

Agenda: Continuous Deployment using Azure Pipelines

- Deploying App to App Service using YAML
- Add the deployment State to the pipeline
- Deploy Apps to Specific Environment
- Deploy Azure Functions

Deploy ASP.NET Core App to Azure App Service using YAML**Add the deployment stage to the pipeline:**

Following step will be added to the YAML file. (Edit WebAppName and use variable created earlier)

```
- task: AzureRmWebAppDeployment@4
  inputs:
    ConnectionType: 'AzureRM'
    azureSubscription: 'AzureDevOpsServicePrincipal'
    appType: 'webAppLinux'
    WebAppName: $(WebAppName)
    packageForLinux: '$(build.artifactstagingdirectory)/**/*.zip'
```

To deploy a build artifact from the pipeline, you need a way to download it from the pipeline to the agent. You use the **DownloadPipelineArtifact@2** task to download artifacts.

This task requires a few inputs. The ones we need here are:

- **buildType**, which specifies whether we want the artifacts from the current build or a specific build. For now, we want to deploy the current build.
- **artifact**, which specifies the name of the artifact to download. We need this input to specify the name of the .zip file.

```
- task: DownloadPipelineArtifact@2
  inputs:
    buildType: current
    artifact: 'drop'
```

OR (The download task is a shortcut for the DownloadPipelineArtifact@2 task).

```
- download: current
  artifact: drop
```

Multi-Stage YAML Pipeline with Jobs (not deployment jobs) – Doesn't support Approvals and Checks...

```
trigger:
- none
```

```
stages:
- stage: "Build"
  jobs:
  - job: "BuildJob"
    pool:
      vmImage: 'windows-latest'

    variables:
      buildConfiguration: 'Release'

    steps:
    - task: DotNetCoreCLI@2
      displayName: "Restore Project"
      inputs:
        command: 'restore'
        projects: '**/*.csproj'

    - task: DotNetCoreCLI@2
      displayName : "Build Project"
      inputs:
        command: 'build'
        projects: '**/*.csproj'
        arguments: '-c $(buildConfiguration)'

    - task: DotNetCoreCLI@2
      displayName : "Build Project"
      inputs:
        command: 'publish'
        publishWebProjects: true
        arguments: '-c $(buildConfiguration) -o $(build.artifactstagingdirectory)'

    - task: PublishBuildArtifacts@1
      displayName: 'Publish Artifact'
      inputs:
        PathtoPublish: '$(build.artifactstagingdirectory)'
```

2

```
- stage: "DeployToDev"
  dependsOn: "Build"
  jobs:
  - job:
    pool:
      vmImage: "windows-latest"
    steps:
    - download: current
      artifact: drop

    - task: AzureRmWebAppDeployment@4
      inputs:
        ConnectionType: 'AzureRM'
        azureSubscription: 'Azure Connection'
        appType: 'webApp'
        WebAppName: 'dssdemo1-dev'
        packageForLinux: '$(Pipeline.Workspace)/**/*.zip'
```

```
- stage: "DeployToQA"
  dependsOn: "DeployToDev"
  jobs:
  - job:
    pool:
      vmImage: "windows-latest"
    steps:
    - download: current
      artifact: drop

    - task: AzureRmWebAppDeployment@4
      inputs:
        ConnectionType: 'AzureRM'
        azureSubscription: 'Azure Connection'
        appType: 'webApp'
        WebAppName: 'dssdemo1-qa'
        packageForLinux: '$(Pipeline.Workspace)/**/*.zip'
```

```
- stage: "DeployToProd"
  dependsOn: "DeployToQA"
  jobs:
  - job:
    pool:
      vmImage: "windows-latest"
    steps:
    - download: current
      artifact: drop
    - task: AzureRmWebAppDeployment@4
      inputs:
        ConnectionType: 'AzureRM'
        azureSubscription: 'Azure Connection'
        appType: 'webApp'
        WebAppName: 'dssdemo1-prod'
        deployToSlotOrASE: true
        ResourceGroupName: 'DemoRG'
        SlotName: 'staging'
        packageForLinux: '$(Pipeline.Workspace)/**/*.zip'
    - task: AzureAppServiceManage@0
      inputs:
        azureSubscription: 'Azure Connection'
        Action: 'Swap Slots'
        WebAppName: 'dssdemo1-prod'
        ResourceGroupName: 'DemoRG'
        SourceSlot: 'staging'
```

Deploy a web application to App Service with Approval using Environment

What is an environment?

You've likely used the term *environment* to refer to **where your application or service is running**. It represents a **collection of resources** such as namespaces within Kubernetes clusters, Azure Web Apps, virtual machines, databases, which can be targeted by deployments from a pipeline.

Typical examples of environments include *Dev, Test, QA, Staging and Production*.

The advantages of using environments include the following.

- **Deployment history** - Pipeline name and run details are recorded for deployments to an environment and its resources. In the context of multiple pipelines targeting the same environment or resource, [deployment history](#) of an environment is useful to identify the source of changes.
- **Traceability of commits and work items** - View jobs within the pipeline run that targeted an environment and the corresponding [commits and work items](#) that were newly deployed to the environment. This allows one to track whether a code change (commit) or feature/bug-fix (work items) reached an environment.
- **Permissions** - User permissions and pipeline permissions can be used to secure environments by specifying which users and pipelines are allowed to target an environment.

Resources:

While environment at its core is a grouping of resources, the resources themselves represent actual deployment targets. The **Kubernetes resource** and **virtual machine resource** types are currently supported.

Target Environment from a deployment job

A deployment job can be used to target an entire environment (group of resources) as shown in the following **Sample** YAML snippet.

```
- stage: deploy
  jobs:
  - deployment: DeployWeb
    displayName: Deploy Web App
    pool:
      vmImage: 'Ubuntu-latest'
    # creates an environment if it doesn't exist
    environment: 'demo-dev'
    strategy:
```

Target a specific resource within an environment

```
environment: 'dssdemoapp-dev.WebServer1'
```

Security:

User permissions: Set appropriate user permissions to ensure that the users are pre-authorized to define a pipeline that targets the environment.

1. Environment → Click  on top right → Security

2. In the **User permissions** blade, click on **+Add** to add a **User or group** and select a suitable **Role**.

Pipeline permissions: Pipeline permissions can be used to authorize either all or specific pipelines for deploying to the environment of concern.

Approvals

You can manually control when a stage should run using approval checks. This is commonly used to control deployments to production environments.

Checks are a mechanism available to the *resource owner* to control if and when a stage in a pipeline can consume a resource. As an owner of a resource, such as an environment, you can define checks that must be satisfied before a stage consuming that resource can start.

Currently, manual approval checks are supported on environments.

```
trigger:
- none

stages:
- stage: 'BuildStage'
  jobs:
  - job: 'BuildJob'
    pool:
      vmImage: windows-latest
    variables:
      buildConfiguration: 'Release'
    steps:
    - task: DotNetCoreCLI@2
      displayName: "Restore"
      inputs:
        command: 'restore'
        projects: '**/*.csproj'
        feedsToUse: 'select'
    - task: DotNetCoreCLI@2
      displayName: "Build"
      inputs:
        command: 'build'
        projects: '**/*.csproj'
    - task: DotNetCoreCLI@2
```

```
    displayName: "Publish"
    inputs:
      command: 'publish'
      publishWebProjects: true
      arguments: '--output $(Build.ArtifactStagingDirectory) --configuration $(buildConfiguration)'
- task: PublishPipelineArtifact@1
  inputs:
    targetPath: '$(Build.ArtifactStagingDirectory)'
    artifact: 'drop'
    publishLocation: 'pipeline'

- stage: 'DeploytoDev'
  dependsOn: 'BuildStage'
  jobs:
    - deployment:
      pool:
        vmImage: 'ubuntu-latest'
      environment: Development
      strategy:
        runOnce:
          deploy:
            steps:
              # - task: DownloadPipelineArtifact@2
              # inputs:
              #   buildType: 'current'
              #   artifactName: 'drop'
              #   targetPath: '$(Pipeline.Workspace)'
            - task: AzureRmWebAppDeployment@4
              inputs:
                ConnectionType: 'AzureRM'
                azureSubscription: 'VS Subscription - SS1 - Automatic'
                appType: 'webApp'
                WebAppName: 'dsdemoapp-dev'
                packageForLinux: '$(Pipeline.Workspace)/**/*.zip'

- stage: 'DeploytoQA'
```

```
dependsOn: 'DeploytoDev'
jobs:
- deployment:
  environment: QA
  pool:
    vmImage: 'ubuntu-latest'
  strategy:
    runOnce:
      deploy:
        steps:
          # - task: DownloadPipelineArtifact@2
          #   inputs:
          #     buildType: 'current'
          #     artifactName: 'drop'
          #     targetPath: '$(Pipeline.Workspace)'
          - task: AzureRmWebAppDeployment@4
            inputs:
              ConnectionType: 'AzureRM'
              azureSubscription: 'VS Subscription - SS1 - Automatic'
              appType: 'webApp'
              WebAppName: 'dsdemoapp-qa'
              packageForLinux: '$(Pipeline.Workspace)/**/*.zip'

- stage: 'DeploytoProd'
  dependsOn: 'DeploytoQA'
  jobs:
  - deployment:
    environment: Production
    pool:
      vmImage: 'ubuntu-latest'
    strategy:
      runOnce:
        deploy:
          steps:
            # - task: DownloadPipelineArtifact@2
            #   inputs:
```



```
# buildType: 'current'
# artifactName: 'drop'
# targetPath: '$(Pipeline.Workspace)'
- task: AzureRmWebAppDeployment@4
  inputs:
    ConnectionType: 'AzureRM'
    azureSubscription: 'VS Subscription - SS1 - Automatic'
    appType: 'webApp'
    WebAppName: 'dsdemoapp-prod'
    deployToSlotOrASE: true
    ResourceGroupName: 'Demo-rg'
    SlotName: 'staging'
    packageForLinux: '$(Pipeline.Workspace)/**/*.zip'
- task: AzureAppServiceManage@0
  inputs:
    azureSubscription: 'VS Subscription - SS1 - Automatic'
    Action: 'Swap Slots'
    WebAppName: 'dsdemoapp-prod'
    ResourceGroupName: 'Demo-rg'
    SourceSlot: 'staging'
```

Note: Deployment Job doesn't need to have download task as it is builtin by default.

Use the below Task for Application Settings:

```
- task: AzureAppServiceSettings@1
  inputs:
    azureSubscription: 'Azure Connection'
    appName: 'dsshellowebapp'
    resourceGroupName: 'Demo-rg'
    slotName: 'staging'
    appSettings: |
    [
      {
        "name": "key1",
        "value": "valueabcd",
        "slotSetting": false
      },
    ]
```

```
{
  "name": "key2",
  "value": "valueefgh",
  "slotSetting": true
}
```

Deploy Apps Virtual Machine

Demo - Environment - Virtual Machine Resource

Step1: Create an Environment

1. Azure DevOps Organization → Project → Pipelines → Environments → New Environment
2. Name = "WebServers", Select Radio Button **Virtual Machines** → **Next**
3. **Virtual machine resource:** Choose **Operating System = Windows**
4. **Copy PS registration script.**

Note: It is possible to create an empty environment and reference the same from deployment jobs to record the deployment history against the environment.

Step2: Install Pipeline Agent on VM

5. **RDP to VM** → Open Powershell in Administrator mode
6. Paste the PS Script copied in previous step
 - a) Tag = Web
 - b) User as "dssadmin" (User which has administrator priviledge)

Repeat Step2 for all the VM's you want to Deploy the Application.

Step3: Create a New Pipeline with following YAML

```
trigger:
- '*'
variables:
  buildConfiguration: 'Release'
stages:
- stage: 'Build'
  displayName: 'Build the web application'
  jobs:
  - job: Build
```

```
displayName: 'Build Job'

pool:
  vmImage: 'ubuntu-16.04'

variables:
  wwwrootDir: 'HelloWorldApp.Web/wwwroot'
  dotnetSdkVersion: '3.1.x'

steps:
- task: DotNetCoreCLI@2
  displayName: 'Restore project dependencies'
  inputs:
    command: 'restore'
    projects: '**/*.csproj'

- task: DotNetCoreCLI@2
  displayName: 'Build the project - $(buildConfiguration)'
  inputs:
    command: 'build'
    arguments: '--no-restore --configuration $(buildConfiguration)'
    projects: '**/*.csproj'

- task: DotNetCoreCLI@2
  displayName: 'Publish the project - $(buildConfiguration)'
  inputs:
    command: 'publish'
    projects: '**/*.csproj'
    publishWebProjects: false
    arguments: '--no-build --configuration $(buildConfiguration) --
output $(Build.ArtifactStagingDirectory)/$(buildConfiguration)'
    zipAfterPublish: true

- task: PublishBuildArtifacts@1
  displayName: 'Publish Artifact: drop'
  condition: succeeded()

- stage: 'Deploy'
```

```
displayName: 'Deploy the web application'
dependsOn: Build
jobs:
- deployment: DeployToVM
  pool:
    vmImage: 'windows-latest'
  environment:
    name: WebServers
    resourceType: VirtualMachine
    tags: web
  strategy:
    runOnce:
      deploy:
        steps:
        - task: IISWebAppManagementOnMachineGroup@0
          inputs:
            EnableIIS: true
            IISDeploymentType: 'IISWebsite'
            ActionIISWebsite: 'CreateOrUpdateWebsite'
            WebsiteName: 'default web site'
            WebsitePhysicalPath: '%SystemDrive%\inetpub\wwwroot'
            WebsitePhysicalPathAuth: 'WebsiteUserPassThrough'
            AddBinding: true
            Protocol: http
            IPAddress: 'All Unassigned'
            port: 80
        - task: IISWebAppDeploymentOnMachineGroup@0
          inputs:
            WebSiteName: 'default web site'
            Package: '$(Pipeline.Workspace)/**/*.zip'
            TakeAppOfflineFlag: true
```

7. Save and Run the Pipeline
8. **View History:** Environment → Click WebServers → Deployments

YAML for Deploying to respective Environment Based on Which Branch Build was Performed

```
trigger:
- master
- development
- release
- hotfix

variables:
  AzureConnection: 'Azure Pass 2'

stages:
- stage: "Build_Stage"
  displayName: "Building the web application"
  jobs:
  - job: 'Build_Job'
    displayName: "Job: Building the application"
    pool:
      vmImage: 'ubuntu-latest'
    variables:
      buildConfiguration: 'Release'
      ProjectName: '**/*.csproj'
    steps:
    - task: DotNetCoreCLI@2
      displayName: Restore
      inputs:
        command: 'restore'
        projects: '$(ProjectName)'
        feedsToUse: 'select'
    - task: DotNetCoreCLI@2
      displayName: 'Build Step'
      inputs:
        command: 'build'
        projects: '$(ProjectName)'
        arguments: '--configuration $(buildConfiguration)'
    - task: DotNetCoreCLI@2
```

```
inputs:
  command: 'publish'
  publishWebProjects: true
  arguments: '--configuration $(buildConfiguration) -o $(Build.ArtifactStagingDirectory)'
- task: PublishBuildArtifacts@1
  inputs:
    PathToPublish: '$(Build.ArtifactStagingDirectory)'
    ArtifactName: 'drop'
    publishLocation: 'Container'

- stage: "DeployToDev_Stage"
  dependsOn: Build_Stage
  condition: eq(variables['Build.SourceBranch'], 'refs/heads/development')
  jobs:
    - deployment: 'DeployToDev'
      displayName: "Job: Deploying to Dev Stage"
      pool:
        vmImage: 'ubuntu-latest'
      environment: dev
      strategy:
        runOnce:
          deploy:
            steps:
              - download: current
                artifact: drop
              - task: AzureRmWebAppDeployment@4
                inputs:
                  ConnectionType: 'AzureRM'
                  azureSubscription: $(AzureConnection)
                  appType: 'webApp'
                  WebAppName: 'dssdemoapp-dev'
                  packageForLinux: '$(Pipeline.Workspace)/**/*.zip'

- stage: "DeployToQA_Stage"
  dependsOn: Build_Stage
  condition: eq(variables['Build.SourceBranch'], 'refs/heads/release')
```

```
jobs:
- deployment: 'DeployToQA_Job'
  displayName: "Job: Deploying to QA Stage"
  pool:
    vmImage: 'ubuntu-latest'
  environment: QA
  strategy:
    runOnce:
      deploy:
        steps:
          - download: current
            artifact: drop
          - task: AzureRmWebAppDeployment@4
            inputs:
              ConnectionType: 'AzureRM'
              azureSubscription: $(AzureConnection)
              appType: 'webApp'
              WebAppName: 'dssdemoapp-qa'
              packageForLinux: '$(Pipeline.Workspace)/**/*.zip'

- stage: "DeployToProd_Stage"
  dependsOn: Build_Stage
  condition: eq(variables['Build.SourceBranch'], 'refs/heads/master')
  jobs:
  - deployment: 'DeployToProd_Job'
    displayName: "Job: Deploying to Prod Stage"
    pool:
      vmImage: 'ubuntu-latest'
    environment: Prod
    strategy:
      runOnce:
        deploy:
          steps:
            - download: current
              artifact: drop
            - task: AzureRmWebAppDeployment@4
```

```
inputs:
  ConnectionType: 'AzureRM'
  azureSubscription: '${AzureConnection}'
  appType: 'webApp'
  WebAppName: 'dssdemoapp-prod'
  deployToSlotOrASE: true
  ResourceGroupName: 'SandeepRG'
  SlotName: 'staging'
  packageForLinux: '${Pipeline.Workspace}/**/*.zip'

- task: AzureAppServiceManage@0
  inputs:
    azureSubscription: $(AzureConnection)
    Action: 'Swap Slots'
    WebAppName: 'dssdemoapp-prod'
    ResourceGroupName: 'SandeepRG'
    SourceSlot: 'staging'
```

Deploy Azure Functions

- Step1: Azure Portal → Create an Azure Function App
- Step2: Azure DevOps → Create an Azure Repo
- Step3: Visual Studio → Create an Azure Function App with HTTPTrigger
- Step4: Push the Local Repo to Remote Repo
- Step5: Use the below YAML File

```
trigger:
  - '*'

variables:
  buildConfiguration: 'Release'

stages:
- stage: 'Build'
  displayName: 'Build the web application'
  jobs:
  - job: Build
    displayName: 'Build Job'
```



```
pool:
  vmImage: 'ubuntu-16.04'
variables:
  dotnetSdkVersion: '3.1.x'
steps:
- task: DotNetCoreCLI@2
  displayName: 'Restore project dependencies'
  inputs:
    command: 'restore'
    projects: '**/*.csproj'

- task: DotNetCoreCLI@2
  displayName: 'Build the project - $(buildConfiguration)'
  inputs:
    command: 'build'
    arguments: '--no-restore --configuration $(buildConfiguration)'
    projects: '**/*.csproj'

- task: DotNetCoreCLI@2
  displayName: 'Publish the project - $(buildConfiguration)'
  inputs:
    command: 'publish'
    projects: '**/*.csproj'
    publishWebProjects: false
    arguments: '--no-build --configuration $(buildConfiguration) --
output $(Build.ArtifactStagingDirectory)/$(buildConfiguration)'
    zipAfterPublish: true

- task: PublishBuildArtifacts@1
  displayName: 'Publish Artifact: drop'
  condition: succeeded()

- stage: 'Deploy'

  displayName: 'Deploy the azure Function'
  dependsOn: Build
  variables:
```

```
appName: dssdemoafunc123
azureSubscription: 'Azure Subscription Connection'
jobs:
- deployment: Deploy
  pool:
    vmImage: 'windows-latest'
  environment: 'Azure-function'
  strategy:
    runOnce:
      deploy:
        steps:
          - task: AzureFunctionApp@1
            displayName: Azure Function App Deploy
            inputs:
              azureSubscription: 'Azure Subscription Connection'
              appType: 'functionAppLinux'
              appName: '${appName}'
              package: '${Pipeline.Workspace}/drop/${buildConfiguration}/*.zip'
```

Setting up a CI/CD pipeline for Azure Functions

<https://azuredevopslabs.com/labs/vstsextend/azurefunctions/>

Azure Pipelines Template Library in GitHub

<https://github.com/microsoft/azure-pipelines-yaml/tree/master/templates>