

Azure DevOps: Pipeline



Pipelines



Pipelines



Environments



Releases



Library



Task groups



Deployment groups

Azure DevOps: Managing Pipeline

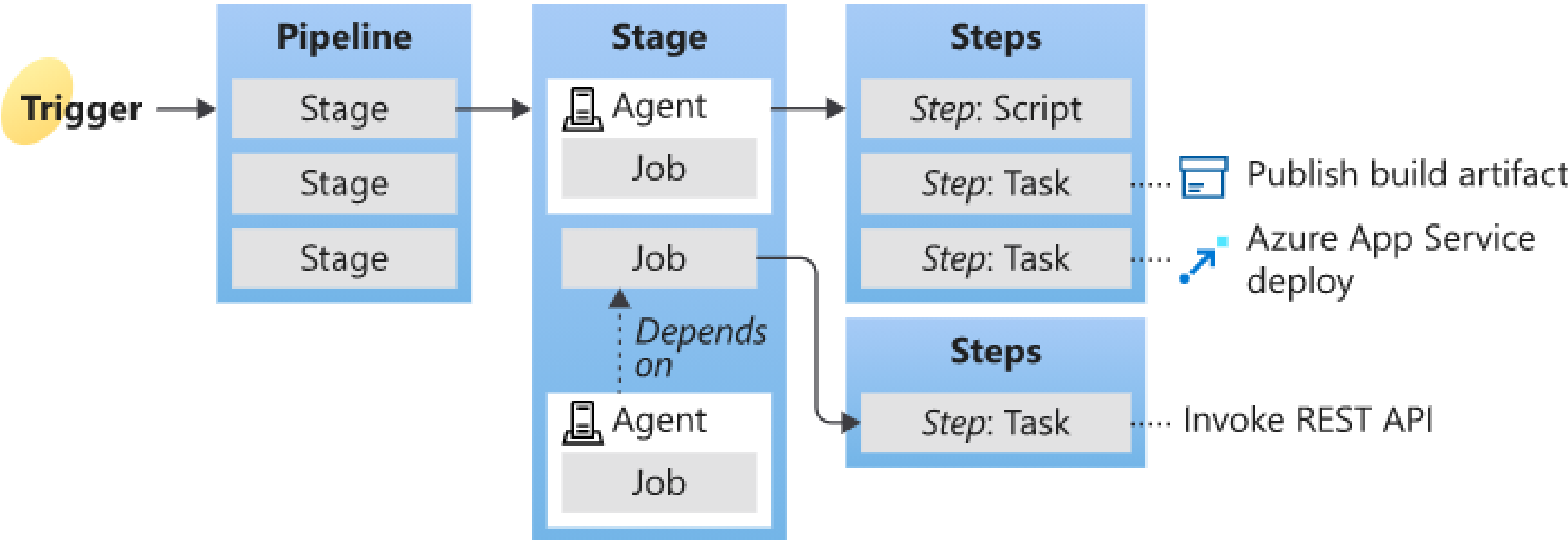
Key concepts: Azure Pipelines

Types of Pipeline



Build Pipeline
Release Pipeline

Key concepts overview



- A trigger tells a Pipeline to run.
- A pipeline is made up of one or more stages. A pipeline can deploy to one or more environments.
- A stage is a way of organizing jobs in a pipeline and each stage can have one or more jobs.
- Each job runs on one agent. A job can also be agentless.
- Each agent runs a job that contains one or more steps.
- A step can be a task or script and is the smallest building block of a pipeline.
- A task is a pre-packaged script that performs an action, such as invoking a REST API or publishing a build artifact.
- An artifact is a collection of files or packages published by a run.

Agent

When your build or deployment runs, the system begins one or more jobs. An agent is computing infrastructure with installed agent software that runs one job at a time. For example, your job could run on a Microsoft-hosted Ubuntu agent.

Approvals

Approvals define a set of validations required before a deployment runs. Manual approval is a common check performed to control deployments to production environments. When checks are configured on an environment, pipelines will stop before starting a stage that deploys to the environment until all the checks are completed successfully.

Artifact

An artifact is a collection of files or packages published by a run. Artifacts are made available to subsequent tasks, such as distribution or deployment. For more information, see [Artifacts in Azure Pipelines](#).

Continuous delivery

Continuous delivery (CD) is a process by which code is built, tested, and deployed to one or more test and production stages. Deploying and testing in multiple stages helps drive quality. Continuous integration systems produce deployable artifacts, which include infrastructure and apps. Automated release pipelines consume these artifacts to release new versions and fixes to existing systems. Monitoring and alerting systems run constantly to drive visibility into the entire CD process. This process ensures that errors are caught often and early.

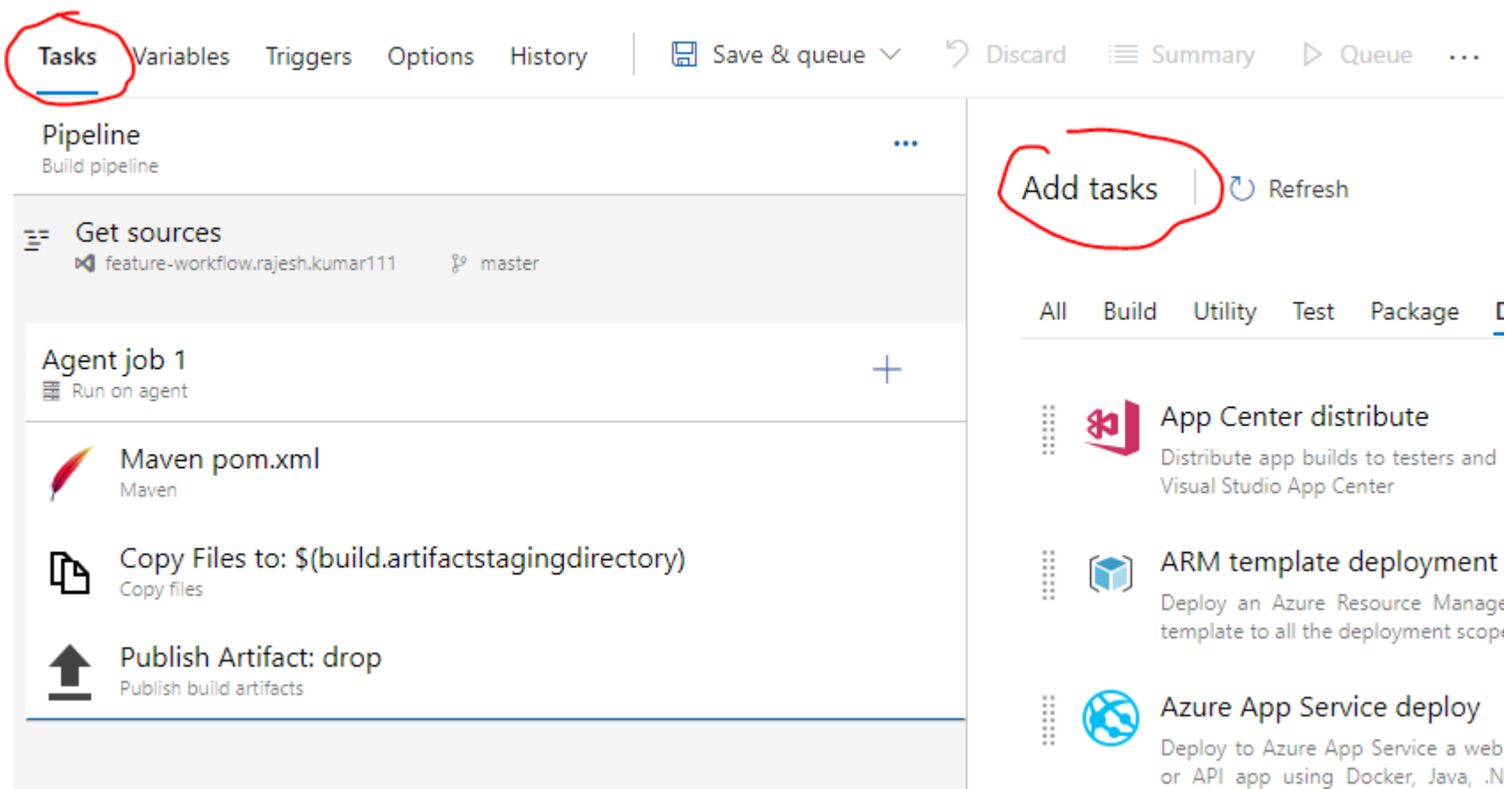
Continuous integration

Continuous integration (CI) is the practice used by development teams to simplify the testing and building of code. CI helps to catch bugs or problems early in the development cycle, which makes them easier and faster to fix. Automated tests and builds are run as part of the CI process. The process can run on a set schedule, whenever code is pushed, or both. Items known as artifacts are produced from CI systems. They're used by the continuous delivery release pipelines to drive automatic deployments.

Deployment

Classic pipelines - For Classic pipelines, a deployment is the action of running the tasks for one stage, which can include running automated tests, deploying build artifacts, and any other actions are specified for that stage.

YAML pipelines - For YAML pipelines, a deployment typically refers to a deployment job. **A deployment job is a collection of steps** that are run sequentially against an environment. You can use strategies like run once, rolling, and canary for deployment jobs.

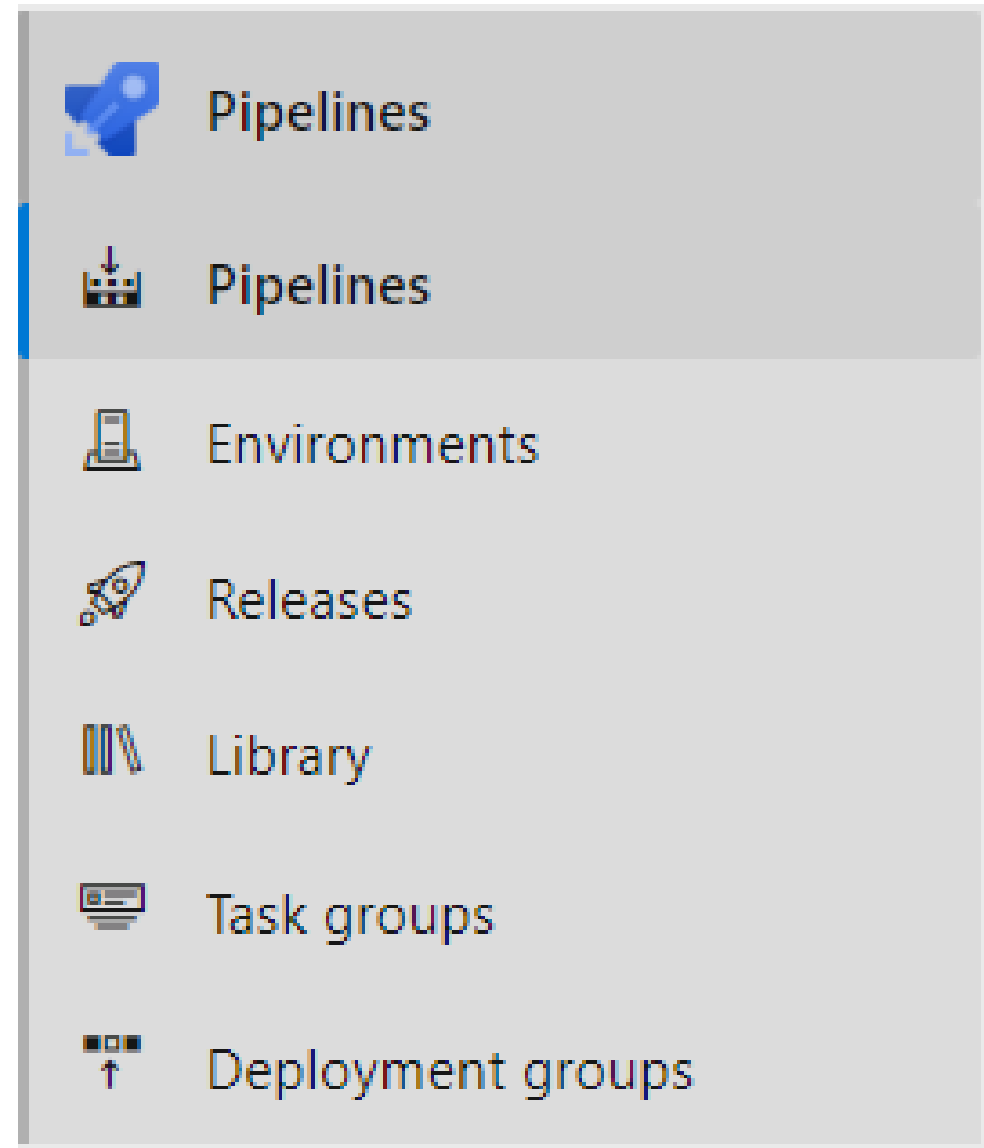


The screenshot shows the Azure DevOps Pipelines interface. The 'Tasks' tab is selected and circled in red. The pipeline is named 'Pipeline' and is a 'Build pipeline'. The 'Get sources' task is selected. The 'Agent job 1' is shown with a list of tasks: 'Maven pom.xml', 'Copy Files to: \$(build.artifactstagingdirectory)', and 'Publish Artifact: drop'. The 'Add tasks' button is circled in red. The right sidebar shows a list of tasks: 'App Center distribute', 'ARM template deployment', and 'Azure App Service deploy'.


```
16
17 task: Ant@1
18 inputs:
19   buildFile: 'build.xml'
20   options:
21     publishJUnitResults: true
22     testResultsFiles: '**/TEST-*.xml'
23     javaHomeOption: 'JDKVersion'
24
25 script: |
26   echo Add other tasks to build, test, and deploy your project.
27   echo See https://aka.ms/yaml
28   displayName: 'Run a multi-line script'
29
```


Environment


An environment is a collection of resources, where you deploy your application. It can contain one or more virtual machines, containers, web apps, or any service that's used to host the application being developed. A pipeline might deploy the app to one or more environments after build is completed and tests are run.





Releases – It's a Pipeline


 Learn
What's new...

 Pipeline, artifacts,
and stages

 Pre and post-deployment
approvals and gates

 Commits
and workitems

 In progress
deployments and logs

 Test results
& other extensions

We've made big improvements to the release summary page!



We've made it much easier to see exactly what's happening with all of your releases. Now you have a detailed summary of the release pipeline, plus one-click drill-down access to more details such as artifacts, stages, approvals, tests, and logs. Quickly see the work items, commits, test results, and much more.



Pipelines



Pipelines



Environments



Releases



Library



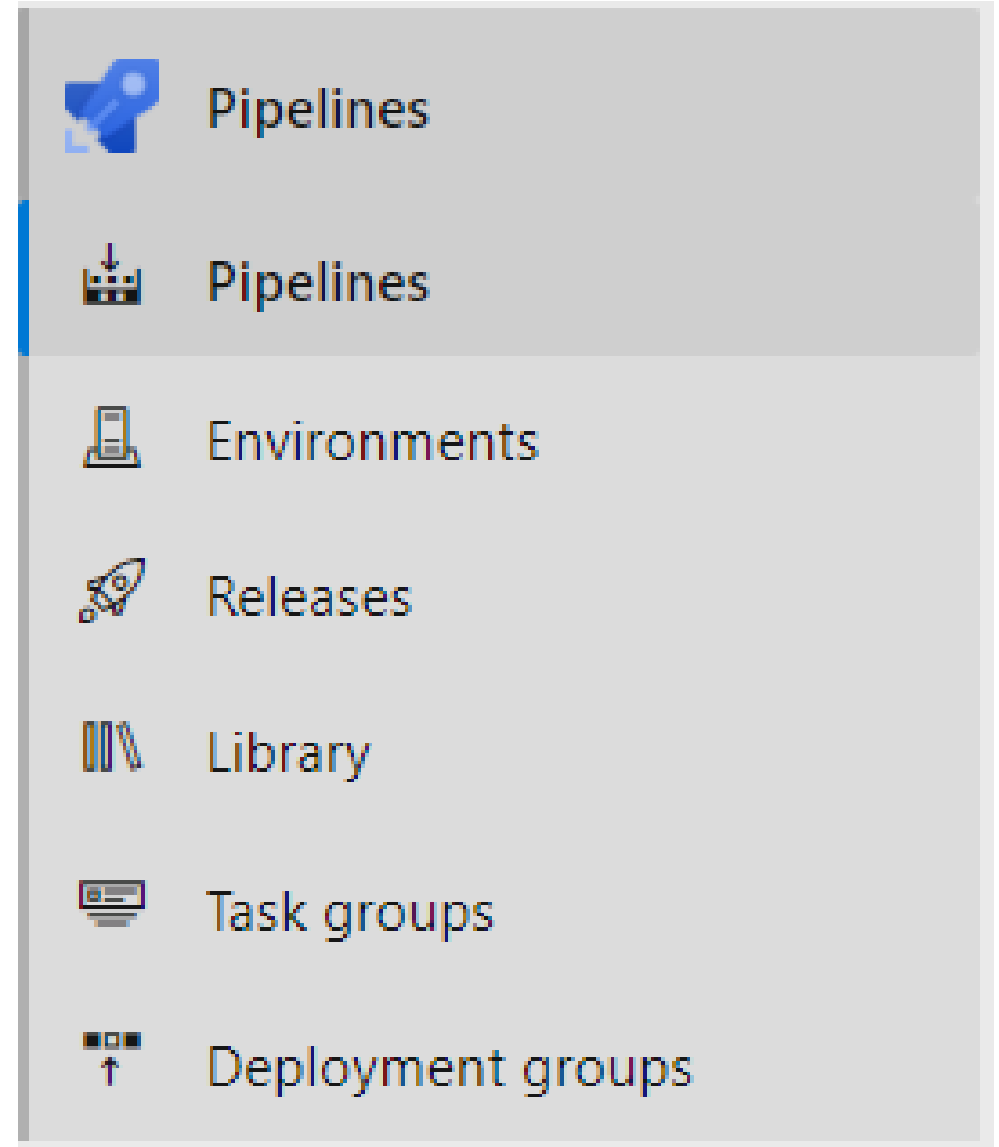
Task groups



Deployment groups

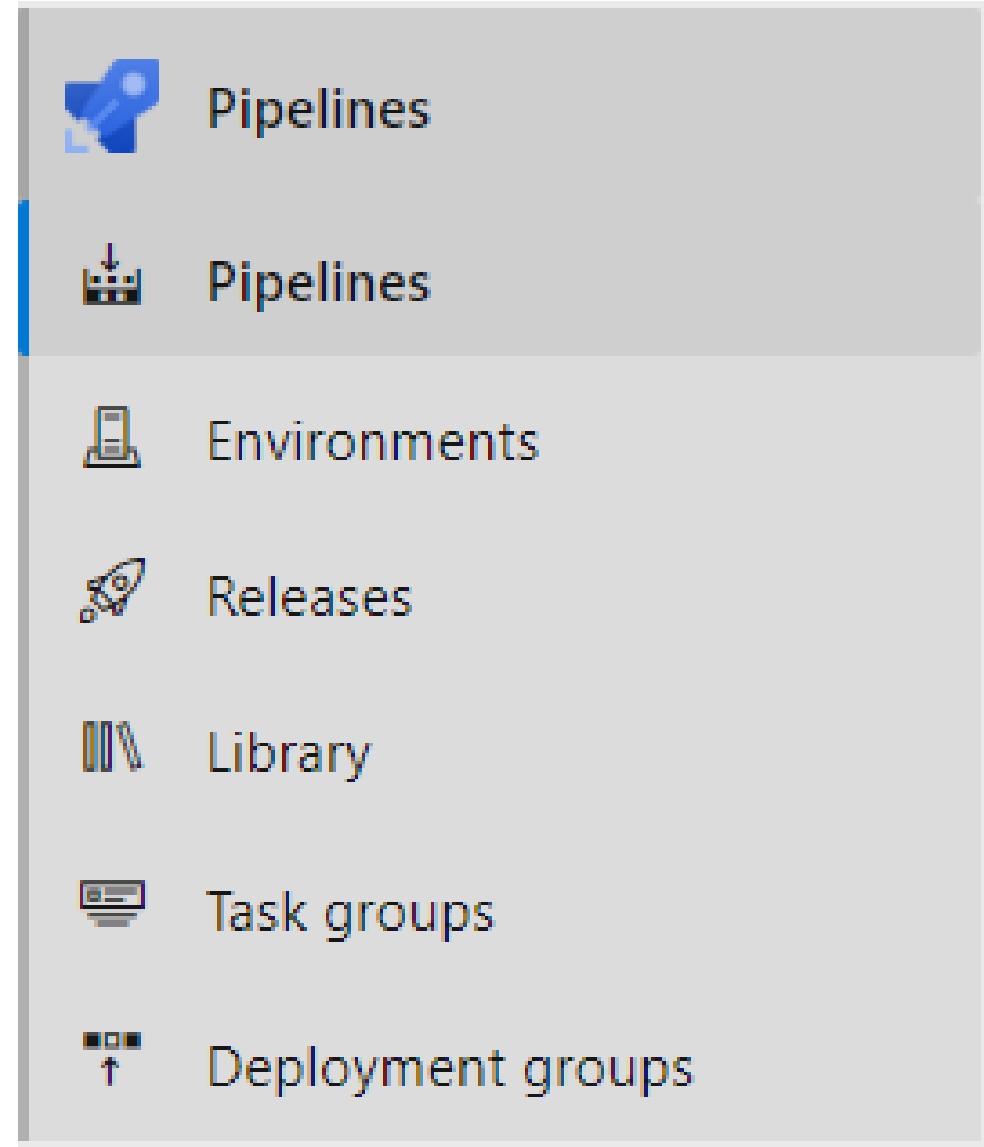
Library

A *library* is a collection of build and release assets for an Azure DevOps project. Assets defined in a library can be used in multiple build and release pipelines of the project. The **Library** tab can be accessed directly in Azure Pipelines.



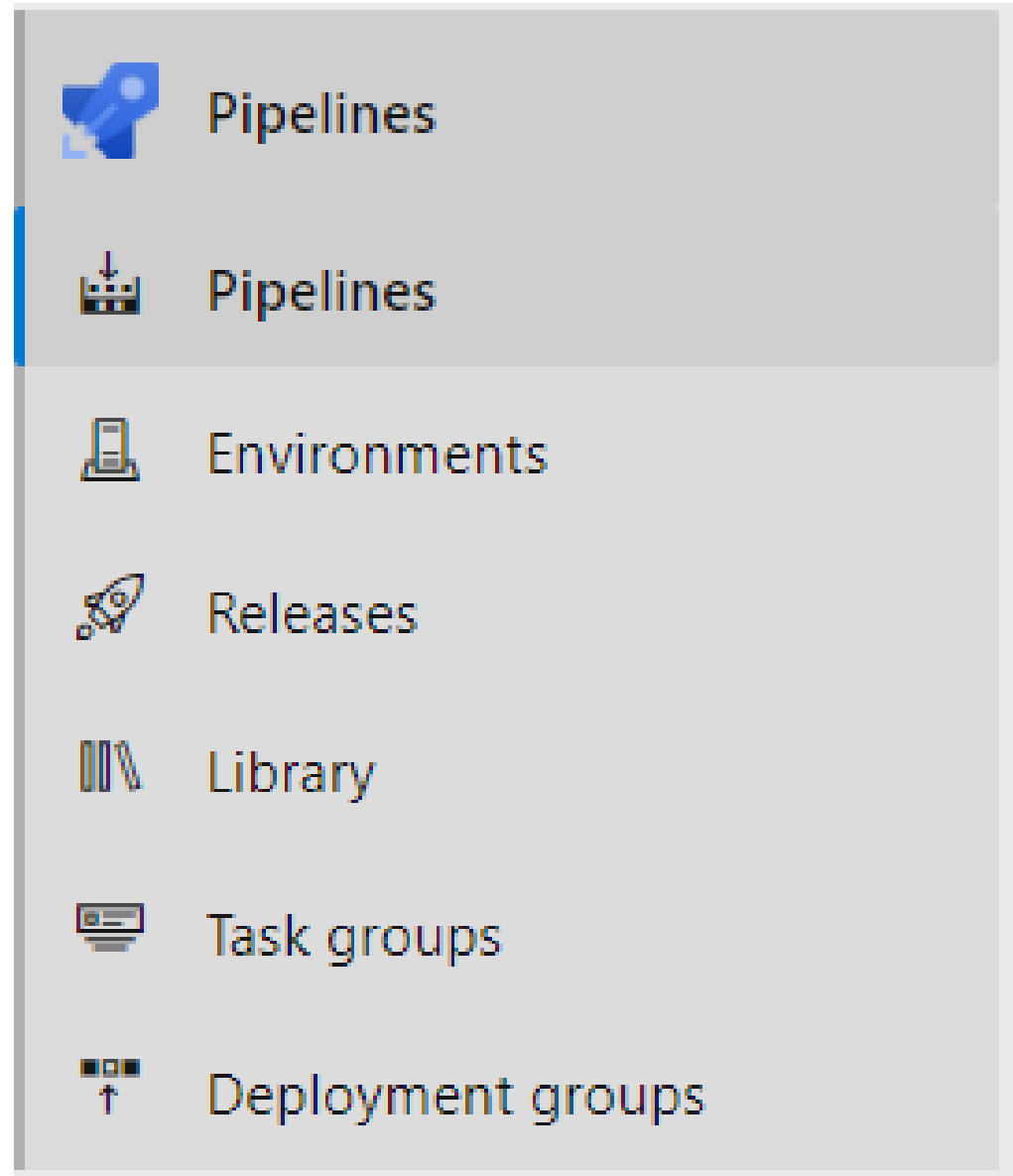
Task Group

A *task group* allows you to encapsulate a sequence of tasks, already defined in a build or a release pipeline, into a single reusable task that can be added to a build or release pipeline, just like any other task. You can choose to extract the parameters from the encapsulated tasks as configuration variables, and abstract the rest of the task information.



Deployment group

A deployment group is a set of deployment target machines that have agents installed. A deployment group is just another grouping of agents, like an agent pool. You can set the deployment targets in a pipeline for a job using a deployment group. Learn more about provisioning agents for deployment groups.



Types of Pipeline



Build Pipeline
Release Pipeline

How a Build Is Set Up



Build definition



Build Steps / Tasks




Build Agent




Pipeline




- Use the classic editor


Select the Build Source


 Azure DevOps


fluentbytes / BuildDemos


Search 


  


 BuildDemos


 Overview


 Boards


 Repos


 Pipelines


 Builds


 Releases


 Library

 Task groups

 Deployment groups

 Test Plans


 Artifacts





Select your repository


Tell us where your sources are.
You can customize how to get these sources from the repository later.


Select a source


 Azure Repos Git


 TFVC

 GitHub


 GitHub Enterprise

 Subversion

 Bitbucket Cloud


 External Git

Team project

 BuildDemos


▼

Repository

 BuildDemos

▼

Default branch for manual and scheduled builds

 master

▼

Continue

Select the Build Template

Azure DevOps

fluentbytes / BuildDemos

Search

BuildDemos

Overview

Boards

Repos

Pipelines

Builds

Releases

Library

Task groups

Deployment groups

Test Plans

Artifacts

Choose a template

Choose a template that builds your kind of app.
Don't worry if it's not an exact match;
you can add and customize the tasks later.

Select a template

Or start with an [Empty job](#)

Search

Configuration as code

YAML
Looking for a better experience to configure your pipelines using YAML files?
Try the new YAML pipeline creation experience. [Learn more](#)

Featured

.NET Desktop
Build and test a .NET or Windows classic desktop solution.

Android
Build, test, sign, and align an Android APK.

ASP.NET
Build and test an ASP.NET web application.

Azure Web App for ASP.NET
Build, package, test, and deploy an ASP.NET Azure Web App.

Docker container
Build a Docker image and push it to a container registry.

Maven
Build and test a Java project with Apache Maven.

Python package
Create and test a Python package on multiple Python versions.

Xcode
Build, test, archive, or package an Xcode workspace on macOS.

Configure the Build Tasks

Azure DevOps fluentbytes / BuildDemos

BuildDemos-ASP.NET with containers-CI

Tasks Variables Triggers Options Retention History Save & queue Discard Summary Queue

Pipeline
Build pipeline

Get sources
BuildDemos master

Agent job 1
Run on agent

- Use NuGet 4.4.1**
NuGet Tool Installer
- NuGet restore**
NuGet
- Build solution ***.sln**
Visual Studio Build
- Build services**
Docker Compose
- Push services**
Docker Compose
- Lock services**
Docker Compose
- Copy Files to: \$(Build.ArtifactStagingDirectory)**
Copy Files
- Publish Artifact: docker-compose**
Publish Build Artifacts

Name *
BuildDemos-ASP.NET with containers-CI

Agent pool * | Pool information | Manage
Hosted VS2017

Parameters | Unlink all

Solution *
***.sln

Docker Compose File *
**/docker-compose.yml

Azure subscription | Manage

Azure Container Registry

Select the Build Agent

The screenshot displays the Azure DevOps web interface. On the left is a navigation sidebar with sections: BuildDemos, Overview, Boards, Repos, Pipelines (containing Builds, Releases, Library, Task groups, Deployment groups), Test Plans, and Artifacts. The main area shows the 'BuildDemos-ASP.NET with containers-CI' pipeline. The 'Pipeline' tab is active, showing a list of tasks: 'Get sources', 'Agent job 1', 'Use NuGet 4.4.1', 'NuGet restore', 'Build solution ***.sln', 'Build services', 'Push services', 'Lock services', 'Copy Files to: \$(Build.ArtifactStagingDirectory)', and 'Publish Artifact: docker-compose'. The 'Agent job 1' task is selected, and its configuration is shown on the right. The 'Agent pool' dropdown is open, displaying a list of available agents. The 'Hosted VS2017' agent is highlighted with a red rectangular box. Other agents in the list include 'Hosted', 'Hosted macOS', 'Hosted Ubuntu 1604', and 'Hosted Windows Container'. The 'Private' section shows 'Default' and 'DemoPool'.

Agent pool configuration details:

Agent Pool	Icon
Hosted	Windows icon
Hosted macOS	Apple icon
Hosted Ubuntu 1604	Linux icon
Hosted VS2017	Visual Studio icon
Hosted Windows Container	Windows icon
Private	
Default	Default icon
DemoPool	Default icon

Demo



Configure and run an ASP.NET build

Build Infrastructure

Agents and Pipelines



Hosted Build Agent



Pipelines



Custom Build Agent

Build Security

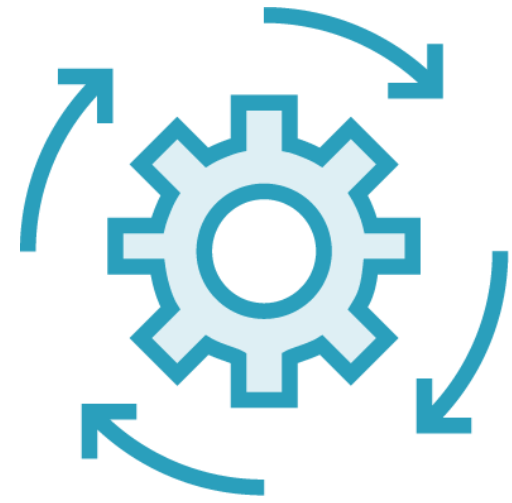
Pools & Queues



(Hosted) Agent



Agent Queues



Agent Pools

Demo



Setting up a custom agent and security

Setting up Continuous Integration Builds

Demo



Continuous integration with Azure
DevOps and GitHub

Configuring More Specialized Builds

Outline



Build details

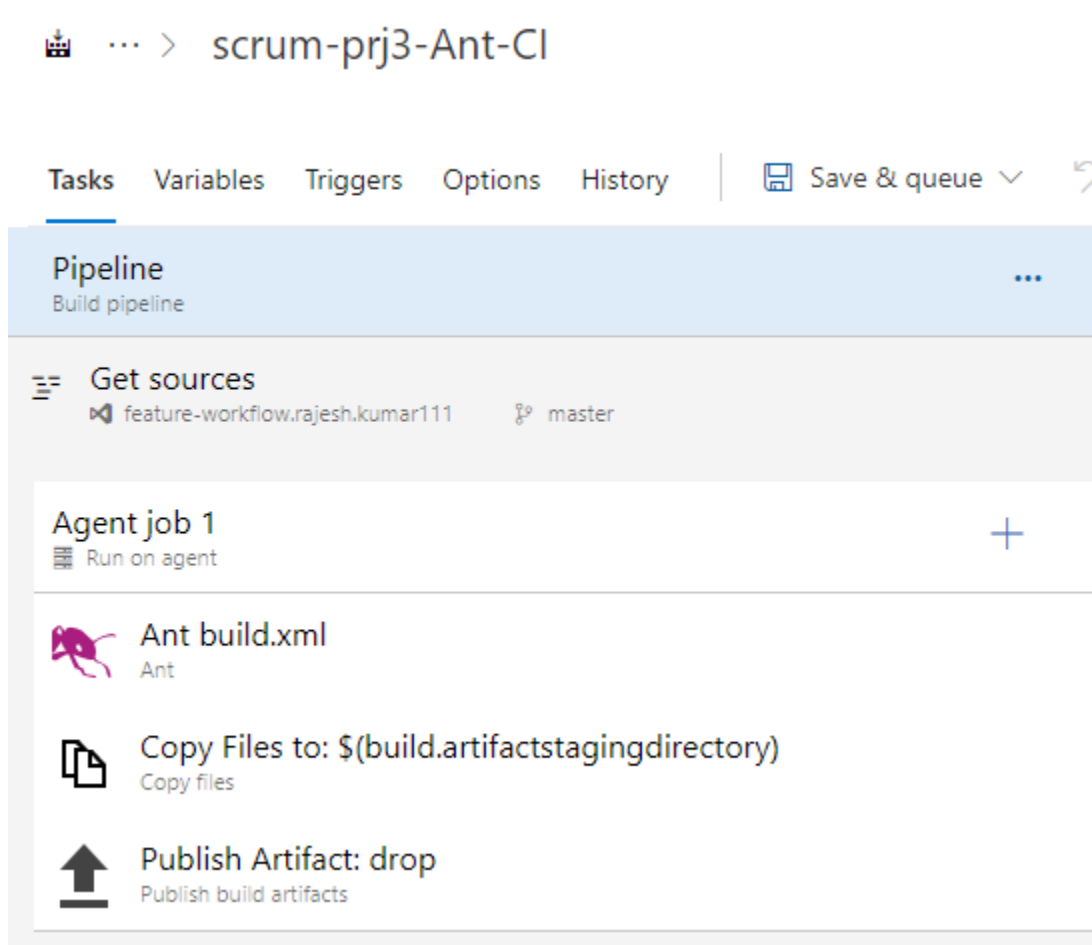
Tasks and the market place

Optimize your builds

Yaml builds

Build Details

Build Variables



Custom Variables

`$(variablename)`

Build In Variables

Variables From PowerShell

Secrets

Environment Variable

Build Triggers

The screenshot shows the Jenkins web interface for a pipeline named 'scrum-prj3-Ant-CI'. The top navigation bar includes tabs for 'Tasks', 'Variables', 'Triggers', 'Options', and 'History', along with a 'Save & queue' button. The 'Pipeline' tab is selected, showing a 'Build pipeline' button. Below this, the 'Get sources' step is configured with the repository 'feature-workflow.rajesh.kumar111' and the branch 'master'. An 'Agent job 1' is listed with the option to 'Run on agent'. The pipeline steps are: 'Ant build.xml' (Ant), 'Copy Files to: \$(build.artifactstagingdirectory)' (Copy files), and 'Publish Artifact: drop' (Publish build artifacts).

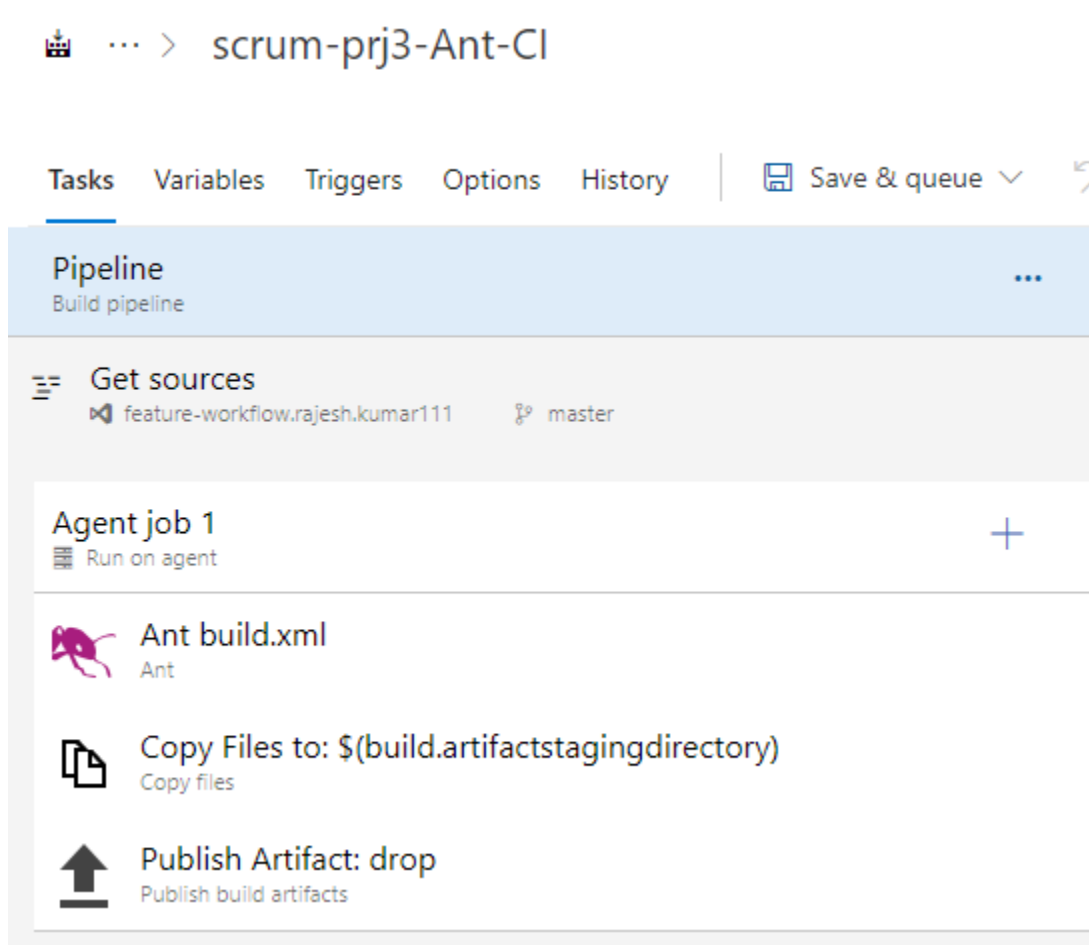
Continuous Integration

Branch Filters

Pull Request

Gated Check-in

Build Options



Build Job Properties

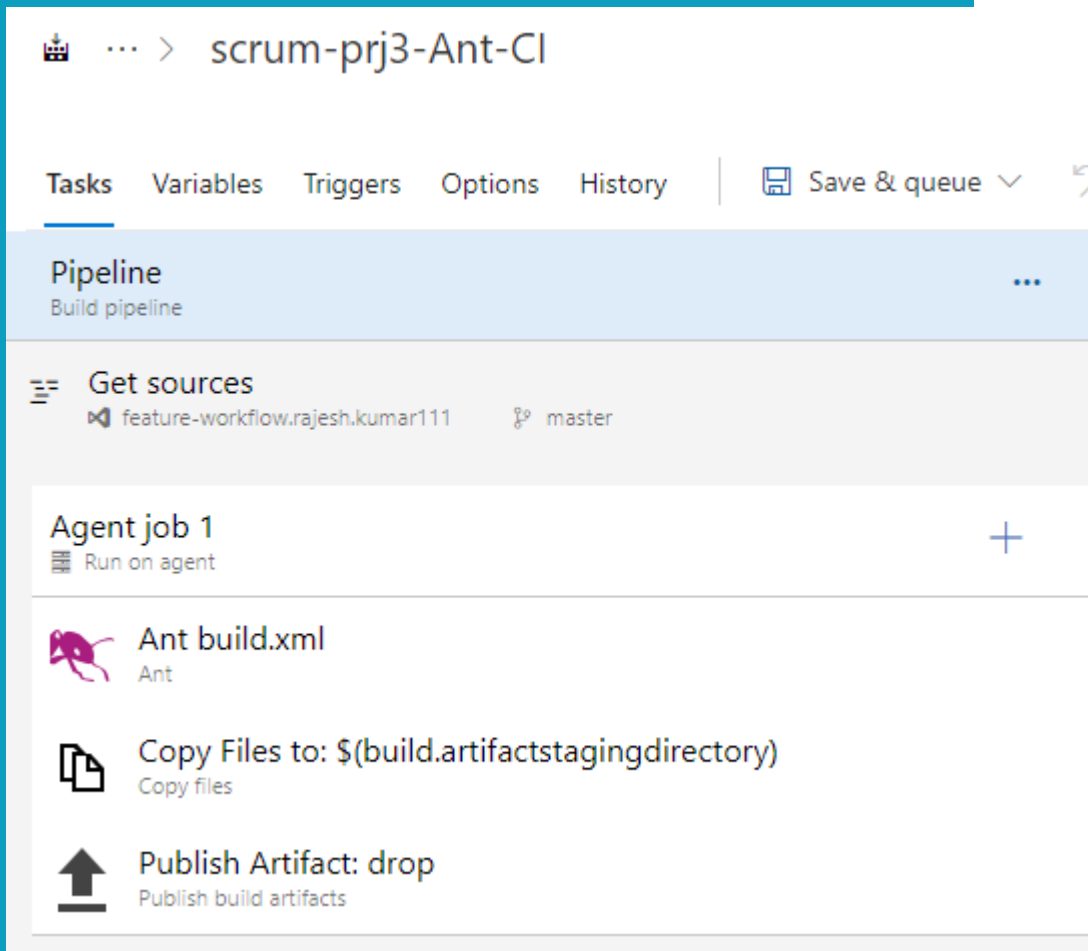
Demands

Build Number Format

Work-Items

Oauth Token

Build Retention & History



Retention Policy
Change History

Tasks and the Market Place

The screenshot displays the 'Add tasks' interface in Azure DevOps. At the top, there is a navigation bar with buttons: 'History', 'Save & queue' (with a dropdown arrow), 'Discard', 'Summary', 'Queue', and a menu icon. Below this, the 'Add tasks' section is visible, featuring a 'Refresh' button and a search bar. A horizontal tab bar contains the following categories: 'All', 'Build', 'Utility', 'Test', 'Package', 'Deploy', 'Tools', and 'Marketplace'. The 'Marketplace' tab is currently selected and is circled in red. Below the tabs, a list of tasks is shown, each with an icon, a title, and a description:

- SARIF SAST Scans Tab**: Adds a 'Scans' tab to each Build Result and Work Item for viewing associated SARIF SAST logs.
- Replace Tokens**: Task to replace tokens in files.
- SonarQube**: Detect bugs, vulnerabilities and code smells across project branches and pull requests.
- ARM Outputs**: (Description partially obscured)

On the left side of the interface, a sidebar shows a list of items, including 'master' and 'stagingdirectory)', with a plus sign indicating more items.

Demo



More build details for various builds

Optimize Your Builds

Fast Feedback Starts with Fast Build Server(s)



Configuration optimized for the task at hand:

- Fast IO (local disks, SSD preferred)
- Fast CPU

Located 'near' the sources and the drop location

Different Builds for Different Purposes

CI Build

(Continuous Integration)



Compile/test

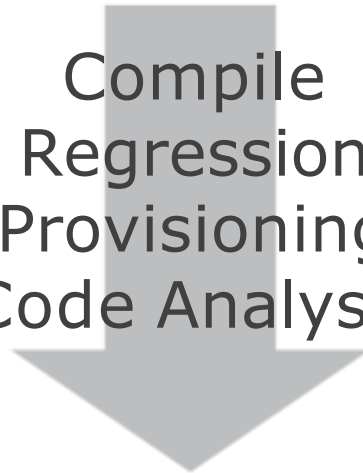


Nightly build

(schedule)



Compile
Regression
Provisioning
Code Analysis



Release build

(manual)



Compile
Test
Regression
nfig management
Archive



Co

Optimizing the Build



Enable parallel build execution for faster results

Enable parallel test execution for faster results

Using multipliers to scale builds over multiple agents

Make integration tests part of the release process instead of the build

Publish to a NuGet feed at the end of a build

Demo



Optimizing the build to run parallel

Configuration and Infrastructure as Code

Outline



Configuration as code

Transform configuration

Infrastructure as code

Artifact location

Configuration as Code

Configuration as Code



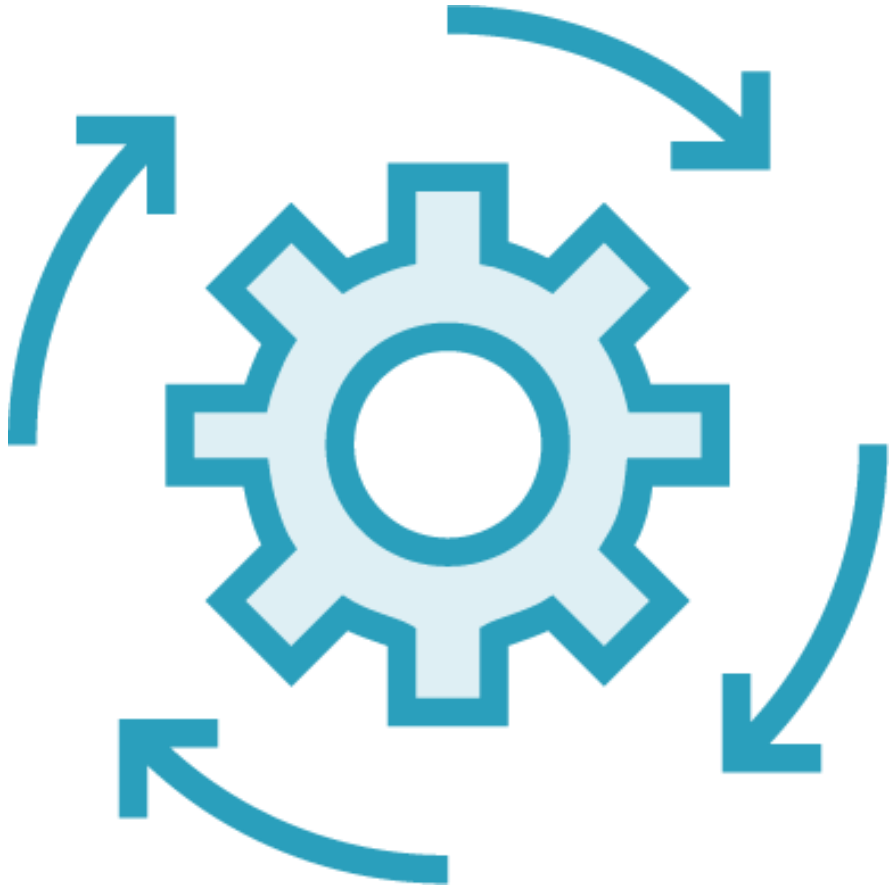
Important Continuous
Delivery concept

Keep configuration in
Source Control

The build outputs the
needed artifacts

Transform Configuration

Configuration and Secrets



Have admin define secrets in variables

Use the build to replace secrets

Use transform tasks

Infrastructure as Code

Important Infrastructure Artifacts



PowerShell Scripts



PowerShell DSC
scripts



ARM Templates



Bash scripts



Puppet modules

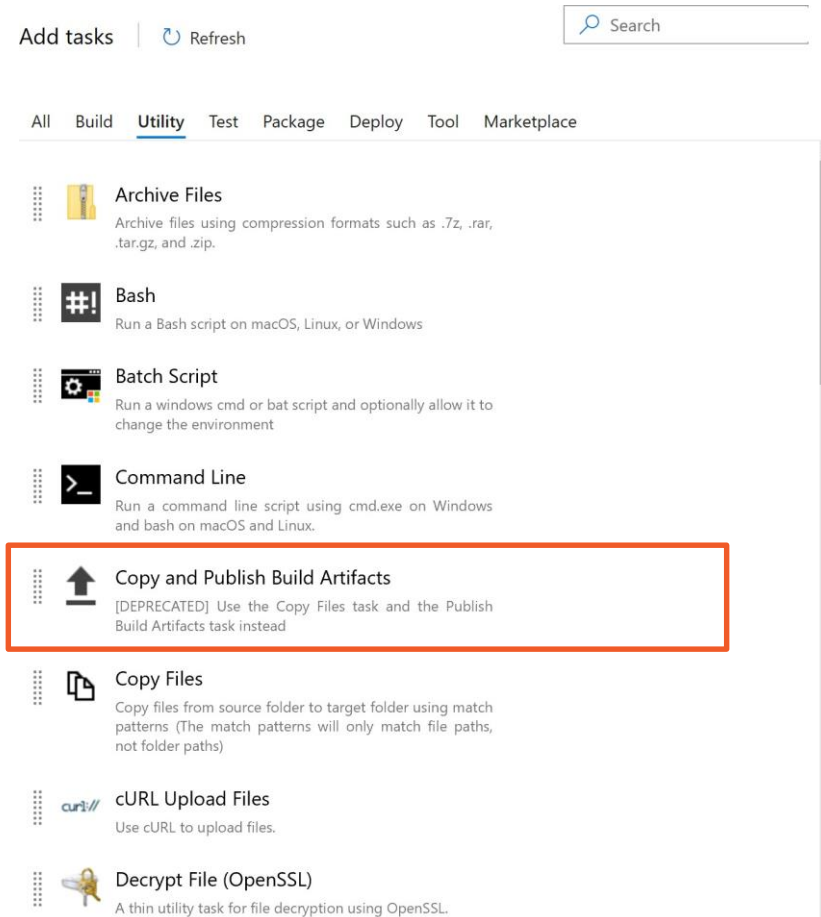


CHEF™

Chef recipes

Artifact Location

Artifacts and the Artifact Store



Azure DevOps has an artifact location buildin

You can copy your artifacts to the artifact repository

This is where you will pull them from when we are using release pipelines in a later stage

Demo



Adding configuration and infrastructure
as code to your build



Pipeline

- Use the Pipeline YAML

This hierarchy is reflected in the structure of a YAML file like:

- Pipeline
 - Stage A
 - Job 1
 - Step 1.1
 - Step 1.2
 - ...
 - Job 2
 - Step 2.1
 - Step 2.2
 - ...
 - Stage B
 - ...

YAML



```
name: $(Date:yyyyMMdd)$(Rev:.r)
variables:
  var1: value1
jobs:
- job: One
  steps:
  - script: echo First step!
```

```
stages:
- stage: Build
  jobs:
  - job: BuildJob
    steps:
    - script: echo Building!
- stage: Test
  jobs:
  - job: TestOnWindows
    steps:
    - script: echo Testing on Windows!
  - job: TestOnLinux
    steps:
    - script: echo Testing on Linux!
- stage: Deploy
  jobs:
  - job: Deploy
    steps:
    - script: echo Deploying the code!
```

YAML

 Copy

```
jobs:
- job: MyJob
  displayName: My First Job
  continueOnError: true
  workspace:
    clean: outputs
  steps:
  - script: echo My first job
```

<https://docs.microsoft.com/en-us/azure/devops/pipelines/yaml-schema?view=azure-devops&tabs=schema%2Cparameter-schema>