean`. Default value: `false`.

Specifies the option to create a website or to update an existing website.

WebSiteName - Website Name

`string`. Required when `CreateWebSite = true`.

Specifies the name of the IIS website that will be created if it does not exist, or it will be updated if it is already present on the IIS server. The name of the website should be the same as that specified in the web deploy zip package file. If a parameter file and override parameters setting is also specified, then the name of the website should be the same as that in the override parameters setting.

WebSitePhysicalPath - Physical Path

`string`. Required when `CreateWebSite = true`. Default value:
`%SystemDrive%\inetpub\wwwroot`.

Specifies the physical path where the website content is stored. The content can reside on the local computer or on a remote directory or share, like `C:\Fabrikam` or
`\ContentShare\Fabrikam`.

WebSitePhysicalPathAuth - Physical Path Authentication

`string`. Required when `CreateWebSite = true`. Allowed values: `WebSiteUserPassThrough` (Application User (Pass-through)), `WebSiteWindowsAuth` (Windows Authentication). Default value: `Application User (Pass-through)`.

Specifies the authentication mechanism for accessing the physical path of the website.

WebSiteAuthUserName - User Name

`string`. Required when `WebSitePhysicalPathAuth = WebSiteWindowsAuth`.

Specifies the user name for accessing the website's physical path.

WebSiteAuthUserPassword - Password

`string`. Optional. Use when `WebSitePhysicalPathAuth = WebSiteWindowsAuth`.

Specifies the password for accessing the website's physical path. If you are using a gMSA, this is not required.

AddBinding - Add Binding

`boolean`. Optional. Use when `CreateWebSite = true`. Default value: `true`.

Specifies the option to add port binding for the website.

AssignDuplicateBinding - Assign Duplicate Binding

`boolean`. Optional. Use when `AddBinding = true`. Default value: `false`.

Specifies the option to add the bindings specified here (even if there is another website with the same bindings). If there are binding conflicts, only one of the websites will start.

Protocol - Protocol

`string`. Required when `AddBinding = true`. Allowed values: `https`, `http`. Default value: `http`.

Specifies either HTTP for the website to have an HTTP binding or HTTPS for the website to have a Secure Sockets Layer (SSL) binding.

IPAddress - IP Address

`string`. Required when `AddBinding = true`. Default value: `All Unassigned`.

Specifies an IP address that users can use to access the website. If `All Unassigned` is selected, the site will respond to requests for all IP addresses on the port and the optional host name that is specified for the site. The site will not respond to requests if another site on the server has a binding on the same port but with a specific IP address.

Port - Port

`string`. Required when `AddBinding = true`. Default value: `80`.

Specifies the port on which Hypertext Transfer Protocol Stack (HTTP.sys) must monitor for requests made to this website.

ServerNameIndication - Server Name Indication Required

`boolean`. Optional. Use when `Protocol = https`. Default value: `false`.

Determines whether the website requires Server Name Indication (SNI). SNI extends the SSL and TLS protocols to indicate what host name the client is attempting to connect to. It allows multiple secure websites with different certificates to use the same IP address.

HostNameWithoutSNI - Host Name

`string`. Optional. Use when `ServerNameIndication = false`.

Assigns one or more host names (or domain names) to a computer that uses a single IP address. If a host name is specified then the clients must use the host name instead of the IP address to access the website.

HostNameWithHttp - Host Name

`string`. Optional. Use when `Protocol = http`.

Assigns one or more host names (or domain names) to a computer that uses a single IP address. If a host name is specified then the clients must use the host name instead of the IP address to access the website.

HostNameWithSNI - Host Name

`string`. Required when `ServerNameIndication = true`.

Assigns one or more host names (or domain names) to a computer that uses a single IP address. If a host name is specified then the clients must use the host name instead of the IP address to access the website.

SSLCertThumbPrint - SSL Certificate Thumb Print

`string`. Required when `Protocol = https`.

Specifies the thumb-print of the Secure Socket Layer certificate that the website is going to use. The certificate should already be installed on the machine and present under the

Local Computer Personal store.

CreateAppPool - Create or Update Application Pool

`boolean`. Default value: `false`.

Specifies the option to create an application pool or to update an existing application pool.

AppPoolName - Name

`string`. Required when `CreateAppPool = true`.

Specifies the name of the IIS application pool to create or update. The existing application pool will be updated with the settings specified.

DotNetVersion - .NET Version

`string`. Required when `CreateAppPool = true`. Allowed values: `v4.0`, `v2.0`, `No Managed Code`. Default value: `v4.0`.

Specifies the version of the .NET Framework that is loaded by this application pool. If the applications assigned to this application pool do not contain managed code, select the **No Managed Code** option from the list.

PipeLineMode - Managed Pipeline Mode

`string`. Required when `CreateAppPool = true`. Allowed values: `Integrated`, `Classic`. Default value: `Integrated`.

Managed pipeline mode specifies how IIS processes requests for managed content. Use classic mode only when the applications in the application pool cannot run in integrated mode.

AppPoolIdentity - Identity

`string`. Required when `CreateAppPool = true`. Allowed values: `ApplicationPoolIdentity`, `LocalService`, `LocalSystem`, `NetworkService`, `SpecificUser` (Custom Account). Default value: `ApplicationPoolIdentity`.

Configures the account under which an application pool's worker process runs. Specify one of the predefined security accounts, or configure a custom account.

AppPoolUsername - Username

`string`. Required when `AppPoolIdentity = SpecificUser`.

AppPoolPassword - Password

`string`. Optional. Use when `AppPoolIdentity = SpecificUser`.

If you are using a gMSA, this is not required.

AppCmdCommands - Additional AppCmd.exe Commands

`string`.

Specifies additional `AppCmd.exe` commands to set the website or application pool properties. For more than one command, use a line separator.

For example:

```
<list app pools>
<list sites>
```

DeployInParallel - Deploy in Parallel

`boolean`. Default value: `true`.

If set to `true`, the Web application is deployed in-parallel on the target machines.

ResourceFilteringMethod - Select Machines By

`string`. Allowed values: `machineNames` (Machine Names), `tags`. Default value: `machineNames`.

Optional. Specifies a subset of machines by providing machine names or tags.

MachineFilter - Deploy to Machines

`string`.

This input is only valid for machine groups and is not supported for a flat list of machines or output variables yet.

Specifies a list of machines, like `dbserver.fabrikam.com`, `webserver.fabrikam.com`, `192.168.12.34` or tags, like `Role:DB; OS:Win8.1`. If multiple tags are provided, the task will run in all of the machines with the specified tags. For Azure Resource Groups,

specify the virtual machine's name, like `ffweb`, `ffdb`. The default runs the task in all machines.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.91.0 or greater
Task category	Deploy

IISWebAppManagementOnMachineGroup@0 - IIS web app manage v0 task

Article • 09/26/2023

Use this task to create or update websites, web apps, virtual directories, or application pools.

Syntax

YAML

```
# IIS web app manage v0
# Create or update websites, web apps, virtual directories, or application
pools.
- task: IISWebAppManagementOnMachineGroup@0
  inputs:
    #EnableIIS: false # boolean. Enable IIS. Default: false.
    IISDeploymentType: 'IISWebsite' # 'IISWebsite' | 'IISWebApplication' |
'IISVirtualDirectory' | 'IISApplicationPool'. Required. Configuration type.
Default: IISWebsite.
      ActionIISWebsite: 'CreateOrUpdateWebsite' # 'CreateOrUpdateWebsite' |
'StartWebsite' | 'StopWebsite'. Required when IISDeploymentType =
IISWebsite. Action. Default: CreateOrUpdateWebsite.
      #ActionIISApplicationPool: 'CreateOrUpdateAppPool' #
'CreateOrUpdateAppPool' | 'StartAppPool' | 'StopAppPool' | 'RecycleAppPool'.
Required when IISDeploymentType = IISApplicationPool. Action. Default:
CreateOrUpdateAppPool.
      #StartStopWebsiteName: # string. Required when ActionIISWebsite =
StartWebsite || ActionIISWebsite = StopWebsite. Website name.
      #Protocol: 'http' # 'https' | 'http'. Required when IISDeploymentType =
randomDeployment. Protocol. Default: http.
      #IPAddress: 'All Unassigned' # string. Required when IISDeploymentType =
randomDeployment. IP address. Default: All Unassigned.
      #Port: '80' # string. Required when IISDeploymentType =
randomDeployment. Port. Default: 80.
      #ServerNameIndication: false # boolean. Optional. Use when
IISDeploymentType = randomDeployment. Server Name Indication required.
Default: false.
      #HostNameWithOutSNI: # string. Optional. Use when IISDeploymentType =
randomDeployment. Host name.
      #HostNameWithHttp: # string. Optional. Use when IISDeploymentType =
randomDeployment. Host name.
      #HostNameWithSNI: # string. Required when IISDeploymentType =
randomDeployment. Host name.
      #SSLCertThumbPrint: # string. Required when IISDeploymentType =
randomDeployment. SSL certificate thumbprint.
      #StartStopRecycleAppPoolName: # string. Required when
ActionIISApplicationPool = StartAppPool || ActionIISApplicationPool =
```

```
StopAppPool || ActionIISApplicationPool = RecycleAppPool. Application pool
name.

# IIS Website
  WebsiteName: # string. Required when ActionIISWebsite =
CreateOrUpdateWebsite. Website name.
  WebsitePhysicalPath: '%SystemDrive%\inetpub\wwwroot' # string. Required
when ActionIISWebsite = CreateOrUpdateWebsite. Physical path. Default:
%SystemDrive%\inetpub\wwwroot.
  WebsitePhysicalPathAuth: 'WebsiteUserPassThrough' #
'WebsiteUserPassThrough' | 'WebsiteWindowsAuth'. Required when
ActionIISWebsite = CreateOrUpdateWebsite. Physical path authentication.
Default: WebsiteUserPassThrough.
  #WebsiteAuthUserName: # string. Required when WebsitePhysicalPathAuth =
WebsiteWindowsAuth && ActionIISWebsite = CreateOrUpdateWebsite. Username.
  #WebsiteAuthUserPassword: # string. Optional. Use when
WebsitePhysicalPathAuth = WebsiteWindowsAuth && ActionIISWebsite =
CreateOrUpdateWebsite. Password.
  #AddBinding: false # boolean. Optional. Use when ActionIISWebsite =
CreateOrUpdateWebsite. Add binding. Default: false.
  #CreateOrUpdateAppPoolForWebsite: false # boolean. Optional. Use when
ActionIISWebsite = CreateOrUpdateWebsite. Create or update app pool.
Default: false.
  #ConfigureAuthenticationForWebsite: false # boolean. Optional. Use when
ActionIISWebsite = CreateOrUpdateWebsite. Configure authentication. Default:
false.

# IIS Bindings
  #Bindings: # string. Required when IISDeploymentType = IISWebsite &&
ActionIISWebsite = CreateOrUpdateWebsite && AddBinding = true. Add bindings.
# IIS Application pool
  #AppPoolNameForWebsite: # string. Required when IISDeploymentType =
IISWebsite && ActionIISWebsite = CreateOrUpdateWebsite &&
CreateOrUpdateAppPoolForWebsite = true. Name.
  #DotNetVersionForWebsite: 'v4.0' # 'v4.0' | 'v2.0' | 'No Managed Code'.
Required when IISDeploymentType = IISWebsite && ActionIISWebsite =
CreateOrUpdateWebsite && CreateOrUpdateAppPoolForWebsite = true. .NET
version. Default: v4.0.
  #PipeLineModeForWebsite: 'Integrated' # 'Integrated' | 'Classic'.
Required when IISDeploymentType = IISWebsite && ActionIISWebsite =
CreateOrUpdateWebsite && CreateOrUpdateAppPoolForWebsite = true. Managed
pipeline mode. Default: Integrated.
  #AppPoolIdentityForWebsite: 'ApplicationPoolIdentity' #
'ApplicationPoolIdentity' | 'LocalService' | 'LocalSystem' |
'NetworkService' | 'SpecificUser'. Required when IISDeploymentType =
IISWebsite && ActionIISWebsite = CreateOrUpdateWebsite &&
CreateOrUpdateAppPoolForWebsite = true. Identity. Default:
ApplicationPoolIdentity.
  #AppPoolUsernameForWebsite: # string. Required when
AppPoolIdentityForWebsite = SpecificUser && IISDeploymentType = IISWebsite
&& ActionIISWebsite = CreateOrUpdateWebsite &&
CreateOrUpdateAppPoolForWebsite = true. Username.
  #AppPoolPasswordForWebsite: # string. Optional. Use when
AppPoolIdentityForWebsite = SpecificUser && IISDeploymentType = IISWebsite
&& ActionIISWebsite = CreateOrUpdateWebsite &&
CreateOrUpdateAppPoolForWebsite = true. Password.

# IIS Authentication
```

```
#AnonymousAuthenticationForWebsite: false # boolean. Optional. Use when
IISDeploymentType = IISWebsite && ActionIISWebsite = CreateOrUpdateWebsite
&& ConfigureAuthenticationForWebsite = true. Anonymous authentication.
Default: false.

#BasicAuthenticationForWebsite: false # boolean. Optional. Use when
IISDeploymentType = IISWebsite && ActionIISWebsite = CreateOrUpdateWebsite
&& ConfigureAuthenticationForWebsite = true. Basic authentication. Default:
false.

#WindowsAuthenticationForWebsite: true # boolean. Optional. Use when
IISDeploymentType = IISWebsite && ActionIISWebsite = CreateOrUpdateWebsite
&& ConfigureAuthenticationForWebsite = true. Windows authentication.
Default: true.

# IIS Virtual directory
#ParentWebsiteNameForVD: # string. Required when IISDeploymentType =
IISVirtualDirectory. Parent website name.
#VirtualPathForVD: # string. Required when IISDeploymentType =
IISVirtualDirectory. Virtual path.
#PhysicalPathForVD: '%SystemDrive%\inetpub\wwwroot' # string. Required
when IISDeploymentType = IISVirtualDirectory. Physical path. Default:
%SystemDrive%\inetpub\wwwroot.

#VDPhysicalPathAuth: 'VDUserPassThrough' # 'VDUserPassThrough' |
'VDWindowsAuth'. Optional. Use when IISDeploymentType = IISVirtualDirectory.
Physical path authentication. Default: VDUserPassThrough.

#VDAuthUserName: # string. Required when VDPhysicalPathAuth =
VDWindowsAuth && IISDeploymentType = IISVirtualDirectory. Username.

#VDAuthUserPassword: # string. Optional. Use when VDPhysicalPathAuth =
VDWindowsAuth && IISDeploymentType = IISVirtualDirectory. Password.

# IIS Application
#ParentWebsiteNameForApplication: # string. Required when
IISDeploymentType = IISWebApplication. Parent website name.
#VirtualPathForApplication: # string. Required when IISDeploymentType =
IISWebApplication. Virtual path.
#PhysicalPathForApplication: '%SystemDrive%\inetpub\wwwroot' # string.
Required when IISDeploymentType = IISWebApplication. Physical path. Default:
%SystemDrive%\inetpub\wwwroot.

#ApplicationPhysicalPathAuth: 'ApplicationUserPassThrough' #
'ApplicationUserPassThrough' | 'ApplicationWindowsAuth'. Optional. Use when
IISDeploymentType = IISWebApplication. Physical path authentication.
Default: ApplicationUserPassThrough.

#ApplicationAuthUserName: # string. Required when
ApplicationPhysicalPathAuth = ApplicationWindowsAuth && IISDeploymentType =
IISWebApplication. Username.

#ApplicationAuthUserPassword: # string. Optional. Use when
ApplicationPhysicalPathAuth = ApplicationWindowsAuth && IISDeploymentType =
IISWebApplication. Password.

#CreateOrUpdateAppPoolForApplication: false # boolean. Optional. Use
when IISDeploymentType = IISWebApplication. Create or update app pool.
Default: false.

# IIS Application pool
#AppPoolNameForApplication: # string. Required when IISDeploymentType =
IISWebApplication && CreateOrUpdateAppPoolForApplication = true. Name.
#DotNetVersionForApplication: 'v4.0' # 'v4.0' | 'v2.0' | 'No Managed
Code'. Required when IISDeploymentType = IISWebApplication &&
CreateOrUpdateAppPoolForApplication = true. .NET version. Default: v4.0.
#PipeLineModeForApplication: 'Integrated' # 'Integrated' | 'Classic'.
```

```

Required when IISDeploymentType = IISWebApplication &&
CreateOrUpdateAppPoolForApplication = true. Managed pipeline mode. Default:
Integrated.

    #AppPoolIdentityForApplication: 'ApplicationPoolIdentity' #
'ApplicationPoolIdentity' | 'LocalService' | 'LocalSystem' |
'NetworkService' | 'SpecificUser'. Required when IISDeploymentType =
IISWebApplication && CreateOrUpdateAppPoolForApplication = true. Identity.
Default: ApplicationPoolIdentity.

    #AppPoolUsernameForApplication: # string. Required when
AppPoolIdentityForApplication = SpecificUser && IISDeploymentType =
IISWebApplication && CreateOrUpdateAppPoolForApplication = true. Username.

    #AppPoolPasswordForApplication: # string. Optional. Use when
AppPoolIdentityForApplication = SpecificUser && IISDeploymentType =
IISWebApplication && CreateOrUpdateAppPoolForApplication = true. Password.

    # IIS Application pool

    AppPoolName: # string. Required when ActionIISApplicationPool =
CreateOrUpdateAppPool. Name.

    DotNetVersion: 'v4.0' # 'v4.0' | 'v2.0' | 'No Managed Code'. Required
when ActionIISApplicationPool = CreateOrUpdateAppPool. .NET version.
Default: v4.0.

    PipelineMode: 'Integrated' # 'Integrated' | 'Classic'. Required when
ActionIISApplicationPool = CreateOrUpdateAppPool. Managed pipeline mode.
Default: Integrated.

    AppPoolIdentity: 'ApplicationPoolIdentity' # 'ApplicationPoolIdentity' |
'LocalService' | 'LocalSystem' | 'NetworkService' | 'SpecificUser'. Required
when ActionIISApplicationPool = CreateOrUpdateAppPool. Identity. Default:
ApplicationPoolIdentity.

    #AppPoolUsername: # string. Required when AppPoolIdentity = SpecificUser
&& ActionIISApplicationPool = CreateOrUpdateAppPool. Username.

    #AppPoolPassword: # string. Optional. Use when AppPoolIdentity =
SpecificUser && ActionIISApplicationPool = CreateOrUpdateAppPool. Password.

    # Advanced

    #AppCmdCommands: # string. Additional appcmd.exe commands.

```

Inputs

`EnableIIS` - Enable IIS

`boolean`. Default value: `false`.

Set to `true` if you want to install IIS on the machine.

`IISDeploymentType` - Configuration type

`string`. Required. Allowed values: `IISWebsite` (IIS Website), `IISWebApplication` (IIS Web Application), `IISVirtualDirectory` (IIS Virtual Directory), `IISApplicationPool` (IIS Application Pool). Default value: `IISWebsite`.

Specifies the configuration type: website, web application, virtual directory, or application pool.

ActionIISWebsite - Action

`string`. Required when `IISDeploymentType = IISWebsite`. Allowed values: `CreateOrUpdateWebsite` (Create Or Update), `StartWebsite` (Start), `StopWebsite` (Stop). Default value: `CreateOrUpdateWebsite`.

Specifies the appropriate action that you want to perform on an IIS website.

Create Or Update will create a website or update an existing website.

Start, Stop will start or stop the website respectively.

ActionIISApplicationPool - Action

`string`. Required when `IISDeploymentType = IISApplicationPool`. Allowed values: `CreateOrUpdateAppPool` (Create Or Update), `StartAppPool` (Start), `StopAppPool` (Stop), `RecycleAppPool` (Recycle). Default value: `CreateOrUpdateAppPool`.

Specifies the appropriate action that you want to perform on an IIS application pool.

Create Or Update will create an application pool or update an existing application pool.

Start, Stop, Recycle will start, stop or recycle the application pool respectively.

StartStopWebsiteName - Website name

`string`. Required when `ActionIISWebsite = StartWebsite || ActionIISWebsite = StopWebsite`.

Specifies the name of the IIS website.

WebsiteName - Website name

`string`. Required when `ActionIISWebsite = CreateOrUpdateWebsite`.

Specifies the name of the IIS website to create or update.

WebsitePhysicalPath - Physical path

`string`. Required when `ActionIISWebsite = CreateOrUpdateWebsite`. Default value: `%SystemDrive%\inetpub\wwwroot`.

Specifies the physical path where the website content will be stored. The content can reside on the local computer, in a remote directory, or on a network share, like `C:\Fabrikam` or `\ContentShare\Fabrikam`.

WebsitePhysicalPathAuth - Physical path authentication

`string`. Required when `ActionIISWebsite = CreateOrUpdateWebsite`. Allowed values: `WebsiteUserPassThrough` (Application User (Pass-through)), `WebsiteWindowsAuth` (Windows Authentication). Default value: `WebsiteUserPassThrough`.

Specifies the authentication mechanism that will be used to access the physical path of the website.

WebsiteAuthUserName - Username

`string`. Required when `WebsitePhysicalPathAuth = WebsiteWindowsAuth && ActionIISWebsite = CreateOrUpdateWebsite`.

Specifies the user name that will be used to access the website's physical path.

WebsiteAuthUserPassword - Password

`string`. Optional. Use when `WebsitePhysicalPathAuth = WebsiteWindowsAuth && ActionIISWebsite = CreateOrUpdateWebsite`.

Specifies the user's password that will be used to access the website's physical path. The best practice is to create a variable in the build or release pipeline, mark it as `Secret` to secure it, and then provide it when using this input, like `$(userCredentials)`.

Note: Special characters in the password are interpreted per [command-line arguments](#).

AddBinding - Add binding

`boolean`. Optional. Use when `ActionIISWebsite = CreateOrUpdateWebsite`. Default value: `false`.

Specifies the option to add port binding for the website.

Protocol - Protocol

`string`. Required when `IISDeploymentType = randomDeployment`. Allowed values: `https`, `http`. Default value: `http`.

Specifies either HTTP for the website to have an HTTP binding or HTTPS for the website to have a Secure Sockets Layer (SSL) binding.

IPAddress - IP address

`string`. Required when `IISDeploymentType = randomDeployment`. Default value: `All Unassigned`.

Specifies an IP address that end-users can use to access this website.

If **All Unassigned** is selected, then the website will respond to requests for all IP addresses on the port and for the host name. The website will not respond to requests if another website on the server has a binding on the same port but with a specific IP address.

Port - Port

`string`. Required when `IISDeploymentType = randomDeployment`. Default value: `80`.

Specifies the port where the Hypertext Transfer Protocol Stack (HTTP.sys) will monitor the website requests.

ServerNameIndication - Server Name Indication required

`boolean`. Optional. Use when `IISDeploymentType = randomDeployment`. Default value: `false`.

Specifies the option to set the Server Name Indication (SNI) for the website.

SNI extends the SSL and TLS protocols to indicate the host name that the clients are attempting to connect to. It allows multiple secure websites with different certificates to use the same IP address.

HostNameWithOutSNI - Host name

`string`. Optional. Use when `IISDeploymentType = randomDeployment`.

Specifies a host name (or domain name) for the website.

If a host name is specified, the clients must use the host name instead of the IP address to access the website.

HostNameWithHttp - Host name

`string`. Optional. Use when `IISDeploymentType = randomDeployment`.

Specifies a host name (or domain name) for the website.
If a host name is specified, the clients must use the host name instead of the IP address to access the website.

HostNameWithSNI - Host name

`string`. Required when `IISDeploymentType = randomDeployment`.

Specifies a host name (or domain name) for the website.
If a host name is specified, the clients must use the host name instead of the IP address to access the website.

SSLCertThumbPrint - SSL certificate thumbprint

`string`. Required when `IISDeploymentType = randomDeployment`.

Specifies the thumb-print of the Secure Socket Layer certificate that the website uses for the HTTPS communication. The thumb-print is a 40 character long hexadecimal string. The SSL certificate should already be installed on the computer in the Local Computer Personal store.

Bindings - Add bindings

`string`. Required when `IISDeploymentType = IISWebsite && ActionIISWebsite = CreateOrUpdateWebsite && AddBinding = true`.

Click on the extension button `...` to add bindings for the website.

CreateOrUpdateAppPoolForWebsite - Create or update app pool

`boolean`. Optional. Use when `ActionIISWebsite = CreateOrUpdateWebsite`. Default value: `false`.

Specifies the option to create or update an application pool. If checked, the website will be created in the specified application pool.

ConfigureAuthenticationForWebsite - Configure authentication

`boolean`. Optional. Use when `ActionIISWebsite = CreateOrUpdateWebsite`. Default value: `false`.

Specifies the option to configure authentication for the website.

AppPoolNameForWebsite - Name

`string`. Required when `IISDeploymentType = IISWebsite && ActionIISWebsite = CreateOrUpdateWebsite && CreateOrUpdateAppPoolForWebsite = true`.

Specifies the name of the IIS application pool to create or update.

DotNetVersionForWebsite - .NET version

`string`. Required when `IISDeploymentType = IISWebsite && ActionIISWebsite = CreateOrUpdateWebsite && CreateOrUpdateAppPoolForWebsite = true`. Allowed values: `v4.0`, `v2.0`, `No Managed Code`. Default value: `v4.0`.

Specifies the version of the .NET Framework that is loaded by the application pool. If the applications assigned to this application pool do not contain managed code, select the **No Managed Code** option from the list.

PipeLineModeForWebsite - Managed pipeline mode

`string`. Required when `IISDeploymentType = IISWebsite && ActionIISWebsite = CreateOrUpdateWebsite && CreateOrUpdateAppPoolForWebsite = true`. Allowed values: `Integrated`, `Classic`. Default value: `Integrated`.

Specifies the managed pipeline mode to determine how IIS processes requests for managed content. Use classic mode only when the applications in the application pool cannot run in integrated mode.

AppPoolIdentityForWebsite - Identity

`string`. Required when `IISDeploymentType = IISWebsite && ActionIISWebsite = CreateOrUpdateWebsite && CreateOrUpdateAppPoolForWebsite = true`. Allowed values: `ApplicationPoolIdentity` (Application Pool Identity), `LocalService` (Local Service), `LocalSystem` (Local System), `NetworkService` (Network Service), `SpecificUser` (Custom Account). Default value: `ApplicationPoolIdentity`.

Configures the account under which an application pool's worker process runs. Specifies one of the predefined security accounts or configures a custom account.

AppPoolUsernameForWebsite - Username

`string`. Required when `AppPoolIdentityForWebsite = SpecificUser && IISDeploymentType = IISWebsite && ActionIISWebsite = CreateOrUpdateWebsite && CreateOrUpdateAppPoolForWebsite = true`.

Specifies the username of the custom account that you want to use.

AppPoolPasswordForWebsite - Password

```
string. Optional. Use when AppPoolIdentityForWebsite = SpecificUser &&  
IISDeploymentType = IISWebsite && ActionIISWebsite = CreateOrUpdateWebsite &&  
CreateOrUpdateAppPoolForWebsite = true.
```

Specifies the password for the custom account.

The best practice is to create a variable in the build or release pipeline, mark it as `Secret` to secure it, and then provide it when using this input, like `$(userCredentials)`.

Note: Special characters in the password are interpreted per [command-line arguments](#).

AnonymousAuthenticationForWebsite - Anonymous authentication

```
boolean. Optional. Use when IISDeploymentType = IISWebsite && ActionIISWebsite =  
CreateOrUpdateWebsite && ConfigureAuthenticationForWebsite = true. Default value:  
false.
```

Specifies the option to enable anonymous authentication for a website.

BasicAuthenticationForWebsite - Basic authentication

```
boolean. Optional. Use when IISDeploymentType = IISWebsite && ActionIISWebsite =  
CreateOrUpdateWebsite && ConfigureAuthenticationForWebsite = true. Default value:  
false.
```

Specifies the option to enable basic authentication for a website.

WindowsAuthenticationForWebsite - Windows authentication

```
boolean. Optional. Use when IISDeploymentType = IISWebsite && ActionIISWebsite =  
CreateOrUpdateWebsite && ConfigureAuthenticationForWebsite = true. Default value:  
true.
```

Specifies the option to enable windows authentication for a website.

ParentWebsiteNameForVD - Parent website name

```
string. Required when IISDeploymentType = IISVirtualDirectory.
```

Specifies the name of the parent website of the virtual directory.

VirtualPathForVD - Virtual path

`string`. Required when `IISDeploymentType = IISVirtualDirectory`.

Specifies the virtual path of the virtual directory.

For example, to create a virtual directory `Site/Application/VDir`, enter `/Application/Vdir`. The parent website and application should already exist.

PhysicalPathForVD - Physical path

`string`. Required when `IISDeploymentType = IISVirtualDirectory`. Default value: `%SystemDrive%\inetpub\wwwroot`.

Specifies the physical path where the virtual directory's content is stored. The content can reside on the local computer, in a remote directory, or on a network share, like `C:\Fabrikam` or `\ContentShare\Fabrikam`.

VDPhysicalPathAuth - Physical path authentication

`string`. Optional. Use when `IISDeploymentType = IISVirtualDirectory`. Allowed values: `VDUserPassThrough` (Application User (Pass-through)), `VDWindowsAuth` (Windows Authentication). Default value: `VDUserPassThrough`.

Specifies the authentication mechanism that is used to access the physical path of the virtual directory.

VDAuthUserName - Username

`string`. Required when `VDPhysicalPathAuth = VDWindowsAuth && IISDeploymentType = IISVirtualDirectory`.

Specifies the user name that is used to access the virtual directory's physical path.

VDAuthUserPassword - Password

`string`. Optional. Use when `VDPhysicalPathAuth = VDWindowsAuth && IISDeploymentType = IISVirtualDirectory`.

Specifies the user's password that is used to access the virtual directory's physical path. The best practice is to create a variable in the build or release pipeline, mark it as `Secret` to secure it, and then provide it when using this input, like `$(userCredentials)`.

Note: Special characters in the password are interpreted per [command-line arguments](#).

ParentWebsiteNameForApplication - Parent website name

`string`. Required when `IISDeploymentType = IISWebApplication`.

Specifies the name of the parent website under which the application will be created or updated.

VirtualPathForApplication - Virtual path

`string`. Required when `IISDeploymentType = IISWebApplication`.

Specifies the virtual path of the application.

For example, to create an application `Site/Application`, enter `/Application`. The parent website should already exist.

PhysicalPathForApplication - Physical path

`string`. Required when `IISDeploymentType = IISWebApplication`. Default value:
`%SystemDrive%\inetpub\wwwroot`.

Specifies the physical path where the application's content is stored. The content can reside on the local computer, in a remote directory, or on a network share, like `C:\Fabrikam` or `\ContentShare\Fabrikam`.

ApplicationPhysicalPathAuth - Physical path authentication

`string`. Optional. Use when `IISDeploymentType = IISWebApplication`. Allowed values: `ApplicationUserPassThrough` (Application User (Pass-through)), `ApplicationWindowsAuth` (Windows Authentication). Default value: `ApplicationUserPassThrough`.

Specifies the authentication mechanism that is used to access the physical path of the application.

ApplicationAuthUserName - Username

`string`. Required when `ApplicationPhysicalPathAuth = ApplicationWindowsAuth && IISDeploymentType = IISWebApplication`.

Specifies the user name that is used to access the application's physical path.

ApplicationAuthUserPassword - Password

`string`. Optional. Use when `ApplicationPhysicalPathAuth = ApplicationWindowsAuth &&`

```
IISDeploymentType = IISWebApplication.
```

Specifies the user's password that is used to access the application's physical path. The best practice is to create a variable in the build or release pipeline, mark it as `Secret` to secure it, and then provide it when using this input, like `$(userCredentials)`.

Note: Special characters in the password are interpreted per [command-line arguments](#).

CreateOrUpdateAppPoolForApplication - Create or update app pool

`boolean`. Optional. Use when `IISDeploymentType = IISWebApplication`. Default value: `false`.

Specifies the option to create or update an application pool. If checked, the application will be created in the specified application pool.

AppPoolNameForApplication - Name

`string`. Required when `IISDeploymentType = IISWebApplication && CreateOrUpdateAppPoolForApplication = true`.

Specifies the name of the IIS application pool to create or update.

DotNetVersionForApplication - .NET version

`string`. Required when `IISDeploymentType = IISWebApplication && CreateOrUpdateAppPoolForApplication = true`. Allowed values: `v4.0`, `v2.0`, `No Managed Code`. Default value: `v4.0`.

Specifies the version of the .NET Framework that is loaded by the application pool. If the applications assigned to this application pool do not contain managed code, select the **No Managed Code** option from the list.

PipeLineModeForApplication - Managed pipeline mode

`string`. Required when `IISDeploymentType = IISWebApplication && CreateOrUpdateAppPoolForApplication = true`. Allowed values: `Integrated`, `Classic`. Default value: `Integrated`.

Specifies the managed pipeline mode to determine how IIS processes requests for managed content. Use classic mode only when the applications in the application pool cannot run in the integrated mode.

AppPoolIdentityForApplication - Identity

`string`. Required when `IISDeploymentType = IISWebApplication && CreateOrUpdateAppPoolForApplication = true`. Allowed values: `ApplicationPoolIdentity` (Application Pool Identity), `LocalService` (Local Service), `LocalSystem` (Local System), `NetworkService` (Network Service), `SpecificUser` (Custom Account). Default value: `ApplicationPoolIdentity`.

Configures the account under which an application pool's worker process runs. Specifies one of the predefined security accounts or configures a custom account.

AppPoolUsernameForApplication - Username

`string`. Required when `AppPoolIdentityForApplication = SpecificUser && IISDeploymentType = IISWebApplication && CreateOrUpdateAppPoolForApplication = true`.

Specifies the username of the custom account that you want to use.

AppPoolPasswordForApplication - Password

`string`. Optional. Use when `AppPoolIdentityForApplication = SpecificUser && IISDeploymentType = IISWebApplication && CreateOrUpdateAppPoolForApplication = true`.

Specifies the password for the custom account.

The best practice is to create a variable in the build or release pipeline, mark it as `Secret` to secure it, and then provide it when using this input, like `$(userCredentials)`.

Note: Special characters in the password are interpreted per [command-line arguments](#).

AppPoolName - Name

`string`. Required when `ActionIISApplicationPool = CreateOrUpdateAppPool`.

Specifies the name of the IIS application pool to create or update.

DotNetVersion - .NET version

`string`. Required when `ActionIISApplicationPool = CreateOrUpdateAppPool`. Allowed values: `v4.0`, `v2.0`, `No Managed Code`. Default value: `v4.0`.

Specifies the version of the .NET Framework that is loaded by the application pool. If the applications assigned to this application pool do not contain managed code, select the

No Managed Code option from the list.

PipeLineMode - Managed pipeline mode

`string`. Required when `ActionIISApplicationPool = CreateOrUpdateAppPool`. Allowed values: `Integrated`, `Classic`. Default value: `Integrated`.

Specifies the managed pipeline mode to determine how IIS processes requests for managed content. Use classic mode only when the applications in the application pool cannot run in the integrated mode.

AppPoolIdentity - Identity

`string`. Required when `ActionIISApplicationPool = CreateOrUpdateAppPool`. Allowed values: `ApplicationPoolIdentity` (Application Pool Identity), `LocalService` (Local Service), `LocalSystem` (Local System), `NetworkService` (Network Service), `SpecificUser` (Custom Account). Default value: `ApplicationPoolIdentity`.

Configures the account under which an application pool's worker process runs. Specifies one of the predefined security accounts or configures a custom account.

AppPoolUsername - Username

`string`. Required when `AppPoolIdentity = SpecificUser && ActionIISApplicationPool = CreateOrUpdateAppPool`.

Specifies the username of the custom account that you want to use.

AppPoolPassword - Password

`string`. Optional. Use when `AppPoolIdentity = SpecificUser && ActionIISApplicationPool = CreateOrUpdateAppPool`.

Specifies the password for the custom account.

The best practice is to create a variable in the build or release pipeline, mark it as `Secret` to secure it, and then provide it when using this input, like `$(userCredentials)`.

Note: Special characters in the password are interpreted per [command-line arguments](#).

StartStopRecycleAppName - Application pool name

`string`. Required when `ActionIISApplicationPool = StartAppPool ||`

```
ActionIISApplicationPool = StopAppPool || ActionIISApplicationPool =
RecycleAppPool.
```

Specifies the name of the IIS application pool.

AppCmdCommands - Additional appcmd.exe commands

```
string.
```

Specifies additional `AppCmd.exe` commands. For more than one command, use a line separator.

For example:

```
list apppools
list sites
recycle apppool /apppool.name:ExampleAppPoolName
```

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to create or update a website, web app, virtual directory, or application pool.

Requirements

Requirement	Description
Pipeline types	Classic release
Runs on	Agent, DeploymentGroup
Demands	None

Requirement	Description
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.111.0 or greater
Task category	Deploy

PublishSymbols@2 - Index sources and publish symbols v2 task

Article • 09/26/2023

Use this task to index your source code and publish your symbols to a file share or Azure Artifacts symbol server.

Indexing your source code allows you to use your symbol files to debug your application on a machine other than the one you used to build your application. For example, you can debug an application built by a build agent from a dev machine that does not have the source code.

Symbol servers enable your debugger to automatically retrieve the correct symbol files without knowing product names, build numbers, or package names.

Syntax

YAML

```
# Index sources and publish symbols v2
# Index your source code and publish symbols to a file share or Azure
Artifacts symbol server.
- task: PublishSymbols@2
  inputs:
    #SymbolsFolder: '$(Build.SourcesDirectory)' # string. Path to symbols
    #folder. Default: $(Build.SourcesDirectory).
    SearchPattern: '**/bin/**/*.pdb' # string. Required. Search pattern.
    #Default: **/bin/**/*.pdb.
    #IndexSources: true # boolean. Index sources. Default: true.
    #PublishSymbols: true # boolean. Publish symbols. Default: true.
    #SymbolServerType: # 'TeamServices' | 'FileShare'. Required when
    PublishSymbols = true. Symbol server type.
    #SymbolsPath: # string. Optional. Use when PublishSymbols = true &&
    #SymbolServerType = FileShare. Path to publish symbols.
    #CompressSymbols: false # boolean. Optional. Use when SymbolServerType =
    #FileShare. Compress symbols. Default: false.
    #SymbolExpirationInDays: '36530' # string. Optional. Use when
    PublishSymbols = true && SymbolServerType = TeamServices. Symbol Expiration
    (in days). Default: 36530.
    # Advanced
    #IndexableFileFormats: 'Default' # 'Default' | 'Pdb' | 'SourceMap' |
    #All'. Optional. Use when PublishSymbols = true && SymbolServerType =
    TeamServices. Symbol file formats to publish. Default: Default.
    #DetailedLog: true # boolean. Verbose logging. Default: true.
    #TreatNotIndexedAsWarning: false # boolean. Warn if not indexed.
    #Default: false.
    #UseNetCoreClientTool: false # boolean. Use NetCore client tool.
```

```
Default: false.  
#SymbolsMaximumWaitTime: # string. Max wait time (min).  
#SymbolsProduct: # string. Product.  
#SymbolsVersion: # string. Version.  
#SymbolsArtifactName: 'Symbols_${BuildConfiguration}' # string. Artifact  
name. Default: Symbols_${BuildConfiguration}.
```

Inputs

SymbolsFolder - Path to symbols folder

`string`. Default value: `$(Build.SourcesDirectory)`.

Specifies the path to the folder that is searched for with symbol files. The default is `$(Build.SourcesDirectory)`. Otherwise, specify a rooted path, such as `$(Build.BinariesDirectory)/MyProject`.

Note

UNC paths aren't supported if you select the Azure Artifacts symbol server as the server type.

SearchPattern - Search pattern

`string`. Required. Default value: `**/bin/**/*.pdb`.

Specifies the pattern used to discover the PDB files to publish. See [File matching patterns reference](#) for more information.

IndexSources - Index sources

`boolean`. Default value: `true`.

Specifies whether to inject source server information into the PDB files. This option is only supported on Windows agents.

PublishSymbols - Publish symbols

`boolean`. Default value: `true`.

Specifies whether to publish the symbol files.

SymbolServerType - Symbol server type

`string`. Required when `PublishSymbols = true`. Allowed values: `TeamServices` (Symbol Server in this organization/collection (requires Azure Artifacts)), `FileShare` (File share).

Specifies where to publish symbols. Symbols published to the Azure Artifacts symbol server are accessible by any user with access to the organization/collection. Azure DevOps Server only supports the `File share` option. See instructions to [Publish symbols for debugging](#) to use Symbol Server in Azure Artifacts.

SymbolsPath - Path to publish symbols

`string`. Optional. Use when `PublishSymbols = true && SymbolServerType = FileShare`.

Specifies the file share that hosts your symbols. This value will be used in the call to `symstore.exe add` as the `/s` parameter. To prepare your SymStore symbol store:

1. Set up a folder on a file-sharing server to store the symbols. For example, set up `\fabrikam-share\symbols`.
2. Grant full control permission to the [build agent service account](#).

If you leave this argument blank, your symbols will be source indexed but not published. You can also store your symbols with your drops. [See Publish Build Artifacts](#).

CompressSymbols - Compress symbols

`boolean`. Optional. Use when `SymbolServerType = FileShare`. Default value: `false`.

Compresses symbols when publishing to file share.

SymbolExpirationInDays - Symbol Expiration (in days)

`string`. Optional. Use when `PublishSymbols = true && SymbolServerType = TeamServices`. Default value: `36530`.

Specifies the number of days that symbols should be retained.

IndexableFileFormats - Symbol file formats to publish

`string`. Optional. Use when `PublishSymbols = true && SymbolServerType = TeamServices`. Allowed values: `Default` (The Default set of symbols to upload), `Pdb` (Only Pdb based symbols Windows pdb's and managed Portable pdb's.), `SourceMap` (Only JavaScript based SourceMap symbols (*.js.map)), `All` (All supported symbol formats). Default value: `Default`.

Specifies which debug formats to publish to the symbol server.

DetailedLog - Verbose logging

`boolean`. Default value: `true`.

Specifies verbose logging.

TreatNotIndexedAsWarning - Warn if not indexed

`boolean`. Default value: `false`.

Specifies whether to warn if sources are not indexed for a PDB file. Otherwise, the messages are logged as normal output.

UseNetCoreClientTool - Use NetCore client tool

`boolean`. Default value: `false`.

Specifies whether to use a version of the symbol upload tool that supports DWARF and ELF files. This option only matters on Windows agents. On non-Windows agents, the version of the symbol upload tool that supports DWARF and ELF files will always be used.

SymbolsMaximumWaitTime - Max wait time (min)

`string`.

Specifies the number of minutes to wait before failing this task.

SymbolsProduct - Product

`string`.

Specifies the product parameter to `symstore.exe`. The default is `$(Build.DefinitionName)`.

SymbolsVersion - Version

`string`.

Specifies the version parameter to `symstore.exe`. The default is `$(Build.BuildNumber)`.

`SymbolsArtifactName` - Artifact name

`string`. Default value: `Symbols_$(BuildConfiguration)`.

Specifies the artifact name to use for the symbols artifact. This should only be used with the FileShare symbol server type. The default is `Symbols_$(BuildConfiguration)`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to index your source code and publish your symbols to a file share or Azure Artifacts symbol server.

Indexing your source code allows you to use your symbol files to debug your application on a machine other than the one you used to build your application. For example, you can debug an application built by a build agent from a dev machine that does not have the source code.

Symbol servers enables your debugger to automatically retrieve the correct symbol files without knowing product names, build numbers, or package names.

ⓘ Important

To delete symbols that were published using the *Index Sources & Publish Symbols* task, you must first delete the build that generated those symbols. This can be accomplished by using [retention policies](#) or by manually [deleting the run](#).

How does indexing work?

By choosing to index the sources, an extra section will be injected into the PDB files. PDB files normally contain references to the local source file paths only E.g:
`C:\BuildAgent_work\1\src\MyApp\Program.cs`. The extra section injected into the PDB file

contains mapping instructions for debuggers. The mapping information indicates how to retrieve the server item corresponding to each local path.

The Visual Studio debugger will use the mapping information to retrieve the source file from the server. An actual command to retrieve the source file is included in the mapping information. Example:

command

```
tf.exe git view /collection:http://SERVER:8080/tfs/DefaultCollection  
/teamproject:"93fc2e4d-0f0f-4e40-9825-01326191395d" /repository:"647ed0e6-  
43d2-4e3d-b8bf-2885476e9c44"  
/commitId:3a9910862e22f442cd56ff280b43dd544d1ee8c9 /path:"/MyApp/Program.cs"  
/output:"C:\Users\username\AppData\Local\SOURCE~1\TFS_COMMIT\3a991086\MyApp\  
Program.cs" /applyfilters
```

Can I use source indexing on a portable PDB created from a .NET Core assembly?

No, but you can use [Source Link](#) instead.

How long are Symbols retained?

Symbols are associated with the build that published to Azure Pipelines they are associated with a build. When the build is deleted either manually or using retention policies, the symbols are also deleted. If you want to retain the symbols indefinitely, mark the build as **Retain Indefinitely**.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater

Requirement	Description
Task category	Build

See also

- [Publish symbols for debugging](#)
- [Debug with Visual Studio](#)
- [Debug with WinDbg](#)

PublishSymbols@1 - Index sources and publish symbols v1 task

Article • 09/26/2023

Use this task to index your source code and publish your symbols to a file share or Azure Artifacts symbol server.

Indexing your source code allows you to use your symbol files to debug your application on a machine other than the one you used to build your application. For example, you can debug an application built by a build agent from a dev machine that does not have the source code.

Symbol servers enable your debugger to automatically retrieve the correct symbol files without knowing product names, build numbers, or package names.

Syntax

YAML

```
# Index sources and publish symbols v1
# Index your source code and publish symbols to a file share.
- task: PublishSymbols@1
  inputs:
    #SymbolsPath: # string. Path to publish symbols.
    SearchPattern: '**/bin/**/*.pdb' # string. Required. Search pattern.
  Default: **/bin/**/*.pdb.
    #SymbolsFolder: # string. Path to symbols folder.
  # Advanced
    #SkipIndexing: false # boolean. Skip indexing. Default: false.
    #TreatNotIndexedAsWarning: false # boolean. Warn if not indexed.
  Default: false.
    #SymbolsMaximumWaitTime: # string. Max wait time (min).
    #SymbolsProduct: # string. Product.
    #SymbolsVersion: # string. Version.
    #SymbolsArtifactName: 'Symbols_${BuildConfiguration}' # string. Artifact name. Default: Symbols_${BuildConfiguration}.
```

Inputs

`SymbolsPath` - Path to publish symbols

`string`.

Specifies the path to the symbol store share. If this value is not set, source indexing will occur, but symbols will not be published.

SearchPattern - Search pattern

`string`. Required. Default value: `**/bin/**/*.pdb`.

Specifies the pattern used to discover the PDB files to publish.

SymbolsFolder - Path to symbols folder

`string`.

Specifies the path to the folder that is searched for symbol files. The default is `$(Build.SourcesDirectory)`. Otherwise, specify a rooted path, for example: `$(Build.BinariesDirectory)/MyProject`.

SkipIndexing - Skip indexing

`boolean`. Default value: `false`.

Specifies whether to skip injecting source server information into the PDB files.

TreatNotIndexedAsWarning - Warn if not indexed

`boolean`. Default value: `false`.

Specifies whether to warn if sources are not indexed for a PDB file. Otherwise, the messages are logged as normal output.

SymbolsMaximumWaitTime - Max wait time (min)

`string`.

The number of minutes to wait before failing the step.

SymbolsProduct - Product

`string`.

Specifies the product parameter to `symstore.exe`. The default is `$(Build.DefinitionName)`.

SymbolsVersion - Version`string`.

Specifies the version parameter to `symstore.exe`. The default is `$(Build.BuildNumber)`.

SymbolsArtifactName - Artifact name`string`. Default value: `Symbols_{BuildConfiguration}`.

Specifies the artifact name to use for the symbols artifact. The default is `Symbols_{BuildConfiguration}`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.95.0 or greater
Task category	Build

InstallAppleCertificate@2 - Install Apple certificate v2 task

Article • 09/26/2023

Use this task to install the Apple certificate that is required to build on a macOS agent. You can use this task to install an Apple certificate that is stored as a [secure file](#) on the server.

Syntax

YAML

```
# Install Apple certificate v2
# Install an Apple certificate required to build on a macOS agent machine.
- task: InstallAppleCertificate@2
  inputs:
    certSecureFile: # string. Required. Certificate (P12).
    #certPwd: # string. Certificate (P12) password.
    # Advanced
    keychain: 'temp' # 'default' | 'temp' | 'custom'. Required. Keychain.
    Default: temp.
    #keychainPassword: # string. Required when keychain = custom || keychain
    = default. Keychain password.
    #customKeychainPath: # string. Required when keychain = custom. Custom
    keychain path.
    #deleteCert: # boolean. Optional. Use when keychain = custom || keychain
    = default. Delete certificate from keychain.
    #deleteCustomKeychain: # boolean. Optional. Use when keychain = custom.
    Delete custom keychain.
    #signingIdentity: # string. Certificate signing identity.
    #setUpPartitionIdACLForPrivateKey: # boolean. Set up partition_id ACL
    for the imported private key.
    #opensslPkcsArgs: # string. OpenSSL arguments for PKCS12.
```

Inputs

`certSecureFile` - Certificate (P12)

`string`. Required.

Specifies the certificate (.p12) that was uploaded to [Secure Files](#) to install on the macOS agent.

certPwd - Certificate (P12) password`string`.

Specifies the password to the Apple certificate (.p12). Use a new build variable with its lock enabled on the `Variables` tab to encrypt this value.

keychain - Keychain

`string`. Required. Allowed values: `default` (Default Keychain), `temp` (Temporary Keychain), `custom` (Custom Keychain). Default value: `temp`.

Specifies the keychain in which to install the Apple certificate. For Microsoft-hosted builds, use `Temporary Keychain`. A temporary keychain will always be deleted after the build or release is complete.

keychainPassword - Keychain password

`string`. Required when `keychain = custom || keychain = default`.

Specifies the password to unlock the keychain. Use a new build variable with its lock enabled on the `Variables` tab to encrypt this value. A password is generated for the temporary keychain if not specified.

customKeychainPath - Custom keychain path

`string`. Required when `keychain = custom`.

Specifies the full path to a custom keychain file. The keychain will be created if it does not already exist.

deleteCert - Delete certificate from keychain

`boolean`. Optional. Use when `keychain = custom || keychain = default`.

Specifies the certificate to delete from the keychain after the build or release is complete.

deleteCustomKeychain - Delete custom keychain

`boolean`. Optional. Use when `keychain = custom`.

Specifies the custom keychain to delete from the agent after the build or release is complete.

`signingIdentity` - Certificate signing identity

`string`.

Specifies the `Common Name` of the subject in the signing certificate. Will attempt to parse the `Common Name` if this is left empty.

`setUpPartitionIdACLForPrivateKey` - Set up partition_id ACL for the imported private key

`boolean`.

If `true`, sets the `partition_id` ACL for the imported private key so that `codesign` won't prompt to use the key for signing. This isn't necessary for temporary keychains on MacOS High Sierra. Learn more about [Open Radar ↗](#).

`opensslPkcsArgs` - OpenSSL arguments for PKCS12

`string`.

Arguments for extraction certificate information using openssl.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

`signingIdentity`

Specifies the resolved `Common Name` of the subject in the signing certificate. Either supplied as an input or parsed from the P12 certificate file.

`keychainPath`

Specifies the path for the keychain file with the certificate.

Remarks

Use this task to install an Apple certificate that is required to build on a macOS agent. You can use this task to install an Apple certificate that is stored as a [secure file](#) on the server.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: xcode
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : signingIdentity, keychainPassword, keychainPath, APPLE_CERTIFICATE_SIGNING_IDENTITY, APPLE_CERTIFICATE_KEYCHAIN
Agent version	2.182.1 or greater
Task category	Utility

InstallAppleCertificate@1 - Install Apple Certificate v1 task

Article • 09/26/2023

Use this task to install the Apple certificate that is required to build on a macOS agent. You can use this task to install an Apple certificate that is stored as a [secure file](#) on the server.

Syntax

YAML

```
# Install Apple Certificate v1
# Install an Apple certificate required to build on a macOS agent.
- task: InstallAppleCertificate@1
  inputs:
    certSecureFile: # string. Required. Certificate (P12).
    #certPwd: # string. Certificate (P12) password.
    # Advanced
    keychain: 'temp' # 'default' | 'temp' | 'custom'. Required. Keychain.
    Default: temp.
    #keychainPassword: # string. Keychain password.
    #customKeychainPath: # string. Required when keychain = custom. Custom keychain path.
    #deleteCert: # boolean. Optional. Use when keychain = custom || keychain = default. Delete certificate from keychain.
    #deleteCustomKeychain: # boolean. Optional. Use when keychain = custom. Delete custom keychain.
    #signingIdentity: # string. Certificate signing identity.
```

Inputs

`certSecureFile` - Certificate (P12)

`string`. Required.

Specifies the certificate (.p12) that was uploaded to [Secure Files](#) to install on the macOS agent.

`certPwd` - Certificate (P12) password

`string`.

Specifies the password to the Apple certificate (.p12). Use a new build variable with its lock enabled on the `Variables` tab to encrypt this value.

`keychain` - Keychain

`string`. Required. Allowed values: `default` (Default Keychain), `temp` (Temporary Keychain), `custom` (Custom Keychain). Default value: `temp`.

Specifies the keychain in which to install the Apple certificate. A temporary keychain will always be deleted after the build or release is complete.

`keychainPassword` - Keychain password

`string`.

Specifies the password to unlock the keychain. Use a new build variable with its lock enabled on the `Variables` tab to encrypt this value. A password is generated for the temporary keychain if one is not already specified.

`customKeychainPath` - Custom keychain path

`string`. Required when `keychain = custom`.

Specifies the full path to a custom keychain file. The keychain will be created if it does not already exist.

`deleteCert` - Delete certificate from keychain

`boolean`. Optional. Use when `keychain = custom || keychain = default`.

Specifies the certificate to delete from the keychain after the build or release is complete.

`deleteCustomKeychain` - Delete custom keychain

`boolean`. Optional. Use when `keychain = custom`.

Specifies the custom keychain to delete from the agent after the build or release is complete.

`signingIdentity` - Certificate signing identity

`string`.

Specifies the `Common Name` of the subject in the signing certificate. Will attempt to parse the `Common Name` if this is left empty.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

`signingIdentity`

Specifies the resolved `Common Name` of the subject in the signing certificate. Either supplied as an input or parsed from the P12 certificate file.

`keychainPath`

Specifies the path for the keychain file with the certificate.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: xcode
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.116.0 or greater
Task category	Utility

InstallAppleCertificate@0 - Install Apple Certificate v0 task

Article • 09/26/2023

Use this task to install the Apple certificate that is required to build on a macOS agent. You can use this task to install an Apple certificate that is stored as a [secure file](#) on the server.

Syntax

YAML

```
# Install Apple Certificate v0
# Install an Apple certificate required to build on a macOS agent.
- task: InstallAppleCertificate@0
  inputs:
    certSecureFile: # string. Required. Certificate (P12).
    #certPwd: # string. Certificate (P12) Password.
    # Advanced
    keychain: 'temp' # 'default' | 'temp' | 'custom'. Required. Keychain.
    Default: temp.
    #keychainPassword: # string. Keychain Password.
    #customKeychainPath: # string. Required when keychain = custom. Custom Keychain Path.
    #deleteCert: # boolean. Optional. Use when keychain = custom || keychain = default. Delete Certificate from Keychain.
    #deleteCustomKeychain: # boolean. Optional. Use when keychain = custom. Delete Custom Keychain.
    #signingIdentity: # string. Certificate Signing Identity.
```

Inputs

certSecureFile - Certificate (P12)

`string`. Required.

Specifies the certificate (.p12) that was uploaded to [Secure Files](#) to install on the macOS agent.

certPwd - Certificate (P12) Password

`string`.

Specifies the password to the Apple certificate (.p12). Use a new build variable with its lock enabled on the `Variables` tab to encrypt this value.

`keychain` - Keychain

`string`. Required. Allowed values: `default` (Default Keychain), `temp` (Temporary Keychain), `custom` (Custom Keychain). Default value: `temp`.

Specifies the keychain in which to install the Apple certificate. A temporary keychain will always be deleted after the build or release is complete.

`keychainPassword` - Keychain Password

`string`.

Specifies the password to unlock the keychain. Use a new build variable with its lock enabled on the `Variables` tab to encrypt this value. A password is generated for the temporary keychain if not specified.

`customKeychainPath` - Custom Keychain Path

`string`. Required when `keychain = custom`.

Specifies the full path to a custom keychain file. The keychain will be created if it does not already exist.

`deleteCert` - Delete Certificate from Keychain

`boolean`. Optional. Use when `keychain = custom || keychain = default`.

Specifies the certificate to delete from the keychain after the build or release is complete.

`deleteCustomKeychain` - Delete Custom Keychain

`boolean`. Optional. Use when `keychain = custom`.

Specifies the custom keychain to delete from the agent after the build or release is complete.

`signingIdentity` - Certificate Signing Identity

`string`.

Specifies the `Common Name` of the subject in the signing certificate. Will attempt to parse the `Common Name` if this is left empty.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: xcode
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.116.0 or greater
Task category	Utility

InstallAppleProvisioningProfile@1 - Install Apple provisioning profile v1 task

Article • 09/26/2023

Use this task to install an Apple provisioning profile, which is required in order to build on a macOS agent.

Syntax

YAML

```
# Install Apple provisioning profile v1
# Install an Apple provisioning profile required to build on a macOS agent
machine.
- task: InstallAppleProvisioningProfile@1
  inputs:
    provisioningProfileLocation: 'secureFiles' # 'secureFiles' |
'sourceRepository'. Required. Provisioning profile location. Default:
secureFiles.
    provProfileSecureFile: # string. Required when
provisioningProfileLocation == secureFiles. Provisioning profile.
    #provProfileSourceRepository: # string. Required when
provisioningProfileLocation == sourceRepository. Provisioning profile.
    #removeProfile: true # boolean. Remove profile after build. Default:
true.
```

Inputs

`provisioningProfileLocation` - Provisioning profile location

`string`. Required. Allowed values: `secureFiles` (Secure Files), `sourceRepository` (Source Repository). Default value: `secureFiles`.

Specifies the location of the provisioning profile to install. The provisioning profile can be uploaded to `Secure Files`, or stored in your source repository or a local path on the agent.

`provProfileSecureFile` - Provisioning profile

`string`. Required when `provisioningProfileLocation == secureFiles`.

Specifies the provisioning profile that was uploaded to `Secure Files` to install on the macOS agent.

`provProfileSourceRepository` - Provisioning profile

`string`. Required when `provisioningProfileLocation == sourceRepository`.

Specifies the provisioning profile from the source repository or the local path to a provisioning profile on the macOS agent.

`removeProfile` - Remove profile after build

`boolean`. Default value: `true`.

Specifies that the provisioning profile should be removed from the agent after the build or release is complete.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

`provisioningProfileUuid`

The UUID property for the selected provisioning profile.

`provisioningProfileName`

The Name property for the selected provisioning profile.

Remarks

You can use this task to install provisioning profiles needed to build iOS Apps, Apple WatchKit apps, and App extensions.

You can install an Apple provisioning profile that is:

- Stored as a [secure file](#) on the server.
- Committed to the source repository or copied to a local path on the macOS agent.
You should encrypt the provisioning profiles if you are committing them to the

source repository. The **Decrypt File** task can be used to decrypt the profiles during a build or release.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: xcode
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : provisioningProfileUuid, provisioningProfileName, APPLE_PROV_PROFILE_UUID
Agent version	2.182.1 or greater
Task category	Utility

InstallAppleProvisioningProfile@0 - Install Apple Provisioning Profile v0 task

Article • 09/26/2023

Use this task to install an Apple provisioning profile, which is required in order to build on a macOS agent.

Syntax

YAML

```
# Install Apple Provisioning Profile v0
# Install an Apple provisioning profile required to build on a macOS agent.
- task: InstallAppleProvisioningProfile@0
  inputs:
    provProfileSecureFile: # string. Required. Provisioning Profile.
    #removeProfile: true # boolean. Remove Profile After Build. Default:
    true.
```

Inputs

`provProfileSecureFile` - Provisioning Profile

`string`. Required.

Specifies the provisioning profile that was uploaded to `Secure Files` to install on the macOS agent.

`removeProfile` - Remove Profile After Build

`boolean`. Default value: `true`.

Specifies that the provisioning profile should be removed from the agent after the build or release is complete.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

You can use this task to install provisioning profiles needed to build iOS Apps, Apple WatchKit apps, and App extensions.

You can install an Apple provisioning profile that is:

- Stored as a [secure file](#) on the server.
- Committed to the source repository or copied to a local path on the macOS agent. You should encrypt the provisioning profiles if you are committing them to the source repository. You can use the [Decrypt File](#) task to decrypt the profiles during a build or release.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: xcode
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.116.0 or greater
Task category	Utility

FuncToolsInstaller@0 - Install Azure Func Core Tools v0 task

Article • 09/26/2023

Use this task to install Azure Functions Core Tools.

Syntax

YAML

```
# Install Azure Func Core Tools v0
# Install Azure Func Core Tools.
- task: FuncToolsInstaller@0
  inputs:
    #version: 'latest' # string. Version. Default: latest.
```

Inputs

`version` - Version

`string`. Default value: `latest`.

Specifies the version of Azure Functions Core Tools to install. For example:

`2.7.1575`

`v2.7.1575`

`latest`

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: Func
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Tool

InstallSSHKey@0 - Install SSH key v0 task

Article • 09/26/2023

Use this task in a pipeline to install an SSH key prior to a build or release step.

Syntax

YAML

```
# Install SSH key v0
# Install an SSH key prior to a build or deployment.
- task: InstallSSHKey@0
  inputs:
    knownHostsEntry: # string. Alias: hostName. Required. Known Hosts Entry.
    sshPublicKey: # string. SSH Public Key.
    sshPassphrase: # string. SSH Passphrase.
    sshKeySecureFile: # string. Required. SSH Key.
  # Advanced
  #addEntryToConfig: false # boolean. Add entry to SSH config. Default: false.
  #configHostAlias: # string. Required when addEntryToConfig = true.
  Alias.
  #configHostname: # string. Required when addEntryToConfig = true. Host name.
  #configUser: # string. Optional. Use when addEntryToConfig = true. User.
  #configPort: # string. Optional. Use when addEntryToConfig = true. Port.
```

Inputs

`knownHostsEntry` - Known Hosts Entry

Input alias: `hostName`. `string`. Required.

Specifies the SSH key entry for the `known_hosts` file.

`sshPublicKey` - SSH Public Key

`string`.

Optional. Specifies the contents of the public SSH key.

sshPassphrase - SSH Passphrase`string`.

Optional. Specifies the passphrase for the SSH key, if any.

sshKeySecureFile - SSH Key`string`. Required.

Specifies the SSH key that was uploaded to `Secure Files` to install on the agent.

addEntryToConfig - Add entry to SSH config`boolean`. Default value: `false`.

Optional. Adds an entry related to the key that was installed to the SSH config file. The key file will be available for all subsequent tasks.

configHostAlias - Alias`string`. Required when `addEntryToConfig = true`.

Specifies the name of the SSH config entry.

configHostname - Host name`string`. Required when `addEntryToConfig = true`.

Specifies the host name property of SSH config entry.

configUser - User`string`. Optional. Use when `addEntryToConfig = true`.

Specifies the user name property of the SSH config entry.

configPort - Port`string`. Optional. Use when `addEntryToConfig = true`.

Specifies the port of the SSH config entry.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task in a pipeline to install an SSH key prior to a build or release step.

ⓘ Note

This task required Git Bash for Windows on the agent.

Usage and best practices

If you install an SSH key in the [hosted pools](#), in later steps in your pipeline, you can connect to a remote system in which the matching public key is already in place. For example, you can connect to a Git repository or to a VM in Azure.

We recommend that you don't pass in your public key as plain text to the task configuration. Instead, [set a secret variable](#) in your pipeline for the contents of your `mykey.pub` file. Then, call the variable in your pipeline definition as `$(myPubKey)`. For the secret part of your key, use the [Secure File library](#) in Azure Pipelines.

To create your task, use the following example of a well-configured Install SSH Key task:

YAML

```
steps:
- task: InstallSSHKey@0
  displayName: 'Install an SSH key'
  inputs:
    knownHostsEntry: 'SHA256:1Hyr55tsxGifESBMc0s+2NtutnR/4+LOkVwr0GrIp8U
johndoe@contoso'
    sshPublicKey: '$(myPubKey)'
    sshKeySecureFile: 'id_rsa'
```

ⓘ Note

Your public key should be added to the repository\organization; otherwise, there will be access issues. For GitHub, follow the guide above. For Azure DevOps Services, use [Add the public key to Azure DevOps Services/TFS](#).

Installing of multiple SSH keys in the same pipeline job

When using more than one key in the same pipeline job, the first one is used by default. To be able to use the desired key when establishing an SSH connection, you can use the `Advanced` section of the `InstallSSHKey` task to set the following parameters:

`addEntryToConfig`, `configHostAlias`, `configHostname`, `configUser`, and `configPort`.

These parameters allow you to add a host to the SSH config file (for example, `/root/.ssh/config` for Linux) in order to use it in custom scripts via an alias.

After the build is completed, the task will attempt to restore the original SSH config file. If there was no SSH config file initially, then the host is removed from the agent.

An example of multiple SSH keys installation. The case with several GitHub repos and their own key for each one:

```
yml

pool: <Some Agent Pool>

steps:
- task: InstallSSHKey@0
  inputs:
    knownHostsEntry: $(known_host)
    sshPublicKey: $(first_public_key)
    sshKeySecureFile: $(first_private_key)
    addEntryToConfig: true
    configHostAlias: <first-host-alias>
    configHostname: github.com
    configUser: git
  displayName: Install First Repo SSH Key

- task: InstallSSHKey@0
  inputs:
    knownHostsEntry: $(known_host)
    sshPublicKey: $(second_public_key)
    sshKeySecureFile: $(second_private_key)
    addEntryToConfig: true
    configHostAlias: <second-host-alias>
    configHostname: github.com
    configUser: git
  displayName: Install Second Repo SSH Key

- bash: git clone git@<first-host-alias>:<owner>/<first-repo>.git
```

```
displayName: Clone First Repo  
  
- bash: git clone git@<second-host-alias>:<owner>/<second-repo>.git  
  displayName: Clone Second Repo
```

Related GitHub docs [↗](#).

Examples

Example setup using GitHub

This section describes how to use a private GitHub repository with YAML from within Azure Pipelines.

If you have a repository that you don't want to expose to the open-source community, a common practice is to make the repository private. However, a CI/CD tool like Azure DevOps needs access to the repository if you want to use the tool to manage the repository. To give Azure DevOps access, you might need an SSH key to authenticate access to GitHub.

Here are the steps to use an SSH key to authenticate access to GitHub:

1. Generate a key pair to use to authenticate access from GitHub to Azure DevOps:
 - a. In GitBash, run the following command:

```
Bash  
  
ssh-keygen -t rsa
```

- b. Enter a name for the SSH key pair. In our example, we use **myKey**.

```
Exampleuser MINGW64 ~/ssh  
$ ssh-keygen -t rsa  
Generating public/private rsa key pair.  
Enter file in which to save the key (/c/Users/Exampleuser/.ssh/id_rsa):
```

- c. (Optional) You can enter a passphrase to encrypt your private key. This step is optional. Using a passphrase is more secure than not using one.

```
Exampleuser MINGW64 ~/ssh  
$ ssh-keygen -t rsa  
Generating public/private rsa key pair.  
Enter file in which to save the key (/c/Users/Exampleuser/.ssh/id_rsa): myKey  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:
```

`ssh-keygen` creates the SSH key pairs, and the following success message appears:

```
Exampleuser MINGW64 ~/ssh
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Exampleuser/.ssh/id_rsa): myKey
Enter passphrase (empty for no passphrase): ...
Enter same passphrase again: ...
Your identification has been saved in myKey.
Your public key has been saved in myKey.pub.
The key fingerprint is:
SHA256:TocLjss ...
The key's randomart image is:
+---[RSA 3072]---+
| |
|o
|o
|o
|= |
|o
|o
|o
|.
+---[SHA256]-----+
```

d. In Windows File Explorer, check your newly created key pair:

This PC > Windows (C:) > Users > Exampleuser > .ssh				
	Name	Date modified	Type	Size
	keys	4/16/2020 6:18 PM	File	4 KB
	keys	4/16/2020 6:18 PM	Microsoft Publish...	1 KB

2. Add the public key to the GitHub repository. (The public key ends in ".pub"). To do this, go the following URL in your browser: [https://github.com/\(organization-name\)/\(repository-name\)/settings/keys](https://github.com/(organization-name)/(repository-name)/settings/keys).

a. Select **Add deploy key**.

b. In the **Add new** dialog box, enter a title, and then copy and paste the SSH key:

Deploy keys / Add new

Title

Key

Begins with 'ssh-rsa', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

Allow write access

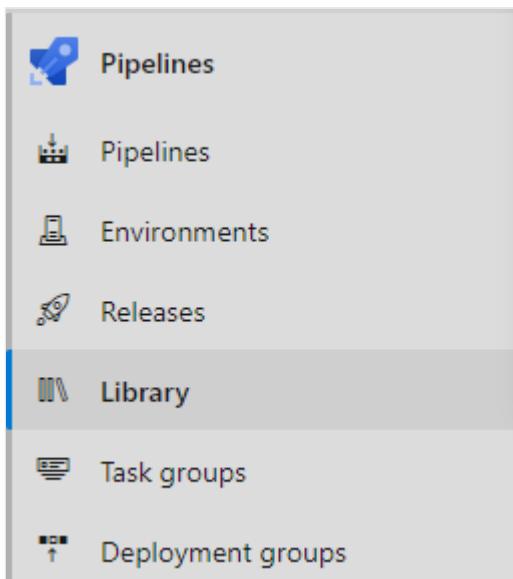
Can this key be used to push to this repository? Deploy keys always have pull access.

Add key

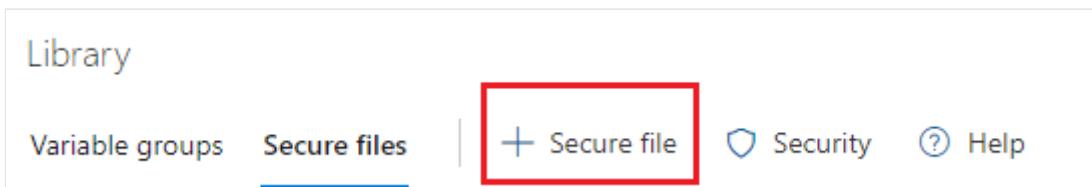
c. Select **Add key**.

3. Upload your private key to Azure DevOps:

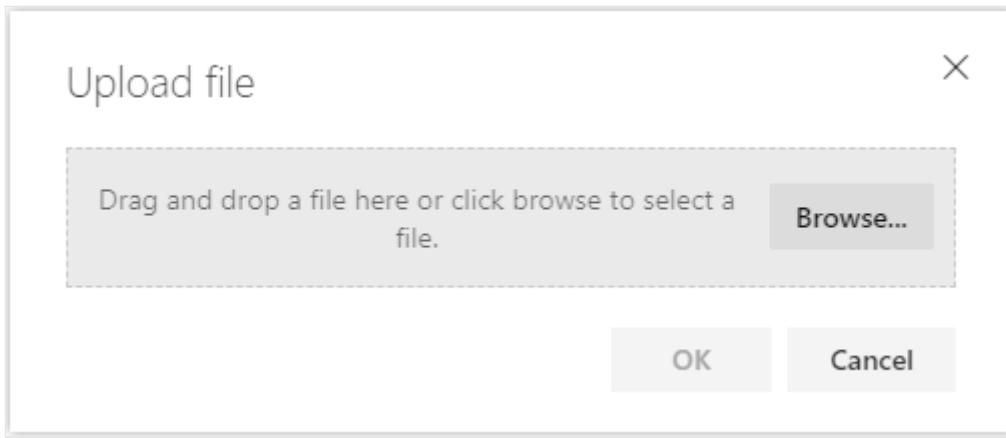
a. In Azure DevOps, in the left menu, select **Pipelines > Library**.



b. Select **Secure files > + Secure file**:



c. Select **Browse**, and then select your private key:



4. Recover your "Known Hosts Entry". In GitBash, enter the following command:

```
Bash  
ssh-keyscan github.com
```

Your "Known Hosts Entry" is the displayed value that doesn't begin with # in the GitBash results:

```
Exampleuser MINGW64 ~/ssh  
$ ssh-keyscan github.com  
# github.com:22 SSH-2.0-babelfd-76c80caa  
github.com ssh-rsa AAAAB3NzaC1yc2EAAAQABIAwAAAQEaQ2A7hRG  
XVal72J+UX2B+2RPW3RcT0e0zQgq1JL3RKrtTJvdsjE3JEAvGq31GHS  
# github.com:22 SSH-2.0-babelfd-76c80caa  
# github.com:22 SSH-2.0-babelfd-76c80caa
```

5. Create a YAML pipeline.

To create a YAML pipeline, in the YAML definition, add the following task:

```
Bash  
  
- task: InstallSSHKey@0  
  inputs:  
    knownHostsEntry: #{Enter your Known Hosts Entry Here}  
    sshPublicKey: #{Enter your Public key Here}  
    sshKeySecureFile: #{Enter the name of your key in "Secure Files"  
Here}
```

The SSH keys are now installed, and you can proceed with the script to connect by using SSH, and not the default HTTPS.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : SSH_AGENT_PID, SSH_AUTH_SOCK, INSTALL_SSH_KEY_CONFIG_LOCATION, INSTALL_SSH_KEY_KNOWN_HOSTS_LOCATION
Agent version	2.182.1 or greater
Task category	Utility

AzureFunction@1 - Invoke Azure Function v1 task

Article • 09/26/2023

Use this task in an [agentless job](#) of a release pipeline to invoke an HTTP triggered function in a function app and parse the response. The function app must be created and hosted in Azure Functions.

Syntax

YAML

```
# Invoke Azure Function v1
# Invoke an Azure Function.
- task: AzureFunction@1
  inputs:
    function: # string. Required. Azure function URL.
    key: # string. Required. Function key.
    method: 'POST' # 'OPTIONS' | 'GET' | 'HEAD' | 'POST' | 'PUT' | 'DELETE'
    | 'TRACE' | 'PATCH'. Required. Method. Default: POST.
    #headers: # string. Headers.
    #queryParameters: # string. Query parameters.
    #body: # string. Optional. Use when method != GET && method != HEAD.
  Body.
    # Advanced
    waitForCompletion: 'false' # 'true' | 'false'. Required. Completion
    event. Default: false.
    #successCriteria: # string. Optional. Use when waitForCompletion =
    false. Success criteria.
```

Inputs

function - Azure function URL

`string`. Required.

The URL of the Azure function to be invoked. Example:

`https://azurefunctionapp.azurewebsites.net/api/HttpTriggerJS1`.

key - Function key

`string`. Required.

The function or the host key used to access and invoke the function. To keep the key secure, use a secret pipeline variable to store the function key. Example:

`$(myFunctionKey)`. `myFunctionKey` is an environment-level secret variable with a value as the secret key.

`method` - Method

`string`. Required. Allowed values: `OPTIONS`, `GET`, `HEAD`, `POST`, `PUT`, `DELETE`, `TRACE`, `PATCH`. Default value: `POST`.

The HTTP method with which the function will be invoked.

`headers` - Headers

```
string. Default value: { \n"Content-Type": "application/json", \n"PlanUrl":  
    "$(system.CollectionUri)", \n"ProjectId": "$(system.TeamProjectId)", \n"HubName":  
    "$(system.HostType)", \n"PlanId": "$(system.PlanId)", \n"JobId": "$(system.JobId)",  
    \n"TimelineId": "$(system.TimelineId)", \n"TaskInstanceId":  
    "$(system.TaskInstanceId)", \n"AuthToken": "$(system.AccessToken)" \n}.
```

The header in JSON format to be attached to the request sent to the function.

`queryParameters` - Query parameters

`string`.

The string query to append to the function URL. Must not start with `?` or `&`.

`body` - Body

`string`. Optional. Use when `method != GET && method != HEAD`.

The request body in JSON format.

`waitForCompletion` - Completion event

`string`. Required. Allowed values: `true` (Callback), `false` (ApiResponse). Default value: `false`.

How the task reports completion.

- `false` - API response - the function returns success and success criteria evaluates to true.

- `true` - **Callback** - the function makes a callback to update the timeline record.

`successCriteria` - Success criteria

`string`. Optional. Use when `waitForCompletion = false`.

The criteria for a successful task. By default, the task returns `200 OK` status when successful.

Example: For response `{"status" : "successful"}`, the expression can be `eq(root['status'], 'successful')`. Learn more about [specifying conditions](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task in an [agentless job](#) of a release pipeline to invoke an HTTP triggered function in a function app that is created and hosted in Azure Functions and parse the response.

Where should a task signal completion when Callback is chosen as the completion event?

To signal completion, the function should POST completion data to the following pipelines REST endpoint.

```
{planUri}/{projectId}/_apis/distributedtask/hubs/{hubName}/plans/{planId}/events?api-version=2.0-preview.1
```

Request Body

```
{ "name": "TaskCompleted", "taskId": "taskInstanceId", "jobId": "jobId",  
"result": "succeeded" }
```

See [this simple cmdline application](#) for specifics. In addition, a C# helper library is available to enable live logging and managing task status for agentless tasks. [Learn more](#)

Why does the task fail within 1 minute when the timeout is longer?

If the function executes for more than 1 minute, use the **Callback** completion event. The API Response completion option is supported for requests that complete within 60 seconds.

Examples

Example of an Azure Function that uses the callback completion mode

```
#r "Newtonsoft.Json"

using System;
using System.Net;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;
using System.Threading;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Primitives;
using Newtonsoft.Json;

public static async Task<IActionResult> Run(HttpContext req, ILogger log)
{
    var url = req.Headers["PlanUrl"];
    var projectId = req.Headers["ProjectId"];
    var hubName = req.Headers["HubName"];
    var planId = req.Headers["PlanId"];
    var jobId = req.Headers["JobId"];
    var timelineId = req.Headers["TimelineId"];
    var taskInstanceId = req.Headers["TaskinstanceId"];
    var authToken = req.Headers["AuthToken"];

    var callbackUrl = $""
{url}/{projectId}/ apis/distributedtask/hubs/{hubName}/plans/{planId}/events
```

```

?api-version=2.0-preview.1";

var successBody = JsonConvert.SerializeObject(new {
    name = "TaskCompleted",
    taskId = taskInstanceId.ToString(),
    jobId = jobId.ToString(),
    result = "succeeded"
});

// the following call does not block
Task.Run(() =>
{
    Thread.Sleep(70000); // simulate long running work
    PostEvent(callbackUrl, successBody, authToken, log);
});

return new OkObjectResult("Long-running job successfully scheduled!");
}

public static void PostEvent(String callbackUrl, String body, String
authToken, ILogger log)
{
    try
    {
        var client = new HttpClient();
        client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", authToken);
        var requestContent = new StringContent(body, Encoding.UTF8,
"application/json");
        var response = client.PostAsync(new Uri(callbackUrl),
requestContent).Result;
        var responseContent = response.Content.ReadAsStringAsync().Result;
        log.LogInformation(response.StatusCode.ToString());
        log.LogInformation(responseContent);
    }
    catch (Exception ex)
    {
        log.LogError(ex.Message);
    }
}
}

```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Server, ServerGate
Demands	None

Requirement	Description
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Utility

See also

- [Automate Azure Functions deployments with Azure Pipelines](#)
- [Agentless job](#)

AzureFunction@0 - Invoke Azure Function v0 task

Article • 09/26/2023

Use this task in an [agentless job](#) of a release pipeline to invoke an HTTP triggered function in a function app and parse the response. The function app must be created and hosted in Azure Functions.

Syntax

YAML

```
# Invoke Azure Function v0
# Invoke Azure function as a part of your process.
- task: AzureFunction@0
  inputs:
    function: # string. Required. Azure function url.
    key: # string. Required. Function key.
    method: 'POST' # 'OPTIONS' | 'GET' | 'HEAD' | 'POST' | 'PUT' | 'DELETE'
    | 'TRACE' | 'PATCH'. Required. Method. Default: POST.
    #headers: # string. Headers.
    #queryParameters: # string. Query parameters.
    #body: '{"JobId": "$(system.jobId)", "PlanId": "$(system.planId)", "TimelineId": "$(system.timelineId)", "ProjectId": "$(system.teamProjectId)", "VstsUrl": "$(system.CollectionUri)", "AuthToken": "$(system.AccessToken)" }' # string. Optional. Use when method != GET && method != HEAD. Body. Default: {"JobId": "$(system.jobId)", "PlanId": "$(system.planId)", "TimelineId": "$(system.timelineId)", "ProjectId": "$(system.teamProjectId)", "VstsUrl": "$(system.CollectionUri)", "AuthToken": "$(system.AccessToken)" }.
    # Completion Options
    waitForCompletion: 'false' # 'true' | 'false'. Required. Complete based on. Default: false.
    #successCriteria: # string. Optional. Use when waitForCompletion = false. Success criteria.
```

Inputs

`function` - Azure function url

`string`. Required.

The URL of the Azure function to be invoked. Example:

`https://azurefunctionapp.azurewebsites.net/api/HttpTriggerJS1`.

key - Function key

`string`. Required.

The function or the host key used to access and invoke the function. To keep the key secure, use a secret pipeline variable to store the function key. Example:

`$(myFunctionKey)`. `myFunctionKey` is an environment-level secret variable with a value as the secret key.

method - Method

`string`. Required. Allowed values: `OPTIONS`, `GET`, `HEAD`, `POST`, `PUT`, `DELETE`, `TRACE`, `PATCH`. Default value: `POST`.

The HTTP method with which the function will be invoked.

headers - Headers

`string`. Default value: `{\n"Content-Type": "application/json"\n}`.

The header in JSON format to be attached to the request sent to the function.

queryParameters - Query parameters

`string`.

The string query to append to the function URL. Must not start with `?` or `&`.

body - Body

`string`. Optional. Use when `method != GET && method != HEAD`. Default value: `{"JobId": "$(system.jobId)", "PlanId": "$(system.planId)", "TimelineId": "$(system.timelineId)", "ProjectId": "$(system.teamProjectId)", "VstsUrl": "$(system.CollectionUri)", "AuthToken": "$(system.AccessToken)"}.`

The request body in JSON format.

waitForCompletion - Complete based on

`string`. Required. Allowed values: `true` (Callback), `false` (ApiResponse). Default value: `false`.

How the task reports completion.

- `false` - API response - the function returns success and success criteria evaluates to true.
- `true` - Callback - the function makes a callback to update the timeline record.

`successCriteria` - Success criteria

`string`. Optional. Use when `waitForCompletion = false`.

The criteria for a successful task. By default, the task returns `200 OK` status when successful.

Example: For response `{"status" : "successful"}`, the expression can be `eq(root['status'], 'successful')`. Learn more about [specifying conditions](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

[AzureFunction@2](#) is a newer version of the Invoke Azure Function task.

Requirements

Requirement	Description
Pipeline types	Classic release
Runs on	Server, ServerGate
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any

Requirement	Description
Settable variables	Any
Agent version	All supported agent versions.
Task category	Utility

See also

- [AzureFunction@2](#) is a newer version of the Invoke Azure Function task.

InvokeRESTAPI@1 - Invoke REST API v1 task

Article • 09/26/2023

Use this task to invoke a REST API as a part of your pipeline.

Syntax

YAML

```
# Invoke REST API v1
# Invoke a REST API as a part of your pipeline.
- task: InvokeRESTAPI@1
  inputs:
    connectionType: 'connectedServiceName' # 'connectedServiceName' |
    'connectedServiceNameARM'. Alias: connectedServiceNameSelector. Required.
    Connection type. Default: connectedServiceName.
    serviceConnection: # string. Alias: connectedServiceName |
    genericService. Required when connectedServiceNameSelector =
    connectedServiceName. Generic service connection.
    #azureServiceConnection: # string. Alias: connectedServiceNameARM |
    azureSubscription. Required when connectedServiceNameSelector =
    connectedServiceNameARM. Azure subscription.
    method: 'POST' # 'OPTIONS' | 'GET' | 'HEAD' | 'POST' | 'PUT' | 'DELETE' |
    'TRACE' | 'PATCH'. Required. Method. Default: POST.
    #headers: # string. Headers.
    #body: # string. Optional. Use when method != GET && method != HEAD.
  Body.
    #urlSuffix: # string. URL suffix and parameters.
  # Advanced
    waitForCompletion: 'false' # 'true' | 'false'. Required. Completion
    event. Default: false.
    #successCriteria: # string. Optional. Use when waitForCompletion =
    false. Success criteria.
```

Inputs

connectionType - Connection type

Input alias: `connectedServiceNameSelector`. `string`. Required. Allowed values: `connectedServiceName` (Generic), `connectedServiceNameARM` (Azure Resource Manager). Default value: `connectedServiceName`.

Specifies the service connection type to use to invoke the REST API. Select **Azure Resource Manager** to invoke an Azure management API or **Generic** for all other APIs.

serviceConnection - Generic service connection

Input alias: `connectedServiceName | genericService`. `string`. Required when
`connectedServiceNameSelector = connectedServiceName`.

Specifies the generic service connection that provides the baseUrl for the call and the authorization to use for the task.

azureServiceConnection - Azure subscription

Input alias: `connectedServiceNameARM | azureSubscription`. `string`. Required when
`connectedServiceNameSelector = connectedServiceNameARM`.

Specifies the Azure Resource Manager subscription to configure and use for invoking Azure management APIs.

method - Method

`string`. Required. Allowed values: `OPTIONS`, `GET`, `HEAD`, `POST`, `PUT`, `DELETE`, `TRACE`, `PATCH`.
Default value: `POST`.

Specifies the HTTP method that invokes the API.

headers - Headers

```
string. Default value: {\n"Content-Type": "application/json", \n"PlanUrl":\n"$(system.CollectionUri)", \n"ProjectId": "$(system.TeamProjectId)", \n"HubName":\n"$(system.HostType)", \n"PlanId": "$(system.PlanId)", \n"JobId": "$(system.JobId)",\n"TimelineId": "$(system.TimelineId)", \n"TaskInstanceId":\n"$(system.TaskInstanceId)", \n"AuthToken": "$(system.AccessToken)"\n}.
```

Defines the header in JSON format. The header is attached with the request sent to the API.

body - Body

`string`. Optional. Use when `method != GET && method != HEAD`.

Specifies the request body for the function call in JSON format.

`urlSuffix` - URL suffix and parameters

`string`.

Specifies the string to append to the baseUrl from the generic service connection while making the HTTP call.

Example: If the service connection URL is `https://...TestProj/_apis/Release/releases` and the URL suffix is `/2/environments/1`, the service connection URL becomes `https://.../TestProj/_apis/Release/releases/2/environments/1`. If the URL suffix is `?definitionId=1&releaseCount=1`, then the service connection URL becomes `https://...TestProj/_apis/Release/releases?definitionId=1&releaseCount=1`.

`waitForCompletion` - Completion event

`string`. Required. Allowed values: `true` (Callback), `false` (ApiResponse). Default value: `false`.

Specifies how the task reports completion. The allowed values are:

- `false` - **API response**: reports completion when the function returns success within 20 seconds, and the success criteria evaluates to true.
- `true` - **Callback**: reports completion when the external service makes a callback to update the timeline record.

`successCriteria` - Success criteria

`string`. Optional. Use when `waitForCompletion = false`.

Specifies the task's criteria for success. The response content does not influence the result if no criteria is defined. By default, the task passes when the call returns `200 OK`.

Example: For response `{"status" : "successful"}`, the expression can be `eq(root['status'], 'successful')`. Learn more about [specifying conditions](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

(!) Note

This task can be used only in an **agentless job**.

Succeeds if the API returns success and the response body parsing is successful, or when the API updates the timeline record with success.

The **Invoke REST API task** does not perform deployment actions directly. Instead, it allows you to invoke any generic HTTP REST API as part of the automated pipeline and, optionally, wait for it to be completed.

The screenshot shows the configuration of the 'Invoke REST API' task within a deployment process named 'Dev'. The task is set to run on an 'Agentless job' and is configured to use a 'Generic' service connection named 'MyGenericConnection'. The method is set to 'POST' and the headers are defined as follows:

```
{  
  "Content-Type": "application/json"  
}
```

The body of the request is defined as follows:

```
{  
  "count": 19  
  "value": [  
    ...  
  ]  
}
```

The task also includes an 'Advanced' section with 'Completion event' set to 'ApiResponse' and 'Success criteria' left empty.

For more information about using this task, see [Approvals and gates overview](#).

What base URLs are used when invoking Azure Management APIs?

Azure management APIs are invoked using *ResourceManagerEndpoint* of the selected environment. For example `https://management.azure.com` is used when the subscription is in an **AzureCloud** environment.

Where should a task signal completion when Callback is chosen as the completion event?

To signal completion, the external service should POST completion data to the following pipelines REST endpoint.

```
{planUri}/{projectId}/_apis/distributedtask/hubs/{hubName}/plans/{planId}/events?api-version=2.0-preview.1

**Request Body**
{
  "name": "TaskCompleted",
  "taskId": "taskInstanceId",
  "jobId": "jobId",
  "result": "succeeded"
}
```

See [this simple cmdline application](#) for specifics.

In addition, a C# helper library is available to enable live logging and managing task status for agentless tasks. [Learn more](#)

Can I use the response body as the input for another task?

No, as this task is an agentless task and uses TFS's internal `HttpRequest`, which doesn't return the content of the HTTP request.

Example

YAML

yml

```

steps:
- task: InvokeRESTAPI@1
  displayName: 'Invoke REST API: GET'
  inputs:
    serviceConnection: 'generic_demo'
    method: GET
    successCriteria: 'eq(root[''count''], ''1425'')
```

In this example, the task succeeds when the response matched our `successCriteria: eq(root["count"], "1425")`.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Server, ServerGate
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Deploy

InvokeRESTAPI@0 - Invoke REST API v0 task

Article • 09/26/2023

Use this task to invoke a REST API as a part of your pipeline.

Syntax

YAML

```
# Invoke REST API v0
# Invoke REST API as a part of your process.
- task: InvokeRESTAPI@0
  inputs:
    serviceConnection: # string. Alias: connectedServiceName. Required.
    Generic endpoint.
    method: 'POST' # 'OPTIONS' | 'GET' | 'HEAD' | 'POST' | 'PUT' | 'DELETE'
    | 'TRACE' | 'PATCH'. Required. Method. Default: POST.
    #headers: # string. Headers.
    #body: '{"JobId": "$(system.jobId)", "PlanId": "$(system.planId)", "TimelineId": "$(system.timelineId)", "ProjectId": "$(system.teamProjectId)", "VstsUrl": "$(system.CollectionUri)", "AuthToken": "$(system.AccessToken)"}' # string. Optional. Use when method != GET && method != HEAD. Body. Default: {"JobId": "$(system.jobId)", "PlanId": "$(system.planId)", "TimelineId": "$(system.timelineId)", "ProjectId": "$(system.teamProjectId)", "VstsUrl": "$(system.CollectionUri)", "AuthToken": "$(system.AccessToken)"}.
    #urlSuffix: # string. Url suffix string.
    # Completion Options
    waitForCompletion: 'false' # 'true' | 'false'. Required. Complete based on. Default: false.
    #successCriteria: # string. Optional. Use when waitForCompletion = false. Success criteria.
```

Inputs

serviceConnection - Generic endpoint

Input alias: `connectedServiceName`. `string`. Required.

Specifies the generic service connection that provides the `baseURL` for the call and the authorization to use for the task.

method - Method

`string`. Required. Allowed values: `OPTIONS`, `GET`, `HEAD`, `POST`, `PUT`, `DELETE`, `TRACE`, `PATCH`.

Default value: `POST`.

Specifies the HTTP method that invokes the API.

headers - Headers

`string`. Default value: `{\n"Content-Type": "application/json"\n}`.

Defines the header in JSON format. The header is attached with the request sent to the API.

body - Body

`string`. Optional. Use when `method != GET && method != HEAD`. Default value: `{"JobId": "$(system.jobId)", "PlanId": "$(system.planId)", "TimelineId": "$(system.timelineId)", "ProjectId": "$(system.teamProjectId)", "VstsUrl": "$(system.CollectionUri)", "AuthToken": "$(system.AccessToken)"}.`

Specifies the request body for the function call in JSON format.

urlSuffix - Url suffix string

`string`.

Specifies the string to append to the `baseUrl` from the generic service connection while making the HTTP call.

Example: If the service connection URL is `https://...TestProj/_apis/Release/releases` and the URL suffix is `/2/environments/1`, the service connection URL becomes `https://.../TestProj/_apis/Release/releases/2/environments/1`. If the URL suffix is `?definitionId=1&releaseCount=1`, then the service connection URL becomes `https://...TestProj/_apis/Release/releases?definitionId=1&releaseCount=1`.

waitForCompletion - Complete based on

`string`. Required. Allowed values: `true` (Callback), `false` (ApiResponse). Default value: `false`.

Specifies how the task reports completion. The allowed values are:

- `false` - **API response**: Reports completion when the function returns success within 20 seconds, and the success criteria evaluates to true.
- `true` - **Callback**: Reports completion when the external service makes a callback to update the timeline record.

`successCriteria` - Success criteria

`string`. Optional. Use when `waitForCompletion = false`.

Specifies the task's criteria for success. The response content does not influence the result if no criteria is defined. By default, the task passes when the call returns `200 OK`.

Example: For response `{"status" : "successful"}`, the expression can be `eq(root['status'], 'successful')`. Learn more about [specifying conditions](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Note

This task can be used only in an [agentless job](#).

Succeeds if the API returns success and the response body parsing is successful, or when the API updates the timeline record with success.

The **Invoke REST API task** does not perform deployment actions directly. Instead, it allows you to invoke any generic HTTP REST API as part of the automated pipeline and, optionally, wait for it to be completed.

The screenshot shows the Azure DevOps pipeline editor. On the left, there's a sidebar with a 'Dev Deployment process' section containing 'Agentless job' and 'Invoke REST API: POST'. The main area is titled 'Invoke REST API' with a 'Version 1.*' dropdown. It includes fields for 'Display name' (set to 'Invoke REST API: POST'), 'Connection type' (set to 'Generic'), a 'Generic service connection' dropdown ('MyGenericConnection'), 'Method' (set to 'POST'), 'Headers' (containing a JSON object with 'Content-Type: application/json'), 'Body' (containing a JSON object with 'count: 19' and 'value: []'), 'URL suffix and parameters' (empty), and an 'Advanced' section with 'Completion event' set to 'ApiResponse'. A 'Control Options' section is also present.

For more information about using this task, see [Approvals and gates overview](#).

What base URLs are used when invoking Azure Management APIs?

Azure management APIs are invoked using *ResourceManagerEndpoint* of the selected environment. For example `https://management.Azure.com` is used when the subscription is in an **AzureCloud** environment.

Where should a task signal completion when Callback is chosen as the completion event?

To signal completion, the external service should POST completion data to the following pipelines REST endpoint.

```
{planUri}/{projectId}/_apis/distributedtask/hubs/{hubName}/plans/{planId}/events?api-version=2.0-preview.1

**Request Body**
{
  "name": "TaskCompleted",
  "taskId": "taskInstanceId",
  "jobId": "jobId",
  "result": "succeeded"
}
```

See this simple cmdline application [↗](#) for specifics.

In addition, a C# helper library is available to enable live logging and managing task status for agentless tasks. [Learn more](#)

Can I use the response body as the input for another task?

No, as this task is an agentless task and uses TFS's internal `HttpRequest`, which doesn't return the content of the HTTP request.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Server, ServerGate
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Deploy

JavaToolInstaller@0 - Java tool installer v0 task

Article • 09/26/2023

Use this task to acquire a specific version of Java from a user-supplied Azure blob or the tool cache and set `JAVA_HOME`.

Syntax

YAML

```
# Java tool installer v0
# Acquire a specific version of Java from a user-supplied Azure blob or the
tool cache and sets JAVA_HOME.
- task: JavaToolInstaller@0
  inputs:
    versionSpec: '8' # string. Required. JDK version. Default: 8.
    jdkArchitectureOption: # 'x64' | 'x86'. Required. JDK architecture.
    jdkSourceOption: # 'AzureStorage' | 'LocalDirectory' | 'PreInstalled'.
Required. JDK source.
    #jdkFile: # string. Required when jdkSourceOption == LocalDirectory. JDK
file.
    #azureResourceManagerEndpoint: # string. Required when jdkSourceOption ==
AzureStorage. Azure subscription.
    #azureStorageAccountName: # string. Required when jdkSourceOption ==
AzureStorage. Storage account name.
    #azureContainerName: # string. Required when jdkSourceOption ==
AzureStorage. Container name.
    #azureCommonVirtualFile: # string. Required when jdkSourceOption ==
AzureStorage. Common virtual path.
    #jdkDestinationDirectory: # string. Required when jdkSourceOption != PreInstalled. Destination directory.
    #azureResourceGroupName: # string. Optional. Use when jdkSourceOption ==
AzureStorage. Resource Group name.
    #cleanDestinationDirectory: true # boolean. Optional. Use when
jdkSourceOption != PreInstalled. Clean destination directory. Default: true.
    #createExtractDirectory: true # boolean. Optional. Use when
jdkSourceOption != PreInstalled. Create directory for extracting. Default:
true.
```

Inputs

`versionSpec` - JDK version

`string`. Required. Default value: `8`.

Specifies the JDK version to make available on the path. Use a whole number version, such as 10.

jdkArchitectureOption - JDK architecture

`string`. Required. Allowed values: `x64`, `x86`.

Specifies the architecture (`x86`, `x64`) of the JDK.

jdkSourceOption - JDK source

`string`. Required. Allowed values: `AzureStorage` (Azure Storage), `LocalDirectory` (Local Directory), `PreInstalled` (Pre-installed).

Specifies the source for the compressed JDK. The source can be Azure blob storage or a local directory on the agent or source repository, or you can use the pre-installed version of Java (available for Microsoft-hosted agents). Please see the example below about how to use the pre-installed version of Java.

jdkFile - JDK file

`string`. Required when `jdkSourceOption == LocalDirectory`.

Specifies the path to the JDK archive file that contains the compressed JDK. The path could be in your source repository or a local path on the agent. The file should be an archive (.zip, .tar.gz, .7z) containing the bin folder on the root level or inside a single directory. MacOS supports .pkg and .dmg files containing only one .pkg file inside.

azureResourceManagerEndpoint - Azure subscription

`string`. Required when `jdkSourceOption == AzureStorage`.

Specifies the Azure Resource Manager subscription for the JDK.

azureStorageAccountName - Storage account name

`string`. Required when `jdkSourceOption == AzureStorage`.

Specifies Azure Classic or Resource Manager storage accounts. Select the storage account name in which the JDK is located.

azureContainerName - Container name

`string`. Required when `jdkSourceOption == AzureStorage`.

Specifies the name of the container in the storage account where the JDK is located.

azureCommonVirtualFile - Common virtual path

`string`. Required when `jdkSourceOption == AzureStorage`.

Specifies the path to the JDK inside the Azure storage container.

jdkDestinationDirectory - Destination directory

`string`. Required when `jdkSourceOption != PreInstalled`.

Specifies the destination directory where the JDK should be extracted. On Linux and Windows, this is used as the destination directory for the JDK installation. On macOS, this directory is used as a temporary folder for extracting .dmg's because macOS doesn't support installing JDK to a specific directory.

azureResourceGroupName - Resource Group name

`string`. Optional. Use when `jdkSourceOption == AzureStorage`.

Resource Group name of the storage account.

cleanDestinationDirectory - Clean destination directory

`boolean`. Optional. Use when `jdkSourceOption != PreInstalled`. Default value: `true`.

Specifies the option to clean the destination directory before JDK is extracted into it.

createExtractDirectory - Create directory for extracting

`boolean`. Optional. Use when `jdkSourceOption != PreInstalled`. Default value: `true`.

By default, the task creates a directory similar to `JAVA_HOME_8_X64_OpenJDK_zip` for extracting JDK. This option disables the creation of that folder and, if set to `false`, JDK is located in the root of `jdkDestinationDirectory` instead.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to acquire a specific version of Java from a user-supplied Azure blob, a location in the source or on the agent, or the tools cache. The task also sets the `JAVA_HOME` environment variable. Use this task to change the version of Java used in Java tasks.

ⓘ Note

To run the **Java Tool Installer** task on macOS, it is required for the user under which the agent is running to have permission to execute the `sudo` command without a password. You can follow the next steps to enable this permission:

1. Run the `sudo visudo` command. The sudoers file opens for editing.
2. Go to the bottom of the file and add the following line: `user ALL=NOPASSWD: /usr/sbin/installer` (Replace user with the actual user alias).
3. Save and close the file.

Examples

Here's an example of getting the archive file from a local directory on Linux. The file should be an archive (.zip, .gz) of the `JAVA_HOME` directory, so it includes the `bin`, `lib`, `include`, `jre`, etc. directories.

YAML

```
- task: JavaToolInstaller@0
  inputs:
    versionSpec: "11"
    jdkArchitectureOption: x64
    jdkSourceOption: LocalDirectory
    jdkFile: "/builds/openjdk-11.0.2_linux-x64_bin.tar.gz"
    jdkDestinationDirectory: "/builds/binaries/externals"
    cleanDestinationDirectory: true
```

Here's an example of downloading the archive file from Azure Storage. The file should be an archive (.zip, .gz) of the `JAVA_HOME` directory, so it includes the `bin`, `lib`, `include`,

jre, etc. directories.

YAML

```
- task: JavaToolInstaller@0
  inputs:
    versionSpec: '6'
    jdkArchitectureOption: 'x64'
    jdkSourceOption: AzureStorage
    azureResourceManagerEndpoint: myARMSERVICEConnection
    azureStorageAccountName: myAzureStorageAccountName
    azureContainerName: myAzureStorageContainerName
    azureCommonVirtualFile: 'jdk1.6.0_45.zip'
    jdkDestinationDirectory: '$(agent.toolsDirectory)/jdk6'
    cleanDestinationDirectory: false
```

Here's an example of using "pre-installed" feature. This feature allows you to use Java versions that are pre-installed on the Microsoft-hosted agent. You can find available pre-installed versions of Java in [the included software column in the hosted agents table](#).

YAML

```
- task: JavaToolInstaller@0
  inputs:
    versionSpec: '8'
    jdkArchitectureOption: 'x64'
    jdkSourceOption: 'PreInstalled'
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: Java, JDK
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : PATH, JAVA_HOME*
Agent version	2.182.1 or greater

Requirement	Description
Task category	Tool

See also

For an explanation of tool installers and examples, see [Tool installers](#).

JenkinsDownloadArtifacts@1 - Jenkins download artifacts v1 task

Article • 09/26/2023

Use this task to download artifacts produced by a Jenkins job.

Syntax

YAML

```
# Jenkins download artifacts v1
# Download artifacts produced by a Jenkins job.
- task: JenkinsDownloadArtifacts@1
  inputs:
    jenkinsServerConnection: # string. Alias: serverEndpoint. Required.
    Jenkins service connection.
    jobName: # string. Required. Job name.
    #jenkinsJobType: # string. Optional. Use when jobName = invalidjobName.
    Jenkins job type.
    saveTo: 'jenkinsArtifacts' # string. Required. Save to. Default:
    jenkinsArtifacts.
    # Advanced
    jenkinsBuild: 'LastSuccessfulBuild' # 'LastSuccessfulBuild' |
    'BuildNumber'. Required. Download artifacts produced by. Default:
    LastSuccessfulBuild.
    #jenkinsBuildNumber: '1' # string. Required when jenkinsBuild ==
    BuildNumber. Jenkins build number. Default: 1.
    #itemPattern: '**' # string. Item Pattern. Default: **.
    #downloadCommitsAndWorkItems: false # boolean. Download Commits and
    WorkItems. Default: false.
    #startJenkinsBuildNumber: # string. Optional. Use when
    downloadCommitsAndWorkItems == true && jenkinsBuild == BuildNumber. Download
    commits and work items from.
    #artifactDetailsFileNameSuffix: # string. Optional. Use when
    downloadCommitsAndWorkItems == invalid. Commit and WorkItem FileName.
    # Propagated Artifacts
    #propagatedArtifacts: false # boolean. Artifacts are propagated to
    Azure. Default: false.
    #artifactProvider: 'azureStorage' # 'azureStorage'. Required when
    propagatedArtifacts == notValid. Artifact Provider. Default: azureStorage.
    #ConnectedServiceNameARM: # string. Required when propagatedArtifacts ==
    true. Azure Subscription.
    #storageAccountName: # string. Required when propagatedArtifacts ==
    true. Storage Account Name.
    #containerName: # string. Required when propagatedArtifacts == true.
    Container Name.
    #commonVirtualPath: # string. Optional. Use when propagatedArtifacts ==
    true. Common Virtual Path.
```

Inputs

`jenkinsServerConnection` - Jenkins service connection

Input alias: `serverEndpoint`. `string`. Required.

Specifies the service connection for your Jenkins instance. To create a new service connection, click the Manage link.

`jobName` - Job name

`string`. Required.

Specifies the name of the Jenkins job to download artifacts from. This must exactly match the job name on the Jenkins server.

`jenkinsJobType` - Jenkins job type

`string`. Optional. Use when `jobName = invalidjobName`.

Automatically specifies the Jenkins job type.

`saveTo` - Save to

`string`. Required. Default value: `jenkinsArtifacts`.

Specifies the directory where Jenkins artifacts are downloaded and saved. This directory is created if it does not exist.

`jenkinsBuild` - Download artifacts produced by

`string`. Required. Allowed values: `LastSuccessfulBuild` (Last Successful Build), `BuildNumber` (Build Number). Default value: `LastSuccessfulBuild`.

Downloads artifacts produced by the last successful build or from a specific build instance.

`jenkinsBuildNumber` - Jenkins build number

`string`. Required when `jenkinsBuild == BuildNumber`. Default value: `1`.

Downloads artifacts produced by this build.

itemPattern - Item Pattern

`string`. Default value: `**`.

Specifies the files to be downloaded as a multi-line minimatch pattern. More Information about [file matching patterns](#).

The default pattern `**` downloads all files across all artifacts produced by the Jenkins job. To download all files within the artifact drop, use `drop/**`.

downloadCommitsAndWorkItems - Download Commits and WorkItems

`boolean`. Default value: `false`.

Enables downloading the commits and work item details associated with the Jenkins Job.

startJenkinsBuildNumber - Download commits and work items from

`string`. Optional. Use when `downloadCommitsAndWorkItems == true && jenkinsBuild == BuildNumber`.

Starts the build number for downloading commits and work items. If provided, all commits and work items between the start build number and the build number given as input to download artifacts are downloaded.

artifactDetailsFileNameSuffix - Commit and WorkItem FileName

`string`. Optional. Use when `downloadCommitsAndWorkItems == invalid`.

Specifies the file name suffix for commits and work item attachments. Attachments are created with `commits_{suffix}.json` and `workitem_{suffix}.json`. If this input is not provided, attachments are created with the names `commits.json` and `workitems.json`.

propagatedArtifacts - Artifacts are propagated to Azure

`boolean`. Default value: `false`.

Use this input if Jenkins artifacts were propagated to Azure. To upload Jenkins artifacts to Azure, refer to this [Jenkins plugin](#).

artifactProvider - Artifact Provider

`string`. Required when `propagatedArtifacts == notValid`. Allowed values: `azureStorage`

(Azure Storage). Default value: `azureStorage`.

Specifies the external storage provider used in Jenkins job to upload the artifacts.

ConnectedServiceNameARM - Azure Subscription

`string`. Required when `propagatedArtifacts == true`.

Specifies the Azure Resource Manager subscription for the artifacts.

storageAccountName - Storage Account Name

`string`. Required when `propagatedArtifacts == true`.

Specifies Azure Classic or Resource Manager storage accounts. Select the storage account name where the artifacts are propagated.

containerName - Container Name

`string`. Required when `propagatedArtifacts == true`.

Specifies the name of the container in the storage account where artifacts are uploaded.

commonVirtualPath - Common Virtual Path

`string`. Optional. Use when `propagatedArtifacts == true`.

Specifies the path to the artifacts inside the Azure storage container.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to download artifacts produced by a Jenkins job.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Utility

JenkinsQueueJob@2 - Jenkins queue job v2 task

Article • 09/26/2023

Use this task to queue a job on a Jenkins server.

Syntax

YAML

```
# Jenkins queue job v2
# Queue a job on a Jenkins server.
- task: JenkinsQueueJob@2
  inputs:
    serverEndpoint: # string. Required. Jenkins service connection.
    jobName: # string. Required. Job name.
    #isMultibranchJob: false # boolean. Job is of multibranch pipeline type.
    Default: false.
    #multibranchPipelineBranch: # string. Required when isMultibranchJob = true. Multibranch pipeline branch.
    #captureConsole: true # boolean. Capture console output and wait for completion. Default: true.
    #capturePipeline: true # boolean. Optional. Use when captureConsole = true. Capture pipeline output and wait for pipeline completion. Default: true.
    # Advanced
    #isParameterizedJob: false # boolean. Alias: parameterizedJob.
    Parameterized job. Default: false.
    #jobParameters: # string. Optional. Use when parameterizedJob = true.
    Job parameters.
    #failOnUnstableResult: false # boolean. Fail on unstable result.
    Default: false.
    #retryCount: '3' # string. Number of retries for failed connection.
    Default: 3.
    #delayBetweenRetries: '60' # string. Time between retries. Default: 60.
```

Inputs

`serverEndpoint` - Jenkins service connection

`string`. Required.

Specifies the service connection for your Jenkins instance. Click the Manage link to create a new Jenkins service connection.

jobName - Job name

`string`. Required.

The name of the Jenkins job to queue. This must exactly match the job name on the Jenkins server.

isMultibranchJob - Job is of multibranch pipeline type

`boolean`. Default value: `false`.

This job is a multibranch pipeline. If specified, add the appropriate branch name. This input requires Team Foundation Server Plugin for Jenkins v5.3.4 or later.

multibranchPipelineBranch - Multibranch pipeline branch

`string`. Required when `isMultibranchJob = true`.

Queues this multibranch pipeline job on the specified branch. This input requires Team Foundation Server Plugin for Jenkins v5.3.4 or later.

captureConsole - Capture console output and wait for completion

`boolean`. Default value: `true`.

If specified, this input captures the Jenkins build console output, waits for the Jenkins build to complete, and succeeds/fails based on the Jenkins build result. Otherwise, once the Jenkins job queues, this task successfully completes without waiting for the Jenkins build to run.

capturePipeline - Capture pipeline output and wait for pipeline completion

`boolean`. Optional. Use when `captureConsole = true`. Default value: `true`.

If specified, this task captures the full Jenkins build pipeline console output, waits for the full Jenkins build pipeline to complete, and succeeds/fails based on the Jenkins build pipeline result. Otherwise, once the first Jenkins job completes, this task successfully completes without waiting for full Jenkins build pipeline to run.

isParameterizedJob - Parameterized job

Input alias: `parameterizedJob`. `boolean`. Default value: `false`.

Specifies if the Jenkins job accepts parameters. Use this input even if all default parameter values are used and no parameters are actually specified.

`jobParameters` - Job parameters

`string`. Optional. Use when `parameterizedJob = true`.

Specifies job parameters, with one per line, in the format of `<parameterName>=<parameterValue>`.

To set a parameter to an empty value, which is useful for overriding a default value, leave off the parameter value. For example, specify `parameterName=`.

Variables are supported. To set a `commitId` parameter value to the Git commit ID of the build, for example, you can use: `commitId=$(Build.SourceVersion)`. For more information, see the [documentation on variables](#).

The supported Jenkins parameter types are:

- `Boolean`
- `Choice`
- `Password`
- `String`

`failOnUnstableResult` - Fail on unstable result

`boolean`. Default value: `false`.

Specifies strictness of a success definition, or whether to consider unstable as a failure or not. The `false` value is for a non-strict version, and the `true` is for a strict version. If set to `true`, an unstable build result is treated as a failure. Otherwise, an unstable result is treated as a success.

`retryCount` - Number of retries for failed connection

`string`. Default value: `3`.

Specifies the amount of connection retries when connection failure or error occurs.

`delayBetweenRetries` - Time between retries

`string`. Default value: `60`.

Specifies the amount of time between connection retries when an error occurs. This value is specified in seconds.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

JENKINS_JOB_ID

The ID of the Jenkins job instance queued by this task. Use this variable in the Jenkins Download Artifacts task to download the artifacts for this particular job instance.

Remarks

Use this task to queue a job on a Jenkins server.

Team Foundation Server Plug-in

You can use Team Foundation Server Plug-in (version 5.2.0 or newer) to automatically collect files from the Jenkins workspace and download them into the build.

To set it up:

1. Install the [Team Foundation Server Plug-in](#) on the Jenkins server.
2. On the Jenkins server, for each job you would like to collect results from, add the **Collect results for Azure Pipelines/TFS** post-build action and then configure it with one or more pairs of result type and include file pattern.
3. On the Jenkins Queue Job, build task enable the **Capture console output and wait for completion** to collect results from the root level job, or the **Capture pipeline output and wait for pipeline completion** to collect results from all pipeline jobs.

Results will be downloaded to the `$(Build.StagingDirectory)/jenkinsResults/Job Name/team-results.zip` and extracted to this location. Each set of result types collected by the plug-in, will be under the team-results directory, `$(Build.StagingDirectory)/jenkinsResults/Job Name/team-results/ResultType/`. This is

the directory where build results can be published by downstream tasks (for example, Publish Test Results, and Publish Code Coverage Results).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Build

JenkinsQueueJob@1 - Jenkins Queue Job v1 task

Article • 09/26/2023

Use this task to queue a job on a Jenkins server.

Syntax

YAML

```
# Jenkins Queue Job v1
# Queue a job on a Jenkins server.
- task: JenkinsQueueJob@1
  inputs:
    serverEndpoint: # string. Required. Jenkins service endpoint.
    jobName: # string. Required. Job name.
    #isMultibranchJob: false # boolean. Job is of Multibranch Pipeline type.
    Default: false.
    #multibranchPipelineBranch: # string. Required when isMultibranchJob = true. Multibranch Pipeline Branch.
    #captureConsole: true # boolean. Capture console output and wait for completion. Default: true.
    #capturePipeline: true # boolean. Optional. Use when captureConsole = true. Capture pipeline output and wait for pipeline completion. Default: true.
    # Advanced
    #parameterizedJob: false # boolean. Parameterized job. Default: false.
    #jobParameters: # string. Optional. Use when parameterizedJob = true.
    Job parameters.
```

Inputs

serverEndpoint - Jenkins service endpoint

`string`. Required.

Specifies the service endpoint for your Jenkins instance. Click the Manage link (when using the task assistant) to create a new Jenkins service endpoint.

jobName - Job name

`string`. Required.

The name of the Jenkins job to queue. This must exactly match the job name on the Jenkins server.

isMultibranchJob - Job is of Multibranch Pipeline type

`boolean`. Default value: `false`.

This job is a multibranch pipeline. If specified, add the appropriate branch name. This input requires Team Foundation Server Plugin for Jenkins v5.3.4 or later.

multibranchPipelineBranch - Multibranch Pipeline Branch

`string`. Required when `isMultibranchJob = true`.

Queues this multibranch pipeline job on the specified branch. This input requires Team Foundation Server Plugin for Jenkins v5.3.4 or later.

captureConsole - Capture console output and wait for completion

`boolean`. Default value: `true`.

If specified, this input captures the Jenkins build console output, waits for the Jenkins build to complete, and succeeds/fails based on the Jenkins build result. Otherwise, once the Jenkins job queues, this step successfully completes without waiting for the Jenkins build to run.

capturePipeline - Capture pipeline output and wait for pipeline completion

`boolean`. Optional. Use when `captureConsole = true`. Default value: `true`.

If specified, this input captures the full Jenkins build pipeline console output, waits for the full Jenkins build pipeline to complete, and succeeds/fails based on the Jenkins build pipeline result. Otherwise, once the first Jenkins job completes, this input successfully completes without waiting for the full Jenkins build pipeline to run.

parameterizedJob - Parameterized job

`boolean`. Default value: `false`.

Specifies if the Jenkins job accepts parameters. Use this input even if all default parameter values are used and no parameters are actually specified.

jobParameters - Job parameters

`string`. Optional. Use when `parameterizedJob = true`.

Specifies job parameters with one per line, for example: `<parameterName>=<parameterValue>`.

To set a parameter to an empty value, which is useful for overriding a default value, leave off the parameter value. For example, specify `<parameterName>=`.

Variables are supported. To set a `commitId` parameter value to the Git commit ID of the build, for example, you can use: `commitId=$(Build.SourceVersion)`. For more information, see the [documentation on variables](#).

The supported Jenkins parameter types are:

- `Boolean`
- `Choice`
- `Password`
- `String`

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.

Requirement	Description
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Build

Kubernetes@1 - Kubectl v1 task

Article • 09/26/2023

Deploy, configure, update a Kubernetes cluster in Azure Container Service by running kubectl commands.

Syntax

YAML

```
# Kubectl v1
# Deploy, configure, update a Kubernetes cluster in Azure Container Service
# by running kubectl commands.
- task: Kubernetes@1
  inputs:
    # Kubernetes Cluster
    #connectionType: 'Kubernetes Service Connection' # 'Azure Resource
    Manager' | 'Kubernetes Service Connection' | 'None'. Required when command
    != logout. Service connection type. Default: Kubernetes Service Connection.
    #kubernetesServiceEndpoint: # string. Required when connectionType =
    Kubernetes Service Connection && command != logout. Kubernetes service
    connection.
    #azureSubscriptionEndpoint: # string. Required when connectionType =
    Azure Resource Manager && command != logout. Azure subscription.
    #azureResourceGroup: # string. Required when connectionType = Azure
    Resource Manager && command != logout. Resource group.
    #kubernetesCluster: # string. Required when connectionType = Azure
    Resource Manager && command != logout. Kubernetes cluster.
    #useClusterAdmin: false # boolean. Optional. Use when connectionType =
    Azure Resource Manager && command != logout. Use cluster admin credentials.
    Default: false.
    #namespace: # string. Optional. Use when command != logout. Namespace.
    # Commands
    #command: # 'apply' | 'create' | 'delete' | 'exec' | 'expose' | 'get' |
    'login' | 'logout' | 'logs' | 'run' | 'set' | 'top'. Command.
    #useConfigurationFile: false # boolean. Optional. Use when command !=
    login && command != logout. Use configuration. Default: false.
    #configurationType: 'configuration' # 'configuration' | 'inline'.
    Optional. Use when useConfigurationFile = true. Configuration type. Default:
    configuration.
    configuration: # string. Required when configurationType =
    configuration. File path.
    #inline: # string. Required when configurationType = inline. Inline
    configuration.
    #arguments: # string. Optional. Use when command != login && command !=
    logout. Arguments.
    # Secrets
    #secretType: 'dockerRegistry' # 'dockerRegistry' | 'generic'. Required
    when command != login && command != logout. Type of secret. Default:
    dockerRegistry.
```

```

    #secretArguments: # string. Optional. Use when secretType = generic &&
    command != login && command != logout. Arguments.

    #containerRegistryType: 'Azure Container Registry' # 'Azure Container
    Registry' | 'Container Registry'. Required when secretType = dockerRegistry
    && command != login && command != logout. Container registry type. Default:
    Azure Container Registry.

    #dockerRegistryEndpoint: # string. Optional. Use when secretType =
    dockerRegistry && containerRegistryType = Container Registry && command !=
    login && command != logout. Docker registry service connection.

    #azureSubscriptionEndpointForSecrets: # string. Optional. Use when
    secretType = dockerRegistry && containerRegistryType = Azure Container
    Registry && command != login && command != logout. Azure subscription.

    #azureContainerRegistry: # string. Optional. Use when secretType =
    dockerRegistry && containerRegistryType = Azure Container Registry &&
    command != login && command != logout. Azure container registry.

    #secretName: # string. Optional. Use when command != login && command !=
    logout. Secret name.

    #forceUpdate: true # boolean. Optional. Use when command != login &&
    command != logout. Force update secret. Default: true.

    # ConfigMaps

    #configMapName: # string. Optional. Use when command != login && command
    != logout. ConfigMap name.

    #forceUpdateConfigMap: false # boolean. Optional. Use when command !=
    login && command != logout. Force update configmap. Default: false.

    #useConfigMapFile: false # boolean. Optional. Use when command != login
    && command != logout. Use file. Default: false.

    #configMapFile: # string. Required when useConfigMapFile = true &&
    command != login && command != logout. ConfigMap file.

    #configMapArguments: # string. Optional. Use when useConfigMapFile =
    false && command != login && command != logout. Arguments.

    # Advanced

    #versionOrLocation: 'version' # 'version' | 'location'. Kubectl.
Default: version.

    #versionSpec: '1.13.2' # string. Optional. Use when versionOrLocation =
    version. Version spec. Default: 1.13.2.

    #checkLatest: false # boolean. Optional. Use when versionOrLocation =
    version. Check for latest version. Default: false.

    #specifyLocation: # string. Required when versionOrLocation = location.
Path to kubectl.

    #workingDirectory: '$(System.DefaultWorkingDirectory)' # string. Alias:
    cwd. Working directory. Default: $(System.DefaultWorkingDirectory).

    #outputFormat: 'json' # 'json' | 'yaml' | 'none'. Output format.
Default: json.

```

Inputs

`connectionType` - Service connection type

`string`. Required when `command != logout`. Allowed values: `Azure Resource Manager`, `Kubernetes Service Connection`, `None`. Default value: `Kubernetes Service Connection`.

Specifies the service connection type: Azure Resource Manager when using Azure Kubernetes Service or Kubernetes Service Connection for any other cluster.

- **Kubernetes Service Connection** - Allows you to provide a KubeConfig file, specify a Service Account, or import an AKS instance with the **Azure Subscription** option. Importing an AKS instance with the **Azure Subscription** option requires Kubernetes cluster access at Service Connection configuration time.
- **Azure Resource Manager** - Lets you select an AKS instance. Does not access Kubernetes cluster at Service Connection configuration time.
- **None** - Use a pre-created Kubernetes configuration stored locally.

For more information, see [Service connection](#) in the following [Remarks](#) section.

kubernetesServiceEndpoint - Kubernetes service connection

`string`. Required when `connectionType = Kubernetes Service Connection && command != logout`.

Select a Kubernetes service connection.

azureSubscriptionEndpoint - Azure subscription

`string`. Required when `connectionType = Azure Resource Manager && command != logout`.

Specifies the Azure Resource Manager subscription, which contains the Azure Container Registry.

Note

To configure a new service connection, specify the Azure subscription from the list and click `Authorize`. If your subscription is not listed or if you want to use an existing Service Principal, you can setup an Azure service connection using the `Add` or `Manage` buttons.

azureResourceGroup - Resource group

`string`. Required when `connectionType = Azure Resource Manager && command != logout`.

Select an Azure resource group.

kubernetesCluster - Kubernetes cluster

`string`. Required when `connectionType = Azure Resource Manager && command != logout`.

Select an Azure managed cluster.

useClusterAdmin - Use cluster admin credentials

`boolean`. Optional. Use when `connectionType = Azure Resource Manager && command != logout`. Default value: `false`.

Use cluster administrator credentials instead of default cluster user credentials.

namespace - Namespace

`string`. Optional. Use when `command != logout`.

Set the namespace for the `kubectl` command by using the `-namespace` flag. If the namespace is not provided, the commands will run in the default namespace.

command - Command

`string`. Allowed values: `apply`, `create`, `delete`, `exec`, `expose`, `get`, `login`, `logout`, `logs`, `run`, `set`, `top`.

Select or specify a `kubectl` command to run.

useConfigurationFile - Use configuration

`boolean`. Optional. Use when `command != login && command != logout`. Default value: `false`.

Specifies the Kubernetes configuration to use with the `kubectl` command. The inline script, filename, directory, or URL to Kubernetes configuration files can be provided.

configurationType - Configuration type

`string`. Optional. Use when `useConfigurationFile = true`. Allowed values: `configuration` (File path), `inline` (Inline configuration). Default value: `configuration`.

Specifies the type of Kubernetes configuration for the `kubectl` command. It can be a file path or an inline script.

configuration - File path

`string`. Required when `configurationType = configuration`.

Specifies the filename, directory, or URL to kubernetes configuration files that is used with the commands.

inline - Inline configuration

`string`. Required when `configurationType = inline`.

Specifies the inline deployment configuration for the `kubectl` command.

arguments - Arguments

`string`. Optional. Use when `command != login && command != logout`.

Arguments to the specified kubectl command.

secretType - Type of secret

`string`. Required when `command != login && command != logout`. Allowed values:

`dockerRegistry`, `generic`. Default value: `dockerRegistry`.

Create/update a generic or docker imagepullsecret. Select dockerRegistry to create/update the imagepullsecret of the selected registry. An imagePullSecret is a way to pass a secret that contains a container registry password to the Kubelet so it can pull a private image on behalf of your Pod.

secretArguments - Arguments

`string`. Optional. Use when `secretType = generic && command != login && command != logout`.

Specifies the keys and literal values to insert in secret. For example, `--from-literal=key1=value1` or `--from-literal=key2="top secret"`.

containerRegistryType - Container registry type

`string`. Required when `secretType = dockerRegistry && command != login && command != logout`. Allowed values: `Azure Container Registry`, `Container Registry`. Default value: `Azure Container Registry`.

Select a Container registry type. The task can use Azure Subscription details to work with an Azure Container registry. Other standard Container registries are also supported.

dockerRegistryEndpoint - Docker registry service connection

```
string. Optional. Use when secretType = dockerRegistry && containerRegistryType = Container Registry && command != login && command != logout.
```

Select a Docker registry service connection. Required for commands that need to authenticate with a registry.

azureSubscriptionEndpointForSecrets - Azure subscription

```
string. Optional. Use when secretType = dockerRegistry && containerRegistryType = Azure Container Registry && command != login && command != logout.
```

Specifies the Azure Resource Manager subscription, which contains Azure Container Registry.

Note

To configure a new service connection, select the Azure subscription from the list and click `Authorize`. If your subscription is not listed or if you want to use an existing Service Principal, you can setup an Azure service connection using the `Add` or `Manage` buttons.

azureContainerRegistry - Azure container registry

```
string. Optional. Use when secretType = dockerRegistry && containerRegistryType = Azure Container Registry && command != login && command != logout.
```

Specifies an Azure Container Registry which is used for pulling container images and deploying applications to the Kubernetes cluster. Required for commands that need to authenticate with a registry.

secretName - Secret name

```
string. Optional. Use when command != login && command != logout.
```

Name of the secret. You can use this secret name in the Kubernetes YAML configuration file.

`forceUpdate` - Force update secret

`boolean`. Optional. Use when `command != login && command != logout`. Default value: `true`.

Delete the secret if it exists and create a new one with updated values.

`configMapName` - ConfigMap name

`string`. Optional. Use when `command != login && command != logout`.

ConfigMaps allow you to decouple configuration artifacts from image content to keep containerized applications portable.

`forceUpdateConfigMap` - Force update configmap

`boolean`. Optional. Use when `command != login && command != logout`. Default value: `false`.

Delete the configmap if it exists and create a new one with updated values.

`useConfigMapFile` - Use file

`boolean`. Optional. Use when `command != login && command != logout`. Default value: `false`.

Creates a `ConfigMap` from an individual file or from multiple files by specifying a directory.

`configMapFile` - ConfigMap file

`string`. Required when `useConfigMapFile = true && command != login && command != logout`.

Specify a file or directory that contains the configMaps.

`configMapArguments` - Arguments

`string`. Optional. Use when `useConfigMapFile = false && command != login && command != logout`.

Specifies the keys and literal values to insert in `configMap`. For example, `--from-literal=key1=value1` or `--from-literal=key2="top secret"`.

versionOrLocation - Kubectl

`string`. Allowed values: `version`, `location` (Specify location). Default value: `version`.

kubectl is a command line interface for running commands against Kubernetes clusters.

versionSpec - Version spec

`string`. Optional. Use when `versionOrLocation = version`. Default value: `1.13.2`.

Specifies the version spec of the version to get. Examples: `1.7.0`, `1.x.0`, `4.x.0`, `6.10.0`,
`>=6.10.0`.

checkLatest - Check for latest version

`boolean`. Optional. Use when `versionOrLocation = version`. Default value: `false`.

Always checks online for the latest available version (stable.txt) that satisfies the version spec. This is typically false unless you have a specific scenario to always get latest. This will cause it to incur download costs when potentially not necessary, especially with the hosted build pool.

specifyLocation - Path to kubectl

`string`. Required when `versionOrLocation = location`.

Specifies the full path to the `kubectl.exe` file.

workingDirectory - Working directory

Input alias: `cwd`. `string`. Default value: `$(System.DefaultWorkingDirectory)`.

Working directory for the Kubectl command.

outputFormat - Output format

`string`. Allowed values: `json`, `yaml`, `none`. Default value: `json`.

Output format.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

KubectlOutput

Stores the output of the `kubectl` command.

Remarks

What's new in Version 1.0.

- Added a new service connection type input for easy selection of Azure AKS clusters.
- Replaced the output variable input with an output variables section that we added in all tasks.

Use this task to deploy, configure, or update a Kubernetes cluster by running `kubectl` commands.

Service connection

The task works with two service connection types: **Azure Resource Manager** and **Kubernetes Service Connection**, described below.

Azure Resource Manager

Set `connectionType` to `Azure Resource Manager` and specify an `azureSubscriptionEndpoint` to use an Azure Resource Manager service connection.

This YAML example shows how Azure Resource Manager is used to refer to the Kubernetes cluster. This is to be used with one of the `kubectl` [commands](#) and the appropriate values required by the command.

YAML

```
variables:  
  azureSubscriptionEndpoint: Contoso  
  azureContainerRegistry: contoso.azurecr.io
```

```
azureResourceGroup: Contoso
kubernetesCluster: Contoso
useClusterAdmin: false

steps:
- task: Kubernetes@1
  displayName: kubectl apply
  inputs:
    connectionType: Azure Resource Manager
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureResourceGroup: $(azureResourceGroup)
    kubernetesCluster: $(kubernetesCluster)
    useClusterAdmin: $(useClusterAdmin)
```

Kubernetes Service Connection

Set `connectionType` to `Kubernetes Service Connection` and specify a `kubernetesServiceEndpoint` to use a Kubernetes service connection.

This YAML example shows how a Kubernetes Service Connection is used to refer to the Kubernetes cluster. This is to be used with one of the `kubectl` [commands](#) and the appropriate values required by the command.

YAML

```
- task: Kubernetes@1
  displayName: kubectl apply
  inputs:
    connectionType: Kubernetes Service Connection
    kubernetesServiceEndpoint: Contoso
```

Kubernetes Service Connection considerations when accessing AKS

You can create a Kubernetes service connection with any of the following options.

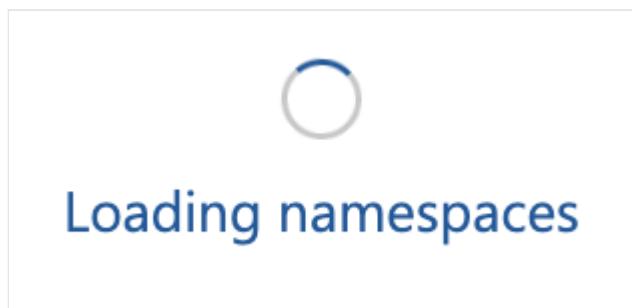
- KubeConfig
- Service Account
- Azure Subscription

New Kubernetes service connection

Authentication method

- KubeConfig
- Service Account
- Azure Subscription

When selecting the **Azure Subscription** option, Kubernetes needs to be accessible to Azure DevOps at service connection configuration time. There may be various reasons a service connection cannot be created, for example you created a private cluster or the cluster has local accounts disabled. In these cases, Azure DevOps is unable to connect to your cluster at service connection configuration time and you will observe the dialog to be stuck at **Loading namespaces**.



Starting with Kubernetes 1.24, long-lived tokens are [no longer created by default](#). Kubernetes recommends not to use long-lived tokens. As a result, tasks using a Kubernetes service connection created using the Azure Subscription option do not have access to the permanent token required to authenticate and can't access your Kubernetes cluster. This also results in the **Loading namespaces** dialog to be frozen.

Use the Azure Resource Manager Service Connection to access AKS

For AKS customers, the Azure Resource Manager service connection type provides the best method to connect to a private cluster, or a cluster that has local accounts disabled. This method is not dependent on cluster connectivity at the time you create a service connection. Access to AKS is deferred to pipeline runtime, which has the following advantages:

- Access to a (private) AKS cluster can be performed from a self-hosted or scale set agent with line of sight to the cluster.
- A token is created for every task that uses an Azure Resource Manager service connection. This ensures you are connecting to Kubernetes with a short-lived token, which is the [Kubernetes recommendation](#).
- AKS can be accessed even when local accounts are disabled.

Service connection FAQ

I receive the following error message: Could not find any secret associated with the service account. What is happening?

You are using the Kubernetes service connection with Azure Subscription option. We are updating this method to create long-lived tokens. This is expected to be available mid-May. However, it is recommended to start using the Azure service connection type and not to use long-lived tokens as per [Kubernetes guidance](#).

I'm using AKS and don't want to change anything, can I continue to use tasks with the Kubernetes service connection?

We are updating this method to create long-lived tokens. This is expected to be available mid-May. However, please be aware that this approach is against [Kubernetes guidance](#).

I'm using the Kubernetes tasks and Kubernetes service connection but not AKS. Should I be concerned?

Your tasks will continue to work as before.

Will the Kubernetes service connection type be removed?

Our Kubernetes tasks work with any Kubernetes cluster, regardless where they are running. The Kubernetes service connection will continue to exist.

I'm an AKS customer and everything is running fine, should I act?

There is no need to change anything. If you are using the Kubernetes service connection and selected Azure Subscription during creation, you should be aware of the [Kubernetes guidance on using long-lived tokens](#).

I'm creating a Kubernetes Environment, and have no option to use service connections

In case you can't access your AKS during environment creation time, you can use an empty environment and set the `connectionType` input to an Azure Resource Manager service connection.

I have AKS configured with Azure Active Directory RBAC, and my pipeline doesn't work. Will these updates resolve that?

Accessing Kubernetes when AAD RBAC is enabled is unrelated to token creation. To prevent an interactive prompt, we will support `kubelogin` in a future update.

Commands

The command input accepts one of the following [kubectl commands](#):

`apply`, `create`, `delete`, `exec`, `expose`, `get`, `login`, `logout`, `logs`, `run`, `set`, or `top`.

This YAML example demonstrates the `apply` command:

YAML

```
- task: Kubernetes@1
  displayName: kubectl apply using arguments
  inputs:
    connectionType: Azure Resource Manager
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureResourceGroup: $(azureResourceGroup)
    kubernetesCluster: $(kubernetesCluster)
    command: apply
    arguments: -f mhc-aks.yaml
```

This YAML example demonstrates the use of a configuration file with the `apply` command:

YAML

```
- task: Kubernetes@1
  displayName: kubectl apply using configFile
  inputs:
    connectionType: Azure Resource Manager
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureResourceGroup: $(azureResourceGroup)
    kubernetesCluster: $(kubernetesCluster)
    command: apply
```

```
useConfigurationFile: true
configuration: mhc-aks.yaml
```

Secrets

Kubernetes objects of type **secret** are intended to hold sensitive information such as passwords, OAuth tokens, and ssh keys. Putting this information in a secret is safer and more flexible than putting it verbatim in a pod definition or in a Docker image. Azure Pipelines simplifies the addition of `ImagePullSecrets` to a service account, or setting up of any generic secret, as described below.

ImagePullSecret

This YAML example demonstrates the setting up of ImagePullSecrets:

YAML

```
- task: Kubernetes@1
  displayName: kubectl apply for secretType dockerRegistry
  inputs:
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureResourceGroup: $(azureResourceGroup)
    kubernetesCluster: $(kubernetesCluster)
    command: apply
    arguments: -f mhc-aks.yaml
    secretType: dockerRegistry
    containerRegistryType: Azure Container Registry
    azureSubscriptionEndpointForSecrets: $(azureSubscriptionEndpoint)
    azureContainerRegistry: $(azureContainerRegistry)
    secretName: mysecretkey2
    forceUpdate: true
```

Generic Secrets

This YAML example creates generic secrets from literal values specified for the **secretArguments** input:

YAML

```
- task: Kubernetes@1
  displayName: secretType generic with literal values
  inputs:
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureResourceGroup: $(azureResourceGroup)
    kubernetesCluster: $(kubernetesCluster)
    command: apply
```

```
arguments: -f mhc-aks.yaml
secretType: generic
secretArguments: --from-literal=contoso=5678
secretName: mysecretkey
```

Pipeline variables can be used to pass arguments for specifying literal values, as shown here:

YAML

```
- task: Kubernetes@1
displayName: secretType generic with pipeline variables
inputs:
  azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
  azureResourceGroup: $(azureResourceGroup)
  kubernetesCluster: $(kubernetesCluster)
  command: apply
  arguments: -f mhc-aks.yaml
  secretType: generic
  secretArguments: --from-literal=contoso=$(contosovalue)
  secretName: mysecretkey
```

ConfigMap

ConfigMaps allow you to decouple configuration artifacts from image content to maintain portability for containerized applications.

This YAML example creates a ConfigMap by pointing to a ConfigMap file:

YAML

```
- task: Kubernetes@1
displayName: kubectl apply
inputs:
  configMapName: myconfig
  useConfigMapFile: true
  configMapFile: src/configmap
```

This YAML example creates a ConfigMap by specifying the literal values directly as the **configMapArguments** input, and setting **forceUpdate** to true:

YAML

```
- task: Kubernetes@1
displayName: configMap with literal values
inputs:
  azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
  azureResourceGroup: $(azureResourceGroup)
```

```
kubernetesCluster: $(kubernetesCluster)
command: apply
arguments: -f mhc-aks.yaml
secretType: generic
secretArguments: --from-literal=contoso=$(contosovalue)
secretName: mysecretkey4
configMapName: myconfig
forceUpdateConfigMap: true
configMapArguments: --from-literal=myname=contoso
```

You can use pipeline variables to pass literal values when creating ConfigMap, as shown here:

YAML

```
- task: Kubernetes@1
  displayName: configMap with pipeline variables
  inputs:
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureResourceGroup: $(azureResourceGroup)
    kubernetesCluster: $(kubernetesCluster)
    command: apply
    arguments: -f mhc-aks.yaml
    secretType: generic
    secretArguments: --from-literal=contoso=$(contosovalue)
    secretName: mysecretkey4
    configMapName: myconfig
    forceUpdateConfigMap: true
    configMapArguments: --from-literal=myname=$(contosovalue)
```

Troubleshooting

My Kubernetes cluster is behind a firewall and I am using hosted agents. How can I deploy to this cluster?

You can grant hosted agents access through your firewall by allowing the IP addresses for the hosted agents. For more details, see [Agent IP ranges](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup

Requirement	Description
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Deploy

Kubernetes@0 - Kubectl v0 task

Article • 09/26/2023

Use this task to deploy, configure, or update a Kubernetes cluster in Azure Container Service by running `kubectl` commands.

Syntax

YAML

```
# Kubectl v0
# Deploy, configure, update a Kubernetes cluster in Azure Container Service
by running kubectl commands.
- task: Kubernetes@0
  inputs:
    #kubernetesServiceConnection: # string. Alias:
    kubernetesServiceEndpoint. Kubernetes service connection.
    #namespace: # string. Namespace.
    # Commands
    #command: # 'apply' | 'create' | 'delete' | 'exec' | 'expose' | 'get' |
    'logs' | 'run' | 'set' | 'top'. Command.
    #useConfigurationFile: false # boolean. Use Configuration files.
  Default: false.
    #configuration: # string. Required when useConfigurationFile = true.
  Configuration file.
    #arguments: # string. Arguments.
  # Secrets
    secretType: 'dockerRegistry' # 'dockerRegistry' | 'generic'. Required.
  Type of secret. Default: dockerRegistry.
    #secretArguments: # string. Optional. Use when secretType = generic.
  Arguments.
    containerRegistryType: 'Azure Container Registry' # 'Azure Container
  Registry' | 'Container Registry'. Required when secretType = dockerRegistry.
  Container Registry type. Default: Azure Container Registry.
    #dockerRegistryConnection: # string. Alias: dockerRegistryEndpoint.
  Optional. Use when secretType = dockerRegistry && containerRegistryType =
  Container Registry. Docker Registry service connection.
    #azureSubscription: # string. Alias: azureSubscriptionEndpoint.
  Optional. Use when secretType = dockerRegistry && containerRegistryType =
  Azure Container Registry. Azure subscription.
    #azureContainerRegistry: # string. Optional. Use when secretType =
  dockerRegistry && containerRegistryType = Azure Container Registry. Azure
  Container Registry.
    #secretName: # string. Secret name.
    #forceUpdate: true # boolean. Force update secret. Default: true.
  # ConfigMaps
    #configMapName: # string. ConfigMap name.
    #forceUpdateConfigMap: false # boolean. Force update configmap. Default:
  false.
    #useConfigMapFile: false # boolean. Use file. Default: false.
```

```
#configMapFile: # string. Required when useConfigMapFile = true.  
ConfigMap file.  
#configMapArguments: # string. Optional. Use when useConfigMapFile =  
false. Arguments.  
# Advanced  
#versionOrLocation: 'version' # 'version' | 'location'. Kubectl.  
Default: version.  
#versionSpec: '1.7.0' # string. Optional. Use when versionOrLocation =  
version. Version spec. Default: 1.7.0.  
#checkLatest: false # boolean. Optional. Use when versionOrLocation =  
version. Check for latest version. Default: false.  
#specifyLocation: # string. Required when versionOrLocation = location.  
Path to Kubectl.  
#workingDirectory: '$(System.DefaultWorkingDirectory)' # string. Alias:  
cwd. Working directory. Default: $(System.DefaultWorkingDirectory).  
# Output  
#outputFormat: 'json' # 'json' | 'yaml'. Output format. Default: json.  
#kubectlOutput: # string. Output variable name.
```

Inputs

`kubernetesServiceConnection` - Kubernetes service connection

Input alias: `kubernetesServiceEndpoint`. `string`.

Select a Kubernetes service connection.

`namespace` - Namespace

`string`.

Specifies the namespace for the `kubectl` command by using the `-namespace` flag. If the namespace is not provided, the commands will run in the default namespace.

`command` - Command

`string`. Allowed values: `apply`, `create`, `delete`, `exec`, `expose`, `get`, `logs`, `run`, `set`, `top`.

Specifies a `kubectl` command to run.

`useConfigurationFile` - Use Configuration files

`boolean`. Default value: `false`.

Use Kubernetes configuration file with the `kubectl` command. Filename, directory, or URL to Kubernetes configuration files can also be provided.

configuration - Configuration file

`string`. Required when `useConfigurationFile = true`.

Specifies the filename, directory, or URL to kubernetes configuration files that is used with the commands.

arguments - Arguments

`string`.

Specifies the arguments to the specified `kubectl` command.

secretType - Type of secret

`string`. Required. Allowed values: `dockerRegistry`, `generic`. Default value: `dockerRegistry`.

Creates or updates a generic or docker `imagepullsecret`. Specify `dockerRegistry` to create or update the `imagepullsecret` of the selected registry. An `imagePullSecret` is a way to pass a secret that contains a container registry password to the Kubelet, so it can pull a private image on behalf of your Pod.

secretArguments - Arguments

`string`. Optional. Use when `secretType = generic`.

Specifies keys and literal values to insert in secret. For example, `--from-literal=key1=value1` or `--from-literal=key2="top secret"`.

containerRegistryType - Container Registry type

`string`. Required when `secretType = dockerRegistry`. Allowed values: `Azure Container Registry`, `Container Registry`. Default value: `Azure Container Registry`.

Select a Container registry type. The task can use Azure Subscription details to work with an Azure Container registry. Other standard Container registries are also supported.

dockerRegistryConnection - Docker Registry service connection

Input alias: `dockerRegistryEndpoint`. `string`. Optional. Use when `secretType = dockerRegistry && containerRegistryType = Container Registry`.

Select a Docker registry service connection. Required for commands that need to authenticate with a registry.

`azureSubscription` - Azure subscription

Input alias: `azureSubscriptionEndpoint`. `string`. Optional. Use when `secretType = dockerRegistry && containerRegistryType = Azure Container Registry`.

Specifies the Azure Resource Manager subscription, which contains Azure Container Registry.

ⓘ Note

To configure a new service connection, select the Azure subscription from the list and click `Authorize`. If your subscription is not listed or if you want to use an existing Service Principal, you can setup an Azure service connection using the `Add` or `Manage` buttons.

`azureContainerRegistry` - Azure Container Registry

`string`. Optional. Use when `secretType = dockerRegistry && containerRegistryType = Azure Container Registry`.

Specifies an Azure Container Registry which is used for pulling container images and deploying applications to the Kubernetes cluster. Required for commands that need to authenticate with a registry.

`secretName` - Secret name

`string`.

Name of the secret. You can use this secret name in the Kubernetes YAML configuration file.

`forceUpdate` - Force update secret

`boolean`. Default value: `true`.

Delete the secret if it exists and create a new one with updated values.

`configMapName` - ConfigMap name

`string`.

ConfigMaps allow you to decouple configuration artifacts from image content to keep containerized applications portable.

`forceUpdateConfigMap` - Force update configmap

`boolean`. Default value: `false`.

Delete the configmap if it exists and create a new one with updated values.

`useConfigMapFile` - Use file

`boolean`. Default value: `false`.

Create a ConfigMap from an individual file, or from multiple files by specifying a directory.

`configMapFile` - ConfigMap file

`string`. Required when `useConfigMapFile = true`.

Specify a file or directory that contains the configMaps.

`configMapArguments` - Arguments

`string`. Optional. Use when `useConfigMapFile = false`.

Specifies keys and literal values to insert in `configMap`. For example, `--from-literal=key1=value1` or `--from-literal=key2="top secret"`.

`versionOrLocation` - Kubectl

`string`. Allowed values: `version`, `location` (Specify location). Default value: `version`.

kubectl is a command line interface for running commands against Kubernetes clusters.

`versionSpec` - Version spec

`string`. Optional. Use when `versionOrLocation = version`. Default value: `1.7.0`.

Specifies the Version Spec of the version to get. Examples: `1.7.0`, `1.x.0`, `4.x.0`, `6.10.0`, `>=6.10.0`.

checkLatest - Check for latest version

`boolean`. Optional. Use when `versionOrLocation = version`. Default value: `false`.

Always checks online for the latest available version (stable.txt) that satisfies the version spec. This is typically false unless you have a specific scenario to always get latest. This will cause it to incur download costs when potentially not necessary, especially with the hosted build pool.

specifyLocation - Path to Kubectl

`string`. Required when `versionOrLocation = location`.

Specifies the full path to the `kubectl.exe` file.

workingDirectory - Working directory

Input alias: `cwd`. `string`. Default value: `$(System.DefaultWorkingDirectory)`.

Working directory for the Kubectl command.

outputFormat - Output format

`string`. Allowed values: `json`, `yaml`. Default value: `json`.

Output format.

kubectlOutput - Output variable name

`string`.

Name of the variable in which output of the command should be saved.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Deploy

KubectlInstaller@0 - Kubectl tool installer v0 task

Article • 09/26/2023

Use this task for installing a specific version of kubectl binary on agents.

Syntax

YAML

```
# Kubectl tool installer v0
# Install Kubectl on agent machine.
- task: KubectlInstaller@0
  inputs:
    #kubectlVersion: 'latest' # string. Kubectl Version Spec. Default:
    latest.
```

Inputs

kubectlVersion - Kubectl Version Spec

`string`. Default value: `latest`.

Specifies the version of kubectl to install. The acceptable values are `latest` or any semantic version string, e.g. `1.15.0`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

This task is used for installing a specific version of kubectl binary on agents.

Examples

The following YAML example showcases the installation of latest version of kubectl binary on the agent:

YAML

```
- task: KubectlInstaller@0
  displayName: Kubectl installer
  inputs:
    kubectlVersion: latest
```

The following YAML example demonstrates the use of an explicit version string rather than installing the latest version available at the time of task execution:

YAML

```
- task: KubectlInstaller@0
  displayName: Kubectl installer
  inputs:
    kubectlVersion: 1.15.0
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: Kubectl
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Tool

KubeloginInstaller@0 - Kubelogin tool installer v0 task

Article • 09/26/2023

Installs kubelogin and adds it to the PATH of your agent.

Syntax

YAML

```
# Kubelogin tool installer v0
# Helps to install kubelogin.
- task: KubeloginInstaller@0
  inputs:
    #kubeloginVersion: 'latest' # string. kubelogin version. Default:
    latest.
```

Inputs

`kubeloginVersion` - kubelogin version

`string`. Default value: `latest`.

The version of kubelogin to use, for example `0.0.30`, or `latest` to use the latest version. For more information about kubelogin versions, see [kubelogin releases](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

The kubelogin installer task acquires the specified version of [kubelogin](#) from the internet or the tools cache and adds it to the PATH of the agent (hosted or private). Use this task to change the version of kubelogin used in subsequent tasks like [KubernetesManifest@1](#), [HelmDeploy@0](#), [AzureFunctionOnKubernetes@1](#), and [Kubernetes@1](#).

Adding `KubeloginInstaller@0` before the previously listed tasks in a build definition ensures that the desired kubelogin version is available at the time of building, testing and publishing your app.

The tool installer approach also allows you to decouple from the agent update cycles. If the kubelogin version you are looking for is missing from the agent (hosted or private), then you can use `KubeloginInstaller@0` to get the right version installed on the agent.

For more information on kubelogin, see [Non-interactive sign-in with kubelogin](#).

Examples

The following example shows how to install the latest version of kubelogin. The default value for `kubeloginVersion` is `latest`, so you can omit the `kubeloginVersion` input if desired.

```
yml
- task: KubeloginInstaller@0

# Other tasks that depend on kubelogin
- task: HelmDeploy@0
# task inputs...
```

To explicitly specify `kubeloginVersion`, use the following syntax.

```
yml
- task: KubeloginInstaller@0
  inputs:
    kubeloginVersion: 'latest' # or a specific version like '0.0.30'
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release

Requirement	Description
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: Kubelogin
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Tool

See also

- [Non-interactive sign-in with kubelogin](#)

ManualIntervention@8 - Manual intervention v8 task

Article • 09/26/2023

Use this task to pause deployment in a release pipeline and wait for manual intervention.

Syntax

YAML

```
# Manual intervention v8
# Pause deployment and wait for manual intervention.
- task: ManualIntervention@8
  inputs:
    #instructions: # string. Instructions.
    #emailRecipients: # string. Notify users.
    #onTimeout: 'reject' # 'reject' | 'resume'. On timeout. Default: reject.
```

Inputs

instructions - Instructions

`string`.

Specifies the instructions that are shown to the user when resuming or rejecting the manual intervention. Based on these instructions, the user will make an informed decision about this manual intervention.

emailRecipients - Notify users

`string`.

Sends a manual intervention pending email to specific users (or groups). Only users with manage deployment permission can act on a manual intervention.

onTimeout - On timeout

`string`. Allowed values: `reject`, `resume`. Default value: `reject`.

Automatically rejects or resumes the manual intervention after it is pending for the specified timeout, or 60 days, whichever is earlier.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task in a release pipeline to pause an active deployment within a stage. This is typically executed to perform various manual steps or actions and then the automated deployment tasks are resumed.

ⓘ Note

This task can only be used in an **agentless job** and is intended for use in a classic release pipeline. This article refers to classic pipelines. For YAML usage, see [Manual Validation task](#).

The screenshot shows the Azure DevOps Pipeline Editor interface. The top navigation bar includes 'All pipelines > Fabrikam', 'Save', 'Release', and a three-dot menu. The main header has tabs for 'Pipeline', 'Tasks', 'Variables', 'Retention', 'Options', and 'History'. The 'Tasks' tab is currently selected. On the left, a list of stages is shown: 'Dev Deployment process', 'Run on agent' (with a 'Run on agent' icon), 'Deploy Azure App Service' (with a 'Deploy' icon), 'Agentless job' (with a 'Run on server' icon), and a 'Manual Intervention' task (highlighted with a red border). The 'Manual Intervention' task details are displayed on the right, including its version (8.*), display name ('Manual Intervention'), and instructions: 'Ready to deploy to \$(Release.EnvironmentName) for customer \${customerName}. Please contact customer to confirm deployment requirements have been met.'

The **Manual Intervention** task does not perform deployment actions directly. Instead, it allows you to pause an active deployment within a stage, typically to perform various manual steps or actions, and then the automated deployment tasks are resumed. For example, the user may need to edit the details of the current release before continuing (perhaps by entering the values for custom variables used by the tasks in the release).

The **Manual Intervention** task configuration includes an **Instructions** parameter that is used to provide related information or to specify the manual steps the user executes during the agentless job. You can configure the task to send email notifications to users and user groups when it is awaiting intervention and specify the automatic response (reject or resume the deployment) after a configurable timeout occurs.

 **Note**

You can use built-in and custom variables to generate portions of your instructions.

When the Manual Intervention task is activated during a deployment, it sets the deployment state to **IN PROGRESS**. A message bar is displayed with a link that opens the Manual Intervention dialog, which contains the instructions. After carrying out the manual steps, the administrator or user can choose to resume the deployment or reject it. Users with **Manage deployment** permission on the stage can resume or reject the manual intervention.

For more information about using this task, see [Approvals and gates overview](#).

Requirements

Requirement	Description
Pipeline types	Classic release
Runs on	Server
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Deploy

ManualValidation@0 - Manual validation v0 task

Article • 09/26/2023

Use this task to pause a YAML pipeline run to wait for manual interaction.

Syntax

YAML

```
# Manual validation v0
# [PREVIEW] Pause a pipeline run to wait for manual interaction. Works only
with YAML pipelines.
- task: ManualValidation@0
  inputs:
    notifyUsers: # string. Required. Notify users.
    #instructions: # string. Instructions.
    #onTimeout: 'reject' # 'reject' | 'resume'. On timeout. Default: reject.
```

Inputs

`notifyUsers` - Notify users

`string`. Required.

Sends a manual validation pending email to specific users (or groups). Only users with queue build permission can act on a manual validation. You can send an email to a group using the `[org name]\group name` syntax.

This task input is required, but you can specify an empty string if you don't want to notify anyone, for example during a test run: `notifyUsers: ''`.

`instructions` - Instructions

`string`.

Specifies the instructions that are shown to the user when resuming or rejecting the manual intervention. Based on these instructions, the user will make an informed decision about this manual intervention.

`onTimeout` - On timeout

`string`. Allowed values: `reject`, `resume`. Default value: `reject`.

Automatically rejects or resumes this manual validation after it is pending for the specified timeout, or 30 days, whichever is earlier.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task in a YAML pipeline to pause a run within a stage. This is typically executed to perform various manual steps or actions and then the run is resumed or rejected.

Important

This task is only supported in YAML pipelines and can only be used in an [agentless job](#) of a YAML pipeline.

The **Manual Validation** task allows you to pause a pipeline run within a stage, typically to perform some manual steps or actions, and then continue with the pipeline. For example, the user may need to manually validate certain deployment configurations before the pipeline starts a long running computational intensive job.

The **Manual Validation** task configuration includes an **instructions** parameter that is used to provide related information or to specify the manual steps the user executes during the pause. You can configure the task to send email notifications to users and user groups when it is awaiting a review and specify the automatic response (reject or resume) after a configurable timeout occurs.

You can specify the timeout value for the task using the optional `timeoutInMinutes` parameter, available in the common task properties.

ⓘ Note

For the task to run completely, the timeout value of the job should be higher than the timeout value of the task. See [default job timeout values](#).

💡 Tip

You can use variables to specify email addresses in the `notifyUsers` parameter.

When the Manual Validation task is activated during a pipeline, it displays a message bar with a link that opens the Manual validation dialog, which contains the instructions. After carrying out the manual steps, the administrator or user can choose to resume the run or reject it. Users with **Queue builds** permission on the pipeline can resume or reject the run.

Examples

YAML

```
jobs:
- job: waitForValidation
  displayName: Wait for external validation
  pool: server
  timeoutInMinutes: 4320 # job times out in 3 days
  steps:
  - task: ManualValidation@0
    timeoutInMinutes: 1440 # task times out in 1 day
    inputs:
      notifyUsers: |
        test@test.com
        example@example.com
      instructions: 'Please validate the build configuration and resume'
      onTimeout: 'resume'
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Server
Demands	None

Requirement	Description
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Deploy

Maven@4 - Maven v4 task

Article • 09/26/2023

Build, test, and deploy with Apache Maven.

Syntax

YAML

```
# Maven v4
# Build, test, and deploy with Apache Maven.
- task: Maven@4
  inputs:
    mavenPOMFile: 'pom.xml' # string. Required. Maven POM file. Default: pom.xml.
    #goals: 'package' # string. Goal(s). Default: package.
    #options: # string. Options.
    # JUnit Test Results
    #publishJUnitResults: true # boolean. Publish to Azure Pipelines.
    Default: true.
    testResultsFiles: '**/surefire-reports/TEST-*.xml' # string. Required
when publishJUnitResults = true. Test results files. Default: **/surefire-
reports/TEST-*.xml.
    #testRunTitle: # string. Optional. Use when publishJUnitResults = true.
Test run title.
    #allowBrokenSymlinks: true # boolean. Alias: allowBrokenSymbolicLinks.
Optional. Use when publishJUnitResults = true. Allow broken symbolic links.
Default: true.
    # Code Coverage
    #codeCoverageToolOption: 'None' # 'None' | 'Cobertura' | 'JaCoCo'.
    Alias: codeCoverageTool. Code coverage tool. Default: None.
    #codeCoverageClassFilter: # string. Alias: classFilter. Optional. Use
when codeCoverageTool != None. Class inclusion/exclusion filters.
    #codeCoverageClassDirectories: # string. Alias:
    classDirectories. Optional. Use when codeCoverageTool = JaCoCo. Class
files directories.
    #codeCoverageSourceDirectories: # string. Alias: srcDirectories.
Optional. Use when codeCoverageTool = JaCoCo. Source files directories.
    #codeCoverageFailIfEmpty: false # boolean. Alias: failIfCoverageEmpty.
Optional. Use when codeCoverageTool != None. Fail when code coverage results
are missing. Default: false.
    #codeCoverageRestoreOriginalPomXml: false # boolean. Alias:
    restoreOriginalPomXml. Optional. Use when codeCoverageTool != None. Restore
original pom.xml after task execution. Default: false.
    # Advanced
    javaHomeOption: 'JDKVersion' # 'JDKVersion' | 'Path'. Alias:
    javaHomeSelection. Required. Set JAVA_HOME by. Default: JDKVersion.
    #jdkVersionOption: 'default' # 'default' | '1.17' | '1.11' | '1.10' |
    '1.9' | '1.8' | '1.7' | '1.6'. Alias: jdkVersion. Optional. Use when
    javaHomeSelection = JDKVersion. JDK version. Default: default.
```

```

    #jdkDirectory: # string. Alias: jdkUserInputPath. Required when
    javaHomeSelection = Path. JDK path.
    #jdkArchitectureOption: 'x64' # 'x86' | 'x64'. Alias: jdkArchitecture.
    Optional. Use when jdkVersion != default. JDK architecture. Default: x64.
    mavenVersionOption: 'Default' # 'Default' | 'Path'. Alias:
    mavenVersionSelection. Required. Maven version. Default: Default.
    #mavenDirectory: # string. Alias: mavenPath. Required when
    mavenVersionSelection = Path. Maven path.
    #mavenSetM2Home: false # boolean. Optional. Use when
    mavenVersionSelection = Path. Set M2_HOME variable. Default: false.
    #mavenOptions: '-Xmx1024m' # string. Alias: mavenOpts. Set MAVEN_OPTS
    to. Default: -Xmx1024m.
    #mavenAuthenticateFeed: false # boolean. Alias: mavenFeedAuthenticate.
    Authenticate with Artifacts feeds. Default: false.
    #effectivePomSkip: false # boolean. Alias: skipEffectivePom. Skip
    generating effective POM while authenticating with Artifacts feeds. Default:
    false.

    # Code Analysis
    #sonarQubeRunAnalysis: false # boolean. Alias: sqAnalysisEnabled. Run
    SonarQube or SonarCloud analysis. Default: false.
    #isJacocoCoverageReportXML: false # boolean. Optional. Use when
    sqAnalysisEnabled = true && codeCoverageTool = JaCoCo. Use XML Jacoco
    reports for SonarQube analysis. Default: false.
    #sqMavenPluginVersionChoice: 'latest' # 'latest' | 'pom'. Required when
    sqAnalysisEnabled = true. SonarQube scanner for Maven version. Default:
    latest.
    #checkStyleRunAnalysis: false # boolean. Alias:
    checkstyleAnalysisEnabled. Run Checkstyle. Default: false.
    #pmdRunAnalysis: false # boolean. Alias: pmdAnalysisEnabled. Run PMD.
    Default: false.
    #findBugsRunAnalysis: false # boolean. Alias: findbugsAnalysisEnabled.
    Run FindBugs. Default: false.
    #spotBugsRunAnalysis: false # boolean. Alias: spotBugsAnalysisEnabled.
    Run SpotBugs analysis. Default: false.
    #spotBugsVersion: '4.5.3.0' # string. Alias: spotBugsMavenPluginVersion.
    Optional. Use when spotBugsAnalysisEnabled = true. Version number. Default:
    4.5.3.0.
    #spotBugsGoal: 'spotbugs' # 'spotbugs' | 'check'. Optional. Use when
    spotBugsAnalysisEnabled = true. The goal for the spotbugs plugin. Default:
    spotbugs.
    #failWhenBugsFound: true # boolean. Alias: spotBugsFailWhenBugsFound |
    sbFailWhenBugsFound. Optional. Use when spotBugsAnalysisEnabled = true &&
    spotBugsGoal = check. Fail when bugs are found with spotbugs:check. Default:
    true.

```

Inputs

mavenPOMFile - Maven POM file

`string`. Required. Default value: `pom.xml`.

Specifies the relative path from the repository root to the Maven POM file. See [Introduction to the POM](#) for more information.

goals - Goal(s)

`string`. Default value: `package`.

(Optional) Set to `package` to compile your code and package it into a .war file. If you leave this argument blank, the build will fail. See [Introduction to the Maven build lifecycle](#) for more information.

options - Options

`string`.

(Optional) Specifies any Maven command-line options you want to use.

publishJUnitResults - Publish to Azure Pipelines

`boolean`. Default value: `true`.

Specifies the option to publish the JUnit test results produced by the Maven build to Azure Pipelines. Each test results file matching `Test Results Files` will be published as a test run in Azure Pipelines.

testResultsFiles - Test results files

`string`. Required when `publishJUnitResults = true`. Default value: `**/surefire-reports/TEST-*.xml`.

Specifies the path and pattern of test results files to publish.

Wildcards can be used.

More information about [file matching patterns](#).

For example, `**/TEST-*.xml` for all XML files whose name starts with `TEST-`. If no root path is specified, files are matched beneath the default working directory, the value of which is available in the variable `$(System.DefaultWorkingDirectory)`. For example, a value of `**/TEST-*.xml` will actually result in matching files from `$(System.DefaultWorkingDirectory)**/TEST-*.xml`.

testRunTitle - Test run title

`string`. Optional. Use when `publishJUnitResults = true`.

Specifies a name for the test run.

`allowBrokenSymlinks` - Allow broken symbolic links

Input alias: `allowBrokenSymbolicLinks`. `boolean`. Optional. Use when `publishJUnitResults = true`. Default value: `true`.

If set to `false`, fails the build when the task finds a broken symbolic link while publishing tests result.

`codeCoverageToolOption` - Code coverage tool

Input alias: `codeCoverageTool`. `string`. Allowed values: `None`, `Cobertura`, `Jacoco`. Default value: `None`.

Specifies the code coverage tool. Enabling code coverage inserts the clean goal into the Maven goals list when Maven runs.

`codeCoverageClassFilter` - Class inclusion/exclusion filters

Input alias: `classFilter`. `string`. Optional. Use when `codeCoverageTool != None`.

Specifies a comma-separated list of filters to include or exclude classes from collecting code coverage. For example, `+:com.*,:org.*,-:my.app*.*`.

`codeCoverageClassFilesDirectories` - Class files directories

Input alias: `classFilesDirectories`. `string`. Optional. Use when `codeCoverageTool = JaCoCo`.

This field is required for a multi-module project.

Specifies a comma-separated list of relative paths from the Maven POM file to directories containing class files and archive files (JAR, WAR, etc.). Code coverage is reported for class files in these directories. For example,

`target/classes,target/testClasses`.

`codeCoverageSourceDirectories` - Source files directories

Input alias: `srcDirectories`. `string`. Optional. Use when `codeCoverageTool = JaCoCo`.

This field is required for a multi-module project.

Specifies a comma-separated list of relative paths from the Maven POM file to source

code directories. Code coverage reports use these to highlight source code. For example, `src/java,src/Test`.

codeCoverageFailIfEmpty - Fail when code coverage results are missing

Input alias: `failIfCoverageEmpty`. `boolean`. Optional. Use when `codeCoverageTool != None`. Default value: `false`.

Fails the build if code coverage did not produce any results to publish.

codeCoverageRestoreOriginalPomXml - Restore original pom.xml after task execution

Input alias: `restoreOriginalPomXml`. `boolean`. Optional. Use when `codeCoverageTool != None`. Default value: `false`.

Code coverage modifies `pom.xml` to produce results. Use this option if you need to keep the original `pom.xml`.

javaHomeOption - Set JAVA_HOME by

Input alias: `javaHomeSelection`. `string`. Required. Allowed values: `JDKVersion` (JDK Version), `Path`. Default value: `JDKVersion`.

Sets `JAVA_HOME` either by selecting a JDK version that will be discovered during builds or by manually entering a JDK path. If you already have Java installed on the agent machine, you can specify it by setting up `javaHomeOption` as `path` and `jdkDirectory` as a path to the JDK installed directory.

jdkVersionOption - JDK version

Input alias: `jdkVersion`. `string`. Optional. Use when `javaHomeSelection = JDKVersion`. Allowed values: `default`, `1.17` (JDK 17), `1.11` (JDK 11), `1.10` (JDK 10 (out of support)), `1.9` (JDK 9 (out of support)), `1.8` (JDK 8), `1.7` (JDK 7), `1.6` (JDK 6 (out of support)). Default value: `default`.

Attempts to discover the path to the selected JDK version and sets `JAVA_HOME` accordingly.

Note: If running on an agent that is not hosted by Microsoft, and the requested Java version is not the one indicated by the `JAVA_HOME` variable set on the agent machine, the task will rely on the variable `JAVA_HOME_{version}_{arch}` (for example: `JAVA_HOME_8_X64`) to locate the necessary JDK. Ensure this variable is set on self-hosted agents for any

version and architecture of the JDK that may be requested by this parameter and/or by `jdkArchitecture`.

`jdkDirectory` - JDK path

Input alias: `jdkUserInputPath`. `string`. Required when `javaHomeSelection = Path`.

Sets `JAVA_HOME` to the given path.

`jdkArchitectureOption` - JDK architecture

Input alias: `jdkArchitecture`. `string`. Optional. Use when `jdkVersion != default`.

Allowed values: `x86`, `x64`. Default value: `x64`.

Supplies the architecture (`x86`, `x64`) of the JDK.

`mavenVersionOption` - Maven version

Input alias: `mavenVersionSelection`. `string`. Required. Allowed values: `Default`, `Path` (Custom Path). Default value: `Default`.

Specifies either the default Maven version or the version in the specified custom path.

`mavenDirectory` - Maven path

Input alias: `mavenPath`. `string`. Required when `mavenVersionSelection = Path`.

Supplies the custom path to the Maven installation (for example: `/usr/share/maven`).

`mavenSetM2Home` - Set `M2_HOME` variable

`boolean`. Optional. Use when `mavenVersionSelection = Path`. Default value: `false`.

Sets the `M2_HOME` variable to a custom Maven installation path.

`mavenOptions` - Set `MAVEN_OPTS` to

Input alias: `mavenOpts`. `string`. Default value: `-Xmx1024m`.

Sets the `MAVEN_OPTS` environment variable, which is used to send command-line arguments to start the JVM. The `-Xmx` flag specifies the maximum memory available to the JVM.

`mavenAuthenticateFeed` - Authenticate with Artifacts feeds

Input alias: `mavenFeedAuthenticate`. `boolean`. Default value: `false`.

Automatically authenticates with Azure Artifacts feeds. If Artifacts feeds are not in use, deselect this option for faster builds.

`effectivePomSkip` - Skip generating effective POM while authenticating with Artifacts feeds

Input alias: `skipEffectivePom`. `boolean`. Default value: `false`.

Authenticates with Artifacts feeds using the POM only.

`sonarQubeRunAnalysis` - Run SonarQube or SonarCloud analysis

Input alias: `sqAnalysisEnabled`. `boolean`. Default value: `false`.

This option has changed from using version 1 of the **Maven** task to using the [SonarQube](#) and [SonarCloud](#) marketplace extensions.

Enable this option to run [SonarQube or SonarCloud analysis](#) after executing goals in the **Goals** field. The `install` or `package` goal should run first. Before this Maven task, you must also add a **Prepare Analysis Configuration** task from one of the extensions to the build pipeline.

`isJacocoCoverageReportXML` - Use XML Jacoco reports for SonarQube analysis

`boolean`. Optional. Use when `sqAnalysisEnabled = true && codeCoverageTool = JaCoCo`. Default value: `false`.

Uses XML Jacoco reports for SonarQube analysis. Learn more about [test reports](#).

`sqMavenPluginVersionChoice` - SonarQube scanner for Maven version

`string`. Required when `sqAnalysisEnabled = true`. Allowed values: `latest` (Use latest release), `pom` (Use version declared in your pom.xml). Default value: `latest`.

Specifies the SonarQube Maven plugin version to use. You can use the latest version or rely on the version in your `pom.xml`.

`checkStyleRunAnalysis` - Run Checkstyle

Input alias: `checkstyleAnalysisEnabled`. `boolean`. Default value: `false`.

Runs the Checkstyle tool with the default Sun checks. If no Checkstyle configuration is specified in the `pom.xml` file, default Sun checks are used. Results are uploaded as build artifacts.

`pmdRunAnalysis` - Run PMD

Input alias: `pmdAnalysisEnabled`. `boolean`. Default value: `false`.

Uses the PMD static analysis tool to look for bugs in the code. Results are uploaded as build artifacts.

`findBugsRunAnalysis` - Run FindBugs

Input alias: `findbugsAnalysisEnabled`. `boolean`. Default value: `false`.

Uses the FindBugs static analysis tool to look for bugs in the code. Results are uploaded as build artifacts.

`spotBugsRunAnalysis` - Run SpotBugs analysis

Input alias: `spotBugsAnalysisEnabled`. `boolean`. Default value: `false`.

Enable this option to run the SpotBugs code analysis plugin. More information about the [SpotBugs Maven plugin](#).

`spotBugsVersion` - Version number

Input alias: `spotBugsMavenPluginVersion`. `string`. Optional. Use when `spotBugsAnalysisEnabled = true`. Default value: `4.5.3.0`.

Learn about [the available versions of SpotBugs](#).

`spotBugsGoal` - The goal for the spotbugs plugin

`string`. Optional. Use when `spotBugsAnalysisEnabled = true`. Allowed values: `spotbugs` ("spotbugs" - Creates a report on found bugs), `check` ("check" - Pipeline fails if bugs were detected). Default value: `spotbugs`.

Specifies the goal of the plugin. Learn more about [SpotBugs goals](#).

`failWhenBugsFound` - Fail when bugs are found with spotbugs:check

Input alias: `spotBugsFailWhenBugsFound | sbFailWhenBugsFound`. `boolean`. Optional. Use

when `spotBugsAnalysisEnabled = true && spotBugsGoal = check`. Default value: `true`.

Fails when bugs are found if **Check Goal** is specified. Learn more about [SpotBug parameter details](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Configuration of the SonarQube analysis was moved to the [SonarQube](#) or [SonarCloud](#) extensions in the task **Prepare Analysis Configuration**.

ⓘ Important

When using the `-q` option in your `MAVEN_OPTS`, an effective pom won't be generated correctly, and Azure Artifacts feeds may not be able to be authenticated.

ⓘ Important

If the JDK version you want to use is already installed on your agent, set `javaHomeOption` to `path` and set the `jdkDirectory` to the path of the JDK version. These options set the `JAVA_HOME_11_X64` environment variable, which is required by the Maven task. This environment variable is set automatically if you are using the Java Tool installer task.

FAQ

I have a multi-module project, but my build is failing. What should I check?

Make sure you have specified `#codeCoverageClassFilesDirectories` and `#codeCoverageSourceDirectories` as a task input. These two parameters are optional for a single module project but are required for multi-module projects.

Examples

- [Build and Deploy your Java application to an Azure Web App.](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: maven
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.89.0 or greater
Task category	Build

See also

- [Maven authenticate](#)
- [Publish Maven artifacts with Azure Pipelines](#)
- [Java Tool Installer](#)
- [Build Java apps](#)

Maven@3 - Maven v3 task

Article • 09/26/2023

Use this task to build, test, and deploy with Apache Maven.

Syntax

YAML

```
# Maven v3
# Build, test, and deploy with Apache Maven.
- task: Maven@3
  inputs:
    mavenPOMFile: 'pom.xml' # string. Required. Maven POM file. Default: pom.xml.
    #goals: 'package' # string. Goal(s). Default: package.
    #options: # string. Options.
    # JUnit Test Results
    #publishJUnitResults: true # boolean. Publish to Azure Pipelines.
    Default: true.
    testResultsFiles: '**/surefire-reports/TEST-*.xml' # string. Required
when publishJUnitResults = true. Test results files. Default: **/surefire-
reports/TEST-*.xml.
    #testRunTitle: # string. Optional. Use when publishJUnitResults = true.
Test run title.
    #allowBrokenSymlinks: true # boolean. Alias: allowBrokenSymbolicLinks.
Optional. Use when publishJUnitResults = true. Allow broken symbolic links.
Default: true.
    # Code Coverage
    #codeCoverageToolOption: 'None' # 'None' | 'Cobertura' | 'JaCoCo'.
    Alias: codeCoverageTool. Code coverage tool. Default: None.
    #codeCoverageClassFilter: # string. Alias: classFilter. Optional. Use
when codeCoverageTool != None. Class inclusion/exclusion filters.
    #codeCoverageClassDirectories: # string. Alias:
    classDirectories. Optional. Use when codeCoverageTool = JaCoCo. Class
files directories.
    #codeCoverageSourceDirectories: # string. Alias: srcDirectories.
Optional. Use when codeCoverageTool = JaCoCo. Source files directories.
    #codeCoverageFailIfEmpty: false # boolean. Alias: failIfCoverageEmpty.
Optional. Use when codeCoverageTool != None. Fail when code coverage results
are missing. Default: false.
    #codeCoverageRestoreOriginalPomXml: false # boolean. Alias:
    restoreOriginalPomXml. Optional. Use when codeCoverageTool != None. Restore
original pom.xml after task execution. Default: false.
    # Advanced
    javaHomeOption: 'JDKVersion' # 'JDKVersion' | 'Path'. Alias:
    javaHomeSelection. Required. Set JAVA_HOME by. Default: JDKVersion.
    #jdkVersionOption: 'default' # 'default' | '1.17' | '1.11' | '1.10' |
    '1.9' | '1.8' | '1.7' | '1.6'. Alias: jdkVersion. Optional. Use when
    javaHomeSelection = JDKVersion. JDK version. Default: default.
```

```

    #jdkDirectory: # string. Alias: jdkUserInputPath. Required when
    javaHomeSelection = Path. JDK path.
    #jdkArchitectureOption: 'x64' # 'x86' | 'x64'. Alias: jdkArchitecture.
    Optional. Use when jdkVersion != default. JDK architecture. Default: x64.
    mavenVersionOption: 'Default' # 'Default' | 'Path'. Alias:
    mavenVersionSelection. Required. Maven version. Default: Default.
    #mavenDirectory: # string. Alias: mavenPath. Required when
    mavenVersionSelection = Path. Maven path.
    #mavenSetM2Home: false # boolean. Optional. Use when
    mavenVersionSelection = Path. Set M2_HOME variable. Default: false.
    #mavenOptions: '-Xmx1024m' # string. Alias: mavenOpts. Set MAVEN_OPTS
    to. Default: -Xmx1024m.
    #mavenAuthenticateFeed: false # boolean. Alias: mavenFeedAuthenticate.
    Authenticate with Artifacts feeds. Default: false.
    #effectivePomSkip: false # boolean. Alias: skipEffectivePom. Skip
    generating effective POM while authenticating with Artifacts feeds. Default:
    false.

    # Code Analysis
    #sonarQubeRunAnalysis: false # boolean. Alias: sqAnalysisEnabled. Run
    SonarQube or SonarCloud analysis. Default: false.
    #isJacocoCoverageReportXML: false # boolean. Optional. Use when
    sqAnalysisEnabled = true && codeCoverageTool = JaCoCo. Use XML Jacoco
    reports for SonarQube analysis. Default: false.
    #sqMavenPluginVersionChoice: 'latest' # 'latest' | 'pom'. Required when
    sqAnalysisEnabled = true. SonarQube scanner for Maven version. Default:
    latest.
    #checkStyleRunAnalysis: false # boolean. Alias:
    checkstyleAnalysisEnabled. Run Checkstyle. Default: false.
    #pmdRunAnalysis: false # boolean. Alias: pmdAnalysisEnabled. Run PMD.
    Default: false.
    #findBugsRunAnalysis: false # boolean. Alias: findbugsAnalysisEnabled.
    Run FindBugs. Default: false.
    #spotBugsRunAnalysis: false # boolean. Alias: spotBugsAnalysisEnabled.
    Run SpotBugs analysis. Default: false.
    #spotBugsVersion: '4.5.3.0' # string. Alias: spotBugsMavenPluginVersion.
    Optional. Use when spotBugsAnalysisEnabled = true. Version number. Default:
    4.5.3.0.
    #spotBugsGoal: 'spotbugs' # 'spotbugs' | 'check'. Optional. Use when
    spotBugsAnalysisEnabled = true. The goal for the spotbugs plugin. Default:
    spotbugs.
    #failWhenBugsFound: true # boolean. Alias: spotBugsFailWhenBugsFound |
    sbFailWhenBugsFound. Optional. Use when spotBugsAnalysisEnabled = true &&
    spotBugsGoal = check. Fail when bugs are found with spotbugs:check. Default:
    true.

```

Inputs

mavenPOMFile - Maven POM file

`string`. Required. Default value: `pom.xml`.

Specifies the relative path from the repository root to the Maven POM file. See [Introduction to the POM](#) for more information.

goals - Goal(s)

`string`. Default value: `package`.

(Optional) Set to `package` to compile your code and package it into a .war file. If you leave this argument blank, the build will fail. See [Introduction to the Maven build lifecycle](#) for more information.

options - Options

`string`.

(Optional) Specifies any Maven command-line options you want to use.

publishJUnitResults - Publish to Azure Pipelines

`boolean`. Default value: `true`.

Specifies the option to publish the JUnit test results produced by the Maven build to Azure Pipelines. Each test results file matching `Test Results Files` will be published as a test run in Azure Pipelines.

testResultsFiles - Test results files

`string`. Required when `publishJUnitResults = true`. Default value: `**/surefire-reports/TEST-*.xml`.

Specifies the path and pattern of test results files to publish.

Wildcards can be used.

More information about [file matching patterns](#).

For example, `**/TEST-*.xml` for all XML files whose name starts with `TEST-`. If no root path is specified, files are matched beneath the default working directory, the value of which is available in the variable `$(System.DefaultWorkingDirectory)`. For example, a value of `**/TEST-*.xml` will actually result in matching files from `$(System.DefaultWorkingDirectory)**/TEST-*.xml`.

testRunTitle - Test run title

`string`. Optional. Use when `publishJUnitResults = true`.

Specifies a name for the test run.

`allowBrokenSymlinks` - Allow broken symbolic links

Input alias: `allowBrokenSymbolicLinks`. `boolean`. Optional. Use when `publishJUnitResults = true`. Default value: `true`.

If set to `false`, fails the build when the task finds a broken symbolic link while publishing tests result.

`codeCoverageToolOption` - Code coverage tool

Input alias: `codeCoverageTool`. `string`. Allowed values: `None`, `Cobertura`, `Jacoco`. Default value: `None`.

Specifies the code coverage tool. Enabling code coverage inserts the clean goal into the Maven goals list when Maven runs.

`codeCoverageClassFilter` - Class inclusion/exclusion filters

Input alias: `classFilter`. `string`. Optional. Use when `codeCoverageTool != None`.

Specifies a comma-separated list of filters to include or exclude classes from collecting code coverage. For example, `+:com.*,:org.*,-:my.app*.*`.

`codeCoverageClassFilesDirectories` - Class files directories

Input alias: `classFilesDirectories`. `string`. Optional. Use when `codeCoverageTool = JaCoCo`.

This field is required for a multi-module project.

Specifies a comma-separated list of relative paths from the Maven POM file to directories containing class files and archive files (JAR, WAR, etc.). Code coverage is reported for class files in these directories. For example,

`target/classes,target/testClasses`.

`codeCoverageSourceDirectories` - Source files directories

Input alias: `srcDirectories`. `string`. Optional. Use when `codeCoverageTool = JaCoCo`.

This field is required for a multi-module project.

Specifies a comma-separated list of relative paths from the Maven POM file to source

code directories. Code coverage reports use these to highlight source code. For example, `src/java,src/Test`.

codeCoverageFailIfEmpty - Fail when code coverage results are missing

Input alias: `failIfCoverageEmpty`. `boolean`. Optional. Use when `codeCoverageTool != None`. Default value: `false`.

Fails the build if code coverage did not produce any results to publish.

codeCoverageRestoreOriginalPomXml - Restore original pom.xml after task execution

Input alias: `restoreOriginalPomXml`. `boolean`. Optional. Use when `codeCoverageTool != None`. Default value: `false`.

Code coverage modifies `pom.xml` to produce results. Use this option if you need to keep the original `pom.xml`.

javaHomeOption - Set JAVA_HOME by

Input alias: `javaHomeSelection`. `string`. Required. Allowed values: `JDKVersion` (JDK Version), `Path`. Default value: `JDKVersion`.

Sets `JAVA_HOME` either by selecting a JDK version that will be discovered during builds or by manually entering a JDK path. If you already have Java installed on the agent machine, you can specify it by setting up `javaHomeOption` as `path` and `jdkDirectory` as a path to the JDK installed directory.

jdkVersionOption - JDK version

Input alias: `jdkVersion`. `string`. Optional. Use when `javaHomeSelection = JDKVersion`. Allowed values: `default`, `1.17` (JDK 17), `1.11` (JDK 11), `1.10` (JDK 10 (out of support)), `1.9` (JDK 9 (out of support)), `1.8` (JDK 8), `1.7` (JDK 7), `1.6` (JDK 6 (out of support)). Default value: `default`.

Attempts to discover the path to the selected JDK version and sets `JAVA_HOME` accordingly.

Note: If running on an agent that is not hosted by Microsoft, and the requested Java version is not the one indicated by the `JAVA_HOME` variable set on the agent machine, the task will rely on the variable `JAVA_HOME_{version}_{arch}` (for example: `JAVA_HOME_8_X64`) to locate the necessary JDK. Ensure this variable is set on self-hosted agents for any

version and architecture of the JDK that may be requested by this parameter and/or by `jdkArchitecture`.

`jdkDirectory` - JDK path

Input alias: `jdkUserInputPath`. `string`. Required when `javaHomeSelection = Path`.

Sets `JAVA_HOME` to the given path.

`jdkArchitectureOption` - JDK architecture

Input alias: `jdkArchitecture`. `string`. Optional. Use when `jdkVersion != default`.

Allowed values: `x86`, `x64`. Default value: `x64`.

Supplies the architecture (`x86`, `x64`) of the JDK.

`mavenVersionOption` - Maven version

Input alias: `mavenVersionSelection`. `string`. Required. Allowed values: `Default`, `Path` (Custom Path). Default value: `Default`.

Specifies either the default Maven version or the version in the specified custom path.

`mavenDirectory` - Maven path

Input alias: `mavenPath`. `string`. Required when `mavenVersionSelection = Path`.

Supplies the custom path to the Maven installation (for example: `/usr/share/maven`).

`mavenSetM2Home` - Set `M2_HOME` variable

`boolean`. Optional. Use when `mavenVersionSelection = Path`. Default value: `false`.

Sets the `M2_HOME` variable to a custom Maven installation path.

`mavenOptions` - Set `MAVEN_OPTS` to

Input alias: `mavenOpts`. `string`. Default value: `-Xmx1024m`.

Sets the `MAVEN_OPTS` environment variable, which is used to send command-line arguments to start the JVM. The `-Xmx` flag specifies the maximum memory available to the JVM.

`mavenAuthenticateFeed` - Authenticate with Artifacts feeds

Input alias: `mavenFeedAuthenticate`. `boolean`. Default value: `false`.

Automatically authenticates with Azure Artifacts feeds. If Artifacts feeds are not in use, deselect this option for faster builds.

`effectivePomSkip` - Skip generating effective POM while authenticating with Artifacts feeds

Input alias: `skipEffectivePom`. `boolean`. Default value: `false`.

Authenticates with Artifacts feeds using the POM only.

`sonarQubeRunAnalysis` - Run SonarQube or SonarCloud analysis

Input alias: `sqAnalysisEnabled`. `boolean`. Default value: `false`.

This option has changed from using version 1 of the **Maven** task to using the [SonarQube](#) and [SonarCloud](#) marketplace extensions.

Enable this option to run [SonarQube or SonarCloud analysis](#) after executing goals in the **Goals** field. The `install` or `package` goal should run first. Before this Maven task, you must also add a **Prepare Analysis Configuration** task from one of the extensions to the build pipeline.

`isJacocoCoverageReportXML` - Use XML Jacoco reports for SonarQube analysis

`boolean`. Optional. Use when `sqAnalysisEnabled = true && codeCoverageTool = JaCoCo`. Default value: `false`.

Uses XML Jacoco reports for SonarQube analysis. Learn more about [test reports](#).

`sqMavenPluginVersionChoice` - SonarQube scanner for Maven version

`string`. Required when `sqAnalysisEnabled = true`. Allowed values: `latest` (Use latest release), `pom` (Use version declared in your pom.xml). Default value: `latest`.

Specifies the SonarQube Maven plugin version to use. You can use the latest version or rely on the version in your `pom.xml`.

`checkStyleRunAnalysis` - Run Checkstyle

Input alias: `checkstyleAnalysisEnabled`. `boolean`. Default value: `false`.

Runs the Checkstyle tool with the default Sun checks. If no Checkstyle configuration is specified in the `pom.xml` file, default Sun checks are used. Results are uploaded as build artifacts.

`pmdRunAnalysis` - Run PMD

Input alias: `pmdAnalysisEnabled`. `boolean`. Default value: `false`.

Uses the PMD static analysis tool to look for bugs in the code. Results are uploaded as build artifacts.

`findBugsRunAnalysis` - Run FindBugs

Input alias: `findbugsAnalysisEnabled`. `boolean`. Default value: `false`.

Uses the FindBugs static analysis tool to look for bugs in the code. Results are uploaded as build artifacts.

`spotBugsRunAnalysis` - Run SpotBugs analysis

Input alias: `spotBugsAnalysisEnabled`. `boolean`. Default value: `false`.

Enable this option to run the SpotBugs code analysis plugin. More information about the [SpotBugs Maven plugin](#).

`spotBugsVersion` - Version number

Input alias: `spotBugsMavenPluginVersion`. `string`. Optional. Use when `spotBugsAnalysisEnabled = true`. Default value: `4.5.3.0`.

Learn about [the available versions of SpotBugs](#).

`spotBugsGoal` - The goal for the spotbugs plugin

`string`. Optional. Use when `spotBugsAnalysisEnabled = true`. Allowed values: `spotbugs` ("spotbugs" - Creates a report on found bugs), `check` ("check" - Pipeline fails if bugs were detected). Default value: `spotbugs`.

Specifies the goal of the plugin. Learn more about [SpotBugs goals](#).

`failWhenBugsFound` - Fail when bugs are found with spotbugs:check

Input alias: `spotBugsFailWhenBugsFound | sbFailWhenBugsFound`. `boolean`. Optional. Use

when `spotBugsAnalysisEnabled = true && spotBugsGoal = check`. Default value: `true`.

Fails when bugs are found if **Check Goal** is specified. Learn more about [SpotBug parameter details](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Configuration of the SonarQube analysis was moved to the [SonarQube](#) or [SonarCloud](#) extensions in the task **Prepare Analysis Configuration**.

ⓘ Important

When using the `-q` option in your `MAVEN_OPTS`, an effective pom won't be generated correctly, and Azure Artifacts feeds may not be able to be authenticated.

ⓘ Important

If the JDK version you want to use is already installed on your agent, set `javaHomeOption` to `path` and set the `jdkDirectory` to the path of the JDK version. These options set the `JAVA_HOME_11_X64` environment variable, which is required by the Maven task. This environment variable is set automatically if you are using the Java Tool installer task.

FAQ

I have a multi-module project, but my build is failing. What should I check?

Make sure you have specified `#codeCoverageClassFilesDirectories` and `#codeCoverageSourceDirectories` as a task input. These two parameters are optional for a single module project but are required for multi-module projects.

Examples

- [Build and Deploy your Java application to an Azure Web App.](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: maven
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.89.0 or greater
Task category	Build

See also

- [Maven authenticate](#)
- [Publish Maven artifacts with Azure Pipelines](#)
- [Java Tool Installer](#)
- [Build Java apps](#)

Maven@2 - Maven v2 task

Article • 09/26/2023

Use this task to build, test, and deploy with Apache Maven.

Syntax

YAML

```
# Maven v2
# Build, test, and deploy with Apache Maven.
- task: Maven@2
  inputs:
    mavenPOMFile: 'pom.xml' # string. Required. Maven POM file. Default: pom.xml.
    #goals: 'package' # string. Goal(s). Default: package.
    #options: # string. Options.
    # JUnit Test Results
    #publishJUnitResults: true # boolean. Publish to Azure Pipelines.
    Default: true.
    testResultsFiles: '**/TEST-*.xml' # string. Required when
    publishJUnitResults = true. Test results files. Default: **/TEST-*.xml.
    #testRunTitle: # string. Optional. Use when publishJUnitResults = true.
    Test run title.
    #allowBrokenSymlinks: true # boolean. Alias: allowBrokenSymbolicLinks.
    Optional. Use when publishJUnitResults = true. Allow broken symbolic links.
    Default: true.
    # Code Coverage
    #codeCoverageToolOption: 'None' # 'None' | 'Cobertura' | 'JaCoCo'.
    Alias: codeCoverageTool. Code coverage tool. Default: None.
    #codeCoverageClassFilter: # string. Alias: classFilter. Optional. Use
    when codeCoverageTool != None. Class inclusion/exclusion filters.
    #codeCoverageClassDirectories: # string. Alias:
    classDirectories. Optional. Use when codeCoverageTool = JaCoCo. Class
    files directories.
    #codeCoverageSourceDirectories: # string. Alias: srcDirectories.
    Optional. Use when codeCoverageTool = JaCoCo. Source files directories.
    #codeCoverageFailIfEmpty: false # boolean. Alias: failIfCoverageEmpty.
    Optional. Use when codeCoverageTool != None. Fail when code coverage results
    are missing. Default: false.
    #codeCoverageRestoreOriginalPomXml: false # boolean. Alias:
    restoreOriginalPomXml. Optional. Use when codeCoverageTool != None. Restore
    original pom.xml after task execution. Default: false.
    # Advanced
    javaHomeOption: 'JDKVersion' # 'JDKVersion' | 'Path'. Alias:
    javaHomeSelection. Required. Set JAVA_HOME by. Default: JDKVersion.
    #jdkVersionOption: 'default' # 'default' | '1.17' | '1.11' | '1.10' |
    '1.9' | '1.8' | '1.7' | '1.6'. Alias: jdkVersion. Optional. Use when
    javaHomeSelection = JDKVersion. JDK version. Default: default.
    #jdkDirectory: # string. Alias: jdkUserInputPath. Required when
```

```

javaHomeSelection = Path. JDK path.
  #jdkArchitectureOption: 'x64' # 'x86' | 'x64'. Alias: jdkArchitecture.
Optional. Use when jdkVersion != default. JDK architecture. Default: x64.
  mavenVersionOption: 'Default' # 'Default' | 'Path'. Alias:
mavenVersionSelection. Required. Maven version. Default: Default.
  #mavenDirectory: # string. Alias: mavenPath. Required when
mavenVersionSelection = Path. Maven path.
  #mavenSetM2Home: false # boolean. Optional. Use when
mavenVersionSelection = Path. Set M2_HOME variable. Default: false.
  #mavenOptions: '-Xmx1024m' # string. Alias: mavenOpts. Set MAVEN_OPTS
to. Default: -Xmx1024m.
  #mavenAuthenticateFeed: true # boolean. Alias: mavenFeedAuthenticate.
Authenticate with Artifacts feeds. Default: true.
  # Code Analysis
    #sonarQubeRunAnalysis: false # boolean. Alias: sqAnalysisEnabled. Run
SonarQube or SonarCloud analysis. Default: false.
    #isJacocoCoverageReportXML: false # boolean. Optional. Use when
sqAnalysisEnabled = true && codeCoverageTool = JaCoCo. Use XML Jacoco
reports for SonarQube analysis. Default: false.
    #sqMavenPluginVersionChoice: 'latest' # 'latest' | 'pom'. Required when
sqAnalysisEnabled = true. SonarQube scanner for Maven version. Default:
latest.
    #checkStyleRunAnalysis: false # boolean. Alias:
checkstyleAnalysisEnabled. Run Checkstyle. Default: false.
    #pmdRunAnalysis: false # boolean. Alias: pmdAnalysisEnabled. Run PMD.
Default: false.
    #findBugsRunAnalysis: false # boolean. Alias: findbugsAnalysisEnabled.
Run FindBugs. Default: false.

```

Inputs

mavenPOMFile - Maven POM file

`string`. Required. Default value: `pom.xml`.

Specifies the relative path from the repository root to the Maven POM file.

goals - Goal(s)

`string`. Default value: `package`.

options - Options

`string`.

publishJUnitResults - Publish to Azure Pipelines

`boolean`. Default value: `true`.

Specifies the option to publish the JUnit test results produced by the Maven build to Azure Pipelines. Each test results file matching `Test Results Files` will be published as a test run in Azure Pipelines.

`testResultsFiles` - Test results files

`string`. Required when `publishJUnitResults = true`. Default value: `**/TEST-*.xml`.

Specifies the path and pattern of test results files to publish. Wildcards can be used. More information about [file matching patterns](#).

For example, `**/TEST-*.xml` for all XML files whose name starts with `TEST-`.

If no root path is specified, files are matched beneath the default working directory, the value of which is available in the variable `$(System.DefaultWorkingDirectory)`. For example, a value of `**/TEST-*.xml` will actually result in matching files from `$(System.DefaultWorkingDirectory)/**/TEST-*.xml`.

`testRunTitle` - Test run title

`string`. Optional. Use when `publishJUnitResults = true`.

Specifies a name for the test run.

`allowBrokenSymlinks` - Allow broken symbolic links

Input alias: `allowBrokenSymbolicLinks`. `boolean`. Optional. Use when `publishJUnitResults = true`. Default value: `true`.

When set to `false`, fails the build when the task finds a broken symbolic link while publishing tests result.

`codeCoverageToolOption` - Code coverage tool

Input alias: `codeCoverageTool`. `string`. Allowed values: `None`, `Cobertura`, `Jacoco`. Default value: `None`.

Specifies the code coverage tool.

`codeCoverageClassFilter` - Class inclusion/exclusion filters

Input alias: `classFilter`. `string`. Optional. Use when `codeCoverageTool != None`.

Specifies a comma-separated list of filters to include or exclude classes from collecting code coverage. For example, `+:com.*,:org.*,-:my.app*.*`.

`codeCoverageClassFilesDirectories` - Class files directories

Input alias: `classFilesDirectories`. `string`. Optional. Use when `codeCoverageTool = JaCoCo`.

This field is required for a multi-module project.

Specifies a comma-separated list of relative paths from the Maven POM file to directories containing class files and archive files (JAR, WAR, etc.). Code coverage is reported for class files in these directories. For example,

`target/classes,target/testClasses`.

`codeCoverageSourceDirectories` - Source files directories

Input alias: `srcDirectories`. `string`. Optional. Use when `codeCoverageTool = JaCoCo`.

This field is required for a multi-module project.

Specifies a comma-separated list of relative paths from the Maven POM file to source code directories. Code coverage reports use these to highlight source code. For example, `src/java,src/Test`.

`codeCoverageFailIfEmpty` - Fail when code coverage results are missing

Input alias: `failIfCoverageEmpty`. `boolean`. Optional. Use when `codeCoverageTool != None`. Default value: `false`.

Fails the build if code coverage did not produce any results to publish.

`codeCoverageRestoreOriginalPomXml` - Restore original pom.xml after task execution

Input alias: `restoreOriginalPomXml`. `boolean`. Optional. Use when `codeCoverageTool != None`. Default value: `false`.

Code coverage modifies `pom.xml` to produce results. Use this option if you need to keep the original `pom.xml`.

`javaHomeOption` - Set JAVA_HOME by

Input alias: `javaHomeSelection`. `string`. Required. Allowed values: `JDKVersion` (JDK Version), `Path`. Default value: `JDKVersion`.

Sets `JAVA_HOME` either by selecting a JDK version that will be discovered during builds or by manually entering a JDK path.

`jdkVersionOption` - JDK version

Input alias: `jdkVersion`. `string`. Optional. Use when `javaHomeSelection = JDKVersion`. Allowed values: `default`, `1.17` (JDK 17), `1.11` (JDK 11), `1.10` (JDK 10 (out of support)), `1.9` (JDK 9 (out of support)), `1.8` (JDK 8), `1.7` (JDK 7), `1.6` (JDK 6 (out of support)). Default value: `default`.

Attempts to discover the path to the selected JDK version and sets `JAVA_HOME` accordingly.

`jdkDirectory` - JDK path

Input alias: `jdkUserInputPath`. `string`. Required when `javaHomeSelection = Path`.

Sets `JAVA_HOME` to the given path.

`jdkArchitectureOption` - JDK architecture

Input alias: `jdkArchitecture`. `string`. Optional. Use when `jdkVersion != default`. Allowed values: `x86`, `x64`. Default value: `x64`.

Supplies the architecture (`x86`, `x64`) of the JDK.

`mavenVersionOption` - Maven version

Input alias: `mavenVersionSelection`. `string`. Required. Allowed values: `Default`, `Path` (Custom Path). Default value: `Default`.

Uses either the default Maven version or the version in the specified custom path.

`mavenDirectory` - Maven path

Input alias: `mavenPath`. `string`. Required when `mavenVersionSelection = Path`.

Specifies the custom path to the Maven installation (for example: `/usr/share/maven`).

`mavenSetM2Home` - Set `M2_HOME` variable

`boolean`. Optional. Use when `mavenVersionSelection = Path`. Default value: `false`.

Sets the `M2_HOME` variable to a custom Maven installation path.

`mavenOptions` - Set MAVEN_OPTS to

Input alias: `mavenOpts`. `string`. Default value: `-Xmx1024m`.

Sets the `MAVEN_OPTS` environment variable, which is used to send command-line arguments to start the JVM. The `-Xmx` flag specifies the maximum memory available to the JVM.

`mavenAuthenticateFeed` - Authenticate with Artifacts feeds

Input alias: `mavenFeedAuthenticate`. `boolean`. Default value: `true`.

Automatically authenticates with Azure Artifacts feeds. If Artifacts feeds are not in use, deselect this option for faster builds.

`sonarQubeRunAnalysis` - Run SonarQube or SonarCloud analysis

Input alias: `sqAnalysisEnabled`. `boolean`. Default value: `false`.

This option has changed from using version 1 of the **Maven** task to using the [SonarQube](#) and [SonarCloud](#) marketplace extensions.

Enable this option to run [SonarQube or SonarCloud analysis](#) after executing goals in the **Goals** field. The **install** or **package** goal should run first. Before this Maven task, you must also add a **Prepare Analysis Configuration** task from one of the extensions to the build pipeline.

`isJacocoCoverageReportXML` - Use XML Jacoco reports for SonarQube analysis

`boolean`. Optional. Use when `sqAnalysisEnabled = true && codeCoverageTool = JaCoCo`. Default value: `false`.

Uses XML Jacoco reports for SonarQube analysis. Learn more about [test reports](#).

`sqMavenPluginVersionChoice` - SonarQube scanner for Maven version

`string`. Required when `sqAnalysisEnabled = true`. Allowed values: `latest` (Use latest release), `pom` (Use version declared in your pom.xml). Default value: `latest`.

Specifies the SonarQube Maven plugin version to use. You can use the latest version or rely on the version in your `pom.xml`.

`checkStyleRunAnalysis` - Run Checkstyle

Input alias: `checkstyleAnalysisEnabled`. `boolean`. Default value: `false`.

Runs the Checkstyle tool with the default Sun checks. Results are uploaded as build artifacts.

pmdRunAnalysis - Run PMD

Input alias: `pmdAnalysisEnabled`. `boolean`. Default value: `false`.

Uses the PMD static analysis tool to look for bugs in the code. Results are uploaded as build artifacts.

findBugsRunAnalysis - Run FindBugs

Input alias: `findbugsAnalysisEnabled`. `boolean`. Default value: `false`.

Uses the FindBugs static analysis tool to look for bugs in the code. Results are uploaded as build artifacts.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Configuration of the SonarQube analysis was moved to the [SonarQube](#) or [SonarCloud](#) extensions in the task [Prepare Analysis Configuration](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: maven

Requirement	Description
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.89.0 or greater
Task category	Build

See also

- [Maven authenticate](#)
- [Publish Maven artifacts with Azure Pipelines](#)
- [Java Tool Installer](#)
- [Build Java apps](#)

Maven@1 - Maven v1 task

Article • 09/26/2023

Use this task to build with Apache Maven.

Syntax

YAML

```
# Maven v1
# Build with Apache Maven.
- task: Maven@1
  inputs:
    mavenPOMFile: 'pom.xml' # string. Required. Maven POM file. Default: pom.xml.
    #goals: 'package' # string. Goal(s). Default: package.
    #options: # string. Options.
    # JUnit Test Results
    #publishJUnitResults: true # boolean. Publish to TFS/Team Services.
    Default: true.
    testResultsFiles: '**/TEST-*.xml' # string. Required when publishJUnitResults = true. Test Results Files. Default: **/TEST-*.xml.
    #testRunTitle: # string. Optional. Use when publishJUnitResults = true. Test Run Title.
    # Code Coverage
    #codeCoverageToolOption: 'None' # 'None' | 'Cobertura' | 'JaCoCo'.
    Alias: codeCoverageTool. Code Coverage Tool. Default: None.
    #codeCoverageClassFilter: # string. Alias: classFilter. Optional. Use when codeCoverageTool != None. Class Inclusion/Exclusion Filters.
    #codeCoverageClassFilesDirectories: # string. Alias: classFilesDirectories. Optional. Use when codeCoverageTool = JaCoCo. Class Files Directories.
    #codeCoverageSourceDirectories: # string. Alias: srcDirectories. Optional. Use when codeCoverageTool = JaCoCo. Source Files Directories.
    #codeCoverageFailIfEmpty: false # boolean. Alias: failIfCoverageEmpty. Optional. Use when codeCoverageTool != None. Fail When Code Coverage Results Are Missing. Default: false.
    # Advanced
    javaHomeOption: 'JDKVersion' # 'JDKVersion' | 'Path'. Alias: javaHomeSelection. Required. Set JAVA_HOME by. Default: JDKVersion.
    #jdkVersionOption: 'default' # 'default' | '1.9' | '1.8' | '1.7' | '1.6'. Alias: jdkVersion. Optional. Use when javaHomeSelection = JDKVersion. JDK Version. Default: default.
    #jdkDirectory: # string. Alias: jdkUserInputPath. Required when javaHomeSelection = Path. JDK Path.
    #jdkArchitectureOption: 'x64' # 'x86' | 'x64'. Alias: jdkArchitecture. Optional. Use when jdkVersion != default. JDK Architecture. Default: x64.
    mavenVersionOption: 'Default' # 'Default' | 'Path'. Alias: mavenVersionSelection. Required. Maven Version. Default: Default.
    #mavenDirectory: # string. Alias: mavenPath. Required when
```

```

mavenVersionSelection = Path. Maven Path.
  #mavenSetM2Home: false # boolean. Optional. Use when
  mavenVersionSelection = Path. Set M2_HOME variable. Default: false.
    #mavenOptions: '-Xmx1024m' # string. Alias: mavenOpts. Set MAVEN_OPTS
    to. Default: -Xmx1024m.
      #mavenAuthenticateFeed: true # boolean. Alias: mavenFeedAuthenticate.
      Authenticate built-in Maven feeds. Default: true.
        # Code Analysis
          #sonarQubeRunAnalysis: false # boolean. Alias: sqAnalysisEnabled. Run
          SonarQube Analysis. Default: false.
            #sonarQubeServiceEndpoint: # string. Alias: sqConnectedServiceName.
            Required when sqAnalysisEnabled = true. SonarQube Endpoint.
              #sonarQubeProjectName: # string. Alias: sqProjectName. Optional. Use
              when sqAnalysisEnabled = true. SonarQube Project Name.
                #sonarQubeProjectKey: # string. Alias: sqProjectKey. Optional. Use when
                sqAnalysisEnabled = true. SonarQube Project Key.
                  #sonarQubeProjectVersion: # string. Alias: sqProjectVersion. Optional.
                  Use when sqAnalysisEnabled = true. SonarQube Project Version.
                    #sonarQubeSpecifyDB: false # boolean. Alias: sqDbDetailsRequired.
                    Optional. Use when sqAnalysisEnabled = true. The SonarQube server version is
                    lower than 5.2. Default: false.
                      #sonarQubeDBUrl: # string. Alias: sqDbUrl. Optional. Use when
                      sqDbDetailsRequired = true. Db Connection String.
                        #sonarQubeDBUsername: # string. Alias: sqDbUsername. Optional. Use when
                        sqDbDetailsRequired = true. Db Username.
                          #sonarQubeDBPassword: # string. Alias: sqDbPassword. Optional. Use when
                          sqDbDetailsRequired = true. Db User Password.
                            #sonarQubeIncludeFullReport: true # boolean. Alias:
                            sqAnalysisIncludeFullReport. Optional. Use when sqAnalysisEnabled = true.
                            Include full analysis report in the build summary (SQ 5.3+). Default: true.
                              #sonarQubeFailWhenQualityGateFails: # boolean. Alias:
                              sqAnalysisBreakBuildIfQualityGateFailed. Optional. Use when
                              sqAnalysisEnabled = true. Fail the build on quality gate failure (SQ 5.3+).
                                #checkStyleRunAnalysis: false # boolean. Alias:
                                checkstyleAnalysisEnabled. Run Checkstyle. Default: false.
                                  #pmdRunAnalysis: false # boolean. Alias: pmdAnalysisEnabled. Run PMD.
                                  Default: false.
                                    #findBugsRunAnalysis: false # boolean. Alias: findbugsAnalysisEnabled.
                                    Run FindBugs. Default: false.

```

Inputs

mavenPOMFile - Maven POM file

`string`. Required. Default value: `pom.xml`.

Specifies the relative path from the repository root to the Maven POM file.

goals - Goal(s)

`string`. Default value: `package`.

`options` - Options

`string`.

`publishJUnitResults` - Publish to TFS/Team Services

`boolean`. Default value: `true`.

Specifies the option to publish the JUnit test results produced by the Maven build to TFS/Team Services. Each test results file matching `Test Results Files` will be published as a test run in TFS/Team Services.

`testResultsFiles` - Test Results Files

`string`. Required when `publishJUnitResults = true`. Default value: `**/TEST-*.xml`.

Specifies the path and pattern of the test results files to publish. For example, `**/TEST-*.xml` for all XML files with a name that starts with `TEST-`. If no root path is specified, files are matched beneath the default working directory, the value of which is available in the variable `$(System.DefaultWorkingDirectory)`. For example, a value of `**/TEST-* .xml` will actually result in matching files from `$(System.DefaultWorkingDirectory)/**/TEST-*.xml`.

`testRunTitle` - Test Run Title

`string`. Optional. Use when `publishJUnitResults = true`.

Specifies a name for the test run.

`codeCoverageToolOption` - Code Coverage Tool

Input alias: `codeCoverageTool`. `string`. Allowed values: `None`, `Cobertura`, `Jacoco`. Default value: `None`.

Specifies the code coverage tool.

`codeCoverageClassFilter` - Class Inclusion/Exclusion Filters

Input alias: `classFilter`. `string`. Optional. Use when `codeCoverageTool != None`.

Specifies a comma-separated list of filters to include or exclude classes from collecting code coverage. For example, `+:com.* , +:org.* , -:my.app*.*`.

`codeCoverageClassFilesDirectories` - Class Files Directories

Input alias: `classFilesDirectories`. `string`. Optional. Use when `codeCoverageTool = JaCoCo`.

This field is required for a multi-module project.

Specifies a comma-separated list of relative paths from the Maven POM file to the directories containing class files and archive files (JAR, WAR, etc.). Code coverage is reported for class files in these directories. For example,

`target/classes,target/testClasses`.

`codeCoverageSourceDirectories` - Source Files Directories

Input alias: `srcDirectories`. `string`. Optional. Use when `codeCoverageTool = JaCoCo`.

This field is required for a multi-module project.

Specifies a comma-separated list of relative paths from the Maven POM file to source code directories. Code coverage reports will use these to highlight source code. For example, `src/java,src/Test`.

`codeCoverageFailIfEmpty` - Fail When Code Coverage Results Are Missing

Input alias: `failIfCoverageEmpty`. `boolean`. Optional. Use when `codeCoverageTool != None`. Default value: `false`.

Fails the build if code coverage did not produce any results to publish.

`javaHomeOption` - Set JAVA_HOME by

Input alias: `javaHomeSelection`. `string`. Required. Allowed values: `JDKVersion` (JDK Version), `Path`. Default value: `JDKVersion`.

Sets `JAVA_HOME` either by selecting a JDK version that will be discovered during builds or by manually entering a JDK path.

`jdkVersionOption` - JDK Version

Input alias: `jdkVersion`. `string`. Optional. Use when `javaHomeSelection = JDKVersion`. Allowed values: `default`, `1.9` (JDK 9), `1.8` (JDK 8), `1.7` (JDK 7), `1.6` (JDK 6). Default value: `default`.

Attempts to discover the path to the selected JDK version, and sets `JAVA_HOME` accordingly.

`jdkDirectory` - JDK Path

Input alias: `jdkUserInputPath`. `string`. Required when `javaHomeSelection = Path`.

Sets `JAVA_HOME` to the given path.

`jdkArchitectureOption` - JDK Architecture

Input alias: `jdkArchitecture`. `string`. Optional. Use when `jdkVersion != default`.

Allowed values: `x86`, `x64`. Default value: `x64`.

Supplies the architecture (`x86`, `x64`) of the JDK.

`mavenVersionOption` - Maven Version

Input alias: `mavenVersionSelection`. `string`. Required. Allowed values: `Default`, `Path` (Custom Path). Default value: `Default`.

Uses either the default Maven version or the version in the specified custom path.

`mavenDirectory` - Maven Path

Input alias: `mavenPath`. `string`. Required when `mavenVersionSelection = Path`.

Supplies the custom path to the Maven installation (for example: `/usr/share/maven`).

`mavenSetM2Home` - Set `M2_HOME` variable

`boolean`. Optional. Use when `mavenVersionSelection = Path`. Default value: `false`.

Sets the `M2_HOME` variable to a custom Maven installation path.

`mavenOptions` - Set `MAVEN_OPTS` to

Input alias: `mavenOpts`. `string`. Default value: `-Xmx1024m`.

Sets the `MAVEN_OPTS` environment variable, which is used to send command-line arguments to start the JVM. The `-Xmx` flag specifies the maximum memory available to the JVM.

mavenAuthenticateFeed - Authenticate built-in Maven feeds

Input alias: `mavenFeedAuthenticate`. `boolean`. Default value: `true`.

Automatically authenticates with Azure Artifacts feeds. If Artifacts feeds are not in use, deselect this option for faster builds.

sonarQubeRunAnalysis - Run SonarQube Analysis

Input alias: `sqAnalysisEnabled`. `boolean`. Default value: `false`.

Runs a [SonarQube analysis](#) after executing the current goals. `install` or `package` goals should be executed first.

sonarQubeServiceEndpoint - SonarQube Endpoint

Input alias: `sqConnectedServiceName`. `string`. Required when `sqAnalysisEnabled = true`.

Specifies the SonarQube server generic endpoint.

sonarQubeProjectName - SonarQube Project Name

Input alias: `sqProjectName`. `string`. Optional. Use when `sqAnalysisEnabled = true`.

Specifies the SonarQube project name, for example `sonar.projectName`.

sonarQubeProjectKey - SonarQube Project Key

Input alias: `sqProjectKey`. `string`. Optional. Use when `sqAnalysisEnabled = true`.

Specifies the SonarQube project unique key, for example `sonar.projectKey`.

sonarQubeProjectVersion - SonarQube Project Version

Input alias: `sqProjectVersion`. `string`. Optional. Use when `sqAnalysisEnabled = true`.

Specifies the SonarQube project version, for example `sonar.projectVersion`.

sonarQubeSpecifyDB - The SonarQube server version is lower than 5.2

Input alias: `sqDbDetailsRequired`. `boolean`. Optional. Use when `sqAnalysisEnabled = true`. Default value: `false`.

If using a SonarQube server 5.1 or lower, you must specify the database connection details.

sonarQubeDBUrl - Db Connection String

Input alias: `sqDbUrl`. `string`. Optional. Use when `sqDbDetailsRequired = true`.

Use for SonarQube server 5.1 and lower only.

Specifies the database connection setting (for example, `sonar.jdbc.url` or

`jdbc:jtds:sqlserver://localhost/sonar;SelectMethod=Cursor`).

sonarQubeDBUsername - Db Username

Input alias: `sqDbUsername`. `string`. Optional. Use when `sqDbDetailsRequired = true`.

Use for SonarQube server 5.1 and lower only.

Specifies the username for the database user (for example, `sonar.jdbc.username`).

sonarQubeDBPassword - Db User Password

Input alias: `sqDbPassword`. `string`. Optional. Use when `sqDbDetailsRequired = true`.

Use for SonarQube server 5.1 and lower only.

Specifies the password for the database user (for example, `sonar.jdbc.password`).

sonarQubeIncludeFullReport - Include full analysis report in the build summary (SQ 5.3+)

Input alias: `sqAnalysisIncludeFullReport`. `boolean`. Optional. Use when

`sqAnalysisEnabled = true`. Default value: `true`.

This option will delay the build until the SonarQube analysis is completed.

sonarQubeFailWhenQualityGateFails - Fail the build on quality gate failure (SQ 5.3+)

Input alias: `sqAnalysisBreakBuildIfQualityGateFailed`. `boolean`. Optional. Use when

`sqAnalysisEnabled = true`.

This option is only available when using a SonarQube server 5.3 or above. Introduces delays, as the build must wait for SonarQube to complete the analysis. More information about [SonarQube quality gates](#).

checkStyleRunAnalysis - Run Checkstyle

Input alias: `checkstyleAnalysisEnabled`. `boolean`. Default value: `false`.

Runs the Checkstyle tool with the default Sun checks. Results are uploaded as build artifacts.

pmdRunAnalysis - Run PMD

Input alias: `pmdAnalysisEnabled`. `boolean`. Default value: `false`.

Uses the PMD static analysis tool to look for bugs in the code. Results are uploaded as build artifacts.

findBugsRunAnalysis - Run FindBugs

Input alias: `findbugsAnalysisEnabled`. `boolean`. Default value: `false`.

Uses the FindBugs static analysis tool to look for bugs in the code. Results are uploaded as build artifacts.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: maven
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any

Requirement	Description
Agent version	1.89.0 or greater
Task category	Build

See also

- [Maven authenticate](#)
- [Publish Maven artifacts with Azure Pipelines](#)
- [Java Tool Installer](#)
- [Build Java apps](#)

MavenAuthenticate@0 - Maven Authenticate v0 task

Article • 09/26/2023

Use this task to provide credentials for Azure Artifacts feeds and external Maven repositories.

Syntax

YAML

```
# Maven Authenticate v0
# Provides credentials for Azure Artifacts feeds and external maven
repositories.
- task: MavenAuthenticate@0
  inputs:
    #artifactsFeeds: # string. Feeds.
    #mavenServiceConnections: # string. Credentials for repositories outside
this organization/collection.
```

Inputs

`artifactsFeeds` - Feeds

`string`.

Specifies a comma-separated list of Azure Artifacts feed names to authenticate with Maven. If you only need authentication for external Maven repositories, leave this field blank.

`mavenServiceConnections` - Credentials for repositories outside this organization/collection

`string`.

Specifies a comma-separated list of [Maven service connection](#) names from external organizations to authenticate with Maven. If you only need authentication for Azure Artifacts feeds, leave this field blank.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Specifies the credentials for Azure Artifacts feeds and external Maven repositories in the current user's `settings.xml` file.

- Where is the `settings.xml` file containing the authenticated repositories located?
- We use the `mvn -s` switch to specify our own `settings.xml` file. How do we authenticate Azure Artifacts feeds there?
- My Pipeline needs to access a feed in a different project

Where is the `settings.xml` file containing the authenticated repositories located?

The Maven Authenticate task searches for the `settings.xml` file in the current user's home directory. For Linux and Mac, the path is `$HOME/.m2/settings.xml`. For Windows, the path is `%USERPROFILE%\.m2\settings.xml`. If the `settings.xml` file doesn't exist, a new one will be created at that path.

We use the `mvn -s` switch to specify our own `settings.xml` file. How do we authenticate Azure Artifacts feeds there?

The Maven Authenticate task doesn't have access to the custom `settings.xml` file that's specified by using an `-s` switch. To add Azure Artifacts authentication to your custom `settings.xml`, add a `server` element inside your `settings.xml` file:

XML

```
<server>
  <id>feedName</id> <!-- Set this to the id of the <repository> element
  inside your pom.xml file. -->
  <username>AzureDevOps</username>
```

```
<password>${env.SYSTEM_ACESSTOKEN}</password>
</server>
```

The access token variable can be set in your pipelines using these [instructions](#).

My Pipeline needs to access a feed in a different project

If the pipeline is running in a different project than the project hosting the feed, you must set up the other project to grant read/write access to the build service. See [Package permissions in Azure Pipelines](#) for more details.

Examples

- [Authenticate Maven feeds inside your organization](#)
- [Authenticate Maven feeds outside your organization](#)

Authenticate Maven feeds inside your organization

In this example, we authenticate two Azure Artifacts feeds within our organization.

Task definition

YAML

```
- task: MavenAuthenticate@0
  displayName: 'Maven Authenticate'
  inputs:
    artifactsFeeds: MyFeedInOrg1,MyFeedInOrg2
```

The `MavenAuthenticate` task updates the `settings.xml` file present in the agent user's `.m2` directory located at `{user.home}/.m2/settings.xml` to add two entries inside the `<servers>` element.

settings.xml

XML

```
<servers>
  <server>
    <id>MyFeedInOrg1</id>
    <username>AzureDevOps</username>
    <password>****</password>
```

```
</server>
<server>
  <id>MyFeedInOrg2</id>
  <username>AzureDevOps</username>
  <password>****</password>
</server>
</servers>
```

To correctly authenticate the task, set the repositories in your project's `pom.xml` to the same `<id>` as the name specified in the task for Maven.

pom.xml

Project scoped feed

XML

```
<repository>
  <id>MyFeedInOrg1</id>

  <url>https://pkgs.dev.azure.com/OrganizationName/ProjectName/_packaging/MyProjectScopedFeed1/Maven/v1</url>
    <releases>
      <enabled>true</enabled>
    </releases>
    <snapshots>
      <enabled>true</enabled>
    </snapshots>
  </repository>
```

Organization scoped feed

XML

```
<repository>
  <id>MyFeedInOrg1</id>

  <url>https://pkgs.dev.azure.com/OrganizationName/_packaging/MyOrgScopedFeed1/Maven/v1</url>
    <releases>
      <enabled>true</enabled>
    </releases>
    <snapshots>
      <enabled>true</enabled>
    </snapshots>
  </repository>
```

The Artifacts feed URL may or may not contain the project. A URL for a project-scoped feed must contain the project, and a URL for an organization-scoped feed must not contain the project. Learn more about [project-scoped feeds](#).

Authenticate Maven feeds outside your organization

In this example, we authenticate two external Maven repositories.

Task definition

YAML

```
- task: MavenAuthenticate@0
  displayName: 'Maven Authenticate'
  inputs:
    MavenServiceConnections: central,MavenOrg
```

The `MavenAuthenticate` task updates the `settings.xml` file present in the agent users' `.m2` directory located at `{user.home}/.m2/settings.xml` to add two entries inside the `<servers>` element.

settings.xml

XML

```
<servers>
  <server>
    <id>central</id>
    <username>centralUsername</username>
    <password>****</password>
  </server>
  <server>
    <id>MavenOrg</id>
    <username>mavenOrgUsername</username>
    <password>****</password>
  </server>
</servers>
```

To correctly authenticate the task, set the repositories in your project's `pom.xml` to the same `<id>` as the name specified in the task for Maven.

pom.xml

XML

```
<repository>
  <id>central</id>
  <url>https://repo1.maven.org/maven2/</url>
  <releases>
    <enabled>true</enabled>
  </releases>
  <snapshots>
    <enabled>true</enabled>
  </snapshots>
</repository>
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Package

VSMobileCenterTest@0 - Mobile Center Test v0 task

Article • 09/26/2023

Use this task to test mobile app packages with Visual Studio Mobile Center.

Syntax

YAML

```
# Mobile Center Test v0
# Test mobile app packages with Visual Studio Mobile Center.
- task: VSMobileCenterTest@0
  inputs:
    app: # string. Required. Binary Application File Path.
    artifactsDir: '$(Build.ArtifactStagingDirectory)/MobileCenterTest' # string. Required. Artifacts Directory. Default: $(Build.ArtifactStagingDirectory)/MobileCenterTest.
    # Prepare Tests
    #enablePrepare: true # boolean. Prepare Tests. Default: true.
    framework: 'appium' # 'appium' | 'espresso' | 'calabash' | 'uitest' | 'xcuitest'. Required when enablePrepare = true. Test Framework. Default: appium.
    #appiumBuildDir: # string. Required when enablePrepare = true && framework = appium. Build Directory.
    #espressoBuildDir: # string. Optional. Use when enablePrepare = true && framework = espresso. Build Directory.
    #espressoTestApkPath: # string. Optional. Use when enablePrepare = true && framework = espresso. Test APK Path.
    #calabashProjectDir: # string. Required when enablePrepare = true && framework = calabash. Project Directory.
    #calabashConfigFile: # string. Optional. Use when enablePrepare = true && framework = calabash. Cucumber Config File.
    #calabashProfile: # string. Optional. Use when enablePrepare = true && framework = calabash. Profile to run.
    #calabashSkipConfigCheck: false # boolean. Optional. Use when enablePrepare = true && framework = calabash. Skip Configuration Check. Default: false.
    #uitestBuildDir: # string. Required when enablePrepare = true && framework = uitest. Build Directory.
    #uitestStoreFile: # string. Optional. Use when enablePrepare = true && framework = uitest. Store File.
    #uitestStorePass: # string. Optional. Use when enablePrepare = true && framework = uitest. Store Password.
    #uitestKeyAlias: # string. Optional. Use when enablePrepare = true && framework = uitest. Key Alias.
    #uitestKeyPass: # string. Optional. Use when enablePrepare = true && framework = uitest. Key Password.
    #uitestToolsDir: # string. Optional. Use when enablePrepare = true &&
```

```

framework = uitest. Test Tools Directory.
  #signInfo: # string. Optional. Use when framework = calabash ||
framework = uitest. Signing Information.
  #xcuitestBuildDir: # string. Optional. Use when enablePrepare = true &&
framework = xcuitest. Build Directory.
  #xcuitestTestIPAPath: # string. Optional. Use when enablePrepare = true
&& framework = xcuitest. Test IPA Path.
  #prepareOpts: # string. Optional. Use when enablePrepare = true.

Additional Options.
  # Run Tests
  #enableRun: true # boolean. Run Tests. Default: true.
  credsType: 'serviceEndpoint' # 'serviceEndpoint' | 'inputs'. Required
when enableRun = true. Authentication Method. Default: serviceEndpoint.
  #serverEndpoint: # string. Required when enableRun = true && credsType =
serviceEndpoint. Mobile Center Connection.
  #username: # string. Required when enableRun = true && credsType =
inputs. Mobile Center Username.
  #password: # string. Required when enableRun = true && credsType =
inputs. Mobile Center Password.
  appSlug: # string. Required when enableRun = true. App Slug.
  devices: # string. Required when enableRun = true. Devices.
  #series: 'master' # string. Optional. Use when enableRun = true. Test
Series. Default: master.
  #dsymDir: # string. Optional. Use when enableRun = true. dSYM Directory.
  locale: 'en_US' # 'da_DK' | 'nl_NL' | 'en_GB' | 'en_US' | 'fr_FR' |
'de_DE' | 'ja_JP' | 'ru_RU' | 'es_MX' | 'es_ES' | 'user'. Required when
enableRun = true. System Language. Default: en_US.
  #userDefinedLocale: # string. Optional. Use when enableRun = true &&
locale = user. Other Locale.
  #loginOpts: # string. Optional. Use when enableRun = true && credsType =
inputs. Addtional Options for Login.
  #runOpts: # string. Optional. Use when enableRun = true. Additional
Options for Run.
  #async: false # boolean. Optional. Use when enableRun = true. Do not
wait for test result. Default: false.
  # Advanced
  #cliLocationOverride: # string. mobile-center CLI Location.
  #debug: false # boolean. Enable Debug Output. Default: false.

```

Inputs

app - Binary Application File Path

string. Required.

Specifies the relative path from the repo root to the .APK or .IPA file you want to test.

artifactsDir - Artifacts Directory

string. Required. Default value: \$(Build.ArtifactStagingDirectory)/MobileCenterTest.

Specifies the directory to place the artifacts that are produced by the prepare step and used by the run step. The directory is created if it does not exist.

enablePrepare - Prepare Tests

`boolean`. Default value: `true`.

If set to `true`, prepares tests.

framework - Test Framework

`string`. Required when `enablePrepare = true`. Allowed values: `appium`, `espresso`, `calabash`, `uitest` (Xamarin UI Test), `xcuitest`. Default value: `appium`.

Specifies the test framework that the task will use.

appiumBuildDir - Build Directory

`string`. Required when `enablePrepare = true && framework = appium`.

Specifies the path to the directory that contains Appium tests.

espressoBuildDir - Build Directory

`string`. Optional. Use when `enablePrepare = true && framework = espresso`.

Specifies the path for the Espresso output directory.

espressoTestApkPath - Test APK Path

`string`. Optional. Use when `enablePrepare = true && framework = espresso`.

Specifies the path to the APK file with Espresso tests. If a value is not set, `build-dir` is used to find the APK file. Wildcards are allowed.

calabashProjectDir - Project Directory

`string`. Required when `enablePrepare = true && framework = calabash`.

Specifies the path for the Calabash workspace directory.

calabashConfigFile - Cucumber Config File

`string`. Optional. Use when `enablePrepare = true && framework = calabash`.

Specifies the file path to the Cucumber configuration file, which is usually `cucumber.yml`.

calabashProfile - Profile to run

`string`. Optional. Use when `enablePrepare = true && framework = calabash`.

Specifies the profile to run. This value must exist in the Cucumber configuration file.

calabashSkipConfigCheck - Skip Configuration Check

`boolean`. Optional. Use when `enablePrepare = true && framework = calabash`. Default value: `false`.

Forces the task to run without a Cucumber profile.

uitestBuildDir - Build Directory

`string`. Required when `enablePrepare = true && framework = uitest`.

Specifies the path to the directory with built test assemblies.

uitestStoreFile - Store File

`string`. Optional. Use when `enablePrepare = true && framework = uitest`.

Specifies the path to the store file.

uitestStorePass - Store Password

`string`. Optional. Use when `enablePrepare = true && framework = uitest`.

Specifies the password for the store file. Use a new variable with its lock enabled on the Variables tab to encrypt this value.

uitestKeyAlias - Key Alias

`string`. Optional. Use when `enablePrepare = true && framework = uitest`.

Specifies the alias that identifies the public/private key pair used in the store file.

uitestKeyPass - Key Password

`string`. Optional. Use when `enablePrepare = true && framework = uitest`.

Specifies the key password for the alias and store file. Use a new variable with its lock enabled on the Variables tab to encrypt this value.

uitestToolsDir - Test Tools Directory

`string`. Optional. Use when `enablePrepare = true && framework = uitest`.

Specifies the path to the directory with Xamarin UI test tools that contains `test-cloud.exe`.

signInfo - Signing Information

`string`. Optional. Use when `framework = calabash || framework = uitest`.

Uses signing information to sign the test server.

xcuitestBuildDir - Build Directory

`string`. Optional. Use when `enablePrepare = true && framework = xcuitest`.

Specifies the path to the build output directory, which is usually
`$(ProjectDir)/Build/Products/Debug-iphoneos`.

xcuitestTestIpaPath - Test IPA Path

`string`. Optional. Use when `enablePrepare = true && framework = xcuitest`.

Specifies the path to the `*.ipa` file with the XCUI Test tests.

prepareOpts - Additional Options

`string`. Optional. Use when `enablePrepare = true`.

Specifies additional arguments to pass to `mobile-center test prepare` step.

enableRun - Run Tests

`boolean`. Default value: `true`.

credsType - Authentication Method

`string`. Required when `enableRun = true`. Allowed values: `serviceEndpoint` (Mobile Center Connection), `inputs` (Credentials). Default value: `serviceEndpoint`.

Specifies the authentication method. Use a Mobile Center service endpoint connection, or specify credentials to connect to Visual Studio Mobile Center.

serverEndpoint - Mobile Center Connection

`string`. Required when `enableRun = true && credsType = serviceEndpoint`.

Specifies the service endpoint for your Visual Studio Mobile Center connection. To create one, click the **Manage link** and create a new service endpoint.

username - Mobile Center Username

`string`. Required when `enableRun = true && credsType = inputs`.

Visit [Azure Mobile Center](#) to set your username.

password - Mobile Center Password

`string`. Required when `enableRun = true && credsType = inputs`.

Visit [Azure Mobile Center](#) to set your password. This string can accept a variable defined in build/release definitions as `$(passwordVariable)`. You may mark the variable type as `secret` to secure it.

appSlug - App Slug

`string`. Required when `enableRun = true`.

The app slug is in the format of `{username}/{app_identifier}`. To locate `{username}` and `{app_identifier}` for an app, find the app's listing on [Azure Mobile Apps](#). The URL is in the format of `https://mobile.azure.com/users/{username}/apps/{app_identifier}`.

devices - Devices

`string`. Required when `enableRun = true`.

Identifies what devices this test will run against. Copy and paste this string when you define a new test run from Mobile Center Test beacon.

series - Test Series

`string`. Optional. Use when `enableRun = true`. Default value: `master`.

Specifies the series name for organizing test runs (e.g. `master`, `production`, `beta`).

dsymDir - dSYM Directory

`string`. Optional. Use when `enableRun = true`.

Specifies the path to the dSYM directory, which contains iOS symbol files.

locale - System Language

`string`. Required when `enableRun = true`. Allowed values: `da_DK` (Danish (Denmark)), `nl_NL` (Dutch (Netherlands)), `en_GB` (English (United Kingdom)), `en_us` (English (United States)), `fr_FR` (French (France)), `de_DE` (German (Germany)), `ja_JP` (Japanese (Japan)), `ru_RU` (Russian (Russia)), `es_MX` (Spanish (Mexico)), `es_ES` (Spanish (Spain)), `user` (Other). Default value: `en_US`.

If your language isn't displayed, specify **Other** and enter its locale, such as `en_US`.

userDefinedLocale - Other Locale

`string`. Optional. Use when `enableRun = true && locale = user`.

Specifies any two-letter ISO-639 language code, along with any two-letter ISO 3166 country code, in the format `[language]_[country]`, such as `en_US`.

loginOpts - Additional Options for Login

`string`. Optional. Use when `enableRun = true && credsType = inputs`.

Specifies additional arguments that are passed to `mobile-center login` step.

runOpts - Additional Options for Run

`string`. Optional. Use when `enableRun = true`.

Specifies additional arguments that are passed to `mobile-center test run`.

async - Do not wait for test result

`boolean`. Optional. Use when `enableRun = true`. Default value: `false`.

When set to `true`, executes commands asynchronously and exits when tests are uploaded without waiting for the test results.

`cliLocationOverride` - mobile-center CLI Location

`string`.

Specifies the path to the `mobile-center` command-line interface (CLI).

`debug` - Enable Debug Output

`boolean`. Default value: `false`.

Adds `--debug` to the `mobile-center` command-line interface (CLI).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Test

MSBuild@1 - MSBuild v1 task

Article • 09/26/2023

Use this task to build with MSBuild.

Syntax

YAML

```
# MSBuild v1
# Build with MSBuild.
- task: MSBuild@1
  inputs:
    solution: '**/*.sln' # string. Required. Project. Default: **/*.sln.
    #msbuildLocationMethod: 'version' # 'version' | 'location'. MSBuild.
    Default: version.
    #msbuildVersion: 'latest' # 'latest' | '17.0' | '16.0' | '15.0' | '14.0'
    | '12.0' | '4.0'. Optional. Use when msbuildLocationMethod = version.
    MSBuild Version. Default: latest.
    #msbuildArchitecture: 'x86' # 'x86' | 'x64'. Optional. Use when
    msbuildLocationMethod = version. MSBuild Architecture. Default: x86.
    #msbuildLocation: # string. Optional. Use when msbuildLocationMethod =
    location. Path to MSBuild.
    #platform: # string. Platform.
    #configuration: # string. Configuration.
    #msbuildArguments: # string. MSBuild Arguments.
    #clean: false # boolean. Clean. Default: false.
    # Advanced
    #maximumCpuCount: false # boolean. Build in Parallel. Default: false.
    #restoreNugetPackages: false # boolean. Restore NuGet Packages. Default:
    false.
    #logProjectEvents: false # boolean. Record Project Details. Default:
    false.
    #createLogFile: false # boolean. Create Log File. Default: false.
    #logFileVerbosity: 'normal' # 'quiet' | 'minimal' | 'normal' |
    'detailed' | 'diagnostic'. Optional. Use when createLogFile = true. Log File
    Verbosity. Default: normal.
```

Inputs

`solution` - Project

`string`. Required. Default value: `**/*.sln`.

If you want to build multiple projects, specify search criteria. You can use a single-folder wildcard (*) and recursive wildcards (**). For example, `**.*proj` searches for all MSBuild project (`.*proj`) files in all subdirectories.

Make sure the projects you specify are downloaded by this build pipeline. On the Repository tab:

- If you use TFVC, make sure that the project is a child of one of the mappings on the Repository tab.
- If you use Git, make sure that the project or project is in your Git repo, in a branch that you're building.

Tip

If you are building a solution, we recommend you use the [Visual Studio build task](#) instead of the MSBuild task.

`msbuildLocationMethod` - MSBuild

`string`. Allowed values: `version`, `location` (Specify Location). Default value: `version`.

`msbuildVersion` - MSBuild Version

`string`. Optional. Use when `msbuildLocationMethod = version`. Allowed values: `latest`, `17.0` (MSBuild 17.0), `16.0` (MSBuild 16.0), `15.0` (MSBuild 15.0), `14.0` (MSBuild 14.0), `12.0` (MSBuild 12.0), `4.0` (MSBuild 4.0). Default value: `latest`.

If the preferred version cannot be found, the latest version found is used instead. On an macOS agent, `xbuild` (Mono) is used if version is lower than `15.0`.

`msbuildArchitecture` - MSBuild Architecture

`string`. Optional. Use when `msbuildLocationMethod = version`. Allowed values: `x86` (MSBuild x86), `x64` (MSBuild x64). Default value: `x86`.

Supplies the MSBuild architecture (x86, x64) to run.

`msbuildLocation` - Path to MSBuild

`string`. Optional. Use when `msbuildLocationMethod = location`.

Supplies the path to MSBuild.

`platform` - Platform

`string`.

💡 Tip

- If you are targeting an MSBuild project (*.proj) file instead of a solution, specify `AnyCPU` (no whitespace).
- Declare a build variable such as `BuildPlatform` on the Variables tab (selecting `Allow` at Queue Time) and reference it here as `$(BuildPlatform)`. This way you can modify the platform when you queue the build and enable building multiple configurations.

`configuration` - Configuration

`string`.

💡 Tip

Declare a build variable such as `BuildConfiguration` on the Variables tab (selecting `Allow` at Queue Time) and reference it here as `$(BuildConfiguration)`. This way you can modify the platform when you queue the build and enable building multiple configurations.

`msbuildArguments` - MSBuild Arguments

`string`.

Specifies additional arguments passed to MSBuild (on Windows) and xbuild (on macOS).

`clean` - Clean

`boolean`. Default value: `false`.

Set to `False` if you want to make this an incremental build. This setting might reduce your build time, especially if your codebase is large. This option has no practical effect unless you also set the `Clean` repository to `False`. Set to `True` if you want to rebuild all the code in the code projects. This is equivalent to the MSBuild `/target:clean` argument. For more information, see [repo options](#)

`maximumCpuCount` - Build in Parallel

`boolean`. Default value: `false`.

If your MSBuild target configuration is compatible with building in parallel, you can check this input to pass the `/m` switch to MSBuild (Windows only). If your target configuration is not compatible with building in parallel, checking this option may cause your build to result in `file-in-use` errors, or intermittent or inconsistent build failures.

`restoreNugetPackages` - Restore NuGet Packages

`boolean`. Default value: `false`.

This option is deprecated. To restore NuGet packages, add a [NuGet](#) task before the build.

`logProjectEvents` - Record Project Details

`boolean`. Default value: `false`.

Optionally records timeline details for each project (Windows only).

`createLogFile` - Create Log File

`boolean`. Default value: `false`.

Optionally creates a log file (Windows only).

`logFileVerbosity` - Log File Verbosity

`string`. Optional. Use when `createLogFile = true`. Allowed values: `quiet`, `minimal`, `normal`, `detailed`, `diagnostic`. Default value: `normal`.

Specifies log file verbosity.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Should I use the Visual Studio Build task or the MSBuild task?

If you are building a solution, in most cases you should use the [Visual Studio Build task](#). This task automatically:

- Sets the `/p:VisualStudioVersion` property for you. This forces MSBuild to use a particular set of targets that increase the likelihood of a successful build.
- Specifies the MSBuild version argument.

In some cases, you might need to use the `MSBuild` task. For example, you should use it if you are building code projects apart from a solution.

Where can I learn more about MSBuild?

[MSBuild reference](#)

[MSBuild command-line reference](#)

How do I build multiple configurations for multiple platforms?

1. On the Variables tab, make sure you have variables defined for your configurations and platforms. To specify multiple values, separate them with commas. For example:

- For a .NET app, you could specify `BuildConfiguration` with debug and release values, and you could specify `BuildPlatform` with any CPU value.
- For a C++ app, you could specify `BuildConfiguration` with debug and release values, and you could specify `BuildPlatform` with any x86 and x64 values.

2. On the Options tab, select `MultiConfiguration` and specify the `Multipliers`, separated by commas. For example: `BuildConfiguration, BuildPlatform Select Parallel` if you want to distribute the jobs (one for each combination of values) to multiple agents in parallel if they are available.

3. On the Build tab, select this step and specify the `Platform` and `Configuration` arguments. For example:

- Platform: `$(BuildPlatform)`
- Configuration: `$(BuildConfiguration)`

Can I build TFSBuild.proj files?

You cannot build `TFSBuild.proj` files. These kinds of files are generated by [TFS 2005](#) and [TFS 2008](#). These files contain tasks, and targets are supported only using [XAML builds](#).

Troubleshooting

This section provides troubleshooting tips for common issues that a user might encounter when using the `MSBuild` task.

- Build failed with the following error: An internal failure occurred while running MSBuild

Build failed with the following error: An internal failure occurred while running MSBuild

- Possible causes
- Troubleshooting suggestions

Possible causes

- Change in the MSBuild version.
- Issues with a third-party extension.
- New updates to Visual Studio that can cause missing assemblies on the build agent.
- Moved or deleted some of the necessary NuGet packages.

Troubleshooting suggestions

- Run the pipeline with diagnostics to retrieve detailed logs
- Try to reproduce the error locally
- What else can I do?

Run the pipeline with diagnostics to retrieve detailed logs

One of the available options to diagnose the issue is to take a look at the generated logs. You can view your pipeline logs by selecting the appropriate task and job in your pipeline run summary.

To get the logs of your pipeline execution [Get logs to diagnose problems](#)

You can also setup and download a customized verbose log to assist with your troubleshooting:

- [Configure verbose logs](#)
- [View and download logs](#)

In addition to the pipeline diagnostic logs, you can also check these other types of logs that contain more information to help you debug and solve the problem:

- [Worker diagnostic logs](#)
- [Agent diagnostic logs](#)
- [Other logs](#) (Environment and capabilities)

Try to reproduce the error locally

If you are using a hosted build agent, you might want to try to reproduce the error locally. This will help you to narrow down whether the failure is the result of the build agent or the build task.

Run the same `MSBuild` command on your local machine using the same arguments. Check out [MSBuild command](#) for reference.

Tip

If you can reproduce the problem on your local machine, then your next step is to investigate the [MSBuild](#) issue.

Learn more about [Microsoft hosted agents](#).

To setup your own self-hosted agent and run the build jobs:

- [Self-hosted Windows agents](#)
- [Self-hosted Linux agents](#)
- [Self-hosted macOS agents](#)

What else can I do?

Some of the MSBuild errors are caused by a change in Visual Studio so you can search on [Visual Studio Developer Community](#) to see if this issue has been reported. We also welcome your questions, suggestions, and feedback.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: msbuild
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.95.0 or greater
Task category	Build

See also

- [Visual Studio Build task](#)

MysqlDeploymentOnMachineGroup@1

- MySQL database deploy v1 task

Article • 09/26/2023

Use this task to run your scripts and make changes to your MySQL Database. There are two ways to deploy: using a script file or writing the script in our inline editor.

ⓘ Note

This is an early preview version. Since this task is server based, it appears on Deployment group jobs.

Syntax

YAML

```
# This task is supported on classic release pipelines only.  
# Use the classic designer to add and configure this task in a classic  
release pipeline.  
# See the following Inputs section for details on the inputs that this task  
supports.
```

Inputs

`TaskNameSelector` - Deploy MySql Using

`string`. Allowed values: `SqlTaskFile` (MySQL Script File), `InlineSqlTask` (Inline MySQL Script). Default value: `SqlTaskFile`.

Specifies either Script File or Inline Script.

`SqlFile` - MySQL Script

`string`. Required when `TaskNameSelector = SqlTaskFile`.

Specifies the full path of the script file on the automation agent or on a UNC path that is accessible to the automation agent, such as `BudgetIT\DeployBuilds\script.sql`. This string can also use predefined system variables, such as `$(agent.releaseDirectory)` and a file containing SQL statements.

SqlInline - Inline MySQL Script

`string`. Required when `TaskNameSelector = InlineSqlTask`.

Specifies the MySQL script to execute on the selected database.

ServerName - Host Name

`string`. Required. Default value: `localhost`.

Specifies the server name of `Database for MySQL`, such as `localhost`. This string is the same value that is used for `Hostname` in `Parameters` in MySQL Workbench.

DatabaseName - Database Name

`string`.

Specifies the name of the database. The script will create a database name if one does not already exist.

SqlUsername - MySQL User Name

`string`. Required.

This string is the same value that is used for `Username` in `Parameters` in MySQL Workbench.

SqlPassword - Password

`string`. Required.

Specifies the password for MySQL Database. The password can be a variable defined in the pipeline, such as `$(password)`, and may be marked as `secret` to secure it.

SqlAdditionalArguments - Additional Arguments

`string`.

Specifies the additional options that are supported by MySQL simple SQL shell. These options will be applied when executing the given file on the Database for MySQL.

Example: You can change to the default tab separated output format, HTML format, or XML format. If you have problems due to insufficient memory for large result sets, use the `--quick` option.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to run your scripts and make changes to your MySQL Database. There are two ways to deploy, either using a script file or writing the script in our inline editor.

ⓘ Note

This is an early preview version. Since this task is server-based, it appears on deployment group jobs.

Prerequisites

- MySQL Client in agent box

The task expects MySQL client must be in agent box.

- **Windows Agent:** Use this [script file](#) to install MySQL client
- **Linux Agent:** Run command 'apt-get install mysql-client' to install MySQL client

Examples

This example creates a sample db in MySQL.

YAML

```
steps:  
- task: MysqlDeploymentOnMachineGroup@1  
  displayName: 'Deploy Using : InlineSqlTask'  
  inputs:  
    TaskNameSelector: InlineSqlTask  
    SqlInline: |  
      CREATE DATABASE IF NOT EXISTS alm;  
      use alm;
```

```
ServerName: localhost  
SqlUsername: root  
SqlPassword: P2ssw0rd
```

Requirements

Requirement	Description
Pipeline types	Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.100.0 or greater
Task category	Deploy

NodeTaskRunnerInstaller@0 - Node.js tasks runner installer v0 task

Article • 09/26/2023

Install specific Node.js version to run node tasks.

Syntax

YAML

```
# Node.js tasks runner installer v0
# Install specific Node.js version to run node tasks.
- task: NodeTaskRunnerInstaller@0
  inputs:
    nodeVersion: '6' # '6' | '10'. Alias: runnerVersion | installVersion.
    Required. Version of runner to install. Default: 6.
```

Inputs

nodeVersion - Version of runner to install

Input alias: `runnerVersion | installVersion`. `string`. Required. Allowed values: `6` (Node.js 6.17.1), `10` (Node.js 10.24.1). Default value: `6`.

Select the node version to install.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

When adopting [agent releases that exclude the Node 6 task runner](#) you may have an occasional need to run tasks that have not been updated to use a newer Node runner. For this scenario we provide a way to still use tasks dependent on Node End-of-Life runners. For more information, see Node runner guidance [blog post](#).

The following task example shows how to install the Node 6 runner just-in-time, so an older task can successfully run.

YAML

```
steps:
- task: NodeTaskRunnerInstaller@0
  inputs:
    runnerVersion: 6
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Utility

NodeTool@0 - Node.js tool installer v0 task

Article • 09/26/2023

Use this task to find, download, and cache a specified version of [Node.js](#) and add it to the PATH.

Syntax

YAML

```
# Node.js tool installer v0
# Finds or downloads and caches the specified version spec of Node.js and
# adds it to the PATH.
- task: NodeTool@0
  inputs:
    versionSource: 'spec' # 'spec' | 'fromFile'. Required. Source of
    version. Default: spec.
    #versionSpec: '6.x' # string. Optional. Use when versionSource = spec.
    Version Spec. Default: 6.x.
    #versionFilePath: # string. Optional. Use when versionSource = fromFile.
    Path to the .nvmrc file.
    #checkLatest: false # boolean. Check for Latest Version. Default: false.
    #force32bit: false # boolean. Use 32 bit version on x64 agents. Default:
    false.
    # Advanced
    #nodejsMirror: 'https://nodejs.org/dist' # string. Set source for
    Node.js binaries. Default: https://nodejs.org/dist.
```

Inputs

`versionSource` - Source of version

`string`. Required. Allowed values: `spec` (Specify Node version), `fromFile` (Get version from file). Default value: `spec`.

`versionSpec` - Version Spec

`string`. Optional. Use when `versionSource = spec`. Default value: `6.x`.

Specifies the version spec of the version to get. Examples: `6.x`, `4.x`, `6.10.0`, `>=6.10.0`.

`versionFilePath` - Path to the .nvmrc file

`string`. Optional. Use when `versionSource = fromFile`.

File path to get version. Example: src/.nvmrc.

`checkLatest` - Check for Latest Version

`boolean`. Default value: `false`.

Specifies the agent to check for the latest available version that satisfies the version spec. For example, you select this option because you run this build on your [self-hosted agent](#), and you want to always use the latest `6.x` version.

💡 Tip

If you're using [the Microsoft-hosted agents](#), you should leave this set to `false`.

Microsoft updates the Microsoft-hosted agents on a regular basis, but they're often slightly behind the latest version. Enabling this parameter could result in your build spending a lot of time updating to a newer minor version.

`force32bit` - Use 32 bit version on x64 agents

`boolean`. Default value: `false`.

Installs the `x86` version of Node regardless of the CPU architecture of the agent.

`nodejsMirror` - Set source for Node.js binaries

`string`. Default value: `https://nodejs.org/dist`.

Use an alternative installation mirror when sourcing the Node.js binaries.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: Node, npm, node.js
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : PATH
Agent version	2.182.1 or greater
Task category	Tool

See also

For an explanation of tool installers and examples, see [Tool installers](#).

Npm@1 - npm v1 task

Article • 09/26/2023

Use this task to install and publish npm packages or to run an `npm` command. Supports [npmjs.com](#) and authenticated registries like Azure Artifacts.

ⓘ Note

The **npm Authenticate task** is the recommended way to authenticate with Azure Artifacts. This task no longer takes new features and only critical bugs are addressed.

Syntax

YAML

```
# npm v1
# Install and publish npm packages, or run an npm command. Supports
npmjs.com and authenticated registries like Azure Artifacts.
- task: Npm@1
  inputs:
    command: 'install' # 'ci' | 'install' | 'publish' | 'custom'. Required.
    Command. Default: install.
    #workingDir: # string. Working folder that contains package.json.
    #customCommand: # string. Required when command = custom. Command and
    arguments.
    # Advanced
    #verbose: # boolean. Optional. Use when command = install || command =
    ci || command = publish. Verbose logging.
    #publishPackageMetadata: true # boolean. Optional. Use when command =
    publish && publishRegistry = useFeed && command = install || command = ci || |
    command = publish. Publish pipeline metadata. Default: true.
    # Custom registries and authentication
    #customRegistry: 'useNpmrc' # 'useNpmrc' | 'useFeed'. Optional. Use when
    command = install || command = ci || command = custom. Registries to use.
    Default: useNpmrc.
    #customFeed: # string. Required when customRegistry = useFeed && command
    = install || command = ci || command = custom. Use packages from this Azure
    Artifacts/TFS registry.
    #customEndpoint: # string. Optional. Use when customRegistry = useNpmrc
    && command = install || command = ci || command = custom. Credentials for
    registries outside this organization/collection.
    # Destination registry and authentication
    #publishRegistry: 'useExternalRegistry' # 'useExternalRegistry' |
    'useFeed'. Optional. Use when command = publish. Registry location. Default:
    useExternalRegistry.
    #publishFeed: # string. Required when publishRegistry = useFeed &
```

```
    command = publish. Target registry.  
    #publishEndpoint: # string. Required when publishRegistry =  
    useExternalRegistry && command = publish. External Registry.
```

Inputs

`command` - Command

`string`. Required. Allowed values: `ci`, `install`, `publish`, `custom`. Default value: `install`.

Specifies the command and arguments, which are passed to `npm` for execution.

If your arguments contain double quotes ("), escape them with a slash (\), and surround the escaped string with double quotes (").

`workingDir` - Working folder that contains package.json

`string`.

Specifies the path to the folder containing the target `package.json` and `.npmrc` files.

Select the folder, not the file. Example: `/packages/mypackage`.

`verbose` - Verbose logging

`boolean`. Optional. Use when `command = install || command = ci || command = publish`.

Prints more information to the console when the task runs.

`customCommand` - Command and arguments

`string`. Required when `command = custom`.

Runs a custom command. Example: `dist-tag ls mypackage`.

`customRegistry` - Registries to use

`string`. Optional. Use when `command = install || command = ci || command = custom`.

Allowed values: `useNpmrc` (Registries in my `.npmrc`), `useFeed` (Registry I select here).

Default value: `useNpmrc`.

Specifies the registries to use. Commit a `.npmrc` file to your source code repository and set its path as the value, or specify a registry from Azure Artifacts as the value.

customFeed - Use packages from this Azure Artifacts/TFS registry

`string`. Required when `customRegistry = useFeed && command = install || command = ci || command = custom`.

Includes the selected feed in the generated `.npmrc`. For project-scoped feeds, use `ProjectName/FeedName` Or `ProjectID/FeedID`. For organization-scoped feeds, the value should be the feed name.

customEndpoint - Credentials for registries outside this organization/collection

`string`. Optional. Use when `customRegistry = useNpmrc && command = install || command = ci || command = custom`.

Credentials to use for external registries located in the project's `.npmrc`. Leave this blank for registries in this account/collection; the task uses the build's credentials automatically.

publishRegistry - Registry location

`string`. Optional. Use when `command = publish`. Allowed values: `useExternalRegistry` (External npm registry (including other accounts/collections)), `useFeed` (Registry I select here). Default value: `useExternalRegistry`.

Specifies the registry that the command will target.

publishFeed - Target registry

`string`. Required when `publishRegistry = useFeed && command = publish`.

Specifies a registry hosted in the account. You must have Package Management installed and licensed to select a registry here.

publishPackageMetadata - Publish pipeline metadata

`boolean`. Optional. Use when `command = publish && publishRegistry = useFeed && command = install || command = ci || command = publish`. Default value: `true`.

Associates the build/release pipeline's metadata (the run # and source code information) with the package.

publishEndpoint - External Registry

`string`. Required when `publishRegistry = useExternalRegistry && command = publish`.

Specifies the credentials to use for publishing to an external registry.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Note

The **Project Collection Build Service** and your project's **Build Service** identity must be set to **Contributor** to publish your packages to a feed using Azure Pipelines. See [Add new users/groups](#) for more details.

Where can I learn npm commands and arguments?

- [npm docs ↗](#)

Examples

- [Build your Node.js app with gulp](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: npm
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.

Requirement	Description
Command restrictions	Any
Settable variables	Any
Agent version	2.115.0 or greater
Task category	Package

Npm@0 - npm v0 task

Article • 09/26/2023

Use this task to install and publish npm packages or to run an `npm` command. Supports [npmjs.com](#) and authenticated registries like Azure Artifacts.

ⓘ Note

The **npm Authenticate task** is the recommended way to authenticate with Azure Artifacts. This task no longer takes new features and only critical bugs are addressed.

Syntax

YAML

```
# npm v0
# Run an npm command. Use NpmAuthenticate@0 task for latest capabilities.
- task: Npm@0
  inputs:
    cwd: # string. working folder.
    command: 'install' # string. Required. npm command. Default: install.
    #arguments: # string. arguments.
```

Inputs

`cwd` - working folder

`string`.

Specifies the working directory where the `npm` command is run. Defaults to the root of the repo.

`command` - npm command

`string`. Required. Default value: `install`.

Specifies the command and arguments, which are passed to `npm` for execution.

If your arguments contain double quotes (`"`), escape them with a slash (`\`), and surround the escaped string with double quotes (`"`).

`arguments` - arguments

`string`.

The additional arguments that are passed to `npm`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Note

The **Project Collection Build Service** and your project's **Build Service** identity must be set to **Contributor** to publish your packages to a feed using Azure Pipelines. See [Add new users/groups](#) for more details.

Where can I learn npm commands and arguments?

- [npm docs ↗](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: npm
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.

Requirement	Description
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Package

npmAuthenticate@0 - npm authenticate (for task runners) v0 task

Article • 09/26/2023

Use this task to provide `npm` credentials to an `.npmrc` file in your repository for the scope of the build. This enables `npm`, as well as `npm` task runners like gulp and Grunt, to authenticate with private registries.

Syntax

YAML

```
# npm authenticate (for task runners) v0
# Don't use this task if you're also using the npm task. Provides npm
credentials to an .npmrc file in your repository for the scope of the build.
This enables npm task runners like gulp and Grunt to authenticate with
private registries.
- task: npmAuthenticate@0
  inputs:
    workingFile: # string. Required. .npmrc file to authenticate.
    #customEndpoint: # string. Credentials for registries outside this
organization/collection.
```

Inputs

`workingFile` - `.npmrc` file to authenticate

`string`. Required.

The path to the `.npmrc` file that specifies the registries you want to work with. Select the file, not the folder, such as `/packages/mypackage.npmrc`.

`customEndpoint` - Credentials for registries outside this organization/collection

`string`.

The comma-separated list of [npm service connection](#) names for registries outside this organization or collection. The specified `.npmrc` file must contain registry entries corresponding to the service connections. If you only need registries in this organization or collection, leave this blank. The build's credentials are used automatically.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to provide `npm` credentials to an `.npmrc` file in your repository for the scope of the build. This enables `npm`, as well as `npm` task runners like gulp and Grunt, to authenticate with private registries.

- [How does this task work?](#)
- [When in my pipeline should I run this task?](#)
- [I have multiple npm projects. Do I need to run this task for each .npmrc file?](#)
- [My agent is behind a web proxy. Will npmAuthenticate set up npm/gulp/Grunt to use my proxy?](#)
- [My Pipeline needs to access a feed in a different project](#)

How does this task work?

This task searches the specified `.npmrc` file for registry entries, then appends authentication details for the discovered registries to the end of the file. For all registries in the current organization/collection, the build's credentials are used. For registries in a different organization or hosted by a third-party, the registry URIs will be compared to the URIs of the [npm service connections](#) specified by the `customEndpoint` input, and the corresponding credentials will be used. The `.npmrc` file will be reverted to its original state at the end of the pipeline execution.

When in my pipeline should I run this task?

This task must run before you use `npm`, or an `npm` task runner, to install or push packages to an authenticated npm repository such as Azure Artifacts. There are no other ordering requirements.

I have multiple npm projects. Do I need to run this task for each .npmrc file?

This task will only add authentication details to one `.npmrc` file at a time. If you need authentication for multiple `.npmrc` files, you can run the task multiple times, once for each `.npmrc` file. Alternately, consider creating an `.npmrc` file that specifies all registries used by your projects, running `npmAuthenticate` on this `.npmrc` file, and then setting an environment variable to designate this `.npmrc` file as the npm per-user configuration file.

```
YAML

- task: npmAuthenticate@0
  inputs:
    workingFile: $(agent.tempdirectory)/.npmrc
- script: echo ##vso[task.setvariable
variable=NPM_CONFIG_USERCONFIG]$(agent.tempdirectory)/.npmrc
- script: npm ci
  workingDirectory: project1
- script: npm ci
  workingDirectory: project2
```

My agent is behind a web proxy. Will `npmAuthenticate` set up `npm/gulp/Grunt` to use my proxy?

The answer is no. While this task itself will work behind a web proxy [your agent has been configured to use](#), it does not configure `npm` or `npm` task runners to use the proxy.

To do so, you can either:

- Set the environment variables `http_proxy`/`https_proxy` and optionally `no_proxy` to your proxy settings. See [npm config](#) for details. Note that these are commonly used variables which other non-`npm` tools (e.g. curl) may also use.
- Add the proxy settings to the [npm configuration](#), either manually, by using `npm config set`, or by setting [environment variables](#) prefixed with `NPM_CONFIG_`.

Caution:

`npm` task runners may not be compatible with all methods of proxy configuration supported by `npm`.

- Specify the proxy with a command line flag when calling `npm`.

YAML

```
- script: npm ci --https-proxy $(agent.proxyurl)
```

If your proxy requires authentication, you may need to add an additional build step to construct an authenticated proxy URI.

YAML

```
- script: node -e "let u = url.parse(`$(agent.proxyurl)`); u.auth = `$(agent.proxyusername):$(agent.proxypassword)`; console.log(`##vso[task.setvariable variable=proxyAuthUri;issecret=true]` + url.format(u))"
- script: npm publish --https-proxy $(proxyAuthUri)
```

My Pipeline needs to access a feed in a different project

If the pipeline is running in a different project than the project hosting the feed, you must set up the other project to grant read/write access to the build service. See [Package permissions in Azure Pipelines](#) for more details.

Examples

- [Restore npm packages for your project from a registry within your organization](#)
- [Restore and publish npm packages outside your organization](#)
- [npmrc](#)
- [npm](#)

Restore `npm` packages for your project from a registry within your organization

If the only authenticated registries you use are Azure Artifacts registries in your organization, you only need to specify the path to an `.npmrc` file to the `npmAuthenticate` task.

`.npmrc`

```
registry=https://pkgs.dev.azure.com/{organization}/_packaging/{feed}/npm/registry/
```

```
always-auth=true
```

npm

YAML

```
- task: npmAuthenticate@0
  inputs:
    workingFile: .npmrc
- script: npm ci
# ...
- script: npm publish
```

Restore and publish npm packages outside your organization

If your `.npmrc` contains Azure Artifacts registries from a different organization or use a third-party authenticated package repository, you'll need to set up [npm service connections](#) and specify them in the `customEndpoint` input. Registries within your Azure Artifacts organization will also be automatically authenticated.

.npmrc

```
registry=https://pkgs.dev.azure.com/{organization}/{project}/_packaging/{feed}/npm/registry/
#{@{scope}}:registry=https://pkgs.dev.azure.com/{otherorganization}/_packaging/{feed}/npm/registry/
@{@{otherscope}}:registry=https://{{thirdPartyRepository}}/npm/registry/
always-auth=true
```

The registry URL pointing to an Azure Artifacts feed may or may not contain the project. An URL for a project scoped feed must contain the project, and the URL for an organization scoped feed must not contain the project. Learn more about [project scoped feeds](#).

npm

YAML

```

- task: npmAuthenticate@0
  inputs:
    workingFile: .npmrc
    customEndpoint: OtherOrganizationNpmConnection,
    ThirdPartyRepositoryNpmConnection
- script: npm ci
# ...
- script: npm publish -registry
  https://pkgs.dev.azure.com/{otherorganization}/_packaging/{feed}/npm/registry/

```

`OtherOrganizationNpmConnection` and `ThirdPartyRepositoryNpmConnection` are the names of [npm service connections](#) that have been configured and authorized for use in your pipeline, and have URLs that match those in the specified `.npmrc` file.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.115.0 or greater
Task category	Package

NuGetCommand@2 - NuGet v2 task

Article • 09/26/2023

Use this task to restore, pack, or push NuGet packages, or run a NuGet command. This task supports NuGet.org and authenticated feeds like Azure Artifacts and MyGet. This task also uses NuGet.exe and works with .NET Framework apps. For .NET Core and .NET Standard apps, use the .NET Core task.

Syntax

YAML

```
# NuGet v2
# Restore, pack, or push NuGet packages, or run a NuGet command. Supports
NuGet.org and authenticated feeds like Azure Artifacts and MyGet. Uses
NuGet.exe and works with .NET Framework apps. For .NET Core and .NET
Standard apps, use the .NET Core task.
- task: NuGetCommand@2
  inputs:
    command: 'restore' # 'restore' | 'pack' | 'push' | 'custom'. Required.
    Command. Default: restore.
    restoreSolution: '**/*.sln' # string. Alias: solution. Required when
    command = restore. Path to solution, packages.config, or project.json.
    Default: **/*.sln.
    #packagesToPush:
    '$(Build.ArtifactStagingDirectory)/**/*.nupkg;!$(Build.ArtifactStagingDirectory)/**/*symbols.nupkg' # string. Alias: searchPatternPush. Required when
    command = push. Path to NuGet package(s) to publish. Default:
    $(Build.ArtifactStagingDirectory)/**/*.nupkg;!$(Build.ArtifactStagingDirectory)/**/*symbols.nupkg.
    #nuGetFeedType: 'internal' # 'internal' | 'external'. Required when
    command = push. Target feed location. Default: internal.
    #publishVstsFeed: # string. Alias: feedPublish. Required when command =
    push && nuGetFeedType = internal. Target feed.
    #allowPackageConflicts: false # boolean. Optional. Use when command =
    push && nuGetFeedType = internal. Allow duplicates to be skipped. Default:
    false.
    #publishFeedCredentials: # string. Alias: externalEndpoint. Required
    when command = push && nuGetFeedType = external. NuGet server.
    #packagesToPack: '**/*.csproj' # string. Alias: searchPatternPack.
    Required when command = pack. Path to csproj or nuspec file(s) to pack.
    Default: **/*.csproj.
    #configuration: '$(BuildConfiguration)' # string. Alias:
    configurationToPack. Optional. Use when command = pack. Configuration to
    package. Default: $(BuildConfiguration).
    #packDestination: '$(Build.ArtifactStagingDirectory)' # string. Alias:
    outputDir. Optional. Use when command = pack. Package folder. Default:
    $(Build.ArtifactStagingDirectory).
    #arguments: # string. Required when command = custom. Command and
```

```
arguments.

# Feeds and authentication
  feedsToUse: 'select' # 'select' | 'config'. Alias: selectOrConfig.
Required when command = restore. Feeds to use. Default: select.

  #vstsFeed: # string. Alias: feedRestore. Optional. Use when
selectOrConfig = select && command = restore. Use packages from this Azure
Artifacts/TFS feed.

  #includeNuGetOrg: true # boolean. Optional. Use when selectOrConfig =
select && command = restore. Use packages from NuGet.org. Default: true.

  #nugetConfigPath: # string. Optional. Use when selectOrConfig = config
&& command = restore. Path to NuGet.config.

  #externalFeedCredentials: # string. Alias: externalEndpoints. Optional.
Use when selectOrConfig = config && command = restore. Credentials for feeds
outside this organization/collection.

# Advanced

  #noCache: false # boolean. Optional. Use when command = restore. Disable
local cache. Default: false.

  #disableParallelProcessing: false # boolean. Optional. Use when command =
= restore. Disable parallel processing. Default: false.

  #restoreDirectory: # string. Alias: packagesDirectory. Optional. Use
when command = restore. Destination directory.

  #verbosityRestore: 'Detailed' # 'Quiet' | 'Normal' | 'Detailed'.
Optional. Use when command = restore. Verbosity. Default: Detailed.

# Advanced

  #publishPackageMetadata: true # boolean. Optional. Use when command =
push && nuGetFeedType = internal && command = push. Publish pipeline
metadata. Default: true.

  #verbosityPush: 'Detailed' # 'Quiet' | 'Normal' | 'Detailed'. Optional.
Use when command = push. Verbosity. Default: Detailed.

# Pack options

  #versioningScheme: 'off' # 'off' | 'byPrereleaseNumber' | 'byEnvVar' |
'byBuildNumber'. Required when command = pack. Automatic package versioning.
Default: off.

  #includeReferencedProjects: false # boolean. Optional. Use when
versioningScheme = off && command = pack. Include referenced projects.
Default: false.

  #versionEnvVar: # string. Required when versioningScheme = byEnvVar &&
command = pack. Environment variable.

  #majorVersion: '1' # string. Alias: requestedMajorVersion. Required when
versioningScheme = byPrereleaseNumber && command = pack. Major. Default: 1.

  #minorVersion: '0' # string. Alias: requestedMinorVersion. Required when
versioningScheme = byPrereleaseNumber && command = pack. Minor. Default: 0.

  #patchVersion: '0' # string. Alias: requestedPatchVersion. Required when
versioningScheme = byPrereleaseNumber && command = pack. Patch. Default: 0.

  #packTimezone: 'utc' # 'utc' | 'local'. Optional. Use when
versioningScheme = byPrereleaseNumber && command = pack. Time zone. Default:
utc.

  #includeSymbols: false # boolean. Optional. Use when command = pack.
Create symbols package. Default: false.

  #toolPackage: false # boolean. Optional. Use when command = pack. Tool
Package. Default: false.

# Advanced

  #buildProperties: # string. Optional. Use when command = pack.
Additional build properties.

  #basePath: # string. Optional. Use when command = pack. Base path.
```

```
#verbosityPack: 'Detailed' # 'Quiet' | 'Normal' | 'Detailed'. Optional.  
Use when command = pack. Verbosity. Default: Detailed.
```

Inputs

`command` - Command

`string`. Required. Allowed values: `restore`, `pack`, `push`, `custom`. Default value: `restore`.

Specifies the NuGet command to run. Use the `custom` value to add arguments or to use a different command.

`restoreSolution` - Path to solution, packages.config, or project.json

Input alias: `solution`. `string`. Required when `command = restore`. Default value: `**/*.sln`.

Specifies the path to the solution, `packages.config`, or `project.json` file that references the packages to be restored.

`feedsToUse` - Feeds to use

Input alias: `selectOrConfig`. `string`. Required when `command = restore`. Allowed values: `select` (Feed(s) I select here), `config` (Feeds in my NuGet.config). Default value: `select`.

Specifies a feed from Azure Artifacts and/or NuGet.org for the task to use with the `select` value. Alternatively, you can commit a `NuGet.config` file to your source code repository and set its path as the value using the `config` value.

`vstsFeed` - Use packages from this Azure Artifacts/TFS feed

Input alias: `feedRestore`. `string`. Optional. Use when `selectOrConfig = select && command = restore`.

Specifies the selected feed in the generated `NuGet.config`. You must have Package Management installed and licensed to specify a feed here.

`includeNuGetOrg` - Use packages from NuGet.org

`boolean`. Optional. Use when `selectOrConfig = select && command = restore`. Default value: `true`.

Includes NuGet.org in the generated `NuGet.config`.

nugetConfigPath - Path to NuGet.config

`string`. Optional. Use when `selectOrConfig = config && command = restore`.

Specifies the path to the `NuGet.config` in your repository that determines the feeds from which to restore packages.

externalFeedCredentials - Credentials for feeds outside this organization/collection

Input alias: `externalEndpoints`. `string`. Optional. Use when `selectOrConfig = config && command = restore`.

Specifies the credentials to use for external registries located in the selected `NuGet.config`. This is the name of your NuGet service connection. For feeds in this organization or collection, leave this blank; the build's credentials are used automatically.

noCache - Disable local cache

`boolean`. Optional. Use when `command = restore`. Default value: `false`.

Prevents NuGet from using packages from local machine caches when set to `true`.

disableParallelProcessing - Disable parallel processing

`boolean`. Optional. Use when `command = restore`. Default value: `false`.

Prevents NuGet from installing multiple packages in parallel processes when set to `true`.

restoreDirectory - Destination directory

Input alias: `packagesDirectory`. `string`. Optional. Use when `command = restore`.

Specifies the folder in which packages are installed. If no folder is specified, packages are restored into a `packages/` folder alongside the selected solution, `packages.config`, or `project.json`.

verbosityRestore - Verbosity

`string`. Optional. Use when `command = restore`. Allowed values: `Quiet`, `Normal`, `Detailed`. Default value: `Detailed`.

Specifies the amount of detail displayed in the output.

packagesToPush - Path to NuGet package(s) to publish

Input alias: `searchPatternPush`. `string`. Required when `command = push`. Default value: `$(Build.ArtifactStagingDirectory)/**/*.nupkg;!$(Build.ArtifactStagingDirectory)/**/*.symbols.nupkg`.

Specifies the pattern to match or path to `nupkg` files to be uploaded. Multiple patterns can be separated by a semicolon.

nuGetFeedType - Target feed location

`string`. Required when `command = push`. Allowed values: `internal` (This organization/collection), `external` (External NuGet server (including other accounts/collections)). Default value: `internal`.

Specifies whether the target feed is an internal feed/collection or an external NuGet server.

publishVstsFeed - Target feed

Input alias: `feedPublish`. `string`. Required when `command = push && nuGetFeedType = internal`.

Specifies a feed hosted in this account. You must have Azure Artifacts installed and licensed to select a feed here.

publishPackageMetadata - Publish pipeline metadata

`boolean`. Optional. Use when `command = push && nuGetFeedType = internal && command = push`. Default value: `true`.

Changes the version number of the subset of changed packages within a set of continually published packages.

allowPackageConflicts - Allow duplicates to be skipped

`boolean`. Optional. Use when `command = push && nuGetFeedType = internal`. Default value: `false`.

Reports task success even if some of your packages are rejected with 409 Conflict errors.

This option is currently only available on Azure Pipelines and Windows agents. If `NuGet.exe` encounters a conflict, the task will fail. This option will not work and publishing will fail if you are within a proxy environment.

`publishFeedCredentials` - NuGet server

Input alias: `externalEndpoint`. `string`. Required when `command = push && nuGetFeedType = external`.

Specifies the NuGet service connection that contains the external NuGet server's credentials.

`verbosityPush` - Verbosity

`string`. Optional. Use when `command = push`. Allowed values: `Quiet`, `Normal`, `Detailed`. Default value: `Detailed`.

Specifies the amount of detail displayed in the output.

`packagesToPack` - Path to csproj or nuspec file(s) to pack

Input alias: `searchPatternPack`. `string`. Required when `command = pack`. Default value: `**/*.csproj`.

Specifies the pattern that the task uses to search for csproj directories to pack.

You can separate multiple patterns with a semicolon, and you can make a pattern negative by prefixing it with `!`. Example: `**/*.csproj; !**/*.Tests.csproj`.

`configuration` - Configuration to package

Input alias: `configurationToPack`. `string`. Optional. Use when `command = pack`. Default value: `$(BuildConfiguration)`.

Specifies the configuration to package when using a csproj file.

`packDestination` - Package folder

Input alias: `outputDir`. `string`. Optional. Use when `command = pack`. Default value: `$(Build.ArtifactStagingDirectory)`.

Specifies the folder where the task creates packages. If the value is empty, the task creates packages at the source root.

`versioningScheme` - Automatic package versioning

`string`. Required when `command = pack`. Allowed values: `off`, `byPrereleaseNumber` (Use the date and time), `byEnvVar` (Use an environment variable), `byBuildNumber` (Use the build number). Default value: `off`.

Applies automatic package versioning depending on the specified value. This string cannot be used with `includeReferencedProjects`. The allowed values are:

- **`byPrereleaseNumber` - Use the date and time:** The task will generate a [SemVer](#)-compliant version formatted as `X.Y.Z-ci-datetime`, where you specify the values of X, Y, and Z.
- **`byEnvVar` - Use an environment variable:** The task will use an environment variable that you specify and contains the version number you want to use.
- **`byBuildNumber` - Use the build number:** The task will use the build number to version the package.

① Note

Under General, set the build format to be

`$(BuildDefinitionName)_$(Year:yyyy).$(Month).$(DayOfMonth)$(Rev:r)`.

`includeReferencedProjects` - Include referenced projects

`boolean`. Optional. Use when `versioningScheme = off && command = pack`. Default value: `false`.

Includes referenced projects either as dependencies or as part of the package. Cannot be used with automatic package versioning. If a referenced project has a corresponding `nuspec` file that has the same name as the project, then that referenced project is added as a dependency. Otherwise, the referenced project is added as part of the package.

Learn more about [using the pack command for NuGet CLI to create NuGet packages](#).

`versionEnvVar` - Environment variable

`string`. Required when `versioningScheme = byEnvVar && command = pack`.

Specifies the variable name without `$`, `$env`, or `%`.

`majorVersion` - Major

Input alias: `requestedMajorVersion`. `string`. Required when `versioningScheme = byPrereleaseNumber && command = pack`. Default value: `1`.

The `X` in version `X.Y.Z`.

`minorVersion` - Minor

Input alias: `requestedMinorVersion`. `string`. Required when `versioningScheme = byPrereleaseNumber && command = pack`. Default value: `0`.

The `Y` in version `X.Y.Z`.

`patchVersion` - Patch

Input alias: `requestedPatchVersion`. `string`. Required when `versioningScheme = byPrereleaseNumber && command = pack`. Default value: `0`.

The `Z` in version `X.Y.Z`.

`packTimezone` - Time zone

`string`. Optional. Use when `versioningScheme = byPrereleaseNumber && command = pack`. Allowed values: `utc`, `local` (Agent local time). Default value: `utc`.

Specifies the desired time zone used to produce the version of the package. Selecting `utc` is recommended if you're using hosted build agents, as their date and time might differ.

`includeSymbols` - Create symbols package

`boolean`. Optional. Use when `command = pack`. Default value: `false`.

Specifies that the package contains sources and symbols. When used with a `.nuspec` file, this creates a regular NuGet package file and the corresponding symbols package.

`toolPackage` - Tool Package

`boolean`. Optional. Use when `command = pack`. Default value: `false`.

Determines if the output files of the project should be in the tool folder.

buildProperties - Additional build properties

`string`. Optional. Use when `command = pack`.

Specifies a list of token=value pairs, separated by semicolons, where each occurrence of `$token$` in the `.nuspec` file will be replaced with the given value. Values can be strings in quotation marks.

basePath - Base path

`string`. Optional. Use when `command = pack`.

Specifies the base path of the files defined in the `nuspec` file.

verbosityPack - Verbosity

`string`. Optional. Use when `command = pack`. Allowed values: `Quiet`, `Normal`, `Detailed`.

Default value: `Detailed`.

Specifies the amount of detail displayed in the output.

arguments - Command and arguments

`string`. Required when `command = custom`.

Specifies the command and arguments that will be passed to `NuGet.exe` for execution. If NuGet 3.5 or later is used, authenticated commands like `list`, `restore`, and `publish` against any feed in this organization or collection that the Project Collection Build Service has access to will be automatically authenticated.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Important

The **NuGet Authenticate** task is the new recommended way to authenticate with Azure Artifacts and other NuGet repositories. This task no longer takes new features, and only critical bugs are addressed.

Use this task to install and update NuGet package dependencies, or package and publish NuGet packages. Uses NuGet.exe and works with .NET Framework apps. For .NET Core and .NET Standard apps, use the .NET Core task.

If your code depends on NuGet packages, make sure to add this step before your [Visual Studio Build task](#). Also make sure to clear the deprecated **Restore NuGet Packages** checkbox in that task.

If you are working with .NET Core or .NET Standard, use the [.NET Core](#) task, which has full support for all package scenarios and is currently supported by dotnet.

Tip

This version of the NuGet task uses NuGet 4.1.0 by default. To select a different version of NuGet, use the [Tool Installer](#).

Versioning schemes

For **byPrereleaseNumber**, the version will be set to the values you choose for the major version, the minor version, and the patch, plus the date and time, in the format

`yyyymmdd-hhmmss`.

For **byEnvVar**, the version will be set to the value of the environment variable that has the name specified by the **versionEnvVar** parameter, e.g. `MyVersion` (no \$, just the environment variable name). Make sure the environment variable is set to a proper SemVer, such as `1.2.3` or `1.2.3-beta1`.

For **byBuildNumber**, the version will be set using the pipeline run's build number. This is the value specified for the pipeline's `name` property, which gets saved to the `BUILD_BUILDDATE` environment variable). Ensure that the build number being used contains a proper SemVer, such as `1.0.$(Rev:r)`. When using **byBuildNumber**, the task will extract the dotted version, `1.2.3.4`, from the build number string, and use only that portion. The rest of the string will be dropped. If you want to use the build number as is, you can use **byEnvVar** as described above, and set **versionEnvVar** to `BUILD_BUILDDATE`.

Examples

Restore

Restore all your solutions with packages from a selected feed.

YAML

```
# Restore from a project scoped feed in the same organization
- task: NuGetCommand@2
  inputs:
    command: 'restore'
    feedsToUse: 'select'
    vstsFeed: 'my-project/my-project-scoped-feed'
    includeNuGetOrg: false
    restoreSolution: '**/*.sln'
```

YAML

```
# Restore from an organization scoped feed in the same organization
- task: NuGetCommand@2
  inputs:
    command: 'restore'
    feedsToUse: 'select'
    vstsFeed: 'my-organization-scoped-feed'
    restoreSolution: '**/*.sln'
```

YAML

```
# Restore from a feed in a different organization
- task: NuGetCommand@2
  inputs:
    command: 'restore'
    feedsToUse: config
    nugetConfigPath: ./nuget.config
    restoreSolution: '**/*.sln'
    externalFeedCredentials: 'MyServiceConnectionName'
    noCache: true
    continueOnError: true
```

YAML

```
# Restore from feed(s) set in nuget.config
- task: NuGetCommand@2
  inputs:
    command: 'restore'
```

```
feedsToUse: 'config'  
nugetConfigPath: 'nuget.config'
```

Package

Create a NuGet package in the destination folder.

YAML

```
# Package a project  
- task: NuGetCommand@2  
  inputs:  
    command: 'pack'  
    packagesToPack: '**/*.csproj'  
    packDestination: '$(Build.ArtifactStagingDirectory)'
```

Push

ⓘ Note

Pipeline artifacts are downloaded to the `Pipeline.Workspace` directory, and to the `System.ArtifactsDirectory` directory for classic release pipelines. `packagesToPush` value can be set to `$(Pipeline.Workspace)/**/*.nupkg` or `$(System.ArtifactsDirectory)/**/*.nupkg` respectively.

- Push/Publish a package to a feed defined in your NuGet.config.

YAML

```
# Push a project  
- task: NuGetCommand@2  
  inputs:  
    command: 'push'  
    packagesToPush: '$(Build.ArtifactStagingDirectory)/**/*.nupkg'  
    feedsToUse: 'config'  
    nugetConfigPath: '$(Build.WorkingDirectory)/NuGet.config'
```

- Push/Publish a package to an organization scoped feed

YAML

```
# Push a project  
- task: NuGetCommand@2  
  inputs:
```

```
command: 'push'  
nuGetFeedType: 'internal'  
publishVstsFeed: 'my-organization-scoped-feed'
```

- Push/Publish a package to a project scoped feed

YAML

```
# Push a project  
- task: NuGetCommand@2  
  inputs:  
    command: 'push'  
    nuGetFeedType: 'internal'  
    publishVstsFeed: 'my-project/my-project-scoped-feed'
```

- Push/Publish a package to NuGet.org

YAML

```
# Push a project  
- task: NuGetCommand@2  
  inputs:  
    command: 'push'  
    feedsToUse: 'config'  
    includeNugetOrg: 'true'
```

Custom

Run any other NuGet command besides the default ones: pack, push, and restore.

YAML

```
# list local NuGet resources.  
- task: NuGetCommand@2  
  displayName: 'list locals'  
  inputs:  
    command: custom  
    arguments: 'locals all -list'
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release

Requirement	Description
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Package

NuGetAuthenticate@1 - NuGet authenticate v1 task

Article • 09/26/2023

Configure NuGet tools to authenticate with Azure Artifacts and other NuGet repositories. Requires NuGet >= 4.8.5385, dotnet >= 6, or MSBuild >= 15.8.166.59604.

Syntax

YAML

```
# NuGet authenticate v1
# Configure NuGet tools to authenticate with Azure Artifacts and other NuGet
repositories. Requires NuGet >= 4.8.5385, dotnet >= 6, or MSBuild >=
15.8.166.59604.
- task: NuGetAuthenticate@1
  inputs:
    #nugetServiceConnections: # string. Service connection credentials for
    feeds outside this organization.
    #forceReinstallCredentialProvider: false # boolean. Reinstall the
    credential provider even if already installed. Default: false.
```

Inputs

`nugetServiceConnections` - Service connection credentials for feeds outside this organization
`string`.

Optional. The comma-separated list of [NuGet service connection](#) names for feeds outside this organization or collection. For feeds in this organization or collection, leave this blank; the build's credentials are used automatically.

`forceReinstallCredentialProvider` - Reinstall the credential provider even if already installed
`boolean`. Default value: `false`.

Optional. Reinstalls the credential provider to the user profile directory, even if it's already installed. If the credential provider is already installed in the user profile, the task determines if it is overwritten with the task-provided credential provider. This may upgrade (or potentially downgrade) the credential provider.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Important

This task is only compatible with NuGet >= 4.8.0.5385, dotnet >= 6, or MSBuild >= 15.8.166.59604.

What tools are compatible with this task?

This task configures tools that support [NuGet cross platform plugins](#). The tools currently include nuget.exe, dotnet, and recent versions of MSBuild with built-in support for restoring NuGet packages.

Specifically, this task will configure:

- nuget.exe (version 4.8.5385 or higher)
- dotnet / .NET 6 SDK or higher (a previous version of this task, NuGetAuthenticateV0, requires .NET Core 2.1, which is no longer supported)
- MSBuild (version 15.8.166.59604 or higher)

Upgrading to the latest stable version is recommended if you encounter any issues.

I get "A task was canceled" errors during a package restore. What should I do?

Known issues in NuGet and in the Azure Artifacts Credential Provider can cause this type of error, and updating to the latest nuget may help.

A [known issue](#) in some versions of nuget/dotnet can cause this error, especially during large restores on resource constrained machines. This issue is fixed in [NuGet 5.2](#), and .NET Core SDK 2.1.80X and 2.2.40X. If you are using an older version, try upgrading your version of NuGet or dotnet. The [.NET Core Tool Installer](#) task can be used to install a newer version of the .NET Core SDK.

There are also known issues with the Azure Artifacts Credential Provider (installed by this task), including [artifacts-credprovider/#77](#) and [artifacts-credprovider/#108](#). If you experience these issues, ensure you have the latest credential provider by setting the input `forceReinstallCredentialProvider` to `true` in the NuGet Authenticate task. This setting will also ensure your credential provider is automatically updated as issues are resolved.

If neither of the above resolves the issue, enable [Plugin Diagnostic Logging](#) and report the issue to [NuGet](#) and the [Azure Artifacts Credential Provider](#).

How is this task different than the NuGetCommand and DotNetCoreCLI tasks?

This task configures nuget.exe, dotnet, and MSBuild to authenticate with Azure Artifacts or other repositories that require authentication. After this task runs, you can then invoke the tools in a later step (either directly or via a script) to restore or push packages.

The NuGetCommand and DotNetCoreCLI tasks require using the task to restore or push packages, as authentication to Azure Artifacts is only configured within the lifetime of the task. This can prevent you from restoring or pushing packages within your own script. It may also prevent you from passing specific command line arguments to the tool.

The NuGetAuthenticate task is the recommended way to use authenticated feeds within a pipeline.

When in my pipeline should I run this task?

This task must run before you use a NuGet tool to restore or push packages to an authenticated package source such as Azure Artifacts. There are no other ordering requirements. For example, this task can safely run either before or after a NuGet or .NET Core tool installer task.

How do I configure a NuGet package source that uses ApiKey ("NuGet API keys"), such as nuget.org?

Some package sources such as nuget.org use API keys for authentication when pushing packages, rather than `username/password` credentials. Due to limitations in NuGet, this task cannot be used to set up a NuGet service connection that uses an API key.

Instead:

1. Configure a [secret variable](#) containing the ApiKey
2. Perform the package push using `nuget push -ApiKey $(myNuGetApiKey)` or `dotnet nuget push --api-key $(myNuGetApiKey)`, assuming you named the variable `myNuGetApiKey`

My agent is behind a web proxy. Will NuGetAuthenticate set up nuget.exe, dotnet, and MSBuild to use my proxy?

No. While this task itself will work behind a web proxy [your agent has been configured to use](#), it does not configure NuGet tools to use the proxy.

To do so, you can either:

- Set the environment variable `http_proxy` and optionally `no_proxy` to your proxy settings. See [NuGet CLI environment variables](#) for details. These variables are commonly used variables which other non-NuGet tools (e.g. curl) may also use.

Caution:

The `http_proxy` and `no_proxy` variables are case-sensitive on Linux and Mac operating systems and must be lowercase. Attempting to use an Azure Pipelines variable to set the environment variable will not work, as it will be converted to uppercase. Instead, set the environment variables on the self-hosted agent's machine and restart the agent.

- Add the proxy settings to the [user-level nuget.config](#) file, either manually or using `nuget config -set` as described in the [nuget.config reference documentation](#).

Caution:

The proxy settings (such as `http_proxy`) must be added to the user-level config. They will be ignored if specified in a different nuget.config file.

How do I debug if I have issues with this task?

To get verbose logs from the pipeline, add a pipeline variable `system.debug` and set to `true`.

How does this task work?

This task installs the [Azure Artifacts Credential Provider](#) into the NuGet plugins directory if it is not already installed. It then sets environment variables such as `VSS_NUGET_URI_PREFIXES` and `VSS_NUGET_ACCESSTOKEN` to configure the credential provider. These variables remain set for the lifetime of the job. When restoring or pushing packages, a NuGet tool executes the credential provider, which uses the above variables to determine if it should return credentials back to the tool.

See the credential provider documentation for more details.

My Pipeline needs to access a feed in a different project

If the pipeline is running in a different project than the project hosting the feed, you must set up the other project to grant read/write access to the build service. See [Package permissions in Azure Pipelines](#) for more details.

Will this work for pipeline runs that are triggered from an external fork?

No. Pipeline runs that are triggered from an external fork don't have access to the proper secrets for internal feed authentication. Thus, it will appear like the authenticate task is successful, but subsequent tasks that require authentication (such as Nuget push) will fail with an error along the lines of: `##[error]The nuget command failed with exit code(1) and error(Response status code does not indicate success: 500 (Internal Server Error - VS800075: The project with id 'vstfs:///Classification/TeamProject/341ec244-e856-40ad-845c-af31c33c2152' does not exist, or you do not have permission to access it. (DevOps Activity ID: C12C19DC-642C-469A-8F58-C89F2D81FEA7))`. After the Pull Request is merged into the origin, then a pipeline that is triggered from that event will authenticate properly.

I updated from NuGetAuthenticateV0 to NuGetAuthenticateV1 and now my dotnet command fails with 401

If you are updating from NuGetAuthenticateV0 to NuGetAuthenticateV1 and get an error running a dotnet command, look for the message `It was not possible to find any compatible framework version` from the logs. For dotnet users, NuGetAuthenticateV1 requires .NET 6 instead of .NET Core 2.1, which is required in NuGetAuthenticateV0 and is no longer supported. To resolve the issue, use the UseDotNet@2 task before the dotnet command to install .NET 6.

YAML

```
- task: UseDotNet@2
  displayName: Use .NET 6 SDK
  inputs:
    packageType: sdk
    version: 6.x
```

Examples

Restore and push NuGet packages within your organization

If all of the Azure Artifacts feeds you use are in the same organization as your pipeline, you can use the NuGetAuthenticate task without specifying any inputs. For project scoped feeds that are in a different project than where the pipeline is running in, you must manually give the project and the feed access to the pipeline's project's build service.

nugget.config

XML

```
<configuration>
  <packageSources>
    <!--
      Any Azure Artifacts feeds within your organization will automatically
      be authenticated. Both dev.azure.com and visualstudio.com domains are
      supported.

      Project scoped feed URL includes the project, organization scoped feed
      URL does not.

      -->
    <add key="MyProjectFeed1"
      value="https://pkgs.dev.azure.com/{organization}/{project}/_packaging/{feed}
      /nuget/v3/index.json" />
    <add key="MyProjectFeed2"
      value="https:///{organization}.pkgs.visualstudio.com/{project}/_packaging/{fe
```

```
ed}/nuget/v3/index.json" />
    <add key="MyOtherProjectFeed1"
value="https://pkgs.dev.azure.com/{organization}/{project}/_packaging/{feed@view}/nuget/v3/index.json" />
    <add key="MyOrganizationFeed1"
value="https://pkgs.dev.azure.com/{organization}/_packaging/{feed}/nuget/v3/
index.json" />
</packageSources>
</configuration>
```

To use a service connection, specify the service connection in the `nuGetServiceConnections` input for the NuGet Authenticate task. You can then reference the service connection with `-ApiKey AzureArtifacts` in a task.

nuget.exe

YAML

```
- task: NuGetAuthenticate@1
  inputs:
    nuGetServiceConnections: OtherOrganizationFeedConnection,
ThirdPartyRepositoryConnection
- task: NuGetToolInstaller@1 # Optional if nuget.exe >= 4.8.5385 is already
on the path
  inputs:
    versionSpec: '*'
    checkLatest: true
- script: nuget restore
# ...
- script: nuget push -ApiKey AzureArtifacts -Source "MyProjectFeed1"
MyProject.*.nupkg
```

dotnet

YAML

```
- task: NuGetAuthenticate@1
  inputs:
    nuGetServiceConnections: OtherOrganizationFeedConnection,
ThirdPartyRepositoryConnection
- task: UseDotNet@2 # Optional if the .NET Core SDK is already installed
- script: dotnet restore
# ...
- script: dotnet nuget push --api-key AzureArtifacts --source
https://pkgs.dev.azure.com/{organization}/_packaging/{feed1}/nuget/v3/index.
json MyProject.*.nupkg
```

In the above examples, `OtherOrganizationFeedConnection` and `ThirdPartyRepositoryConnection` are the names of [NuGet service connections](#) that have been configured and authorized for use in your pipeline, and have URLs that match those in your `nuget.config` or command line argument.

The package source URL pointing to an Azure Artifacts feed may or may not contain the project. An URL for a project scoped feed must contain the project, and a URL for an organization scoped feed must not contain the project. Learn more about [project scoped feeds](#).

Restore and push NuGet packages outside your organization

If you use Azure Artifacts feeds from a different organization or use a third-party authenticated package repository, you'll need to set up [NuGet service connections](#) and specify them in the `nuGetServiceConnections` input. Feeds within your Azure Artifacts organization will also be automatically authenticated.

nuget.config

XML

```
<configuration>
  <packageSources>
    <!-- Any Azure Artifacts feeds within your organization will
        automatically be authenticated -->
    <add key="MyProjectFeed1"
        value="https://pkgs.dev.azure.com/{organization}/{project}/_packaging/{feed}
        /nuget/v3/index.json" />
    <add key="MyOrganizationFeed"
        value="https://pkgs.dev.azure.com/{organization}/_packaging/{feed}/nuget/v3/
        index.json" />
    <!-- Any package source listed here whose URL matches the URL of a
        service connection in nuGetServiceConnections will also be authenticated.
        The key name here does not need to match the name of the service
        connection. -->
    <add key="OtherOrganizationFeed"
        value="https://pkgs.dev.azure.com/{otherorganization}/_packaging/{feed}/nuge
        t/v3/index.json" />
    <add key="ThirdPartyRepository"
        value="https://{{thirdPartyRepository}}/index.json" />
  </packageSources>
</configuration>
```

nuget.exe

YAML

```
- task: NuGetAuthenticate@1
  inputs:
    nuGetServiceConnections: OtherOrganizationFeedConnection,
ThirdPartyRepositoryConnection
- task: NuGetToolInstaller@1 # Optional if nuget.exe >= 4.8.5385 is already
on the path
  inputs:
    versionSpec: '*'
    checkLatest: true
- script: nuget restore
# ...
- script: nuget push -ApiKey AzureArtifacts -Source "MyProjectFeed1"
MyProject.*.nupkg
```

dotnet

YAML

```
- task: NuGetAuthenticate@1
  inputs:
    nuGetServiceConnections: OtherOrganizationFeedConnection,
ThirdPartyRepositoryConnection
- task: UseDotNet@2 # Optional if the .NET Core SDK is already installed
- script: dotnet restore
# ...
- script: dotnet nuget push --api-key AzureArtifacts --source
"MyProjectFeed1" MyProject.*.nupkg
```

`OtherOrganizationFeedConnection` and `ThirdPartyRepositoryConnection` are the names of [NuGet service connections](#) that have been configured and authorized for use in your pipeline, and have URLs that match those in your nuget.config or command line argument.

The package source URL pointing to an Azure Artifacts feed may or may not contain the project. An URL for a project scoped feed must contain the project, and a URL for an organization scoped feed must not contain the project. Learn more about [project scoped feeds](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release

Requirement	Description
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Package

NuGetAuthenticate@0 - NuGet authenticate v0 task

Article • 09/26/2023

This version of the task is deprecated. Use [NuGetAuthenticate@1](#) instead. Use this task to configure NuGet tools to authenticate with Azure Artifacts and other NuGet repositories. Requires NuGet >= 4.8.5385, dotnet >= 2.1.400, or MSBuild >= 15.8.166.59604.

Syntax

YAML

```
# NuGet authenticate v0
# This version of the task is deprecated, use NuGetAuthenticateV1 instead.
Configure NuGet tools to authenticate with Azure Artifacts and other NuGet
repositories. Requires NuGet >= 4.8.5385, dotnet >= 2.1.400, or MSBuild >=
15.8.166.59604.
- task: NuGetAuthenticate@0
  inputs:
    #nugetServiceConnections: # string. Service connection credentials for
    feeds outside this organization.
    #forceReinstallCredentialProvider: false # boolean. Reinstall the
    credential provider even if already installed. Default: false.
```

Inputs

`nugetServiceConnections` - Service connection credentials for feeds outside this organization
`string`.

Optional. The comma-separated list of [NuGet service connection](#) names for feeds outside this organization or collection. For feeds in this organization or collection, leave this blank; the build's credentials are used automatically.

`forceReinstallCredentialProvider` - Reinstall the credential provider even if already installed
`boolean`. Default value: `false`.

Optional. Reinstalls the credential provider to the user profile directory, even if it's already installed. If the credential provider is already installed in the user profile, the task determines if it is overwritten with the task-provided credential provider. This may upgrade (or potentially downgrade) the credential provider.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to configure NuGet tools to authenticate with Azure Artifacts and other NuGet repositories. Requires NuGet >= 4.8.5385, dotnet >= 2.1.400, or MSBuild >= 15.8.166.59604.

This version of the task is deprecated. Use [NuGetAuthenticate@1](#) instead. Configure NuGet tools to authenticate with Azure Artifacts and other NuGet repositories.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.120.0 or greater
Task category	Package

See also

- [NuGetAuthenticate@1](#)

NuGet@0 - NuGet command v0 task

Article • 09/26/2023

NuGet@0 is deprecated. Use the [NuGetCommand](#) task instead. It works with the new Tool Installer framework so you can easily use new versions of NuGet without waiting for a task update, provides better support for authenticated feeds outside this organization/collection, and uses NuGet 4 by default.

Syntax

YAML

```
# NuGet command v0
# Deprecated: use the "NuGet" task instead. It works with the new Tool
Installer framework so you can easily use new versions of NuGet without
waiting for a task update, provides better support for authenticated feeds
outside this organization/collection, and uses NuGet 4 by default.
- task: NuGet@0
  inputs:
    command: # string. Required. Command.
    #arguments: # string. Arguments.
```

Inputs

command - Command

`string`. Required.

Specifies the NuGet command to run. Set the value as `Custom` to add arguments or to use a different command. The allowed values are `restore`, `pack`, `custom`, and `push`.

arguments - Arguments

`string`.

Specifies the command and arguments that will be passed to `NuGet.exe` for execution. If NuGet 3.5 or later is used, authenticated commands like `list`, `restore`, and `publish` against any feed in this organization or collection that the Project Collection Build Service has access to will be automatically authenticated.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Package

NuGetInstaller@0 - NuGet Installer v0 task

Article • 09/26/2023

Installs or restores missing NuGet packages. This task is deprecated; use [NuGetAuthenticate@0](#).

Syntax

YAML

```
# NuGet Installer v0
# Installs or restores missing NuGet packages. Use NuGetAuthenticate@0 task
# for latest capabilities.
- task: NuGetInstaller@0
  inputs:
    solution: '**/*.sln' # string. Required. Path to solution or
    packages.config. Default: **/*.sln.
    #nugetConfigPath: # string. Path to NuGet.config.
    restoreMode: 'restore' # 'restore' | 'install'. Required. Installation
    type. Default: restore.
    #noCache: false # boolean. Disable local cache. Default: false.
    #nuGetRestoreArgs: # string. NuGet arguments.
    # Advanced
    #verbosity: '-' # '-' | 'Quiet' | 'Normal' | 'Detailed'. Verbosity.
    Default: -.
    nuGetVersion: '3.3.0' # '3.3.0' | '3.5.0.1829' | '4.0.0.2283' |
    'custom'. Required. NuGet Version. Default: 3.3.0.
    #nuGetPath: # string. Path to NuGet.exe.
```

Inputs

`solution` - Path to solution or packages.config

`string`. Required. Default value: `**/*.sln`.

The path to the Visual Studio solution file or NuGet packages.config.

`nugetConfigPath` - Path to NuGet.config

`string`.

Equivalent to the `-ConfigFile` NuGet.exe command line argument.

restoreMode - Installation type

`string`. Required. Allowed values: `restore`, `install`. Default value: `restore`.

Restore will restore the packages a solution depends upon, and is generally what you want.

Install will install packages from a packages.config file. Use this option if you want to install a standalone tool package.

noCache - Disable local cache

`boolean`. Default value: `false`.

Equivalent to the `-NoCache` NuGet.exe command line argument.

nuGetRestoreArgs - NuGet arguments

`string`.

Additional arguments passed to NuGet.exe restore or install. [More Information ↗](#).

verbosity - Verbosity

`string`. Allowed values: `-`, `Quiet`, `Normal`, `Detailed`. Default value: `-`.

NuGet's verbosity level.

nuGetVersion - NuGet Version

`string`. Required. Allowed values: `3.3.0`, `3.5.0.1829` (3.5.0), `4.0.0.2283` (4.0.0), `custom`.

Default value: `3.3.0`.

The version of NuGet to use, or external version.

nuGetPath - Path to NuGet.exe

`string`.

Optionally supply the path to NuGet.exe. Will override version selection.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.115.0 or greater
Task category	Package

NuGetPackager@0 - NuGet packager v0 task

Article • 09/26/2023

NuGetPackager@0 is deprecated. Use the NuGet task instead. It works with the new Tool Installer framework so you can easily use new versions of NuGet without waiting for a task update, provides better support for authenticated feeds outside this organization/collection, and uses NuGet 4 by default.

Syntax

YAML

```
# NuGet packager v0
# Deprecated: use the "NuGet" task instead. It works with the new Tool
Installer framework so you can easily use new versions of NuGet without
waiting for a task update, provides better support for authenticated feeds
outside this organization/collection, and uses NuGet 4 by default.
- task: NuGetPackager@0
  inputs:
    searchPattern: '**\*.csproj' # string. Required. Path to csproj or
nuspec file(s) to pack. Default: **\*.csproj.
    #outputdir: # string. Package Folder.
  # Pack options
    #includeReferencedProjects: false # boolean. Include referenced
projects. Default: false.
    versionByBuild: 'false' # 'false' | 'byPrereleaseNumber' | 'byEnvVar' |
'true'. Required. Automatic package versioning. Default: false.
    #versionEnvVar: # string. Required when versionByBuild = byEnvVar.
Environment variable.
    #requestedMajorVersion: '1' # string. Required when versionByBuild =
byPrereleaseNumber. Major. Default: 1.
    #requestedMinorVersion: '0' # string. Required when versionByBuild =
byPrereleaseNumber. Minor. Default: 0.
    #requestedPatchVersion: '0' # string. Required when versionByBuild =
byPrereleaseNumber. Patch. Default: 0.
  # Advanced
    #configurationToPack: '$(BuildConfiguration)' # string. Configuration to
Package. Default: $(BuildConfiguration).
    #buildProperties: # string. Additional build properties.
    #nuGetAdditionalArgs: # string. NuGet Arguments.
    #nuGetPath: # string. Path to NuGet.exe.
```

Inputs

searchPattern - Path to csproj or nuspec file(s) to pack

string. Required. Default value: `***.csproj`.

The pattern that the task uses to search for `csproj` or `nuspec` files to pack.

You can separate multiple patterns with a semicolon, and you can make a pattern negative by prefixing it with `-:`. Example: `***.csproj;-:***.Tests.csproj`.

outputdir - Package Folder

string.

The folder where the task creates packages. If this string is empty, packages will be created in the folder where the `csproj` or `nuspec` file is located.

includeReferencedProjects - Include referenced projects

boolean. Default value: `false`.

Includes referenced projects either as dependencies or as part of the package. Cannot be used with automatic package versioning. If a referenced project has a corresponding `nuspec` file that has the same name as the project, then that referenced project is added as a dependency. Otherwise, the referenced project is added as part of the package. Learn more about [using the pack command for NuGet CLI to create NuGet packages](#).

versionByBuild - Automatic package versioning

string. Required. Allowed values: `false` (Off), `byPrereleaseNumber` (Use the date and time), `byEnvVar` (Use an environment variable), `true` (Use the build number). Default value: `false`.

Applies automatic package versioning depending on the specified value. This string cannot be used with `includeReferencedProjects`. The allowed values are:

- **byPrereleaseNumber** - **Use the date and time**: The task will generate a [SemVer](#)-compliant version formatted as `X.Y.Z-ci-datetime`, where you specify the values of X, Y, and Z.
- **byEnvVar** - **Use an environment variable**: The task will use an environment variable that you specify and contains the version number you want to use.
- **true** - **Use the build number**: The task will use the build number to version the package.

Note

Under General, set the build format to be

`$(BuildDefinitionName)_$(Year:yyyy).$(Month).$(DayOfMonth)$(Rev:.r).`

`versionEnvVar` - Environment variable

`string`. Required when `versionByBuild = byEnvVar`.

Specifies the variable name without `$`, `$env`, or `%`.

`requestedMajorVersion` - Major

`string`. Required when `versionByBuild = byPrereleaseNumber`. Default value: `1`.

The `X` in version `X.Y.Z`.

`requestedMinorVersion` - Minor

`string`. Required when `versionByBuild = byPrereleaseNumber`. Default value: `0`.

The `Y` in version `X.Y.Z`.

`requestedPatchVersion` - Patch

`string`. Required when `versionByBuild = byPrereleaseNumber`. Default value: `0`.

The `Z` in version `X.Y.Z`.

`configurationToPack` - Configuration to Package

`string`. Default value: `$(BuildConfiguration)`.

Specifies the configuration to package when using a `csproj` file.

`buildProperties` - Additional build properties

`string`.

The semicolon delimited list of properties used to build the package.

`nuGetAdditionalArgs` - NuGet Arguments

`string`.

The additional arguments passed to `NuGet.exe pack`. Learn more about [using the pack command for NuGet CLI to create NuGet packages](#).

`nuGetPath` - Path to NuGet.exe

`string`.

Optional. Supplies the path to `NuGet.exe`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: Cmd
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.83.0 or greater
Task category	Package

NuGetPublisher@0 - NuGet publisher v0 task

Article • 09/26/2023

NuGetPublisher@0 is deprecated. Use the “NuGet” task instead. It works with the new Tool Installer framework so you can easily use new versions of NuGet without waiting for a task update, provides better support for authenticated feeds outside this organization/collection, and uses NuGet 4 by default.

Syntax

YAML

```
# NuGet publisher v0
# Deprecated: use the “NuGet” task instead. It works with the new Tool
Installer framework so you can easily use new versions of NuGet without
waiting for a task update, provides better support for authenticated feeds
outside this organization/collection, and uses NuGet 4 by default.
- task: NuGetPublisher@0
  inputs:
    searchPattern:
      '**/*.nupkg;-:**/packages/**/*.*.nupkg;-:**/*.*.symbols.nupkg' # string.
      Required. Path/Pattern to nupkg. Default:
      '**/*.nupkg;-:**/packages/**/*.*.nupkg;-:**/*.*.symbols.nupkg.
        nuGetFeedType: 'external' # 'external' | 'internal'. Required. Feed
        type. Default: external.
        connectedServiceName: # string. Required when nuGetFeedType = external.
        NuGet Service Connection.
        #feedName: # string. Required when nuGetFeedType = internal. Internal
        Feed URL.
        # Advanced
        #nuGetAdditionalArgs: # string. NuGet Arguments.
        #verbosity: '-' # '-' | 'Quiet' | 'Normal' | 'Detailed'. Verbosity.
        Default: -.
        nuGetVersion: '3.3.0' # '3.3.0' | '3.5.0.1829' | '4.0.0.2283' |
        'custom'. Required. NuGet Version. Default: 3.3.0.
        #nuGetPath: # string. Path to NuGet.exe.
        #continueOnEmptyNupkgMatch: false # boolean. Continue if no packages
        match the "Path/Pattern to nupkg". Default: false.
```

Inputs

`searchPattern` - Path/Pattern to nupkg

`string`. Required. Default value:

```
**/*.nupkg;-:**/packages/**/*.nupkg;-:**/*.*symbols.nupkg.
```

The pattern that the task uses to match or path to `nupkg` files to be uploaded. Multiple patterns can be separated by a semicolon.

`nuGetFeedType` - Feed type

`string`. Required. Allowed values: `external` (External NuGet Feed), `internal` (Internal NuGet Feed). Default value: `external`.

Specifies whether the target feed is an internal feed/collection or an external NuGet server.

`connectedServiceName` - NuGet Service Connection

`string`. Required when `nuGetFeedType = external`.

Specifies the NuGet server generic service connection. Set the key `Password/Token Key` field to your NuGet API key.

`feedName` - Internal Feed URL

`string`. Required when `nuGetFeedType = internal`.

Specifies the URL of a NuGet feed hosted in this account.

`nuGetAdditionalArgs` - NuGet Arguments

`string`.

The additional arguments passed to `NuGet.exe push`. Learn more about the [push command in the NuGet CLI](#).

`verbosity` - Verbosity

`string`. Allowed values: `-`, `Quiet`, `Normal`, `Detailed`. Default value: `-`.

Specifies the amount of detail displayed in the output.

`nuGetVersion` - NuGet Version

`string`. Required. Allowed values: `3.3.0`, `3.5.0.1829` (3.5.0), `4.0.0.2283` (4.0.0), `custom`. Default value: `3.3.0`.

Specifies the version of NuGet or a custom version to use.

nuGetPath - Path to NuGet.exe

`string`.

Optional. Supplies the path to `NuGet.exe`. Will override version selection.

continueOnEmptyNupkgMatch - Continue if no packages match the "Path/Pattern to nupkg"

`boolean`. Default value: `false`.

Continues the task instead of failing the task if no packages match the `searchPattern` string.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: Cmd
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any

Requirement	Description
Agent version	2.144.0 or greater
Task category	Package

NuGetRestore@1 - NuGet Restore v1 task

Article • 09/26/2023

Use this task to restore NuGet packages in preparation for a Visual Studio Build step.

This task will be deprecated soon, please switch to using the [NuGetCommand@2](#) `restore` option.

Syntax

YAML

```
# NuGet Restore v1
# Restores NuGet packages in preparation for a Visual Studio Build step.
- task: NuGetRestore@1
  inputs:
    solution: '**/*.sln' # string. Required. Path to solution,
    packages.config, or project.json. Default: **/*.sln.
    selectOrConfig: 'select' # 'select' | 'config'. Required. Feeds to use.
    Default: select.
    #feed: # string. Optional. Use when selectOrConfig = select. Use
    packages from this Azure Artifacts feed.
    #includeNuGetOrg: true # boolean. Optional. Use when selectOrConfig =
    select. Use packages from NuGet.org. Default: true.
    #nugetConfigPath: # string. Optional. Use when selectOrConfig = config.
    Path to NuGet.config.
    # Advanced
    #noCache: false # boolean. Disable local cache. Default: false.
    #packagesDirectory: # string. Destination directory.
    #verbosity: 'Detailed' # '-' | 'Quiet' | 'Normal' | 'Detailed'.
    Verbosity. Default: Detailed.
```

Inputs

`solution` - Path to solution, `packages.config`, or `project.json`

`string`. Required. Default value: `**/*.sln`.

The path to the solution, `packages.config`, or `project.json` file that references the packages to be restored.

`selectOrConfig` - Feeds to use

`string`. Required. Allowed values: `select` (Feed(s) I select here), `config` (Feeds in my NuGet.config). Default value: `select`.

Specifies the feed(s) to use. Specify one feed from VSTS and/or NuGet.org using the `select` value. Specify multiple feeds by committing a `nuget.config` file to your source code repository and setting its path with the `config` value.

`feed` - Use packages from this Azure Artifacts feed

`string`. Optional. Use when `selectOrConfig = select`.

Includes the specified VSTS feed in the generated `NuGet.config` file.

`includeNuGetOrg` - Use packages from NuGet.org

`boolean`. Optional. Use when `selectOrConfig = select`. Default value: `true`.

Includes the specified NuGet.org feed in the generated `NuGet.config`.

`nugetConfigPath` - Path to NuGet.config

`string`. Optional. Use when `selectOrConfig = config`.

Specifies the path to the `NuGet.config` in your repository that specifies the feeds from which to restore packages.

`noCache` - Disable local cache

`boolean`. Default value: `false`.

Prevents NuGet from using packages from local machine caches. Equivalent to the `-NoCache` `NuGet.exe` command line argument.

`packagesDirectory` - Destination directory

`string`.

Specifies the folder in which packages are installed. If no folder is specified, packages are restored into a `packages/` folder alongside the selected solution, `packages.config`, or `project.json`. Equivalent to the `-PackagesDirectory` `NuGet.exe` command line argument.

verbosity - Verbosity

`string`. Allowed values: `-`, `Quiet`, `Normal`, `Detailed`. Default value: `Detailed`.

Specifies the amount of detail displayed in the output.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Package

NuGetToolInstaller@1 - NuGet tool installer v1 task

Article • 09/26/2023

Acquires a specific version of NuGet from the internet or the tools cache and adds it to the PATH. Use this task to change the version of NuGet used in the NuGet tasks.

Syntax

YAML

```
# NuGet tool installer v1
# Acquires a specific version of NuGet from the internet or the tools cache
# and adds it to the PATH. Use this task to change the version of NuGet used
# in the NuGet tasks.
- task: NuGetToolInstaller@1
  inputs:
    # Advanced
    #versionSpec: # string. Version of NuGet.exe to install.
    #checkLatest: false # boolean. Always check for new versions. Default:
    #false.
```

Inputs

`versionSpec` - Version of NuGet.exe to install

`string`.

A version or version range that specifies the NuGet version to make available on the path. Use x as a wildcard. See the [list of available NuGet versions ↗](#).

If you want to match a pre-release version, the specification must contain a major, minor, patch, and pre-release version from the list above. If a version isn't specified, then one will be chosen automatically.

Examples: `4.x`, `3.3.x`, `2.8.6`, `>=4.0.0-0`.

`checkLatest` - Always check for new versions

`boolean`. Default value: `false`.

When this boolean is set to `true`, the task always checks for and downloads the latest available version of `NuGet.exe` that satisfies the version spec. This option will also always incur download time, even if the selected version of NuGet is already cached.

Enabling this option could cause unexpected build breaks when a new version of NuGet is released.

Tip

If you're using [the Microsoft-hosted agents](#), you should leave this set to false. Microsoft updates the Microsoft-hosted agents on a regular basis, but they're often slightly behind the latest version. Enabling this parameter could result in your build spending a lot of time updating to a newer minor version.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to find, download, and cache a specified version of [NuGet](#) and add it to the PATH. For information on the tools cache, see the [azure-pipelines-tool-lib](#) repo.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: NuGet

Requirement	Description
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Tool

See also

For an explanation of tool installers and examples, see [Tool installers](#).

NuGetToolInstaller@0 - NuGet tool installer v0 task

Article • 09/26/2023

Use this task to find, download, and cache a specified version of [NuGet](#) and add it to the PATH. For information on the tools cache, see the [azure-pipelines-tool-lib](#) repo.

Syntax

YAML

```
# NuGet tool installer v0
# Acquires a specific version of NuGet from the internet or the tools cache
# and adds it to the PATH. Use this task to change the version of NuGet used
# in the NuGet tasks.
- task: NuGetToolInstaller@0
  inputs:
    #versionSpec: # string. Version of NuGet.exe to install.
    #checkLatest: false # boolean. Always download the latest matching
    #version. Default: false.
```

Inputs

versionSpec - Version of NuGet.exe to install

`string.`

A version or version range that specifies the NuGet version to make available on the path. Use x as a wildcard. See the [list of available NuGet versions](#).

If you want to match a pre-release version, the specification must contain a major, minor, patch, and pre-release version from the list above. If a version isn't specified, then one will be chosen automatically.

Examples: `4.x`, `3.3.x`, `2.8.6`, `>=4.0.0-0`.

checkLatest - Always download the latest matching version

`boolean.` Default value: `false`.

When this boolean is set to `true`, the task always checks for and downloads the latest available version of `NuGet.exe` that satisfies the version spec. This option will also always

incur download time, even if the selected version of NuGet is already cached.

Enabling this option could cause unexpected build breaks when a new version of NuGet is released.

Tip

If you're using [the Microsoft-hosted agents](#), you should leave this set to false.

Microsoft updates the Microsoft-hosted agents on a regular basis, but they're often slightly behind the latest version. Enabling this parameter could result in your build spending a lot of time updating to a newer minor version.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: NuGet
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Tool

HelmDeploy@0 - Package and deploy Helm charts v0 task

Article • 09/26/2023

Use this task to deploy, configure, or update a Kubernetes cluster in Azure Container Service by running helm commands.

Syntax

YAML

```
# Package and deploy Helm charts v0
# Deploy, configure, update a Kubernetes cluster in Azure Container Service
by running helm commands.
- task: HelmDeploy@0
  inputs:
    # Kubernetes Cluster
    #connectionType: 'Azure Resource Manager' # 'Azure Resource Manager' |
    'Kubernetes Service Connection' | 'None'. Required when command != logout &&
    command != package && command != save. Connection Type. Default: Azure
    Resource Manager.
    #azureSubscription: # string. Alias: azureSubscriptionEndpoint. Required
    when connectionType = Azure Resource Manager && command != logout && command
    != package && command != save. Azure subscription.
    #azureResourceGroup: # string. Required when connectionType = Azure
    Resource Manager && command != logout && command != package && command !=
    save. Resource group.
    #kubernetesCluster: # string. Required when connectionType = Azure
    Resource Manager && command != logout && command != package && command !=
    save. Kubernetes cluster.
    #useClusterAdmin: false # boolean. Optional. Use when connectionType =
    Azure Resource Manager && command != logout && command != package && command
    != save. Use cluster admin credentials. Default: false.
    #kubernetesServiceConnection: # string. Alias:
    kubernetesServiceEndpoint. Required when connectionType = Kubernetes Service
    Connection && command != logout && command != package && command != save.
    Kubernetes Service Connection.
    #namespace: # string. Optional. Use when command != logout && command !=
    package && command != save. Namespace.
    # Azure Container Registry
    #azureSubscriptionForACR: # string. Alias:
    azureSubscriptionEndpointForACR. Required when command == save. Azure
    subscription for Container Registry.
    #azureResourceGroupForACR: # string. Required when command == save.
    Resource group.
    #azureContainerRegistry: # string. Required when command == save. Azure
    Container Registry.
    # Commands
```

```
    command: 'ls' # 'create' | 'delete' | 'expose' | 'get' | 'init' |
'install' | 'login' | 'logout' | 'ls' | 'package' | 'rollback' | 'save' |
'upgrade' | 'uninstall'. Required. Command. Default: ls.
    #chartType: 'Name' # 'Name' | 'FilePath'. Required when command ==
install || command == upgrade. Chart Type. Default: Name.
    chartName: # string. Required when chartType == Name. Chart Name.
    #chartPath: # string. Required when chartType == FilePath || command ==
package. Chart Path.
    #chartVersion: # string. Alias: version. Optional. Use when command ==
package || command == install || command == upgrade. Version.
    #releaseName: # string. Optional. Use when command == install || command
== upgrade. Release Name.
    #overrideValues: # string. Optional. Use when command == install ||
command == upgrade. Set Values.
    #valueFile: # string. Optional. Use when command == install || command
== upgrade. Value File.
    #destination: '$(Build.ArtifactStagingDirectory)' # string. Optional.
Use when command == package. Destination. Default:
$(Build.ArtifactStagingDirectory).
    #canaryimage: false # boolean. Optional. Use when command == init. Use
canary image version. Default: false.
    #upgradetiller: true # boolean. Optional. Use when command == init.
Upgrade Tiller. Default: true.
    #updatedependency: false # boolean. Optional. Use when command ==
install || command == package. Update Dependency. Default: false.
    #save: true # boolean. Optional. Use when command == package. Save.
Default: true.
    #install: true # boolean. Optional. Use when command == upgrade. Install
if release not present. Default: true.
    #recreate: false # boolean. Optional. Use when command == upgrade.
Recreate Pods. Default: false.
    #resetValues: false # boolean. Optional. Use when command == upgrade.
Reset Values. Default: false.
    #force: false # boolean. Optional. Use when command == upgrade. Force.
Default: false.
    #waitForExecution: true # boolean. Optional. Use when command == init ||
command == install || command == upgrade. Wait. Default: true.
    #arguments: # string. Optional. Use when command != login && command !=
logout. Arguments.
    #chartNameForACR: # string. Required when command == save. Chart Name
For Azure Container Registry.
    #chartPathForACR: # string. Required when command == save. Chart Path
for Azure Container Registry.
    # TLS
    #enableTls: false # boolean. Optional. Use when command != login &&
command != logout && command != package && command != save. Enable TLS.
Default: false.
    #caCert: # string. Required when enableTls == true && command != login
&& command != logout && command != package && command != save. CA
certificate.
    #certificate: # string. Required when enableTls == true && command !=
login && command != logout && command != package && command != save.
Certificate.
    #privatekey: # string. Required when enableTls == true && command !=
login && command != logout && command != package && command != save. Key.
```

```
# Advanced
#tillernamespace: # string. Optional. Use when command != login &&
command != logout && command != package && command != save. Tiller
namespace.
#failOnStderr: false # boolean. Optional. Use when command != login &&
command != logout && command != package && command != save. Fail on Standard
Error. Default: false.
#publishPipelineMetadata: true # boolean. Optional. Use when command !=
login && command != logout && command != package && command != save. Publish
pipeline metadata. Default: true.
```

Inputs

`connectionType` - Connection Type

`string`. Required when `command != logout && command != package && command != save`. Allowed values: `Azure Resource Manager`, `Kubernetes Service Connection`, `None`. Default value: `Azure Resource Manager`.

Specifies the connection type.

- `Kubernetes Service Connection` - Specify `Kubernetes Service Connection` to connect to any Kubernetes cluster by using `kubeconfig` or the Azure Service Account. Allows you to provide a KubeConfig file, specify a Service Account, or import an AKS instance with the `Azure Subscription` option. Importing an AKS instance with the `Azure Subscription` option requires Kubernetes cluster access at Service Connection configuration time.
- `Azure Resource Manager` - Specify `Azure Resource Manager` to connect to an Azure Kubernetes Service by using Azure Service Connection. Does not access Kubernetes cluster at Service Connection configuration time.
- `None` - Use a pre-created Kubernetes configuration stored locally.

For more information, see [Service connection](#) in the following [Remarks](#) section.

`azureSubscription` - Azure subscription

Input alias: `azureSubscriptionEndpoint`. `string`. Required when `connectionType = Azure Resource Manager && command != logout && command != package && command != save`.

The name of the Azure Service Connection. Specify an Azure subscription that has your container registry.

azureResourceGroup - Resource group

```
string. Required when connectionType = Azure Resource Manager && command != logout  
&& command != package && command != save.
```

The name of the resource group within the subscription. Specify an Azure Resource Group.

kubernetesCluster - Kubernetes cluster

```
string. Required when connectionType = Azure Resource Manager && command != logout  
&& command != package && command != save.
```

The name of the AKS cluster. Specify an Azure Managed Cluster.

useClusterAdmin - Use cluster admin credentials

```
boolean. Optional. Use when connectionType = Azure Resource Manager && command !=  
logout && command != package && command != save. Default value: false.
```

Uses cluster administrator credentials instead of default cluster user credentials.

kubernetesServiceConnection - Kubernetes Service Connection

Input alias: `kubernetesServiceEndpoint`. string. Required when connectionType =
Kubernetes Service Connection && command != logout && command != package && command
!= save.

Specifies a Kubernetes Service Connection.

namespace - Namespace

```
string. Optional. Use when command != logout && command != package && command !=  
save.
```

The namespace on which you run the `kubectl` commands. If not specified, the task uses the default namespace. Specify the Kubernetes namespace to use. You can specify the Tiller namespace in the advanced section of the task or by passing the `--tiller-namespace` option as an argument.

azureSubscriptionForACR - Azure subscription for Container Registry

Input alias: `azureSubscriptionEndpointForACR`. string. Required when command == save.

Specifies an Azure subscription that has your Azure Container Registry.

azureResourceGroupForACR - Resource group

`string`. Required when `command == save`.

Specifies an Azure Resource Group that has your Container Registry.

azureContainerRegistry - Azure Container Registry

`string`. Required when `command == save`.

Specifies an Azure Container Registry to be used for pushing Helm charts.

command - Command

`string`. Required. Allowed values: `create`, `delete`, `expose`, `get`, `init`, `install`, `login`, `logout`, `ls`, `package`, `rollback`, `save`, `upgrade`, `uninstall`. Default value: `ls`.

Specifies a Helm command.

chartType - Chart Type

`string`. Required when `command == install || command == upgrade`. Allowed values: `Name`, `FilePath` (File Path). Default value: `Name`.

Specifies how you want to enter chart information. You can either provide the name of the chart or folder/file path to the chart.

chartName - Chart Name

`string`. Required when `chartType == Name`.

The name of the chart reference to install. This can be a url or a chart name. For example, if the chart name is `stable/mysql`, the task runs `helm install stable/mysql`.

chartPath - Chart Path

`string`. Required when `chartType == FilePath || command == package`.

The path to the chart to install. This can be a path to a packaged chart or a path to an unpacked chart directory. For example, if you specify `./redis`, the task runs `helm install ./redis`. If you're consuming a chart that's published as an artifact, then the path will be `$(System.DefaultWorkingDirectory)/ARTIFACT-NAME/Charts/CHART-NAME`.

chartVersion - Version

`Input alias: version. string.` Optional. Use when `command == package || command == install || command == upgrade.`

Specifies the exact chart version to install. If you don't specify the chart version, the task installs the latest version. Set the version on the chart to this semver version.

releaseName - Release Name

`string.` Optional. Use when `command == install || command == upgrade.`

The release name. If you don't specify the release name, the task autogenerated one for you. The `releaseName` input is only valid for `install` and `upgrade` commands.

overrideValues - Set Values

`string.` Optional. Use when `command == install || command == upgrade.`

Specifies values on the command line. This input can specify multiple or separate values with commas: `key1=val1,key2=val2`.

You can also specify multiple values by delimiting them with a new line, as follows:

- `key1=val1`
- `key2=val2`

If you have a value that contains new lines, use the `valueFile` option. Otherwise, the task treats the new line as a delimiter. The task constructs the Helm command by using these set values. For example, you can set the value using a command like the following:

```
helm install --set key1=val1 ./redis.
```

valueFile - Value File

`string.` Optional. Use when `command == install || command == upgrade.`

Specifies values in a YAML file or a URL. For example, specifying `myvalues.yaml` results in `helm install --values=myvals.yaml`.

destination - Destination

`string.` Optional. Use when `command == package.` Default value:
`$(Build.ArtifactStagingDirectory).`

Specifies values in a YAML file or a URL.

canaryimage - Use canary image version.

`boolean`. Optional. Use when `command == init`. Default value: `false`.

Specifies the canary Tiller image. Use the latest pre-release version of Tiller.

upgradetiller - Upgrade Tiller

`boolean`. Optional. Use when `command == init`. Default value: `true`.

If `true`, this input upgrades Tiller if Tiller is already installed.

updatedependency - Update Dependency

`boolean`. Optional. Use when `command == install || command == package`. Default value: `false`.

If `true`, this input updates a Helm dependency update before installing the chart.

Updates dependencies from `requirements.yaml` to the `charts/` directory before packaging.

save - Save

`boolean`. Optional. Use when `command == package`. Default value: `true`.

Saves the packaged chart to the local chart repository when set to `true`.

install - Install if release not present.

`boolean`. Optional. Use when `command == upgrade`. Default value: `true`.

If a release by this name doesn't already exist, this input runs an install.

recreate - Recreate Pods.

`boolean`. Optional. Use when `command == upgrade`. Default value: `false`.

Performs pods restart for the resource, if applicable.

resetValues - Reset Values.

`boolean`. Optional. Use when `command == upgrade`. Default value: `false`.

Resets the values to the values built into the chart.

force - Force

`boolean`. Optional. Use when `command == upgrade`. Default value: `false`.

Forces a resource update through a delete or recreate action, if needed.

waitForExecution - Wait

`boolean`. Optional. Use when `command == init || command == install || command == upgrade`. Default value: `true`.

Blocks the action until the command execution completes.

arguments - Arguments

`string`. Optional. Use when `command != login && command != logout`.

The Helm command options.

enableTls - Enable TLS

`boolean`. Optional. Use when `command != login && command != logout && command != package && command != save`. Default value: `false`.

Enables using SSL between Helm and Tiller.

caCert - CA certificate

`string`. Required when `enableTls == true && command != login && command != logout && command != package && command != save`.

The CA cert used to issue a certificate for the Tiller and Helm client.

certificate - Certificate

`string`. Required when `enableTls == true && command != login && command != logout && command != package && command != save`.

Specify the Tiller certificate or the Helm client certificate.

privatekey - Key

`string`. Required when `enableTls == true && command != login && command != logout && command != package && command != save`.

Specify the Tiller key or the Helm client key.

tillernamespace - Tiller namespace

`string`. Optional. Use when `command != login && command != logout && command != package && command != save`.

Specify Tiller's Kubernetes namespace.

failOnStderr - Fail on Standard Error

`boolean`. Optional. Use when `command != login && command != logout && command != package && command != save`. Default value: `false`.

If this input is `true`, this task fails if any errors are written to the error pipeline, or if any data is written to the Standard Error stream. Otherwise, the task relies on the exit code to determine failure.

publishPipelineMetadata - Publish pipeline metadata

`boolean`. Optional. Use when `command != login && command != logout && command != package && command != save`. Default value: `true`.

If this input is `true`, the task collects and publishes deployment metadata.

chartNameForACR - Chart Name For Azure Container Registry

`string`. Required when `command == save`.

The chart's name in the Azure Container Registry.

chartPathForACR - Chart Path for Azure Container Registry

`string`. Required when `command == save`.

The file path to the chart directory in the Azure Container Registry.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

`helmExitCode`

The exit code emitted from the execution of specified Helm command.

`helmOutput`

The output emitted from the execution of specified Helm command.

Remarks

Use HelmDeploy@0 to deploy, configure, or update a Kubernetes cluster in Azure Container Service by running Helm commands. Helm is a tool that streamlines deploying and managing Kubernetes apps using a packaging format called charts.

You can define, version, share, install, and upgrade even the most complex Kubernetes app by using Helm.

- Helm helps you combine multiple Kubernetes manifests (yaml) such as service, deployments, configmaps, and more into a single unit called Helm Charts. You don't need to either invent or use a tokenization or a templating tool.
- Helm Charts help you manage application dependencies and deploy as well as rollback as a unit. They are also easy to create, version, publish, and share with other partner teams.

Azure Pipelines has built-in support for Helm charts:

- The [Helm Tool installer task](#) can be used to install the correct version of Helm onto the agents.
- The Helm package and deploy task can be used to package the app and deploy it to a Kubernetes cluster. You can use the task to install or update Tiller to a Kubernetes namespace, to securely connect to Tiller over TLS for deploying charts, or to run any Helm command such as `lint`.
- The Helm task supports connecting to an Azure Kubernetes Service by using an Azure service connection. You can connect to any Kubernetes cluster by using `kubeconfig` or a service account.

- Helm deployments can be supplemented by using the **Kubectl** task; for example, `create/update`, `imagepullsecret`, and others.

Service connection

HelmDeploy@0 works with two service connection types: **Azure Resource Manager** and **Kubernetes Service Connection**. See [Examples](#) for examples on configuring these two connection types.

 **Note**

A service connection isn't required if an environment resource that points to a Kubernetes cluster has already been specified in the pipeline's stage.

Kubernetes Service Connection considerations when accessing AKS

You can create a Kubernetes service connection with any of the following options.

- KubeConfig
- Service Account
- Azure Subscription

New Kubernetes service connection

Authentication method

- KubeConfig
- Service Account
- Azure Subscription

When selecting the **Azure Subscription** option, Kubernetes needs to be accessible to Azure DevOps at service connection configuration time. There may be various reasons a service connection cannot be created, for example you created a private cluster or the cluster has local accounts disabled. In these cases, Azure DevOps is unable to connect to your cluster at service connection configuration time and you will observe the dialog to be stuck at **Loading namespaces**.



Loading namespaces

Starting with Kubernetes 1.24, long-lived tokens are [no longer created by default](#).

Kubernetes recommends not to use long-lived tokens. As a result, tasks using a Kubernetes service connection created using the Azure Subscription option do not have access to the permanent token required to authenticate and can't access your Kubernetes cluster. This also results in the **Loading namespaces** dialog to be frozen.

Use the Azure Resource Manager Service Connection to access AKS

For AKS customers, the Azure Resource Manager service connection type provides the best method to connect to a private cluster, or a cluster that has local accounts disabled. This method is not dependent on cluster connectivity at the time you create a service connection. Access to AKS is deferred to pipeline runtime, which has the following advantages:

- Access to a (private) AKS cluster can be performed from a self-hosted or scale set agent with line of sight to the cluster.
- A token is created for every task that uses an Azure Resource Manager service connection. This ensures you are connecting to Kubernetes with a short-lived token, which is the [Kubernetes recommendation](#).
- AKS can be accessed even when local accounts are disabled.

Service connection FAQ

I receive the following error message: Could not find any secret associated with the service account. What is happening?

You are using the Kubernetes service connection with Azure Subscription option. We are updating this method to create long-lived tokens. This is expected to be available mid-May. However, it is recommended to start using the Azure service connection type and not to use long-lived tokens as per [Kubernetes guidance](#).

I'm using AKS and don't want to change anything, can I continue to use tasks with the Kubernetes service connection?

We are updating this method to create long-lived tokens. This is expected to be available mid-May. However, please be aware that this approach is against [Kubernetes guidance](#).

I'm using the Kubernetes tasks and Kubernetes service connection but not AKS. Should I be concerned?

Your tasks will continue to work as before.

Will the Kubernetes service connection type be removed?

Our Kubernetes tasks work with any Kubernetes cluster, regardless where they are running. The Kubernetes service connection will continue to exist.

I'm an AKS customer and everything is running fine, should I act?

There is no need to change anything. If you are using the Kubernetes service connection and selected Azure Subscription during creation, you should be aware of the [Kubernetes guidance on using long-lived tokens](#).

I'm creating a Kubernetes Environment, and have no option to use service connections

In case you can't access your AKS during environment creation time, you can use an empty environment and set the `connectionType` input to an Azure Resource Manager service connection.

I have AKS configured with Azure Active Directory RBAC, and my pipeline doesn't work. Will these updates resolve that?

Accessing Kubernetes when AAD RBAC is enabled is unrelated to token creation. To prevent an interactive prompt, we will support [kubelogin](#) in a future update.

Command values

The command input accepts one of the following [helm commands](#):

create/delete/expose/get/init/install/login/logout/ls/package/rollback/upgrade.

Examples are provided in the [Examples](#) section.

Each command input maps to a set of task inputs. The commands that map to a task input are designated in the YAML syntax block and in the task inputs table

Troubleshooting

HelmDeploy task throws error 'unknown flag: --wait' while running 'helm init --wait --client-only' on Helm 3.0.2 version.

There are some breaking changes between Helm 2 and Helm 3. One of them includes removal of tiller, and hence `helm init` command is no longer supported. Remove command: init when you use Helm 3.0+ versions.

When using Helm 3, if System.debug is set to true and Helm upgrade is the command being used, the pipeline fails even though the upgrade was successful.

This is a known issue with Helm 3, as it writes some logs to stderr. Helm Deploy Task is marked as failed if there are logs to stderr or exit code is non-zero. Set the task input failOnStderr: false to ignore the logs printed to stderr.

Examples

Azure Resource Manager

This YAML example shows how Azure Resource Manager is used to refer to the Kubernetes cluster. This is used with one of the helm [commands](#) and the appropriate values required for the command:

YAML

```
variables:
  azureSubscriptionEndpoint: Contoso
  azureContainerRegistry: contoso.azurecr.io
  azureResourceGroup: Contoso
  kubernetesCluster: Contoso

- task: HelmDeploy@0
  displayName: Helm deploy
  inputs:
    connectionType: Azure Resource Manager
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureResourceGroup: $(azureResourceGroup)
    kubernetesCluster: $(kubernetesCluster)
```

Kubernetes Service Connection

This YAML example shows how Kubernetes service connection is used to refer to the Kubernetes cluster. This is used with one of the helm [commands](#) and the appropriate values required for the command:

YAML

```
- task: HelmDeploy@0
  displayName: Helm deploy
  inputs:
    connectionType: Kubernetes Service Connection
    kubernetesServiceEndpoint: Contoso
```

Commands

The command input accepts one of the following [helm commands](#):
create/delete/expose/get/init/install/login/logout/ls/package/rollback/upgrade.

This YAML example demonstrates the ls command:

YAML

```
- task: HelmDeploy@0
  displayName: Helm list
  inputs:
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureResourceGroup: $(azureResourceGroup)
    kubernetesCluster: $(kubernetesCluster)
    command: ls
    arguments: --all
```

init command

This YAML example demonstrates the init command:

YAML

```
- task: HelmDeploy@0
  displayName: Helm init
  inputs:
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureResourceGroup: $(azureResourceGroup)
    kubernetesCluster: $(kubernetesCluster)
    command: init
    upgradetiller: true
    waitForExecution: true
    arguments: --client-only
```

install command

This YAML example demonstrates the **install** command:

YAML

```
- task: HelmDeploy@0
  displayName: Helm install
  inputs:
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureResourceGroup: $(azureResourceGroup)
    kubernetesCluster: $(kubernetesCluster)
    command: install
    chartType: FilePath
    chartPath: Application/charts/sampleapp
```

package command

This YAML example demonstrates the **package** command:

YAML

```
- task: HelmDeploy@0
  displayName: Helm package
  inputs:
    command: package
    chartPath: Application/charts/sampleapp
    destination: $(Build.ArtifactStagingDirectory)
```

upgrade command

This YAML example demonstrates the **upgrade** command:

YAML

```
- task: HelmDeploy@0
  displayName: Helm upgrade
  inputs:
    azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
    azureResourceGroup: $(azureResourceGroup)
    kubernetesCluster: $(kubernetesCluster)
    command: upgrade
    chartType: filepath
    chartPath: $(Build.ArtifactStagingDirectory)/sampleapp-v0.2.0.tgz
    releaseName: azureddevopsdemo
    install: true
    waitForExecution: false
```

save command

This YAML example demonstrates the `save` command:

YAML

```
- task: HelmDeploy@0
  displayName: Helm save
  inputs:
    command: save
    chartNameForACR: mycontainerregistry.azurecr.io/helm/hello-world:v1
    chartPathForACR: Application/charts/sampleapp
    azureSubscriptionEndpointForACR: $(azureSubscriptionEndpointForACR)
    azureResourceGroupForACR: $(azureResourceGroupForACR)
    azureContainerRegistry: $(azureContainerRegistry)
```

Package and sign Helm charts

In this section you'll learn how to package and sign Helm charts in a pipeline.

Generate a private-public key pair to sign the helm chart using GPG

1. Download [GPG](#).
2. Launch the command prompt in an administrator mode. Run the following command to generate a private-public key pair to sign the helm chart using gpg. While creating the key, you'll be prompted for the username and email address. The "name email address" is later used to name the private-public key pair that is created.

Windows Command Prompt

```
gpg --full-generate-key
```

```
C:\WINDOWS\system32>gpg --full-generate-key
gpg (GnuPG) 2.2.11; Copyright (C) 2018 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
 (1) RSA and RSA (default)
 (2) DSA and Elgamal
 (3) DSA (sign only)
 (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
Requested keysize is 2048 bits
Please specify how long the key should be valid.
    0 = key does not expire
    <n>  = key expires in n days
    <n>w = key expires in n weeks
    <n>m = key expires in n months
    <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y

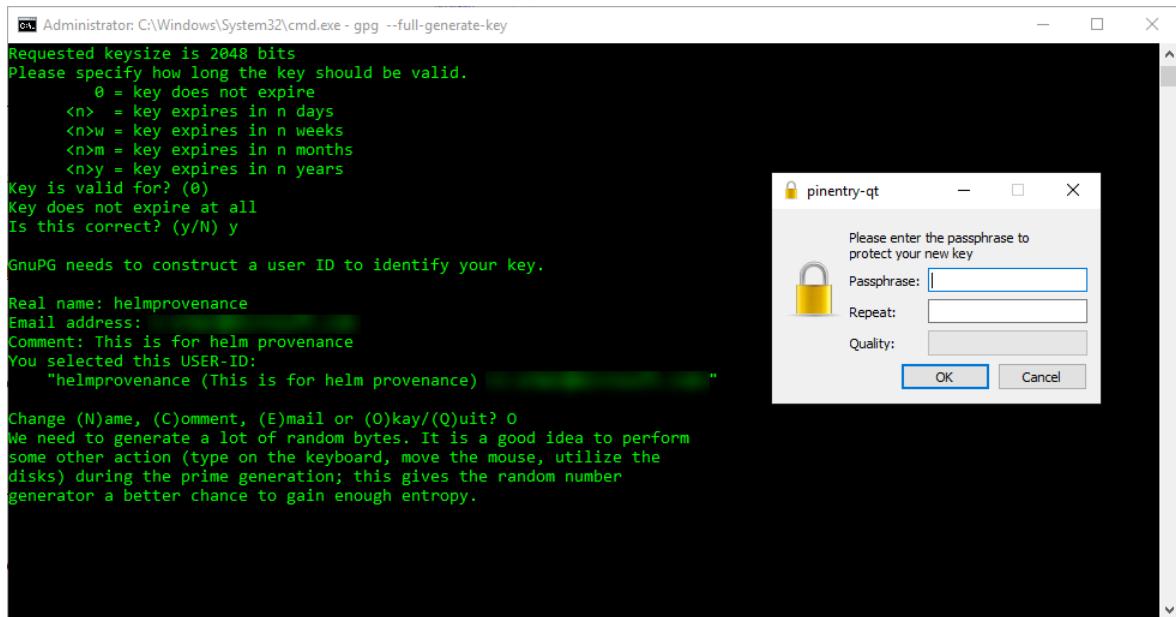
GnuPG needs to construct a user ID to identify your key.

Real name: helmprovance
Email address: [REDACTED]
Comment: This is for helm provenance
You selected this USER-ID:
    "helmprovance (This is for helm provenance)"

Change (N)ame, (C)omment, (E)mail or (O)key/(Q)uit? 0
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key 8D713E39690BDE89 marked as ultimately trusted
gpg: revocation certificate stored as 'C:/Users/srivatsam/AppData/Roaming/gnupg/openpgp-revocs.d\296EB600AB393EAAA3F9B7F98D713E39690BDE89.rev'
public and secret key created and signed.

pub    rsa2048 2018-11-30 [SC]
uid            helmprovance (This is for helm provenance) [REDACTED]
sub    rsa2048 2018-11-30 [E]
```

3. You'll be prompted for the passphrase. Give the value and click ok.



4. After creating the key, you can see the list of keys which contains both private and public using the following command.

- To see list of private keys

Windows Command Prompt

```
gpg --list-secret-keys
```

```
C:\Users\srivatsam>gpg --list-secret-keys  
C:/Users/srivatsam/AppData/Roaming/gnupg/pubring.kbx  
-----  
sec rsa2048 2018-11-30 [SC]  
          [ultimate] contoso (This is for helm)  
sub rsa2048 2018-11-30 [E]  
  
C:\Users\srivatsam>
```

- To see the list of public keys

Windows Command Prompt

```
gpg --list-keys
```

```
C:\Users\srivatsam>gpg --list-keys  
C:/Users/srivatsam/AppData/Roaming/gnupg/pubring.kbx  
-----  
pub rsa2048 2018-11-30 [SC]  
          [ultimate] contoso (This is for helm)  
sub rsa2048 2018-11-30 [E]  
  
C:\Users\srivatsam>
```

5. Store the private and public keys in 2 different files with the extension **gpg** as shown below.

- For a private key

Windows Command Prompt

```
gpg --export-secret-key 94325E18E53EDD99DD8339C3CFD9DAF0707CB788  
contoso@microsoft.com > C:/somepath/privatekeys.gpg
```

You'll see the **privatekeys.gpg** file exported to the path which was mentioned above.

- For a public key

Windows Command Prompt

```
gpg --export-key 94325E18E53EDD99DD8339C3CFD9DAF0707CB788  
contoso@microsoft.com > C:/somepath/publickey.gpg
```

You'll see the **publickey.gpg** file exported to the path which was mentioned above.

In Azure DevOps, save the `privatekey.gpg` file in the library **secure files** section.

Example

YAML

```
pool:
  name: Hosted Ubuntu 1604

variables:
  # The below variable should be secure
  HelmKeyPassphrase: contoso@123
  keyName: contoso contoso@microsoft.com
  azureSubscriptionEndpoint: contoso
  azureResourceGroup: contoso
  kubernetesCluster: contoso

steps:
  - task: DownloadSecureFile@1
    displayName: Download Secure file
    inputs:
      secureFile: privatekey.gpg
      name: privateKeyRing

  - task: HelmInstaller@0
    displayName: Install Helm 2.12.0
    inputs:
      helmVersion: 2.12.0

  - task: HelmDeploy@0
    displayName: helm init
    inputs:
      azureSubscriptionEndpoint: $(azureSubscriptionEndpoint)
      azureResourceGroup: $(azureResourceGroup)
      kubernetesCluster: $(kubernetesCluster)
      command: init
      arguments: --client-only

  - task: HelmDeploy@0
    displayName: helm package
    inputs:
      command: package
      chartPath: Application/charts/sampleapp
      arguments: --sign --key "$(keyName)" --keyring
      $(privateKeyRing.secureFilePath)
    env:
      HelmKeyPassphrase: $(HelmKeyPassphrase)
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Deploy

PowerShell@2 - PowerShell v2 task

Article • 09/26/2023

Use this task to run a PowerShell script on Linux, macOS, or Windows.

ⓘ Note

By default, PowerShell v2 uses PowerShell Core for Linux agents and Windows PowerShell for Windows agents. To use the latest version of PowerShell on Windows agents, set the `pwsh` parameter to `true`. PowerShell Core will then be used instead.

Syntax

YAML

```
# PowerShell v2
# Run a PowerShell script on Linux, macOS, or Windows.
- task: PowerShell@2
  inputs:
    targetType: 'filePath' # 'filePath' | 'inline'. Type. Default:
  filePath.
    filePath: # string. Required when targetType = filePath. Script Path.
    #arguments: # string. Optional. Use when targetType = filePath.
  Arguments.
    #script: # string. Required when targetType = inline. Script.
  # Preference Variables
    #errorActionPreference: 'stop' # 'default' | 'stop' | 'continue' |
  'silentlyContinue'. ErrorActionPreference. Default: stop.
    #warningPreference: 'default' # 'default' | 'stop' | 'continue' |
  'silentlyContinue'. WarningPreference. Default: default.
    #informationPreference: 'default' # 'default' | 'stop' | 'continue' |
  'silentlyContinue'. InformationPreference. Default: default.
    #verbosePreference: 'default' # 'default' | 'stop' | 'continue' |
  'silentlyContinue'. VerbosePreference. Default: default.
    #debugPreference: 'default' # 'default' | 'stop' | 'continue' |
  'silentlyContinue'. DebugPreference. Default: default.
    #progressPreference: 'silentlyContinue' # 'default' | 'stop' |
  'continue' | 'silentlyContinue'. ProgressPreference. Default:
  silentlyContinue.
  # Advanced
    #failOnStderr: false # boolean. Fail on Standard Error. Default: false.
    #showWarnings: false # boolean. Show warnings as Azure DevOps warnings.
  Default: false.
    #ignoreLASTEXITCODE: false # boolean. Ignore $LASTEXITCODE. Default:
  false.
    #pwsh: false # boolean. Use PowerShell Core. Default: false.
```

```
#workingDirectory: # string. Working Directory.  
#runScriptInSeparateScope: false # boolean. Run script in the separate  
scope. Default: false.
```

Inputs

`targetType` - Type

`string`. Allowed values: `filePath` (File Path), `inline`. Default value: `filePath`.

Specifies the type of script for the task to run: an inline script or a path to a `.ps1` file.

`filePath` - Script Path

`string`. Required when `targetType = filePath`.

Specifies the path of the script to execute. Must be a fully qualified path or relative to `$(System.DefaultWorkingDirectory)`.

`arguments` - Arguments

`string`. Optional. Use when `targetType = filePath`.

Specifies the arguments passed to the PowerShell script. Arguments can be ordinal parameters or named parameters. For example, `-Name someName -Path -Value "Some long string value"`.

`arguments` is not used when `targetType` is set to `inline`.

`script` - Script

`string`. Required when `targetType = inline`. Default value: `# Write your PowerShell commands here.\n\nWrite-Host "Hello World"`.

Specifies the contents of the script. The maximum supported inline script length is 20000 characters. Use a script from a file if you want to use a longer script.

`errorActionPreference` - ErrorActionPreference

`string`. Allowed values: `default`, `stop`, `continue`, `silentlyContinue`. Default value: `stop`.

Prepends the line `$ErrorActionPreference = 'VALUE'` at the top of your script.

warningPreference - WarningPreference

`string`. Allowed values: `default`, `stop`, `continue`, `silentlyContinue`. Default value: `default`.

When not set to `Default`, prepends the line `$WarningPreference = 'VALUE'` at the top of your script.

informationPreference - InformationPreference

`string`. Allowed values: `default`, `stop`, `continue`, `silentlyContinue`. Default value: `default`.

When not set to `Default`, prepends the line `$InformationPreference = 'VALUE'` at the top of your script.

verbosePreference - VerbosePreference

`string`. Allowed values: `default`, `stop`, `continue`, `silentlyContinue`. Default value: `default`.

When not set to `Default`, prepends the line `$VerbosePreference = 'VALUE'` at the top of your script.

debugPreference - DebugPreference

`string`. Allowed values: `default`, `stop`, `continue`, `silentlyContinue`. Default value: `default`.

When not set to `Default`, prepends the line `$DebugPreference = 'VALUE'` at the top of your script.

progressPreference - ProgressPreference

`string`. Allowed values: `default`, `stop`, `continue`, `silentlyContinue`. Default value: `silentlyContinue`.

When not set to `Default`, prepends the line `$ProgressPreference = 'VALUE'` at the top of your script.

failOnStderr - Fail on Standard Error

`boolean`. Default value: `false`.

If the value of this boolean is `true`, the task fails if any errors are written to the error pipeline or if any data is written to the Standard Error stream. Otherwise, the task relies on the exit code to determine failure.

`showWarnings` - Show warnings as Azure DevOps warnings

`boolean`. Default value: `false`.

If the value is set to `true`, and your script writes warnings, then the warnings will appear as warnings in Pipeline logs.

`ignoreLASTEXITCODE` - Ignore \$LASTEXITCODE

`boolean`. Default value: `false`.

If the value is set to `false`, the line `if ((Test-Path -LiteralPath variable:\LASTEXITCODE)) { exit $LASTEXITCODE }` is appended to the end of your script. This will cause the last exit code from an external command to be propagated as the exit code of `powershell`. Otherwise, the line is not appended to the end of your script.

`pwsh` - Use PowerShell Core

`boolean`. Default value: `false`.

If this is true, then tasks running on Windows agents will use `pwsh.exe` from your path instead of `powershell.exe`.

`workingDirectory` - Working Directory

`string`.

Specifies the working directory where the script is run. If a value is not specified, the working directory is `$(Build.SourcesDirectory)`.

`runScriptInSeparateScope` - Run script in the separate scope

`boolean`. Default value: `false`.

This input allows executing PowerShell scripts using `&` operator instead of the default `.`. If this input is set to `true`, the script will be executed in a separate scope, and globally scoped PowerShell variables won't be updated.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Each PowerShell session lasts only for the duration of the job in which it runs. Tasks that depend on what has been bootstrapped must be in the same job as the bootstrap.

Task shortcuts

`PowerShell@2` has two shortcuts in YAML: `steps.powershell` and `steps.pwsh`.

- `powershell` runs using either Windows PowerShell (on Windows) or `pwsh` (Linux and macOS).
- `pwsh` runs PowerShell Core, the cross-platform edition of PowerShell built on .NET Core.

```
yml
steps:
- powershell: # Run a script in Windows PowerShell on Windows, and pwsh on Linux and macOS.
- pwsh: # Run a script in PowerShell Core on Windows, macOS, and Linux.
```

Set a variable so it can be read by subsequent scripts and tasks

To learn more about defining build variables in a script, see [Define and modify your build variables in a script](#).

To learn more about defining release variables in a script, see [Define and modify your release variables in a script](#).

Passing pipeline secrets in script, but secret is not masked in pipeline logs

Be aware that PowerShell cuts off error messages, so if you use pipeline secrets in a script, the secrets could be trimmed and exposed. For example, in the inline script below:

```
PowerShell
```

```
./script.ps1 --arg1 value1 --arg2 <some_secret_which_will_be_masked_here>
```

There could be an exception like: At <path_to_temp_script_file>:4 char:3:

```
PowerShell
```

```
+ ./script.ps1 --arg1 value1 --arg2 <unmasked_part_of_original_secret> ...
+ ~~~~~
+ <Additional exception details>
```

To avoid this issue, you can handle these exceptions on a script level, or avoid cases when pipeline secrets could appear in source code lines within error messages.

Examples

- [Invoke a script from a file](#)
- [Write a warning](#)
- [Write an error](#)
- [Call PowerShell script with multiple arguments](#)

Invoke a script from a file

The following is a sample PowerShell file named `test.ps1` located in the root of your repository.

```
PowerShell
```

```
Write-Host "Hello World from $Env:AGENT_NAME."
Write-Host "My ID is $Env:AGENT_ID."
Write-Host "AGENT_WORKFOLDER contents:"
gci $Env:AGENT_WORKFOLDER
Write-Host "AGENT_BUILDDIRECTORY contents:"
gci $Env:AGENT_BUILDDIRECTORY
Write-Host "BUILD_SOURCESDIRECTORY contents:"
```

```
gci $Env:BUILD_SOURCESDIRECTORY  
Write-Host "Over and out."
```

You can invoke this script in your pipeline like this.

```
yml  
  
steps:  
- task: PowerShell@2  
  inputs:  
    targetType: 'filePath'  
    filePath: `test.ps1`
```

Write a warning

```
yml  
  
- task: PowerShell@2  
  inputs:  
    targetType: 'inline'  
    script: Write-Host "##vso[task.LogIssue type=warning;]This is the  
warning"  
    # Writes a warning to build summary and to log in yellow text
```

Write an error

```
yml  
  
- task: PowerShell@2  
  inputs:  
    targetType: 'inline'  
    script: Write-Host "##vso[task.LogIssue type=error;]This is the error"  
    # Writes an error to build summary and to log in red text
```

If you want this error to fail the build, add `exit 1` to the script.

```
yml  
  
- task: PowerShell@2  
  inputs:  
    targetType: 'inline'  
    script: |  
      Write-Host "##vso[task.LogIssue type=error;]This is the error"  
      exit 1  
    # Writes an error to build summary and to log in red text
```

Call PowerShell script with multiple arguments

Create PowerShell script `test2.ps1`:

```
PowerShell

param ($input1, $input2)
Write-Host "$input1 $input2"
```

In your YAML pipeline, call:

```
YAML

- task: PowerShell@2
  inputs:
    targetType: 'filePath'
    filePath: $(System.DefaultWorkingDirectory)\test2.ps1
    arguments: > # Use this to avoid newline characters in multiline string
      -input1 "Hello"
      -input2 "World"
  displayName: 'Print Hello World'
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.115.0 or greater
Task category	Utility

See also

- [Use a PowerShell script to customize your pipeline - ApplyVersionToAssemblies.ps1](#)
- Learn more about PowerShell scripts

- [Scripting with Windows PowerShell](#)
- [Microsoft Script Center \(the Scripting Guys\)](#) ↗
- [PowerShell.org](#) ↗

PowerShell@1 - PowerShell v1 task

Article • 09/26/2023

Run a PowerShell script.

Syntax

YAML

```
# PowerShell v1
# Run a PowerShell script.
- task: PowerShell@1
  inputs:
    scriptType: 'filePath' # 'inlineScript' | 'filePath'. Required. Type.
    Default: filePath.
    scriptName: # string. Required when scriptType = filePath. Script Path.
    #arguments: # string. Arguments.
    #inlineScript: # string. Required when scriptType = inlineScript. Inline
    Script.
    # Advanced
    #workingFolder: # string. Working folder.
    #failOnStandardError: true # boolean. Fail on Standard Error. Default:
    true.
```

Inputs

`scriptType` - Type

`string`. Required. Allowed values: `inlineScript` (Inline Script), `filePath` (File Path).

Default value: `filePath`.

Specifies the type of script for the task to run: an inline script or a path to a `.ps1` file.

`scriptName` - Script Path

`string`. Required when `scriptType = filePath`.

Specifies the type of script for the task to run: an inline script or a path to a `.ps1` file.

`arguments` - Arguments

`string`.

Specifies the arguments passed to the PowerShell script. Arguments can be ordinal parameters or named parameters. For example, `-Name someName -Path -Value "Some long string value"`.

`arguments` is not used when `targetType` is set to `inline`.

`workingFolder` - Working folder

`string`.

Specifies the working directory where the script is run. If a value is not specified, the working directory is `$(Build.SourcesDirectory)`.

`inlineScript` - Inline Script

`string`. Required when `scriptType = inlineScript`. Default value: `# You can write your powershell scripts inline here. \n# You can also pass predefined and custom variables to this scripts using arguments\n\n Write-Host "Hello World"`.

Specifies the contents of the script. The maximum supported inline script length is 500 characters. Use a script from a file if you want to use a longer script.

`failOnStandardError` - Fail on Standard Error

`boolean`. Default value: `true`.

If the value of this boolean is `true`, the task fails if any errors are written to the error pipeline or if any data is written to the Standard Error stream. Otherwise, the task relies on the exit code to determine failure.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

`PowerShell@1` runs only on Windows agents. To run PowerShell on other agent types, use `PowerShell@2`.

Each PowerShell session lasts only for the duration of the job in which it runs. Tasks that depend on what has been bootstrapped must be in the same job as the bootstrap.

Set a variable so it can be read by subsequent scripts and tasks

To learn more about defining build variables in a script, see [Define and modify your build variables in a script](#).

To learn more about defining release variables in a script, see [Define and modify your release variables in a script](#).

Passing pipeline secrets in script, but secret is not masked in pipeline logs

Be aware that PowerShell cuts off error messages, so if you use pipeline secrets in a script, the secrets could be trimmed and exposed. For example, in the inline script below:

```
PowerShell  
./script.ps1 --arg1 value1 --arg2 <some_secret_which_will_be_masked_here>
```

There could be an exception like: `At <path_to_temp_script_file>:4 char:3:`

```
PowerShell  
+ ./script.ps1 --arg1 value1 --arg2 <unmasked_part_of_original_secret> ...  
+ ~~~~~  
+ <Additional exception details>
```

To avoid this issue, you can handle these exceptions on a script level, or avoid cases when pipeline secrets could appear in source code lines within error messages.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release

Requirement	Description
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: DotNetFramework
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.102 or greater
Task category	Utility

See also

- [Use a PowerShell script to customize your pipeline - ApplyVersionToAssemblies.ps1](#)
- Learn more about PowerShell scripts
 - [Scripting with Windows PowerShell](#)
 - [Microsoft Script Center \(the Scripting Guys\)](#) ↗
 - [PowerShell.org](#) ↗

PowerShellOnTargetMachines@3 - PowerShell on target machines v3 task

Article • 09/26/2023

Use this task to execute PowerShell scripts on remote machines using PSSession and Invoke-Command for remoting.

Syntax

YAML

```
# PowerShell on target machines v3
# Execute PowerShell scripts on remote machines using PSSession and Invoke-
#Command for remoting.
- task: PowerShellOnTargetMachines@3
  inputs:
    Machines: # string. Required. Machines.
    #UserName: # string. Username.
    #UserPassword: # string. Password.
    # Script options
    #ScriptType: 'Inline' # 'FilePath' | 'Inline'. Script Type. Default:
    Inline.
    #ScriptPath: # string. Required when ScriptType = FilePath. Script File
    Path.
    InlineScript: # string. Required when ScriptType = Inline. Script.
    #ScriptArguments: # string. Optional. Use when ScriptType = FilePath.
    Script Arguments.
    #InitializationScript: # string. Optional. Use when ScriptType =
    FilePath. Initialization script.
    #SessionVariables: # string. Optional. Use when ScriptType = FilePath.
    Session Variables.
    # PSSession options
    #CommunicationProtocol: 'Https' # 'Http' | 'Https'. Protocol. Default:
    Https.
    #AuthenticationMechanism: 'Default' # 'Default' | 'Credssp'.
    Authentication. Default: Default.
    #NewPsSessionOptionArguments: '-SkipCACheck -IdleTimeout 7200000 -
    OperationTimeout 0 -OutputBufferingMode Block' # string. Session Option
    parameters. Default: -SkipCACheck -IdleTimeout 7200000 -OperationTimeout 0 -
    OutputBufferingMode Block.
    # Error handling options
    #ErrorActionPreference: 'stop' # 'stop' | 'continue' |
    'silentlyContinue'. ErrorActionPreference. Default: stop.
    #failOnStderr: false # boolean. Fail on Standard Error. Default: false.
    #ignoreLASTEXITCODE: false # boolean. Ignore $LASTEXITCODE. Default:
    false.
    # Advanced
    #WorkingDirectory: # string. Working Directory.
```

```
#RunPowerShellInParallel: true # boolean. Run PowerShell in Parallel.  
Default: true.
```

Inputs

Machines - Machines

`string`. Required.

Specifies a comma-separated list of machine FQDNs or IP addresses, and optionally includes the port number. Can be:

- The name of an [Azure Resource Group](#).
- A comma-delimited list of machine names. Example:
`dbserver.fabrikam.com,dbserver_int.fabrikam.com:5986,192.168.34:5986`
- An output variable from a previous task.

If you do not specify a port, the default WinRM port is used. This depends on the protocol you have configured. For WinRM 2.0, the default HTTP port is `5985` and the default HTTPS port is `5986`.

UserName - Username

`string`.

Specifies the username of a domain or a local administrative account on the target host(s).

- Formats such as `username`, `domain\username`, `machine-name\username`, and `.\username` are supported.
- UPN formats such as `username@domain.com` and built-in system accounts such as `NT Authority\System` are not supported.

UserPassword - Password

`string`.

Specifies the password for the target machines. Variables defined in build/release definitions as `$(passwordVariable)` are accepted. You may mark the variable type as `secret` to secure it.

ScriptType - Script Type

`string`. Allowed values: `FilePath` (File Path), `Inline`. Default value: `Inline`.

Specifies the type of script to execute: Inline or File Path.

ScriptPath - Script File Path

`string`. Required when `ScriptType = FilePath`.

Specifies the location of the PowerShell script on the target machines or on a UNC path, like `C:\BudgetIT\Web\Deploy\Website.ps1`, which should be accessible from the target machine.

InlineScript - Script

`string`. Required when `ScriptType = Inline`. Default value: `# Write your powershell commands here.\n\nWrite-Output "Hello World".`

ScriptArguments - Script Arguments

`string`. Optional. Use when `ScriptType = FilePath`.

Specifies the arguments for the PowerShell script. Can be ordinal or named parameters, like `-testParam test`. For example: `-applicationPath $(applicationPath)`, `-username $(vmusername)`, `-password $(vmpassword)`.

InitializationScript - Initialization script

`string`. Optional. Use when `ScriptType = FilePath`.

Specifies the location of the data script for the DSC on the target machines or on a UNC path, like `C:\BudgetIT\Web\Deploy\WebsiteConfiguration.ps1`. It's recommended to use arguments instead of an initialization script.

SessionVariables - Session Variables

`string`. Optional. Use when `ScriptType = FilePath`.

Used to set up the session variables for the PowerShell scripts.

Specifies a comma-separated list, such as `$varx=valuex, $vary=valuey`. Most commonly used for backward compatibility with earlier versions of the release service. It's recommended to use arguments instead of session variables.

CommunicationProtocol - Protocol

`string`. Allowed values: `Http`, `Https`. Default value: `Https`.

Specifies the protocol to use for the WinRM service connection with the machine(s). The default value is `HTTPS`.

AuthenticationMechanism - Authentication

`string`. Allowed values: `Default`, `Credssp`. Default value: `Default`.

Specifies the authentication mechanism used for creating the PSSession. For `CredSSP` authentication, the username and password fields are mandatory.

NewPsSessionOptionArguments - Session Option parameters

`string`. Default value: `-SkipCACheck -IdleTimeout 7200000 -OperationTimeout 0 -OutputBufferingMode Block`.

Advanced options for a remote session (`New-PSSessionOption`). For example, `-SkipCACheck`, `-SkipCNCheck`, `-SkipRevocationCheck`, etc. See a [complete list of all session options](#) to learn more.

ErrorActionPreference - ErrorActionPreference

`string`. Allowed values: `stop`, `continue`, `silentlyContinue`. Default value: `stop`.

Prepends the line `$ErrorActionPreference = 'VALUE'` at the top of your script.

failOnStderr - Fail on Standard Error

`boolean`. Default value: `false`.

If set to `true`, fails if any errors are written to the error pipeline or if any data is written to the Standard Error stream. Otherwise, the task relies on the exit code to determine failure.

ignoreLASTEXITCODE - Ignore \$LASTEXITCODE

`boolean`. Default value: `false`.

If set to `false`, the line `if ((Test-Path -LiteralPath variable:\LASTEXITCODE)) { exit $LASTEXITCODE }` is executed at the end of your script. This causes the last exit code from

an external command to be propagated as the exit code of PowerShell. Otherwise, the line is not executed to the end of your script.

WorkingDirectory - Working Directory

`string`.

Specifies the working directory where the script is run.

RunPowershellInParallel - Run PowerShell in Parallel

`boolean`. Default value: `true`.

If set to `true`, runs the PowerShell scripts in parallel on the target machines.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to execute PowerShell scripts on remote machine(s).

This task can run both PowerShell scripts and PowerShell-DSC scripts:

- For PowerShell scripts, the computers must have PowerShell 2.0 or higher installed.
- For PowerShell-DSC scripts, the computers must have [the latest version of the Windows Management Framework](#) installed. This is installed by default on Windows 8.1, Windows Server 2012 R2, and subsequent versions.

Prerequisites

This task uses [Windows Remote Management](#) (WinRM) to access on-premises physical computers or virtual computers that are domain-joined or workgroup-joined.

[To set up WinRM for on-premises physical computers or virtual machines](#)

Follow the steps described in [domain-joined](#)

To set up WinRM for Microsoft Azure Virtual Machines

Azure Virtual Machines require WinRM to use the HTTPS protocol. You can use a self-signed Test Certificate. In this case, the automation agent will not validate the authenticity of the certificate as being issued by a trusted certification authority.

- **Azure Classic Virtual Machines.** When you create a [classic virtual machine](#) from the Azure portal, the virtual machine is already set up for WinRM over HTTPS, with the default port 5986 already opened in the firewall and a self-signed certificate installed on the machine. These virtual machines can be accessed with no further configuration required. Existing Classic virtual machines can be also selected by using the [Azure Resource Group Deployment](#) task.
- **Azure Resource Group.** If you have an [Azure Resource Group](#) already defined in the Azure portal, you must configure it to use the WinRM HTTPS protocol. You need to open port 5986 in the firewall, and install a self-signed certificate.

To dynamically deploy Azure Resource Groups that contain virtual machines, use the [Azure Resource Group Deployment](#) task. This task has a checkbox named **Enable Deployment Prerequisites**. Select this to automatically set up the WinRM HTTPS protocol on the virtual machines, open port 5986 in the firewall, and install a test certificate. The virtual machines are then ready for use in the deployment task.

What's new in this task version

- Uses PSSession and invoke-command to perform remoting on target machines.
- Added support for inline script execution.
- Default and CredSSP authentication are supported.
- Added options for error handling: `ErrorActionPreference`, `ignoreLASTEXITCODE` and `Fail on Standard Error`.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None

Requirement	Description
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.134.0 or greater
Task category	Deploy

PowerShellOnTargetMachines@2 - PowerShell on Target Machines v2 task

Article • 09/26/2023

Use this task to execute PowerShell scripts on remote machine(s).

Syntax

YAML

```
# PowerShell on Target Machines v2
# Execute PowerShell scripts on remote machine(s).
- task: PowerShellOnTargetMachines@2
  inputs:
    EnvironmentName: # string. Required. Machines.
    #AdminUserName: # string. Admin Login.
    #AdminPassword: # string. Password.
    #Protocol: # 'Http' | 'Https'. Protocol.
    #TestCertificate: true # boolean. Optional. Use when Protocol = Https.
    Test Certificate. Default: true.
    # Deployment
    ScriptPath: # string. Required. PowerShell Script.
    #ScriptArguments: # string. Script Arguments.
    #InitializationScriptPath: # string. Initialization Script.
    #SessionVariables: # string. Session Variables.
    # Advanced Options
    #RunPowershellInParallel: true # boolean. Run PowerShell in Parallel.
    Default: true.
    #ResourceFilteringMethod: 'machineNames' # 'machineNames' | 'tags'.
    Select Machines By. Default: machineNames.
    #MachineNames: # string. Filter Criteria.
```

Inputs

EnvironmentName - Machines

`string`. Required.

Specifies a comma-separated list of machine IP addresses or FQDNs, along with ports.

The default port is based on the selected protocol.

For example:

`dbserver.fabrikam.com, dbserver_int.fabrikam.com:5986, 192.168.12.34:5986`

You can also provide the output variable of other tasks, for example `$(variableName)`. If

you're using HTTPS, the name or IP of the machine should match the CN in the certificate.

AdminUserName - Admin Login

`string`.

Specifies the administrator login for the target machines.

For example: `Domain\Admin User`, `Admin User@Domain`, `.\Admin User`.

AdminPassword - Password

`string`.

Specifies the administrator password for the target machines. Variables defined in build/release definitions as `$(passwordVariable)` are accepted. You may mark the variable type as `secret` to secure it.

Protocol - Protocol

`string`. Allowed values: `Http`, `Https`.

Specifies the protocol to use for the WinRM service connection with the machine(s). The default value is `HTTPS`.

TestCertificate - Test Certificate

`boolean`. Optional. Use when `Protocol = Https`. Default value: `true`.

Specifies the option to skip validating the authenticity of the machine's certificate by a trusted certification authority. The parameter is required for the WinRM HTTPS protocol.

ScriptPath - PowerShell Script

`string`. Required.

Specifies the location of the PowerShell script on the target machines or on a UNC path, like `C:\BudgetIT\Web\Deploy\Website.ps1`.

ScriptArguments - Script Arguments

`string`.

Specifies the arguments for the PowerShell script. Can be ordinal or named parameters, like `-testParam test`.

InitializationScriptPath - Initialization Script

`string`.

Specifies the location of the data script for DSC on the target machines or on a UNC path, like `C:\BudgetIT\Web\Deploy\WebsiteConfiguration.ps1`.

SessionVariables - Session Variables

`string`.

Specifies the common session variables for both scripts. For example, `$variable = value` or `$var1 = "value, 123"`.

RunPowershellInParallel - Run PowerShell in Parallel

`boolean`. Default value: `true`.

If set to `true`, runs the PowerShell scripts in parallel on the target machines.

ResourceFilteringMethod - Select Machines By

`string`. Allowed values: `machineNames` (Machine Names), `tags`. Default value: `machineNames`.

Optional. Specifies a subset of machines by providing machine names or tags.

MachineNames - Filter Criteria

`string`.

This input is valid only for machine groups or output variables. It is not supported for a flat list of machines yet.

Specifies a list of machines, like `dbserver.fabrikam.com, webserver.fabrikam.com, 192.168.12.34` or tags, like `Role:DB; OS:Win8.1`. If multiple tags are specified, the task will run in all machines with the specified tags. The default runs the task in all machines.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

What's new in Version 2.0:

- Removed support of legacy DTL machines.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.104.0 or greater
Task category	Deploy

PowerShellOnTargetMachines@1 - PowerShell on Target Machines v1 task

Article • 09/26/2023

Use this task to execute PowerShell scripts on remote machine(s).

Syntax

YAML

```
# PowerShell on Target Machines v1
# Execute PowerShell scripts on remote machine(s).
- task: PowerShellOnTargetMachines@1
  inputs:
    EnvironmentName: # string. Required. Machines.
    #AdminUserName: # string. Admin Login.
    #AdminPassword: # string. Password.
    #Protocol: # 'Http' | 'Https'. Protocol.
    #TestCertificate: true # boolean. Optional. Use when Protocol = Https.
    Test Certificate. Default: true.
    # Deployment
    ScriptPath: # string. Required. PowerShell Script.
    #ScriptArguments: # string. Script Arguments.
    #InitializationScriptPath: # string. Initialization Script.
    #SessionVariables: # string. Session Variables.
    # Advanced Options
    #RunPowershellInParallel: true # boolean. Run PowerShell in Parallel.
    Default: true.
    #ResourceFilteringMethod: 'machineNames' # 'machineNames' | 'tags'.
    Select Machines By. Default: machineNames.
    #MachineNames: # string. Filter Criteria.
```

Inputs

EnvironmentName - Machines

`string`. Required.

Specifies a comma-separated list of machine IP addresses or FQDNs, along with ports.

The default port is based on the selected protocol.

For example:

`dbserver.fabrikam.com, dbserver_int.fabrikam.com:5986, 192.168.12.34:5986`

You can also provide the output variable of other tasks, for example `$(variableName)`. If

you're using HTTPS, the name or IP of the machine should match the CN in the certificate.

AdminUserName - Admin Login

`string`.

Specifies the administrator login for the target machines.

AdminPassword - Password

`string`.

Specifies the administrator password for the target machines. Variables defined in build/release definitions as `$(passwordVariable)` are accepted. You may mark the variable type as `secret` to secure it.

Protocol - Protocol

`string`. Allowed values: `Http`, `Https`.

Specifies the protocol to use for the WinRM connection with the machine(s). The default value is `HTTPS`.

TestCertificate - Test Certificate

`boolean`. Optional. Use when `Protocol = Https`. Default value: `true`.

Skips validating the authenticity of the machine's certificate by a trusted certification authority. The parameter is required for the WinRM HTTPS protocol.

ScriptPath - PowerShell Script

`string`. Required.

Specifies the location of the PowerShell script on the target machines or on a UNC path, like `C:\BudgetIT\Web\Deploy\Website.ps1`.

ScriptArguments - Script Arguments

`string`.

Specifies the arguments for the PowerShell script. Can be ordinal or named parameters, like `-testParam test`.

InitializationScriptPath - Initialization Script

`string`.

Specifies the location of the data script for DSC on the target machines or on a UNC path, like `c:\BudgetIT\Web\Deploy\WebsiteConfiguration.ps1`.

SessionVariables - Session Variables

`string`.

Specifies the common session variables for both scripts. For example, `$variable = value` or `$var1 = "value, 123"`.

RunPowershellInParallel - Run PowerShell in Parallel

`boolean`. Default value: `true`.

If set to `true`, runs the PowerShell scripts in parallel on the target machines.

ResourceFilteringMethod - Select Machines By

`string`. Allowed values: `machineNames` (Machine Names), `tags`. Default value: `machineNames`.

Optional. Specifies a subset of machines by providing machine names or tags.

MachineNames - Filter Criteria

`string`.

This input is valid only for machine groups or output variables. It is not supported for a flat list of machines yet.

Specifies a list of machines, like `dbserver.fabrikam.com, webserver.fabrikam.com, 192.168.12.34` or tags, like `Role:DB; OS:Win8.1`. If multiple tags are specified, the task will run in all machines with the specified tags. The default runs the task in all machines.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.104.0 or greater
Task category	Deploy

SonarQubePrepare@5 - Prepare Analysis Configuration v5 task

Article • 09/26/2023

Use this task to prepare a SonarQube analysis configuration.

Syntax

YAML

```
# Prepare Analysis Configuration v5
# Prepare SonarQube analysis configuration.
- task: SonarQubePrepare@5
  inputs:
    SonarQube: # string. Required. SonarQube Server Endpoint.
    scannerMode: 'MSBuild' # 'MSBuild' | 'Other' | 'CLI'. Required. Choose
the way to run the analysis. Default: MSBuild.
    #configMode: 'file' # 'file' | 'manual'. Required when scannerMode =
CLI. Mode. Default: file.
    #configFile: 'sonar-project.properties' # string. Optional. Use when
scannerMode = CLI && configMode = file. Settings File. Default: sonar-
project.properties.
    #cliProjectKey: # string. Required when scannerMode = CLI && configMode
= manual. Project Key.
    projectKey: # string. Required when scannerMode = MSBuild. Project Key.
    #cliProjectName: # string. Optional. Use when scannerMode = CLI &&
configMode = manual. Project Name.
    # projectName: # string. Optional. Use when scannerMode = MSBuild.
Project Name.
    #cliProjectVersion: '1.0' # string. Optional. Use when scannerMode = CLI
&& configMode = manual. Project Version. Default: 1.0.
    #projectVersion: '1.0' # string. Optional. Use when scannerMode =
MSBuild. Project Version. Default: 1.0.
    #cliSources: '.' # string. Required when scannerMode = CLI && configMode
= manual. Sources directory root. Default: ..
    # Advanced
    #extraProperties: # string. Additional Properties.
```

Inputs

SonarQube - SonarQube Server Endpoint

string. Required.

Specifies the SonarQube server endpoint for your project. To create one, click the [Manage](#) link, create a new SonarQube Server Endpoint, and enter your server url and

token.

scannerMode - Choose the way to run the analysis

`string`. Required. Allowed values: `MSBuild` (Integrate with MSBuild), `Other` (Integrate with Maven or Gradle), `CLI` (Use standalone scanner). Default value: `MSBuild`.

MSBuild

- Put this task before your MSBuild task.
- Add the `Run Code Analysis` task after the MSBuild/VSTest tasks.

Maven/Gradle

- Put this task before the Maven/Gradle task.
- Tick the `Run SonarQube Analysis` checkbox in the Maven/Gradle task configuration.

Others

- For other cases, you can use the standalone scanner (`sonar-scanner`), set all configurations with this task, and then add the `Run Code Analysis` task.

configMode - Mode

`string`. Required when `scannerMode = CLI`. Allowed values: `file` (Store configuration with my source code (`sonar-project.properties`)), `manual` (Manually provide configuration). Default value: `file`.

Specifies your preferred configuration method.

configFile - Settings File

`string`. Optional. Use when `scannerMode = CLI && configMode = file`. Default value: `sonar-project.properties`.

Specifies the configuration settings and project properties. Learn more about the [SonarQube Extension for Azure DevOps ↗](#).

cliProjectKey - Project Key

`string`. Required when `scannerMode = CLI && configMode = manual`.

Specifies the SonarQube project unique key. For example, `sonar.projectKey`.

projectKey - Project Key

`string`. Required when `scannerMode = MSBuild`.

Specifies the SonarQube project unique key. For example, `sonar.projectKey`.

cliProjectName - Project Name

`string`. Optional. Use when `scannerMode = CLI && configMode = manual`.

Specifies the SonarQube project name. For example, `sonar.projectName`.

 projectName - Project Name

`string`. Optional. Use when `scannerMode = MSBuild`.

Specifies the SonarQube project name. For example, `sonar.projectName`.

cliProjectVersion - Project Version

`string`. Optional. Use when `scannerMode = CLI && configMode = manual`. Default value:
`1.0`.

Specifies the SonarQube project version. For example, `sonar.projectVersion`.

projectVersion - Project Version

`string`. Optional. Use when `scannerMode = MSBuild`. Default value: `1.0`.

Specifies the SonarQube project version. For example, `sonar.projectVersion`.

cliSources - Sources directory root

`string`. Required when `scannerMode = CLI && configMode = manual`. Default value: `..`.

Specifies the path to the root directory containing source files. This value is set to the `sonar.sources` SonarQube property.

extraProperties - Additional Properties

`string`. Default value: `# Additional properties that will be passed to the scanner,`
`\n# Put one key=value per line, example:\n# sonar.exclusions=**/*.bin.`

Specifies [additional properties](#) to be passed to the scanner. Specify each `key=value` pair on a new line.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

- **Support non-MSBuild projects:** This task can also configure analysis for non-MSBuild projects.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Build

See also

- [SonarQube Azure DevOps Integration](#)

SonarQubePrepare@4 - Prepare Analysis Configuration v4 task

Article • 09/26/2023

Use this task to prepare a SonarQube analysis configuration.

Syntax

YAML

```
# Prepare Analysis Configuration v4
# Prepare SonarQube analysis configuration.
- task: SonarQubePrepare@4
  inputs:
    SonarQube: # string. Required. SonarQube Server Endpoint.
    scannerMode: 'MSBuild' # 'MSBuild' | 'Other' | 'CLI'. Required. Choose
the way to run the analysis. Default: MSBuild.
    #configMode: 'file' # 'file' | 'manual'. Required when scannerMode =
CLI. Mode. Default: file.
    #configFile: 'sonar-project.properties' # string. Optional. Use when
scannerMode = CLI && configMode = file. Settings File. Default: sonar-
project.properties.
    #cliProjectKey: # string. Required when scannerMode = CLI && configMode
= manual. Project Key.
    projectKey: # string. Required when scannerMode = MSBuild. Project Key.
    #cliProjectName: # string. Optional. Use when scannerMode = CLI &&
configMode = manual. Project Name.
    # projectName: # string. Optional. Use when scannerMode = MSBuild.
Project Name.
    #cliProjectVersion: '1.0' # string. Optional. Use when scannerMode = CLI
&& configMode = manual. Project Version. Default: 1.0.
    #projectVersion: '1.0' # string. Optional. Use when scannerMode =
MSBuild. Project Version. Default: 1.0.
    #cliSources: '.' # string. Required when scannerMode = CLI && configMode
= manual. Sources directory root. Default: ..
    # Advanced
    #extraProperties: # string. Additional Properties.
```

Inputs

SonarQube - SonarQube Server Endpoint

string. Required.

Specifies the SonarQube server endpoint for your project. To create one, click the [Manage](#) link, create a new SonarQube Server Endpoint, and enter your server url and

token.

scannerMode - Choose the way to run the analysis

`string`. Required. Allowed values: `MSBuild` (Integrate with MSBuild), `Other` (Integrate with Maven or Gradle), `CLI` (Use standalone scanner). Default value: `MSBuild`.

MSBuild

- Put this task before your MSBuild task.
- Add the `Run Code Analysis` task after the MSBuild/VSTest tasks.

Maven/Gradle

- Put this task before the Maven/Gradle task.
- Tick the `Run SonarQube Analysis` checkbox in the Maven/Gradle task configuration.

Others

- For other cases, you can use the standalone scanner (`sonar-scanner`), set all configurations with this task, and then add the `Run Code Analysis` task.

configMode - Mode

`string`. Required when `scannerMode = CLI`. Allowed values: `file` (Store configuration with my source code (`sonar-project.properties`)), `manual` (Manually provide configuration). Default value: `file`.

Specifies your preferred configuration method.

configFile - Settings File

`string`. Optional. Use when `scannerMode = CLI && configMode = file`. Default value: `sonar-project.properties`.

Specifies the configuration settings and project properties. Learn more about the [SonarQube Extension for Azure DevOps ↗](#).

cliProjectKey - Project Key

`string`. Required when `scannerMode = CLI && configMode = manual`.

Specifies the SonarQube project unique key. For example, `sonar.projectKey`.

projectKey - Project Key

`string`. Required when `scannerMode = MSBuild`.

Specifies the SonarQube project unique key. For example, `sonar.projectKey`.

cliProjectName - Project Name

`string`. Optional. Use when `scannerMode = CLI && configMode = manual`.

Specifies the SonarQube project name. For example, `sonar.projectName`.

 projectName - Project Name

`string`. Optional. Use when `scannerMode = MSBuild`.

Specifies the SonarQube project name. For example, `sonar.projectName`.

cliProjectVersion - Project Version

`string`. Optional. Use when `scannerMode = CLI && configMode = manual`. Default value:
`1.0`.

Specifies the SonarQube project version. For example, `sonar.projectVersion`.

projectVersion - Project Version

`string`. Optional. Use when `scannerMode = MSBuild`. Default value: `1.0`.

Specifies the SonarQube project version. For example, `sonar.projectVersion`.

cliSources - Sources directory root

`string`. Required when `scannerMode = CLI && configMode = manual`. Default value: `..`.

Specifies the path to the root directory containing source files. This value is set to the `sonar.sources` SonarQube property.

extraProperties - Additional Properties

`string`. Default value: `# Additional properties that will be passed to the scanner,`
`\n# Put one key=value per line, example:\n# sonar.exclusions=**/*.bin.`

Specifies [additional properties](#) to be passed to the scanner. Specify each `key=value` pair on a new line.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

- **Support non-MSBuild projects:** This task can also configure analysis for non-MSBuild projects.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.95.1 or greater
Task category	Build

See also

- [SonarQube Azure DevOps Integration](#)

PublishBuildArtifacts@1 - Publish build artifacts v1 task

Article • 09/26/2023

Use this task in a build pipeline to publish build artifacts to Azure Pipelines, TFS, or a file share.

Syntax

YAML

```
# Publish build artifacts v1
# Publish build artifacts to Azure Pipelines or a Windows file share.
- task: PublishBuildArtifacts@1
  inputs:
    PathtoPublish: '$(Build.ArtifactStagingDirectory)' # string. Required.
    Path to publish. Default: $(Build.ArtifactStagingDirectory).
    ArtifactName: 'drop' # string. Required. Artifact name. Default: drop.
    publishLocation: 'Container' # 'Container' | 'FilePath'. Alias:
    ArtifactType. Required. Artifact publish location. Default: Container.
    #MaxArtifactSize: '0' # string. Max Artifact Size. Default: 0.
    #TargetPath: # string. Required when ArtifactType = FilePath. File share
    path.
    #Parallel: false # boolean. Optional. Use when ArtifactType = FilePath.
    Parallel copy. Default: false.
    #ParallelCount: '8' # string. Optional. Use when ArtifactType = FilePath
    && Parallel = true. Parallel count. Default: 8.
    # Advanced
    #StoreAsTar: false # boolean. Tar the artifact before uploading.
    Default: false.
```

Inputs

PathtoPublish - Path to publish

string. Required. Default value: \$(Build.ArtifactStagingDirectory).

Specifies the folder or file path to publish. This can be a fully qualified path or a path relative to the root of the repository. Wildcards are not supported. [Variables](#) are supported. Example: \$(Build.ArtifactStagingDirectory). For more information, see [Artifacts in pipelines - overview](#).

ArtifactName - Artifact name

`string`. Required. Default value: `drop`.

Specifies the name of the artifact to create in the publish location. The following special characters are not allowed: `+`, `%`, `{`, `}`

publishLocation - Artifact publish location

Input alias: `ArtifactType`. `string`. Required. Allowed values: `Container` (Azure Pipelines), `FilePath` (A file share). Default value: `Container`.

Specifies whether to store the artifact in Azure Pipelines (Container), or to copy it to a file share (FilePath) that must be accessible from the build agent. For more information, see [Artifacts in Azure Pipelines](#).

MaxArtifactSize - Max Artifact Size

`string`. Default value: `0`.

Maximum limit on the size of artifacts to be published in bytes. Put 0 if you don't want to set any limit.

TargetPath - File share path

`string`. Required when `ArtifactType = FilePath`.

Specifies the path to the file share where you want to copy the files. The path must be a fully qualified path or a valid path relative to the root directory of your repository. Publishing artifacts from a Linux or macOS agent to a file share is not supported.

Example: `\my\share\$(Build.DefinitionName)\$(Build.BuildNumber)`.

Parallel - Parallel copy

`boolean`. Optional. Use when `ArtifactType = FilePath`. Default value: `false`.

Specifies whether to copy files in parallel using multiple threads for greater potential throughput. If this setting is not enabled, a single thread will be used.

ParallelCount - Parallel count

`string`. Optional. Use when `ArtifactType = FilePath && Parallel = true`. Default value: `8`.

Specifies the degree of parallelism (the number of threads) used to perform the copy. The value must be at least 1 and not greater than 128. Choose a value based on CPU capabilities of the build agent.

StoreAsTar - Tar the artifact before uploading

`boolean`. Default value: `false`.

Adds all files from the publish path to a tar archive before uploading. This allows you to preserve the UNIX file permissions. Use `extractTars` option of the [DownloadBuildArtifacts](#) task to extract the downloaded items automatically. This setting is ignored on Windows agents.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

ⓘ Note

You cannot use **Bin**, **App_Data** and other folder names reserved by IIS as an Artifact name because this content is not served in response to Web requests. Please see [ASP.NET Web Project Folder Structure](#) for more details.

Examples

yml

```
steps:
- task: CopyFiles@2
  inputs:
    contents: '_buildOutput/**'
    targetFolder: $(Build.ArtifactStagingDirectory)
- task: PublishBuildArtifacts@1
```

```
inputs:  
  pathToPublish: $(Build.ArtifactStagingDirectory)  
  artifactName: MyBuildOutputs
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.91.0 or greater
Task category	Utility

See also

- [File matching patterns reference](#)
- [How do I use this task to publish artifacts](#)
- Learn how to use [verbose logs](#) for [troubleshooting](#).

PublishCodeCoverageResults@2 - Publish code coverage results v2 task

Article • 09/26/2023

Use this task to get code coverage results from a build.

Syntax

YAML

```
# Publish code coverage results v2
# Publish any of the code coverage results from a build.
- task: PublishCodeCoverageResults@2
  inputs:
    summaryFileLocation: # string. Required. Path to summary files.
    #pathToSources: # string. Path to Source files.
    #failIfCoverageEmpty: false # boolean. Fail if code coverage results are
    missing. Default: false.
```

Inputs

`summaryFileLocation` - Path to summary files

`string`. Required.

Specifies the path of the summary file containing code coverage statistics, such as line, method, and class coverage. Multiple summary files are merged into a single report. The value may contain minimatch patterns. For example:

`$(System.DefaultWorkingDirectory)/MyApp/**/site/cobertura/coverage.xml`. [More information on minimatch patterns ↗](#).

`pathToSources` - Path to Source files

`string`.

Specifying a path to source files is required when coverage XML reports don't contain an absolute path to source files. For example, JaCoCo reports don't use absolute paths, so when publishing JaCoCo coverage for Java apps, the pattern is similar to

`$(System.DefaultWorkingDirectory)/MyApp/src/main/java/`. This input should point to an absolute path to source files on the host. For example,

`$(System.DefaultWorkingDirectory)/MyApp/`.

This input can be used if tests are run in a Docker container.

`failIfCoverageEmpty` - Fail if code coverage results are missing

`boolean`. Default value: `false`.

Fails the task if code coverage did not produce any results to publish.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task in a build pipeline to publish code coverage results produced when running tests to Azure Pipelines or TFS and after generating the coverage xml files in order to obtain code coverage tab and coverage reporting details in the pipeline. The task supports code coverage generated xml formats. This task generates a cJSON file which contains the code coverage details. It will also produce a code coverage HTML report under the build artifacts.

This task is only supported in build pipelines, not release pipelines.

Tasks such as [Visual Studio Test](#), [.NET Core](#), [Ant](#), [Maven](#), [Gulp](#), and [Grunt](#) also provide the option to publish code coverage data to the pipeline. If you are using these tasks, you do not need a separate Publish Code Coverage Results task in the pipeline.

Prerequisite- To use the Publish Code Coverage Results v2 task in the pipeline, please use the [dotnet 7.0.x](#) task as a pre-requisite in the pipeline. Use the dotnet core task before the Publish Code Coverage v2 task.

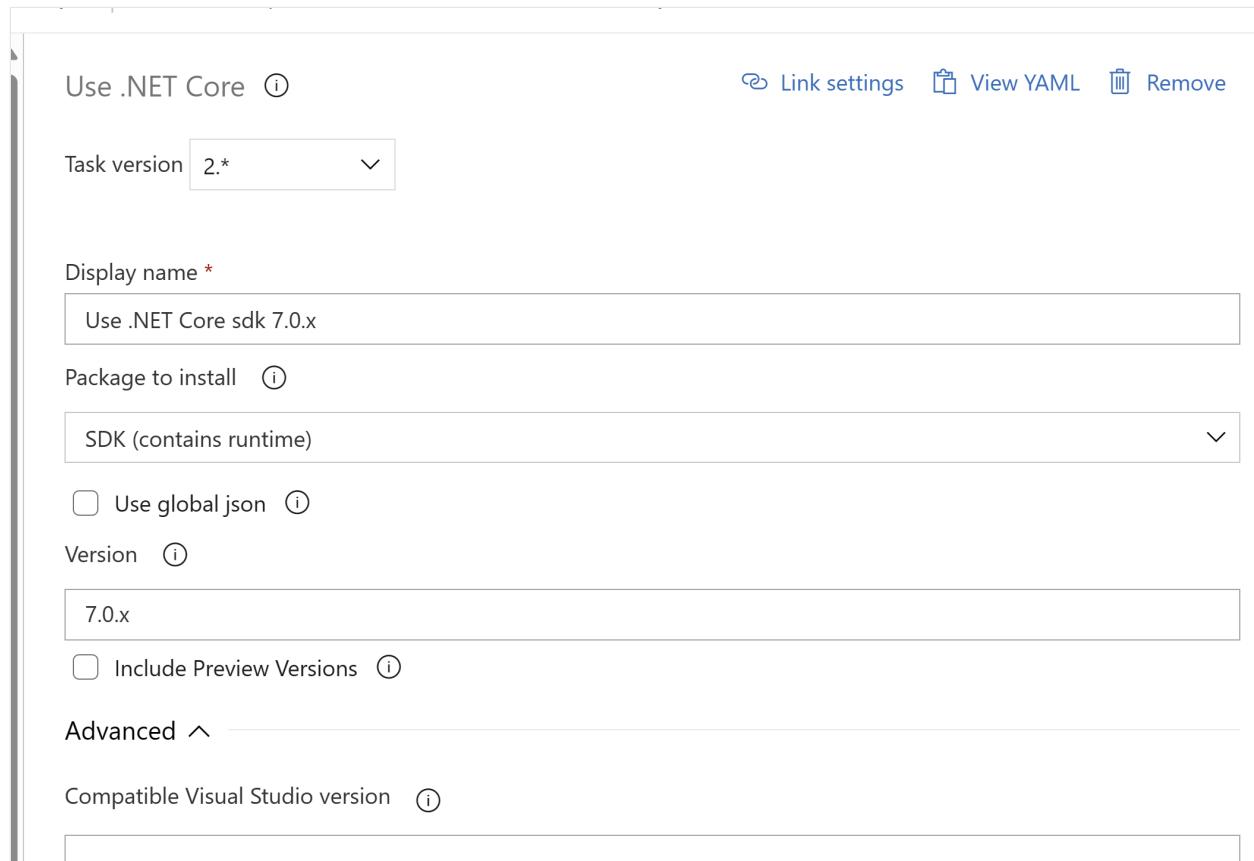
Prerequisites

To configure the prerequisites using a YAML pipeline:

`YAML`

```
# Dotnet core sdk task 7.0.x
- task: UseDotNet@2
  displayName: 'Use .NET Core sdk 7.0.x'
  inputs:
    version: 7.0.x
```

To configure the prerequisites using the designer:



1. Configure the Publish Code Coverage Results version 2 task using the following settings.

Publish code coverage results ⓘ

Task version 2.*

Display name *

Path to summary files *

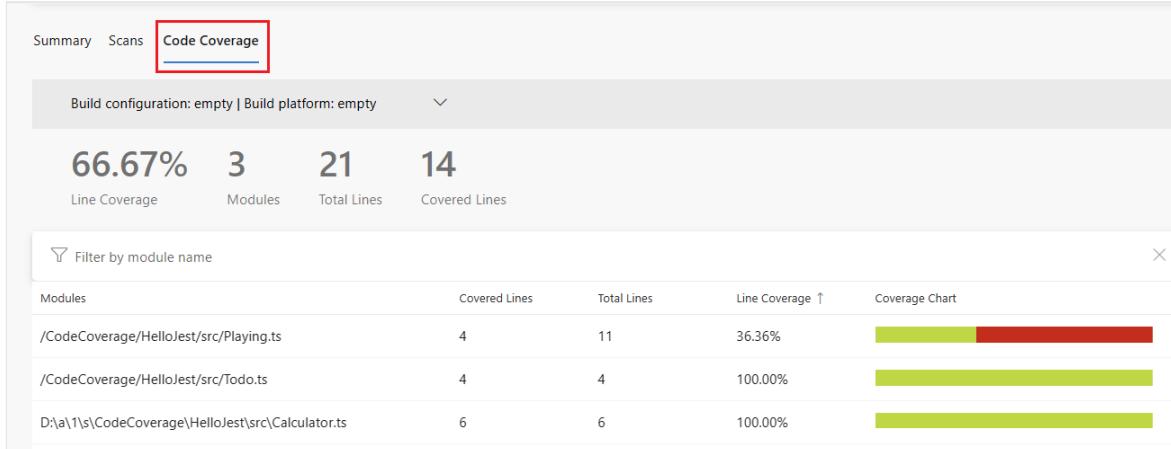
Path to Source files ⓘ

Fail if code coverage results are missing ⓘ

Control Options

Output Variables

- After the build completes and the Publish Code Coverage Results v2 task succeeds, select the **Code Coverage** tab in the pipeline run summary to view the code coverage results.



Code coverage results for JavaScript with Istanbul using YAML

To publish code coverage results for JavaScript with Istanbul using YAML, see [JavaScript](#) in the Ecosystems section of these topics, which also includes examples for other languages.

See an [example of publishing code coverage using Cobertura](#).

Docker

For apps using Docker, build and tests may run inside the container and generate code coverage results within the container. In order to publish the results to the pipeline, the resulting artifacts should be made available to the **Publish Code Coverage Results** task. For reference, you can see a similar example for publishing test results under the [Build, test, and publish results with a Docker file](#) section for **Docker**.

View results

In order to view the code coverage results in the pipeline, see [Review code coverage results](#).

Known issues

The publish code coverage results v2 task generates a cJSON file and publishes the code coverage report under the code coverage tab. It also produces a build artifacts which is a set of HTML files that are linked from the main *index.html* file. If the code coverage tab fails to show the code coverage report, check whether the input code coverage xml file is in the correct format and has the valid details.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Test

See also

- Publish Test Results

PublishCodeCoverageResults@1 - Publish code coverage results v1 task

Article • 09/26/2023

Use this task to publish Cobertura or JaCoCo code coverage results from a build.

Syntax

YAML

```
# Publish code coverage results v1
# Publish Cobertura or JaCoCo code coverage results from a build.
- task: PublishCodeCoverageResults@1
  inputs:
    codeCoverageTool: 'JaCoCo' # 'Cobertura' | 'JaCoCo'. Required. Code
    coverage tool. Default: JaCoCo.
    summaryFileLocation: # string. Required. Summary file.
    #pathToSources: # string. Path to Source files.
    #reportDirectory: # string. Report directory.
    #additionalCodeCoverageFiles: # string. Additional files.
    #failIfCoverageEmpty: false # boolean. Fail when code coverage results
    are missing. Default: false.
```

Inputs

`codeCoverageTool` - Code coverage tool

`string`. Required. Allowed values: `Cobertura`, `JaCoCo`. Default value: `JaCoCo`.

Specifies the tool that generates code coverage results.

`summaryFileLocation` - Summary file

`string`. Required.

Specifies the path of the summary file containing code coverage statistics, such as line, method, and class coverage. Multiple summary files are merged into a single report. The value may contain minimatch patterns. For example:

`$(System.DefaultWorkingDirectory)/MyApp/**/site/cobertura/coverage.xml`.

`pathToSources` - Path to Source files

`string`.

Specifying a path to source files is required when coverage XML reports don't contain an absolute path to source files. For example, JaCoCo reports don't use absolute paths, so when publishing JaCoCo coverage for Java apps, the pattern is similar to `$(System.DefaultWorkingDirectory)/MyApp/src/main/java/`. This input should point to an absolute path to source files on the host. For example,

```
$(System.DefaultWorkingDirectory)/MyApp/.
```

This input can be used if tests are run in a Docker container.

Multiple sources can be added by delimiting each list item with the `;` character, for example `pathToSources:`

```
$(System.DefaultWorkingDirectory)/path/to/first/source;$(System.DefaultWorkingDirectory)/path/to/second/source.
```

`reportDirectory` - Report directory

`string`.

Specifies the path of the code coverage HTML report directory. The report directory is published for later viewing as an artifact of the build. The value may contain minimatch patterns. For example: `$(System.DefaultWorkingDirectory)/MyApp/**/site/cobertura`.

`additionalCodeCoverageFiles` - Additional files

`string`.

Specifies the file path pattern and notes any additional code coverage files to be published as artifacts of the build. The value may contain minimatch patterns. For example: `$(System.DefaultWorkingDirectory)/**/*.*exec`.

`failIfCoverageEmpty` - Fail when code coverage results are missing

`boolean`. Default value: `false`.

Fails the task if code coverage did not produce any results to publish.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task in a build pipeline to publish code coverage results produced when running tests to Azure Pipelines or TFS in order to obtain coverage reporting. The task supports popular coverage result formats such as [Cobertura](#) and [JaCoCo](#).

This task is only supported in build pipelines, not release pipelines.

Tasks such as [Visual Studio Test](#), [.NET Core](#), [Ant](#), [Maven](#), [Gulp](#), and [Grunt](#) also provide the option to publish code coverage data to the pipeline. If you are using these tasks, you do not need a separate Publish Code Coverage Results task in the pipeline.

To generate the HTML code coverage report you need dotnet framework 2.0.0 or later on the agent. The dotnet folder must be in the environment path. If there are multiple folders containing dotnet, the one with version 2.0.0 must be before any others in the path list.

Code coverage results for JavaScript with Istanbul using YAML

To publish code coverage results for JavaScript with Istanbul using YAML, see [JavaScript](#) in the Ecosystems section of these topics, which also includes examples for other languages.

See an [example of publishing code coverage using Cobertura](#).

Docker

For apps using Docker, build and tests may run inside the container and generate code coverage results within the container. In order to publish the results to the pipeline, the resulting artifacts should be made available to the [Publish Code Coverage Results](#) task. For reference, you can see a similar example for publishing test results under the [Build, test, and publish results with a Docker file](#) section for [Docker](#).

View results

In order to view the code coverage results in the pipeline, see [Review code coverage results](#).

Is code coverage data merged when multiple files are provided as input to the task or multiple tasks are used in the pipeline?

At present, the code coverage reporting functionality provided by this task is limited, and it does not merge coverage data. If you provide multiple files as input to the task, only the first match is considered. If you use multiple publish code coverage tasks in the pipeline, the summary and report is shown for the last task. Any previously uploaded data is ignored.

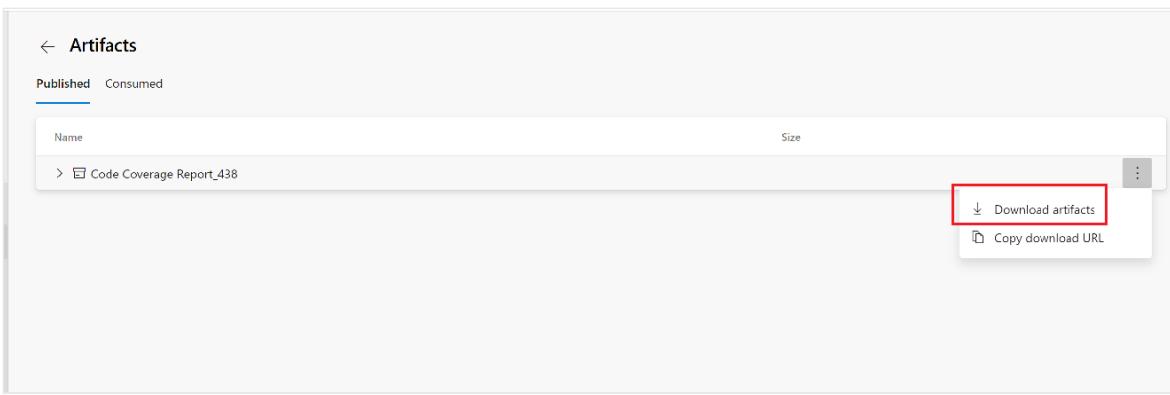
Known issues

The publish code coverage results task generates and publishes the HTML report, which is a set of HTML files that are linked from the main *index.html* file. If the code coverage tab fails to show the code coverage report, check whether the size of the *index.html* file is close to or larger than 7 MB. Complete the following steps to check the size of the file. Then, if the file size is close to or larger than 7 MB, you can use the following workaround to view the coverage report.

1. Select the build **Summary** tab, and then select the **published** link:

The screenshot shows the Azure DevOps build summary page for a build named "#438 Merged PR 28: Log responses". The "Summary" tab is highlighted with a red box. Below the tabs, there's a section titled "Manually run by" with details about the repository and version. To the right, there are sections for "Time started and elapsed" (May 6 at 1:53 PM, 3m 59s), "Related" (0 work items), and "Tests and coverage" (1 published; 1 consumed). A red box highlights the "1 published; 1 consumed" link. At the bottom, there's a table titled "Jobs" showing one job named "Agent job 1" with status "Success" and duration "3m 41s".

2. Next to the *Code Coverage Report_** artifact, select **Download artifacts**:



3. When the code coverage report is downloaded, extract the .zip file.
4. In the code coverage report, check the size of *index.html* to help determine whether the file size is causing the issue described here.
5. Open *index.html* in a browser to view the code coverage report.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	This task has permission to set the following variables : Setting variables is disabled
Agent version	2.182.1 or greater
Task category	Test

See also

- [Publish Test Results](#)

PublishPipelineArtifact@1 - Publish Pipeline Artifacts v1 task

Article • 09/26/2023

Use this task to publish (upload) a file or directory as a named artifact for the current run.

Syntax

YAML

```
# Publish Pipeline Artifacts v1
# Publish (upload) a file or directory as a named artifact for the current
run.
- task: PublishPipelineArtifact@1
  inputs:
    targetPath: '$(Pipeline.Workspace)' # string. Alias: path. Required.
    File or directory path. Default: $(Pipeline.Workspace).
    #artifact: # string. Alias: artifactName. Artifact name.
    publishLocation: 'pipeline' # 'pipeline' | 'filepath'. Alias:
    artifactType. Required. Artifact publish location. Default: pipeline.
    #fileSharePath: # string. Required when artifactType = filepath. File
    share path.
    #parallel: false # boolean. Optional. Use when artifactType = filepath.
    Parallel copy. Default: false.
    #parallelCount: '8' # string. Optional. Use when artifactType = filepath
    && parallel = true. Parallel count. Default: 8.
    #properties: # string. Custom properties.
```

Inputs

`targetPath` - File or directory path

Input alias: `path`. `string`. Required. Default value: `$(Pipeline.Workspace)`.

Specifies the path of the file or directory to publish. Can be absolute or relative to the default working directory. Can include [variables](#), but wildcards are not supported. See [Artifacts in Azure Pipelines](#) for more information.

`artifact` - Artifact name

Input alias: `artifactName`. `string`.

Specifies the name of the artifact to publish. It can be any name you choose, for example `drop`. If not set, the default is a unique ID scoped to the job.

 **Important**

Artifact name cannot contain \, /, ", :, <, >, |, *, or ?.

publishLocation - Artifact publish location

Input alias: `artifactType`. `string`. Required. Allowed values: `pipeline` (Azure Pipelines), `filepath` (A file share). Default value: `pipeline`.

Specifies whether to store the artifact in Azure Pipelines or to copy it to a file share that must be accessible from the pipeline agent.

fileSharePath - File share path

`string`. Required when `artifactType = filepath`.

Specifies the file share where the artifact files are copied. This can include variables, for example `\my\share\$(Build.DefinitionName)\$(Build.BuildNumber)`. Publishing artifacts from a Linux or macOS agent to a file share is not supported, for example `\server\folderName`.

parallel - Parallel copy

`boolean`. Optional. Use when `artifactType = filepath`. Default value: `false`.

Specifies whether to copy files in parallel using multiple threads for greater potential throughput. If this setting is not enabled, one thread will be used.

parallelCount - Parallel count

`string`. Optional. Use when `artifactType = filepath && parallel = true`. Default value: `8`.

Specifies the degree of parallelism, or the number of threads used, to perform the copy. The value must be between 1 and 128.

properties - Custom properties

`string`.

Specifies the custom properties to associate with the artifact. Use a valid JSON string with the prefix `user-` on all keys.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Publishing is not supported in classic release pipelines.

ⓘ Note

Publish Pipeline Artifacts is not supported in on-premises. Please use [Publish Build Artifacts](#) if you're using Azure DevOps Server or TFS 2018.

The `publish` and `download` keywords are shortcuts for the `PublishPipelineArtifact@1` and `DownloadPipelineArtifact@2` tasks. See [steps.publish](#) and [steps.download](#) for more details.

💡 Tip

You can use the `.artifactignore` file to control which files will be published.

I'm having issues with publishing my artifacts. How can I view the detailed logs?

To enable detailed logs for your pipeline:

1. Edit your pipeline and select **Variables**
2. Add a new variable with the name `System.Debug` and value `true`
3. **Save**

Which variables are available to me?

A: `$(Build.SourcesDirectory)` and `$(Agent.BuildDirectory)` are just few of the variables you can use in your pipeline. Variables are available as [expressions](#) or scripts.

See [Define variables, predefined variables](#), and [Classic release and artifacts variables](#) to learn about the different types of variables.

The task allows me to publish artifacts in deployment job in yaml pipeline, but I am not able to use it in downstream pipeline?

A: Deployment jobs do not have the context of source branches and are hence not appropriate for publishing artifacts. They have been primarily designed to consume artifacts. A workaround would be to isolate that logic into a separate job (with dependencies on your deployment jobs).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.199 or greater
Task category	Utility

See also

- Looking to get started with build artifacts? See [Artifacts in Azure Pipelines](#).

PublishPipelineArtifact@0 - Publish Pipeline Artifacts v0 task

Article • 09/26/2023

Use this task to publish a local directory or file as a named artifact for the current pipeline.

This task is deprecated; use [PublishPipelineArtifact@1](#).

Syntax

YAML

```
# Publish Pipeline Artifacts v0
# Publish a local directory or file as a named artifact for the current
pipeline.
- task: PublishPipelineArtifact@0
  inputs:
    artifactName: 'drop' # string. Required. The name of this artifact.
  Default: drop.
    targetPath: # string. Required. Path to publish.
    #properties: # string. Custom properties.
```

Inputs

artifactName - The name of this artifact

`string`. Required. Default value: `drop`.

Specifies the name of the artifact.

targetPath - Path to publish

`string`. Required.

Specifies the folder or file path to publish. This can be a fully qualified path or a path relative to the root of the repository. Wildcards are not supported. [Variables ↗](#) are supported.

properties - Custom properties

`string`.

Specifies the custom properties to associate with the artifact. Use a valid JSON string with the prefix `user-` on all keys.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.199 or greater
Task category	Utility

PublishPipelineMetadata@0 - Publish Pipeline Metadata v0 task

Article • 09/26/2023

Use this task to publish Pipeline Metadata to the Evidence store.

Syntax

YAML

```
# Publish Pipeline Metadata v0
# Publish Pipeline Metadata to Evidence store.
- task: PublishPipelineMetadata@0
  inputs: # none
```

Inputs

None.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.

Requirement	Description
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Utility

SonarQubePublish@5 - Publish Quality Gate Result v5 task

Article • 09/26/2023

Use this task to publish SonarQube's Quality Gate result on the Azure DevOps build result. Use this after the analysis.

Syntax

YAML

```
# Publish Quality Gate Result v5
# Publish SonarQube's Quality Gate result on the Azure DevOps build result,
# to be used after the actual analysis.
- task: SonarQubePublish@5
  inputs:
    pollingTimeoutSec: '300' # string. Required. Timeout (s). Default: 300.
```

Inputs

`pollingTimeoutSec` - Timeout (s)
string. Required. Default value: 300.

This task polls SonarQube until the analysis is completed or until the timeout is reached. It also adds a build property with the Quality Gate status of the current build(s) analyses.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Build

See also

- [SonarQube Azure DevOps Integration ↗](#)

SonarQubePublish@4 - Publish Quality Gate Result v4 task

Article • 09/26/2023

Use this task to publish SonarQube's Quality Gate result on the Azure DevOps build result. Use this after the analysis.

Syntax

YAML

```
# Publish Quality Gate Result v4
# Publish SonarQube's Quality Gate result on the Azure DevOps build result,
# to be used after the actual analysis.
- task: SonarQubePublish@4
  inputs:
    pollingTimeoutSec: '300' # string. Required. Timeout (s). Default: 300.
```

Inputs

`pollingTimeoutSec` - Timeout (s)
`string`. Required. Default value: `300`.

This task polls SonarQube until the analysis is completed or until the timeout is reached. This task also adds a build property with the Quality Gate status of the current build(s) analyses.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.95.1 or greater
Task category	Build

See also

- [SonarQube Azure DevOps Integration ↗](#)

PublishTestResults@2 - Publish Test Results v2 task

Article • 09/25/2023

Publish test results to Azure Pipelines.

Syntax

YAML

```
# Publish Test Results v2
# Publish test results to Azure Pipelines.
- task: PublishTestResults@2
  inputs:
    testResultsFormat: 'JUnit' # 'JUnit' | 'NUnit' | 'VSTest' | 'XUnit' | 'CTest'. Alias: testRunner. Required. Test result format. Default: JUnit.
    testResultsFiles: '**/TEST-*.xml' # string. Required. Test results files. Default: **/TEST-*.xml.
    #searchFolder: '$(System.DefaultWorkingDirectory)' # string. Search folder. Default: $(System.DefaultWorkingDirectory).
    #mergeTestResults: false # boolean. Merge test results. Default: false.
    #failTaskOnFailedTests: false # boolean. Fail if there are test failures. Default: false.
    #failTaskOnMissingResultsFile: false # boolean. Fail if no result files are found. Default: false.
    #testRunTitle: # string. Test run title.
  # Advanced
  #buildPlatform: # string. Alias: platform. Build Platform.
  #buildConfiguration: # string. Alias: configuration. Build Configuration.
  #publishRunAttachments: true # boolean. Upload test results files. Default: true.
```

Inputs

`testResultsFormat` - Test result format

Input alias: `testRunner`. `string`. Required. Allowed values: `JUnit`, `NUnit`, `VSTest`, `XUnit`, `CTest`. Default value: `JUnit`.

Specifies the format of the results files you want to publish. The following formats are supported: [CTest](#), [JUnit](#), [NUnit 2](#), [NUnit 3](#), Visual Studio Test (TRX) and [xUnit 2](#).

`testResultsFiles` - Test results files

`string`. Required. Default value: `**/TEST-*.xml`.

Specifies one or more test results files.

- You can use a single-folder wildcard (`*`) and recursive wildcards (`**`). For example, `**/TEST-*.xml` searches for all the XML files whose names start with `TEST-` in all subdirectories. If using VSTest as the test result format, the file type should be changed to `.trx` e.g. `**/TEST-*.trx`
- Multiple paths can be specified, separated by a new line.
- Additionally accepts [minimatch patterns](#).

For example, `!TEST[1-3].xml` excludes files named `TEST1.xml`, `TEST2.xml`, or `TEST3.xml`.

`searchFolder` - Search folder

`string`. Default value: `$(System.DefaultWorkingDirectory)`.

Optional. Specifies the folder to search for the test result files.

`mergeTestResults` - Merge test results

`boolean`. Default value: `false`.

When this boolean's value is `true`, the task reports test results from all the files against a single `test run`. If the value is `false`, the task creates a separate test run for each test result file.

⚠ Note

Use the merge test results setting to combine files from the same test framework to ensure results mapping and duration are calculated correctly.

`failTaskOnFailedTests` - Fail if there are test failures`boolean`. Default value: `false`.

Optional. When this boolean's value is `true`, the task will fail if any of the tests in the results file are marked as failed. The default is `false`, which will simply publish the results from the results file.

`failTaskOnMissingResultsFile` - Fail if no result files are found`boolean`. Default value: `false`.

Fail the task if no result files are found.

`testRunTitle` - Test run title`string`.

Optional. Specifies a name for the test run against which the results will be reported. Variable names declared in the build or release pipeline can be used.

`buildPlatform` - Build PlatformInput alias: `platform`. `string`.

Optional. Specifies the build platform against which the test run should be reported. For example: `x64` or `x86`. If you defined a variable for the platform in your build task, use it here.

`buildConfiguration` - Build ConfigurationInput alias: `configuration`. `string`.

Optional. Specifies the build configuration against which the test run should be reported. For example: `Debug` or `Release`. If you defined a variable for the configuration in your build task, use it here.

`publishRunAttachments` - Upload test results files`boolean`. Default value: `true`.

Optional. When this boolean's value is `true`, the task uploads all the test result files as attachments to the test run.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

- [Prerequisites](#)
- [Task defaults](#)
- [Result formats mapping](#)
- [Attachments support](#)

This task publishes test results to Azure Pipelines or TFS when tests are executed to provide a comprehensive test reporting and analytics experience. You can use the test runner of your choice that supports the results format you require. Supported results formats include [CTest](#), [JUnit](#) (including [PHPUnit](#), [NUnit 2](#), [NUnit 3](#)), Visual Studio Test (TRX), and [xUnit 2](#).

Other built-in tasks, such as [Visual Studio Test task](#) and [Dot NetCore CLI task](#) automatically publish test results to the pipeline. Tasks such as [Ant](#), [Maven](#), [Gulp](#), [Grunt](#), and [Xcode](#) provide publishing results as an option within the task, or build libraries such as [Cobertura](#) and [JaCoCo](#). If you are using any of these tasks, you do not need a separate [Publish Test Results](#) task in the pipeline.

The published test results are displayed in the [Tests tab](#) in the pipeline summary. The results help you to measure pipeline quality, review traceability, troubleshoot failures, and drive failure ownership.

The following example shows the task is configured to publish test results.

The screenshot shows the Azure Pipelines interface. On the left, there's a sidebar with 'Pipeline' and 'Build pipeline' options. Below that is a 'Get sources' step for '1ES Demo' branch. Under 'Agent job 1', there's a 'Publish Test Results' task. The task configuration window is open on the right, showing the following settings:

- Task version:** 2*
- Display name:** Publish Test Results **/TEST-*.xml
- Test result format:** JUnit
- Test results files:** **/TEST-*.xml
- Search folder:** \$(System.DefaultWorkingDirectory)
- Merge test results:**
- Fail if there are test failures:**
- Test run title:** (empty)

You can also use this task in a build pipeline to [publish code coverage results](#) produced when running tests to Azure Pipelines or TFS in order to obtain coverage reporting.

Prerequisites

If you're using a Windows self-hosted agent, your machine must have this prerequisite installed:

- [.NET Framework](#) 4.6.2 or a later version

Task defaults

The default option uses JUnit format to publish test results. When using VSTest as the **testRunner**, the **testResultsFiles** option should be changed to `**/TEST-*.trx`.

testResultsFormat is an alias for the **testRunner** input name. The results files can be produced by multiple runners, not just a specific runner. For example, the jUnit results format is supported by many runners and not just jUnit.

To publish test results for Python using YAML, see [Python](#) in the **Ecosystems** section of these topics, which also includes examples for other languages.

Result formats mapping

This table lists the fields reported in the [Tests tab](#) in a build or release summary, and the corresponding mapping with the attributes in the supported test result formats.

Scope	Field	Visual Studio Test (TRX)
Test run	Title	Test run title specified in the task
	Date started	/TestRun/Times.Attributes["start"].Value

Scope	Field	Visual Studio Test (TRX)
	Date completed	/TestRun/Times.Attributes["finish"].Value
	Duration	Date completed - Date started
	Attachments	Refer to Attachments support section below
Test result	Title	/TestRun/Results/UnitTestResult.Attributes["testName"].Value Or /TestRun/Results/WebTestResult.Attributes["testName"].Value Or /TestRun/Results/TestResultAggregation.Attributes["testName"].Value
	Date started	/TestRun/Results/UnitTestResult.Attributes["startTime"].Value Or /TestRun/Results/WebTestResult.Attributes["startTime"].Value Or /TestRun/Results/TestResultAggregation.Attributes["startTime"].Value
	Date completed	/TestRun/Results/UnitTestResult.Attributes["startTime"].Value + /TestRun/Results/UnitTestResult.Attributes["duration"].Value Or /TestRun/Results/WebTestResult.Attributes["startTime"].Value + /TestRun/Results/WebTestResult.Attributes["duration"].Value Or /TestRun/Results/TestResultAggregation.Attributes["startTime"].Value + /TestRun/Results/TestResultAggregation.Attributes["duration"].Value
	Duration	/TestRun/Results/UnitTestResult.Attributes["duration"].Value Or /TestRun/Results/WebTestResult.Attributes["duration"].Value Or /TestRun/Results/TestResultAggregation.Attributes["duration"].Value
	Owner	/TestRun/TestDefinitions/UnitTest/Owners/Owner.Attributes["name"].Value
	Outcome	/TestRun/Results/UnitTestResult.Attributes["outcome"].Value Or /TestRun/Results/WebTestResult.Attributes["outcome"].Value Or /TestRun/Results/TestResultAggregation.Attributes["outcome"].Value
	Error message	/TestRun/Results/UnitTestResult/Output/ErrorInfo/Message.InnerText Or /TestRun/Results/WebTestResultOutput/ErrorInfo/Message.InnerText Or /TestRun/Results/TestResultAggregation/Output/ErrorInfo/Message.InnerText
	Stack trace	/TestRun/Results/UnitTestResult/Output/ErrorInfo/StackTrace.InnerText Or /TestRun/Results/WebTestResultOutput/ErrorInfo/StackTrace.InnerText Or /TestRun/Results/TestResultAggregation/Output/ErrorInfo/StackTrace.InnerText
	Attachments	Refer to Attachments support section below
	Console log	/TestRun/Results/UnitTestResult/Output/StdOut.InnerText Or /TestRun/Results/WebTestResultOutput/Output/StdOut.InnerText Or /TestRun/Results/TestResultAggregation/Output/StdOut.InnerText
	Console error log	/TestRun/Results/UnitTestResult/Output/StdErr.InnerText Or /TestRun/Results/WebTestResultOutput/Output/StdErr.InnerText Or /TestRun/Results/TestResultAggregation/Output/StdErr.InnerText
	Agent name	/TestRun/Results/UnitTestResult.Attributes["computerName"].Value Or /TestRun/Results/WebTestResult.Attributes["computerName"].Value Or /TestRun/Results/TestResultAggregation.Attributes["computerName"].Value
	Test file	/TestRun/TestDefinitions/UnitTest.Attributes["storage"].Value
	Priority	/TestRun/TestDefinitions/UnitTest.Attributes["priority"].Value

Note

Duration is used only when **Date started** and **Date completed** are not available.

The fully qualified name format for **testName** is **Namespace.Testclass.Methodname** with a character limit of 512. If the test is data driven and has parameters, the character limit will include the parameters.

Attachments support

The Publish Test Results task provides support for attachments for both test run and test results for the following formats. For public projects, we support 2GB of total attachments.

Visual Studio Test (TRX)

Visual Studio Test (TRX)

Scope	Type	Path
Test run	Data Collector	/TestRun/ResultSummary/CollectorDataEntries/Collector/UriAttachments/UriAttachment/A.Attributes["href"].Value
	Test Result	/TestRun/ResultSummary/ResultFiles/ResultFile.Attributes["path"].Value
	Code Coverage	/TestRun/TestSettings/Execution/AgentRule/DataCollectors/DataCollector/Configuration/CodeCoverage/Regular/CodeCoverageItem.Attribute And /TestRun/TestSettings/Execution/AgentRule/DataCollectors/DataCollector/Configuration/CodeCoverage/Regular/CodeCoverageItem.Attribute
Test result	Data Collectors	/TestRun/Results/UnitTestResult/CollectorDataEntries/Collector/UriAttachments/UriAttachment/A.Attributes["href"].Value Or /TestRun/Results/WebTestResult/CollectorDataEntries/Collector/UriAttachments/UriAttachment/A.Attributes["href"].Value Or /TestRun/Results/TestResultAggregation/CollectorDataEntries/Collector/UriAttachments/UriAttachment/A.Attributes["href"].Value
	Test Result	/TestRun/Results/UnitTestResult/ResultFiles/ResultFile.Attributes["path"].Value Or /TestRun/Results/WebTestResult/ResultFiles/ResultFile.Attributes["path"].Value Or /TestRun/Results/TestResultAggregation/ResultFiles/ResultFile.Attributes["path"].Value

ⓘ Note

The option to upload the test results file as an attachment is a default option in the task, applicable to all formats.

Examples

Docker

For Docker based apps, there are many ways to build your application and run tests:

- **Build and test in a build pipeline:** builds and tests execute in the pipeline and test results are published using the **Publish Test Results** task.
- **Build and test with a multi-stage Dockerfile:** builds and tests execute inside the container using a multi-stage Docker file, as such test results are not published back to the pipeline.
- **Build, test, and publish results with a Dockerfile:** builds and tests execute inside the container, and results are published back to the pipeline. See the example below.

Build, test, and publish results with a Docker file

In this approach, you build your code and run tests inside the container using a Docker file. The test results are then copied to the host to be published to the pipeline. To publish the test results to Azure Pipelines, you can use the **Publish Test Results** task. The final image will be published to Docker or Azure Container Registry.

Get the code

1. Create a `Dockerfile.build` file at the root of your project directory with the following:

```
Dockerfile

# Build and run tests inside the docker container
FROM mcr.microsoft.com/dotnet/sdk:2.1
WORKDIR /app
# copy the contents of agent working directory on host to workdir in container
COPY . .
# dotnet commands to build, test, and publish
RUN dotnet restore
RUN dotnet build -c Release
RUN dotnet test dotnetcore-tests/dotnetcore-tests.csproj -c Release --logger "trx;LogFile=filename=testresults.trx"
RUN dotnet publish -c Release -o out
ENTRYPOINT dotnet dotnetcore-sample/out/dotnetcore-sample.dll
```

This file contains the instructions to build code and run tests. The tests are then copied to a file `testresults.trx` inside the container.

2. To make the final image as small as possible, containing only the runtime and deployment artifacts, replace the contents of the existing `Dockerfile` with the following:

```
Dockerfile
```

```

# This Dockerfile creates the final image to be published to Docker or
# Azure Container Registry
# Create a container with the compiled asp.net core app
FROM mcr.microsoft.com/dotnet/aspnet:2.1
# Create app directory
WORKDIR /app
# Copy only the deployment artifacts
COPY /out .
ENTRYPOINT ["dotnet", "dotnetcore-sample.dll"]

```

Define the build pipeline

YAML

1. If you have a Docker Hub account, and want to push the image to your Docker registry, replace the contents of the `.vsts-ci.docker.yml` file with the following:

YAML

```

# Build Docker image for this app, to be published to Docker Registry
pool:
  vmImage: 'ubuntu-latest'
variables:
  buildConfiguration: 'Release'
steps:
- script: |
    docker build -f Dockerfile.build -t $(dockerId)/dotnetcore-build:$BUILD_BUILDID .
    docker run --name dotnetcoreapp --rm -d $(dockerId)/dotnetcore-build:$BUILD_BUILDID
    docker cp dotnetcoreapp:/app/dotnetcore-tests/TestResults $(System.DefaultWorkingDirectory)
    docker cp dotnetcoreapp:/app/dotnetcore-sample/out $(System.DefaultWorkingDirectory)
    docker stop dotnetcoreapp

- task: PublishTestResults@2
  inputs:
    testRunner: VSTest
    testResultsFiles: '**/*.trx'
    failTaskOnFailedTests: true

- script: |
    docker build -f Dockerfile -t $(dockerId)/dotnetcore-sample:$BUILD_BUILDID .
    docker login -u $(dockerId) -p $pswd
    docker push $(dockerId)/dotnetcore-sample:$BUILD_BUILDID
  env:
    pswd: $(dockerPassword)

```

Alternatively, if you configure an Azure Container Registry and want to push the image to that registry, replace the contents of the `.vsts-ci.yml` file with the following:

YAML

```

# Build Docker image for this app to be published to Azure Container Registry
pool:
  vmImage: 'ubuntu-latest'
variables:
  buildConfiguration: 'Release'

steps:
- script: |
    docker build -f Dockerfile.build -t $(dockerId)/dotnetcore-build:$BUILD_BUILDID .
    docker run --name dotnetcoreapp --rm -d $(dockerId)/dotnetcore-build:$BUILD_BUILDID
    docker cp dotnetcoreapp:/app/dotnetcore-tests/TestResults $(System.DefaultWorkingDirectory)
    docker cp dotnetcoreapp:/app/dotnetcore-sample/out $(System.DefaultWorkingDirectory)
    docker stop dotnetcoreapp

- task: PublishTestResults@2
  inputs:
    testRunner: VSTest
    testResultsFiles: '**/*.trx'
    failTaskOnFailedTests: true

- script: |
    docker build -f Dockerfile -t $(dockerId).azurecr.io/dotnetcore-sample:$BUILD_BUILDID .
    docker login -u $(dockerId) -p $pswd $(dockerId).azurecr.io
    docker push $(dockerId).azurecr.io/dotnetcore-sample:$BUILD_BUILDID

```

```
env:  
  pswd: $(dockerPassword)
```

2. Push the change to the main branch in your repository.
3. If you use Azure Container Registry, ensure you have [pre-created the registry](#) in the Azure portal. Copy the admin user name and password shown in the **Access keys** section of the registry settings in Azure portal.
4. Update your build pipeline with the following
 - **Agent pool:** `Hosted Ubuntu 1604`
 - **dockerId:** Set the value to your Docker ID for DockerHub or the admin user name for Azure Container Registry.
 - **dockerPassword:** Set the value to your password for DockerHub or the admin password Azure Container Registry.
 - **YAML file path:** `./vsts-ci.docker.yml`
5. Queue a new build and watch it create and push a Docker image to your registry and the test results to Azure DevOps.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.0.0 or greater
Task category	Test

PublishTestResults@1 - Publish test results v1 task

Article • 09/26/2023

Publish test results to Azure Pipelines.

Syntax

YAML

```
# Publish test results v1
# Publish test results to Azure Pipelines.
- task: PublishTestResults@1
  inputs:
    testRunner: 'JUnit' # 'JUnit' | 'NUnit' | 'VSTest' | 'XUnit'. Required.
    Test Result Format. Default: JUnit.
    testResultsFiles: '**/TEST-*.xml' # string. Required. Test Results
    Files. Default: **/TEST-*.xml.
    #mergeTestResults: false # boolean. Merge Test Results. Default: false.
    #testRunTitle: # string. Test Run Title.
    # Advanced
    #platform: # string. Platform.
    #configuration: # string. Configuration.
    #publishRunAttachments: true # boolean. Upload Test Attachments.
    Default: true.
```

Inputs

`testRunner` - Test Result Format

`string`. Required. Allowed values: `JUnit`, `NUnit`, `VSTest`, `XUnit`. Default value: `JUnit`.

Specifies the format of the results files you want to publish. The following formats are supported: [CTest](#), [JUnit](#), [NUnit 2](#), [NUnit 3](#), Visual Studio Test (TRX) and [xUnit 2](#).

`testResultsFiles` - Test Results Files

`string`. Required. Default value: `**/TEST-*.xml`.

Specifies one or more test results files.

- You can use a single-folder wildcard (*) and recursive wildcards (**). For example, `**/TEST-*.xml` searches for all the XML files whose names start with `TEST-` in all subdirectories. If using VSTest as the test result format, the file type should be changed to `.trx` e.g. `**/TEST-*.trx`
- Multiple paths can be specified, separated by a new line.
- Additionally accepts [minimatch patterns](#).

For example, `!TEST[1-3].xml` excludes files named `TEST1.xml`, `TEST2.xml`, or `TEST3.xml`.

`mergeTestResults` - Merge Test Results

`boolean`. Default value: `false`.

When this boolean's value is `true`, the task reports test results from all the files against a single [test run](#). If the value is `false`, the task creates a separate test run for each test result file.

ⓘ Note

Use the merge test results setting to combine files from the same test framework to ensure results mapping and duration are calculated correctly.

`testRunTitle` - Test Run Title

`string`.

Optional. Specifies a name for the test run against which the results will be reported. Variable names declared in the build or release pipeline can be used.

`platform` - Platform

`string`.

Optional. Specifies the build platform against which the test run should be reported. For example: `x64` or `x86`. If you defined a variable for the platform in your build task, use it here.

`configuration` - Configuration

`string`.

Optional. Specifies the build configuration against which the test run should be reported. For example: `Debug` or `Release`. If you defined a variable for the configuration in your build task, use it here.

`publishRunAttachments` - Upload Test Attachments

`boolean`. Default value: `true`.

Optional. When this boolean's value is `true`, the task uploads all the test result files as attachments to the test run.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

[PublishTestResults@2](#) is a newer version of this task that provides NUnit3 support and support for Minimatch files patterns.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater

Requirement	Description
Task category	Test

See also

- [PublishTestResults@2](#)

PublishToAzureServiceBus@1 - Publish To Azure Service Bus v1 task

Article • 09/26/2023

Use this task to send a message to Azure Service Bus using a service connection (no agent is required).

Syntax

YAML

```
# Publish To Azure Service Bus v1
# Sends a message to Azure Service Bus using a service connection (no agent
is required).
- task: PublishToAzureServiceBus@1
  inputs:
    azureSubscription: # string. Alias: connectedServiceName. Required.
    Azure Service Bus service connection.
    #messageBody: # string. Message body.
    #waitForCompletion: false # boolean. Wait for task completion. Default:
    false.
    #useDataContractSerializer: true # boolean. Use .NET data contract
    serializer. Default: true.
    # Advanced
    #sessionId: # string. Session Id.
    #signPayload: false # boolean. Sign the Message. Default: false.
    #certificateString: # string. Required when signPayload = true.
    Certificate Variable.
    #signatureKey: 'signature' # string. Optional. Use when signPayload =
    true. Signature Property Key. Default: signature.
```

Inputs

`azureSubscription` - Azure Service Bus service connection

Input alias: `connectedServiceName`. `string`. Required.

Specifies an Azure Service Bus service connection.

`messageBody` - Message body

`string`.

Specifies the JSON `messageBody`.

sessionId - Session Id`string.`

Specifies the session ID with which the message is published. For session-based queues, the publishing fails if a value is not specified. For non session-based queues, a value does not need to be specified.

signPayload - Sign the Message`boolean.` Default value: `false`.

If set to `true`, a private certificate will be added to the message.

certificateString - Certificate Variable`string.` Required when `signPayload = true`.

Specifies the secret variable that contains the certificate content. This can also be a certificate stored in an Azure key vault that is [linked](#) to a variable group used by the release pipeline.

signatureKey - Signature Property Key`string.` Optional. Use when `signPayload = true`. Default value: `signature`.

In Message Properties, specifies the key where the signature is. If left empty, the default value is `signature`.

waitForCompletion - Wait for task completion`boolean.` Default value: `false`.

If set to `true`, this task will wait for the TaskCompleted event for the specified task timeout.

useDataContractSerializer - Use .NET data contract serializer.`boolean.` Default value: `true`.

Set `useDataContractSerializer` to `false` if you want to pass your message as a stream instead of an object.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task in an [agentless job](#) of a release pipeline to send a message to an Azure Service Bus using a service connection (without using an agent).

 **Note**

Can only be used in an [agentless job](#) of a release pipeline.

Where should a task signal completion?

To signal completion, the external service should POST completion data to the following pipelines REST endpoint.

```
{planUri}/{projectId}/_apis/distributedtask/hubs/{hubName}/plans/{planId}/events?api-version=2.0-preview.1

**Request Body**
{
  "name": "TaskCompleted",
  "taskId": "taskInstanceId",
  "jobId": "jobId",
  "result": "succeeded"
}
```

See [this simple cmdline application](#) for specifics.

In addition, a C# helper library is available to enable live logging and managing the task status for agentless tasks. Learn more about [Async HTTP agentless tasks](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Server
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Utility

PublishToAzureServiceBus@0 - Publish To Azure Service Bus v0 task

Article • 09/26/2023

Use this task to send a message to Azure Service Bus using a service connection (no agent required).

Syntax

YAML

```
# Publish To Azure Service Bus v0
# Sends a message to azure service bus using a service connection (no agent
required).
- task: PublishToAzureServiceBus@0
  inputs:
    azureSubscription: # string. Alias: connectedServiceName. Required.
    Azure service bus connection.
    messageBody: '{"JobId": "$(system.jobId)", "PlanId": "$(system.planId)", "TimelineId": "$(system.timelineId)", "ProjectId": "$(system.teamProjectId)", "VstsUrl": "$(system.CollectionUri)", "AuthToken": "$(system.AccessToken)" }' # string. Required. Message body. Default:
    {"JobId": "$(system.jobId)", "PlanId": "$(system.planId)", "TimelineId": "$(system.timelineId)", "ProjectId": "$(system.teamProjectId)", "VstsUrl": "$(system.CollectionUri)", "AuthToken": "$(system.AccessToken)" }.
    #waitForCompletion: false # boolean. Wait for task completion. Default:
    false.
```

Inputs

`azureSubscription` - Azure service bus connection

Input alias: `connectedServiceName`. `string`. Required.

Specifies an Azure Service Bus connection.

`messageBody` - Message body

`string`. Required. Default value: `{"JobId": "$(system.jobId)", "PlanId": "$(system.planId)", "TimelineId": "$(system.timelineId)", "ProjectId": "$(system.teamProjectId)", "VstsUrl": "$(system.CollectionUri)", "AuthToken": "$(system.AccessToken)" }`.

Specifies the JSON `messageBody`.

`waitForCompletion` - Wait for task completion

`boolean`. Default value: `false`.

If set to `true`, this task will wait for the `TaskCompleted` event for the specified task timeout.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task in an [agentless job](#) of a release pipeline to send a message to Azure Service Bus using a service connection (without using an agent).

Note

Can only be used in an [agentless job](#) of a release pipeline.

Where should a task signal completion?

To signal completion, the external service should POST completion data to the following pipelines REST endpoint.

```
{planUri}/{projectId}/_apis/distributedtask/hubs/{hubName}/plans/{planId}/events?api-version=2.0-preview.1
```

Request Body

```
{ "name": "TaskCompleted", "taskId": "taskInstanceId", "jobId": "jobId", "result": "succeeded" }
```

See this simple cmdline application [↗](#) for specifics.

In addition, a C# helper library is available to enable live logging and managing the task status for agentless tasks. Learn more about [Async HTTP agentless tasks](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Server
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Utility

PyPIPublisher@0 - PyPI publisher v0 task

Article • 09/26/2023

Use this task to create and upload an sdist or wheel to a PyPI-compatible index using Twine.

Syntax

YAML

```
# PyPI publisher v0
# Create and upload an sdist or wheel to a PyPI-compatible index using
Twine.
- task: PyPIPublisher@0
  inputs:
    pypiConnection: # string. Alias: serviceEndpoint. Required. PyPI service
connection.
    packageDirectory: # string. Alias: wd. Required. Python package
directory.
    #alsoPublishWheel: false # boolean. Alias: wheel. Also publish a wheel.
Default: false.
```

Inputs

pypiConnection - PyPI service connection

Input alias: `serviceEndpoint`. `string`. Required.

Specifies a generic service connection for connecting to the package index.

packageDirectory - Python package directory

Input alias: `wd`. `string`. Required.

Specifies the directory of the Python package that is created and published where `setup.py` is present.

alsoPublishWheel - Also publish a wheel

Input alias: `wheel`. `boolean`. Default value: `false`.

Specifies whether to create and publish a universal wheel package (platform independent) in addition to an sdist package. More information about [packaging Python projects](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

ⓘ Important

The PyPI Publisher task has been deprecated. You can now [publish PyPI packages using Twine authentication and custom scripts](#).

Use this task to create and upload an sdist or wheel to a PyPI-compatible index using Twine.

This task builds an sdist package by running `python setup.py sdist` with the Python instance in `PATH`. In addition to the sdist, it can optionally build a universal wheel. It will upload the package to a PyPI index using `twine`. The task will install the `wheel` and `twine` packages with `python -m pip install --user`.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.

Requirement	Description
Command restrictions	Any
Settable variables	Any
Agent version	2.0.0 or greater
Task category	Package

PipAuthenticate@1 - Python pip authenticate v1 task

Article • 09/26/2023

Use this task to provide authentication for the `pip` client that installs Python distributions.

Syntax

YAML

```
# Python pip authenticate v1
# Authentication task for the pip client used for installing Python
distributions.
- task: PipAuthenticate@1
  inputs:
    # Feeds and Authentication
    #artifactFeeds: # string. My feeds (select below).
    #pythonDownloadServiceConnections: # string. Feeds from external
organizations.
    #onlyAddExtraIndex: false # boolean. Don't set primary index URL.
    Default: false.
```

Inputs

`artifactFeeds` - My feeds (select below)

`string`.

Specifies a comma-separated list of Azure Artifacts feeds to authenticate with pip.

`pythonDownloadServiceConnections` - Feeds from external organizations

`string`.

Specifies a comma-separated list of [pip service connection](#) names from external organizations to authenticate with pip.

`onlyAddExtraIndex` - Don't set primary index URL

`boolean`. Default value: `false`.

If this task is set to `true`, no feed will be set as the primary index URL. All of the configured feeds/endpoints will be set as extra index URLs.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Provides authentication for the `pip` client that is used to install Python distributions.

- [When in my pipeline should I run this task?](#)
- [My agent is behind a web proxy. Will PipAuthenticate set up pip to use my proxy?](#)
- [My Pipeline needs to access a feed in a different project](#)

When in my pipeline should I run this task?

This task must run before you use pip to download Python distributions to an authenticated package source such as Azure Artifacts. There are no other ordering requirements. Multiple invocations of this task will not stack credentials. Every run of the task will erase any previously stored credentials.

My agent is behind a web proxy. Will PipAuthenticate set up pip to use my proxy?

No. While this task itself will work behind a [web proxy](#) your agent has been configured to [use](#), it does not configure pip to use the proxy.

To do so, you can:

- Set the environment variables `http_proxy`, `https_proxy` and optionally `no_proxy` to your proxy settings. See [Pip official guidelines](#) for details. These are commonly used variables, which other non-Python tools (e.g. curl) may also use.

✖ Caution

The `http_proxy` and `no_proxy` variables are case-sensitive on Linux and Mac operating systems and must be lowercase. Attempting to use an Azure Pipelines variable to set the environment variable will not work, as it will be converted to uppercase. Instead, set the environment variables on the self-hosted agent's machine and restart the agent.

- Add the proxy settings to the `pip config file` file using `proxy` key.
- Use the `--proxy` command-line option to specify proxy in the form
`[user:passwd@]proxy.server:port`.

My Pipeline needs to access a feed in a different project

If the pipeline is running in a different project than the project hosting the feed, you must set up the other project to grant read/write access to the build service. See [Package permissions in Azure Pipelines](#) for more details.

Examples

Download Python distributions from Azure Artifacts feeds without consulting official Python registry

In this example, we are setting authentication for downloading from private Azure Artifacts feeds. The authenticate task creates environment variables `PIP_INDEX_URL` and `PIP_EXTRA_INDEX_URL` that are required to download the distributions. The task sets the variables with authentication credentials the task generates for the provided Artifacts feeds. `HelloTestPackage` must be present in either `myTestFeed1` or `myTestFeed2`; otherwise, the install will fail.

For project-scoped feeds that are in a different project than where the pipeline is running, you must manually give the project and the feed access to the pipeline's project's build service.

YAML

```
- task: PipAuthenticate@1
  displayName: 'Pip Authenticate'
  inputs:
    # Provide list of feed names which you want to authenticate.
    # Project scoped feeds must include the project name in addition to the
```

```
feed name.  
    artifactFeeds: 'project1/myTestFeed1, myTestFeed2'  
  
    # Use command line tool to 'pip install'.  
    - script: |  
        pip install HelloTestPackage
```

Consult official Python registry and then download Python distributions from Azure Artifacts feeds

In this example, we are setting authentication for downloading from a private Azure Artifacts feed, but [pypi](#) is consulted first. The authenticate task creates an environment variable `PIP_EXTRA_INDEX_URL`, which contains auth credentials required to download the distributions. `HelloTestPackage` will be downloaded from the authenticated feeds only if it's not present in [pypi](#).

For project-scoped feeds that are in a different project than where the pipeline is running, you must manually give the project and the feed access to the pipeline's project's build service.

YAML

```
- task: PipAuthenticate@1  
  displayName: 'Pip Authenticate'  
  inputs:  
    # Provide list of feed names which you want to authenticate.  
    # Project scoped feeds must include the project name in addition to the  
    # feed name.  
    artifactFeeds: 'project1/myTestFeed1, myTestFeed2'  
    # Setting this variable to "true" will force pip to get distributions  
    # from official python registry first and fallback to feeds mentioned above if  
    # distributions are not found there.  
    onlyAddExtraIndex: true  
  
    # Use command line tool to 'pip install'.  
    - script: |  
        pip install HelloTestPackage
```

Download Python distributions from other private Python servers

In this example, we are setting authentication for downloading from an external Python distribution server. Create a [pip service connection](#) entry for the external service. The authenticate task uses the service connection to create an environment variable `PIP_INDEX_URL`, which contains auth credentials required to download the distributions.

`HelloTestPackage` has to be present in the `pypitest` service connection; otherwise, install will fail. If you want [pypi](#) to be consulted first, set `onlyAddExtraIndex` to `true`.

YAML

```
- task: PipAuthenticate@1
  displayName: 'Pip Authenticate'
  inputs:
    # In this case, name of the service connection is "pypitest".
    pythonDownloadServiceConnections: pypitest

  # Use command line tool to 'pip install'.
- script: |
  pip install HelloTestPackage
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Package

PipAuthenticate@0 - Python pip authenticate v0 task

Article • 09/26/2023

Use this task to provide authentication for the `pip` client that installs Python distributions.

Syntax

YAML

```
# Python pip authenticate v0
# Authentication task for the pip client used for installing Python
distributions.
- task: PipAuthenticate@0
  inputs:
    # Feeds and Authentication
    #artifactFeeds: # string. Alias: feedList. My feeds (select below).
    #externalFeeds: # string. Alias: externalSources. Feeds from external
organizations.
```

Inputs

`artifactFeeds` - My feeds (select below)

Input alias: `feedList`. `string`.

Specifies the feeds to authenticate as present in the organization.

`externalFeeds` - Feeds from external organizations

Input alias: `externalSources`. `string`.

Specifies the endpoints to authenticate outside the organization.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Package

PythonScript@0 - Python script v0 task

Article • 09/26/2023

Use this task to run a Python file or inline script.

Syntax

```
# Python script v0
# Run a Python file or inline script.
- task: PythonScript@0
  inputs:
    scriptSource: 'filePath' # 'filePath' | 'inline'. Required. Script
    source. Default: filePath.
    scriptPath: # string. Required when scriptSource = filePath. Script
    path.
    #script: # string. Required when scriptSource = inline. Script.
    #arguments: # string. Arguments.
    # Advanced
    #pythonInterpreter: # string. Python interpreter.
    #workingDirectory: # string. Working directory.
    #failOnStderr: false # boolean. Fail on standard error. Default: false.
```

Inputs

`scriptSource` - Script source

`string`. Required. Allowed values: `filePath` (File path), `inline`. Default value: `filePath`.

Specifies whether the script is a file in the source tree or is written inline in this task.

`scriptPath` - Script path

`string`. Required when `scriptSource = filePath`.

Specifies the path of the script to execute. Must be a fully qualified path or relative to `$(System.DefaultWorkingDirectory)`.

`script` - Script

`string`. Required when `scriptSource = inline`.

Specifies the Python script to run.

`arguments` - Arguments

`string`.

Specifies the arguments passed to the script execution available through `sys.argv`, as if you passed them on the command line.

`pythonInterpreter` - Python interpreter

`string`.

Specifies the absolute path to the Python interpreter to use. If not specified, the task will use the interpreter in PATH.

Run the [Use Python Version](#) task to add a version of Python to PATH.

`workingDirectory` - Working directory

`string`.

Specifies the working directory where the script will run. If not specified, the value of `System.DefaultWorkingDirectory` will be used. For builds, this variable defaults to the root of the repository. For releases, it defaults to the root of the artifacts directory.

`failOnStderr` - Fail on standard error

`boolean`. Default value: `false`.

If set to `true`, this task will fail if any text is written to the `stderr` stream.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

By default, this task will invoke `python` from the system path. Run [Use Python Version](#) to put the version you want in the system path.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Utility

TwineAuthenticate@1 - Python twine upload authenticate v1 task

Article • 09/26/2023

Use this task to authenticate uploads of Python distributions using twine. Add `-r FeedName/EndpointName --config-file $(PYPIRC_PATH)` to your twine upload command. For feeds present in this organization, use the feed name as the repository (`-r`). Otherwise, use the endpoint name defined in the service connection.

Syntax

YAML

```
# Python twine upload authenticate v1
# Authenticate for uploading Python distributions using twine. Add '-r
# FeedName/EndpointName --config-file $(PYPIRC_PATH)' to your twine upload
# command. For feeds present in this organization, use the feed name as the
# repository (-r). Otherwise, use the endpoint name defined in the service
# connection.
- task: TwineAuthenticate@1
  inputs:
    # Feeds and Authentication
    #artifactFeed: # string. My feed name (select below).
    #pythonUploadServiceConnection: # string. Feed from external
    organizations.
```

Inputs

`artifactFeed` - My feed name (select below)

`string`.

Specifies the Azure artifact's feed name to authenticate with twine. The authenticating feed must be present within the organization. For project-scoped feeds, use the syntax `projectName/feedNameSelect`.

`pythonUploadServiceConnection` - Feed from external organizations

`string`.

A [twine service connection](#) name from an external organization to authenticate with twine. The credentials stored in the endpoint must have package upload permissions.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Provides `twine` credentials to a `PYPIRC_PATH` environment variable for the scope of the build. This enables you to publish Python packages to feeds with `twine` from your build.

- [When in my pipeline should I run this task?](#)
- [My agent is behind a web proxy. Will TwineAuthenticate set up twine to use my proxy?](#)
- [My Pipeline needs to access a feed in a different project](#)

When in my pipeline should I run this task?

This task must run before you use `twine` to upload Python distributions to an authenticated package source, such as Azure Artifacts. There are no other ordering requirements. Multiple invocations of this task will not stack credentials. Every task run will erase any previously stored credentials.

My agent is behind a web proxy. Will TwineAuthenticate set up twine to use my proxy?

No. While this task itself will work behind a [web proxy](#) your agent has been configured to use, it does not configure `twine` to use the proxy.

My Pipeline needs to access a feed in a different project

If the pipeline is running in a different project than the project hosting the feed, you must set up the other project to grant read/write access to the build service. See [Package permissions in Azure Pipelines](#) for more details.

Examples

The following examples demonstrate how to publish python distribution to Azure Artifacts feed and the official python registry.

- [Publish Python distribution to Azure Artifacts feed](#)
- [Publish Python distribution to the official Python registry](#)

Publish Python distribution to Azure Artifacts feed

In this example, we are setting authentication for publishing to a private Azure Artifacts Feed. The authenticate task creates a `.pypirc` file that contains the auth credentials required to publish a distribution to the feed.

```
YAML

# Install python distributions like wheel, twine etc
- script: |
    pip install wheel
    pip install twine

# Build the python distribution from source
- script: |
    python setup.py bdist_wheel

- task: TwineAuthenticate@1
  displayName: Twine Authenticate
  inputs:
    # In this case, name of the feed is 'myTestFeed' in the project
    'myTestProject'. Project is needed because the feed is project scoped.
    artifactFeed: myTestProject/myTestFeed

# Use command line script to 'twine upload', use -r to pass the repository
name and --config-file to pass the environment variable set by the
authenticate task.
- script: |
    python -m twine upload -r myTestFeed --config-file $(PYPIRC_PATH)
    dist/*.whl
```

The `artifactFeed` input will contain the project and the feed name if the feed is project scoped. If the feed is organization scoped, only the feed name must be provided. [Learn more](#).

Publish Python distribution to the official Python registry

In this example, we are setting up authentication for publishing to the official Python registry. Create a [twine service connection](#) entry for [pypi](#). The authenticate task uses that service connection to create a `.pypirc` file that contains the auth credentials required to publish the distribution.

```
YAML

# Install python distributions like wheel, twine etc
- script: |
    pip install wheel
    pip install twine

# Build the python distribution from source
- script: |
    python setup.py bdist_wheel

- task: TwineAuthenticate@1
  displayName: Twine Authenticate
  inputs:
    # In this case, name of the service connection is "pypitest".
    pythonUploadServiceConnection: pypitest

# Use command line script to 'twine upload', use -r to pass the repository
# name and --config-file to pass the environment variable set by the
# authenticate task.
- script: |
    python -m twine upload -r "pypitest" --config-file $(PYPIRC_PATH)
    dist/*.whl
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Package

TwineAuthenticate@0 - Python twine upload authenticate v0 task

Article • 09/26/2023

Provides `twine` credentials to a `PYPIRC_PATH` environment variable for the scope of the build. This enables you to publish Python packages to feeds with `twine` from your build.

Syntax

YAML

```
# Python twine upload authenticate v0
# Authenticate for uploading Python distributions using twine. Add '-r
FeedName/EndpointName --config-file $(PYPIRC_PATH)' to your twine upload
command. For feeds present in this organization, use the feed name as the
repository (-r). Otherwise, use the endpoint name defined in the service
connection.
- task: TwineAuthenticate@0
  inputs:
    # Feeds and Authentication
    #artifactFeeds: # string. Alias: feedList. My feeds (select below).
    #externalFeeds: # string. Alias: externalSources. Feeds from external
organizations.
    # Advanced
    #publishPackageMetadata: true # boolean. Publish pipeline metadata.
Default: true.
```

Inputs

`artifactFeeds` - My feeds (select below)

Input alias: `feedList`. `string`.

Specifies the Azure artifact's feed name to authenticate with twine. The authenticating feed must be present within the organization. For project-scoped feeds, use the syntax `projectName/feedNameSelect`.

`externalFeeds` - Feeds from external organizations

Input alias: `externalSources`. `string`.

A [twine service connection](#) name from an external organization to authenticate with twine. The credentials stored in the endpoint must have package upload permissions.

`publishPackageMetadata` - Publish pipeline metadata

`boolean`. Default value: `true`.

Associates this build/release pipeline's metadata (such as run # and source code information) with the package when uploading to feeds.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Provides `twine` credentials to a `PYPIRC_PATH` environment variable for the scope of the build. This enables you to publish Python packages to feeds with `twine` from your build.

When in my pipeline should I run this task?

This task must run before you use `twine` to upload Python distributions to an authenticated package source, such as Azure Artifacts. There are no other ordering requirements. Multiple invocations of this task will not stack credentials. Every task run will erase any previously stored credentials.

My agent is behind a web proxy. Will TwineAuthenticate set up `twine` to use my proxy?

No. While this task itself will work behind a [web proxy your agent has been configured to use](#), it does not configure `twine` to use the proxy.

My Pipeline needs to access a feed in a different project

If the pipeline is running in a different project than the project hosting the feed, you must set up the other project to grant read/write access to the build service. See

[Package permissions in Azure Pipelines](#) for more details.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Package

AzureMonitor@1 - Query Azure Monitor alerts v1 task

Article • 09/26/2023

Observe the configured Azure Monitor rules for active alerts.

Syntax

YAML

```
# Query Azure Monitor alerts v1
# Observe the configured Azure Monitor rules for active alerts.
- task: AzureMonitor@1
  inputs:
    connectedServiceNameARM: # string. Required. Azure subscription.
    ResourceGroupName: # string. Required. Resource group.
    # Advanced
    filterType: 'none' # 'resource' | 'alertrule' | 'none'. Required. Filter type. Default: none.
    #resource: # string. Required when filterType = resource. Resource.
    #alertRule: # string. Required when filterType = alertrule. Alert rule.
    #severity: 'Sev0,Sev1,Sev2,Sev3,Sev4' # 'Sev0' | 'Sev1' | 'Sev2' |
    'Sev3' | 'Sev4'. Severity. Default: Sev0,Sev1,Sev2,Sev3,Sev4.
    #timeRange: '1h' # '1h' | '1d' | '7d' | '30d'. Time range. Default: 1h.
    #alertState: 'Acknowledged,New' # 'New' | 'Acknowledged' | 'Closed'.
    Alert state. Default: Acknowledged,New.
    #monitorCondition: 'Fired' # 'Fired' | 'Resolved'. Monitor condition.
    Default: Fired.
```

Inputs

`connectedServiceNameARM` - Azure subscription

`string`. Required.

Selects an Azure Resource Manager subscription to monitor.

`ResourceGroupName` - Resource group

`string`. Required.

Provides the name of a resource group to monitor in the subscription.

filterType - Filter type

`string`. Required. Allowed values: `resource` (By resource), `alertrule` (By alert rule), `none`. Default value: `none`.

Filters the type by a specific resource or alert rule.

resource - Resource

`string`. Required when `filterType = resource`.

Selects the Azure resource to monitor.

alertrule - Alert rule

`string`. Required when `filterType = alertrule`.

Selects from the currently configured alert rules to query for the status.

The default value is to select all.

severity - Severity

`string`. Allowed values: `Sev0`, `Sev1`, `Sev2`, `Sev3`, `Sev4`. Default value: `Sev0,Sev1,Sev2,Sev3,Sev4`.

Filters by severity.

timeRange - Time range

`string`. Allowed values: `1h` (Past hour), `1d` (Past 24 hours), `7d` (Past 7 days), `30d` (Past 30 days). Default value: `1h`.

Filters by the time range.

alertState - Alert state

`string`. Allowed values: `New`, `Acknowledged`, `Closed`. Default value: `Acknowledged,New`.

Filters by the state of the alert instance.

monitorCondition - Monitor condition

`string`. Allowed values: `Fired` (Fired), `Resolved`. Default value: `Fired`.

Represents whether the underlying conditions have crossed the defined alert rule thresholds.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

To observe the configured Azure monitor rules for active alerts, use this task in an [agentless job](#) of a release pipeline.

 **Note**

This task can only be used in an [agentless job](#) of a release pipeline.

The task succeeds if none of the alert rules are activated at the time of sampling.

For more information about using this task, see [Approvals and gates overview](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Server, ServerGate
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.

Requirement	Description
Task category	Utility

AzureMonitor@0 - Query Classic Azure Monitor alerts v0 task

Article • 09/26/2023

Observe the configured classic Azure Monitor rules for active alerts.

This task is deprecated; use [AzureMonitor@1](#).

Syntax

YAML

```
# Query Classic Azure Monitor alerts v0
# Observe the configured classic Azure Monitor rules for active alerts.
- task: AzureMonitor@0
  inputs:
    connectedServiceNameARM: # string. Required. Azure subscription.
    ResourceGroupName: # string. Required. Resource group.
    ResourceType: 'Microsoft.Insights/components' #
      'Microsoft.Insights/components' | 'Microsoft.Web/sites' |
      'Microsoft.Storage/storageAccounts' | 'Microsoft.Compute/virtualMachines'.
    Required. Resource type. Default: Microsoft.Insights/components.
    resourceName: # string. Required. Resource name.
    alertRules: # string. Required. Alert rules.
```

Inputs

`connectedServiceNameARM` - Azure subscription

`string`. Required.

Selects an Azure Resource Manager subscription to monitor.

`ResourceGroupName` - Resource group

`string`. Required.

Provides the name of a resource group to monitor.

`ResourceType` - Resource type

`string`. Required. Allowed values: `Microsoft.Insights/components` (Application Insights), `Microsoft.Web/sites` (App Services), `Microsoft.Storage/storageAccounts` (Storage)

Account), `Microsoft.Compute/virtualMachines` (Virtual Machines). Default value: `Microsoft.Insights/components`.

Selects the Azure resource type to monitor.

resourceName - Resource name

`string`. Required.

Selects the name of the Azure resource to monitor.

alertRules - Alert rules

`string`. Required.

A list of Azure alert rules to monitor.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Server, ServerGate
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.

Requirement	Description
Task category	Utility

queryWorkItems@0 - Query work items v0 task

Article • 09/26/2023

Use this task in an [agentless](#) job of a release pipeline to ensure the number of matching items returned by a work item query is within the configured thresholds.

Can only be used in an [agentless](#) job of a release pipeline.

Syntax

YAML

```
# Query work items v0
# Execute a work item query and check the number of items returned.
- task: queryWorkItems@0
  inputs:
    queryId: # string. Required. Query.
    maxThreshold: '0' # string. Required. Upper threshold. Default: 0.
    # Advanced
    minThreshold: '0' # string. Required. Lower threshold. Default: 0.
```

Inputs

queryId - Query

`string`. Required.

Specifies a saved work item query within the current project to execute. Can be a built-in or custom query.

maxThreshold - Upper threshold

`string`. Required. Default value: `0`.

Specifies the maximum number of matching work items from the query.

minThreshold - Lower threshold

`string`. Required. Default value: `0`.

Specifies the minimum number of matching work items from the query.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task in an [agentless job](#) of a release pipeline to ensure the number of matching items returned by a work item query is within the configured thresholds.

ⓘ Note

This task can only be used in an [agentless job](#) of a release pipeline.

This task succeeds if `_minimum-threshold_ <= _#-matching-workitems_ <= _maximum-threshold_`.

For more information about using this task, see the [Approvals and gates overview](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Server, ServerGate
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.

Requirement	Description
Task category	Utility

ReviewApp@0 - Review App v0 task

Article • 09/26/2023

Use this task under a deploy phase provider to create a resource dynamically.

Syntax

YAML

```
# Review App v0
# Use this task under deploy phase provider to create a resource
dynamically.
- task: ReviewApp@0
  inputs:
    resourceName: # string. Required. Resource name.
    #baseEnvironmentName: # string. Environment name.
    #reviewResourceName: # string. Review Resource Name.
```

Inputs

`resourceName` - Resource name

`string`. Required.

Specifies the name of an existing resource in the environment, which will be used for resource-type information.

`baseEnvironmentName` - Environment name

`string`.

`reviewResourceName` - Review Resource Name

`string`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task under a deploy phase provider to create a resource dynamically.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Utility

SonarQubeAnalyze@5 - Run Code Analysis v5 task

Article • 09/26/2023

Use this task to run the scanner and upload the results to the SonarQube server.

Syntax

YAML

```
# Run Code Analysis v5
# Run scanner and upload the results to the SonarQube server.
- task: SonarQubeAnalyze@5
  inputs:
    jdkversion: 'JAVA_HOME_11_X64' # 'JAVA_HOME' | 'JAVA_HOME_11_X64' |
'JAVA_HOME_17_X64'. Required. JDK version source for analysis. Default:
JAVA_HOME_11_X64.
```

Inputs

jdkversion - JDK version source for analysis

`string`. Required. Allowed values: `JAVA_HOME` (Use `JAVA_HOME`), `JAVA_HOME_11_X64` (Use built-in `JAVA_HOME_11_X64` (hosted agent)), `JAVA_HOME_17_X64` (Use built-in `JAVA_HOME_17_X64` (hosted agent)). Default value: `JAVA_HOME_11_X64`.

Select the wanted Java version for the analysis : You can choose with either Self provided `JAVA_HOME` which will pick up the value of this env variable, or you can choose the built-in `JAVA_HOME_XX_X64` value on hosted agent. Default value is `JAVA_HOME_11_X64`, however if you choose either of the proposed value and they are not available, `JAVA_HOME` value will be picked up instead.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

This task is to be used with the new version of the [Prepare Analysis Configuration](#) task.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: java
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Build

See also

- [SonarQube Azure DevOps Integration ↗](#)

SonarQubeAnalyze@4 - Run Code Analysis v4 task

Article • 09/26/2023

Use this task to run the scanner and upload the results to the SonarQube server.

Syntax

YAML

```
# Run Code Analysis v4
# Run scanner and upload the results to the SonarQube server.
- task: SonarQubeAnalyze@4
  inputs: # none
```

Inputs

None.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

This task is to be used with the new version of the [Prepare Analysis Configuration](#) task.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build

Requirement	Description
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: java
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.119.1 or greater
Task category	Build

See also

- [SonarQube Azure DevOps Integration ↗](#)

RunVisualStudioTestsusingTestAgent@1

- Run functional tests v1 task

Article • 09/26/2023

RunVisualStudioTestsusingTestAgent@1 and its companion task (Visual Studio Test Agent Deployment) are deprecated. Use the Visual Studio Test task instead. The VSTest task can run unit as well as functional tests. Run tests on one or more agents using the multi-agent job setting. Use the Visual Studio Test Platform task to run tests without needing Visual Studio on the agent. VSTest task also brings new capabilities, such as automatically rerunning failed tests.

Syntax

YAML

```
# Run functional tests v1
# Deprecated: This task and it's companion task (Visual Studio Test Agent Deployment) are deprecated. Use the 'Visual Studio Test' task instead. The VSTest task can run unit as well as functional tests. Run tests on one or more agents using the multi-agent job setting. Use the 'Visual Studio Test Platform' task to run tests without needing Visual Studio on the agent. VSTest task also brings new capabilities such as automatically rerunning failed tests.
- task: RunVisualStudioTestsusingTestAgent@1
  inputs:
    # Setup Options
    testMachineGroup: # string. Required. Machines.
    dropLocation: # string. Required. Test Drop Location.
    # Execution Options
    testSelection: 'testAssembly' # 'testAssembly' | 'testPlan'. Required. Test Selection. Default: testAssembly.
    #testPlan: # string. Required when testSelection = testPlan. Test Plan.
    #testSuite: # string. Required when testSelection = testPlan. Test Suite.
    #testConfiguration: # string. Required when testSelection = testPlan. Test Configuration.
    sourcefilters: '**\*test*.dll' # string. Required when testSelection = testAssembly. Test Assembly. Default: **\*test*.dll.
    #testFilterCriteria: # string. Optional. Use when testSelection = testAssembly. Test Filter criteria.
    #runSettingsFile: # string. Run Settings File.
    #overrideRunParams: # string. Override Test Run Parameters.
    #codeCoverageEnabled: false # boolean. Code Coverage Enabled. Default: false.
    #customSlicingEnabled: false # boolean. Distribute tests by number of machines. Default: false.
    # Reporting Options
```

```
#testRunTitle: # string. Test Run Title.  
#platform: # string. Platform.  
#configuration: # string. Configuration.  
#testConfigurations: # string. Test Configurations.  
#autMachineGroup: # string. Application Under Test Machines.
```

Inputs

`testMachineGroup` - Machines

`string`. Required.

A comma separated list of machine FQDNs or IP addresses, which may include the port number. The maximum is 32 machines or 32 agents. The list items can be:

- The name of an [Azure Resource Group](#).
- A comma-delimited list of machine names. Example:
`dbserver.fabrikam.com,dbserver_int.fabrikam.com:5986,192.168.34:5986`
- An output variable from a previous task.

`dropLocation` - Test Drop Location

`string`. Required.

Specifies the location on the test machine(s) where the test binaries have been copied by a [Windows Machine File Copy](#) or an [Azure File Copy](#) task. System stage variables from the test agent machines can be used to specify the drop location. Examples:

`c:\tests` and `%systemdrive%\Tests`.

`testSelection` - Test Selection

`string`. Required. Allowed values: `testAssembly` (Test Assembly), `testPlan` (Test Plan).

Default value: `testAssembly`.

Specifies how tests are run: using test assemblies or Test Plan.

`testPlan` - Test Plan

`string`. Required when `testSelection = testPlan`.

Specifies a test plan that is already configured for this organization.

testSuite - Test Suite

`string`. Required when `testSelection = testPlan`.

Specifies a test suite from the selected test plan.

testConfiguration - Test Configuration

`string`. Required when `testSelection = testPlan`.

Specifies a test configuration from the selected test plan.

sourcefilters - Test Assembly

`string`. Required when `testSelection = testAssembly`. Default value: `***test*.dll`.

Specifies the test binaries to run tests on. Wildcards can be used. For example,

`***test*.dll`; for all `.dll` files containing `test` in the file name.

testFilterCriteria - Test Filter criteria

`string`. Optional. Use when `testSelection = testAssembly`.

The filter that specifies the tests to execute within the test assembly files. Works the same way as the `/TestCaseFilter` option in `vstest.console.exe`. Example:

`Owner=james&Priority=1`.

runSettingsFile - Run Settings File

`string`.

Specifies the file path to the `runsettings` or `testsettings` file to use with the tests.

overrideRunParams - Override Test Run Parameters

`string`.

Specifies the override parameters that are defined in the `TestRunParameters` section of the `runsettings` file or the `Properties` section of the `testsettings` file. Example:

`AppURL=$(DeployURL);Port=8080`.

(!) Note

The properties specified in the `testsettings` file can be accessed via `TestContext` using Test Agent 2017 Update 4 or higher.

`codeCoverageEnabled` - Code Coverage Enabled

`boolean`. Default value: `false`.

Specifies if Code Coverage is enabled for the task.

`customSlicingEnabled` - Distribute tests by number of machines

`boolean`. Default value: `false`.

When the value of this boolean is set to `true`, the tests are distributed based on the number of machines provided instead of the number of test containers.

 **Note**

Tests within a `.dll` might also be distributed to multiple machines.

`testRunTitle` - Test Run Title

`string`.

Specifies a name for the test run.

`platform` - Platform

`string`.

Specifies the platform against which the tests should be reported. If you have defined a variable for `platform` in your build task, use the variable as the value.

`configuration` - Configuration

`string`.

Specifies the configuration against which the tests should be reported. If you have defined a variable for `configuration` in your build task, use the variable as the value.

`testConfigurations` - Test Configurations

`string`.

Optional. Associates a test case filter against a test configuration ID. Syntax: `<Filter1>:<Id1>;DefaultTestConfiguration:<Id3>`. Example: `FullyQualifiedNamespace~Chrome:12`.

`autMachineGroup` - Application Under Test Machines

`string`.

A comma separated list of machines, output variables, or machine group names on which server processes, such as `W3WP.exe`, are running.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.104.0 or greater
Task category	Test

ServiceFabricDeploy@1 - Service Fabric application deployment v1 task

Article • 09/26/2023

Use this task to deploy a Service Fabric application to a cluster. This task deploys an Azure Service Fabric application to a cluster according to the settings defined in the publish profile.

ⓘ Note

This task does not support **Azure Resource Manager authentication with workflow identity federation**.

Syntax

YAML

```
# Service Fabric application deployment v1
# Deploy an Azure Service Fabric application to a cluster.
- task: ServiceFabricDeploy@1
  inputs:
    applicationPackagePath: # string. Required. Application Package.
    serviceConnectionName: # string. Required. Cluster Service Connection.
    #publishProfilePath: # string. Publish Profile.
    #applicationParameterPath: # string. Application Parameters.
    #overrideApplicationParameter: false # boolean. Override Application
Parameters. Default: false.
    # Advanced Settings
    #compressPackage: false # boolean. Compress Package. Default: false.
    #copyPackageTimeoutSec: # string. CopyPackageTimeoutSec.
    #registerPackageTimeoutSec: # string. RegisterPackageTimeoutSec.
    overwriteBehavior: 'SameAppTypeAndVersion' # 'Always' | 'Never' |
'SameAppTypeAndVersion'. Required. Overwrite Behavior. Default:
SameAppTypeAndVersion.
    #skipUpgradeSameTypeAndVersion: false # boolean. Skip upgrade for same
Type and Version. Default: false.
    #skipPackageValidation: false # boolean. Skip package validation.
Default: false.
    # Upgrade Settings
    #useDiffPackage: false # boolean. Use Diff Package. Default: false.
    #overridePublishProfileSettings: false # boolean. Override All Publish
Profile Upgrade Settings. Default: false.
    #isUpgrade: true # boolean. Optional. Use when
overridePublishProfileSettings = true. Upgrade the Application. Default:
true.
```

```
#unregisterUnusedVersions: true # boolean. Unregister Unused Versions.
Default: true.

#upgradeMode: 'Monitored' # 'Monitored' | 'UnmonitoredAuto' |
'UnmonitoredManual'. Required when overridePublishProfileSettings = true &&
isUpgrade = true. Upgrade Mode. Default: Monitored.

#FailureAction: 'Rollback' # 'Rollback' | 'Manual'. Required when
overridePublishProfileSettings = true && isUpgrade = true && upgradeMode =
Monitored. FailureAction. Default: Rollback.

#UpgradeReplicaSetCheckTimeoutSec: # string. Optional. Use when
overridePublishProfileSettings = true && isUpgrade = true.
UpgradeReplicaSetCheckTimeoutSec.

#TimeoutSec: # string. Optional. Use when overridePublishProfileSettings =
true && isUpgrade = true. TimeoutSec.

#ForceRestart: false # boolean. Optional. Use when
overridePublishProfileSettings = true && isUpgrade = true. ForceRestart.
Default: false.

#HealthCheckRetryTimeoutSec: # string. Optional. Use when
overridePublishProfileSettings = true && isUpgrade = true && upgradeMode =
Monitored. HealthCheckRetryTimeoutSec.

#HealthCheckWaitDurationSec: # string. Optional. Use when
overridePublishProfileSettings = true && isUpgrade = true && upgradeMode =
Monitored. HealthCheckWaitDurationSec.

#HealthCheckStableDurationSec: # string. Optional. Use when
overridePublishProfileSettings = true && isUpgrade = true && upgradeMode =
Monitored. HealthCheckStableDurationSec.

#UpgradeDomainTimeoutSec: # string. Optional. Use when
overridePublishProfileSettings = true && isUpgrade = true && upgradeMode =
Monitored. UpgradeDomainTimeoutSec.

#ConsiderWarningAsError: false # boolean. Optional. Use when
overridePublishProfileSettings = true && isUpgrade = true && upgradeMode =
Monitored. ConsiderWarningAsError. Default: false.

#DefaultServiceTypeHealthPolicy: # string. Optional. Use when
overridePublishProfileSettings = true && isUpgrade = true && upgradeMode =
Monitored. DefaultServiceTypeHealthPolicy.

#MaxPercentUnhealthyDeployedApplications: # string. Optional. Use when
overridePublishProfileSettings = true && isUpgrade = true && upgradeMode =
Monitored. MaxPercentUnhealthyDeployedApplications.

#UpgradeTimeoutSec: # string. Optional. Use when
overridePublishProfileSettings = true && isUpgrade = true && upgradeMode =
Monitored. UpgradeTimeoutSec.

#ServiceTypeHealthPolicyMap: # string. Optional. Use when
overridePublishProfileSettings = true && isUpgrade = true && upgradeMode =
Monitored. ServiceTypeHealthPolicyMap.

# Docker Settings

#configureDockerSettings: false # boolean. Configure Docker settings.
Default: false.

#registryCredentials: 'AzureResourceManagerEndpoint' #
'AzureResourceManagerEndpoint' | 'ContainerRegistryEndpoint' |
'UsernamePassword'. Required when configureDockerSettings = true. Registry
Credentials Source. Default: AzureResourceManagerEndpoint.

#dockerRegistryConnection: # string. Alias: dockerRegistryEndpoint.
Required when configureDockerSettings = true && registryCredentials =
ContainerRegistryEndpoint. Docker Registry Service Connection.

#azureSubscription: # string. Alias: azureSubscriptionEndpoint. Required
when configureDockerSettings = true && registryCredentials =
```

```
AzureResourceManagerEndpoint. Azure subscription.  
  #registryUserName: # string. Optional. Use when configureDockerSettings  
  = true && registryCredentials = UsernamePassword. Registry User Name.  
  #registryPassword: # string. Optional. Use when configureDockerSettings  
  = true && registryCredentials = UsernamePassword. Registry Password.  
  #passwordEncrypted: true # boolean. Optional. Use when  
  configureDockerSettings = true && registryCredentials = UsernamePassword.  
  Password Encrypted. Default: true.
```

Inputs

`applicationPackagePath` - Application Package

`string`. Required.

Specifies the path to the application package that is to be deployed. [Variables](#) and wildcards can be used in the path.

`serviceConnectionName` - Cluster Service Connection

`string`. Required.

Specifies the Azure Service Fabric service connection to be used to connect to the cluster. The settings defined in this referenced service connection override those defined in the publish profile. Choose [Manage](#) to register a new service connection.

To connect to the cluster, the service fabric task uses the machine cert store to store the information about the certificate. If two releases run together on one machine using the same certificate, they will start properly. However, if one of the tasks is complete, then the certificate from the machine cert store will be cleaned up, which affects the second release.

`publishProfilePath` - Publish Profile

`string`.

Optional. Specifies the path to the publish profile file that defines the settings to use. [Variables](#) and wildcards can be used in the path. Learn more about [how to create publish profiles in Visual Studio](#).

`applicationParameterPath` - Application Parameters

`string`.

Optional. Specifies the path to the application parameters file. [Variables](#) and wildcards can be used in the path. If specified, this overrides the value in the publish profile. Learn more about [how to create an application parameters file in Visual Studio](#).

`overrideApplicationParameter` - Override Application Parameters

`boolean`. Default value: `false`.

Optional. Specifies the variables defined in the build or release pipeline are matched against the `Parameter Name` entries in the application manifest file. Learn more about [how to create an application parameters file in Visual Studio](#). Example:

```
<Parameters>
<Parameter Name="SampleApp_PartitionCount" Value="1" />
<Parameter Name="SampleApp_InstanceCount" DefaultValue="-1" />
</Parameters>
```

If your application has a parameter defined as in the above example, and you want to change the partition count to `2`, you can define a release pipeline or an environment variable `SampleApp_PartitionCount` and its value as `2`.

ⓘ Note

If the same variables are defined in the release pipeline and in the environment, then the environment variables will supersede the release pipeline variables.

`compressPackage` - Compress Package

`boolean`. Default value: `false`.

Optional. Specifies whether the application package should be compressed before copying to the image store. If enabled, this overrides the value in the publish profile. Learn more about [compressing packages](#).

`copyPackageTimeoutSec` - CopyPackageTimeoutSec

`string`.

Optional. Specifies the timeout in seconds for copying application package to the image store. If specified, this overrides the value in the publish profile.

registerPackageTimeoutSec - RegisterPackageTimeoutSec`string`.

Optional. Specifies the timeout in seconds for registering or un-registering an application package.

overwriteBehavior - Overwrite Behavior`string`. Required. Allowed values: `Always`, `Never`, `SameAppTypeAndVersion`. Default value: `SameAppTypeAndVersion`.

Overwrites behavior if an application exists in the cluster with the same name and upgrades have not been configured.

`Never` will not remove the existing application. This is the default behavior.

`Always` will remove the existing application, even if its application type and version is different from the application being created.

`SameAppTypeAndVersion` will remove the existing application only if its application type and version is the same as the application being created.

skipUpgradeSameTypeAndVersion - Skip upgrade for same Type and Version`boolean`. Default value: `false`.

Optional. Specifies whether an upgrade will be skipped if the same application type and version already exists in the cluster; otherwise, the upgrade fails during validation. If enabled, re-deployments are idempotent.

skipPackageValidation - Skip package validation`boolean`. Default value: `false`.

Optional. Specifies whether the package should be validated or not before deployment. Learn more about [package validation](#).

useDiffPackage - Use Diff Package`boolean`. Default value: `false`.

Optional. Upgrades by using a diff package that contains only the updated application files, the updated application manifest, and the service manifest files.

A diff package is created by comparing the package specified in the application package input against the package that is currently registered in the target cluster. If a service version in the cluster's current package is the same as the new package, then this service package will be removed from the new application package. Learn more about [diff packages](#).

`overridePublishProfileSettings` - Override All Publish Profile Upgrade Settings

`boolean`. Default value: `false`.

Optional. Overrides all upgrade settings with either specified values or the default value (if not specified). Learn more about [upgrade settings](#).

`isUpgrade` - Upgrade the Application

`boolean`. Optional. Use when `overridePublishProfileSettings = true`. Default value: `true`.

Overwrites the application if the value is set to `false`.

`unregisterUnusedVersions` - Unregister Unused Versions

`boolean`. Default value: `true`.

Optional. Indicates whether all unused versions of the application type will be removed after an upgrade.

`upgradeMode` - Upgrade Mode

`string`. Required when `overridePublishProfileSettings = true && isUpgrade = true`.

Allowed values: `Monitored`, `UnmonitoredAuto`, `UnmonitoredManual`. Default value: `Monitored`.

`FailureAction` - FailureAction

`string`. Required when `overridePublishProfileSettings = true && isUpgrade = true && upgradeMode = Monitored`. Allowed values: `Rollback`, `Manual`. Default value: `Rollback`.

`UpgradeReplicaSetCheckTimeoutSec` - UpgradeReplicaSetCheckTimeoutSec

`string`. Optional. Use when `overridePublishProfileSettings = true && isUpgrade = true`.

TimeoutSec - **TimeoutSec**

`string`. Optional. Use when `overridePublishProfileSettings = true && isUpgrade = true`.

ForceRestart - **ForceRestart**

`boolean`. Optional. Use when `overridePublishProfileSettings = true && isUpgrade = true`. Default value: `false`.

HealthCheckRetryTimeoutSec - **HealthCheckRetryTimeoutSec**

`string`. Optional. Use when `overridePublishProfileSettings = true && isUpgrade = true && upgradeMode = Monitored`.

HealthCheckWaitDurationSec - **HealthCheckWaitDurationSec**

`string`. Optional. Use when `overridePublishProfileSettings = true && isUpgrade = true && upgradeMode = Monitored`.

HealthCheckStableDurationSec - **HealthCheckStableDurationSec**

`string`. Optional. Use when `overridePublishProfileSettings = true && isUpgrade = true && upgradeMode = Monitored`.

UpgradeDomainTimeoutSec - **UpgradeDomainTimeoutSec**

`string`. Optional. Use when `overridePublishProfileSettings = true && isUpgrade = true && upgradeMode = Monitored`.

ConsiderWarningAsError - **ConsiderWarningAsError**

`boolean`. Optional. Use when `overridePublishProfileSettings = true && isUpgrade = true && upgradeMode = Monitored`. Default value: `false`.

DefaultServiceTypeHealthPolicy - **DefaultServiceTypeHealthPolicy**

`string`. Optional. Use when `overridePublishProfileSettings = true && isUpgrade = true && upgradeMode = Monitored`.

MaxPercentUnhealthyDeployedApplications -

MaxPercentUnhealthyDeployedApplications

`string`. Optional. Use when `overridePublishProfileSettings = true && isUpgrade = true && upgradeMode = Monitored`.

UpgradeTimeoutSec - UpgradeTimeoutSec

`string`. Optional. Use when `overridePublishProfileSettings = true && isUpgrade = true && upgradeMode = Monitored`.

ServiceTypeHealthPolicyMap - ServiceTypeHealthPolicyMap

`string`. Optional. Use when `overridePublishProfileSettings = true && isUpgrade = true && upgradeMode = Monitored`.

configureDockerSettings - Configure Docker settings

`boolean`. Default value: `false`.

Configures the application with the specified Docker settings.

registryCredentials - Registry Credentials Source

`string`. Required when `configureDockerSettings = true`. Allowed values:

`AzureResourceManagerEndpoint` (Azure Resource Manager Service Connection),
`ContainerRegistryEndpoint` (Container Registry Service Connection), `UsernamePassword` (Username and Password). Default value: `AzureResourceManagerEndpoint`.

Specifies how credentials for the Docker registry are provided.

dockerRegistryConnection - Docker Registry Service Connection

Input alias: `dockerRegistryEndpoint`. `string`. Required when `configureDockerSettings = true && registryCredentials = ContainerRegistryEndpoint`.

Specifies a Docker registry service connection. Required for commands that need to authenticate with a registry.

ⓘ Note

The task tries to encrypt the registry secret before transmitting it to the service fabric cluster. However, the task needs the cluster's server certificate to be installed

on the agent machine. If the certificate is not present, the registry secret will not be encrypted.

azureSubscription - Azure subscription

Input alias: `azureSubscriptionEndpoint`. `string`. Required when `configureDockerSettings = true && registryCredentials = AzureResourceManagerEndpoint`.

Specifies an Azure subscription.

ⓘ Note

The task will try to encrypt the registry secret before transmitting it to the service fabric cluster. However, the task needs the cluster's server certificate to be installed on the agent machine. If the certificate is not present, the registry secret will not be encrypted.

registryUserName - Registry User Name

`string`. Optional. Use when `configureDockerSettings = true && registryCredentials = UsernamePassword`.

Specifies the username for the Docker registry.

registryPassword - Registry Password

`string`. Optional. Use when `configureDockerSettings = true && registryCredentials = UsernamePassword`.

Specifies the password for the Docker registry. If the password is not encrypted, you should use a custom release pipeline secret variable to store it.

passwordEncrypted - Password Encrypted

`boolean`. Optional. Use when `configureDockerSettings = true && registryCredentials = UsernamePassword`. Default value: `true`.

You should encrypt your password using [Invoke-ServiceFabricEncryptText](#). If you do not, and a certificate matching the Server Certificate Thumbprint in the Cluster Service Connection is installed on the build agent, that certificate will be used to encrypt the password; otherwise, an error will occur.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to deploy a Service Fabric application to a cluster. This task deploys an Azure Service Fabric application to a cluster according to the settings defined in the publish profile.

Service Fabric

This task uses a Service Fabric installation to connect and deploy to a Service Fabric cluster. [Download and install Service Fabric](#) on the build agent.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: Cmd
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.95.0 or greater
Task category	Deploy

ServiceFabricComposeDeploy@0 - Service Fabric Compose deploy v0 task

Article • 09/26/2023

Use this task to deploy a Docker Compose application to a Service Fabric cluster. This task deploys an Azure Service Fabric application to a cluster according to the settings defined in the Compose file.

ⓘ Note

This task does not support **Azure Resource Manager authentication with workflow identity federation**.

Syntax

YAML

```
# Service Fabric Compose deploy v0
# Deploy a Docker Compose application to an Azure Service Fabric cluster.
- task: ServiceFabricComposeDeploy@0
  inputs:
    clusterConnection: # string. Alias: serviceConnectionName. Required.
    Cluster Service Connection.
    composeFilePath: '**/docker-compose.yml' # string. Required. Compose
    File Path. Default: **/docker-compose.yml.
    applicationName: 'fabric:/Application1' # string. Required. Application
    Name. Default: fabric:/Application1.
    # Registry Settings
    registryCredentials: 'AzureResourceManagerEndpoint' #
    'AzureResourceManagerEndpoint' | 'ContainerRegistryEndpoint' |
    'UsernamePassword' | 'None'. Required. Registry Credentials Source. Default:
    AzureResourceManagerEndpoint.
    #dockerRegistryConnection: # string. Alias: dockerRegistryEndpointName.
    Optional. Use when registryCredentials = ContainerRegistryEndpoint. Docker
    Registry Service Connection.
    azureSubscription: # string. Alias: azureSubscriptionEndpoint. Required
    when registryCredentials = AzureResourceManagerEndpoint. Azure subscription.
    #registryUserName: # string. Optional. Use when registryCredentials =
    UsernamePassword. Registry User Name.
    #registryPassword: # string. Optional. Use when registryCredentials =
    UsernamePassword. Registry Password.
    #passwordEncrypted: true # boolean. Optional. Use when
    registryCredentials = UsernamePassword. Password Encrypted. Default: true.
    # Advanced Settings
    #upgrade: false # boolean. Upgrade. Default: false.
```

```
#deployTimeoutSec: # string. Deploy Timeout (s).  
#removeTimeoutSec: # string. Remove Timeout (s).  
#getStatusTimeoutSec: # string. Get Status Timeout (s).
```

Inputs

`clusterConnection` - Cluster Service Connection

Input alias: `serviceConnectionName`. `string`. Required.

Specifies an Azure Service Fabric service connection to be used to connect to the cluster. Choose `Manage` to register a new service connection.

`composeFilePath` - Compose File Path

`string`. Required. Default value: `**/docker-compose.yml`.

Specifies the path to the compose file that is to be deployed. [Variables](#) and wildcards can be used in the path. Example:

```
$(System.DefaultWorkingDirectory)/**/drop/projectartifacts/**/docker-compose.yml.
```

ⓘ Note

Combining compose files is not supported as part of this task.

`applicationName` - Application Name

`string`. Required. Default value: `fabric:/Application1`.

Specifies the Service Fabric application name of the deployed application. Use `fabric:/` as a prefix. Application names within a Service Fabric cluster must be unique.

`registryCredentials` - Registry Credentials Source

`string`. Required. Allowed values: `AzureResourceManagerEndpoint` (Azure Resource Manager service connection), `ContainerRegistryEndpoint` (Container Registry service connection), `UsernamePassword` (Username and Password), `None`. Default value: `AzureResourceManagerEndpoint`.

Specifies how credentials for the Docker container registry will be provided to the deployment task. The allowed values are:

- `AzureResourceManagerEndpoint` (Azure Resource Manager service connection): uses `azureSubscription` to obtain a service principal ID and key for an Azure Container Registry.
- `ContainerRegistryEndpoint` (Container Registry service connection): uses `dockerRegistryConnection` to select a Docker registry service connection. If a certificate matching the Server Certificate Thumbprint in the Cluster Service Connection is installed on the build agent, it will be used to encrypt the password; otherwise, the password will not be encrypted.
- `UsernamePassword` (Username and Password): uses `registryUsername` and `registryPassword` to store the username and password for the Docker registry. Passwords should be encrypted using [Invoke-ServiceFabricEncryptText](#) with the `Password Encrypted` option. If passwords are not encrypted with [Invoke-ServiceFabricEncryptText](#), and a certificate matching the Server Certificate Thumbprint in the Cluster Connection is installed on the build agent, the certificate will be used to encrypt the password. Otherwise, the password will not be encrypted and will be sent in clear text.
- `None`: No registry credentials are provided. This is used for accessing public container registries.

`dockerRegistryConnection` - Docker Registry Service Connection

Input alias: `dockerRegistryEndpointName`. `string`. Optional. Use when `registryCredentials = ContainerRegistryEndpoint`.

Specifies a Docker registry service connection. If a certificate matching the Server Certificate Thumbprint in the Cluster Service Connection is installed on the build agent, it will be used to encrypt the password; otherwise, the password will not be encrypted.

`azureSubscription` - Azure subscription

Input alias: `azureSubscriptionEndpoint`. `string`. Required when `registryCredentials = AzureResourceManagerEndpoint`.

Specifies an Azure subscription.

`registryUserName` - Registry User Name

`string`. Optional. Use when `registryCredentials = UsernamePassword`.

Specifies the username for the Docker registry.

registryPassword - Registry Password

`string`. Optional. Use when `registryCredentials = UsernamePassword`.

Specifies the password for the Docker registry. If the password is not encrypted, it is recommended that you use a custom release pipeline secret variable to store it.

passwordEncrypted - Password Encrypted

`boolean`. Optional. Use when `registryCredentials = UsernamePassword`. Default value: `true`.

Encrypts your password using [Invoke-ServiceFabricEncryptText](#). If you do not encrypt your password, and a certificate matching the Server Certificate Thumbprint in the Cluster Service Connection is installed on the build agent, it will be used to encrypt the password; otherwise, an error will occur.

upgrade - Upgrade

`boolean`. Default value: `false`.

Upgrades an existing deployment rather than removing it.

deployTimeoutSec - Deploy Timeout (s)

`string`.

Specifies the timeout, in seconds, for deploying the application.

removeTimeoutSec - Remove Timeout (s)

`string`.

Specifies the timeout, in seconds, for removing an existing application.

getStatusTimeoutSec - Get Status Timeout (s)

`string`.

Specifies the timeout, in seconds, for getting the status of an existing application.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to deploy a Docker-compose application to a Service Fabric cluster. This task deploys an Azure Service Fabric application to a cluster according to the settings defined in the compose file.

ⓘ Note

This task is currently in preview and requires a preview version of Service Fabric that supports compose deploy. See [Docker Compose deployment support in Azure Service Fabric](#).

Service Fabric

- This task uses a Service Fabric installation to connect and deploy to a Service Fabric cluster.
- Download and install [Azure Service Fabric Core SDK](#) on the build agent.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: Cmd
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any

Requirement	Description
Agent version	1.95.0 or greater
Task category	Deploy

ServiceFabricPowerShell@1 - Service Fabric PowerShell v1 task

Article • 09/26/2023

Use this task to run a PowerShell script within the context of an Azure Service Fabric cluster connection. Runs any PowerShell command or script in a PowerShell session that has a Service Fabric cluster connection initialized.

Syntax

YAML

```
# Service Fabric PowerShell v1
# Run a PowerShell script in the context of an Azure Service Fabric cluster
connection.
- task: ServiceFabricPowerShell@1
  inputs:
    clusterConnection: # string. Alias: serviceConnectionName. Required.
Cluster Service Connection.
    ScriptType: 'FilePath' # 'FilePath' | 'InlineScript'. Required. Script
Type. Default: FilePath.
    #ScriptPath: # string. Optional. Use when ScriptType = FilePath. Script
Path.
    #Inline: # string. Optional. Use when ScriptType = InlineScript. Inline
Script.
    #ScriptArguments: # string. Script Arguments.
```

Inputs

clusterConnection - Cluster Service Connection

Input alias: `serviceConnectionName`. `string`. Required.

Specifies the Azure Service Fabric cluster which will have an established service connection when the specified PowerShell script is executed.

ScriptType - Script Type

`string`. Required. Allowed values: `FilePath` (Script File Path), `InlineScript` (Inline Script). Default value: `FilePath`.

Specifies whether the script is provided as a file or inline in the task.

ScriptPath - Script Path

`string`. Optional. Use when `ScriptType = FilePath`.

Specifies the path to the PowerShell script to run. Can include wildcards and variables.

Example: `$(system.defaultworkingdirectory)/**/drop/projectartifacts/**/docker-compose.yml`.

Note

Combining Compose files is not supported as part of this task.

Inline - Inline Script

`string`. Optional. Use when `ScriptType = InlineScript`. Default value: `# You can write your PowerShell scripts inline here. \n# You can also pass predefined and custom variables to this script using arguments.`

Specifies the PowerShell commands to run on the build agent. Learn more about [PowerShell tasks](#).

ScriptArguments - Script Arguments

`string`.

Specifies the additional parameters to pass to PowerShell. Can be either ordinal or named parameters.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to run a PowerShell script within the context of an Azure Service Fabric cluster connection. Runs any PowerShell command or script in a PowerShell session that has a Service Fabric cluster connection initialized.

Service Fabric

- This task uses a Service Fabric installation to connect and deploy to a Service Fabric cluster.
- [Azure Service Fabric Core SDK](#) on the build agent.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: Cmd
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.95.0 or greater
Task category	Utility

ShellScript@2 - Shell script v2 task

Article • 09/26/2023

Use this task to run a shell script using `bash`.

Syntax

YAML

```
# Shell script v2
# Run a shell script using Bash.
- task: ShellScript@2
  inputs:
    scriptPath: # string. Required. Script Path.
    #args: # string. Arguments.
    # Advanced
    #disableAutoCwd: false # boolean. Specify Working Directory. Default:
    false.
    # cwd: # string. Optional. Use when disableAutoCwd = true. Working
    Directory.
    #failOnStandardError: false # boolean. Fail on Standard Error. Default:
    false.
```

Inputs

`scriptPath` - Script Path

`string`. Required.

Specifies the relative path from the repo root to the shell script file that you want to run.

`args` - Arguments

`string`.

Specifies the arguments that you want to pass to the script.

`disableAutoCwd` - Specify Working Directory

`boolean`. Default value: `false`.

Specifies the working directory where the task runs the script. If the value is left empty, the task defaults to the folder where the script is located.

`cwd` - Working Directory

`string`. Optional. Use when `disableAutoCwd = true`.

Specifies the working directory where the script is run. If the value is left empty, the task uses the root of the repo (build) or artifacts (release), which is `$(System.DefaultWorkingDirectory)`.

`failOnStandardError` - Fail on Standard Error

`boolean`. Default value: `false`.

If the value is `true`, the task will fail if errors are written to the StandardError stream.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Where can I learn about Bash scripts?

- [Beginners/BashScripting](#) ↗ to get started.
- [Awesome Bash](#) ↗ to go deeper.

How do I set a variable so that it can be read by subsequent scripts and tasks?

To learn more about defining build variables in a script, see [Define and modify your build variables in a script](#).

To learn more about defining release variables in a script, see [Define and modify your release variables in a script](#)

Examples

Create `test.sh` at the root of your repo. We recommend creating this file from a Linux environment (such as a real Linux machine or Windows Subsystem for Linux) so that line endings are correct. Also, don't forget to `chmod +x test.sh` before you commit it.

```
sh

#!/bin/bash
echo "Hello World"
echo "AGENT_WORKFOLDER is $AGENT_WORKFOLDER"
echo "AGENT_WORKFOLDER contents:"
ls -1 $AGENT_WORKFOLDER
echo "AGENT_BUILDDIRECTORY is $AGENT_BUILDDIRECTORY"
echo "AGENT_BUILDDIRECTORY contents:"
ls -1 $AGENT_BUILDDIRECTORY
echo "SYSTEM_HOSTTYPE is $SYSTEM_HOSTTYPE"
echo "Over and out."
```

Add the following task to your pipeline to run the previous script.

```
yml

- task: ShellScript@2
  inputs:
    scriptPath: 'test.sh'
```

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: sh
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.

Requirement	Description
Task category	Utility

SqlDacpacDeploymentOnMachineGroup

@0 - SQL Server database deploy v0 task

Article • 09/26/2023

Use this task to deploy a SQL Server database using DACPAC or SQL scripts.

Syntax

YAML

```
# This task is supported on classic release pipelines only.  
# Use the classic designer to add and configure this task in a classic  
release pipeline.  
# See the following Inputs section for details on the inputs that this task  
supports.
```

Inputs

TaskType - Deploy SQL Using

`string`. Required. Allowed values: `dacpac` (Sql Dacpac), `sqlQuery` (Sql Query File), `sqlInline` (Inline Sql). Default value: `dacpac`.

Specifies the way you want to deploy the database: using Dacpac or SQL Scripts.

DacpacFile - DACPAC File

`string`. Required when `TaskType = dacpac`.

Specifies the location of the DACPAC file on the target machines or on a UNC path, like `\BudgetIT\Web\Deploy\FabrikamDB.dacpac`. The UNC path should be accessible to the machine's administrator account. Environment variables are also supported, like `$env:windir`, `$env:systemroot`, or `$env:windir\FabrikamFibre\DB`. Wildcards can be used. For example, `**/*.dacpac` for the DACPAC file that's present in all sub folders.

SqlFile - Sql File

`string`. Required when `TaskType = sqlQuery`.

Specifies the location of the SQL file on the target. Provide a semi-colon separated list of SQL script files to execute multiple files. The SQL scripts are executed in the order given. The location can also be a UNC path, like `\BudgetIT\Web\Deploy\FabrikamDB.sql`. The UNC path should be accessible to the machine's administrator account. Environment variables are also supported, like `$env:windir`, `$env:systemroot`, or `$env:windir\FabrikamFibre\DB`. Wildcards can be used. For example, `**/*.sql` for the SQL file present in all sub folders.

ExecuteInTransaction - Execute within a transaction

`boolean`. Optional. Use when `TaskType = sqlQuery`. Default value: `false`.

Executes the SQL script(s) within a transaction.

ExclusiveLock - Acquire an exclusive app lock while executing script(s)

`boolean`. Optional. Use when `ExecuteInTransaction = true`. Default value: `false`.

Acquires an exclusive app lock while executing script(s).

AppLockName - App lock name

`string`. Required when `ExclusiveLock = true`.

Specifies the app lock name.

InlineSql - Inline Sql

`string`. Required when `TaskType = sqlInline`.

Specifies the SQL queries inline.

TargetMethod - Specify SQL Using

`string`. Required when `TaskType = dacpac`. Allowed values: `server`, `connectionString` (Connection String), `publishProfile` (Publish Profile). Default value: `server`.

Specifies the option to connect to the target SQL Server database. You can provide the SQL Server database details, the SQL Server connection string, or the publish profile XML file.

ServerName - Server Name

`string`. Required when `TargetMethod = server || TaskType = sqlQuery || TaskType = sqlInline`. Default value: `localhost`.

Specifies the SQL Server name, like `machinename\FabriakmSQL,1433`, `localhost`, or `.\SQL2012R2`. Specifying `localhost` will connect to the default SQL Server instance on the machine.

DatabaseName - Database Name

`string`. Required when `TargetMethod = server || TaskType = sqlQuery || TaskType = sqlInline`.

Specifies the name of the SQL Server database.

AuthScheme - Authentication

`string`. Required when `TargetMethod = server || TaskType = sqlQuery || TaskType = sqlInline`. Allowed values: `windowsAuthentication` (Windows Authentication), `sqlServerAuthentication` (SQL Server Authentication). Default value: `windowsAuthentication`.

Specifies the authentication mode for connecting to the SQL Server. In Windows authentication mode, the account used to configure the deployment agent is used to connect to the SQL Server. In SQL Server authentication mode, the SQL login and password must be provided in the parameters below.

SqlUsername - SQL User name

`string`. Required when `AuthScheme = sqlServerAuthentication`.

Specifies the SQL login to connect to the SQL Server. This option is only available if SQL Server authentication mode has been selected.

SqlPassword - SQL Password

`string`. Required when `AuthScheme = sqlServerAuthentication`.

Specifies the password of the SQL login. This option is only available if SQL Server authentication mode has been selected.

ConnectionString - Connection String

`string`. Required when `TargetMethod = connectionString`.

Specifies the SQL Server connection string, like

```
Server=localhost;Database=Fabrikam;User  
ID=AccountPlaceholder;Password=PasswordPlaceholder; .
```

PublishProfile - Publish Profile

`string`. Optional. Use when `TaskType = dacpac`.

Provides fine-grained control over SQL Server database deployments.

Specifies the path to the publish profile XML file on the target machine or on a UNC share that is accessible by the machine administrator's credentials.

AdditionalArguments - Additional Arguments

`string`. Optional. Use when `TaskType = dacpac`.

Specifies additional `SqlPackage.exe` arguments that will be applied when deploying the SQL Server database, like `/p:IgnoreAnsiNulls=True` or `/p:IgnoreComments=True`. These arguments will override the settings in the publish profile XML file (if provided).

AdditionalArgumentsSql - Additional Arguments

`string`. Optional. Use when `TaskType = sqlQuery || TaskType = sqlInline`.

Specifies additional `Invoke-Sqlcmd` arguments that are applied when deploying the SQL Server database.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.102.0 or greater
Task category	Deploy

SqlServerDacpacDeployment@1 - SQL Server database deploy (Deprecated) v1 task

Article • 09/26/2023

Use this task to deploy a SQL Server database using DACPAC.

This task is deprecated.

Syntax

YAML

```
# SQL Server database deploy (Deprecated) v1
# Deploy a SQL Server database using DACPAC.
- task: SqlServerDacpacDeployment@1
  inputs:
    EnvironmentName: # string. Required. Machines.
    #AdminUserName: # string. Admin Login.
    #AdminPassword: # string. Password.
    #Protocol: # 'Http' | 'Https'. Protocol.
    #TestCertificate: true # boolean. Optional. Use when Protocol = Https.
    Test Certificate. Default: true.
    # Deployment
    DacpacFile: # string. Required. DACPAC File.
    # Target
    TargetMethod: 'server' # 'server' | 'connectionString' |
    'publishProfile'. Required. Specify SQL Using. Default: server.
    ServerName: 'localhost' # string. Required when TargetMethod = server.
    Server Name. Default: localhost.
    DatabaseName: # string. Required when TargetMethod = server. Database
    Name.
    #SqlUsername: # string. Optional. Use when TargetMethod = server. SQL
    Username.
    #SqlPassword: # string. Optional. Use when TargetMethod = server. SQL
    Password.
    #ConnectionString: # string. Required when TargetMethod =
    connectionString. Connection String.
    #PublishProfile: # string. Publish Profile.
    #AdditionalArguments: # string. Additional Arguments.
    # Advanced
    #DeployInParallel: true # boolean. Deploy in Parallel. Default: true.
    #ResourceFilteringMethod: 'machineNames' # 'machineNames' | 'tags'.
    Select Machines By. Default: machineNames.
    #MachineFilter: # string. Deploy to Machines.
```

Inputs

EnvironmentName - Machines

`string`. Required.

Specifies a comma-separated list of machine IP addresses or FQDNs along with ports. The default port is based on the selected protocol. For example:

`dbserver.fabrikam.com,dbserver_int.fabrikam.com:5986,192.168.12.34:5986` Output variables of other tasks can also be provided, for example `$(variableName)`.

AdminUserName - Admin Login

`string`.

Specifies the administrator login for the target machines.

AdminPassword - Password

`string`.

Specifies the administrator password for the target machines. Variables defined in build or release definitions are accepted as `$(passwordVariable)`. You can mark the variable type as `secret` to secure it.

Protocol - Protocol

`string`. Allowed values: `Http`, `Https`.

Specifies the protocol to use for the WinRM connection with the machine(s). The default value is `HTTPS`.

TestCertificate - Test Certificate

`boolean`. Optional. Use when `Protocol = Https`. Default value: `true`.

Skips the authenticity validation of the machine's certificate by a trusted certification authority. The parameter is required for the WinRM HTTPS protocol.

DacpacFile - DACPAC File

`string`. Required.

Specifies the location of the DACPAC file on the target machines or on a UNC path, like `\BudgetIT\Web\Deploy\FabrikamDB.dacpac`. The UNC path should be accessible to the machine's administrator account. Environment variables are also supported, like `$env:windir`, `$env:systemroot`, and `$env:windir\FabrikamFibre\Web`.

TargetMethod - Specify SQL Using

`string`. Required. Allowed values: `server`, `connectionString` (Connection String), `publishProfile` (Publish Profile). Default value: `server`.

Specifies the option to connect to the target SQL Server database. You can provide SQL Server database details, a SQL Server connection string, or a publish profile XML file.

ServerName - Server Name

`string`. Required when `TargetMethod = server`. Default value: `localhost`.

Specifies the SQL Server name, like `machinename\FabrikamSQL,1433` or `localhost` or `.\SQL2012R2`. Specifying `localhost` connects to the default SQL Server instance on the machine.

DatabaseName - Database Name

`string`. Required when `TargetMethod = server`.

Specifies the name of the SQL Server database.

SqlUsername - SQL Username

`string`. Optional. Use when `TargetMethod = server`.

If the SQL Server login is specified, it is used to connect to the SQL Server. The default, Integrated Authentication, uses the machine administrator's credentials.

SqlPassword - SQL Password

`string`. Optional. Use when `TargetMethod = server`.

If the SQL Server login user name is specified, provide the SQL Server password. The default, Integrated Authentication, uses the machine administrator's credentials.

ConnectionString - Connection String

`string`. Required when `TargetMethod = connectionString`.

Specifies the SQL Server connection string, like

```
Server=localhost;Database=Fabrikam;User  
ID=AccountPlaceholder;Password=PasswordPlaceholder; .
```

PublishProfile - Publish Profile

`string`.

Provides fine-grained control over SQL Server database creation or upgrades. Specifies the path to the publish profile XML file on the target machine or on a UNC share that is accessible by the machine administrator's credentials.

AdditionalArguments - Additional Arguments

`string`.

Specifies additional `SqlPackage.exe` arguments that are applied when creating or updating the SQL Server database, like `/p:IgnoreAnsiNulls=True` or `/p:IgnoreComments=True`. These arguments will override the settings in the publish profile XML file (if provided).

DeployInParallel - Deploy in Parallel

`boolean`. Default value: `true`.

When set to `true`, runs the database deployment task in parallel on the target machines.

ResourceFilteringMethod - Select Machines By

`string`. Allowed values: `machineNames` (Machine Names), `tags`. Default value: `machineNames`.

Optional. Specifies a subset of machines by providing machine names or tags.

MachineFilter - Deploy to Machines

`string`.

This input is only valid for machine groups and is not supported for a flat list of machines or output variables yet.

Specifies a list of machines, like `dbserver.fabrikam.com`, `webserver.fabrikam.com`, `192.168.12.34`, or tags, like `Role:DB; OS:Win8.1`. If multiple tags are provided, the task runs in all machines with the specified tags. For Azure Resource Groups, provide the virtual machine's name, like `ffweb` or `ffdb`. The default runs the task in all machines.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.96.2 or greater
Task category	Deploy

SSH@0 - SSH v0 task

Article • 09/26/2023

Use this task to run shell commands or a script on a remote machine using SSH. This task enables you to connect to a remote machine using SSH and run commands or a script.

Syntax

YAML

```
# SSH v0
# Run shell commands or a script on a remote machine using SSH.
- task: SSH@0
  inputs:
    sshEndpoint: # string. Required. SSH service connection.
    runOptions: 'commands' # 'commands' | 'script' | 'inline'. Required.
    Run. Default: commands.
    commands: # string. Required when runOptions = commands. Commands.
    #scriptPath: # string. Required when runOptions = script. Shell script
    path.
    #inline: # string. Required when runOptions = inline. Inline Script.
    #interpreterCommand: '/bin/bash' # string. Optional. Use when runOptions
    = inline. Interpreter command. Default: /bin/bash.
    #args: # string. Optional. Use when runOptions = script. Arguments.
    # Advanced
    #failOnStdErr: true # boolean. Fail on STDERR. Default: true.
    #interactiveSession: false # boolean. Enable interactive session.
    Default: false.
    readyTimeout: '20000' # string. Required. SSH handshake timeout.
    Default: 2000.
    #interactiveKeyboardAuthentication: false # boolean. Use interactive-
    keyboard authentication. Default: false.
```

Inputs

`sshEndpoint` - SSH service connection

`string`. Required.

Specifies the name of an SSH service connection containing connection details for the remote machine. The hostname or IP address of the remote machine, the port number, and the user name are required to create an SSH service connection.

- The private key and the passphrase must be specified for authentication.

- A password can be used to authenticate to remote Linux machines, but this is not supported for macOS or Windows systems.

`runOptions` - Run

`string`. Required. Allowed values: `commands`, `script` (Script File), `inline` (Inline Script).

Default value: `commands`.

Runs shell commands or a shell script on the remote machine.

`commands` - Commands

`string`. Required when `runOptions = commands`.

Specifies the shell commands to run on the remote machine. This parameter is available only when **Commands** is selected for the **Run** option. Enter each command together with its arguments on a new line of the multi-line textbox. To run multiple commands together, enter them on the same line separated by semicolons. Example: `cd /home/user/myFolder;build.`

ⓘ Note

Each command runs in a separate process. If you want to run a series of commands that are interdependent (for example, changing the current folder before executing a command), use the **Inline Script** option instead.

`scriptPath` - Shell script path

`string`. Required when `runOptions = script`.

Specifies the path to the shell script file to run on the remote machine. This parameter is available only when **Shell script** is selected for the **Run** option.

`inline` - Inline Script

`string`. Required when `runOptions = inline`.

Writes the shell script to run on the remote machine.

`interpreterCommand` - Interpreter command

`string`. Optional. Use when `runOptions = inline`. Default value: `/bin/bash`.

Specifies the path to the command interpreter used to execute the script. Adds a shebang line to the beginning of the script. Relevant only for UNIX-like operating systems. Use an empty string for Windows-based remote hosts. Learn more about [shebang \(#!\) ↴](#).

`args` - Arguments

`string`. Optional. Use when `runOptions = script`.

Specifies the arguments to pass to the shell script. This parameter is available only when **Shell script** is selected for the **Run** option.

`failOnStdErr` - Fail on STDERR

`boolean`. Default value: `true`.

If the value is `true`, the build fails when the remote commands or script write to `STDERR`.

`interactiveSession` - Enable interactive session

`boolean`. Default value: `false`.

Starts an interactive session. Password requests are filled by the user's password.

Interactive sessions can be useful for running commands, such as `sudo`.

`readyTimeout` - SSH handshake timeout

`string`. Required. Default value: `20000`.

Specifies how long (in milliseconds) the task waits for the SSH handshake to complete.

`interactiveKeyboardAuthentication` - Use interactive-keyboard authentication

`boolean`. Default value: `false`.

Enables interactive-keyboard authentication. Set to `true` if your destination SSH server requires Interactive Keyboard Authentication (`PasswordAuthentication` is disabled on the target machine/set to No in `sshd_config`).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to run shell commands or a script on a remote machine using SSH. This task enables you to connect to a remote machine using SSH and run commands or a script.

Prerequisites

- The task supports use of an SSH key pair to connect to the remote machine(s).
- The public key must be pre-installed or copied to the remote machine(s).

Supported algorithms

Key pair algorithms

- RSA
- DSA

Encryption algorithms

- aes256-cbc
- aes192-cbc
- aes128-cbc
- blowfish-cbc
- 3des-cbc
- arcfour256
- arcfour128
- cast128-cbc
- arcfour

For OpenSSL v1.0.1 and higher (on agent):

- aes256-ctr

- aes192-ctr
- aes128-ctr

For OpenSSL v1.0.1 and higher, NodeJS v0.11.12 and higher (on agent):

- aes128-gcm
- aes128-gcm@openssh.com
- aes256-gcm
- aes256-gcm@openssh.com

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Deploy

See also

- [Install SSH Key task](#)
- [Copy Files Over SSH](#)
- Blog post [SSH build task ↗](#)

UniversalPackages@0 - Universal packages v0 task

Article • 09/26/2023

Use this task to download, or package and publish Universal Packages.

Syntax

YAML

```
# Universal packages v0
# Download or publish Universal Packages.
- task: UniversalPackages@0
  inputs:
    command: 'download' # 'download' | 'publish'. Required. Command.
    Default: download.
    downloadDirectory: '$(System.DefaultWorkingDirectory)' # string.
    Required when command = download. Destination directory. Default:
    $(System.DefaultWorkingDirectory).
    #publishDirectory: '$(Build.ArtifactStagingDirectory)' # string.
    Required when command = publish. Path to file(s) to publish. Default:
    $(Build.ArtifactStagingDirectory).
    # Feed & package details
    feedsToUse: 'internal' # 'internal' | 'external'. Alias:
    internalOrExternalDownload. Required when command = download. Feed location.
    Default: internal.
    #externalFeedCredentials: # string. Alias: externalEndpoint. Optional.
    Use when internalOrExternalDownload = external && command = download.
    organization/collection connection.
    #vstsFeed: # string. Alias: feedListDownload. Required when
    internalOrExternalDownload = internal && command = download. Feed.
    #vstsFeedPackage: # string. Alias: packageListDownload. Required when
    internalOrExternalDownload = internal && command = download. Package name.
    #vstsPackageVersion: # string. Alias: versionListDownload. Required when
    internalOrExternalDownload = internal && command = download. Version.
    #feedDownloadExternal: # string. Required when
    internalOrExternalDownload = external && command = download. Feed (or
    Project/Feed if the feed was created in a project).
    #packageDownloadExternal: # string. Required when
    internalOrExternalDownload = external && command = download. Package name.
    #versionDownloadExternal: # string. Required when
    internalOrExternalDownload = external && command = download. Version.
    # Feed & package details
    #feedsToUsePublish: 'internal' # 'internal' | 'external'. Alias:
    internalOrExternalPublish. Required when command = publish. Feed location.
    Default: internal.
    #publishFeedCredentials: # string. Alias: externalEndpoints. Required
    when internalOrExternalPublish = external && command = publish.
    organization/collection connection.
```

```
#vstsFeedPublish: # string. Alias: feedListPublish. Required when internalOrExternalPublish = internal && command = publish. Destination Feed.  
#vstsFeedPackagePublish: # string. Alias: packageListPublish. Required when internalOrExternalPublish = internal && command = publish. Package name.  
#feedPublishExternal: # string. Required when internalOrExternalPublish = external && command = publish. Feed (or Project/Feed if the feed was created in a project).  
#packagePublishExternal: # string. Required when internalOrExternalPublish = external && command = publish. Package name.  
#versionOption: 'patch' # 'major' | 'minor' | 'patch' | 'custom'. Alias: versionPublishSelector. Required when command = publish. Version. Default: patch.  
#versionPublish: # string. Required when versionPublishSelector = custom && command = publish. Custom version.  
#packagePublishDescription: # string. Optional. Use when command = publish. Description.  
# Advanced  
#publishPackageMetadata: true # boolean. Optional. Use when command = publish && internalOrExternalPublish = internal. Publish pipeline metadata. Default: true.  
#verbosity: 'None' # 'None' | 'Trace' | 'Debug' | 'Information' | 'Warning' | 'Error' | 'Critical'. Verbosity. Default: None.  
# Output  
#publishedPackageVar: # string. Optional. Use when command = publish. Package Output Variable.
```

Inputs

`command` - Command

`string`. Required. Allowed values: `download`, `publish`. Default value: `download`.

Specifies the NuGet command to run.

`downloadDirectory` - Destination directory

`string`. Required when `command = download`. Default value: `$(System.DefaultWorkingDirectory)`.

Specifies the folder path where the task downloads the package's contents.

`feedsToUse` - Feed location

Input alias: `internalOrExternalDownload`. `string`. Required when `command = download`. Allowed values: `internal` (This organization/collection), `external` (Another organization/collection). Default value: `internal`.

Specifies a feed from this collection or another collection in Azure Artifacts.

externalFeedCredentials - organization/collection connection

Input alias: `externalEndpoint`. `string`. Optional. Use when `internalOrExternalDownload = external && command = download`.

Specifies the credentials to use for external registries located in the selected `NuGet.config`. For feeds in this organization or collection, leave this blank; the build's credentials are used automatically.

vstsFeed - Feed

Input alias: `feedListDownload`. `string`. Required when `internalOrExternalDownload = internal && command = download`.

Includes the selected feed. You must have Azure Artifacts installed and licensed to select a feed here. Specifies the *FeedName* for an organization-scoped feed and *projectName/FeedName* or *ProjectID/FeedID* for a project-scoped feed.

vstsFeedPackage - Package name

Input alias: `packageListDownload`. `string`. Required when `internalOrExternalDownload = internal && command = download`.

Specifies the name of the package for the task to download.

vstsPackageVersion - Version

Input alias: `versionListDownload`. `string`. Required when `internalOrExternalDownload = internal && command = download`.

Specifies the package version or uses a variable containing the version to download. This entry can also be a wildcard expression, such as `*`, to get the highest version. Examples: `1.*` gets the highest version with major version 1, and `1.2.*` gets the highest patch release with major version 1 and minor version 2.

feedDownloadExternal - Feed (or Project/Feed if the feed was created in a project)

`string`. Required when `internalOrExternalDownload = external && command = download`.

Specifies a feed in another organization/collection.

For project-scoped feeds, the value should be `Project/Feed`, where `Project` is the project's name or ID, and `Feed` is the feed's name/ID. For organization-scoped feeds, the value should be only the feed name.

`packageDownloadExternal` - Package name

`string`. Required when `internalOrExternalDownload = external && command = download`.

Specifies the package name to download.

`versionDownloadExternal` - Version

`string`. Required when `internalOrExternalDownload = external && command = download`.

Specifies the package version or uses a variable containing the version to download.

This entry can also be a wildcard expression, such as `*`, to get the highest version.

Examples: `1.*` gets the highest version with major version 1, and `1.2.*` gets the highest patch release with major version 1 and minor version 2. Wildcard patterns are not supported with pre-release packages.

`publishDirectory` - Path to file(s) to publish

`string`. Required when `command = publish`. Default value:

`$(Build.ArtifactStagingDirectory)`.

Specifies the path to list of files to be published.

`feedsToUsePublish` - Feed location

Input alias: `internalOrExternalPublish`. `string`. Required when `command = publish`.

Allowed values: `internal` (This organization/collection), `external` (Another organization/collection). Default value: `internal`.

Specifies a feed from this collection or another collection in Azure Artifacts.

`publishFeedCredentials` - organization/collection connection

Input alias: `externalEndpoints`. `string`. Required when `internalOrExternalPublish = external && command = publish`.

Specifies the credentials to use for external feeds.

`vstsFeedPublish` - Destination Feed

Input alias: `feedListPublish`. `string`. Required when `internalOrExternalPublish = internal && command = publish`.

Specifies the project and the feed's name/GUID to publish to.

`publishPackageMetadata` - Publish pipeline metadata

`boolean`. Optional. Use when `command = publish && internalOrExternalPublish = internal`. Default value: `true`.

Associates this build/release pipeline's metadata (such as run # and source code information) with the package.

`vstsFeedPackagePublish` - Package name

Input alias: `packageListPublish`. `string`. Required when `internalOrExternalPublish = internal && command = publish`.

Specifies a package ID to publish or creates a new package ID if you've never published a version of this package before. Package names must be lower case and can only use letters, numbers, and dashes (-).

`feedPublishExternal` - Feed (or Project/Feed if the feed was created in a project)

`string`. Required when `internalOrExternalPublish = external && command = publish`.

Specifies the external feed name to publish to.

If the feed was created in a project, the value should be `Project/Feed`, where `Project` is the project's name or ID, and `Feed` is the feed's name. If the feed was not created in a project, the value should be only the feed name.

`packagePublishExternal` - Package name

`string`. Required when `internalOrExternalPublish = external && command = publish`.

Specifies the package name when publishing to an external feed.

`versionOption` - Version

Input alias: `versionPublishSelector`. `string`. Required when `command = publish`. Allowed

values: `major` (Next major), `minor` (Next minor), `patch` (Next patch), `custom`. Default value: `patch`.

Specifies a version increment strategy. The `custom` value to input your package version manually. For new packages, the first version will be 1.0.0 if you specify `major`, 0.1.0 if you specify `minor`, or 0.0.1 if you specify `patch`. See the [Semantic Versioning spec](#) for more information.

versionPublish - Custom version

`string`. Required when `versionPublishSelector = custom && command = publish`.

Specifies a custom version schema for the package.

packagePublishDescription - Description

`string`. Optional. Use when `command = publish`.

Specifies the description of the package contents and/or the changes made in this version of the package.

verbosity - Verbosity

`string`. Allowed values: `None`, `Trace`, `Debug`, `Information`, `Warning`, `Error`, `Critical`.

Default value: `None`.

Specifies the amount of detail displayed in the output.

publishedPackageVar - Package Output Variable

`string`. Optional. Use when `command = publish`.

Specifies a name for the variable that will contain the published package name and version.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to download, or package and publish Universal Packages.

My Pipeline needs to access a feed in a different project

If the pipeline is running in a different project than the project hosting the feed, you must set up the other project to grant read/write access to the build service. See [Package permissions in Azure Pipelines](#) for more details.

Examples

The simplest way to get started with the Universal Package task is to use the Pipelines task editor to generate the YAML. You can then copy the generated code into your project's `azure-pipelines.yml` file. In this example, the sample demonstrates how to quickly generate the YAML using a pipeline that builds a GatsbyJS progressive web app (PWA).

Universal Packages are a useful way to both encapsulate and version a web app. Packaging a web app into a Universal Package enables quick rollbacks to a specific version of your site and eliminates the need to build the site in the deployment pipeline.

This example pipeline demonstrates how to fetch a tool from a feed within your project. The Universal Package task is used to download the tool, run a build, and again uses the Universal Package task to publish the entire compiled GatsbyJS PWA to a feed as a versioned Universal Package.

The screenshot shows the Azure DevOps Pipeline editor. On the left, there's a list of pipeline steps: 'Get sources', 'Agent job 1' (which contains 'Use Node.JS 12.x'), and other tasks like 'Universal download', 'Install dependencies from package.json', 'Run gatsby build', and 'Universal publish'. On the right, the 'Use Node.js ecosystem' task is being configured. It has a 'Display name' of 'Use NodeJS 12.x', a 'Version' of '12.x', and a checked 'Check for Latest Version' option. There are also sections for 'Control Options' and 'Output Variables'.

Download a package with the Universal Package task

The second task in the sample project uses the Universal Package task to fetch a tool, imagemagick, from a feed that is within a different project in the same organization. The tool, imagemagick, is required by the subsequent build step to resize images.

1. Add the Universal Package task by clicking the plus icon, typing "universal" in the search box, and clicking the **Add** button to add the task to your pipeline.

The screenshot shows the Azure DevOps Pipeline editor. A search bar at the top right contains the text 'universal'. Below it, a modal window titled 'Add tasks' is open, showing a list of tasks. One task, 'Universal packages', is highlighted with a red box. It has a description 'Download or publish Universal Packages' and a 'by Microsoft Corporation' note. At the bottom right of the modal is a blue 'Add' button, which is also highlighted with a red box.

2. Click the newly added **Universal Package** task and the **Command** to [Download](#).
3. Choose the **Destination directory** to use for the tool download.
4. Select a source **Feed** that contains the tool, set the **Package name**, and choose **Version** of the imagemagick tool from the source **Feed**.

The screenshot shows the Azure DevOps Pipeline builder interface. On the left, a list of tasks is visible: Get sources, Agent job 1, Use NodeJS 12.x, Universal download, Install dependencies from package.json, Run gatsby build, and another Universal download task. The second Universal download task is selected. On the right, the configuration pane for this task is displayed. The 'Command' field is set to 'Download' and the 'Destination directory' is set to 'Application'. These fields are highlighted with a red border. Below them, the 'Feed & package details' section is shown, with 'Feed' set to 'pwa-test-feed', 'Package name' set to 'imagemagick', and 'Version' set to '1.0.0'. These fields are also highlighted with a red border.

5. After completing the fields, click **View YAML** to see the generated YAML.

The screenshot shows the Azure DevOps Pipeline builder interface again. The 'View YAML' button in the top right of the configuration pane is highlighted with a red border. A modal window titled 'Copy to clipboard' is open, containing the generated YAML code:

```
steps:
- task: UniversalPackages@0
  displayName: 'Universal download'
  inputs:
    downloadDirectory: Application
    vstsFeed: '00000000-0000-0000-000000000000/00000000-0000-000000000001'
    vstsFeedPackage: imagemagick
    vstsPackageVersion: 1.0.0
```

A 'Copy to clipboard' button at the bottom of the modal is also highlighted with a red border.

6. The **Universal Package** task builder generates simplified YAML that contains non-default values. Copy the generated YAML into your `azure-pipelines.yml` file at the

root of your project's git repo.

YAML

```
# Download Universal Package
steps:
- task: UniversalPackages@0
  displayName: 'Universal download'
  inputs:
    downloadDirectory: Application
    vstsFeed: '00000000-0000-0000-0000-000000000000/00000000-0000-0000-0000-000000000001'
    vstsFeedPackage: imagemagick
    vstsPackageVersion: 1.0.0
```

Publish a package with the Universal Package task

The last step in this sample pipeline uses the Universal Package task to upload the production-ready Gatsby PWA that was produced by the `Run gatsby build` step to a feed as a versioned Universal Package. Once in a feed, you have a permanent copy of your complete site that can be deployed to hosting provider and started with `gatsby serve`.

1. Add another Universal Package task to the end of the pipeline by clicking the plus icon, typing "universal" in the search box, and clicking the **Add** button to add the task to your pipeline. This task gathers all of the production-ready assets produced by the `Run gatsby build` step, produce a versioned Universal Package, and publish the package to a feed.

The screenshot shows the Azure DevOps Pipeline editor interface. On the left, there is a list of pipeline steps:

- Get sources (azure-devops-pwa - CI)
- Agent job 1 (Run on agent)
 - Use NodeJS 12.x (PREVIEW) (Use Node.js ecosystem)
 - Universal download (Universal packages)
 - Install dependencies from package.json (npm)
 - Run gatsby build (npm)
 - Universal download (Some settings need attention)

On the right, a search bar at the top right contains the text "universal". Below it, a modal window titled "Add tasks" is open, showing a search result for "Universal packages" by Microsoft Corporation. The "Add" button is visible in the bottom right corner of this modal. At the bottom of the page, there is a "Marketplace" section listing other related tasks.

2. Set the Command to `Publish`.

3. Set Path to file(s) to publish to the directory containing your GatsbyJS project's `package.json`.

4. Choose a destination feed, a package name, and set your versioning strategy.

The screenshot shows the Azure DevOps Pipeline editor for a pipeline named "azure-devops-pwa - CI". The pipeline consists of several tasks:

- "Get sources" (azure-devops-pwa, master branch)
- "Agent job 1" (Run on agent)
 - "Use Node.js 12.x" (PREVIEW, Use Node.js ecosystem)
 - "Universal download" (Universal packages)
 - "Install dependencies from package.json" (npm)
 - "Run gatsby build" (npm)
- "Universal publish" (Universal packages, selected)

The "Universal publish" task is highlighted with a blue selection bar. Its configuration options are displayed on the right side of the screen, with several fields highlighted by red boxes:

- Command ***: `Publish`
- Path to file(s) to publish ***: `$(Build.ArtifactStagingDirectory)`
- Feed & package details**
 - Feed location ***: This organization/collection Another organization/collection
 - Destination Feed ***: `azure-devops-pwa`
 - Package name ***: `mygatsbsite`
 - Version ***: Next patch Next major Next minor Custom
 - Description**: `A test package`

5. After completing the required fields, click **View YAML**.

6. Copy the resulting YAML into your `azure-pipelines.yml` file as before. The YAML for this sample project displays below.

YAML

```
# Publish Universal Package
steps:
- task: UniversalPackages@0
  displayName: 'Universal publish'
  inputs:
    command: publish
    publishDirectory: Application
    vstsFeedPublish: '00000000-0000-0000-000000000000/00000000-0000-0000-000000000002' # You can also use '< projectName >/< feedName >' instead of the GUIDs
```

```
vstsFeedPackagePublish: mygatsbysite  
packagePublishDescription: 'A test package'
```

This example demonstrated how to use the Pipelines task builder to quickly generate the YAML for the Universal Package task, which can then be placed into your `azure-pipelines.yml` file. The Universal Package task builder supports all of the advanced configurations that can be created with **Universal Package** task's arguments.

 **Note**

Publishing a package directly to a view is not supported in Azure Artifacts. You must publish the package to your feed first, then promote it to a view.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Package

ServiceFabricUpdateAppVersions@1 - Update Service Fabric App Versions v1 task

Article • 09/26/2023

Use this task in a build pipeline to automatically update the versions of a packaged Service Fabric app. This task appends a version suffix to all service and app versions, specified in the manifest files, in an Azure Service Fabric app package.

Syntax

YAML

```
# Update Service Fabric App Versions v1
# Automatically updates the versions of a packaged Service Fabric
application.
- task: ServiceFabricUpdateAppVersions@1
  inputs:
    applicationPackagePath: # string. Required. Application Package.
    versionSuffix: '.$(Build.BuildNumber)' # string. Required. Version
    Value. Default: $(Build.BuildNumber).
    #versionBehavior: 'Append' # 'Append' | 'Replace'. Version Behavior.
    Default: Append.
    #updateOnlyChanged: false # boolean. Update only if changed. Default:
    false.
    #pkgArtifactName: # string. Optional. Use when updateOnlyChanged = true.
    Package Artifact Name.
    #logAllChanges: true # boolean. Optional. Use when updateOnlyChanged =
    true. Log all changes. Default: true.
    #compareType: 'LastSuccessful' # 'LastSuccessful' | 'Specific'.
    Optional. Use when updateOnlyChanged = true. Compare against. Default:
    LastSuccessful.
    #buildNumber: # string. Optional. Use when compareType = Specific. Build
    Number.
```

Inputs

`applicationPackagePath` - Application Package

`string`. Required.

Specifies the location of the Service Fabric application package to be deployed to the cluster. Example: `$(system.defaultworkingdirectory)/**/drop/applicationpackage`.

[Variables](#) and wildcards can be used in the path.

versionSuffix - Version Value

`string`. Required. Default value: `.$(Build.BuildNumber)`.

The value used to specify the version in the manifest files.

Tip

You can modify the build number format directly or use a logging command to dynamically set a variable in any format. For example, you can use `$VersionSuffix` defined in a PowerShell task:

```
$versionSuffix = ".${([DateTimeOffset]::UtcNow.ToString('yyyyMMdd.HHmmss'))}"  
  
Write-Host "##vso[task.setvariable variable=VersionSuffix;]$versionSuffix"
```

versionBehavior - Version Behavior

`string`. Allowed values: `Append`, `Replace`. Default value: `Append`.

Appends the version value to existing values in the manifest files or replaces them.

updateOnlyChanged - Update only if changed

`boolean`. Default value: `false`.

Incrementally updates only the packages that have changed. Use the [deterministic compiler flag](#) to ensure builds with the same inputs produce the same outputs.

pkgArtifactName - Package Artifact Name

`string`. Optional. Use when `updateOnlyChanged = true`.

Specifies the name of the artifact containing the application package from the previous build.

logAllChanges - Log all changes

`boolean`. Optional. Use when `updateOnlyChanged = true`. Default value: `true`.

Compares all files in every package and logs if the file was added, removed, or if its content changed. Otherwise, compares files in a package only until the first change is found for faster performance.

compareType - Compare against

`string`. Optional. Use when `updateOnlyChanged = true`. Allowed values: `LastSuccessful` (Last Successful Build), `Specific` (Specific Build). Default value: `LastSuccessful`.

Compares against the last completed and successful build or against a specific build.

buildNumber - Build Number

`string`. Optional. Use when `compareType = Specific`.

Specifies the build number for comparison if the task is comparing against a specific build.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: Cmd
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any

Requirement	Description
Settable variables	Any
Agent version	1.95.0 or greater
Task category	Utility

ServiceFabricUpdateManifests@2 - Update Service Fabric manifests v2 task

Article • 09/26/2023

Use this task in a build pipeline to automatically update the versions of a packaged Service Fabric app. This task appends a version suffix to all service and app versions, specified in the manifest files, in an Azure Service Fabric app package.

Syntax

YAML

```
# Update Service Fabric manifests v2
# Automatically update portions of application and service manifests in a
# packaged Azure Service Fabric application.
- task: ServiceFabricUpdateManifests@2
  inputs:
    updateType: 'Manifest versions' # 'Manifest versions' | 'Docker image
    settings'. Required. Update Type. Default: Manifest versions.
    applicationPackagePath: # string. Required. Application Package.
    #versionSuffix: '.$(Build.BuildNumber)' # string. Required when
    updateType = Manifest versions. Version Value. Default:
    .$(Build.BuildNumber).
    #versionBehavior: 'Append' # 'Append' | 'Replace'. Optional. Use when
    updateType = Manifest versions. Version Behavior. Default: Append.
    #updateOnlyChanged: false # boolean. Optional. Use when updateType =
    Manifest versions. Update only if changed. Default: false.
    #pkgArtifactName: # string. Optional. Use when updateType = Manifest
    versions && updateOnlyChanged = true. Package Artifact Name.
    #logAllChanges: true # boolean. Optional. Use when updateType = Manifest
    versions && updateOnlyChanged = true. Log all changes. Default: true.
    #compareType: 'LastSuccessful' # 'LastSuccessful' | 'Specific'.
    Optional. Use when updateType = Manifest versions && updateOnlyChanged =
    true. Compare against. Default: LastSuccessful.
    #buildNumber: # string. Optional. Use when updateType = Manifest
    versions && compareType = Specific. Build Number.
    #overwriteExistingPkgArtifact: true # boolean. Optional. Use when
    updateType = Manifest versions && updateOnlyChanged = true. Overwrite
    Existing Package Artifact. Default: true.
    #imageNamesPath: # string. Optional. Use when updateType = Docker image
    settings. Image Names Path.
    #imageDigestsPath: # string. Required when updateType = Docker image
    settings. Image Digests Path.
```

Inputs

updateType - Update Type

`string`. Required. Allowed values: `Manifest versions`, `Docker image settings`. Default value: `Manifest versions`.

Specifies the type of update that should be made to the manifest files. In order to use both update types, add an instance of this task to the build pipeline for each type of update to be executed.

applicationPackagePath - Application Package

`string`. Required.

Specifies the path to the application package. [Variables](#) and wildcards can be used in the path. `applicationPackagePath` must not have a trailing slash, either `\` or `/`.

versionSuffix - Version Value

`string`. Required when `updateType = Manifest versions`. Default value: `.$(Build.BuildNumber)`.

Specifies the version in the manifest files.

Tip

You can modify the build number format directly or use a logging command to dynamically set a variable in a format. For example, you can use `$(VersionSuffix)` defined in a PowerShell task:

```
$versionSuffix = ".${([DateTimeOffset]::UtcNow.ToString('yyyyMMdd.HHmmss'))}"
Write-Host "##vso[task.setvariable variable=VersionSuffix;]$versionSuffix"
```

versionBehavior - Version Behavior

`string`. Optional. Use when `updateType = Manifest versions`. Allowed values: `Append`, `Replace`. Default value: `Append`.

Specifies whether to append the version value to existing values in the manifest files or replace them.

`updateOnlyChanged` - Update only if changed

`boolean`. Optional. Use when `updateType = Manifest versions`. Default value: `false`.

Appends the new version suffix to only the packages that have changed from a previous build. If no changes are found, the version suffix from the previous build will be appended.

Note

By default, the compiler will create different outputs even if no changes were made.

Use the [deterministic compiler flag ↗](#) to ensure builds with the same inputs produce the same outputs.

`pkgArtifactName` - Package Artifact Name

`string`. Optional. Use when `updateType = Manifest versions && updateOnlyChanged = true`.

Specifies the name of the artifact containing the application package for comparison.

`logAllChanges` - Log all changes

`boolean`. Optional. Use when `updateType = Manifest versions && updateOnlyChanged = true`. Default value: `true`.

Compares all files in every package and log if the file was added, removed, or if its content changed. Otherwise, this boolean compares files in a package only until the first change is found for faster performance.

`compareType` - Compare against

`string`. Optional. Use when `updateType = Manifest versions && updateOnlyChanged = true`. Allowed values: `LastSuccessful` (Last Successful Build), `Specific` (Specific Build). Default value: `LastSuccessful`.

Specifies whether to compare against the last completed and successful build or against a specific build.

`buildNumber` - Build Number

`string`. Optional. Use when `updateType = Manifest versions && compareType = Specific`.

Specifies the build number for comparison.

overwriteExistingPkgArtifact - Overwrite Existing Package Artifact

`boolean`. Optional. Use when `updateType = Manifest` versions && `updateOnlyChanged = true`. Default value: `true`.

Downloads a new copy of the artifact. Otherwise, this boolean uses an existing copy if present.

imageNamesPath - Image Names Path

`string`. Optional. Use when `updateType = Docker image settings`.

Specifies the path to a text file that contains the names of the Docker images associated with the Service Fabric application that should be updated with digests. Each image name must be on its own line and must be in the same order as the digests in the Image Digests file. If the images are created by the Service Fabric project, this file is generated as part of the Package target, and its output location is controlled by the property `BuiltDockerImagesFilePath`.

imageDigestsPath - Image Digests Path

`string`. Required when `updateType = Docker image settings`.

Specifies the path to a text file that contains the digest values of the Docker images associated with the Service Fabric application. This file can be output by the [Docker task](#) when using the push action. The file should contain lines of text in the format of `registry/image_name@digest_value`.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task in a build pipeline to automatically update the versions of a packaged Service Fabric app. This task appends a version suffix to all service and app versions, specified in the manifest files, in an Azure Service Fabric app package.

ⓘ Note

This task requires Windows PowerShell.

This task is not available in **release** pipelines.

This task can only be used in a build pipeline to automatically update the versions of a packaged Service Fabric app.

This task support two types of updates:

1. **Manifest version:** Updates Service and Application versions specified in manifest files in a Service fabric application package. If specified, `manifest version` compares current files against a previous build and updates the version only for those changed services.
2. **Docker image settings:** Updates docker container image settings specified in manifest files in a Service fabric application package. The image settings to be placed are picked from two files:
 - a. **Image names file:** This file is generated by the build task.
 - b. **Image digests file:** This file is generated by the docker task when it pushes images to registry.

Examples

- [*Service Fabric Application Deployment task](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: Cmd

Requirement	Description
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.95.0 or greater
Task category	Utility

UseDotNet@2 - Use dotnet v2 task

Article • 09/26/2023

Use this task to acquire a specific version of the .NET Core SDK from the internet or the local cache and add it to the PATH. Use this task to change the version of .NET Core that is used in subsequent tasks. This task also provides proxy support.

Syntax

YAML

```
# Use .NET Core v2
# Acquires a specific version of the .NET Core SDK from the internet or the
local cache and adds it to the PATH. Use this task to change the version of
.NET Core used in subsequent tasks. Additionally provides proxy support.
- task: UseDotNet@2
  inputs:
    #packageType: 'sdk' # 'runtime' | 'sdk'. Package to install. Default:
    sdk.
    #useGlobalJson: false # boolean. Optional. Use when packageType = sdk.
    Use global json. Default: false.
    #workingDirectory: # string. Optional. Use when useGlobalJson = true.
    Working Directory.
    #version: # string. Optional. Use when useGlobalJson = false ||
    packageType = runtime. Version.
    #includePreviewVersions: false # boolean. Optional. Use when
    useGlobalJson = false || packageType = runtime. Include Preview Versions.
    Default: false.
    # Advanced
    #vsVersion: # string. Compatible Visual Studio version.
    #installationPath: '$(Agent.ToolsDirectory)/dotnet' # string. Path To
    Install .Net Core. Default: $(Agent.ToolsDirectory)/dotnet.
    #performMultiLevelLookup: false # boolean. Perform Multi Level Lookup.
    Default: false.
```

Inputs

packageType - Package to install

`string`. Allowed values: `runtime`, `sdk` (SDK (contains runtime)). Default value: `sdk`.

Specifies whether to install only the .NET runtime or the SDK.

useGlobalJson - Use global json

`boolean`. Optional. Use when `packageType = sdk`. Default value: `false`.

Installs all SDKs from `global.json` files. These files are searched from `system.DefaultWorkingDirectory`. You can change the search root path by setting the working directory input.

The `6.x` and `6.1.x` format (using `.x` as a wildcard) described in the `UseDotNet@2.version` input is for use in the `version` input in the task, not the `sdk.version` parameter in `global.json`.

If you receive an error message like `##[error]Version 6.0.x is not allowed. Allowed version types are: majorVersion.x, majorVersion.minorVersion.x, majorVersion.minorVersion.patchVersion.` More details: Only explicit versions are accepted, such as: `2.2.301`. Version: `6.0.x` is not valid. and you are using `global.json`, check the `sdk.version` in your `global.json`.

For more information on `global.json`, see [Select the .NET version to use](#).

`workingDirectory` - Working Directory

`string`. Optional. Use when `useGlobalJson = true`.

Specifies the path from where `global.json` files should be searched when using `useGlobalJson`. If the value is empty, `system.DefaultWorkingDirectory` will be considered as the root path.

`version` - Version

`string`. Optional. Use when `useGlobalJson = false || packageType = runtime`.

Specifies the version of the .NET Core SDK or runtime to install. The version value formats are shown with examples:

- `2.x`: Installs the latest SDK or runtime with the specified major version, `2`.
- `3.1.x`: Installs the latest SDK or runtime with the specified major and minor versions, `3` and `1`.
- `3.1.402`: Installs the specified SDK or runtime version, `3.1.402`.

The version values for SDK or runtime installations are in the `releases.json` file. The link to the `releases.json` of a major/minor version is in the [releases-index](#) file. For example, the link to the [releases.json file for version 3.1](#).

`vsVersion` - Compatible Visual Studio version

`string`.

Specifies a compatible Visual Studio version for a corresponding .NET Core SDK installation. The value must be a complete version number, such as `16.6.4`, which contains a major version, a minor version, and a patch number.

The version values for SDK or runtime installations, which are used for the `version` string, are in the `releases.json` file. The link to the `releases.json` of a major/minor version is in the [releases-index ↗](#) file. For example, the link to the [releases.json file for version 3.1 ↗](#).

`includePreviewVersions` - Include Preview Versions

`boolean`. Optional. Use when `useGlobalJson = false || packageType = runtime`. Default value: `false`.

If set to `true`, includes preview versions when the task searches for latest runtime/SDK versions, such as searching for `2.2.x` or `3.1.x`. This setting is ignored if you specify an exact version, such as `3.0.100-preview3-010431`.

`installationPath` - Path To Install .Net Core

`string`. Default value: `$(Agent.ToolsDirectory)/dotnet`.

Specifies where the .NET Core SDK/Runtime should be installed. Different paths can have the following impact on .NET's behavior.

- `$(Agent.ToolsDirectory)`: Using this path caches the installation on the agent, as this directory is not cleaned across pipelines. All pipelines running on the agent have access to the previously installed versions.
- `$(Agent.TempDirectory)`: Using this path ensures that a pipeline doesn't use a cached version of .NET Core, as this folder is cleaned after each pipeline.
- **Another path:** You can use any path if the agent process has access to the path. This will change the state of the machine and impact all processes running on it.

Note

You can use the **Multi-Level Lookup** setting, `performMultiLevelLookup`, to configure how the .NET host searches for versions.

`performMultiLevelLookup` - Perform Multi Level Lookup

`boolean`. Default value: `false`.

Configures the behavior of the .NET host process when it searches for a suitable shared framework. The values are:

- `false`: The host process searches only for versions that are present in the folder that is specified by the task.
- `true`: The host process will search in predefined global locations using multi-level lookup. The default global locations are:
 - `C:\Program Files\dotnet` (64-bit processes)
 - `C:\Program Files (x86)\dotnet` (32-bit processes)

Learn more about [multi-level SharedFX lookup ↗](#).

ⓘ Note

`performMultiLevelLookup` is only applicable to Windows based agents.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

The Use .NET Core task acquires a specific version of [.NET Core](#) from internet or the tools cache and adds it to the PATH of the Azure Pipelines Agent (hosted or private). Use this task to change the version of .NET Core used in subsequent tasks like [DotNetCoreCLI@2](#). Adding this task before the [DotNetCoreCLI@2](#) in a build definition ensures that the version would be available at the time of building, testing and publishing your app.

The tool installer approach also allows you to decouple from the agent update cycles. If the .NET Core version you are looking for is missing from the Azure Pipelines agent

(Hosted or Private), then you can use this task to get the right version installed on the agent.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: DotNetCore
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Tool

UseNode@1 - Use Node.js ecosystem v1 task

Article • 09/26/2023

Use this task to find, download, and cache a specified version of [Node.js](#) and add it to the PATH. This task also provides proxy support.

Syntax

YAML

```
# Use Node.js ecosystem v1
# Set up a Node.js environment and add it to the PATH, additionally
# providing proxy support.
- task: UseNode@1
  inputs:
    #version: '10.x' # string. Version. Default: 10.x.
    #checkLatest: false # boolean. Check for Latest Version. Default: false.
    #force32bit: false # boolean. Use 32 bit version on x64 agents. Default:
    #false.
```

Inputs

`version` - Version

`string`. Default value: `10.x`.

Required. Specifies the [Node.js version](#) using SemVer's version range syntax.

Examples: `10.x`, `10.15.1`, `>=10.15.0`.

`checkLatest` - Check for Latest Version

`boolean`. Default value: `false`.

Checks online for the latest available version that satisfies the version spec. This should be `false` unless you need to always have the latest version. Setting the value to `true` will cause the task to incur download costs that may be unnecessary, especially with the hosted build pool.

`force32bit` - Use 32 bit version on x64 agents

`boolean`. Default value: `false`.

Installs the x86 version of Node.js on a 64-bit Windows agent. Only works on Windows agents.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: Node, npm, node.js
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Tool

UsePythonVersion@0 - Use Python version v0 task

Article • 09/26/2023

Use this task to download or select a version of Python to run on an agent, and optionally add it to PATH.

Syntax

YAML

```
# Use Python version v0
# Use the specified version of Python from the tool cache, optionally adding
it to the PATH.
- task: UsePythonVersion@0
  inputs:
    versionSpec: '3.x' # string. Required. Version spec. Default: 3.x.
    #disableDownloadFromRegistry: false # boolean. Disable downloading
releases from the GitHub registry. Default: false.
    #allowUnstable: false # boolean. Optional. Use when
disableDownloadFromRegistry = false. Allow downloading unstable releases.
Default: false.
    #githubToken: # string. Optional. Use when disableDownloadFromRegistry =
false. GitHub token for GitHub Actions python registry.
    #addToPath: true # boolean. Add to PATH. Default: true.
    # Advanced
    architecture: 'x64' # 'x86' | 'x64'. Required. Architecture. Default:
x64.
```

Inputs

versionSpec - Version spec

string. Required. Default value: 3.x.

Specifies the version range or exact version of a Python version to use, using SemVer's version range syntax. Learn more about [SemVer](#).

disableDownloadFromRegistry - Disable downloading releases from the GitHub registry

boolean. Default value: false.

Disables downloading missing Python versions from the [Github Actions registry](#). This boolean should only be true if using a local installation of Python.

`allowUnstable` - Allow downloading unstable releases

`boolean`. Optional. Use when `disableDownloadFromRegistry = false`. Default value: `false`.

Downloads unstable Python versions from the [Github Actions Python versions registry](#) if set to `true`.

`githubToken` - GitHub token for GitHub Actions python registry

`string`. Optional. Use when `disableDownloadFromRegistry = false`.

Specifies the GitHub token that enforces the anonymous requests limit in the [Github Actions python versions registry](#). Leaving this empty may cause download failures. Not needed if using a local installation of Python.

`addToPath` - Add to PATH

`boolean`. Default value: `true`.

Prepends the retrieved Python version to the PATH environment variable to make it available in subsequent tasks or scripts without using the output variable.

`architecture` - Architecture

`string`. Required. Allowed values: `x86`, `x64`. Default value: `x64`.

Specifies the target architecture (`x86` or `x64`) of the Python interpreter.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

`pythonLocation`

The directory of the installed Python distribution. Use this in subsequent tasks to access this installation of Python.

Remarks

Use this task to download or select a version of Python to run on an agent, and optionally add it to PATH.

Prerequisites

- A [Microsoft-hosted agent](#) with side-by-side versions of Python installed, or a self-hosted agent with `Agent.ToolsDirectory` configured (see [FAQ](#)).
- Downloading python versions is not supported on self-hosted agents.

This task will fail if no Python versions are found in `Agent.ToolsDirectory`. Available Python versions on Microsoft-hosted agents can be found [here](#).

Note

x86 and x64 versions of Python are available on Microsoft-hosted Windows agents, but not on Linux or macOS agents.

As of version 0.150 of the task, version spec will also accept `pypy2` or `pypy3`.

As of version 0.213.1 of the task, version spec will also accept `pypy2.x` or `pypy3.x`.

If the task completes successfully, the task's output variable will contain the directory of the Python installation:

The screenshot shows the 'Use Python Version (Preview)' task configuration. At the top, there are links for 'Link settings', 'View YAML', and 'Remove'. Below that, the 'Version' dropdown is set to '0.* (preview)'. The 'Display name' is 'Use Python Version'. The 'Version spec' is '\$(python.version)'. The 'Add to PATH' checkbox is checked. The 'Control Options' section has an 'Output Variables' subsection. This subsection is highlighted with a red box. It contains a 'Reference name' field (empty) and a 'Variables list' with the entry '.pythonLocation'. There is also a small 'Info' icon next to '.pythonLocation'.

After running this task with "Add to PATH," the `python` command in subsequent scripts will be for the highest available version of the interpreter matching the version spec and architecture.

The versions of Python installed on the Microsoft-hosted Ubuntu and macOS images follow the symlinking structure for Unix-like systems that are defined in [PEP 394](#).

For example, `python3.11` is the actual interpreter for Python 3.11.

`python3` is symlinked to that interpreter, and `python` is a symlink to that symlink.

On the Microsoft-hosted Windows images, the interpreter is just `python`.

For Microsoft-hosted agents, x86 is supported only on Windows. This is because Windows can run executables compiled for the x86 architecture with the WoW64 subsystem. Hosted Ubuntu and Hosted macOS run 64-bit operating systems and run only 64-bit Python.

How can I configure a self-hosted agent to use this task?

ⓘ Important

Downloading python versions is not supported on self-hosted agents. You may only use pre-installed versions.

The desired Python version needs to be added to the tool cache on the self-hosted agent so the task can use it. Normally, the tool cache is located under the `_work/_tool` directory of the agent; alternatively, the path can be overridden by the environment

variable `AGENT_TOOLSDIRECTORY`. Under that directory, create the following directory structure based off of your Python version:

```
$AGENT_TOOLSDIRECTORY/
  Python/
    {version number}/
      {platform}/
        {tool files}
      {platform}.complete
```

The `version number` should follow the format of `1.2.3`. The `platform` should either be `x86` or `x64`. The `tool files` should be the unzipped Python version files. The `{platform}.complete` should be a 0 byte file that looks like `x86.complete` or `x64.complete` and just signifies the tool has been installed in the cache properly.

As a complete and concrete example, here is how a completed download of Python 3.11.4 for x64 would look in the tool cache:

```
$AGENT_TOOLSDIRECTORY/
  Python/
    3.11.4/
      x64/
        {tool files}
      x64.complete
```

Learn more about the [tool cache](#).

To make your scripts work as they would on Microsoft-hosted agents, use the symlink structure from [PEP 394](#) on Unix-like systems.

Also note that the embeddable ZIP release of Python requires [extra configuration for installed modules](#), including `pip`. If possible, we recommend using the [full installer](#) to get a `pip`-compatible Python installation.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup

Requirement	Description
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : pythonLocation, PATH
Agent version	2.182.1 or greater
Task category	Tool

UseRubyVersion@0 - Use Ruby version v0 task

Article • 09/26/2023

Use this task to select a version of Ruby to run on an agent. Optionally, the task can add the Ruby version to PATH.

Syntax

YAML

```
# Use Ruby version v0
# Use the specified version of Ruby from the tool cache, optionally adding
it to the PATH.
- task: UseRubyVersion@0
  inputs:
    versionSpec: '>= 2.4' # string. Required. Version spec. Default: >= 2.4.
    #addToPath: true # boolean. Add to PATH. Default: true.
```

Inputs

versionSpec - Version spec

`string`. Required. Default value: `>= 2.4`.

Specifies the version range or a version of a Ruby version to use.

addToPath - Add to PATH

`boolean`. Default value: `true`.

Optional. Prepends the retrieved Ruby version to the PATH environment variable to make it available in subsequent tasks or scripts without using the output variable.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

This task defines the following [output variables](#), which you can consume in downstream steps, jobs, and stages.

`rubyLocation`

The resolved folder of the Ruby distribution.

Remarks

Use this task to select a version of Ruby to run on an agent, and optionally add it to PATH.

Prerequisites

- A [Microsoft-hosted agent](#) with side-by-side versions of Ruby installed, or a self-hosted agent with `Agent.ToolsDirectory` configured (see [FAQ](#)).

This task will fail if no Ruby versions are found in `Agent.ToolsDirectory`. See other available Ruby versions on [Microsoft-hosted agents](#).

Where can I learn more about tool installers?

For an explanation of tool installers and examples, see [Tool installers](#).

How can I configure a self-hosted agent to use this task?

You can run this task on a self-hosted agent with your own Ruby versions. To run this task on a self-hosted agent, set up `Agent.ToolsDirectory` by following the [Tool Cache instructions](#). The tool name to use is `Ruby`.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None

Requirement	Description
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	This task runs using the following command restrictions : restricted
Settable variables	This task has permission to set the following variables : rubyLocation, PATH
Agent version	2.182.1 or greater
Task category	Tool

See also

- [Tool installers](#)

VSBuild@1 - Visual Studio build v1 task

Article • 09/26/2023

Use this task to build with MSBuild and set the Visual Studio version property. Learn more about installing [Visual Studio images on Azure](#).

Syntax

YAML

```
# Visual Studio build v1
# Build with MSBuild and set the Visual Studio version property.
- task: VSBuild@1
  inputs:
    solution: '**\*.sln' # string. Required. Solution. Default: **\*.sln.
    #vsVersion: 'latest' | '17.0' | '16.0' | '15.0' | '14.0' |
    '12.0' | '11.0'. Visual Studio Version. Default: latest.
    #msbuildArgs: # string. MSBuild Arguments.
    #platform: # string. Platform.
    #configuration: # string. Configuration.
    #clean: false # boolean. Clean. Default: false.
  # Advanced
    #maximumCpuCount: false # boolean. Build in Parallel. Default: false.
    #restoreNugetPackages: false # boolean. Restore NuGet Packages. Default:
    false.
    #msbuildArchitecture: 'x86' # 'x86' | 'x64'. MSBuild Architecture.
    Default: x86.
    #logProjectEvents: true # boolean. Record Project Details. Default:
    true.
    #createLogFile: false # boolean. Create Log File. Default: false.
    #logFileVerbosity: 'normal' | 'quiet' | 'minimal' | 'normal' |
    'detailed' | 'diagnostic'. Optional. Use when createLogFile = true. Log File
    Verbosity. Default: normal.
    #enableDefaultLogger: true # boolean. Enable Default Logger. Default:
    true.
    #customVersion: # string. Custom Version.
```

Inputs

`solution` - Solution

`string`. Required. Default value: `***.sln`.

Specifies the solution for the task to use in the build process.

If you want to build a single solution, click the ... button and specify the solution.

If you want to build multiple solutions, specify the search criteria. You can use a single-folder wildcard (*) and recursive wildcards (**). For example, **.sln searches for all .sln files in all subdirectories.

Make sure the solutions you specify are downloaded by this build pipeline. On the Repository tab:

- If you use TFVC, make sure that the solution is a child of one of the mappings on the Repository tab.
- If you use Git, make sure that the project or solution is in your Git repo, and in a branch that you're building.

Tip

- You can also build MSBuild project (*.proj) files.
- If you are building a customized MSBuild project file, we recommend you use the MSBuild task instead of the Visual Studio Build task.

`vsVersion` - Visual Studio Version

`string`. Allowed values: `latest`, `17.0` (Visual Studio 2022), `16.0` (Visual Studio 2019), `15.0` (Visual Studio 2017), `14.0` (Visual Studio 2015), `12.0` (Visual Studio 2013), `11.0` (Visual Studio 2012). Default value: `latest`.

The value of this input must match the version of Visual Studio used to create your solution.

Adds the `/p:VisualStudioVersion={numeric_visual_studio_version}` argument to the MSBuild command run by the build. For example, if you specify **Visual Studio 2015**, `/p:VisualStudioVersion=14.0` is added to the MSBuild command.

Azure Pipelines: If your team wants to use Visual Studio with the Microsoft-hosted agents, select **windows-latest** as your default build pool. See [Microsoft-hosted agents](#).

`msbuildArgs` - MSBuild Arguments

`string`.

Passes additional arguments to MSBuild. For syntax, see [MSBuild Command-Line Reference](#).

`platform` - Platform

`string`.

Specifies the platform you want to build, such as `Win32`, `x86`, `x64`, or `any cpu`.

💡 Tip

- If you are targeting an MSBuild project (*.proj) file instead of a solution, specify `AnyCPU` (no whitespace).
- Declare a build variable such as `BuildPlatform` on the Variables tab (selecting Allow at Queue Time) and reference it here as `$(BuildPlatform)`. This way you can modify the platform when you queue the build and enable building multiple configurations.

`configuration` - Configuration

`string`.

Specifies the configuration you want to build, such as `debug` or `release`.

💡 Tip

Declare a build variable such as `BuildConfiguration` on the Variables tab (selecting Allow at Queue Time) and reference it here as `$(BuildConfiguration)`. This way you can modify the platform when you queue the build and enable building multiple configurations.

`clean` - Clean

`boolean`. Default value: `false`.

If set to `false`, the task makes an incremental build. This setting might reduce your build time, especially if your codebase is large. This option has no practical effect unless you also set the Clean repository to `false`.

If set to `true`, the task rebuilds all of the code in the code projects. This is equivalent to the MSBuild `/target:clean` argument.

`maximumCpuCount` - Build in Parallel

`boolean`. Default value: `false`.

Optional. If your MSBuild target configuration is compatible with building in parallel, you can check this input to pass the `/m` switch to MSBuild (Windows only). If your target configuration is not compatible with building in parallel, checking this option may cause your build to result in file-in-use errors, or intermittent or inconsistent build failures.

`restoreNugetPackages` - Restore NuGet Packages

`boolean`. Default value: `false`.

This input is deprecated. To restore NuGet packages, add a [NuGet Tool Installer](#) task before the build.

`msbuildArchitecture` - MSBuild Architecture

`string`. Allowed values: `x86` (MSBuild x86), `x64` (MSBuild x64). Default value: `x86`.

Optional. Supplies the architecture (`x86` or `x64`) of MSBuild to run.

Tip

Because Visual Studio runs as a 32-bit application, you may experience problems when your build is processed by a build agent that is running the 64-bit version of Team Foundation Build Service. By selecting MSBuild `x86`, you may resolve these issues.

`logProjectEvents` - Record Project Details

`boolean`. Default value: `true`.

Optional. Records timeline details for each project.

`createLogFile` - Create Log File

`boolean`. Default value: `false`.

Optional. Creates a log file (Windows only).

logFileVerbosity - Log File Verbosity

`string`. Optional. Use when `createLogFile = true`. Allowed values: `quiet`, `minimal`, `normal`, `detailed`, `diagnostic`. Default value: `normal`.

Specifies the verbosity level in log files.

enableDefaultLogger - Enable Default Logger

`boolean`. Default value: `true`.

If set to `true`, enables the default logger for MSBuild.

customVersion - Custom Version

`string`.

Sets a custom version of Visual Studio. Examples: `15.0`, `16.0`, `17.0`. The required version of Visual Studio must be installed in the system.

Azure Pipelines: If your team wants to use Visual Studio 2022 with the Microsoft-hosted agents, select `windows-2022` as your default build pool. For more info see [Microsoft-hosted agents](#).

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Learn more about installing [Visual Studio images on Azure](#).

Important

This task is only supported on agents running Windows.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: msbuild, visualstudio
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.95.0 or greater
Task category	Build

VSTest@2 - Visual Studio Test v2 task

Article • 09/26/2023

Use this task to run unit and functional tests (Selenium, Appium, Coded UI test, etc.) using the Visual Studio Test (VSTest) runner. You can run test frameworks that have a Visual Studio test adapter. Example frameworks are MSTest, xUnit, NUnit, Chutzpah (for JavaScript tests using QUnit, Mocha and Jasmine), etc. Tests can be distributed on multiple agents using this task (version 2).

Syntax

YAML

```
# Visual Studio Test v2
# Run unit and functional tests (Selenium, Appium, Coded UI test, etc.)
using the Visual Studio Test (VsTest) runner. Test frameworks that have a
Visual Studio test adapter such as MsTest, xUnit, NUnit, Chutzpah (for
JavaScript tests using QUnit, Mocha and Jasmine), etc. can be run. Tests can
be distributed on multiple agents using this task (version 2).
- task: VSTest@2
  inputs:
    # Test selection
    testSelector: 'testAssemblies' # 'testAssemblies' | 'testPlan' |
'testRun'. Required. Select tests using. Default: testAssemblies.
    testAssemblyVer2: # string. Required when testSelector = testAssemblies.
Test files.
    #testPlan: # string. Required when testSelector = testPlan. Test plan.
    #testSuite: # string. Required when testSelector = testPlan. Test suite.
    #testConfiguration: # string. Required when testSelector = testPlan.
Test configuration.
    #tcmTestRun: '$(test.RunId)' # string. Optional. Use when testSelector =
testRun. Test Run. Default: $(test.RunId).
    searchFolder: '$(System.DefaultWorkingDirectory)' # string. Required.
Search folder. Default: $(System.DefaultWorkingDirectory).
    #resultsFolder: '$(Agent.TempDirectory)\TestResults' # string. Test
results folder. Default: $(Agent.TempDirectory)\TestResults.
    #testFiltercriteria: # string. Optional. Use when testSelector =
testAssemblies. Test filter criteria.
    #runOnlyImpactedTests: False # boolean. Optional. Use when testSelector
= testAssemblies. Run only impacted tests. Default: False.
    #runAllTestsAfterXBuilds: '50' # string. Optional. Use when testSelector
= testAssemblies && runOnlyImpactedTests = true. Number of builds after
which all tests should be run. Default: 50.
    #uiTests: false # boolean. Test mix contains UI tests. Default: false.
    # Execution options
    #vstestLocationMethod: 'version' # 'version' | 'location'. Select test
platform using. Default: version.
    #vsTestVersion: 'latest' # 'latest' | '17.0' | '16.0' | '15.0' | '14.0'
| 'toolsInstaller'. Optional. Use when vstestLocationMethod = version. Test
```

```
platform version. Default: latest.

    #vstestLocation: # string. Optional. Use when vstestLocationMethod =
location. Path to vstest.console.exe.

    #runSettingsFile: # string. Settings file.

    #overrideTestrunParameters: # string. Override test run parameters.

    #pathToCustomTestAdapters: # string. Path to custom test adapters.

    #runInParallel: False # boolean. Run tests in parallel on multi-core
machines. Default: False.

    #runTestsInIsolation: False # boolean. Run tests in isolation. Default:
False.

    #codeCoverageEnabled: False # boolean. Code coverage enabled. Default:
False.

    #otherConsoleOptions: # string. Other console options.

    #diagnosticsEnabled: false # boolean. Collect advanced diagnostics in
case of catastrophic failures. Default: false.

    #collectDumpOn: 'onAbortOnly' # 'onAbortOnly' | 'always' | 'never'.
Optional. Use when diagnosticsEnabled = true. Collect process dump and
attach to test run report. Default: onAbortOnly.

    #rerunFailedTests: False # boolean. Rerun failed tests. Default: False.

    #rerunType: 'basedOnTestFailurePercentage' #
'basedOnTestFailurePercentage' | 'basedOnTestFailureCount'. Optional. Use
when rerunFailedTests = true. Do not rerun if test failures exceed specified
threshold. Default: basedOnTestFailurePercentage.

    #rerunFailedThreshold: '30' # string. Optional. Use when
rerunFailedTests = true && rerunType = basedOnTestFailurePercentage. %
failure. Default: 30.

    #rerunFailedTestCasesMaxLimit: '5' # string. Optional. Use when
rerunFailedTests = true && rerunType = basedOnTestFailureCount. # of failed
tests. Default: 5.

    #rerunMaxAttempts: '3' # string. Optional. Use when rerunFailedTests =
true. Maximum # of attempts. Default: 3.

    # Advanced execution options

    #distributionBatchType: 'basedOnTestCases' # 'basedOnTestCases' |
'basedOnExecutionTime' | 'basedOnAssembly'. Batch tests. Default:
basedOnTestCases.

    #batchingBasedOnAgentsOption: 'autoBatchSize' # 'autoBatchSize' |
'customBatchSize'. Optional. Use when distributionBatchType =
basedOnTestCases. Batch options. Default: autoBatchSize.

    #customBatchSizeValue: '10' # string. Required when
distributionBatchType = basedOnTestCases && batchingBasedOnAgentsOption =
customBatchSize. Number of tests per batch. Default: 10.

    #batchingBasedOnExecutionTimeOption: 'autoBatchSize' # 'autoBatchSize' |
'customTimeBatchSize'. Optional. Use when distributionBatchType =
basedOnExecutionTime. Batch options. Default: autoBatchSize.

    #customRunTimePerBatchValue: '60' # string. Required when
distributionBatchType = basedOnExecutionTime &&
batchingBasedOnExecutionTimeOption = customTimeBatchSize. Running time (sec)
per batch. Default: 60.

    #dontDistribute: False # boolean. Replicate tests instead of
distributing when multiple agents are used in the job. Default: False.

    # Reporting options

    #testRunTitle: # string. Test run title.

    #platform: # string. Build platform.

    #configuration: # string. Build configuration.

    #publishRunAttachments: true # boolean. Upload test attachments.
```

```
Default: true.  
#failOnMinTestsNotRun: False # boolean. Fail the task if a minimum  
number of tests are not run. Default: False.  
#minimumExpectedTests: '1' # string. Optional. Use when  
failOnMinTestsNotRun = true. Minimum # of tests. Default: 1.
```

Inputs

`testSelector` - Select tests using

`string`. Required. Allowed values: `testAssemblies` (Test assemblies), `testPlan` (Test plan), `testRun` (Test run). Default value: `testAssemblies`.

- **Test assembly:** Specifies one or more test assemblies that contain your tests. You can optionally specify a filter criteria to select only specific tests.
- **Test plan:** Runs tests from your test plan that have an automated test method associated with it. To learn more about how to associate tests with a test case work item, see [Associate automated tests with test cases](#).
- **Test run:** Use this option when you are setting up an environment to run tests from [test plans](#). This option should not be used when running tests in a continuous integration/continuous deployment (CI/CD) pipeline.

`testAssemblyVer2` - Test files

`string`. Required when `testSelector = testAssemblies`. Default value:

`**\bin***test.dll\n**\bin***tests.dll`.

Runs tests from the specified files. Ordered tests and webtests can be run by specifying the `.orderedtest` and `.webtest` files respectively. To run `.webtest`, Visual Studio 2017 Update 4 or higher is needed. The file paths are relative to the search folder. This input supports multiple lines of [minimatch patterns](#).

`testPlan` - Test plan

`string`. Required when `testSelector = testPlan`.

Specifies a test plan containing test suites with automated test cases.

`testSuite` - Test suite

`string`. Required when `testSelector = testPlan`.

Specifies one or more test suites containing automated test cases. Test case work items must be associated with an [automated test method](#).

testConfiguration - Test configuration

`string`. Required when `testSelector = testPlan`.

Specifies the test configuration.

tcmTestRun - Test Run

`string`. Optional. Use when `testSelector = testRun`. Default value: `$(test.RunId)`.

Specifies the test run-based selection that is used when triggering automated test runs from [test plans](#). This option cannot be used for running tests in the CI/CD pipeline.

searchFolder - Search folder

`string`. Required. Default value: `$(System.DefaultWorkingDirectory)`.

Specifies the folder to search for the test assemblies.

resultsFolder - Test results folder

`string`. Default value: `$(Agent.TempDirectory)\TestResults`.

Specifies the folder to store test results. When using the default directory, it is cleaned at the end of a pipeline run. The results directory will always be cleaned up at the start of the `vstest` task before the tests are run. The relative folder path, if provided, will be considered relative to `$(Agent.TempDirectory)`.

testFiltercriteria - Test filter criteria

`string`. Optional. Use when `testSelector = testAssemblies`.

Specifies additional criteria to filter tests from test assemblies. For example:

`Priority=1|Name=MyTestMethod`. Learn about [command-line options](#).

runOnlyImpactedTests - Run only impacted tests

`boolean`. Optional. Use when `testSelector = testAssemblies`. Default value: `False`.

Automatically specifies and runs the tests needed to validate the code change. Learn about using [Test Impact Analysis](#).

runAllTestsAfterXBuilds - Number of builds after which all tests should be run

`string`. Optional. Use when `testSelector = testAssemblies && runOnlyImpactedTests = true`. Default value: `50`.

Specifies the number of builds to be executed before all tests are automatically run. Test Impact Analysis stores the mapping between test cases and source code. It is recommended to regenerate the mapping by running all tests on a regular basis.

uiTests - Test mix contains UI tests

`boolean`. Default value: `false`.

To run UI tests, ensure that the agent is set to run in [interactive mode](#) with Autologon enabled. Setting up an agent to run interactively must be done before queueing the build/release. Checking this box does not configure the agent in interactive mode automatically. This option serves as a reminder to configure the agent appropriately to avoid failures. Hosted Windows agents from the VS 2015 and 2017 pools can be used to run UI tests.

vstestLocationMethod - Select test platform using

`string`. Allowed values: `version`, `location` (Specific location). Default value: `version`.

Specifies which test platform to use.

vsTestVersion - Test platform version

`string`. Optional. Use when `vstestLocationMethod = version`. Allowed values: `latest`, `17.0` (Visual Studio 2022), `16.0` (Visual Studio 2019), `15.0` (Visual Studio 2017), `14.0` (Visual Studio 2015), `toolsInstaller` (Installed by Tools Installer). Default value: `latest`.

Specifies the version of Visual Studio Test to use. If **latest** is specified, this input chooses Visual Studio 2017 or Visual Studio 2015 depending on what is installed. Visual Studio 2013 is not supported. To run tests without needing Visual Studio on the agent, use the **Installed by tools installer** option. Be sure to include the **Visual Studio Test Platform Installer** task to acquire the test platform from NuGet.

vstestLocation - Path to vstest.console.exe

`string`. Optional. Use when `vstestLocationMethod = location`.

Specifies the path to VSTest.

runSettingsFile - Settings file

`string`.

Specifies the path to a `runsettings` or `testsettings` file to use with the tests. For Visual Studio 15.7 and higher, use `runsettings` for all test types. Learn more about [converting a .testsettings file to a .runsettings file](#).

overrideTestRunParameters - Override test run parameters

`string`.

Overrides the parameters defined in the `TestRunParameters` section of a `runsettings` file or the `Properties` section of a `testsettings` file. For example: `-key1 value1 -key2 value2`. *Note:* Properties specified in a `testsettings` file can be accessed via the `TestContext` using Visual Studio 2017 (update 4 or higher).

pathToCustomTestAdapters - Path to custom test adapters

`string`.

Specifies the directory path to custom test adapters. Adapters residing in the same folder as the test assemblies are automatically discovered.

runInParallel - Run tests in parallel on multi-core machines

`boolean`. Default value: `False`.

If set to `true`, tests are run in parallel and leverage available cores of the machine. This will override the `MaxCpuCount` if specified in your `runsettings` file. Learn more about how [tests are run in parallel](#).

runTestsInIsolation - Run tests in isolation

`boolean`. Default value: `False`.

Runs the tests in an isolated process. This likely leads to fewer errors in the `vstest.console.exe` test process, but tests might run slower. This option currently cannot be used when running with the multi-agent job setting.

`codeCoverageEnabled` - Code coverage enabled

`boolean`. Default value: `False`.

Collects code coverage information from the test run.

`otherConsoleOptions` - Other console options

`string`.

[Other console options](#) that can be passed to vstest.console.exe.

These options are not supported and will be ignored when running tests using the **Multi-agent parallel** setting of an agent job, when running tests using the **Test plan** or **Test run** option, or when a custom batching option is selected. The options can be specified using a settings file instead.

`distributionBatchType` - Batch tests

`string`. Allowed values: `basedOnTestCases` (Based on number of tests and agents), `basedOnExecutionTime` (Based on past running time of tests), `basedOnAssembly` (Based on test assemblies). Default value: `basedOnTestCases`.

A batch is a group of tests. A batch of tests runs its tests at the same time, and results are published for the batch. If the job in which the task runs is set to use multiple agents, each agent picks up any available batches of tests to run in parallel. A batch can be run:

based on the number of tests and agents. Simple batching based on the number of tests and agents participating in the test run.

based on the past running time of tests. This batching considers the past running time to create batches of tests where each batch has approximately equal running time.

based on test assemblies. Tests from an assembly are batched together.

`batchingBasedOnAgentsOption` - Batch options

`string`. Optional. Use when `distributionBatchType = basedOnTestCases`. Allowed values: `autoBatchSize` (Automatically determine the batch size), `customBatchSize` (Specify a batch size). Default value: `autoBatchSize`.

Specifies simple batching based on the number of tests and agents participating in the test run. When the batch size is automatically determined, each batch contains `(total`

`number of tests / number of agents`) tests. If a batch size is specified, each batch will contain the specified number of tests.

`customBatchSizeValue` - Number of tests per batch

`string`. Required when `distributionBatchType = basedOnTestCases && batchingBasedOnAgentsOption = customBatchSize`. Default value: `10`.

Specifies the batch size.

`batchingBasedOnExecutionTimeOption` - Batch options

`string`. Optional. Use when `distributionBatchType = basedOnExecutionTime`. Allowed values: `autoBatchSize` (Automatically determine the batch time), `customTimeBatchSize` (Specify running time per batch). Default value: `autoBatchSize`.

This batching considers past running times to create batches of tests where each batch has approximately equal running time. Quick-running tests will be batched together, while longer-running tests may belong to a separate batch. When this option is used with the multi-agent job setting, the total test time is reduced to a minimum.

`customRunTimePerBatchValue` - Running time (sec) per batch

`string`. Required when `distributionBatchType = basedOnExecutionTime && batchingBasedOnExecutionTimeOption = customTimeBatchSize`. Default value: `60`.

Specifies the running time (in seconds) per batch.

`dontDistribute` - Replicate tests instead of distributing when multiple agents are used in the job

`boolean`. Default value: `False`.

Choosing this option will not distribute tests across agents when the task is running in a multi-agent job. Each of the selected test(s) will be repeated on each agent. This option is not applicable when the agent job is configured to run with no parallelism or with the multi-config option.

`testRunTitle` - Test run title

`string`.

Specifies a name for the test run.

platform - Build platform

`string`.

Specifies the build platform against which the tests should be reported. If you have defined a variable for the platform in your build task, use that with this input.

configuration - Build configuration

`string`.

Specifies the build configuration against which the tests should be reported. If you have defined a variable for configuration in your build task, use that with this input.

publishRunAttachments - Upload test attachments

`boolean`. Default value: `true`.

Opts in or out of publishing run level attachments.

failOnMinTestsNotRun - Fail the task if a minimum number of tests are not run.

`boolean`. Default value: `False`.

Fails the task if a minimum number of tests are not run. This may be useful if any changes to task inputs or underlying test adapter dependencies lead to only a subset of the desired tests to be found.

minimumExpectedTests - Minimum # of tests

`string`. Optional. Use when `failOnMinTestsNotRun = true`. Default value: `1`.

Specifies the minimum number of tests to run for the task to succeed. The total tests executed is calculated as the sum of passed, failed and aborted tests.

diagnosticsEnabled - Collect advanced diagnostics in case of catastrophic failures

`boolean`. Default value: `false`.

Collects diagnostic data to troubleshoot catastrophic failures, such as a test crash. When this option is checked, a sequence XML file is generated and attached to the test run. The sequence file contains information about the sequence in which the tests had run, so a potential culprit test can be identified.

collectDumpOn - Collect process dump and attach to test run report
`string`. Optional. Use when `diagnosticsEnabled = true`. Allowed values: `onAbortOnly` (On abort only), `always`, `never`. Default value: `onAbortOnly`.

Collects a mini dump that can be used for further analysis.

- **onAbortOnly** - a mini dump will be collected only when the test run is aborted.
- **Always** - a mini dump will always be collected regardless of whether the test run completes or not.
- **Never** - a mini dump will not be collected regardless of whether the test run completes or not.

rerunFailedTests - Rerun failed tests

`boolean`. Default value: `False`.

Reruns any failed tests until they pass or until the maximum number of attempts is reached.

rerunType - Do not rerun if test failures exceed specified threshold

`string`. Optional. Use when `rerunFailedTests = true`. Allowed values: `basedOnTestFailurePercentage` (% failure), `basedOnTestFailureCount` (# of failed tests).

Default value: `basedOnTestFailurePercentage`.

Avoids rerunning tests when the failure rate crosses the specified threshold. This is applicable if environment issues lead to massive failures. You can specify the percentage of failures or number of failed tests as a threshold.

rerunFailedThreshold - % failure

`string`. Optional. Use when `rerunFailedTests = true && rerunType = basedOnTestFailurePercentage`. Default value: `30`.

Avoids rerunning tests when the percentage of failed test cases crosses the specified threshold. This is applicable if environment issues lead to massive failures.

rerunFailedTestCasesMaxLimit - # of failed tests

`string`. Optional. Use when `rerunFailedTests = true && rerunType =`

`basedOnTestFailureCount`. Default value: 5.

Avoids rerunning tests when the number of failed test cases crosses the specified limit. This is applicable if environment issues lead to massive failures.

`rerunMaxAttempts` - Maximum # of attempts

`string`. Optional. Use when `rerunFailedTests = true`. Default value: 3.

Specifies the maximum number of times a failed test should be retried. If a test passes before the maximum number of attempts is reached, it will not be rerun again.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to run unit and functional tests (Selenium, Appium, Coded UI test, and more) using the Visual Studio Test runner. Along with MSTest-based tests, test frameworks that have a Visual Studio test adapter can also be executed, such as xUnit, NUnit, or Chutzpah.

Tests that the target .NET core framework can be executed by specifying the appropriate target framework value in the [.runsettings file](#).

Tests can be distributed on multiple agents using version 2 of this task. For more information, see [Run tests in parallel using the Visual Studio Test task](#).

Check prerequisites

If you're using a Windows self-hosted agent, this prerequisite must be installed:

- [.NET Framework](#) 4.6.2 or a later version

Demands

The agent must have the following capability:

vstest

The vstest demand can be satisfied in two ways:

1. Visual Studio is installed on the agent machine.
2. By using the [Visual Studio Test Platform Installer task](#) in the pipeline definition.

How can I run tests that use TestCase as a data source?

To run automated tests that use TestCase as a data source, the following is needed:

1. You must have Visual Studio 2017.6 or higher on the agent machine. Visual Studio Test Platform Installer task cannot be used to run tests that use TestCase as a data source.
2. Create a [PAT](#) that is authorized for the scope "Work Items (full)".
3. Add a secure build or release variable called `Test.TestCaseAccessToken` with the value set to the PAT created in the previous step.

I am running into issues when running data-driven xUnit and NUnit tests with some of the task options. Are there known limitations?

Data-driven tests that use xUnit and NUnit test frameworks have some known limitations and cannot be used with the following task options:

1. Rerun failed tests.
2. Distributing tests on multiple agents and batching options.
3. Test Impact Analysis.

The above limitations are because of how the adapters for these test frameworks discover and report data-driven tests.

Does the VSTest task support running tests that target multiple target frameworks at a time?

Yes, starting from version 17.3 VSTest does support running tests that target multiple target frameworks at a time. Prior to that, this wasn't possible due to a limitation from the [VSTest platform](#) side.

If you want to run tests that belong to multiple target frameworks, you'll need to install a compatible version of VSTest via **Visual Studio Test Platform Installer** and set `vsTestVersion` to `toolsInstaller` to use it.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: vstest
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.103.0 or greater
Task category	Test

VSTest@1 - Visual Studio Test v1 task

Article • 09/26/2023

Use this task to run tests with Visual Studio test runner.

Syntax

YAML

```
# Visual Studio Test v1
# Run tests with Visual Studio test runner.
- task: VSTest@1
  inputs:
    # Execution Options
    testAssembly: '**\*test*.dll;:-:**\obj\**' # string. Required. Test Assembly. Default: **\*test*.dll;:-:**\obj\**.
    #testFiltercriteria: # string. Test Filter criteria.
    #runSettingsFile: # string. Run Settings File.
    #overrideTestrunParameters: # string. Override TestRun Parameters.
    #codeCoverageEnabled: False # boolean. Code Coverage Enabled. Default: False.
    #runInParallel: false # boolean. Run In Parallel. Default: false.
    # Advanced Execution Options
    #vstestLocationMethod: 'version' # 'version' | 'location'. VSTest.
    Default: version.
    #vsTestVersion: '14.0' # 'latest' | '14.0' | '12.0'. Optional. Use when vstestLocationMethod = version. VSTest version. Default: 14.0.
    #vstestLocation: # string. Optional. Use when vstestLocationMethod = location. Path to vstest.console.exe.
    #pathToCustomTestAdapters: # string. Path to Custom Test Adapters.
    #otherConsoleOptions: # string. Other console options.
    # Reporting Options
    #testRunTitle: # string. Test Run Title.
    #platform: # string. Platform.
    #configuration: # string. Configuration.
    #publishRunAttachments: true # boolean. Upload Test Attachments.
    Default: true.
```

Inputs

testAssembly - Test Assembly

string. Required. Default value: ***test*.dll;:-:**\obj**.

Specifies which test binaries to run tests on. Wildcards can be used. For example, using `***test*.dll;:-:**\obj**` for all DLLs with "test" in the name and excluding files in any subdirectory named "obj".

`testFilterCriteria` - Test Filter criteria

`string`.

Specifies additional criteria to filter tests from test assemblies. For example:

`Priority=1|Name=MyTestMethod`.

`runSettingsFile` - Run Settings File

`string`.

Specifies the path to the `runsettings` file to use with the tests. Use

`$(Build.SourcesDirectory)` to access the Project folder.

`overrideTestRunParameters` - Override TestRun Parameters

`string`.

Override parameters defined in the `TestRunParameters` section of the `runsettings` file.

For example: `AppURL=$(DeployURL);Port=8080`.

`codeCoverageEnabled` - Code Coverage Enabled

`boolean`. Default value: `False`.

Collects code coverage information from the test run.

`runInParallel` - Run In Parallel

`boolean`. Default value: `false`.

Enables a parallel execution of your tests.

`vstestLocationMethod` - VSTest

`string`. Allowed values: `version`, `location` (Specify Location). Default value: `version`.

`vsTestVersion` - VSTest version

`string`. Optional. Use when `vstestLocationMethod = version`. Allowed values: `latest`, `14.0` (Visual Studio 2015), `12.0` (Visual Studio 2013). Default value: `14.0`.

Specifies the version of Visual Studio Test to use.

vstestLocation - Path to vstest.console.exe

`string`. Optional. Use when `vstestLocationMethod = location`.

Specifies the path to VSTest.

pathToCustomTestAdapters - Path to Custom Test Adapters

`string`.

Specifies the directory path to the custom test adapters. NuGet restored adapters are automatically searched for.

otherConsoleOptions - Other console options

`string`.

Specifies other Console options that can be passed to `vstest.console.exe`.

testRunTitle - Test Run Title

`string`.

Specifies a name for the test run.

platform - Platform

`string`.

Specifies the platform against which the tests should be reported. If you have defined a variable for the platform in your build task, use that when providing this input.

configuration - Configuration

`string`.

Specifies the configuration against which the tests should be reported. If you have defined a variable for configuration in your build task, use that when providing this input.

publishRunAttachments - Upload Test Attachments

`boolean`. Default value: `true`.

Opts in or out of publishing test run level attachments.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: vstest
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.89.0 or greater
Task category	Test

DeployVisualStudioTestAgent@2 - Visual Studio test agent deployment v2 task

Article • 09/26/2023

DeployVisualStudioTestAgent@2 is deprecated. Use the Visual Studio Test task to run unit and functional tests.

Syntax

YAML

```
# Visual Studio test agent deployment v2
# DeployVisualStudioTestAgent@2 is deprecated. Use the Visual Studio Test
task to run unit and functional tests.
- task: DeployVisualStudioTestAgent@2
  inputs:
    # Test Machines
    testMachines: # string. Required. Machines.
    adminUserName: # string. Required. Admin login.
    adminPassword: # string. Required. Admin password.
    winRmProtocol: 'Http' # 'Http' | 'Https'. Required. Protocol. Default:
Http.
    #testCertificate: true # boolean. Optional. Use when winRmProtocol =
Https. Test Certificate. Default: true.
    # Agent Configuration
    machineUserName: # string. Required. Username.
    machinePassword: # string. Required. Password.
    #runAsProcess: false # boolean. Run UI tests. Default: false.
    #isDataCollectionOnly: false # boolean. Enable data collection only.
Default: false.
    # Advanced
    #testPlatform: '14.0' # '15.0' | '14.0'. Test agent version. Default:
14.0.
    #agentLocation: # string. Test agent location.
    #updateTestAgent: false # boolean. Update test agent. Default: false.
```

Inputs

testMachines - Machines

`string`. Required.

This input has three options:

- Provides a comma separated list of machine IP addresses or FQDNs along with ports. The default port is based on the selected protocol. For example, dbserver.fabrikam.com,dbserver_int.fabrikam.com:5986,192.168.12.34:5986.
- Provides the output variable of other tasks. For example, \$(variableName).
- Provides a machine group name. If you are using HTTPS, the name/IP of the machine should match the CN on the certificate.

adminUserName - Admin login

`string`. Required.

Specifies the administrator login for the target machines.

adminPassword - Admin password

`string`. Required.

Specifies the administrator password for the target machines. This input can accept a variable defined in build/release definitions as `$(passwordVariable)`. You may mark the variable type as `secret` to secure it.

winRmProtocol - Protocol

`string`. Required. Allowed values: `Http`, `Https`. Default value: `Http`.

Specifies the protocol to use for the WinRM connection with the machine(s). The default value is `HTTPS`.

testCertificate - Test Certificate

`boolean`. Optional. Use when `winRmProtocol = Https`. Default value: `true`.

Provides the option to skip the authenticity validation of the machine's certificate by a trusted certification authority. The parameter is required for the WinRM HTTPS protocol.

machineUserName - Username

`string`. Required.

Specifies the username with which test agent needs to run.

machinePassword - Password

`string`. Required.

Specifies the password for the username given above.

runAsProcess - Run UI tests

`boolean`. Default value: `false`.

Denotes if the test agent needs to run as an interactive process. This input is needed for Coded UI Tests.

isDataCollectionOnly - Enable data collection only

`boolean`. Default value: `false`.

Optional. Specifies if the test agent is used only for data collection and not for running tests. This can typically be found on the application under the test (AUT) machine group.

testPlatform - Test agent version

`string`. Allowed values: `15.0` (Visual Studio 2017), `14.0` (Visual Studio 2015). Default value: `14.0`.

Specifies the version of Visual Studio test agent. Chooses an appropriate version to match the VS version using the test binaries that were built.

agentLocation - Test agent location

`string`.

Optional. Supplies the path to vstf_testagent.exe from the network or local location. If no path is provided, it will be automatically downloaded from the [download center](#).

[Install the Test Agent 2015 Update 3](#).

[Install Test Agent 2017](#).

updateTestAgent - Update test agent

`boolean`. Default value: `false`.

If the Test Agent is already deployed on a machine, this option checks to see if an update is available for that version.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

What's new in this task version:

- Support for Visual Studio Test Agent 2017: You can now deploy and run tests using multiple versions of Visual Studio Test Agent. Versions 2015 and 2017 are supported.
- Machine groups created from the test hub are no longer supported. You can continue to use a list of machines or Azure resource groups.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	2.0.0 or greater
Task category	Test

DeployVisualStudioTestAgent@1 - Visual Studio Test Agent Deployment v1 task

Article • 09/26/2023

This task deploys and configures the Test Agent to run tests on a set of machines.

Syntax

YAML

```
# Visual Studio Test Agent Deployment v1
# Deploy and configure Test Agent to run tests on a set of machines.
- task: DeployVisualStudioTestAgent@1
  inputs:
    # Test Machine Group
    testMachineGroup: # string. Required. Machines.
    #adminUserName: # string. Admin Login.
    #adminPassword: # string. Admin Password.
    #winRmProtocol: # 'Http' | 'Https'. Protocol.
    #testCertificate: true # boolean. Optional. Use when winRmProtocol =
    Htts. Test Certificate. Default: true.
    #resourceFilteringMethod: 'machineNames' # 'machineNames' | 'tags'.
    Select Machines By. Default: machineNames.
    #testMachines: # string. Filter Criteria.
    # Agent Configuration
    machineUserName: # string. Required. Username.
    machinePassword: # string. Required. Password.
    #runAsProcess: false # boolean. Interactive Process. Default: false.
    # Advanced
    #agentLocation: # string. Test Agent Location.
    #updateTestAgent: true # boolean. Update Test Agent. Default: true.
    #isDataCollectionOnly: false # boolean. Enable Data Collection Only.
    Default: false.
```

Inputs

`testMachineGroup` - Machines

`string`. Required.

This input has three options:

- Provides a comma separated list of machine IP addresses or FQDNs along with ports. The default port is based on the selected protocol. For example,
`dbserver.fabrikam.com,dbserver_int.fabrikam.com:5986,192.168.12.34:5986`.

- Provides the output variable of other tasks. For example, `$(variableName)`.
- Provides a machine group name. If you are using HTTPS, the name/IP of the machine should match the CN on the certificate.

adminUserName - Admin Login

`string`.

Specifies the administrator login for the target machines.

adminPassword - Admin Password

`string`.

Specifies the administrator password for the target machines. This input can accept a variable defined in build/release definitions as `$(passwordVariable)`. You may mark the variable type as `secret` to secure it.

winRmProtocol - Protocol

`string`. Allowed values: `Http`, `Https`.

Specifies the protocol to use for the WinRM connection with the machine(s). The default value is `HTTPS`.

testCertificate - Test Certificate

`boolean`. Optional. Use when `winRmProtocol = Https`. Default value: `true`.

Provides the option to skip the authenticity validation of the machine's certificate by a trusted certification authority. The parameter is required for the WinRM HTTPS protocol.

resourceFilteringMethod - Select Machines By

`string`. Allowed values: `machineNames` (Machine Names), `tags`. Default value: `machineNames`.

testMachines - Filter Criteria

`string`.

Provides a list of machines like `dbserver.fabrikam.com`, `dbserver_int.fabrikam.com`, `192.168.12.34` or tags like `Role:DB;OS:Win8.1`. Returns machines that have either of the

tags. For Azure Resource Group, provide the VM host name for the machine name. The default deploys an agent on all machines represented in the Machines field.

machineUserName - Username

`string`. Required.

Specifies the username with which the test agent needs to run.

machinePassword - Password

`string`. Required.

Specifies the password for the username given above.

runAsProcess - Interactive Process

`boolean`. Default value: `false`.

Denotes if the test agent needs to run as an interactive process. This input is needed for Coded UI Tests.

agentLocation - Test Agent Location

`string`.

Optional. Supplies the path to vstf_testagent.exe from the network or local location. If no path is provided, it will be downloaded from [the download center](#).

updateTestAgent - Update Test Agent

`boolean`. Default value: `true`.

Optional. Specifies if the test agent needs to be updated.

isDataCollectionOnly - Enable Data Collection Only

`boolean`. Default value: `false`.

Optional. Specifies if the test agent is used only for data collection and not for running tests. This can typically be found on the application under the test (AUT) machine group.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.104.0 or greater
Task category	Test

VisualStudioTestPlatformInstaller@1 - Visual Studio test platform installer v1 task

Article • 09/26/2023

Use this task to acquire the [Microsoft test platform](#) from nuget.org or a specified feed, and add it to the tools cache. The installer task satisfies the `vstest` demand, and a subsequent Visual Studio Test task in a build or release pipeline can run without needing a full Visual Studio install on the agent machine.

Syntax

YAML

```
# Visual Studio test platform installer v1
# Acquire the test platform from nuget.org or the tool cache. Satisfies the
# 'vstest' demand and can be used for running tests and collecting diagnostic
# data using the Visual Studio Test task.
- task: VisualStudioTestPlatformInstaller@1
  inputs:
    # Package settings
    packageFeedSelector: 'nugetOrg' # 'nugetOrg' | 'customFeed' |
    'netShare'. Required. Package Feed. Default: nugetOrg.
    #versionSelector: 'latestPreRelease' # 'latestPreRelease' |
    'latestStable' | 'specificVersion'. Required when packageFeedSelector =
    nugetOrg || packageFeedSelector = customFeed. Version. Default:
    latestPreRelease.
    #testPlatformVersion: # string. Required when versionSelector =
    specificVersion. Test Platform Version.
    #customFeed: # string. Required when packageFeedSelector = customFeed.
    Package Source.
    #username: # string. Optional. Use when packageFeedSelector =
    customFeed. User Name.
    #password: # string. Optional. Use when packageFeedSelector =
    customFeed. Password.
    #netShare: # string. Required when packageFeedSelector = netShare. UNC
    Path.
```

Inputs

`packageFeedSelector` - Package Feed

`string`. Required. Allowed values: `nugetOrg` (Official Nuget), `customFeed` (Custom Feed),

`netShare` (Network path). Default value: `nugetOrg`.

Specifies the feed where the task fetches the Visual Studio Test Platform NuGet package.

`nugetOrg` - **Official NuGet**: Acquires the [test platform package from NuGet](#). This option requires internet connectivity on the agent machine.

`customFeed` - **Custom feed**: Acquires the test platform package from a custom feed or a package management feed in Azure DevOps or TFS.

`netShare` - **Network path**: Installs the test platform from a network share. The specified `Microsoft.TestPlatform.nupkg` version must be downloaded from NuGet and placed on a network share that the build/release agent can access.

`versionSelector` - Version

`string`. Required when `packageFeedSelector = nugetOrg || packageFeedSelector = customFeed`. Allowed values: `latestPreRelease` (Latest (Includes Pre-Release)), `latestStable` (Latest Stable), `specificVersion` (Specific Version). Default value: `latestPreRelease`.

Installs the latest version or a specific version of the Visual Studio Test Platform. If you use the test platform installer to run Coded UI tests, the chosen Visual Studio Test Platform must match the major version of the Visual Studio installation that built the test binaries. For example, if the Coded UI test project was built using Visual Studio 2017 (version 15.x), you must use Test Platform version 15.x.

`testPlatformVersion` - Test Platform Version

`string`. Required when `versionSelector = specificVersion`.

Specifies the version of Visual Studio Test Platform to install on the agent. Available versions can be viewed on [NuGet](#).

`customFeed` - Package Source

`string`. Required when `packageFeedSelector = customFeed`.

Specifies the URL of a custom feed or a package management feed in Azure DevOps or TFS that contains the test platform package. Public and private feeds can be specified.

username - User Name

`string`. Optional. Use when `packageFeedSelector = customFeed`.

Specifies the user name to authenticate the feed specified in the **Package Source** argument. This input is not required if the `password` input uses a personal access token (PAT).

password - Password

`string`. Optional. Use when `packageFeedSelector = customFeed`.

Specifies the password or personal access token (PAT) for authenticating the feed specified in the `customFeed` input.

netShare - UNC Path

`string`. Required when `packageFeedSelector = netShare`.

Specifies the full UNC path to the `Microsoft.TestPlatform.nupkg` file. The specified `Microsoft.TestPlatform.nupkg` version must be downloaded from [NuGet](#) and placed on a network share that the build/release agent can access.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Note

If you are using a hosted agent, check the **software table** for the agent you are using to see if Visual Studio is installed. If Visual Studio is installed, you don't need to run the Visual Studio test platform installer task.

Use this task to acquire the [Microsoft test platform](#) from nuget.org or a specified feed, and add it to the tools cache. The installer task satisfies the `vstest` demand, and a subsequent [Visual Studio Test task](#) in a build or release pipeline can run without needing a full Visual Studio install on the agent machine.

 **Note**

- The **Visual Studio Test Platform Installer** task must appear before the **Visual Studio Test** task in the build or release pipeline.
- The **Test platform version** option in the **Visual Studio Test** task must be set to **Installed by Tools Installer**.

See [Run automated tests from test plans](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	Running this task satisfies the following demands for any subsequent tasks in the same job: VsTest
Command restrictions	Any
Settable variables	Any
Agent version	2.144.0 or greater
Task category	Tool

WindowsMachineFileCopy@2 - Windows machine file copy v2 task

Article • 09/26/2023

Use this task to copy files to remote Windows machines.

Syntax

YAML

```
# Windows machine file copy v2
# Copy files to remote Windows machines.
- task: WindowsMachineFileCopy@2
  inputs:
    SourcePath: # string. Required. Source.
    MachineNames: # string. Required. Machines.
    AdminUserName: # string. Required. Admin Login.
    AdminPassword: # string. Required. Password.
    TargetPath: # string. Required. Destination Folder.
  # Advanced Options
  #CleanTargetBeforeCopy: false # boolean. Clean Target. Default: false.
  #CopyFilesInParallel: true # boolean. Copy Files in Parallel. Default:
  true.
  #AdditionalArguments: # string. Additional Arguments.
```

Inputs

SourcePath - Source

string. Required.

The path to the files to copy. Specifies the absolute path of the source folder or file on the local machine or a UNC Share, like `c:\fabrikamfiber` or `\\\fabrikamshare\fabrikamfiber`. You can use predefined system variables, such as `$(Build.Repository.LocalPath)` (the working folder on the agent computer), which makes it easy to specify the location of the build artifacts on the computer that hosts the automation agent.

MachineNames - Machines

string. Required.

Specifies a comma-separated list of machine IP addresses or FQDNs, optionally including the port number.

For example: `dbserver.fabrikam.com`, `dbserver_int.fabrikam.com:5986`, `192.168.12.34`

You can also specify the output variable of other tasks, for example `$(variableName)`, or you can use the name of an [Azure Resource Group](#).

AdminUserName - Admin Login

`string`. Required.

Specifies the username of a domain or a local administrative account on the target host(s). Formats such as `domain\username`, `username`, and `machine-name\username` are supported. UPN formats, such as `username@domain.com`, and built-in system accounts, such as `NT Authority\System`, are not supported.

AdminPassword - Password

`string`. Required.

Specifies the password for the administrator login for the target machines. Variables defined in build or release pipelines, such as `$(passwordVariable)`, are accepted. You can mark the variable as `secret` to secure it.

TargetPath - Destination Folder

`string`. Required.

Specifies the local path on the target machines or an accessible UNC path for copying the files from the source, like `d:\fabrikam` or `\fabrikam\Web`.

CleanTargetBeforeCopy - Clean Target

`boolean`. Default value: `false`.

Deletes all files in the target folder before copying the new files to it.

CopyFilesInParallel - Copy Files in Parallel

`boolean`. Default value: `true`.

Copies files to all target machines in parallel, which can speed up the copying process.

`AdditionalArguments` - Additional Arguments

`string`.

Specifies additional RoboCopy arguments that are applied when copying files, like

`/min:33553332 /1.`

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to copy application files and other artifacts, such as PowerShell scripts and PowerShell-DSC modules, which are required to install the application on Windows machines. It uses RoboCopy, the command-line utility built for fast copying of data.

Why do I get a system error 53 when using this task?

Typically this occurs when the specified path cannot be located. This may be due to a firewall blocking the necessary ports for file and printer sharing or an invalid path specification. For more details, see [Error 53](#) on TechNet.

What's new in Version 2.0

- Proxy support is being added.
- Removed support of legacy DTL machines.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup

Requirement	Description
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.104.0 or greater
Task category	Deploy

WindowsMachineFileCopy@1 - Windows machine file copy v1 task

Article • 09/26/2023

Use this task to copy files to remote Windows machines.

Syntax

YAML

```
# Windows machine file copy v1
# Copy files to remote Windows machines.
- task: WindowsMachineFileCopy@1
  inputs:
    SourcePath: # string. Required. Source.
    #EnvironmentName: # string. Machines.
    #AdminUserName: # string. Admin Login.
    #AdminPassword: # string. Password.
    TargetPath: # string. Required. Destination Folder.
    # Advanced Options
    #CleanTargetBeforeCopy: false # boolean. Clean Target. Default: false.
    #CopyFilesInParallel: true # boolean. Copy Files in Parallel. Default:
    true.
    #AdditionalArguments: # string. Additional Arguments.
    #ResourceFilteringMethod: 'machineNames' # 'machineNames' | 'tags'.
    Select Machines By. Default: machineNames.
    #MachineNames: # string. Filter Criteria.
```

Inputs

SourcePath - Source

`string`. Required.

Specifies the absolute path of the source folder or file on the local machine or a UNC Share, like `c:\fabrikamfiber` or `\fabrikamshare\fabrikamfiber`.

EnvironmentName - Machines

`string`.

Specifies a comma-separated list of machine IP addresses or FQDNs, for example, `dbserver.fabrikam.com,192.168.12.34`. You can also specify the output variable of other

tasks, for example `$(variableName)`.

AdminUserName - Admin Login

`string`.

Specifies the administrator login for the target machines.

AdminPassword - Password

`string`.

Specifies the password for the administrator login for the target machines. Variables defined in build/release definitions as `$(passwordVariable)` are accepted. You can mark the variable type as `secret` to secure it.

TargetPath - Destination Folder

`string`. Required.

Specifies the local path on the target machine or an accessible UNC path for copying the files from the source, like `d:\fabrikam` or `\fabrikam\Web`.

CleanTargetBeforeCopy - Clean Target

`boolean`. Default value: `false`.

Cleans the destination folder before copying the files.

CopyFilesInParallel - Copy Files in Parallel

`boolean`. Default value: `true`.

Copies files in parallel to the machines.

AdditionalArguments - Additional Arguments

`string`.

Specifies additional robocopy arguments that are applied when copying files, like `/min:33553332 /1`.

ResourceFilteringMethod - Select Machines By

`string`. Allowed values: `machineNames` (Machine Names), `tags`. Default value: `machineNames`.

MachineNames - Filter Criteria

`string`.

This input is only valid for machine groups and is not supported for a flat list of machines or output variables yet.

Specifies a comma-separated list of machines, like `dbserver.fabrikam.com`, `webserver.fabrikam.com`, `192.168.12.34`, or tags, like `Role:DB; OS:Win8.1`. If multiple tags are provided, the task will run in all machines with the specified tags. The default runs the task in all machines.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.104.0 or greater

Requirement	Description
Task category	Deploy

XamarinComponentRestore@0 - Xamarin Component Restore v0 task

Article • 09/26/2023

This task is deprecated. Use 'NuGet' instead.

Syntax

YAML

```
# Xamarin Component Restore v0
# This task is deprecated. Use 'NuGet' instead.
- task: XamarinComponentRestore@0
  inputs:
    solutionFile: '**/*.sln' # string. Alias: solution. Required. Path to
    solution. Default: **/*.sln.
    email: # string. Required. Email.
    password: # string. Required. Password.
```

Inputs

solutionFile - Path to solution

Input alias: `solution`. `string`. Required. Default value: `**/*.sln`.

Specifies the path to the Visual Studio solution file.

email - Email

`string`. Required.

Specifies the Xamarin account email address.

password - Password

`string`. Required.

Specifies the Xamarin account password. Use a new build variable with its lock enabled on the Variables tab to encrypt this value.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.96.0 or greater
Task category	Package

XamarinLicense@1 - Xamarin License v1 task

Article • 09/26/2023

XamarinLicense@1 is deprecated because you no longer need a Xamarin license to build your Xamarin app. You can now use the free version of [Xamarin](#).

This task was originally used in a build or release pipeline to activate or deactivate Xamarin licenses.

Syntax

YAML

```
# Xamarin License v1
# [Deprecated] Upgrade to free version of Xamarin:
https://store.xamarin.com.
- task: XamarinLicense@1
  inputs:
    action: 'Activate' # 'Activate' | 'Deactivate'. Required. Action.
  Default: Activate.
    email: # string. Required. Email.
    password: # string. Required. Password.
    product: 'MA' # 'MA' | 'MT' | 'MM'. Required. Xamarin Product. Default:
  MA.
  # Advanced
  #timeout: '30' # string. Timeout in Seconds. Default: 30.
```

Inputs

`action` - Action

`string`. Required. Allowed values: `Activate`, `Deactivate`. Default value: `Activate`.

Specifies `activate` for the first instance of this build task before any instances of the `Xamarin.Android` or `Xamarin.iOS` tasks. Specifies `deactivate` for the second instance of this build task after all instances of the `Xamarin.Android` and `Xamarin.iOS` tasks. You should also select `Always run` under `Control options` for the last instance of the Xamarin license task.

`email` - Email

`string`. Required.

Specifies the Xamarin account email address.

`password` - Password

`string`. Required.

Specifies the Xamarin account password. Use a [secret variable](#) with its lock enabled on the variables tab to encrypt this value.

`product` - Xamarin Product

`string`. Required. Allowed values: `MA` (Xamarin.Android), `MT` (Xamarin.iOS), `MM` (Xamarin.Mac). Default value: `MA`.

Specifies the Xamarin product name.

`timeout` - Timeout in Seconds

`string`. Default value: `30`.

Specifies how long you want to allow the build task to wait for the activation or deactivation.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

This task is deprecated because you no longer need a Xamarin license to [build your Xamarin app](#). Use the free version of Xamarin from <https://store.xamarin.com>.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.83.0 or greater
Task category	Utility

See also

- [build your Xamarin app](#)

XamarinTestCloud@1 - Xamarin Test Cloud v1 task

Article • 09/26/2023

XamarinTestCloud@1 is deprecated. Originally, this task was used in a build or release pipeline to test mobile apps with Xamarin Test Cloud using Xamarin.UITest.

ⓘ Note

You can now [sign up with App Center](#) and use the [AppCenterTest](#) task instead.

Syntax

YAML

```
# Xamarin Test Cloud v1
# [Deprecated] Test mobile apps with Xamarin Test Cloud using
Xamarin.UITest. Instead, use the 'App Center test' task.
- task: XamarinTestCloud@1
  inputs:
    appFile: # string. Alias: app. Required. App file.
    #dsymFile: # string. Alias: dsym. dSYM file (iOS only).
    teamApiKey: # string. Required. Team API key.
    email: # string. Alias: user. Required. User email.
    devices: # string. Required. Devices.
    series: 'master' # string. Required. Series. Default: master.
    testAssemblyDirectory: # string. Alias: testDir. Required. Test assembly
directory.
    # Advanced
    parallelizationOption: 'none' # 'none' | '--fixture-chunk' | '--test-
chunk'. Alias: parallelization. Required. Parallelization. Default: none.
    localeOption: 'en_US' # 'da_DK' | 'nl_NL' | 'en_GB' | 'en_US' | 'fr_FR'
| 'de_DE' | 'ja_JP' | 'ru_RU' | 'es_MX' | 'es_ES' | 'user'. Alias: locale.
Required. System language. Default: en_US.
    #userDefinedLocale: # string. Optional. Use when locale = user. Other
locale.
    testCloudFile: '**/packages/**/tools/test-cloud.exe' # string. Alias:
testCloudLocation. Required. test-cloud.exe location. Default:
**/packages/**/tools/test-cloud.exe.
    #optionalArgs: # string. Optional arguments.
    #publishNUnitResults: true # boolean. Publish results to Azure
Pipelines. Default: true.
```

Inputs

`appFile` - App file

Input alias: `app`. `string`. Required.

Specifies the relative path from repo root of the app(s) to test. Wildcards can be used. For example, `**/*.apk` for all APK files in all subfolders. Learn more about [file matching patterns](#).

`dsymFile` - dSYM file (iOS only)

Input alias: `dsym`. `string`.

Provides a path relative to the `.ipa` file. To make crash logs easier to read, you can upload a dSYM file that is associated with your app. This field only applies to iOS apps. Wildcards can be used. For example: `*.dsym`. Learn more about [file matching patterns](#).

`teamApiKey` - Team API key

`string`. Required.

Specifies your Xamarin Test Cloud Team API key, which can be found under [Teams & Apps](#). Use a [secret variable](#) to avoid exposing this value.

`email` - User email

Input alias: `user`. `string`. Required.

Specifies the email address of your [Xamarin Test Cloud account](#).

`devices` - Devices

`string`. Required.

Specifies the devices string generated by Xamarin Test Cloud. The string can be found as the value of the `--devices` command line argument of a Test Cloud test run.

`series` - Series

`string`. Required. Default value: `master`.

Specifies the series name for organizing test runs (e.g. `master`, `production`, `beta`).

`testAssemblyDirectory` - Test assembly directory

Input alias: `testDir`. `string`. Required.

Specifies the relative path to the folder containing the test assemblies, such as:

`SolutionName/TestsProjectName/bin/Release`.

parallelizationOption - Parallelization

Input alias: `parallelization`. `string`. Required. Allowed values: `none`, `--fixture-chunk` (By test fixture), `--test-chunk` (By test method). Default value: `none`.

Specifies tests to run simultaneously.

localeOption - System language

Input alias: `locale`. `string`. Required. Allowed values: `da_DK` (Danish (Denmark)), `nl_NL` (Dutch (Netherlands)), `en_GB` (English (United Kingdom)), `en_US` (English (United States)), `fr_FR` (French (France)), `de_DE` (German (Germany)), `ja_JP` (Japanese (Japan)), `ru_RU` (Russian (Russia)), `es_MX` (Spanish (Mexico)), `es_ES` (Spanish (Spain)), `user` (Other).

Default value: `en_US`.

Specifies your language. If your language isn't displayed, select `Other` and enter its locale below, such as `en_US`.

userDefinedLocale - Other locale

`string`. Optional. Use when `locale = user`.

Enters any two-letter ISO-639 language code along with any two-letter ISO 3166 country code in the format [language]_[country], such as `en_US`.

testCloudFile - test-cloud.exe location

Input alias: `testCloudLocation`. `string`. Required. Default value:

`**/packages/**/tools/test-cloud.exe`.

Specifies the path to `test-cloud.exe`. Wildcards can be used, and when they are, the first occurrence of `test-cloud.exe` is used. Learn more about [file matching patterns](#).

optionalArgs - Optional arguments

`string`.

Specifies the additional arguments passed to `test-cloud.exe`.

`publishNUnitResults` - Publish results to Azure Pipelines

`boolean`. Default value: `true`.

Specifies the `--nunit-xml` option to be passed to `test-cloud.exe` so that results from the NUnit xml file are published to Azure Pipelines.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build, Classic release
Runs on	Agent, DeploymentGroup
Demands	None
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.83.0 or greater
Task category	Test

XamarinAndroid@1 - Xamarin.Android

v1 task

Article • 09/26/2023

Use this task to build an Android app with Xamarin.

Syntax

YAML

```
# Xamarin.Android v1
# Build an Android app with Xamarin.
- task: XamarinAndroid@1
  inputs:
    projectFile: '**/*.csproj' # string. Alias: project. Required. Project.
  Default: **/*.csproj.
    #target: # string. Target.
    #outputDirectory: # string. Alias: outputDir. Output directory.
    #configuration: # string. Configuration.
    #createAppPackage: true # boolean. Create app package. Default: true.
    #clean: false # boolean. Clean. Default: false.
  # MSBuild Options
    #msbuildLocationOption: 'version' # 'version' | 'location'. Alias:
  msbuildLocationMethod. MSBuild. Default: version.
    #msbuildVersionOption: '15.0' # 'latest' | '17.0' | '16.0' | '15.0' |
  '14.0' | '12.0' | '4.0'. Alias: msbuildVersion. Optional. Use when
  msbuildLocationMethod = version. MSBuild version. Default: 15.0.
    #msbuildFile: # string. Alias: msbuildLocation. Required when
  msbuildLocationMethod = location. MSBuild location.
    #msbuildArchitectureOption: 'x86' # 'x86' | 'x64'. Alias:
  msbuildArchitecture. Optional. Use when msbuildLocationMethod = version.
  MSBuild architecture. Default: x86.
    #msbuildArguments: # string. Additional arguments.
  # JDK Options
    jdkOption: 'JDKVersion' # 'JDKVersion' | 'Path'. Alias: jdkSelection.
  Required. Select JDK to use for the build. Default: JDKVersion.
    #jdkVersionOption: 'default' # 'default' | '1.11' | '1.10' | '1.9' |
  '1.8' | '1.7' | '1.6'. Alias: jdkVersion. Optional. Use when jdkSelection =
  JDKVersion. JDK version. Default: default.
    #jdkDirectory: # string. Alias: jdkUserInputPath. Required when
  jdkSelection = Path. JDK path.
    #jdkArchitectureOption: 'x64' # 'x86' | 'x64'. Alias: jdkArchitecture.
  Optional. Use when jdkVersion != default. JDK architecture. Default: x64.
```

Inputs

`projectFile` - Project

Input alias: `project`. `string`. Required. Default value: `**/*.csproj`.

Specifies the relative path from repo root of `Xamarin.Android` project(s) to build.

Wildcards can be used. For more information, see the [File matching patterns reference](#).

For example, `**/*.csproj` for all csproj files in all subfolders. The project must have a

`PackageForAndroid` target if `Create App Package` is selected.

`target` - Target

`string`.

Specifies which targets to build in this project. Use a semicolon to separate multiple targets.

`outputDirectory` - Output directory

Input alias: `outputDir`. `string`.

Optional. Provides the output directory for the build. Example:

`$(build.binariesDirectory)/bin/Release`.

`configuration` - Configuration

`string`.

Specifies the configuration you want to build. For example, `debug` or `release`.

Tip

Declare a build variable such as `BuildConfiguration` on the variables tab (selecting `Allow at Queue Time`) and reference it here as `$(BuildConfiguration)`. You can then modify the platform when you queue the build and enable building multiple configurations.

`createAppPackage` - Create app package

`boolean`. Default value: `true`.

Passes the target, `(/t:PackageForAndroid)`, during the build to generate an APK.

`clean` - Clean

`boolean`. Default value: `false`.

Passes the clean target, `(/t:clean)`, during the build.

`msbuildLocationOption` - MSBuild

Input alias: `msbuildLocationMethod`. `string`. Allowed values: `version`, `location` (Specify Location). Default value: `version`.

Specifies the path to MSBuild (on Windows) or xbuild (on macOS). The default behavior is to search for the latest version.

`msbuildVersionOption` - MSBuild version

Input alias: `msbuildVersion`. `string`. Optional. Use when `msbuildLocationMethod = version`. Allowed values: `latest`, `17.0` (MSBuild 17.0), `16.0` (MSBuild 16.0), `15.0` (MSBuild 15.0), `14.0` (MSBuild 14.0), `12.0` (MSBuild 12.0), `4.0` (MSBuild 4.0). Default value: `15.0`.

Specifies the use of the latest version if the preferred version cannot be found. On macOS, xbuild (Mono) or MSBuild (Visual Studio for Mac) will be used.

`msbuildFile` - MSBuild location

Input alias: `msbuildLocation`. `string`. Required when `msbuildLocationMethod = location`.

Optional. Supplies the path to MSBuild (on Windows) or xbuild (on macOS).

`msbuildArchitectureOption` - MSBuild architecture

Input alias: `msbuildArchitecture`. `string`. Optional. Use when `msbuildLocationMethod = version`. Allowed values: `x86` (MSBuild x86), `x64` (MSBuild x64). Default value: `x86`.

Supplies the architecture (x86, x64) of the MSBuild you want to run.

`msbuildArguments` - Additional arguments

`string`.

Specifies additional arguments passed to MSBuild (on Windows) or xbuild (on macOS).

`jdkOption` - Select JDK to use for the build

Input alias: `jdkSelection`. `string`. Required. Allowed values: `JDKVersion` (JDK Version), `Path`. Default value: `JDKVersion`.

Specifies the JDK version that the task uses during the build process. The `JDKVersion` value specifies a JDK version that the task discovers during builds. The `Path` value specifies a file path for a JDK version.

`jdkVersionOption` - JDK version

Input alias: `jdkVersion`. `string`. Optional. Use when `jdkSelection = JDKVersion`. Allowed values: `default`, `1.11` (JDK 11), `1.10` (JDK 10 (out of support)), `1.9` (JDK 9 (out of support)), `1.8` (JDK 8), `1.7` (JDK 7), `1.6` (JDK 6 (out of support)). Default value: `default`.

Specifies the JDK version to use during the build.

`jdkDirectory` - JDK path

Input alias: `jdkUserInputPath`. `string`. Required when `jdkSelection = Path`.

Specifies the JDK version to use during the build at the `jdkSelection` path.

`jdkArchitectureOption` - JDK architecture

Input alias: `jdkArchitecture`. `string`. Optional. Use when `jdkVersion != default`. Allowed values: `x86`, `x64`. Default value: `x64`.

Supplies the architecture (x86, x64) of JDK.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to build an Android app with Xamarin.

Examples

- [Build your Xamarin app](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: MSBuild, Xamarin.Android
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	1.83.0 or greater
Task category	Build

XamariniOS@2 - Xamarin.iOS v2 task

Article • 09/26/2023

Use this task in a pipeline to build an iOS app with Xamarin on macOS. For more information, see the [Xamarin guidance](#) and [Sign your app during CI](#).

Syntax

YAML

```
# Xamarin.iOS v2
# Build an iOS app with Xamarin on macOS.
- task: XamariniOS@2
  inputs:
    solutionFile: '**/*.sln' # string. Alias: solution. Required. Solution.
    Default: **/*.sln.
    configuration: 'Release' # string. Required. Configuration. Default:
    Release.
    #clean: false # boolean. Clean. Default: false.
    #packageApp: true # boolean. Create app package. Default: true.
    #buildForSimulator: false # boolean. Alias: forSimulator. Build for iOS
    Simulator. Default: false.
    # Advanced
    #runNugetRestore: false # boolean. Run NuGet restore. Default: false.
    #args: # string. Arguments.
    #workingDirectory: # string. Alias: cwd. Working directory.
    #mdtoolFile: # string. Alias: buildToolLocation | mdtoolLocation. Build
    tool path.
    # Signing & Provisioning
    #signingIdentity: # string. Alias: iosSigningIdentity. Signing identity.
    #signingProvisioningProfileID: # string. Alias: provProfileUuid.
    Provisioning profile UUID.
```

Inputs

`solutionFile` - Solution

Input alias: `solution`. `string`. Required. Default value: `**/*.sln`.

Specifies the relative path from the repository root of the `Xamarin.iOS` solution or `csproj` project to the build. May contain wildcards.

`configuration` - Configuration

`string`. Required. Default value: `Release`.

Specifies the configuration. Standard configurations are Ad-Hoc, AppStore, Debug, and Release.

clean - Clean

`boolean`. Default value: `false`.

Optional. Runs a clean build (`/t:clean`) prior to the build.

packageApp - Create app package

`boolean`. Default value: `true`.

If set to `true`, generates an IPA as a part of the build.

buildForSimulator - Build for iOS Simulator

Input alias: `forSimulator`. `boolean`. Default value: `false`.

Optional. Builds for the iOS Simulator instead of physical iOS devices.

runNugetRestore - Run NuGet restore

`boolean`. Default value: `false`.

Runs `nuget restore` on the Xamarin iOS solution to install all referenced packages before the build. The `nuget` tool in the PATH of the build agent machine is used. To use a different version of NuGet or set additional arguments, use the [NuGet Installer Task](#).

args - Arguments

`string`.

Optional. Specifies additional command line arguments that are used to build.

workingDirectory - Working directory

Input alias: `cwd`. `string`.

Optional. Specifies the working directory in which builds will run. If the value is empty, the root of the repository is used.

`mdtoolFile` - Build tool path

Input alias: `buildToolLocation` | `mdtoolLocation`. `string`.

Optional. Supplies the path to xbuild (the Xamarin Studio mono build tool) or MSBuild (the Visual Studio for Mac build tool). If the value is empty, the default xbuild or MSBuild path is used.

`signingIdentity` - Signing identity

Input alias: `iosSigningIdentity`. `string`.

Optional. Overrides the signing identity that will be used to sign the build. If the value is empty, the setting in the Xcode project will be used. You may need to select `signingUnlockDefaultKeychain` if you use this option.

`signingProvisioningProfileID` - Provisioning profile UUID

Input alias: `provProfileUuid`. `string`.

Optional. Specifies the UUID of an installed provisioning profile override to be used for this build.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task in a pipeline to build an iOS app with Xamarin on macOS. For more information, see the [Xamarin guidance](#) and [Sign your app during CI](#).

What's new in this task version

- iOS signing set up has been removed from the task. Use `Secure Files` with supporting tasks `Install Apple Certificate` and `Install Apple Provisioning`

[Profile](#) to setup signing. Updated options to work better with [Visual Studio for Mac](#).

Examples

- Build your Xamarin app

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: Xamarin.iOS
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Build

XamariniOS@1 - Xamarin.iOS v1 task

Article • 09/26/2023

Use this task in a pipeline to build an iOS app with Xamarin on macOS. For more information, see the [Xamarin guidance](#) and [Sign your app during CI](#).

Syntax

YAML

```
# Xamarin.iOS v1
# Build an iOS app with Xamarin on macOS.
- task: XamariniOS@1
  inputs:
    solutionFile: '**/*.sln' # string. Alias: solution. Required. Solution.
    Default: **/*.sln.
    configuration: 'Release' # string. Required. Configuration. Default:
    Release.
    #clean: false # boolean. Clean. Default: false.
    #packageApp: true # boolean. Create app package. Default: true.
    #buildForSimulator: false # boolean. Alias: forSimulator. Build for iOS
    Simulator. Default: false.
    # Advanced
    #runNugetRestore: true # boolean. Run NuGet restore. Default: true.
    #args: # string. Arguments.
    #workingDirectory: # string. Alias: cwd. Working directory.
    #buildToolOption: 'xbuild' # 'xbuild' | 'msbuild'. Alias: buildTool.
    Build tool. Default: xbuild.
    #mdtoolFile: # string. Alias: mdtoolLocation. Build tool path.
    # Signing & Provisioning
    #signingOption: 'file' # 'file' | 'id'. Alias: signMethod. Override
    using. Default: file.
    #signingIdentity: # string. Alias: iosSigningIdentity. Optional. Use
    when signMethod = id. Signing identity.
    #signingUnlockDefaultKeychain: false # boolean. Alias:
    unlockDefaultKeychain. Optional. Use when signMethod = id. Unlock default
    keychain. Default: false.
    #signingDefaultKeychainPassword: # string. Alias:
    defaultKeychainPassword. Optional. Use when signMethod = id. Default
    keychain password.
    #signingProvisioningProfileID: # string. Alias: provProfileUuid.
    Optional. Use when signMethod = id. Provisioning profile UUID.
    #signingP12File: # string. Alias: p12. Optional. Use when signMethod =
    file. P12 certificate file.
    #signingP12Password: # string. Alias: p12pwd. Optional. Use when
    signMethod = file. P12 password.
    #signingProvisioningProfileFile: # string. Alias: provProfile. Optional.
    Use when signMethod = file. Provisioning profile file.
```

```
#signingRemoveProfile: false # boolean. Alias: removeProfile. Optional.  
Use when signMethod = file. Remove profile after build. Default: false.
```

Inputs

`solutionFile` - Solution

Input alias: `solution`. `string`. Required. Default value: `**/*.sln`.

Specifies the relative path from the repository root of the `Xamarin.iOS` solution to the build. May contain wildcards.

`configuration` - Configuration

`string`. Required. Default value: `Release`.

Specifies the configuration. Standard configurations are Ad-Hoc, AppStore, Debug, and Release.

`clean` - Clean

`boolean`. Default value: `false`.

Optional. Runs a clean build (`/t:clean`) prior to the build.

`packageApp` - Create app package

`boolean`. Default value: `true`.

If set to `true`, generates an IPA as a part of the build.

`buildForSimulator` - Build for iOS Simulator

Input alias: `forSimulator`. `boolean`. Default value: `false`.

Optional. Builds for the iOS Simulator instead of physical iOS devices.

`runNugetRestore` - Run NuGet restore

`boolean`. Default value: `true`.

Runs `nuget restore` on the Xamarin iOS solution to install all referenced packages before the build. The `nuget` tool in the PATH of the build agent machine is used. To use a different version of NuGet or set additional arguments, use the [NuGet Installer Task](#).

args - Arguments

`string`.

Optional. Specifies additional command line arguments that are used to the build.

workingDirectory - Working directory

Input alias: `cwd`. `string`.

Optional. Specifies the working directory in which builds will run. If the value is empty, the root of the repository is used.

buildToolOption - Build tool

Input alias: `buildTool`. `string`. Allowed values: `xbuild` (Xamarin Studio), `msbuild` (MSBuild (Visual Studio for Mac)). Default value: `xbuild`.

Specifies the build tools that the task will use.

mdtoolFile - Build tool path

Input alias: `mdtoolLocation`. `string`.

Optional. Supplies the path to xbuild (the Xamarin Studio mono build tool) or MSBuild (the Visual Studio for Mac build tool). If the value is empty, the default xbuild or MSBuild path is used.

signingOption - Override using

Input alias: `signMethod`. `string`. Allowed values: `file` (File Contents), `id` (Identifiers).

Default value: `file`.

Use this input if the build uses a signing or provisioning method that is different than the default. Choose `file` to use a P12 certificate and provisioning profile. Choose `id` to retrieve signing settings from the default Keychain and pre-installed profiles. Leave the corresponding fields blank if you do not wish to override the default build settings.

signingIdentity - Signing identity

Input alias: `iosSigningIdentity`. `string`. Optional. Use when `signMethod = id`.

Overrides the signing identity that will be used to sign the build. If the value is empty, the setting in the Xcode project will be used. You may need to select `signingUnlockDefaultKeychain` if you use this option.

`signingUnlockDefaultKeychain` - Unlock default keychain

Input alias: `unlockDefaultKeychain`. `boolean`. Optional. Use when `signMethod = id`.

Default value: `false`.

Resolves "User interaction is not allowed" errors by unlocking the default keychain.

`signingDefaultKeychainPassword` - Default keychain password

Input alias: `defaultKeychainPassword`. `string`. Optional. Use when `signMethod = id`.

Specifies the password to unlock the default keychain when `signingUnlockDefaultKeychain` is set.

`signingProvisioningProfileID` - Provisioning profile UUID

Input alias: `provProfileUuid`. `string`. Optional. Use when `signMethod = id`.

Specifies the UUID of an installed provisioning profile to be used for this build.

`signingP12File` - P12 certificate file

Input alias: `p12`. `string`. Optional. Use when `signMethod = file`.

Specifies the relative path to a PKCS12-formatted P12 certificate file containing a signing certificate to be used for this build.

`signingP12Password` - P12 password

Input alias: `p12pwd`. `string`. Optional. Use when `signMethod = file`.

Specifies the password to the P12 certificate file. Use a build variable to encrypt this value.

`signingProvisioningProfileFile` - Provisioning profile file

Input alias: `provProfile`. `string`. Optional. Use when `signMethod = file`.

Specifies the UUID of an installed provisioning profile override to be used for this build.

`signingRemoveProfile` - Remove profile after build

Input alias: `removeProfile`. `boolean`. Optional. Use when `signMethod = file`. Default value: `false`.

Specifies that the contents of the provisioning profile file should be removed from the build agent after the build is complete. **Only enable this if you are running one agent per user.**

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: Xamarin.iOS
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Build

Xcode@5 - Xcode v5 task

Article • 09/26/2023

Use this task to build, test, or archive an Xcode workspace on macOS, and optionally package an app.

Syntax

YAML

```
# Xcode v5
# Build, test, or archive an Xcode workspace on macOS. Optionally package an
app.
- task: Xcode@5
  inputs:
    actions: 'build' # string. Required. Actions. Default: build.
    #configuration: '$(Configuration)' # string. Configuration. Default:
$(Configuration).
    #sdk: '$(SDK)' # string. SDK. Default: $(SDK).
    #xcWorkspacePath: '**/*.xcodeproj/project.xcworkspace' # string.
    Workspace or project path. Default: **/*.xcodeproj/project.xcworkspace.
    #scheme: # string. Scheme.
    #xcodeVersion: 'default' # '8' | '9' | '10' | '11' | '12' | '13' |
'default' | 'specifyPath'. Xcode version. Default: default.
    #xcodeDeveloperDir: # string. Optional. Use when xcodeVersion ==
specifyPath. Xcode developer path.
    # Package options
    #packageApp: false # boolean. Create app package. Default: false.
    #archivePath: # string. Optional. Use when packageApp == true. Archive
path.
    #exportPath: 'output/$(SDK)/$(Configuration)' # string. Optional. Use
when packageApp == true. Export path. Default:
output/$(SDK)/$(Configuration).
    #exportOptions: 'auto' # 'auto' | 'plist' | 'specify'. Optional. Use
when packageApp == true. Export options. Default: auto.
    #exportMethod: 'development' # string. Required when exportOptions ==
specify. Export method. Default: development.
    #exportTeamId: # string. Optional. Use when exportOptions == specify.
Team ID.
    #exportOptionsPlist: # string. Required when exportOptions == plist.
    Export options plist.
    #exportArgs: # string. Optional. Use when packageApp == true. Export
arguments.
    # Signing & provisioning
    #signingOption: 'nosign' # 'nosign' | 'default' | 'manual' | 'auto'.
    Signing style. Default: nosign.
    #signingIdentity: # string. Optional. Use when signingOption = manual.
    Signing identity.
    #provisioningProfileUuid: # string. Optional. Use when signingOption =
manual. Provisioning profile UUID.
```

```
#provisioningProfileName: # string. Optional. Use when signingOption = manual. Provisioning profile name.  
#teamId: # string. Optional. Use when signingOption = auto. Team ID.  
# Devices & simulators  
#destinationPlatformOption: 'default' # 'default' | 'iOS' | 'tvOS' | 'macOS' | 'custom'. Destination platform. Default: default.  
#destinationPlatform: # string. Optional. Use when destinationPlatformOption == custom. Custom destination platform.  
#destinationTypeOption: 'simulators' # 'simulators' | 'devices'. Optional. Use when destinationPlatformOption != default && destinationPlatformOption != macOS. Destination type. Default: simulators.  
#destinationSimulators: # string. Optional. Use when destinationPlatformOption != default && destinationPlatformOption != macOS && destinationTypeOption == simulators. Simulator.  
#destinationDevices: # string. Optional. Use when destinationPlatformOption != default && destinationPlatformOption != macOS && destinationTypeOption == devices. Device.  
# Advanced  
#args: # string. Arguments.  
#workingDirectory: # string. Alias: cwd. Working directory.  
#useXcpretty: true # boolean. Use xcpretty. Default: true.  
#xcprettyArgs: # string. Optional. Use when useXcpretty == true. Xcpretty arguments.  
#publishJUnitResults: false # boolean. Publish test results to Azure Pipelines. Default: false.  
#testRunTitle: # string. Optional. Use when publishJUnitResults == true. Test run title.
```

Inputs

`actions` - Actions

`string`. Required. Default value: `build`.

Specifies a space-delimited list of actions. Some valid options are `build`, `clean`, `test`, `analyze`, and `archive`. For example, `clean build` performs a clean build. See [Apple: Building from the command line with Xcode FAQ](#).

`configuration` - Configuration

`string`. Default value: `$(Configuration)`.

Specifies the Xcode project or workspace configuration to build. When using a variable, specify a value (for example, `Release`) on the [Variables](#) tab.

`sdk` - SDK

`string`. Default value: `$(SDK)`.

Specifies an SDK to use when building the Xcode project or workspace. From the macOS Terminal application, run `xcodebuild -showsdk`s to display the valid list of SDKs. When using a variable, specify a value (for example, `iphonesimulator`) on the [Variables](#) tab.

`xcWorkspacePath` - Workspace or project path

`string`. Default value: `**/*.{xcodeproj/project.xcworkspace}`.

Optional. Specifies a relative path from the root of the repository to the Xcode workspace or project. For example, `MyApp/MyApp.xcworkspace` or `MyApp/MyApp.xcodeproj`. Wildcards can be used. Learn more about [file matching patterns](#).

`scheme` - Scheme

`string`.

Optional. Specifies an Xcode scheme name. *Must be a shared scheme* (shared checkbox under **Managed Schemes** in Xcode). If you do not specify a scheme, and the specified workspace has a single shared scheme, the workspace scheme will be used.

`xcodeVersion` - Xcode version

`string`. Allowed values: `8` (Xcode 8), `9` (Xcode 9), `10` (Xcode 10), `11` (Xcode 11), `12` (Xcode 12), `13` (Xcode 13), `default`, `specifyPath` (Specify path). Default value: `default`.

Specifies the target version of Xcode. Select `Default` to use the default version of Xcode on the agent machine. Specifying a version number (for example, `xcode 9`) relies on the version's location to be set by environment variables on the agent machine (for example, `XCODE_9_DEVELOPER_DIR=/Applications/Xcode_9.0.0.app/Contents/Developer`). Select `Specify path` to provide a specific path to the Xcode developer directory.

`xcodeDeveloperDir` - Xcode developer path

`string`. Optional. Use when `xcodeVersion == specifyPath`.

Specifies a path to a specific Xcode developer directory (for example, `/Applications/Xcode_9.0.0.app/Contents/Developer`). This input is useful when multiple versions of Xcode are installed on the agent machine.

`packageApp` - Create app package

`boolean`. Default value: `false`.

Specifies whether an IPA app package file is generated as a part of the build.

archivePath - Archive path

`string`. Optional. Use when `packageApp == true`.

Specifies a directory where created archives are placed.

exportPath - Export path

`string`. Optional. Use when `packageApp == true`. Default value:
`output/$(SDK)/$(Configuration)`.

Specifies the destination for the product exported from the archive.

exportOptions - Export options

`string`. Optional. Use when `packageApp == true`. Allowed values: `auto` (Automatic),
`plist`, `specify`. Default value: `auto`.

Specifies options for exporting the archive. When the default value of `Automatic` is selected, the export method is automatically detected from the archive. Select `Plist` to specify a plist file containing export options. Select `Specify` to provide a specific **Export method** and **Team ID**.

exportMethod - Export method

`string`. Required when `exportOptions == specify`. Default value: `development`.

Specifies the method that Xcode uses to export the archive. For example: `app-store`,
`package`, `ad-hoc`, `enterprise`, or `development`.

exportTeamId - Team ID

`string`. Optional. Use when `exportOptions == specify`.

Specifies the Apple Developer Portal 10-character team ID to use during the export.

exportOptionsPlist - Export options plist

`string`. Required when `exportOptions == plist`.

Specifies the path to the plist file that contains options to use during the export.

exportArgs - Export arguments

`string`. Optional. Use when `packageApp == true`.

Specifies additional command line arguments used during the export.

signingOption - Signing style

`string`. Allowed values: `nosign` (Do not code sign), `default` (Project defaults), `manual` (Manual signing), `auto` (Automatic signing). Default value: `nosign`.

Specifies the method of signing the build. Select `Do not code sign` to disable signing. Select `Project defaults` to use only the project's signing configuration. Select `Manual signing` to force manual signing and optionally specify a signing identity and provisioning profile. Select `Automatic signing` to force automatic signing and optionally specify a development team ID. If your project requires signing, use the **Install Apple...** tasks to install certificates and provisioning profiles prior to the Xcode build.

signingIdentity - Signing identity

`string`. Optional. Use when `signingOption = manual`.

Specifies a signing identity override with which to sign the build. Unlocking the default keychain on the agent machine may be required. If no value is entered, the Xcode project's setting is used.

provisioningProfileUuid - Provisioning profile UUID

`string`. Optional. Use when `signingOption = manual`.

Specifies the UUID of an installed provisioning profile used for the build. Use separate build tasks with different schemes or targets to specify provisioning profiles by target in a single workspace (iOS, tvOS, watchOS).

provisioningProfileName - Provisioning profile name

`string`. Optional. Use when `signingOption = manual`.

Specifies the name of an installed provisioning profile used for the build. If specified, this takes precedence over the provisioning profile UUID. Use separate build tasks with different schemes or targets to specify provisioning profiles by target in a single workspace (iOS, tvOS, watchOS).

teamId - Team ID

`string`. Optional. Use when `signingOption = auto`.

Required if you are a member of multiple development teams. Specifies the 10-character development team ID.

destinationPlatformOption - Destination platform

`string`. Allowed values: `default`, `ios` (iOS and watchOS), `tvOS`, `macOS`, `custom`. Default value: `default`.

Specifies the destination device's platform used for UI testing when the generic build device isn't valid. Choose `custom` to specify a platform not included in this list. When `Default` is selected, no simulators or devices are targeted.

destinationPlatform - Custom destination platform

`string`. Optional. Use when `destinationPlatformOption == custom`.

Specifies a destination device's platform used for UI testing when the generic build device isn't valid. Choose `custom` to specify a platform not included in the list. When `Default` is selected, no simulators nor devices are targeted.

destinationTypeOption - Destination type

`string`. Optional. Use when `destinationPlatformOption != default && destinationPlatformOption != macOS`. Allowed values: `simulators` (Simulator), `devices` (Connected Device). Default value: `simulators`.

Specifies the destination type to use for UI testing. Devices must be connected to the Mac performing the build via a cable or network connection. See **Devices and Simulators** in Xcode for more information.

destinationSimulators - Simulator

`string`. Optional. Use when `destinationPlatformOption != default && destinationPlatformOption != macOS && destinationTypeOption == simulators`.

Specifies an Xcode simulator name used for UI testing. For example, `iPhone X` (iOS and watchOS) or `Apple TV 4K` (tvOS). An optional target OS version can be specified in the format `OS=<versionNumber>`, such as `iPhone X,OS=11.1`. See this [list of simulators installed on the Hosted macOS agent](#) for more information.

destinationDevices - Device

`string`. Optional. Use when `destinationPlatformOption != default && destinationPlatformOption != macOS && destinationTypeOption == devices`.

Specifies the name of the device used for UI testing, such as `Raisa's iPad`. Only one device is currently supported. Note that Apple does not allow apostrophes ('') in device names. Instead, right single quotation marks (') can be used.

args - Arguments

`string`.

Optional. Specifies additional command line arguments with which to build. This input is useful for specifying `-target` or `-project` arguments instead of a workspace/project and scheme. See [Apple: Building from the command line with Xcode FAQ ↗](#).

workingDirectory - Working directory

Input alias: `cwd`. `string`.

Optional. Specifies the working directory in which to run the build. If no value is entered, the root of the repository is used.

useXcpretty - Use xcpretty

`boolean`. Default value: `true`.

Specifies whether to use `xcpretty` to format `xcodebuild` output. `xcpretty` must be installed on the agent machine (It is preinstalled on Azure Pipelines hosted build agents). If `xcpretty` is not installed, raw `xcodebuild` output is shown. See [xcpretty ↗](#) for more information.

xcprettyArgs - Xcpretty arguments

`string`. Optional. Use when `useXcpretty == true`.

If `xcpretty` is enabled, this input specifies arguments for `xcpretty`. See [a list of xcpretty arguments on GitHub ↗](#).

`publishJUnitResults` - Publish test results to Azure Pipelines

`boolean`. Default value: `false`.

Specifies whether to publish JUnit test results to Azure Pipelines. This requires `xcpretty` to be enabled to generate JUnit test results.

`testRunTitle` - Test run title

`string`. Optional. Use when `publishJUnitResults == true`.

If `xcpretty` and `publishJUnitResults` are enabled, you can specify the test run title.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to build, test, or archive an Xcode workspace on macOS, and optionally package an app.

Using multiple provisioning profiles

Currently there's no support of multiple provisioning profiles for the Xcode task (for example for iOS App Extension).

Examples

[Build your Xcode app](#)

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: xcode
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Build

Xcode@4 - Xcode v4 task

Article • 09/26/2023

Use this task to build, test, or archive an Xcode workspace on macOS, and optionally package an app.

Syntax

YAML

```
# Xcode v4
# Build, test, or archive an Xcode workspace on macOS. Optionally package an
app.
- task: Xcode@4
  inputs:
    actions: 'build' # string. Required. Actions. Default: build.
    #configuration: '$(Configuration)' # string. Configuration. Default:
$(Configuration).
    #sdk: '$(SDK)' # string. SDK. Default: $(SDK).
    #xcWorkspacePath: '**/*.xcodeproj/project.xcworkspace' # string.
    Workspace or project path. Default: **/*.xcodeproj/project.xcworkspace.
    #scheme: # string. Scheme.
    #xcodeVersion: 'default' # '8' | '9' | 'default' | 'specifyPath'. Xcode
version. Default: default.
    #xcodeDeveloperDir: # string. Optional. Use when xcodeVersion ==
specifyPath. Xcode developer path.
    # Package options
    #packageApp: false # boolean. Create app package. Default: false.
    #archivePath: # string. Optional. Use when packageApp == true. Archive
path.
    #exportPath: 'output/$(SDK)/$(Configuration)' # string. Optional. Use
when packageApp == true. Export path. Default:
output/$(SDK)/$(Configuration).
    #exportOptions: 'auto' # 'auto' | 'plist' | 'specify'. Optional. Use
when packageApp == true. Export options. Default: auto.
    #exportMethod: 'development' # string. Required when exportOptions ==
specify. Export method. Default: development.
    #exportTeamId: # string. Optional. Use when exportOptions == specify.
Team ID.
    #exportOptionsPlist: # string. Required when exportOptions == plist.
    Export options plist.
    #exportArgs: # string. Optional. Use when packageApp == true. Export
arguments.
    # Signing & provisioning
    #signingOption: 'nosign' # 'nosign' | 'default' | 'manual' | 'auto'.
    Signing style. Default: nosign.
    #signingIdentity: # string. Optional. Use when signingOption = manual.
    Signing identity.
    #provisioningProfileUuid: # string. Optional. Use when signingOption =
manual. Provisioning profile UUID.
```

```
#teamId: # string. Optional. Use when signingOption = auto. Team ID.  
# Devices & simulators  
#destinationPlatformOption: 'default' # 'default' | 'iOS' | 'tvOS' |  
'macOS' | 'custom'. Destination platform. Default: default.  
#destinationPlatform: # string. Optional. Use when  
destinationPlatformOption == custom. Custom destination platform.  
#destinationTypeOption: 'simulators' # 'simulators' | 'devices'.  
Optional. Use when destinationPlatformOption != default &&  
destinationPlatformOption != macOS. Destination type. Default: simulators.  
#destinationSimulators: 'iPhone 7' # string. Optional. Use when  
destinationPlatformOption != default && destinationPlatformOption != macOS  
&& destinationTypeOption == simulators. Simulator. Default: iPhone 7.  
#destinationDevices: # string. Optional. Use when  
destinationPlatformOption != default && destinationPlatformOption != macOS  
&& destinationTypeOption == devices. Device.  
# Advanced  
#args: # string. Arguments.  
#workingDirectory: # string. Alias: cwd. Working directory.  
#outputPattern: # string. Output directory.  
#useXcpretty: false # boolean. Use xcpretty. Default: false.  
#publishJUnitResults: false # boolean. Publish test results to VSTS/TFS.  
Default: false.
```

Inputs

`actions` - Actions

`string`. Required. Default value: `build`.

Specifies a space-delimited list of actions. Valid options are `build`, `clean`, `test`, `analyze`, and `archive`. For example, `clean build` performs a clean build. See the [Apple: Building from the command line with Xcode FAQ](#).

`configuration` - Configuration

`string`. Default value: `$(Configuration)`.

Specifies the Xcode project or workspace configuration to build. When using a variable, specify a value (for example, `Release`) on the [Variables](#) tab.

`sdk` - SDK

`string`. Default value: `$(SDK)`.

Specifies an SDK to use when building the Xcode project or workspace. From the macOS Terminal application, run `xcodebuild -showsdk` to display the valid list of SDKs. When using a variable, specify a value (for example, `iphonesimulator`) on the [Variables](#) tab.

xcWorkspacePath - Workspace or project path

`string`. Default value: `**/*.{xcworkspace, xcproj}`.

Optional. Specifies a relative path from the root of the repository to the Xcode workspace or project. If you specify a value, you must also specify the scheme. Do not specify a value if you are specifying `-target` flag in Advanced Arguments. For example, `MyApp/MyApp.xcworkspace` or `MyApp/MyApp.xcodeproj`.

scheme - Scheme

`string`.

Optional. Specifies an Xcode scheme name. *Must be a shared scheme* (shared checkbox under **Managed Schemes** in Xcode). If you do not specify a scheme, and the specified workspace has a single shared scheme, the workspace scheme will be used.

xcodeVersion - Xcode version

`string`. Allowed values: `8` (Xcode 8), `9` (Xcode 9), `default`, `specifyPath` (Specify path).

Default value: `default`.

Specifies the target version of Xcode. Select `Default` to use the default version of Xcode on the agent machine. Specifying a version number (for example, `Xcode 9`) relies on the version's location to be set by environment variables on the agent machine (for example, `XCODE_9_DEVELOPER_DIR=/Applications/Xcode_9.0.0.app/Contents/Developer`). Select `Specify path` to provide a specific path to the Xcode developer directory.

xcodeDeveloperDir - Xcode developer path

`string`. Optional. Use when `xcodeVersion == specifyPath`.

Specifies a path to a specific Xcode developer directory (for example, `/Applications/Xcode_9.0.0.app/Contents/Developer`). This input is useful when multiple versions of Xcode are installed on the agent machine.

packageApp - Create app package

`boolean`. Default value: `false`.

Specifies whether an IPA app package file should be generated as a part of the build.

archivePath - Archive path

`string`. Optional. Use when `packageApp == true`.

Specifies a directory where created archives are placed.

exportPath - Export path

`string`. Optional. Use when `packageApp == true`. Default value:
`output/$(SDK)/$(Configuration)`.

Specifies the destination for the product exported from the archive.

exportOptions - Export options

`string`. Optional. Use when `packageApp == true`. Allowed values: `auto` (Automatic),
`plist`, `specify`. Default value: `auto`.

Specifies options for exporting the archive. When the default value of `Automatic` is selected, the export method is automatically detected from the archive. Select `Plist` to specify a plist file containing export options. Select `Specify` to provide a specific **Export method** and **Team ID**.

exportMethod - Export method

`string`. Required when `exportOptions == specify`. Default value: `development`.

Specifies the method that Xcode uses to export the archive. For example: `app-store`,
`package`, `ad-hoc`, `enterprise`, or `development`.

exportTeamId - Team ID

`string`. Optional. Use when `exportOptions == specify`.

Specifies the Apple Developer Portal 10-character team ID to use during the export.

exportOptionsPlist - Export options plist

`string`. Required when `exportOptions == plist`.

Specifies the path to the plist file that contains options to use during the export.

`exportArgs` - Export arguments

`string`. Optional. Use when `packageApp == true`.

Specifies additional command line arguments used during the export.

`signingOption` - Signing style

`string`. Allowed values: `nosign` (Do not code sign), `default` (Project defaults), `manual` (Manual signing), `auto` (Automatic signing). Default value: `nosign`.

Specifies the method of signing the build. Select `Do not code sign` to disable signing. Select `Project defaults` to use only the project's signing configuration. Select `Manual signing` to force manual signing and optionally specify a signing identity and provisioning profile. Select `Automatic signing` to force automatic signing and optionally specify a development team ID. If your project requires signing, use the **Install Apple...** tasks to install certificates and provisioning profiles prior to the Xcode build.

`signingIdentity` - Signing identity

`string`. Optional. Use when `signingOption = manual`.

Specifies a signing identity override with which to sign the build. Unlocking the default keychain on the agent machine may be required. If no value is entered, the Xcode project's setting is used.

`provisioningProfileUuid` - Provisioning profile UUID

`string`. Optional. Use when `signingOption = manual`.

Specifies the UUID of an installed provisioning profile used for the build. Use separate build tasks with different schemes or targets to specify provisioning profiles by target in a single workspace (iOS, tvOS, watchOS).

`teamId` - Team ID

`string`. Optional. Use when `signingOption = auto`.

Required if you are a member of multiple development teams. Specifies the 10-character development team ID.

`destinationPlatformOption` - Destination platform

`string`. Allowed values: `default`, `ios` (iOS and watchOS), `tvOS`, `macOS`, `custom`. Default

`value: default.`

Specifies the destination device's platform used for UI testing when the generic build device isn't valid. Choose `custom` to specify a platform not included in the list. When `Default` is selected, no simulators or devices are targeted.

`destinationPlatform` - Custom destination platform

`string`. Optional. Use when `destinationPlatformOption == custom`.

Specifies a destination device's platform used for UI testing when the generic build device isn't valid.

`destinationTypeOption` - Destination type

`string`. Optional. Use when `destinationPlatformOption != default && destinationPlatformOption != macOS`. Allowed values: `simulators` (Simulator), `devices` (Connected Device). Default value: `simulators`.

Specifies the destination type used for UI testing. Devices must be connected to the Mac performing the build via a cable or network connection. See [Devices and Simulators](#) in Xcode for more information.

`destinationSimulators` - Simulator

`string`. Optional. Use when `destinationPlatformOption != default && destinationPlatformOption != macOS && destinationTypeOption == simulators`. Default value: `iPhone 7`.

Specifies an Xcode simulator name used for UI testing. For example, `iPhone x` (iOS and watchOS) or `Apple TV 4K` (tvOS). An optional target OS version can be specified in the format `OS=<versionNumber>`, such as `iPhone X,OS=11.1`. Learn more about [installed simulators on the Hosted macOS Preview agent](#).

`destinationDevices` - Device

`string`. Optional. Use when `destinationPlatformOption != default && destinationPlatformOption != macOS && destinationTypeOption == devices`.

Specifies the name of the device used for UI testing, such as `Raisa's iPad`.

`args` - Arguments

`string`.

Optional. Specifies additional command line arguments with which to build. This input is useful for specifying `-target` or `-project` arguments instead of a workspace/project and scheme. See the [Apple: Building from the command line with Xcode FAQ](#).

`workingDirectory` - Working directory

Input alias: `cwd`. `string`.

Optional. Specifies the working directory in which to run the build. If no value is entered, the root of the repository is used.

`outputPattern` - Output directory

`string`.

Optional. Specifies a relative path to the working directory where build output (binaries) are placed. For example: `output/$(SDK)/$(Configuration)` or `output/$(TestSDK)/$(TestConfiguration)`. Archive and export paths are configured separately. Specify values on the [Variables tab](#).

`useXcpretty` - Use xcpretty

`boolean`. Default value: `false`.

Specifies whether to use `xcpretty` to format `xcodebuild` output, and generates JUnit test results. `xcpretty` must be installed on the agent machine (It is preinstalled on VSTS hosted build agents). See [xcpretty](#) for more information.

`publishJUnitResults` - Publish test results to VSTS/TFS

`boolean`. Default value: `false`.

If `xcpretty` is enabled, this input specifies whether to publish the JUnit test results to VSTS/TFS.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to build an Xcode workspace on macOS.

Examples

- Build, test, and deploy Xcode apps

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: xcode
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Build

Xcode@3 - Xcode Build v3 task

Article • 09/26/2023

Use this task to build an Xcode workspace on macOS.

Syntax

YAML

```
# Xcode Build v3
# Build an Xcode workspace on macOS.
- task: Xcode@3
  inputs:
    actions: 'build' # string. Required. Actions. Default: build.
    #configuration: '$(Configuration)' # string. Configuration. Default: $(Configuration).
    #sdk: '$(SDK)' # string. SDK. Default: $(SDK).
    #xcWorkspacePath: '**/*.xcodeproj/*.xcworkspace' # string.
    Workspace/Project Path. Default: **/*.xcodeproj/*.xcworkspace.
    #scheme: # string. Scheme.
    #packageApp: true # boolean. Create App Package. Default: true.
    # Package Options
    #archivePath: # string. Archive Path.
    #exportPath: 'output/$(SDK)/$(Configuration)' # string. Export Path.
    Default: output/$(SDK)/$(Configuration).
    #exportOptions: 'auto' # 'auto' | 'plist' | 'specify'. Export Options.
    Default: auto.
    #exportMethod: 'development' # string. Required when exportOptions == specify. Export Method. Default: development.
    #exportTeamId: # string. Optional. Use when exportOptions == specify.
    Team ID.
    #exportOptionsPlist: # string. Required when exportOptions == plist.
    Export Options Plist.
    #exportArgs: # string. Export Arguments.
    # Signing & Provisioning
    #xcode8AutomaticSigning: false # boolean. Automatic Signing. Default: false.
    #teamId: # string. Optional. Use when xcode8AutomaticSigning = true.
    Team ID.
    #signMethod: 'file' # 'file' | 'id'. Override Using. Default: file.
    #iosSigningIdentity: # string. Optional. Use when signMethod = id.
    Signing Identity.
    #unlockDefaultKeychain: false # boolean. Optional. Use when signMethod = id. Unlock Default Keychain. Default: false.
    #defaultKeychainPassword: # string. Optional. Use when signMethod = id.
    Default Keychain Password.
    #provProfileUuid: # string. Optional. Use when signMethod = id.
    Provisioning Profile UUID.
    #p12: # string. Optional. Use when signMethod = file. P12 Certificate File.
```

```
#p12pwd: # string. Optional. Use when signMethod = file. P12 Password.  
#provProfile: # string. Optional. Use when signMethod = file.  
Provisioning Profile File.  
#removeProfile: false # boolean. Optional. Use when signMethod = file.  
Remove Profile After Build. Default: false.  
# Advanced  
#args: # string. Arguments.  
#cwd: # string. Working Directory.  
outputPattern: 'output/${SDK}/${Configuration}' # string. Required.  
Output Directory. Default: output/${SDK}/${Configuration}.  
#xcodeDeveloperDir: # string. Xcode Developer Path.  
#useXcpretty: false # boolean. Use xcpretty. Default: false.  
#publishJUnitResults: false # boolean. Publish to VSTS/TFS. Default:  
false.
```

Inputs

`actions` - Actions

`string`. Required. Default value: `build`.

Specifies a space-delimited list of actions. Valid options are `build`, `clean`, `test`, `analyze`, and `archive`. For example: `build clean` performs a clean build. See the [Apple: Building from the command line with Xcode FAQ](#).

`configuration` - Configuration

`string`. Default value: `$(Configuration)`.

Specifies the Xcode project or workspace configuration to build. When using a variable, specify a value (for example, `Release`) on the **Variables** tab.

`sdk` - SDK

`string`. Default value: `$(SDK)`.

Builds an Xcode project or workspace against the specified SDK. Run `xcodebuild -showsdk` to see a valid list of SDKs.

`xcworkspacePath` - Workspace/Project Path

`string`. Default value: `**/*.{xcodeproj,*.xcworkspace}`.

Optional. Specifies the relative path from the repo root to the Xcode workspace or project. For example: `MyApp/MyApp.xcworkspace` or

`MyApp/MyApp.xcworkspace/MyApp.xcodeproj`. Leave blank if you intend to use `-target` flag under Advanced Arguments.

`scheme` - Scheme

`string`.

Optional. Specifies the Xcode scheme name. *Must be a shared scheme* (shared checkbox under Managed Schemes in Xcode). Required if Workspace is specified.

`packageApp` - Create App Package

`boolean`. Default value: `true`.

Specifies whether an IPA is generated as a part of the build. For exporting archives with Xcode 7 and Xcode 8, review additional inputs in the **Package Options** section.

`archivePath` - Archive Path

`string`.

Optional. Specifies a directory where created archives are placed.

`exportPath` - Export Path

`string`. Default value: `output/$(SDK)/$(Configuration)`.

Optional. Specifies the destination for the product exported from the archive.

`exportOptions` - Export Options

`string`. Allowed values: `auto`, `plist`, `specify`. Default value: `auto`.

Specifies a way to pass in **Export Options** when exporting the archive.

`exportMethod` - Export Method

`string`. Required when `exportOptions == specify`. Default value: `development`.

Specifies the method Xcode uses to export the archive. For example, `app-store`, `package`, `ad-hoc`, `enterprise`, or `development`.

`exportTeamId` - Team ID

`string`. Optional. Use when `exportOptions == specify`.

Specifies the Apple Developer Portal 10-digit team ID to use for the export.

`exportOptionsPlist` - Export Options Plist

`string`. Required when `exportOptions == plist`.

Specifies the path to a plist file that configures archive exporting.

`exportArgs` - Export Arguments

`string`.

Specifies additional command line arguments used to export.

`xcode8AutomaticSigning` - Automatic Signing

`boolean`. Default value: `false`.

Use this input if you have an Xcode 8 or Xcode 9 project configured for Automatic Signing.

`teamId` - Team ID

`string`. Optional. Use when `xcode8AutomaticSigning = true`.

Specifies the 10-digit developer team ID. This is required if you are a member of multiple development teams.

`signMethod` - Override Using

`string`. Allowed values: `file` (File Contents), `id` (Identifiers). Default value: `file`.

Use this input if the build uses a signing or provisioning method that is different than the default. Choose `File Contents` to use a P12 certificate and provisioning profile. Choose `Identifiers` to retrieve signing settings from the default keychain and pre-installed profiles. Leave the corresponding fields blank if you do not wish to override the default build settings.

`iosSigningIdentity` - Signing Identity

`string`. Optional. Use when `signMethod = id`.

Specifies the signing identity override that is used to sign the build. Defaults to the Xcode project setting. **Unlock Default Keychain** may need to be selected.

unlockDefaultKeychain - Unlock Default Keychain

`boolean`. Optional. Use when `signMethod = id`. Default value: `false`.

Resolves **User interaction is not allowed** errors by unlocking the default keychain.

defaultKeychainPassword - Default Keychain Password

`string`. Optional. Use when `signMethod = id`.

Specifies the password to unlock the default keychain.

provProfileUuid - Provisioning Profile UUID

`string`. Optional. Use when `signMethod = id`.

Specifies the UUID of an installed provisioning profile to use for the build. Use separate build tasks with different schemes or targets to specify provisioning profiles by target in a single workspace (iOS, WatchKit, tvOS).

p12 - P12 Certificate File

`string`. Optional. Use when `signMethod = file`.

Specifies the relative path to a PKCS12 formatted P12 certificate file that contains a signing certificate to be used for the build.

p12pwd - P12 Password

`string`. Optional. Use when `signMethod = file`.

Specifies the password to a P12 certificate file. Use a build variable to encrypt.

provProfile - Provisioning Profile File

`string`. Optional. Use when `signMethod = file`.

Specifies the relative path to a file containing a provisioning profile override to be used for the build. Use separate build tasks with different schemes or targets to specify provisioning profiles by target in a single workspace (iOS, WatchKit, tvOS).

`removeProfile` - Remove Profile After Build

`boolean`. Optional. Use when `signMethod = file`. Default value: `false`.

Removes the contents of the provisioning profile file from the build agent after the build is complete. **Only check if you are running one agent per user.**

`args` - Arguments

`string`.

Specifies additional command line arguments used to build. This input is useful if you want to use `-target` or `-project` instead of specifying a workspace and scheme.

`cwd` - Working Directory

`string`.

Specifies the working directory for build runs. Defaults to the root of the repository.

`outputPattern` - Output Directory

`string`. Required. Default value: `output/$(SDK)/$(Configuration)`.

Specifies the relative path where build output (binaries) are placed.

`xcodeDeveloperDir` - Xcode Developer Path

`string`.

Optional. Specifies the path to the Xcode Developer folder if it's not the system default. For use when multiple versions of Xcode are installed on a system. For example:

`/Applications/Xcode 7.app/Contents/Developer`.

`useXcpretty` - Use xcpretty

`boolean`. Default value: `false`.

Formats `xcodebuild` output and generates a JUnit test results report. Must be installed on agent hosts. Learn more about [xcpretty](#).

`publishJUnitResults` - Publish to VSTS/TFS

`boolean`. Default value: `false`.

JUnit test results that were produced using `xctool` are published to VSTS/TFS.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: xcode
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Build

Xcode@2 - Xcode Build v2 task

Article • 09/26/2023

Use this task to build an Xcode workspace on macOS.

Syntax

YAML

```
# Xcode Build v2
# Build an Xcode workspace on Mac OS.
- task: Xcode@2
  inputs:
    actions: 'build' # string. Required. Actions. Default: build.
    #configuration: '$(Configuration)' # string. Configuration. Default: $(Configuration).
    #sdk: '$(SDK)' # string. SDK. Default: $(SDK).
    #xcWorkspacePath: '**/*.xcodeproj/*.xcworkspace' # string.
    Workspace/Project Path. Default: **/*.xcodeproj/*.xcworkspace.
    #scheme: # string. Scheme.
    #packageApp: true # boolean. Create App Package. Default: true.
    # Package Options
    packageTool: 'xcodebuild' # 'xcrun' | 'xcodebuild'. Required. Create
    Package (IPA) using. Default: xcodebuild.
    #archivePath: # string. Optional. Use when packageTool == xcodebuild.
    Archive Path.
    #exportPath: 'output/$(SDK)/$(Configuration)' # string. Optional. Use
    when packageTool == xcodebuild. Export Path. Default:
    output/$(SDK)/$(Configuration).
    #exportOptions: 'auto' # 'auto' | 'plist' | 'specify'. Optional. Use
    when packageTool == xcodebuild. Export Options. Default: auto.
    #exportMethod: 'development' # string. Required when exportOptions ==
    specify. Export Method. Default: development.
    #exportTeamId: # string. Optional. Use when exportOptions == specify.
    Team ID.
    #exportOptionsPlist: # string. Required when exportOptions == plist.
    Export Options Plist.
    # Signing & Provisioning
    #xcode8AutomaticSigning: false # boolean. Automatic Signing. Default:
    false.
    #teamId: # string. Optional. Use when xcode8AutomaticSigning = true.
    Team ID.
    #signMethod: 'file' # 'file' | 'id'. Override Using. Default: file.
    #iosSigningIdentity: # string. Optional. Use when signMethod = id.
    Signing Identity.
    #unlockDefaultKeychain: false # boolean. Optional. Use when signMethod =
    id. Unlock Default Keychain. Default: false.
    #defaultKeychainPassword: # string. Optional. Use when signMethod = id.
    Default Keychain Password.
    #provProfileUuid: # string. Optional. Use when signMethod = id.
```

```
Provisioning Profile UUID.  
  #p12: # string. Optional. Use when signMethod = file. P12 Certificate  
  File.  
  #p12pwd: # string. Optional. Use when signMethod = file. P12 Password.  
  #provProfile: # string. Optional. Use when signMethod = file.  
Provisioning Profile File.  
  #removeProfile: false # boolean. Optional. Use when signMethod = file.  
Remove Profile After Build. Default: false.  
# Advanced  
#args: # string. Arguments.  
#cwd: # string. Working Directory.  
outputPattern: 'output/${SDK}/${Configuration}' # string. Required.  
Output Directory. Default: output/${SDK}/${Configuration}.  
#xcodeDeveloperDir: # string. Xcode Developer Path.  
#useXcpretty: false # boolean. Use xcpretty. Default: false.  
#publishJUnitResults: false # boolean. Publish to VSTS/TFS. Default:  
false.  
# xctool (deprecated)  
#useXctool: # boolean. Use xctool.  
#xctoolReporter: # string. xctool Test Reporter Format.
```

Inputs

`actions` - Actions

`string`. Required. Default value: `build`.

Specifies a space-delimited list of actions. Valid options are `build`, `clean`, `test`, `analyze`, and `archive`. For example, `build clean` performs a clean build. See the [Apple: Building from the command line with Xcode FAQ ↗](#).

`configuration` - Configuration

`string`. Default value: `$(Configuration)`.

Specifies the Xcode project or workspace configuration to build. When using a variable, specify a value (for example, `Release`) on the **Variables** tab.

`sdk` - SDK

`string`. Default value: `$(SDK)`.

Builds an Xcode project or workspace against the specified SDK. Run `xcodebuild - showsdks` to see a valid list of SDKs.

xcworkspacePath - Workspace/Project Path

`string`. Default value: `**/*.{xcodeproj,*.xcworkspace}`.

Optional. Specifies the relative path from the repo root to the Xcode workspace or project. For example: `MyApp/MyApp.xcworkspace` or

`MyApp/MyApp.xcworkspace/MyApp.xcodeproj`.

Leave blank if you intend to use `-target` flag under **Advanced Arguments**.

scheme - Scheme

`string`.

Optional. Specifies the Xcode scheme name. *Must be a shared scheme* (shared checkbox under **Managed Schemes** in Xcode). **Required if Workspace is specified**.

packageApp - Create App Package

`boolean`. Default value: `true`.

Specifies whether an IPA is generated as a part of the build. For exporting archives with Xcode 7 and Xcode 8, review additional inputs in the **Package Options** section.

packageTool - Create Package (IPA) using

`string`. Required. Allowed values: `xcrun` (xcrun (deprecated by Apple)), `xcdebuild` (xcdebuild archive and export). Default value: `xcdebuild`.

Specifies the tool to use for generating the IPA.

archivePath - Archive Path

`string`. Optional. Use when `packageTool == xcdebuild`.

Specifies a directory where created archives are placed.

exportPath - Export Path

`string`. Optional. Use when `packageTool == xcdebuild`. Default value: `output/$(SDK)/$(Configuration)`.

Specifies the destination for the product exported from the archive.

exportOptions - Export Options

`string`. Optional. Use when `packageTool == xcdebuild`. Allowed values: `auto`, `plist`, `specify`. Default value: `auto`.

Specifies a way to pass in **Export Options** when exporting the archive.

exportMethod - Export Method

`string`. Required when `exportOptions == specify`. Default value: `development`.

Specifies the method Xcode uses to export the archive. For example, `app-store`, `package`, `ad-hoc`, `enterprise`, or `development`.

exportTeamId - Team ID

`string`. Optional. Use when `exportOptions == specify`.

Specifies the Apple Developer Portal 10-digit team ID to use for the export.

exportOptionsPlist - Export Options Plist

`string`. Required when `exportOptions == plist`.

Specifies the path to a plist file that configures archive exporting.

xcode8AutomaticSigning - Automatic Signing

`boolean`. Default value: `false`.

Use this input if you have an Xcode 8 or Xcode 9 project configured for Automatic Signing.

teamId - Team ID

`string`. Optional. Use when `xcode8AutomaticSigning = true`.

Specifies the 10-digit developer team ID. This is required if you are a member of multiple development teams.

signMethod - Override Using

`string`. Allowed values: `file` (File Contents), `id` (Identifiers). Default value: `file`.

Use this input if the build uses a signing or provisioning method that is different than the default. Choose `File Contents` to use a P12 certificate and provisioning profile. Choose `Identifiers` to retrieve signing settings from the default keychain and pre-installed profiles. Leave the corresponding fields blank if you do not wish to override the default build settings.

`iosSigningIdentity` - Signing Identity

`string`. Optional. Use when `signMethod = id`.

Specifies the signing identity override that is used to sign the build. Defaults to the Xcode project setting. **Unlock Default Keychain** may need to be selected.

`unlockDefaultKeychain` - Unlock Default Keychain

`boolean`. Optional. Use when `signMethod = id`. Default value: `false`.

Resolves **User interaction is not allowed** errors by unlocking the default keychain.

`defaultKeychainPassword` - Default Keychain Password

`string`. Optional. Use when `signMethod = id`.

Specifies the password to unlock the default keychain.

`provProfileUuid` - Provisioning Profile UUID

`string`. Optional. Use when `signMethod = id`.

Specifies the UUID of an installed provisioning profile to use for the build. Use separate build tasks with different schemes or targets to specify provisioning profiles by target in a single workspace (iOS, WatchKit, tvOS).

`p12` - P12 Certificate File

`string`. Optional. Use when `signMethod = file`.

Specifies the relative path to a PKCS12 formatted P12 certificate file that contains a signing certificate to be used for the build.

`p12pwd` - P12 Password

`string`. Optional. Use when `signMethod = file`.

Specifies the password to a P12 certificate file if specified. Use a build variable to encrypt.

provProfile - Provisioning Profile File

`string`. Optional. Use when `signMethod = file`.

Specifies the relative path to a file containing a provisioning profile override to be used for the build. Use separate build tasks with different schemes or targets to specify provisioning profiles by target in a single workspace (iOS, WatchKit, tvOS).

removeProfile - Remove Profile After Build

`boolean`. Optional. Use when `signMethod = file`. Default value: `false`.

Removes the contents of the provisioning profile file from the build agent after the build is complete. **Only check if you are running one agent per user.**

args - Arguments

`string`.

Specifies additional command line arguments used to build. This input is useful if you want to use `-target` or `-project` instead of specifying a workspace and scheme.

cwd - Working Directory

`string`.

Specifies the working directory for build runs. Defaults to the root of the repository.

outputPattern - Output Directory

`string`. Required. Default value: `output/${SDK}/${Configuration}`.

Specifies the relative path where build output (binaries) are placed.

xcodeDeveloperDir - Xcode Developer Path

`string`.

Optional. Specifies the path to the Xcode Developer folder if it's not the system default. For use when multiple versions of Xcode are installed on a system. For example:
`/Applications/Xcode 7.app/Contents/Developer`.

`useXcpretty` - Use xcpretty

`boolean`. Default value: `false`.

Formats `xcodebuild` output and generates a JUnit test results report. Must be installed on agent hosts. Learn more about [xcpretty](#).

`publishJUnitResults` - Publish to VSTS/TFS

`boolean`. Default value: `false`.

JUnit test results that were produced using `xctool` are published to VSTS/TFS.

`useXctool` - Use xctool

`boolean`.

Uses `xctool` instead of `xcodebuild`. Must be installed on agent hosts. Learn more about [xctool](#).

Note: `xctool` is deprecated and does not work with Xcode 8.

`xctoolReporter` - xctool Test Reporter Format

`string`.

Tests the reporter format to use when the `test` action is specified and **Use xctool** is checked. Specify `junit:output-file-path-here.xml` to generate a file format compatible with the Publish Test Results task. When specified, `plain` is automatically added. `xctool` must be installed on agent hosts. Learn more about [xctool](#).

Note: `xctool` is deprecated and does not work with Xcode 8.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: xcode
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Build

XcodePackageiOS@0 - Xcode Package iOS v0 task

Article • 09/26/2023

Use this task to generate an .ipa file from Xcode build output using xcrun (Xcode 7 or below).

This task is deprecated.

Syntax

YAML

```
# Xcode Package iOS v0
# Generate an .ipa file from Xcode build output using xcrun (Xcode 7 or
below).
- task: XcodePackageiOS@0
  inputs:
    appName: 'name.app' # string. Required. Name of .app. Default: name.app.
    ipaName: 'name.ipa' # string. Required. Name of .ipa. Default: name.ipa.
    provisioningProfile: # string. Required. Provisioning Profile Name.
    sdk: 'iphoneos' # string. Required. SDK. Default: iphoneos.
  # Advanced
    appPath: '$(SDK)/$(Configuration)/build.sym/$(Configuration)-$(SDK)' #
string. Required. Path to .app. Default:
$(SDK)/$(Configuration)/build.sym/$(Configuration)-$(SDK).
    ipaPath:
      '$(SDK)/$(Configuration)/build.sym/$(Configuration)-$(SDK)/output' # string.
Required. Path to place .ipa. Default:
$(SDK)/$(Configuration)/build.sym/$(Configuration)-$(SDK)/output.
```

Inputs

`appName` - Name of .app

`string`. Required. Default value: `name.app`.

Specifies the name of the .app, which is sometimes different from the .ipa.

`ipaName` - Name of .ipa

`string`. Required. Default value: `name.ipa`.

Specifies the name of the .ipa, which is sometimes different from the .app.

provisioningProfile - Provisioning Profile Name

string. Required.

Specifies the name of the provisioning profile to use when signing.

sdk - SDK

string. Required. Default value: `iphoneos`.

Specifies the SDK. Run `xcodebuild -showsdk`s to see the valid list of SDKs.

appPath - Path to .app

string. Required. Default value:

`$(SDK)/$(Configuration)/build.sym/$(Configuration)-$(SDK)`.

Specifies the relative path to the built .app file.

ipaPath - Path to place .ipa

string. Required. Default value:

`$(SDK)/$(Configuration)/build.sym/$(Configuration)-$(SDK)/output`.

Specifies the relative path where the .ipa is placed. The directory is created if it doesn't exist.

Task control options

All tasks have control options in addition to their task inputs. For more information, see [Control options and common task properties](#).

Output variables

None.

Remarks

Use this task to generate an .ipa file from Xcode build output.

 **Important**

The Xcode Package iOS task has been deprecated. It is relevant only if you are using Xcode 6.4. Otherwise, use the [latest version of the Xcode task](#).

Requirements

Requirement	Description
Pipeline types	YAML, Classic build
Runs on	Agent, DeploymentGroup
Demands	Self-hosted agents must have capabilities that match the following demands to run jobs that use this task: xcode
Capabilities	This task does not satisfy any demands for subsequent tasks in the job.
Command restrictions	Any
Settable variables	Any
Agent version	All supported agent versions.
Task category	Build