LetsDevOps: YAML Pipeline Tutorial, Setting up CI/CD using YAML Pipeline, Multi Stage/Job Setup.

Introduction

This article is for understanding the core concept of YAML Pipeline in Azure DevOps. Further it describe how you can write your own YAML file to implement CI/CD.

What is YAML

YAML is a human-readable data-serialization language and it helps to configure pipeline as a Code.

Indentation is very important in YAML.

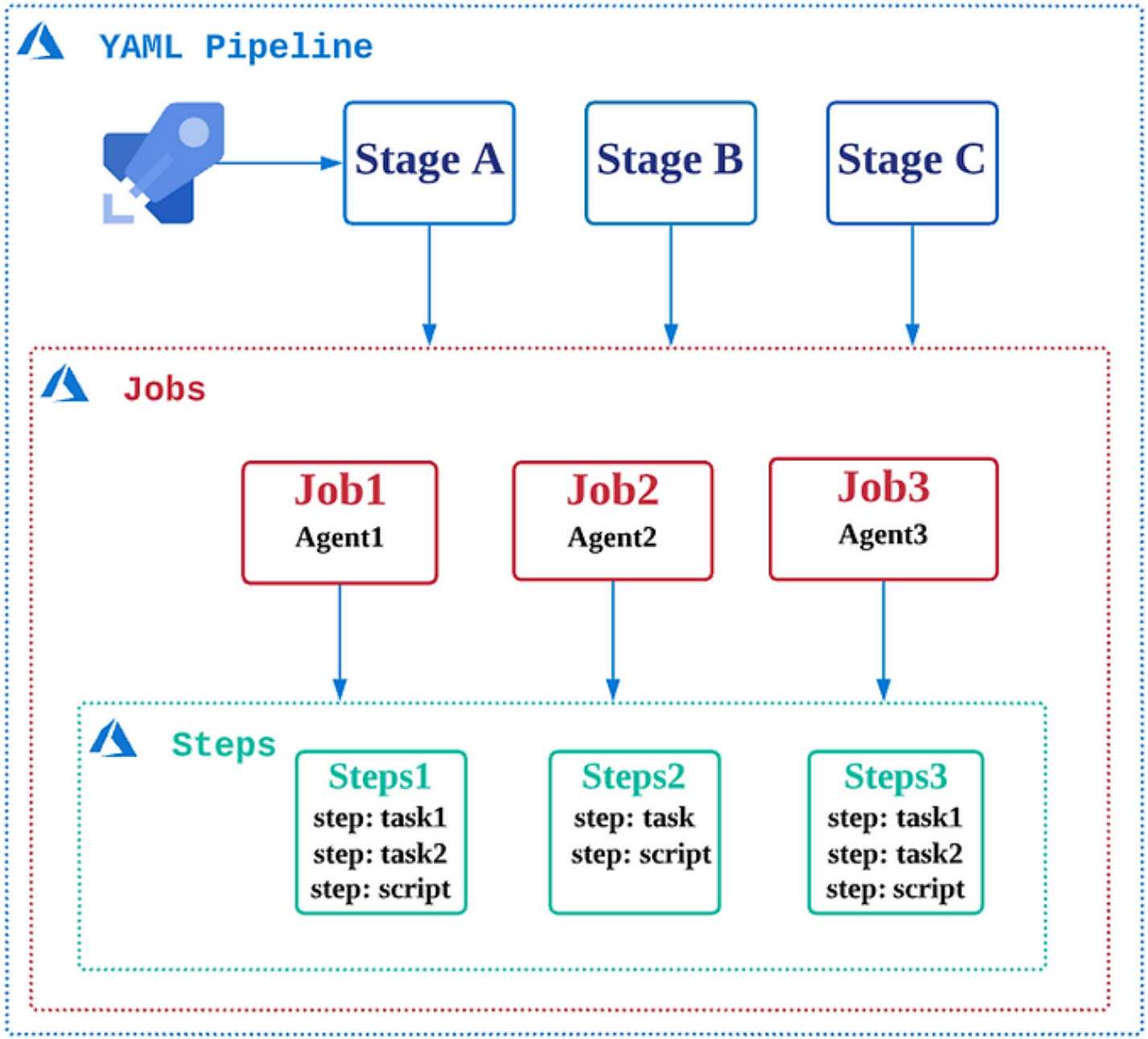
YAML Pipeline

In Azure DevOps, Pipelines helps to setup Continuous Integration/ Continuous Deployment and to achieve this we have below two option .

1. Classic Editor
2. YAML Pipeline.

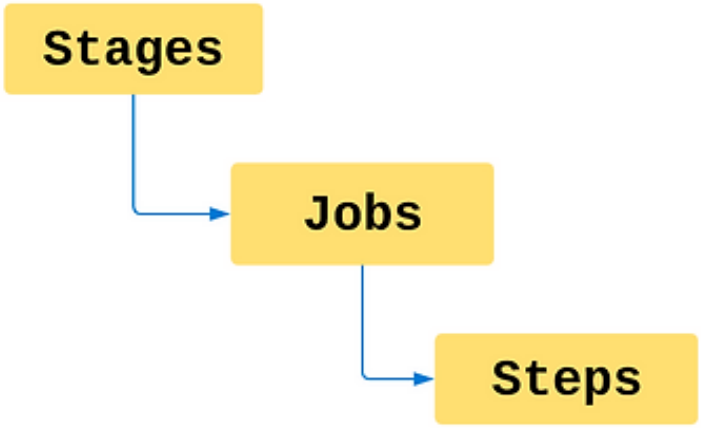
In this Document we will discuss setup YAML Pipeline.

YAML Pipeline Structure


YAML Pipeline
Stage A
Stage B
Stage C
Jobs
Job1
Agent1
Job2
Agent2
Job3
Agent3
Steps
Steps1
step: task1
step: task2
step: script
Steps2
step: task
step: script
Steps3
step: task1
step: task2
step: script

Hierarchy of YAML File

```yaml
stages:
- stage: Build
  jobs:
  - job: BuildPackage
    steps:
    - Build
    - Package
    - Publish
- stage: Deploy
  jobs:
  - job: startDeployment
    steps:
    - Deploypackage
```



Component of YAML Pipeline

Here is the list of component which we use in YAML pipeline creation.

1. Stages

Stage is collection of Jobs which runs sequentially.

```yaml
stages:
- stage: Build
  jobs:
  - job: BuildOnWindows
    steps:
    - Build
  - job: BuildOnMac
    steps:
    - Build
```

2. Jobs

Job is Collection of Steps that runs on agents/environment.

```yaml
jobs:
- job: BuildPackage
  steps:
  - Build
```

```
- Package
- Publish
```

3. Steps

Steps helps to define the set of process to setup your task or any activity which you want to perform on any specific job.

Kind of Steps

- Task
- Script
- templatereference

```
steps:
- Build
- Package
- Publish
```

Schema of YAML Pipeline file.

```
name: string # Define Custom Build number/Name
variables: #Define variable for Pipeline
trigger: trigger/none #it defines which branch to enable for CI
```

```
stages:[stage| templateReference ]
  jobs:[job| templateReference ]
    steps:[script | task | templateReference ]
```

Important Note:-

1.In some cases if there is only one stage required for pipeline we can omit stage keyword and can directly start from Jobs.

2. In some build setup where only one agent is required for pipeline in that case we can omit Job and directly define the Steps.

DEMO:

We can begin with demo to setup YAML pipeline for below three scenarios.

> Single Stage, Single Job YAML Pipeline

> Single Stage Multi Job YAML Pipeline

> Multi Stage YAML Pipeline

1. Single Stage, Single Job YAML Pipeline

Assume that if you have assigned to setup build for an application. In this scenario we can omit stages or Jobs.

```yaml
name: sampleWeb_$(Date:yyyyMMdd)$(Rev:.r)
trigger:
- main
pool:
  name: Default
steps:
- task: VSBuild@1
  inputs:
    solution: '**\*.sln'
    msbuildArgs: '/p:DeployOnBuild=true /p:WebPublishMethod=Package
 /p:PackageAsSingleFile=true /p:SkipInvalidConfigurations=true
/p:PackageLocation="$(build.artifactstagingdirectory)\\"'
    platform: 'any cpu'
    configuration: 'release'
```

Here you can notice that we did not include the Stage and Job section in the YAML pipeline as it is not required in this scenario.

2. Single Stage Multi Job YAML Pipeline

Let's assume you have to perform some activity on different Agent. Like building application on Windows, Mac OS, Linux. In this scenario we can create multiple job and each job can be assigned with respective agent.

```yaml
name: sampleWeb_$(Date:yyyyMMdd)$(Rev:.r)
trigger:
- main
jobs:
- job: ActivityonLinux
  pool:
   name: default
  steps:
  - script: echo Hello, world!
    displayName: 'Run a one-line script'
- job: ActivityonWindows
  pool:
   name: Default
  steps:
   - task: PowerShell@2
     inputs:
       targetType: 'inline'
       script: |
         # Write your PowerShell commands here.

         Write-Host "Hello World"
```

3. Multi Stage YAML Pipeline.

In this scenario you have assigned one application where you need to build the Project, package it. After that you need to deploy the application. In this case we can create two stage.

Stage1 --> Build the Application
Stage 2 --> Deploy the Application

```
name: sampleApp_$(Date:yyyyMMdd)$(Rev:.r)
trigger:
- main
stages:
- stage: Build
  jobs:
  - job: Build
    pool:
     name: Default
    workspace:
     clean: outputs
    steps:
    - task: VSBuild@1
      inputs:
        solution: '**\*.sln'
        msbuildArgs: '/p:DeployOnBuild=true /p:WebPublishMethod=Package
/p:PackageAsSingleFile=true /p:SkipInvalidConfigurations=true
/p:PackageLocation="$(build.artifactstagingdirectory)\\"'
        platform: 'any cpu'
        configuration: 'release'
    - task: PublishBuildArtifacts@1
```

```
      inputs:
        PathtoPublish: '$(Build.ArtifactStagingDirectory)'
        ArtifactName: 'drop'
        publishLocation: 'Container'
- stage: Deploy
  jobs:
  - job: DeployWeb
    pool:
      name: Default
    steps:
    - powershell: "write-host"
    - task: DownloadBuildArtifacts@1
      inputs:
        buildType: 'specific'
        project: '0a1af395-65ea-473b-b60e-01fdc0d3f93e'
        pipeline: '11'
        buildVersionToDownload: 'latest'
        downloadType: 'single'
        downloadPath: '$(System.ArtifactsDirectory)'
    - task: AzureRmWebAppDeployment@4
      inputs:
        ConnectionType: 'AzureRM'
        azureSubscription: 'testconn'
        appType: 'webApp'
        WebAppName: 'demotest0103'
        packageForLinux: '$(System.ArtifactsDirectory)/**/*.zip'
```

LetsDevOps: YAML Pipeline Tutorial, Setting up CI/CD using YAML Pipeline, Multi Stage/Job Setup.