Visual Studio LIVE! | Las Vegas
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

I Love YAML: Using Azure DevOps to Build and Release to Azure

Benjamin Day
www.benday.com
@benday

#VSLIVE

Your Code Powers the World.
Our Training Powers You.

Benjamin Day

Brookline, MA
Consultant & Trainer
Rental Chief Technology Officer
Therapist for Teams
Scrum, DevOps, Azure,
Team Foundation Server,
Software Architecture & Testing
Microsoft MVP
Pluralsight Author
Scrum.org Trainer
@benday

Microsoft Most Valuable Professional
Benjamin Day Consulting
PLURALSIGHT
Scrum.org The Home of Scrum

Azure DevOps Services Fundamentals

Azure DevOps Server 2020 Fundamentals

Scrum Master Skills

On with the show.

Overview

- YAML-based Pipelines
- Goal: Build, Test, and Deploy with YAML Pipelines
- Pools, Triggers, and Variables
- Use Docker Containers in Pipelines
  - SQL Server
- Deploy database changes
  - EF Core Migrations
- Multi-environment Deploys
- Multi-environment Approvals

Next up: YAML-based Pipelines

YAML-based Pipelines

Build & Release Pipelines in Azure DevOps

Classic Build & Release Pipelines
- ~10 years old
- Use a designer to describe your pipeline
- Under the surface, JSON-based
- Stored in build/release system
- Separates builds from releases

YAML-Based Pipelines
- Recent addition to the product
- Mostly text-based
- Stored in version control
- Builds & releases are “stages” in the same pipeline definition

What is YAML?

Yet Another Markup Language

YAML Ain't Markup Language

Specification @ yaml.org

- "YAML is a human-friendly data serialization language for all programing languages."

Confusing.

It's what we'll use to describe our release pipelines in Azure DevOps

- (It's also what GitHub Actions uses.)

Azure DevOps Classic Pipelines vs YAML Pipelines

Classic Pipelines	YAML Pipelines
Uses JSON	Uses YAML
Stored somewhere in Azure DevOps database	Scripts are stored in Azure DevOps Git repository
Available regardless of version control option	Not available in TFVC → Requires that you use Git
Splits build activities from release activities	Describes the entire build and release pipeline in a single file
Nice designer	Editable in any text editor of your choice!

Giant benefit of YAML-based pipelines:

Pipeline scripts are stored in version control!

YAML Pipeline scripts in Git? Who cares?

Powerful but subtle benefits

Versions your pipeline scripts along with your code

→ Let's you branch & merge your pipeline scripts along with your code

Next up:
Create a YAML build

Demo

Create a YAML-based pipeline
- Build
- Continuous integration trigger
- Define a variable

Demo

Create a YAML-based pipeline
- Publish an artifact

Demo

Create a YAML-based pipeline
- Run unit tests

Stages, Jobs, & Steps in YAML

Basic Structure of an Azure DevOps YAML-based Pipeline

Pipeline
- Stage A
  - Job 1
    - Step 1.1
    - Step 1.*n*
  - Job 2
    - Step *n*
- Stage B
- Stage *n*

https://docs.microsoft.com/en-us/ azure/devops/pipelines/yaml-schema

Pipeline

Top level item

Name of the pipeline

Global variables

Agent pool

Triggers
- Manual only: “none”
- Branches
- File paths
- Tags

Pull request triggers

Has a collection of Stages

Stage

Collection of related jobs

A stage has…
- Display name
- Conditions
- Variables
- Collection of jobs

My typical stage structure:
- Build
- Deploy to test environment
- Wait for human approval
- Deploy to prod environment

Jobs

Collection of steps

Run on the server or agent machine

Can be run in parallel within a stage

NOTE: I usually only have one Job per Stage

Steps

"a linear sequence of operations that make up a job"

Basically: it does something

Think of it as a command line call

Each step runs in its own process

- (Remember this if environment variables get weird for you!)

Pools

Defines the agent for your pipeline jobs

Agents
- Hosted by Microsoft
- Self-hosted

Hosted Agents

Use a Microsoft-hosted agent

YAML | Classic

In YAML pipelines, if you do not specify a pool, pipelines will default to the Azure Pipelines agent pool. You simply need to specify which virtual machine image you want to use.

YAML Copy

```yaml
jobs:
- job: Linux
  pool:
    vmImage: 'ubuntu-latest'
  steps:
  - script: echo hello from Linux
- job: macOS
  pool:
    vmImage: 'macOS-latest'
  steps:
  - script: echo hello from macOS
- job: Windows
  pool:
    vmImage: 'windows-latest'
  steps:
  - script: echo hello from Windows
```

https://docs.microsoft.com/en-us/ azure/devops/pipelines/agents/hosted

Hosted Agent Types

Set the "pool → vmImage" value

Linux
- 'ubuntu-latest'

MacOS
- 'macOS-latest'

Windows
- 'windows-latest'

Self-hosted Agents

Agents that you install & maintain

Set the "pool → name" value

Variables

Can be defined at
- Pipeline level (root, global)
- Stage level
- Job level

Variables

YAML Copy

```yaml
variables:
- name: one
  value: initialValue

steps:
  - script: |
      echo ${{ variables.one }} # outputs initialValue
      echo $(one)
    displayName: First variable pass
  - bash: echo '##vso[task.setvariable variable=one]secondValue'
    displayName: Set new variable value
  - script: |
      echo ${{ variables.one }} # outputs initialValue
      echo $(one) # outputs secondValue
    displayName: Second variable pass
```

https://docs.microsoft.com/en-us/azure/devops/pipelines/process/variables

Demo

Use a Docker container in the pipeline
SQL Server container
Deploy EF Core migrations
Run integration tests

Demo

Multiple stages in a pipeline
Part 1 of 3
Separate build from deploy

Demo

Multiple stages in a pipeline

Part 2 of 3

Use Marketplace Extensions in a pipeline

Server-side pipeline variables

Configure database connection strings

Deploy EF Core migrations

Demo

Multiple stages in a pipeline

Part 3 of 3

Deploy to an Azure App Service
- Deploy to a Deployment Slot

Set up a service connection
- From: Azure DevOps
- To: Azure Subscription

Demo

Review what's in 'test' before deploying to 'production'

Approvals between stages

Manual Validation Step
- Server-side step
- Pauses execution
- Accept or reject

Demo

Run a YAML-based pipeline using a self-hosted agent
- Self-hosted agent setup demo is in the previous module

YAML conditions

Summary

- YAML-based Pipelines
- Goal: Build, Test, and Deploy with YAML Pipelines
- Pools, Triggers, and Variables
- Use Docker Containers in Pipelines
  - SQL Server
- Deploy database changes
  - EF Core Migrations
- Multi-environment Deploys
- Multi-environment Approvals

Any questions?

Thank you.

www.benday.com | benday@benday.com