Form Scriber DevSecOps CI/CD Pipeline: Technical Design Document

Ivy Pham & Vincent Leung

University of Maryland Global Campus

Spring 2021

UMGC Capstone Project

Form Scriber DevSecOps CI/CD Pipeline

Technical Design Document

Version 3.0

Name: Ivy Pham & Vincent Leung

SWEN670 9040

Date: 02/02/2021

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 02/02/2021 | 1.0 | Initial document | Vincent Leung |
| 02/07/2021 | 1.1 | Added section 1 | Vincent Leung |
| 02/16/2021 | 1.2 | Added and modified various sections | Ivy Pham |
| 02/19/2021 | 1.3 | Added and modified various sections | Ivy Pham |
| 02/20/2021 | 1.4 | Added and modified various sections | Ivy Pham |
| 02/20/2021 | 1.5 | Added and modified various sections | Ivy Pham |
| 02/21/2021 | 1.6 | Modified various sections | Vincent Leung |
| 03/15/2021 | 2.0 | Modified for Milestone 3 | Ivy Pham |
| 03/30/2021 | 3.0 | Final Review | Ivy Pham |

Table of Contents

[1 INTRODUCTION 5](#_Toc64826361)

[1.1 Purpose 5](#_Toc64826362)

[1.2 Scope 5](#_Toc64826363)

[1.3 Definitions and Acronyms 5](#_Toc64826364)

[2 SYSTEM ARCHITECTURE 7](#_Toc64826365)

[2.1 Overall Architectural Design 7](#_Toc64826366)

[2.2 Code Repository 8](#_Toc64826367)

[2.3 Azure DevOps 8](#_Toc64826368)

[2.3.1 Build Pipeline 9](#_Toc64826369)

[2.3.2 Release Pipeline 10](#_Toc64826370)

[2.4 Microsoft Azure 11](#_Toc64826371)

[2.4.1 Azure Kubernetes Service (AKS) 11](#_Toc64826372)

[2.4.2 Azure Virtual Machine and App Services 13](#_Toc64826373)

[2.5 Advanced Deployment Factory (ADF) 13](#_Toc64826374)

[3 COMPONENT DESIGN 13](#_Toc64826375)

[3.1 Technologies 13](#_Toc64826376)

[3.2 Build Pipelines 14](#_Toc64826377)

[3.2.1 Java 14](#_Toc64826378)

[3.2.2 JavaScript 15](#_Toc64826379)

[3.2.3 Flutter 16](#_Toc64826380)

[3.2.4 Go 17](#_Toc64826381)

[3.3 Release Pipelines 18](#_Toc64826382)

[3.4 Azure Kubernetes Service 19](#_Toc64826383)

[3.5 Advanced Deployment Factory (ADF) 21](#_Toc64826384)

# INTRODUCTION

## Purpose

This Technical Design Document (TDD) describes the design and architecture of the Form Scriber DevSecOps continuous integration and continuous delivery (CI/CD) pipeline. This will detail the components, workflow, and the infrastructure required to implement said system as per requirements. The DevSecOps team will use the TDD as a point of reference for specifications during the implementation phases. As such, this document shall represent the overall blueprint to systematically realize the design of the CI/CD pipeline. This shall be accomplished in two over-arching parts, the former of which shall consist of the introduction and architecture overview. The latter part shall detail the component design.

## Scope

The DevSecOps CI/CD pipeline will serve to manage the integration and deployment of the Form Scriber application. This document will be the basis of the design and implementation of said infrastructure. With the current pace and cadence of modern software development efforts, it is no longer effective or conducive to market expectations and timelines to rely on manual development operations processes. The CI/CD pipeline will therefore execute automated operations activities related to the building, testing, scanning, and deployment of applications in a highly integrated and monitored solution. The pipeline will enable the Form Scriber development teams to better deliver higher quality code more efficiently and frequently. The system is to be a collection of distributed software services on the web configured specifically for the technical needs of the Form Scriber development teams. Implementation shall encompass the configuration and setup of each pipeline component to deliver the core functionalities of integrating and deploying the two Form Scriber components, DialogFlow and Mobile, and shall exclude those such as any user interface(s) external to the defaults already provided by the services and the command line, continually hosted DevSecOps resources, ongoing support for those resource instances, and access to development accounts.

## Definitions and Acronyms

**ADF**– Advance Development Factory

**AKS** – Azure Kubernetes Service which offers serverless Kubernetes and integrated CI/CD

**CI CD** – Continuous Integration and Continuous Deployment

**cluster –** contains Kubernetes nodes, master node, and slave node (worker node)

**Docker container** – when the image is started, and the container environment is created

**Docker image** – the actual application package, an artifact that can be moved around, such as in a Docker repo

**DSO –** DevSecOps

**kubectl** – CLI tool for Kubernetes clusters

**namespaces** – clusters inside a cluster, groups Kubernetes resources together for organizational purposes

**node** – the smallest package of Kubernetes to manage, usually has one container inside

**pod** – the smallest deployable unit in Kubernetes, a layer of abstraction over containers

**VM** – Virtual Machine

**YAML** – data serialization language used for configuration files

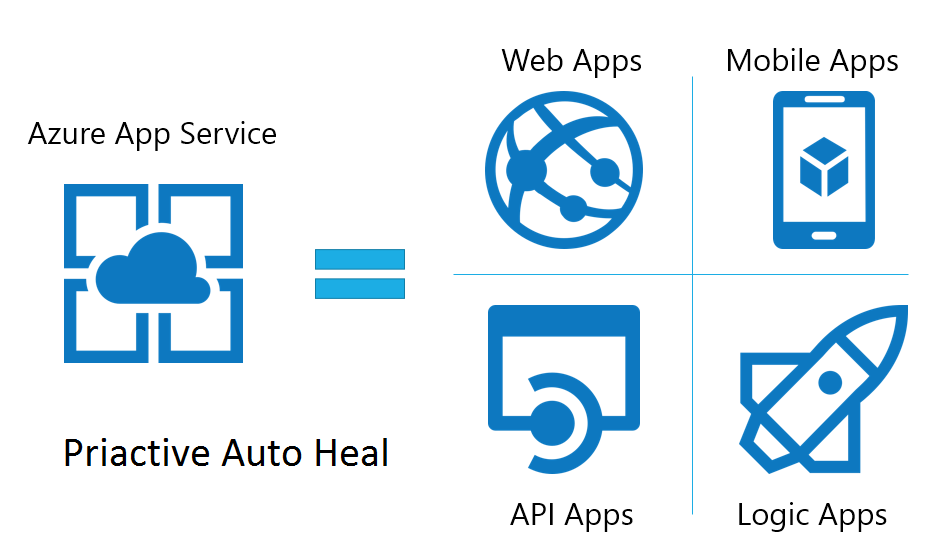
# SYSTEM ARCHITECTURE

## Overall Architectural Design

The Form Scriber DevSecOps project requires the use of free GitHub and Azure services. Therefore, the solution is designed to be deployed on these platforms. GitHub will provide the code repository for collaboration via distributed version control. The Form Scriber project will utilize two main services from Azure. **Azure DevOps** provides a set of tools to improve collaboration and productivity for development and IT operations. **Microsoft Azure** provides cloud services related to infrastructure such as virtual computing, storage, networking, containers, and other services. Please see Figure 1 for the overall architecture.

**Figure 1**

*Overall Architect Diagram*



**RELEASE**

**Code Analysis**

**Overall Architect Diagram**

**Deploy**

**Publish**

**Test**

**Build**

**BUILD**



**Engineers**

**Repository**

**Virtual Machine**

**Repository**

## Code Repository

All development teams (DialogFlow and Mobile) are required to utilize GitHub repository. The DSO team is responsible for the creation of the project and appropriate repositories. Each team will have access to the appropriate repository with a set of policies created by the DSO team. Please see the structure of the repository below.

Diagram

Description automatically generated

Figure 2. Branch Overview

*Note*. Read from bottom to top

* The master branch is protected, which requires the approval of the DSO team for the merge from the development branch in order to check for code quality and security. Direct merge to master branch from the feature branch is not prohibited.
* The development branch is restricted to the lead developer or one required review from the team member.
* The feature branch allows developers to create a new branch for each feature being worked on.

## Azure DevOps

GitHub repository for each development team will be integrated with Azure DevOps pipeline. Azure DevOps pipeline will monitor changes in the repository and trigger processes as configured. There are two types of pipeline in Azure DevOps. The **Build** pipeline provides steps for the build (compile), test, code analysis, package/publish artifacts, the process known as CI (Continuous Integration). The **Release** pipeline provides the means to deploy artifacts to a specific environment such as a Virtual Machine (VM), Docker image, or Kubernetes in Microsoft Azure or other supported cloud providers. The Release pipeline is known as CD (Continuous Deployment), which is a part of the CI/CD process.

### Build Pipeline

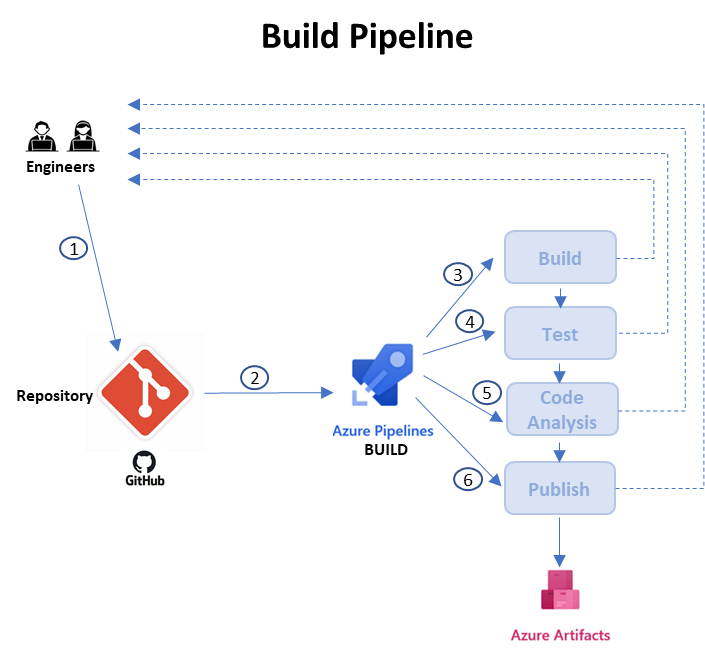


Figure 3. Build Pipeline

For the development branch, Azure pipeline will be configured only to run Build and Test to ensure the code is compiled and passes unit tests. The developer will get notification from the pipeline for the status of each merge to the development branch. For the master branch, the merge from development is controlled by the DSO team as previously mentioned in section 2.2. Therefore, DSO team will do a manual code review and compliance checks before approving the merge to the master branch. Azure DevOps will run the full pipeline which includes Build, Test, Code Analysis and Publish. DSO team will be notified with the status for every merge to the master branch. The gated check-in for Azure pipeline will be considered for the master branch.

The Form Scriber application utilizes 3 (3) main programming languages, which includes JavaScript (Vue), Dart (Flutter) and Go. Therefore, one pipeline (Dialogflow) will cover Go and JS and the other (Flutter/Dart) covers Go. Below are the high-level steps of the Build pipeline.

1. Merge (Pull) is requested to a branch of the GitHub repository.
2. Azure pipeline automatically detects changes and triggers the Build task.
3. The Build task is configured to build (compile) code and can automatically trigger the next task in the pipeline if the build is successful.
4. The Test task is configured to run unit tests and can automatically trigger the subsequent tests if the unit test is passed.
5. The Code Analysis task is configured to check for code quality and security. The plan is to review the report and remediate according to policies agreed among the teams.
6. The Publish task is configured to package and store the artifacts such as executables, zip, or application package to a binary repository which can be Azure Artifacts or a staged location used by the Build pipeline.

### Release Pipeline

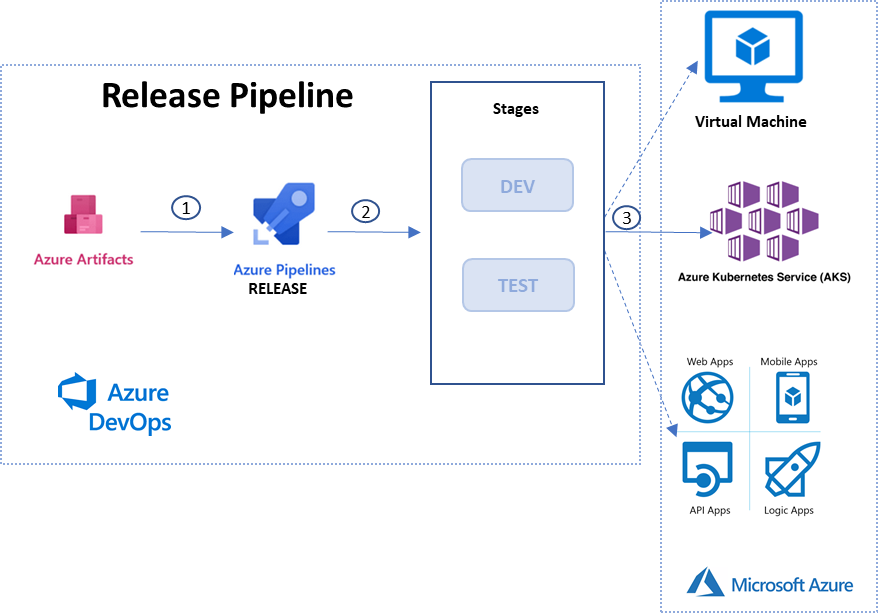


Figure 4. Release Pipeline

The release pipeline can be configured to monitor an artifact repository for changes in versions to deploy the application to a specified environment such as development or test. In Figure 4, the release pipeline monitors Azure Artifacts, which can be the staged or temporary repository for the build pipeline to publish application binaries or packages.

1. The release pipeline detects new version of the artifact that is published by the Azure Build pipeline and triggers the deployment.
2. The release is configured with stages such as development and test.
3. The release pipeline deploys the artifact to a specific environment.

## Microsoft Azure

As described previously, **Microsoft Azure** provides cloud services relating to infrastructure such as virtual computing, storage, networking, containers, and other services. The pipeline shall deploy a Kubernetes cluster in Azure utilizing AKS (Azure Kubernetes Service). The fallback plan is to leverage traditional Azure virtual machines and app services where it is not feasible or possible to deploy Form Scriber components on AKS.

### Azure Kubernetes Service (AKS)

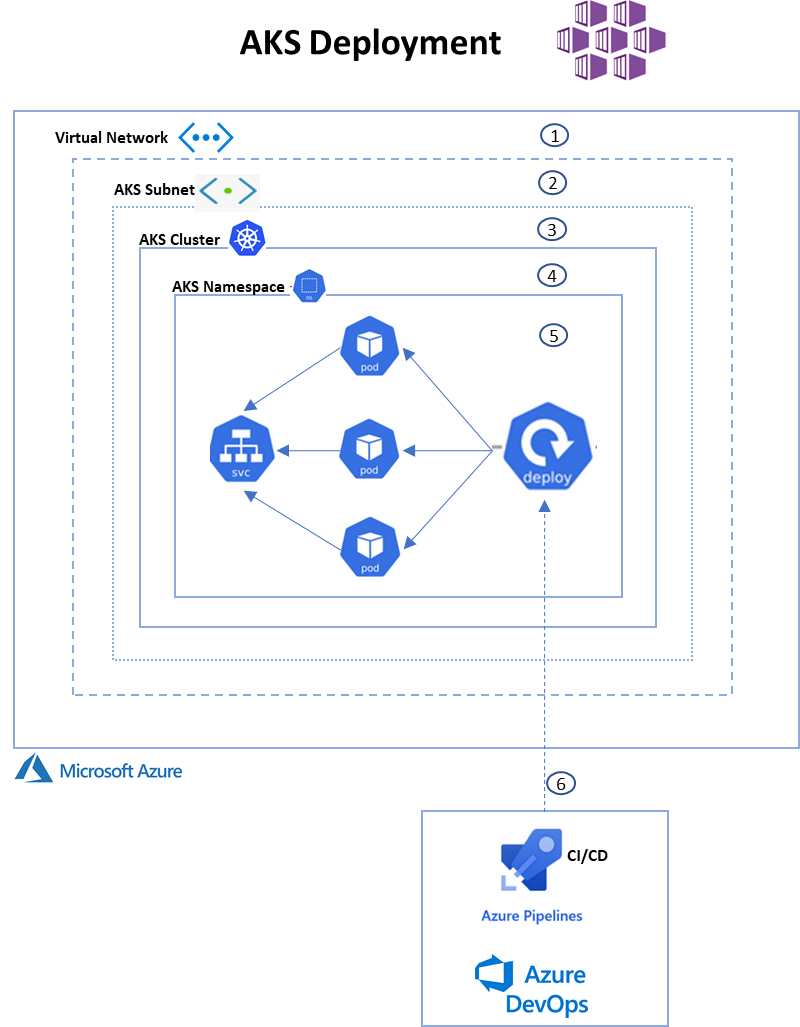


Figure 5. AKS Deployment

Azure provides several methods to deploy an AKS cluster. Most components can be deployed at one time. However, each component can be deployed separately for more customization. Figure 5 shows each AKS component.

* 1. Virtual Network can be created separately to provide an internal virtual network within an organization.
  2. AKS Subnet provides the separation and boundary between networks within an organization.
  3. AKS Cluster provides a group of all related components and services to manage, deploy and scale containers.
  4. AKS Namespace groups logical resources such as pods and deployments, which can be designated to separate each project with access restrictions.
  5. AKS resources include services, deployments, and pods (containers) for applications, which can be scaled rapidly.
  6. CI/CD pipeline automates the deployment of container applications to AKS cluster.

### Azure Virtual Machine and App Services

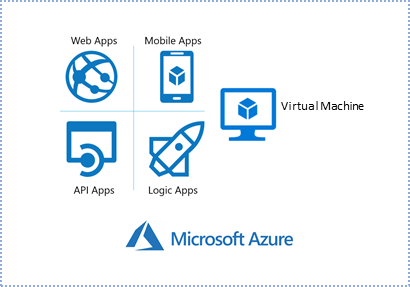


Figure 6. Microsoft Azure Services

The traditional virtual machines and app services offered by Microsoft Azure will be considered where it is not feasible to deploy Form Scriber components on AKS. Additional information will be provided as needed.

## Advanced Deployment Factory (ADF)

To satisfy a requirement of using ADF framework that was created from the previous semester, an ADF solution for Go will be written with the Make file template. ADF will be leveraged to provide developers tools and the method to deploy to AKS.

# COMPONENT DESIGN

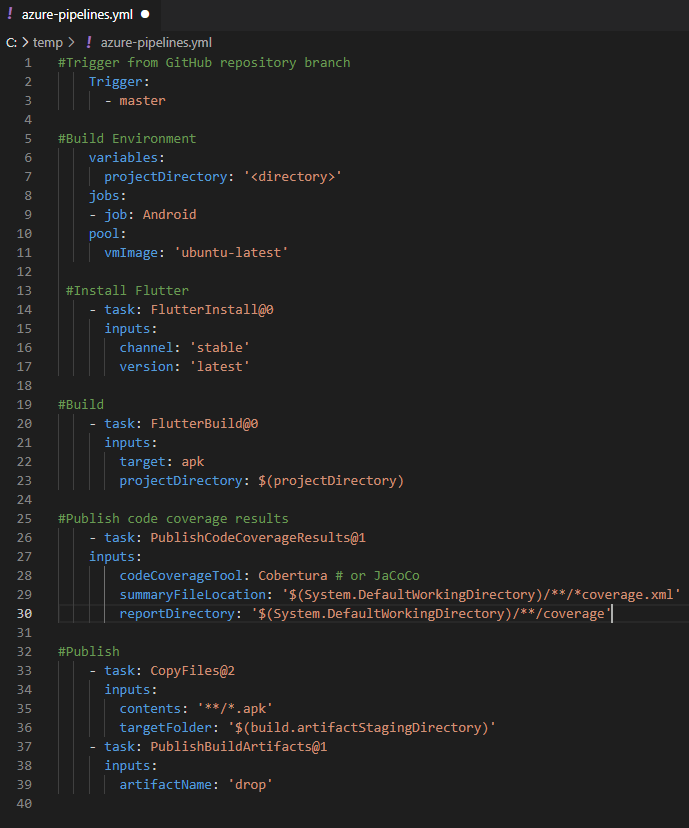
## Technologies

* **GitHub** – Distributed Code Repository
* **Microsoft Azure** – Infrastructure Cloud Services
* **Azure DevOps** – CI/CD tools
* **Docker.OI**

## Build Pipelines

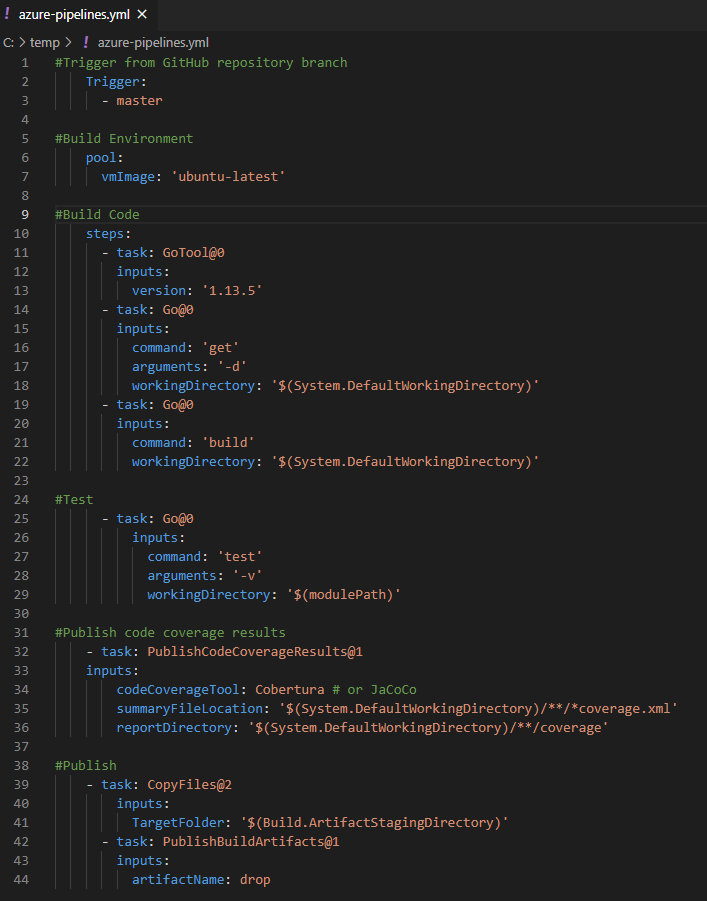
### Flutter and JavaScript

For the Dart (Flutter) language which will be utilized by the development team, an Azure pipeline will be created using the YAML method. JavaScript has small footprint in the Mobile application. Therefore, the Go pipeline will be installed NPM (Native Package Manager) to do some checks on JavaScript codes. Please see the example below:



### Go

For the Go language which will be utilized by the development team, an Azure pipeline will be created using the YAML method. Please see the example below:



## Release Pipelines

An AKS environment will be added to the Azure DevOps pipeline to deploy Docker image with Form Scriber components. Please see the example in Figure 7.

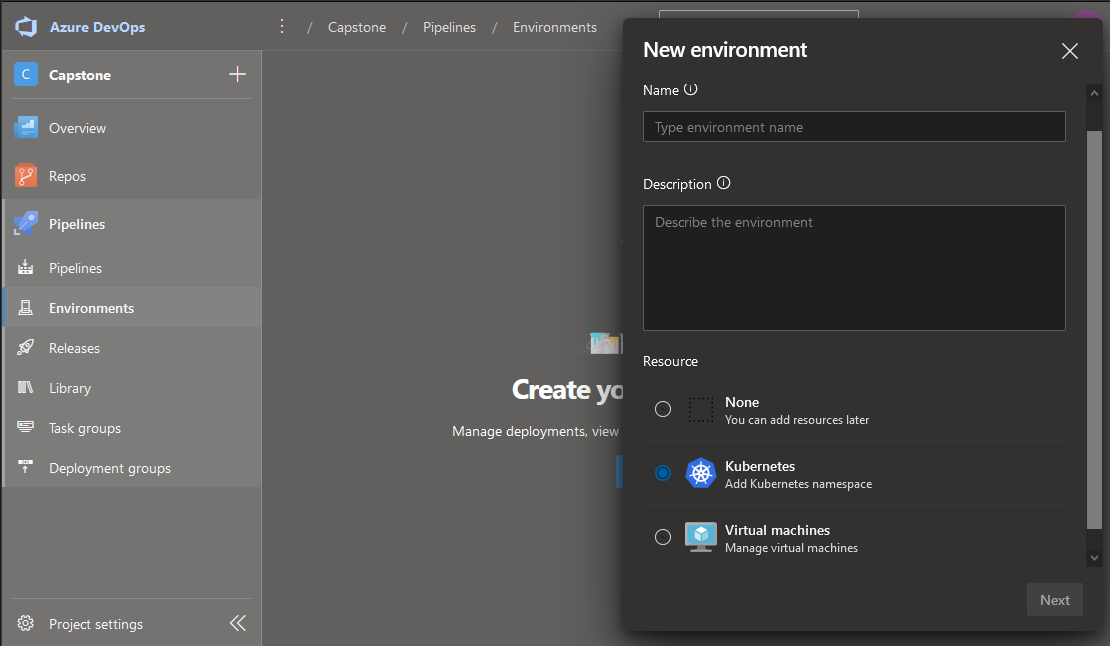


Figure 7. Azure DevOps and AKS

A release pipeline will be created to contain one stage called “Test Environment” to deploy Docker image with Form Scriber components. The release pipeline will also deploy the Docker image to the AKS cluster on Azure. Please see the example in Figure 8.

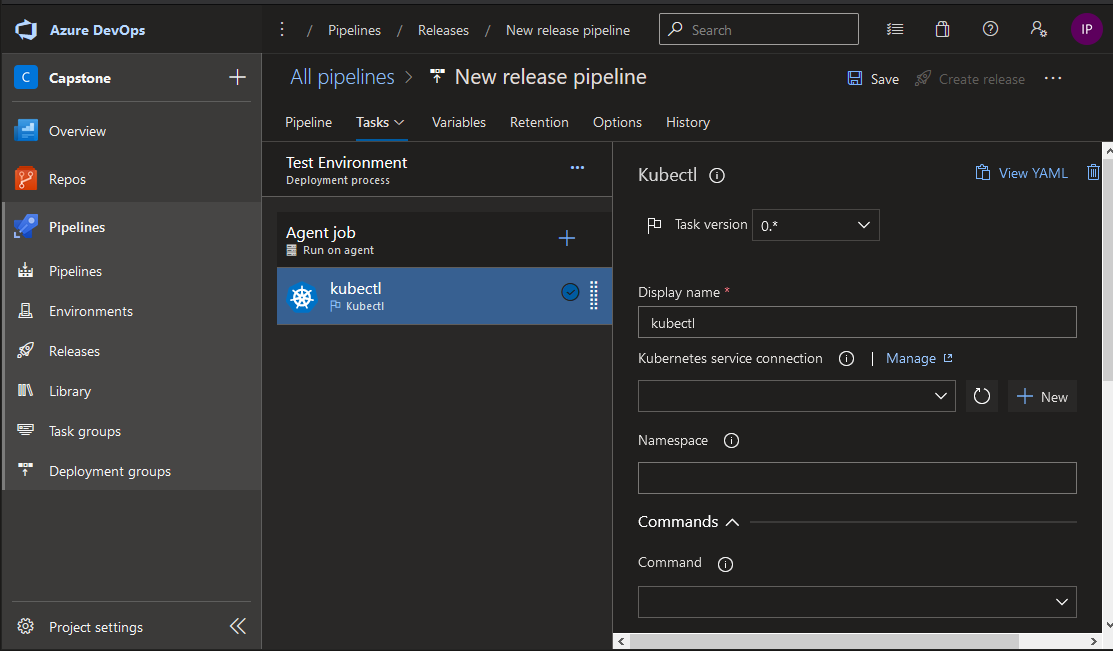
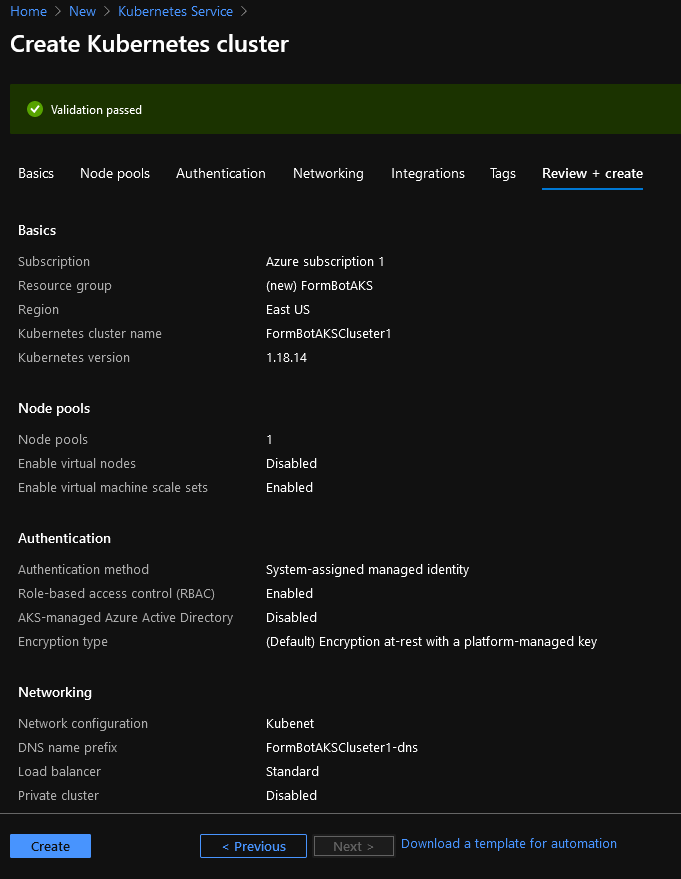


Figure 8. Release Pipeline Stage

## Azure Kubernetes Service

An AKS cluster will be created on Microsoft Azure. The cluster will be named as “Form ScriberAKS”. Creating the AKS cluster can be completed and managed using command line (kubectl) or the wizard provided by Azure. For customization, each layer can be deployed separately. For this project, AKS will be deployed with recommended defaults. Please see the example in Figure 9. The AKS will be added to the student free account with $200 credit for the first month.

  
Figure 9. Deploy AKS

## Advance Development Factory (ADF)

DSO will create the ADF for Go to cover all development tools. The docker image will include all necessary GO tools, Azure CLI, Helm and KuberCTL for developers to use. Developers will only need to install GIT and Docker to able to able to leverage GO ADF.