

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













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

BDC

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.)



@benday | www.benday.com

Azure DevOps Classic Pipelines vs YAML Pipelines

Classic Pipelines

Uses JSON

Stored somewhere in Azure DevOps database

Available regardless of version control option

Splits build activities from release activities

Nice designer

YAML Pipelines

Uses YAML

Scripts are stored in Azure DevOps Git repository

Not available in TFVC → Requires that you use Git

Describes the entire build and release pipeline in a single file

Editable in any text editor of your choice!

BDC

Giant benefit of YAML-based pipelines:

Pipeline scripts are stored in version control!

YAML Pipeline scripts in Git?
Who cares?

Powerful but subtle benefits

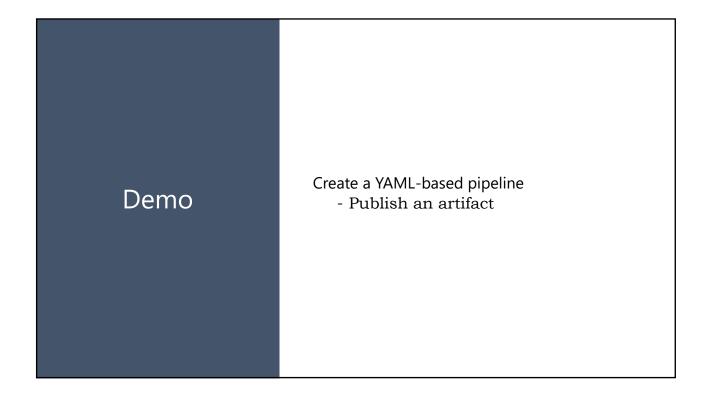
Versions your pipeline scripts along with your code

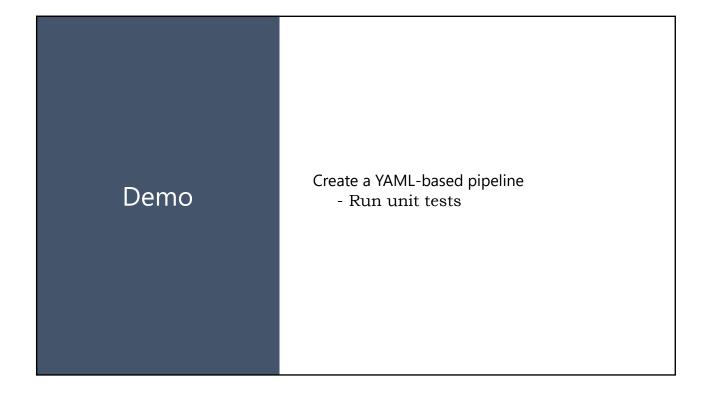
→ Let's you branch & merge your pipeline scripts along with your code

BDC

Next up: Create a YAML build

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





Stages, Jobs, & Steps in YAML

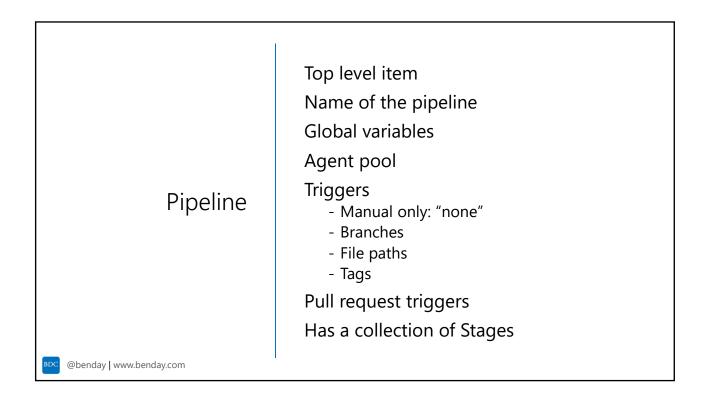
Basic Structure of an Azure DevOps YAML-based Pipeline

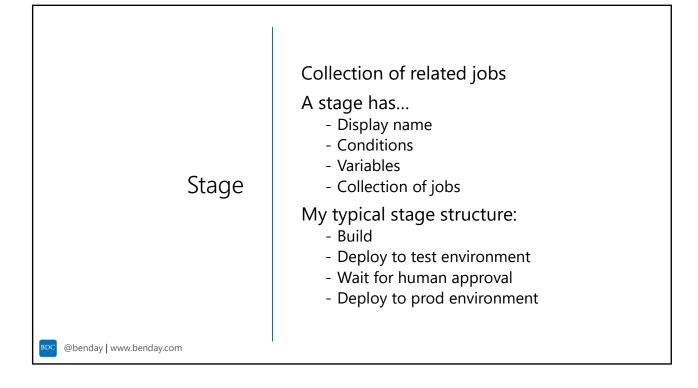
Pipeline

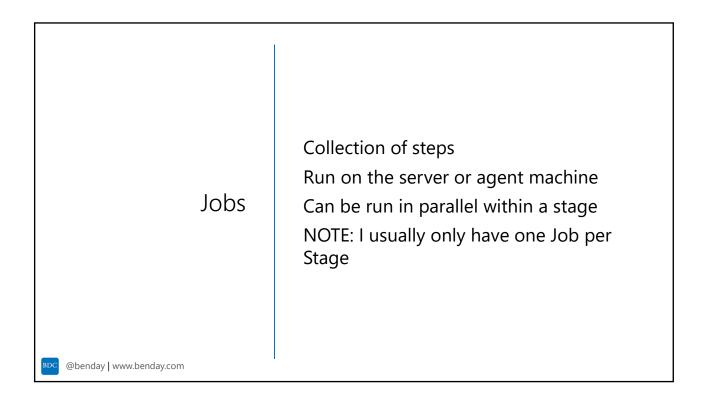
- Stage A
 - Job 1
 - Step 1.1
 - Step 1.*n*
 - Job 2
 - Step *n*
- Stage B
- Stage *n*

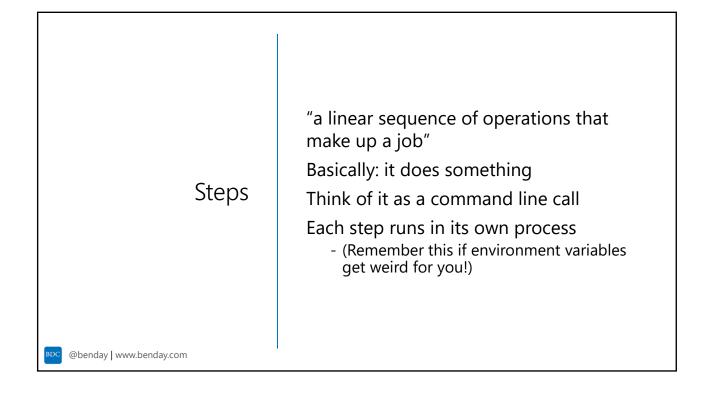
https://docs.microsoft.com/en-us/ azure/devops/pipelines/yaml-schema

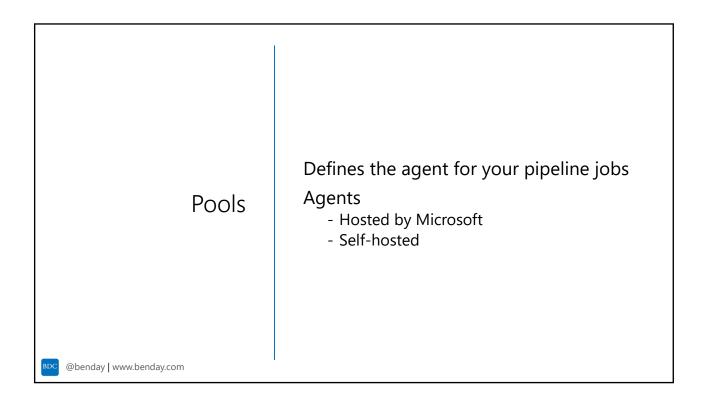
BDC

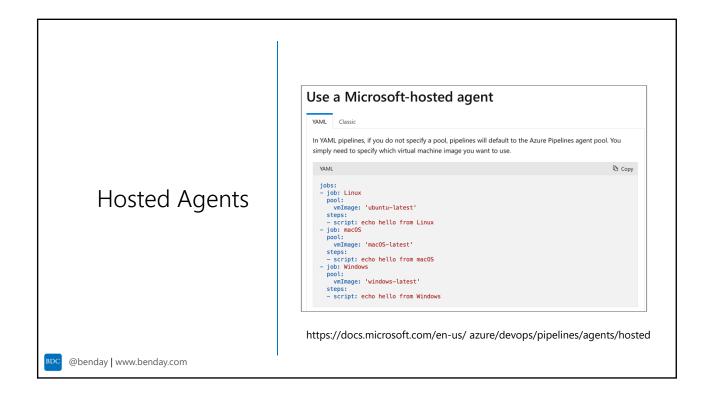


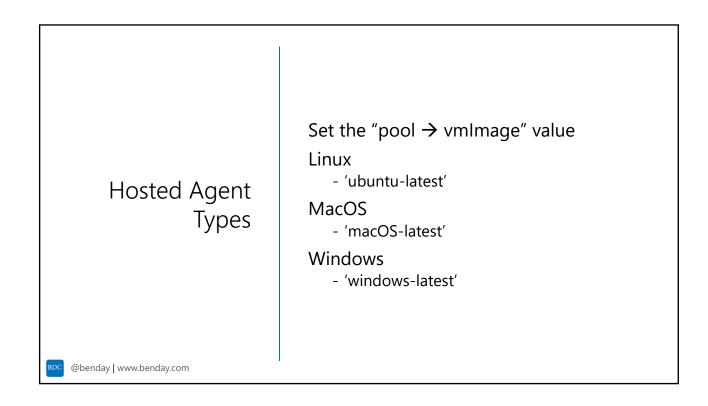




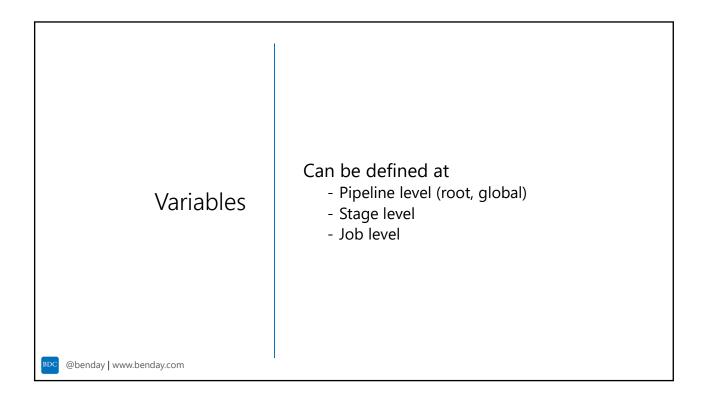


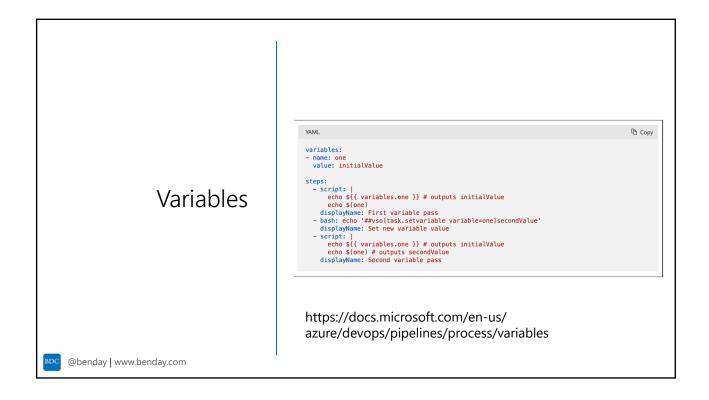


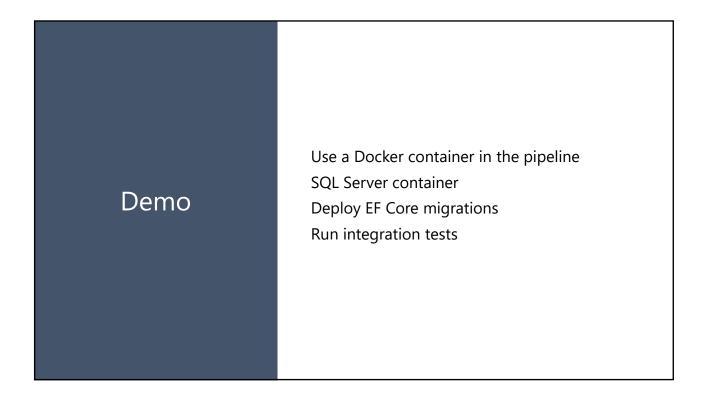




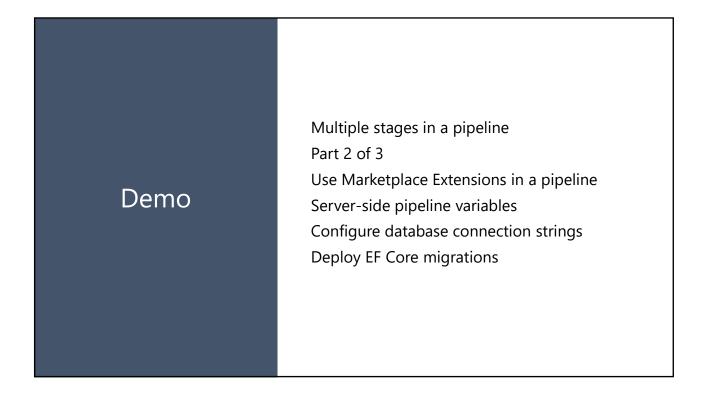




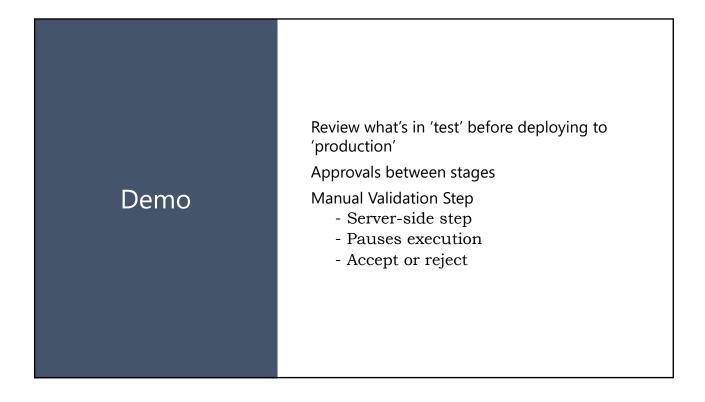




Multiple stages in a pipeline
Part 1 of 3
Separate build from deploy

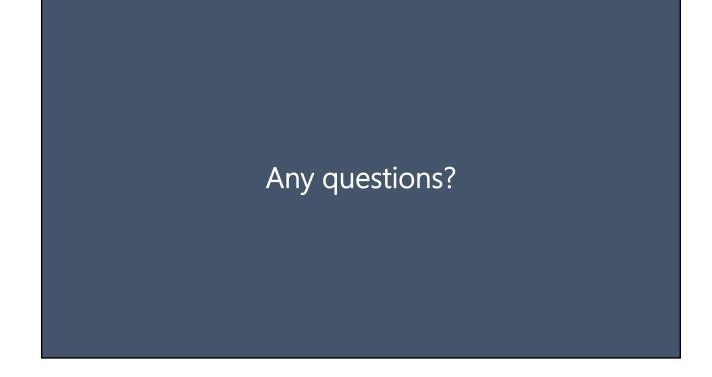


Multiple stages in a pipeline
Part 3 of 3
Deploy to an Azure App Service
- Deploy to a Deployment Slot
Set up a service connection
- From: Azure DevOps
- To: Azure Subscription



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

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







www.benday.com | benday@benday.com