Form Scriber DevSecOps

Programmer’s Guide

Version 1.4

Ivy Pham

Vincent Leung

SWEN 670 Software Engineering Project

University of Maryland Global Campus

March 30, 2021

**REVISION HISTORY**

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Description** |
| 03/17/2021 | 1.0 | Initial draft |
| 03/22/2021 | 1.1 | Added DSO environment setup, email notifications, and code quality checks setup |
| 3/23/2021 | 1.2 | Added ADF version for Golang |
| 3/25/2021 | 1.3 | Added Release Pipeline and Kubernetes Dashboard |
| 3/27/2021 | 1.4 | Added various sections |

Contents

[1. Set Up DevSecOps Environment 4](#_Toc67987584)

[2. Add Email Notifications for Azure Build/Release Status 5](#_Toc67987585)

[3. Add Code Quality Checks for Project 8](#_Toc67987586)

[4. Advance Development Factory (ADF) for Golang 10](#_Toc67987587)

[5. Access the Kubernetes Dashboard 16](#_Toc67987588)

[6. Azure Release Pipeline to Azure Kubernetes Service 18](#_Toc67987589)

[7. Azure Build Pipeline Mobile (Flutter/Dart) 24](#_Toc67987590)

[8. Azure Build Pipeline Dialogflow (Golang) 29](#_Toc67987591)

[9. Advance Development Factory (ADF) – Dialogflow/AKS Deployment 33](#_Toc67987592)

[10. Azure Release Pipeline – Google Play Store Push 37](#_Toc67987593)

# 1. Set Up DevSecOps Environment

You may choose to use any operating system desired. However, Linux or MacOS is recommended. Below are the necessary tools required to use the DevSecOps environment on a local development machine. A web search for install instructions for the tools is recommended if needed for your operating system.

* Git (refer to FormScriber DevSecOps Runbook for install steps)
* Bash shell (<https://www.gnu.org/software/bash/>)
* Make (<https://www.gnu.org/software/make/>)
* Docker (<https://docs.docker.com/get-docker/>)
* Azure CLI (<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli>)
* Note: This dependency is only required for DSO team members. The others above are required for both DSO and development team members.

# 2. Add Email Notifications for Azure Build/Release Status

1. Ensure you are logged in to the Azure DevOps admin account or DevSecOps team account with granted permissions (see Runbook, Azure section).
2. Go to the UMGC Capstone project within Azure DevOps: <https://dev.azure.com/umgccapstone/Capstone>
3. Click **Project settings** on the bottom left of the page.

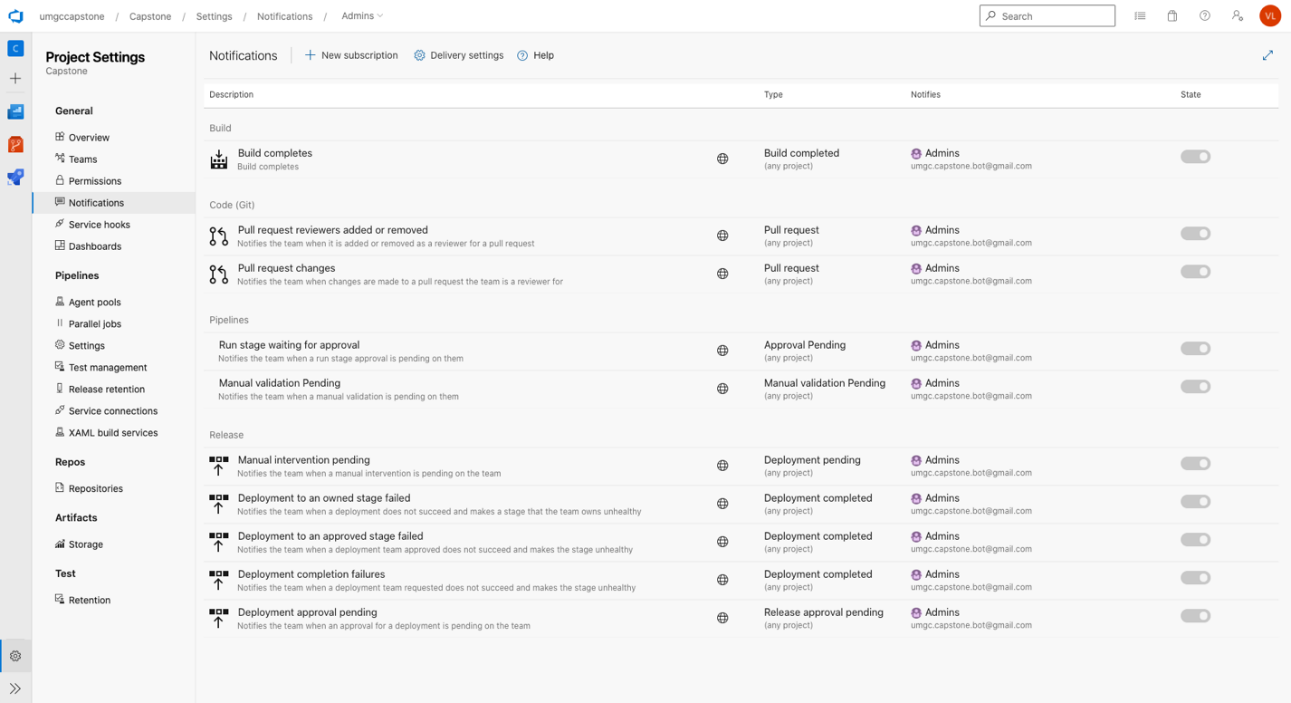


Figure. 1. Azure DevOps Project Settings

1. Click **Notifications** from the left menu.
2. Click **Delivery settings.**

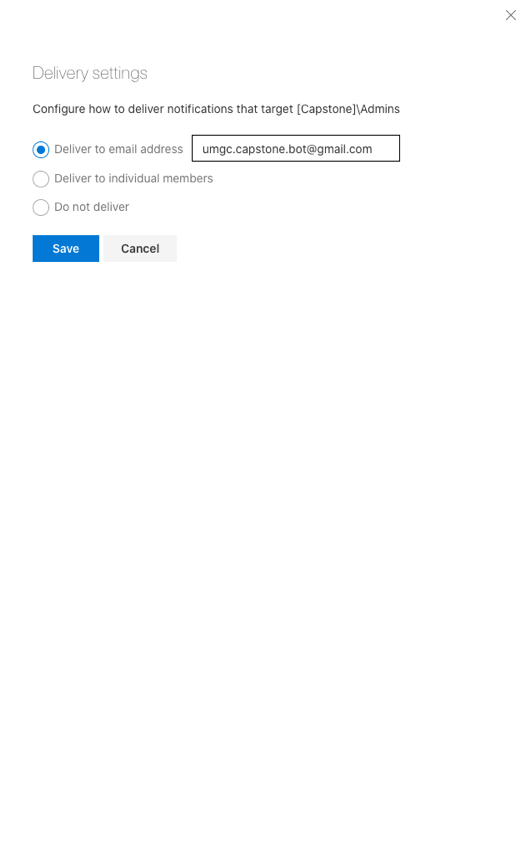


Figure 2. Azure DevOps Delivery Settings

1. Select **Deliver to email address,** enter email desired, and click **Save.**
2. Click **New subscription** to create a new email notification.
3. By default, build success and deployment failure emails are included. To create build failure notifications, click **Build** > **A build fails**, then **Next.**

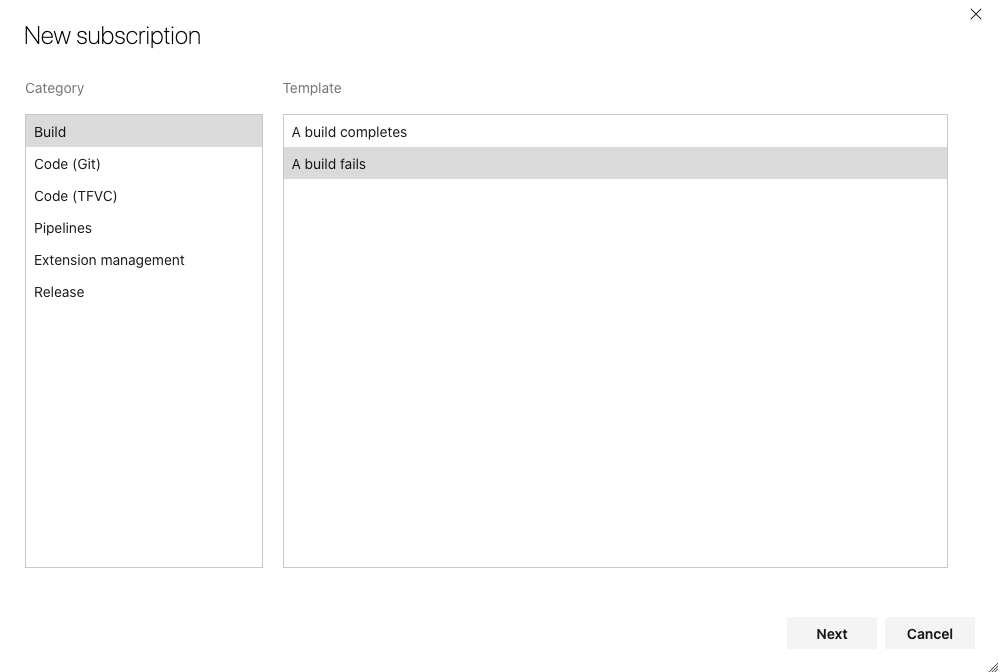


Figure 3. Azure DevOps Notification Subscription

1. In the Deliver to dropdown, select **Custom email address**, and enter email.
2. Under Filter, select **Any team project.**
3. Under the Field dropdown, select **Status**, then in the Value dropdown, select **Failed.**

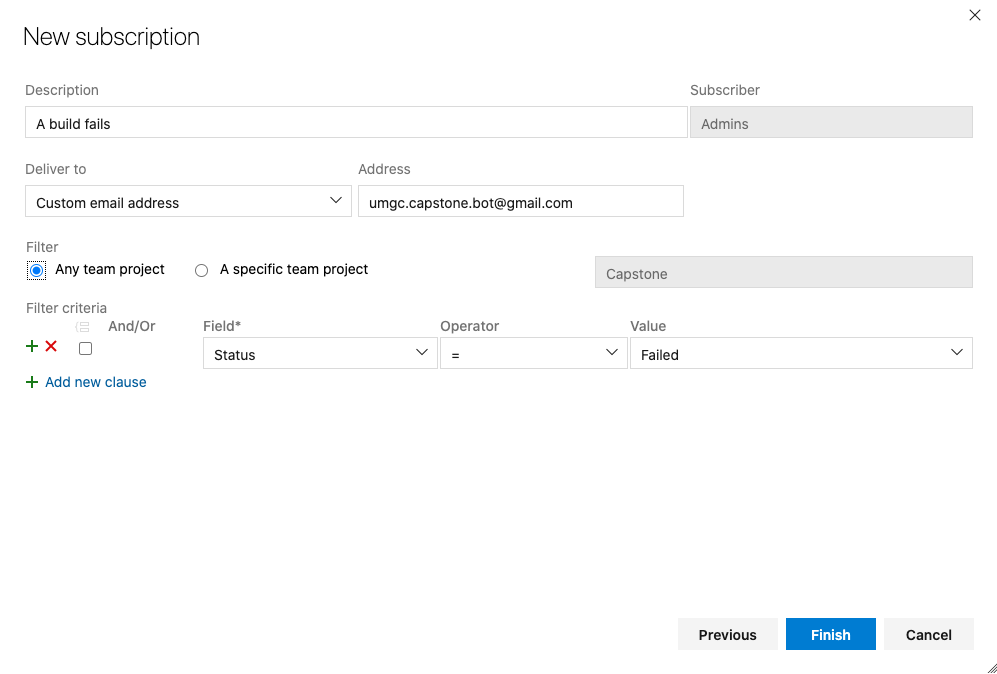


Figure 4. Azure DevOps Email Build Subscription

1. Click **Finish.**
2. Repeat the flow from step 8 to create notifications for deployments by choosing **Release** > **A deployment is completed** and creating a filter for **Deployment Status** being **Succeeded.**

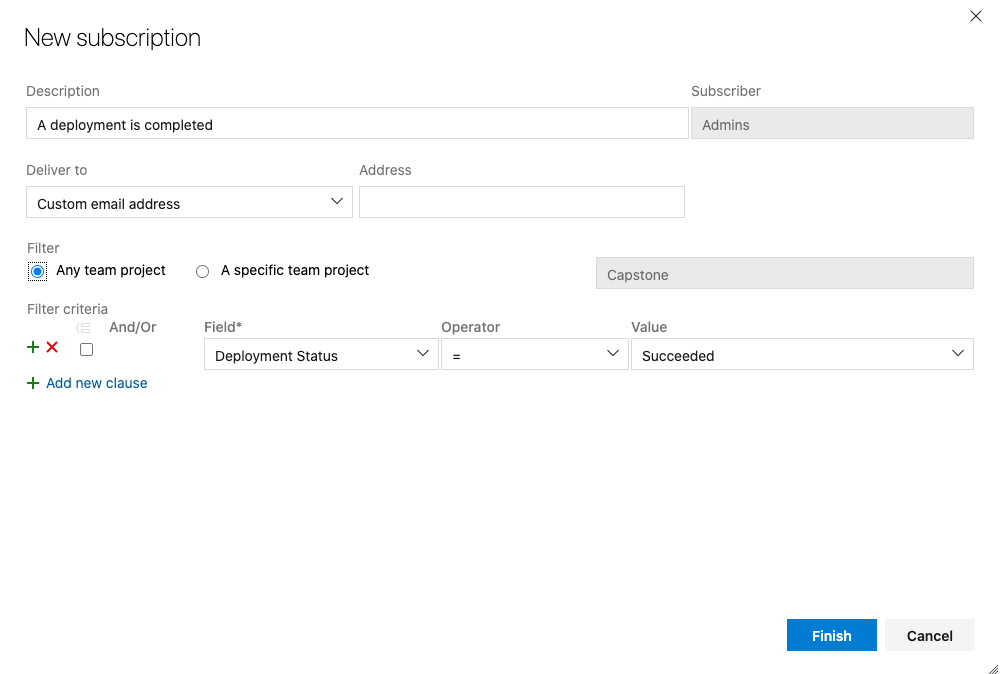


Figure 5. Azure DevOps Email Deployment Subscription

# 3. Add Code Quality Checks for Project

1. Log in to [www.sonarcloud.io](http://www.sonarcloud.io) with UMGC Capstone course admin account using provided credentials.
2. Click the **+ icon** at the top right of the page and select **Analyze new project.**

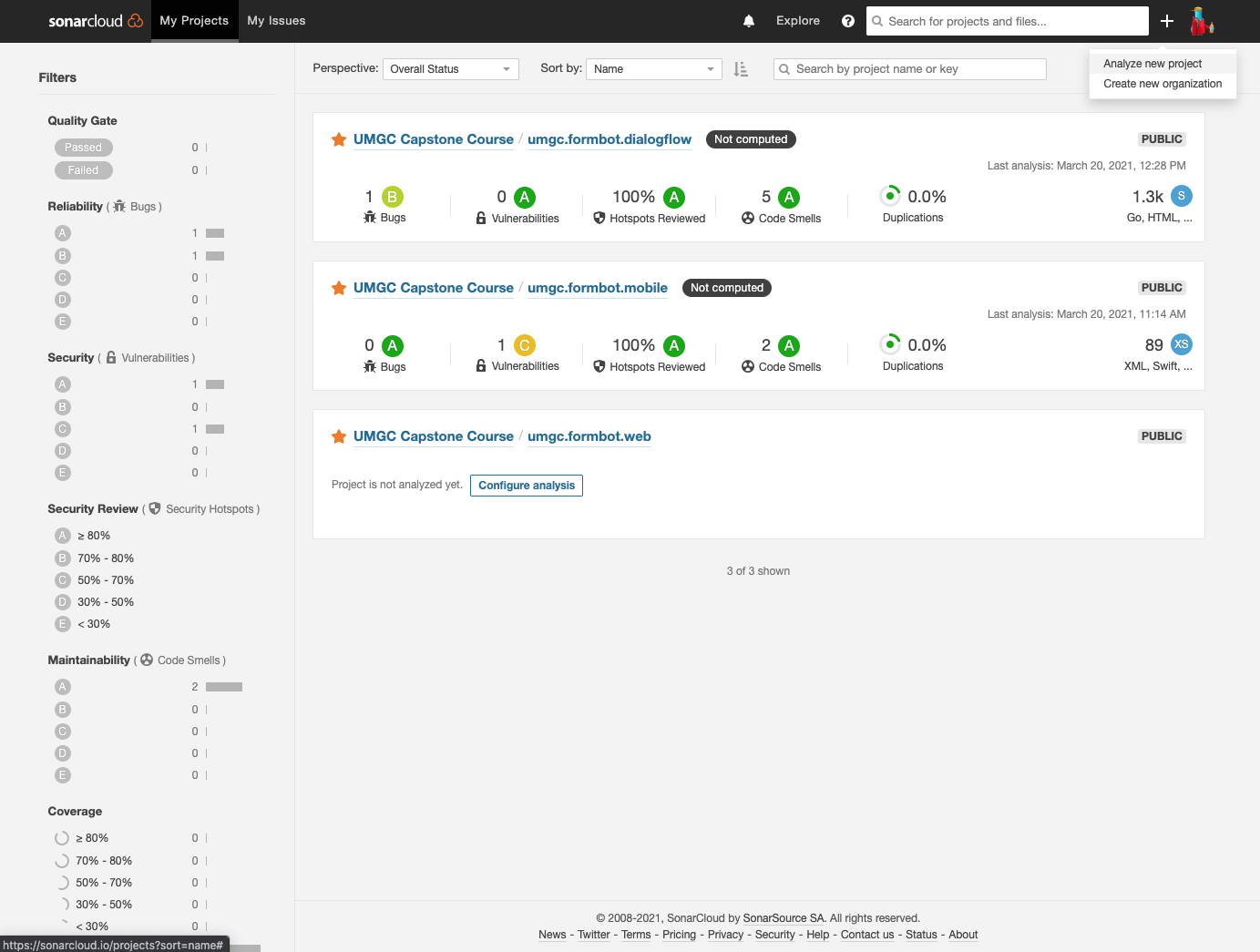
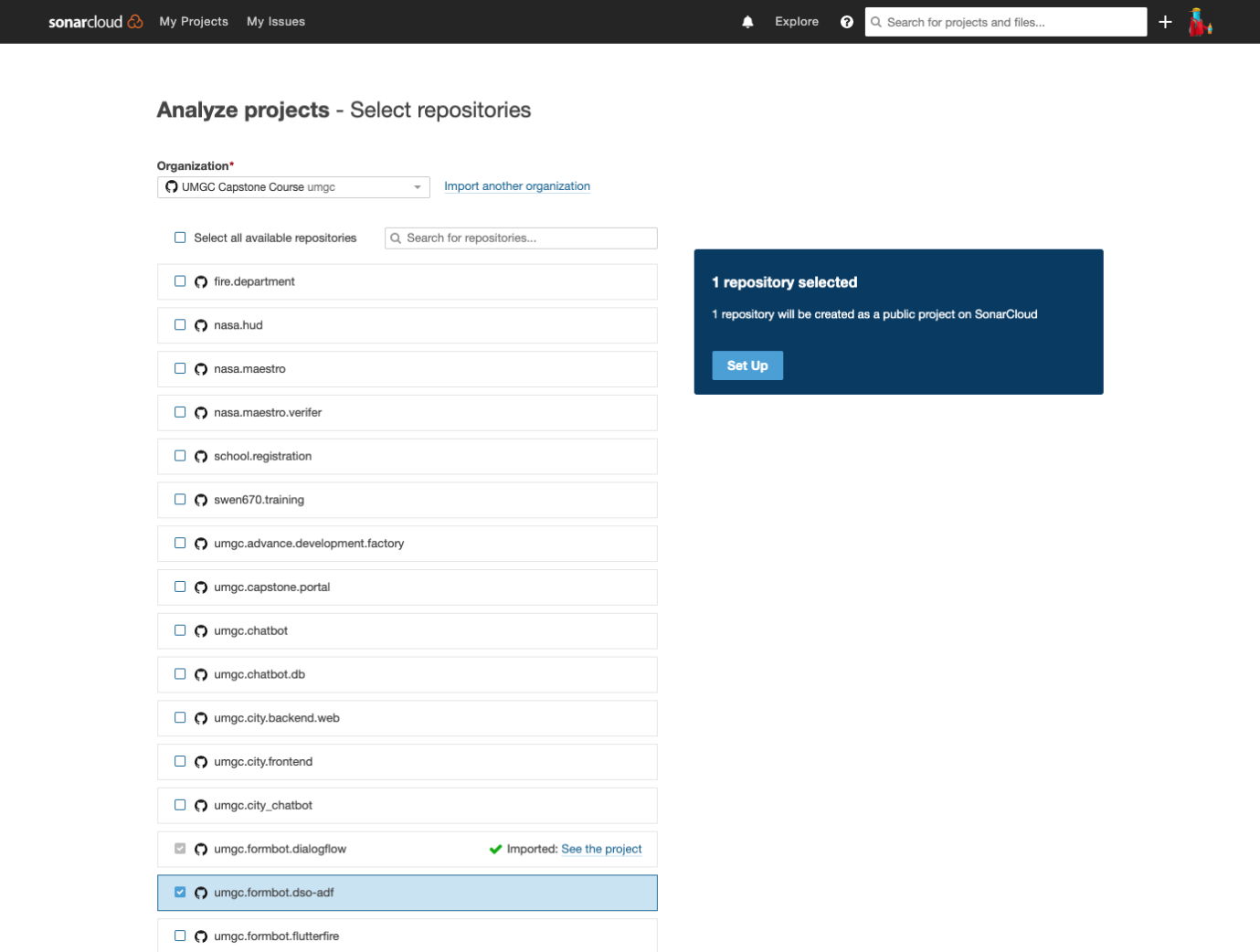


Figure 6. SonarCloud Project

1. In the **Organization** dropdown, select **UMGC Capstone Course** GitHub organization.
2. Check the box for the desired repository.
3. Click **Set Up.**

  
Figure 7. SonarCloud - Analyze Project

1. In the Projects page, select and deselect the star icon(s) for the repository to highlight as favorite.
2. Go to [www.sonarcloud.io/projects](http://www.sonarcloud.io/projects) to see the list of repositories for the semester/project.
3. Select the repository and click **Configure analysis** to enable code quality checks to be automatically run when pull requests are created.

# 4. Advance Development Factory (ADF) for Golang

Expanding upon the UMGC Capstone Fall 2020 semester’s DevSecOps framework, ADF, requires different configurations for different projects and programming languages as the original version was built to support projects using the Java programming language. As an example, to create an implementation for Golang, such as used by the FormScriber Dialogflow component, the following is done:

1. Create a new GitHub repository (refer to the FormScriber DevSecOps Runbook).

Note: The one created for Go in this example is accessible at the following location: <https://github.com/umgc/umgc.formbot.dso-adf>This version for Golang includes formatting and documentation improvements to all files, many of which can be shared with future versions of ADF. As this is the current working version and the original ADF is no longer under development, we will be mirroring this repository.

1. Create a mirror of the ADF repository.
   * Create a bare clone of the repository.

git clone --bare https://github.com/umgc/umgc.formbot.dso-adf.git

* + Push a mirror of the repository to the new repository.

cd umgc.formbot.dso-adf.git

git push --mirror https://github.com/githubaccount/new-repository.git

* + Remove the local repository (to avoid confusion).

cd ..

rm -rf umgc.formbot.dso-adf.git

* + Clone the new repository.

git clone https://github.com/githubaccount/new-repository.git

1. Modify **Makefile** to create a new name for the eventual Docker image to be created, using the **DOCKER\_NAME** variable, then save the file.

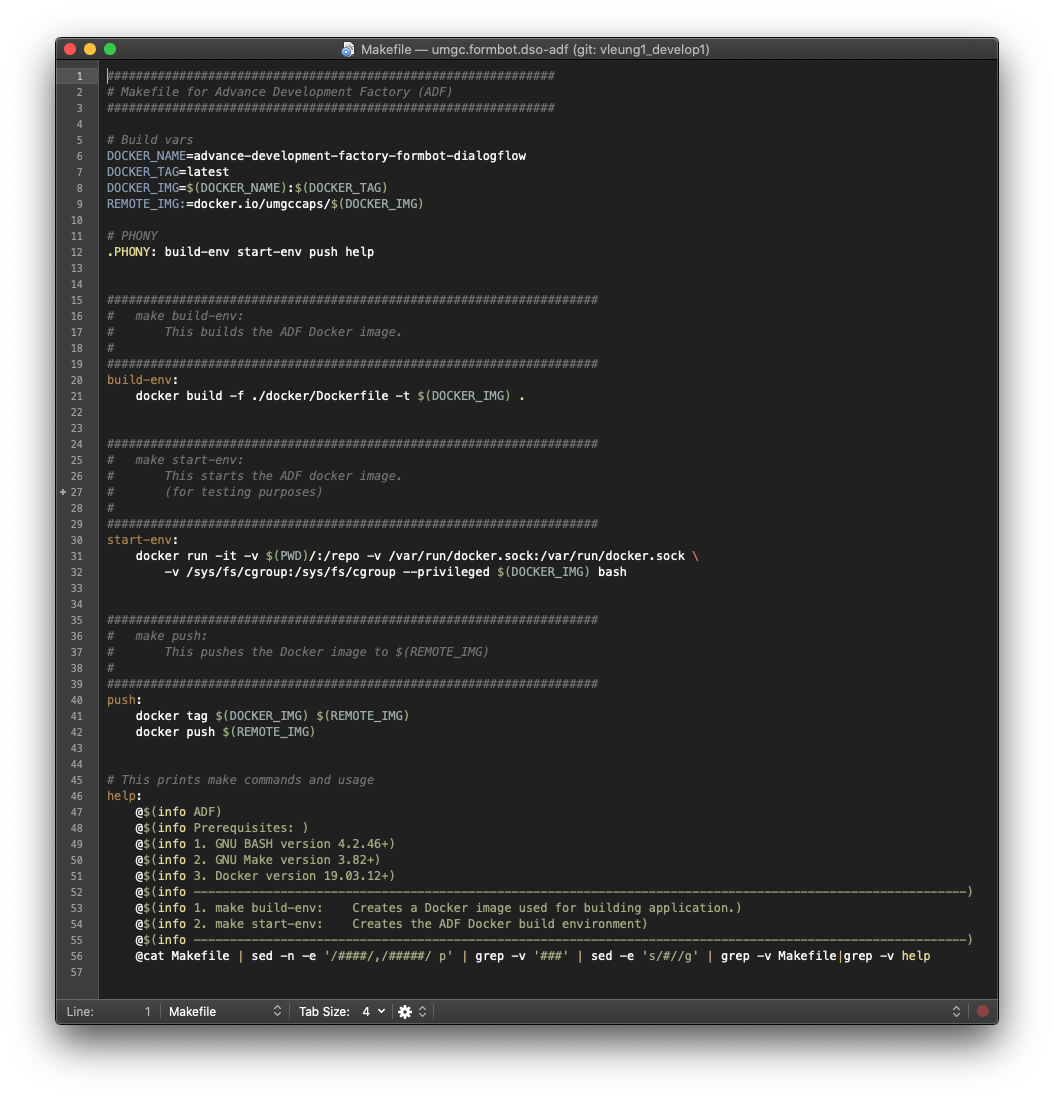


Figure 8. ADF Makefile

1. Modify **/docker/Dockerfile** to create the environment and dependencies needed to eventually build the application executable, using a Docker image that is closest to the environment needed.

In this example, we will use ‘golang’ image as the base image as it is the official supported Golang Docker image.

Within the Dockerfile, we will define the commands needed to install the dependencies using the package manager included in the image.

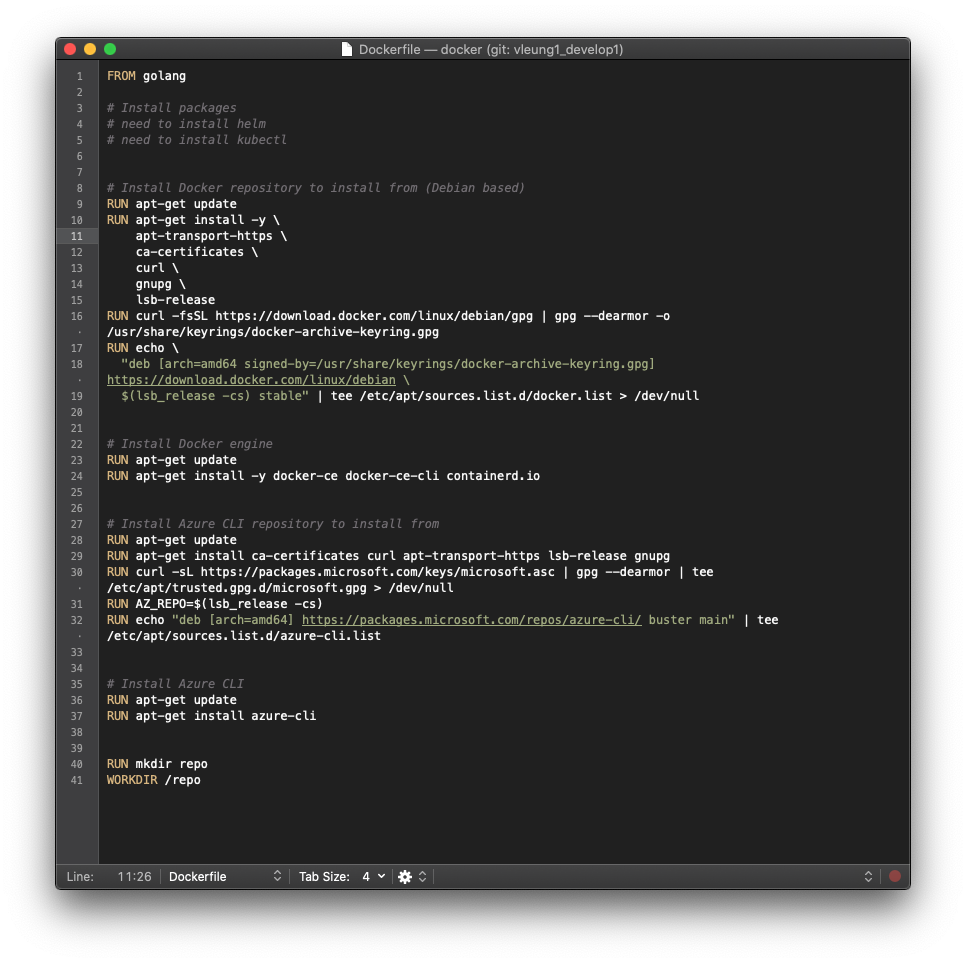


Figure 9. ADF Dockerfile

1. Some core dependencies may or may not be pre-installed in the Docker image you use, so if they are not, include them in the Makefile to install. Such dependencies include, but may not be limited to: **git**, **make**, **wget**, **which, Docker, Azure CLI**
2. Create the new ADF Docker image by running:

make build-env

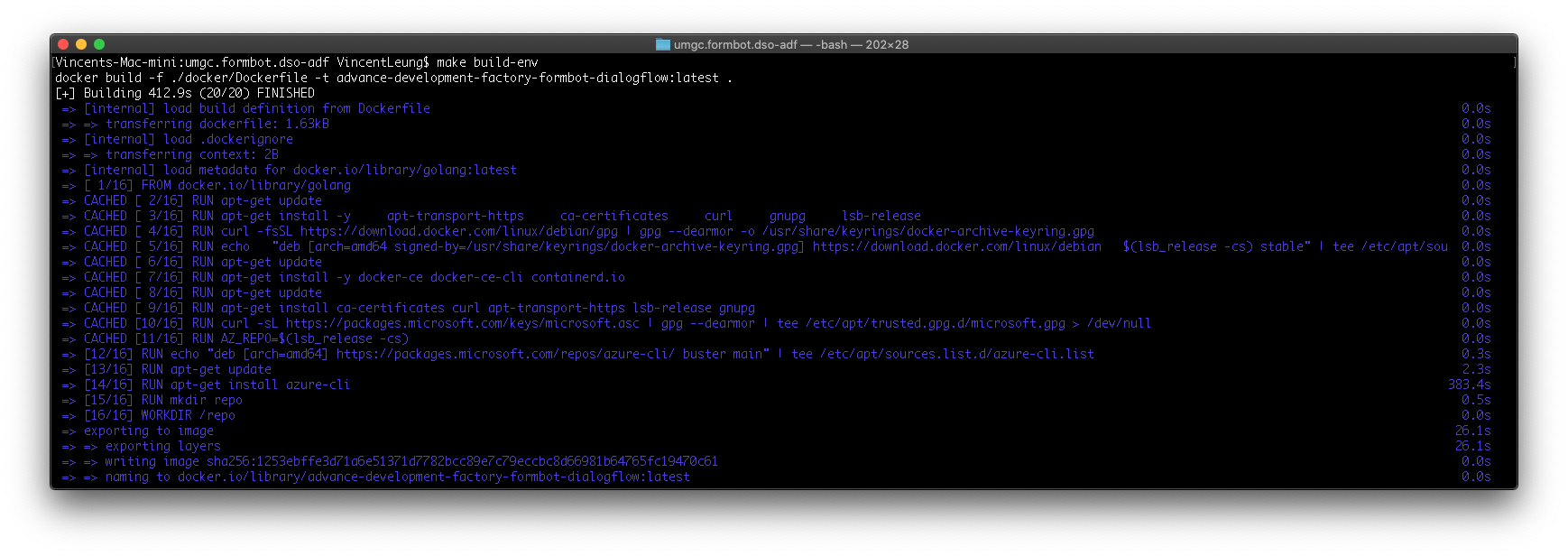


Figure 10. ADF Build Run

This should finish without errors.

1. Push the new ADF Docker image to docker.io.  
   * On your local machine, ensure you are logged in to the Docker client with the Capstone course’s Docker account using provided credentials.

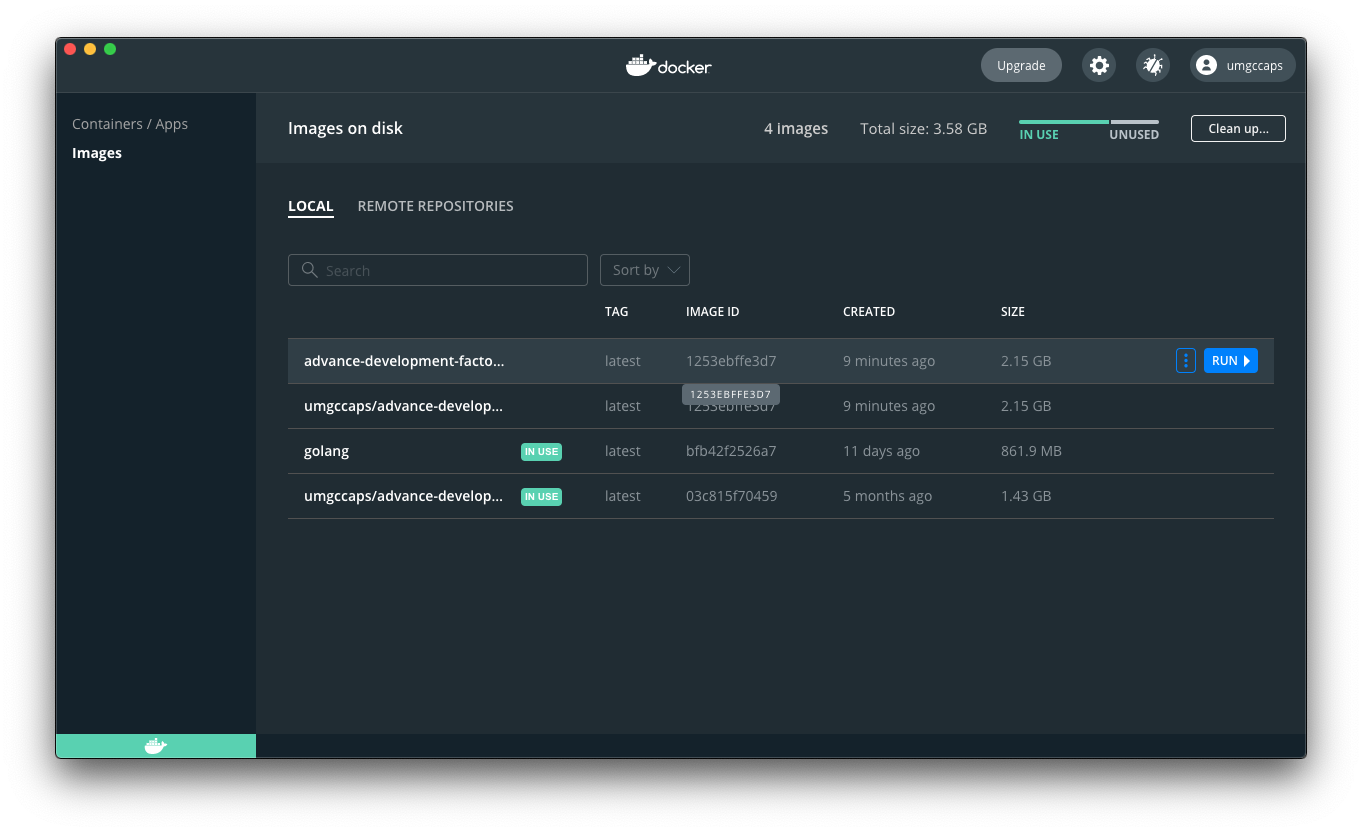


Figure 11. ADF Docker.io

Then run the make command:

make push

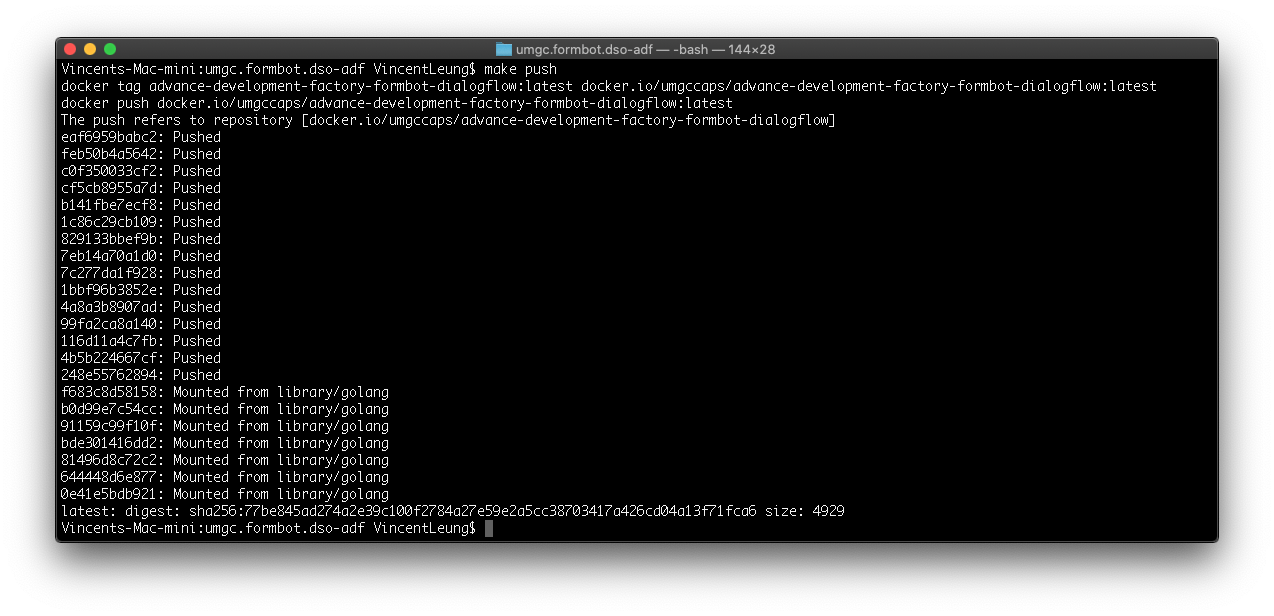
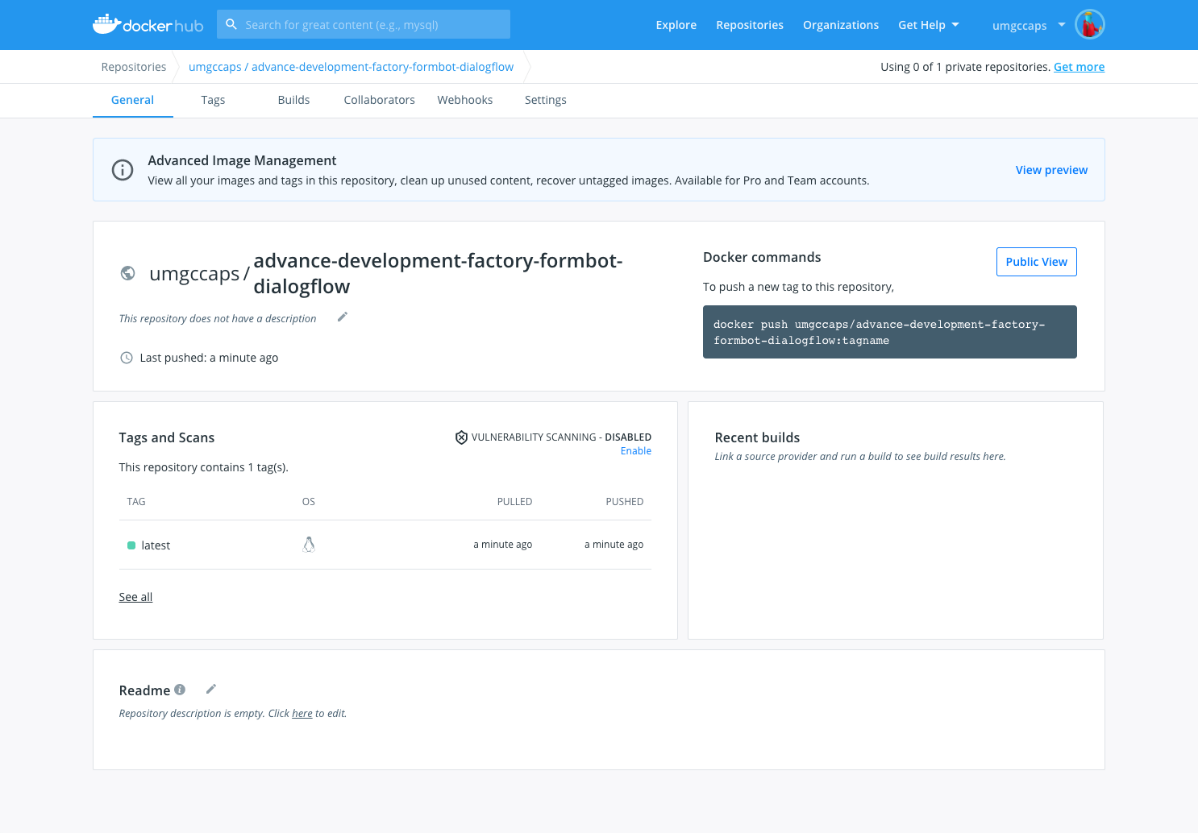


Figure 12. ADF Docker Push

This should finish without errors.

1. You should now be able to see the Docker image in a new repository on docker.io at: <https://hub.docker.com/repositories>
   * Log into the Capstone course account if needed.
   * This Docker image will be the image that the development team project will utilize to create their development environment that builds their application and creates the executable.

  
Figure 13. ADF Docker Image

1. To automate these ADF build and push processes, a .yaml configuration file is included in the **/azure/** directory. An automated build pipeline can be created using this file via the Azure DevOps suite (refer to the FormScriber DevSecOps Runbook).

When this is configured, whenever there is a pull request for ADF or merge to an integration branch, the pipeline will run to check for Docker image build success. Additionally, whenever there is a push to master, the pipeline will also log into Docker and push the new image to the Docker Hub repository.

Within the pipeline, you will have to configure Variables to account for the **HOST**, **USERNAME**, and **PASSWORD** to log in and push to Docker Hub.



Figure 14. Azure DevOps ADF

1. To implement this new ADF configuration in a development team’s project, refer to the UMGC Capstone Fall 2020 semester DevSecOps Programmer’s Guide.

For the development team’s project, the new ADF Docker image will be referenced in that project repository’s Makefile under the **BUILD\_IMG** variable, such as:

**BUILD\_IMG=docker.io/umgccaps/advance-development-factory-formbot-dialogflow:latest**

# 5. Access the Kubernetes Dashboard

Kubernetes Dashboard allows you to see the status and monitor the Kubernetes cluster(s) you have created via a web UI. Once an application has been deployed to Azure Kubernetes Service (AKS), it will be useful to access the Dashboard to see the resources.

1. Ensure that you have installed Azure CLI tool (refer to the **Set Up DevSecOps Environment** section of this guide).
2. Install kubectl in your CLI by running the command:

sudo az aks install-cli

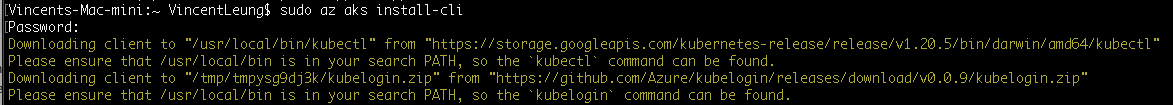


Figure 15. Azure CLI AKS Install

1. Get credentials to log into your cluster by running the command:

az aks get-credentials --resource-group <your-resource-group-name> --name <your-cluster-name>



Figure 16. Azure AKS Credentials

You may be prompted to log in first, if you have not already:

az login

Follow the on-screen prompts to log into your student subscription if needed.



Figure 17. Azure CLI Login

1. Access the Kubernetes Dashboard

az aks browse --resource-group <your-resource-group-name> --name <your-cluster-name>

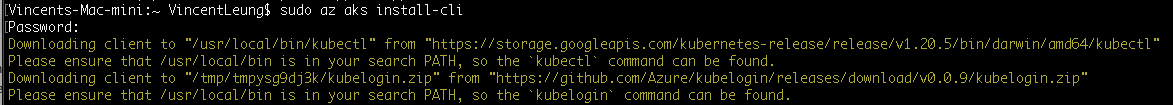


Figure 18. AKS - Install Azure CLI

The Dashboard will appear in a web browser.



Figure 19. Azure AKS Dashboard

# 6. Azure Release Pipeline to Azure Kubernetes Service

An Azure release pipeline can be created to continuously deploy an application to Azure Kubernetes Service, which we will use to create a Kubernetes cluster.

The release pipeline will utilize an artifact created by a previously set-up Azure build pipeline. The artifact is a Docker image created by the build pipeline and pushed to a Docker repository.

We will use Docker Hub as the image repository and integrate it with the Azure DevOps infrastructure via a service connection.

1. Ensure you are logged in to the Azure DevOps admin account or DevSecOps team account with granted permissions (see Runbook, Azure section).
2. Go to the UMGC Capstone project within Azure DevOps: <https://dev.azure.com/umgccapstone/Capstone>
3. Click **Project settings** on the bottom left of the page.
4. Click **Service connections** from the left menu.
5. Click the **New service connection** button, choose **Docker Registry** option, then click **Next**
6. Choosethe **Docker Hub** option, enter **umgccaps** as the Docker ID, enter the Docker ID password, and click **Verify** to check

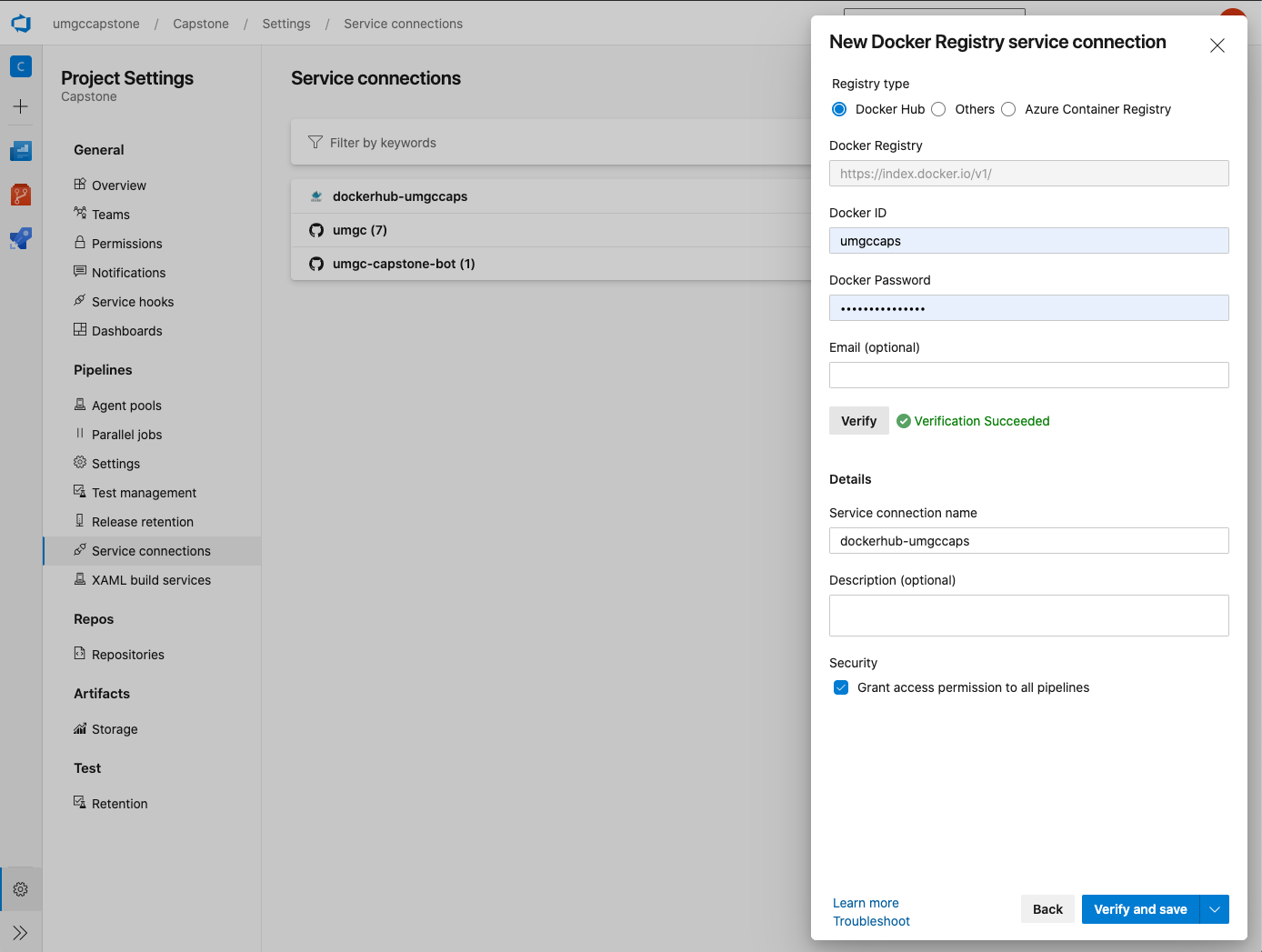


Figure 20. Azure DevOps Service Connection - Docker

1. Give the connection a name, and click **Verify and save**. You will now be able to use this connection to Docker Hub for the pipelines.
2. To start the Kubernetes setup, we will need to access Azure CLI command line tool. You can do this via your local machine command prompt or terminal if you have Azure CLI installed (refer to the **Access the Kubernetes Dashboard** section of this guide).   
     
   Alternatively, you can access the Azure Portal at https://portal.azure.com/ and click on the **Cloud Shell command prompt icon** in the top navigation bar. This will bring up an Azure CLI interactive shell within the web browser.
3. Create the resource group by running:

az group create --name <your-resource-group-name> --location eastus

1. Create a container registry.

az acr create --resource-group <your-resource-group-name> --name <your-container-registry-name> --sku Basic

1. Create a Kubernetes cluster

az aks create \

--resource-group <your-resource-group-name> \

--name <your-container-registry-name> \

--node-count 1 \

--enable-addons monitoring \

--generate-ssh-keys \

--kubernetes-version 1.18.14

Wait for 5-10 minutes for the cluster to deploy, and once it is complete, within the **Project settings** > **Service connections** page, you will see the service connections.

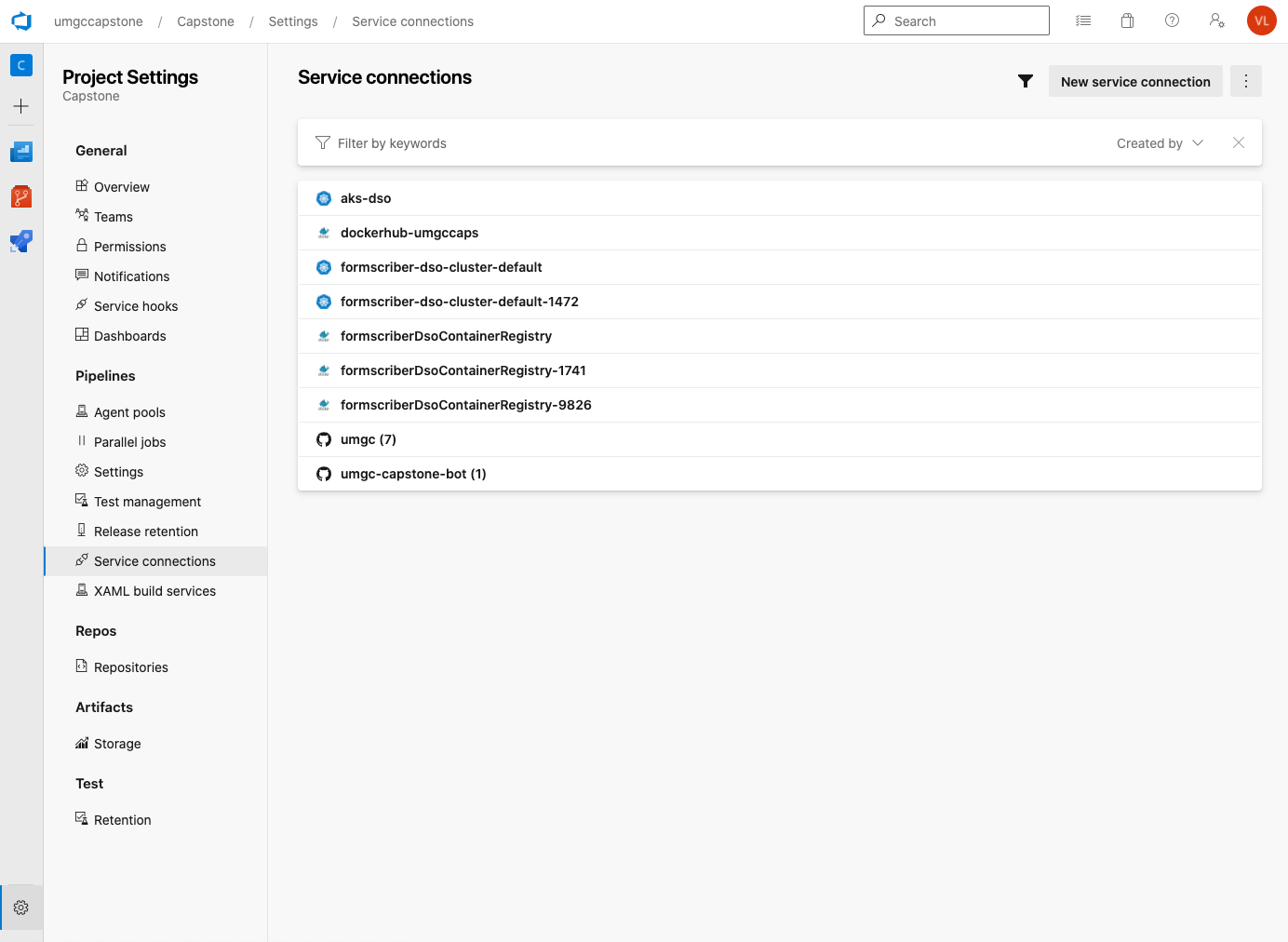


Figure 21. Azure DevOps Service Connections

1. Set up a build pipeline to build and push a Docker image of the application to Docker Hub.

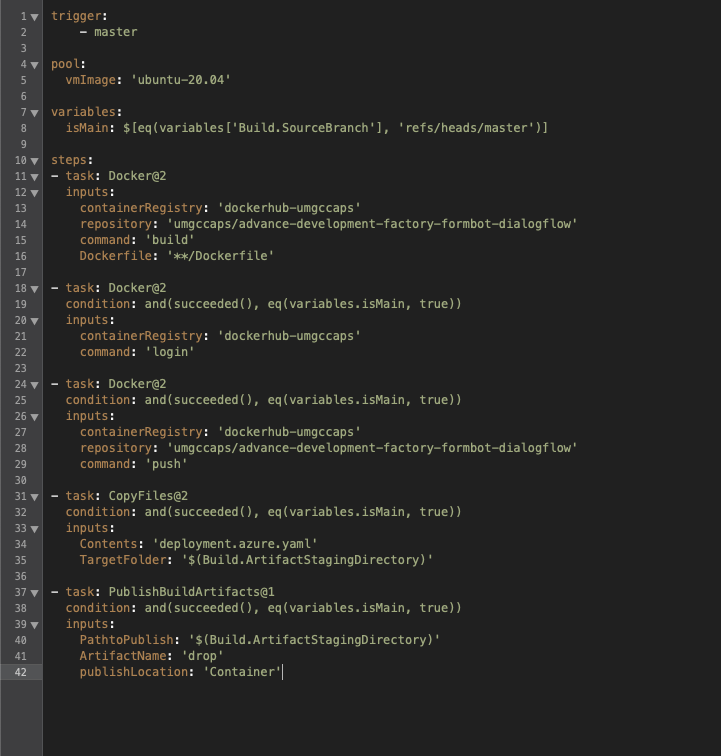
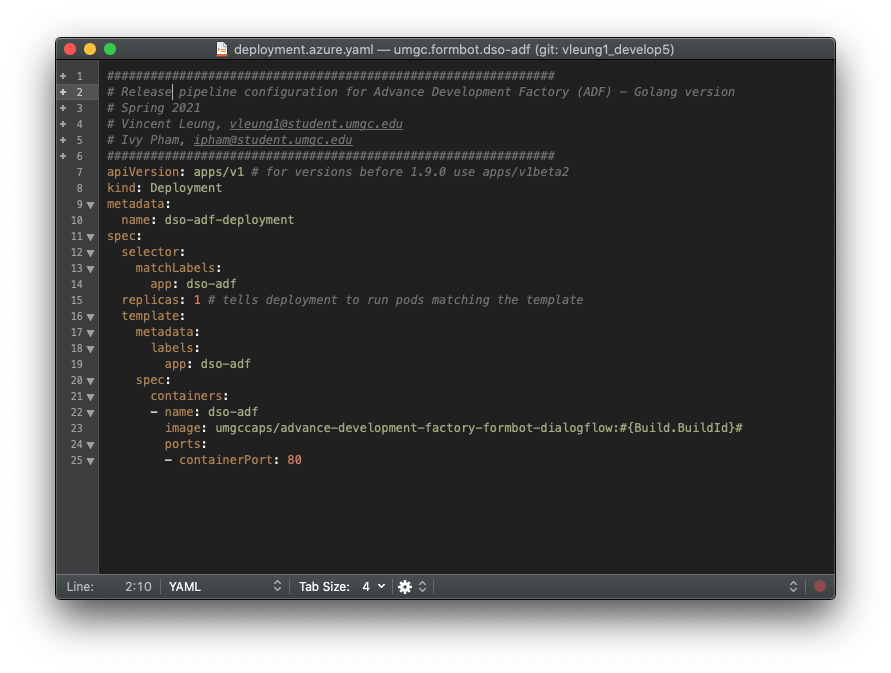


Figure 22. Azure DevOps Build Pipeline Docker Push

1. Using the created Kubernetes cluster and service connections above, configure a release pipeline to deploy the application. We will use a **kubectl apply** command on an Azure deployment yaml configuration file.



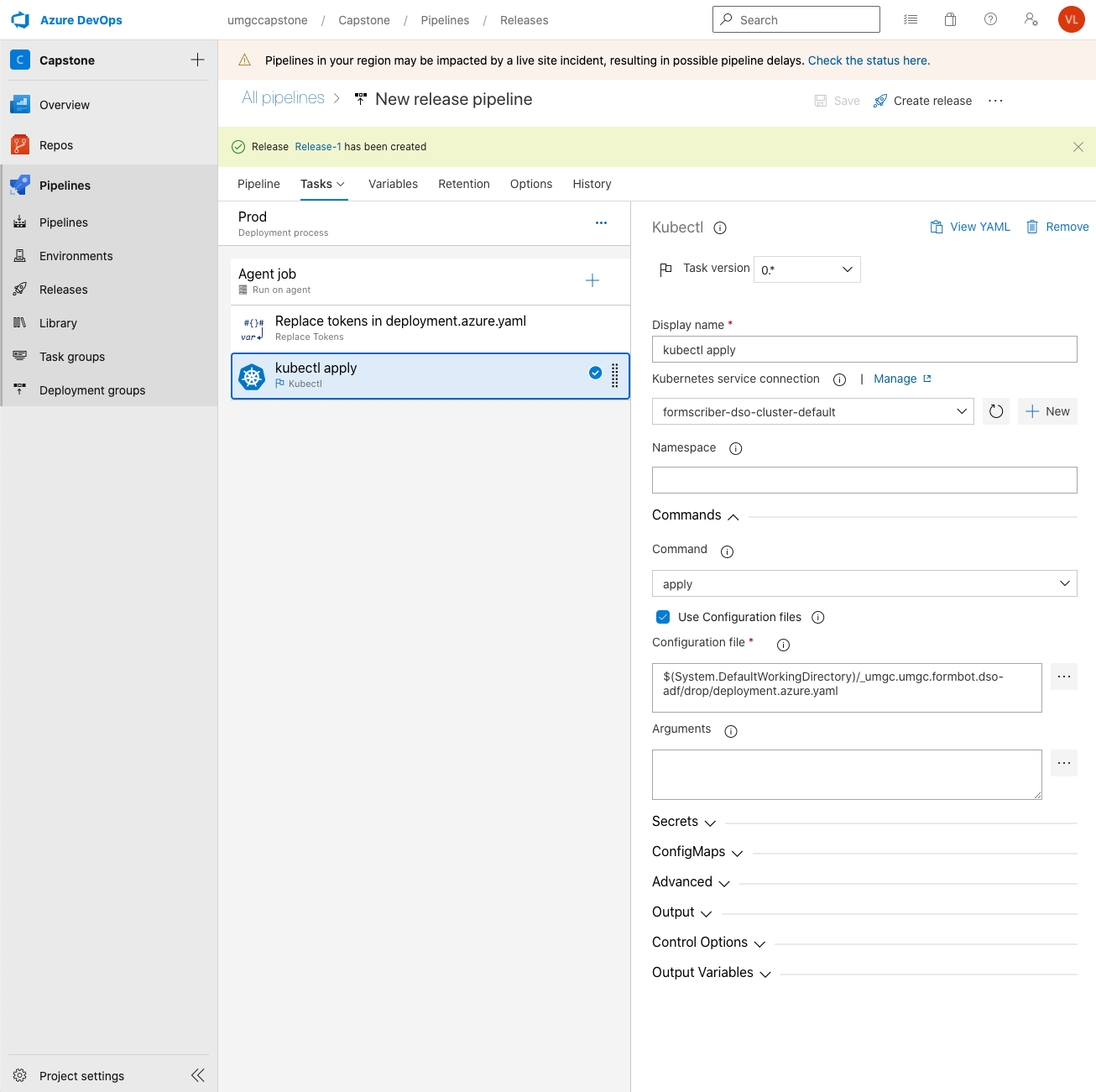


Figure 23. Azure DevOps Release Pipeline - AKS

1. You can now run the release pipeline by clicking **Create release**. To enable continuous deployment, set the trigger (refer to FormScriber DevSecOps runbook).

# 7. Azure Build Pipeline Mobile (Flutter/Dart)

Form Scriber is comprised of two major components which include Mobile and Dialogflow. Mobile team has decided to build the mobile component with Flutter. To automate the build of mobile component in Azure DevOps, a community plugin is required as Microsoft does not a native support for Flutter in Azure DevOps.

1. To install the Flutter plugin, please login to <https://dev.azure.com> with the organizational administrative account (umgc.capton.bot) and go to **Organization Settings**.
2. Click on **Extensions** and **Browse marketplace**.
3. Search for “Flutter” and select “Flutter Tasks” by Hey24sheep which is actively maintained by the community.

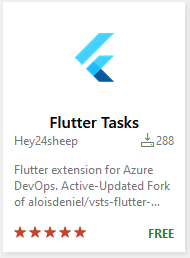


Figure 24. Azure DevOps Flutter Tasks Plugin

1. Select **Get it Free** and **Install.**

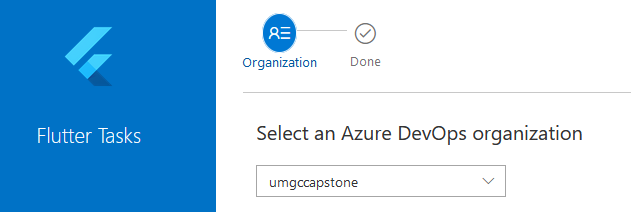


Figure 25. Azure DevOps Flutter Tasks Plugin Install

1. Azure DevOps Extensions should show all installed plugins.

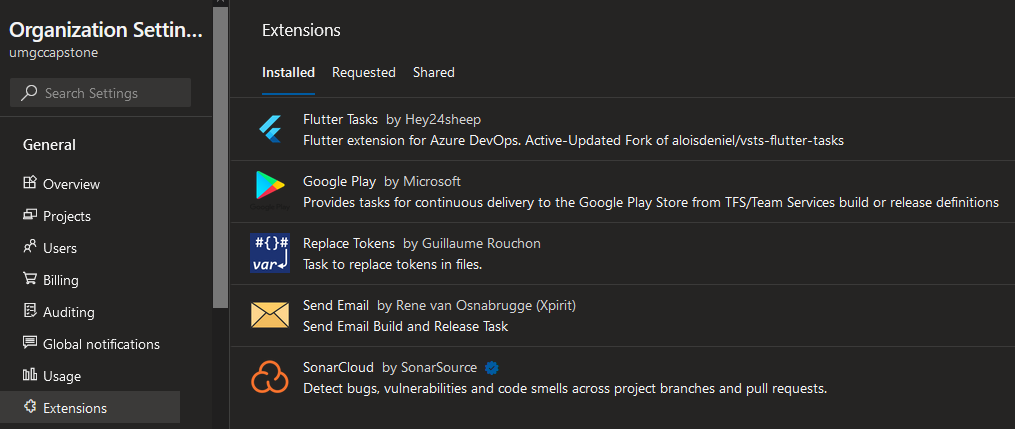


Figure 26. Azure DevOps Extensions

1. Click on **New pipeline** from **Pipelines** tab.
2. Select **GitHub** as the code repository.
3. As the last semester has already authorized the connection from Azure DevOps to UMGC GitHub repository, a list of all repositories is shown.
4. Change the drop down to **All repositories** and select **umgc/umgc.formbot.mobile.**

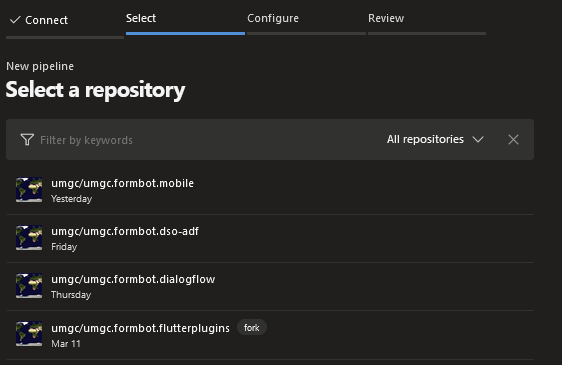


Figure 27. Azure DevOps GitHub Repositories

1. Select **Starter pipeline**.

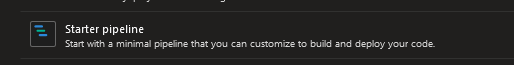


Figure 28. Azure DevOps Starter Pipeline

1. Insert the appropriate tasks (Install Flutter Tools, Build, Test and Publish) in Figure 29.

Note: At this point of writing, developers are working on the automatic testing. The Flutter Test Task can be activated by uncommenting the section when automatic testing is ready.

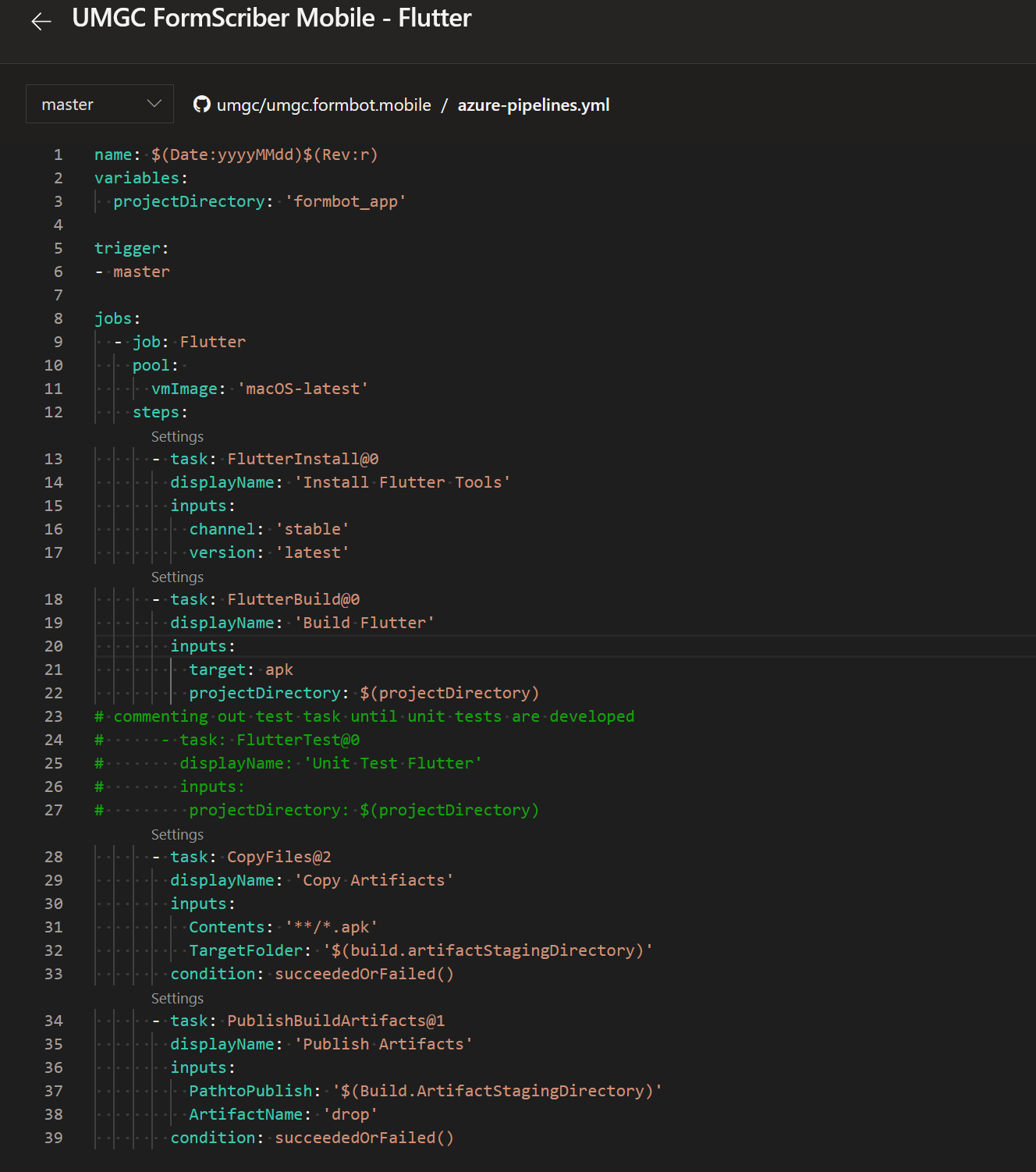


Figure 29. Azure DevOps Form Scriber (Flutter) YML Config File

1. Optional. The pipeline supports the Android Signing of the APK file. Please see below for future implementation.

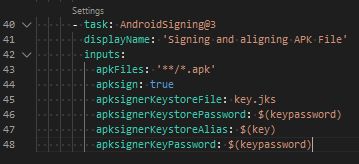


Figure 30. Azure DevOps Pipeline Flutter APK Android Signing

1. The pipeline is configured in Azure to automatically trigger when a pull is requested on the GitHub mobile repository. GitHub is also configured to trigger the pipeline when a pull is requested on any branch in the repository. Please see Figure 31 for a successful run.

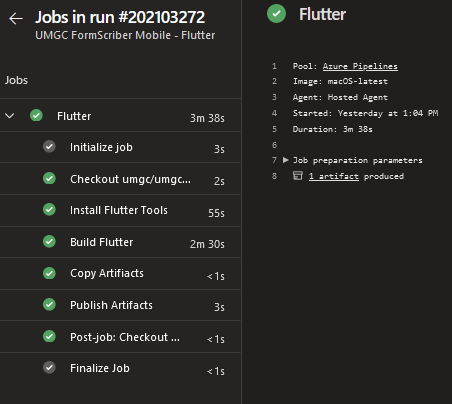


Figure 31. Azure DevOps Flutter Job

1. The pipeline can be triggered manually by clicking on **Run pipeline** in Mobile Flutter pipeline. Pipeline can be triggered against any branches in the repository.

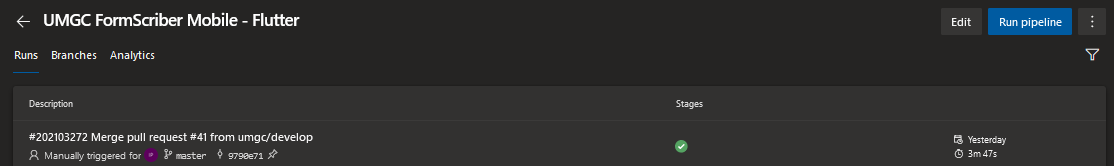


Figure 32. Azure DevOps Flutter Manual Trigger

1. Logs can be downloaded to troubleshoot any issues in the pipeline. Please see Figure 33.

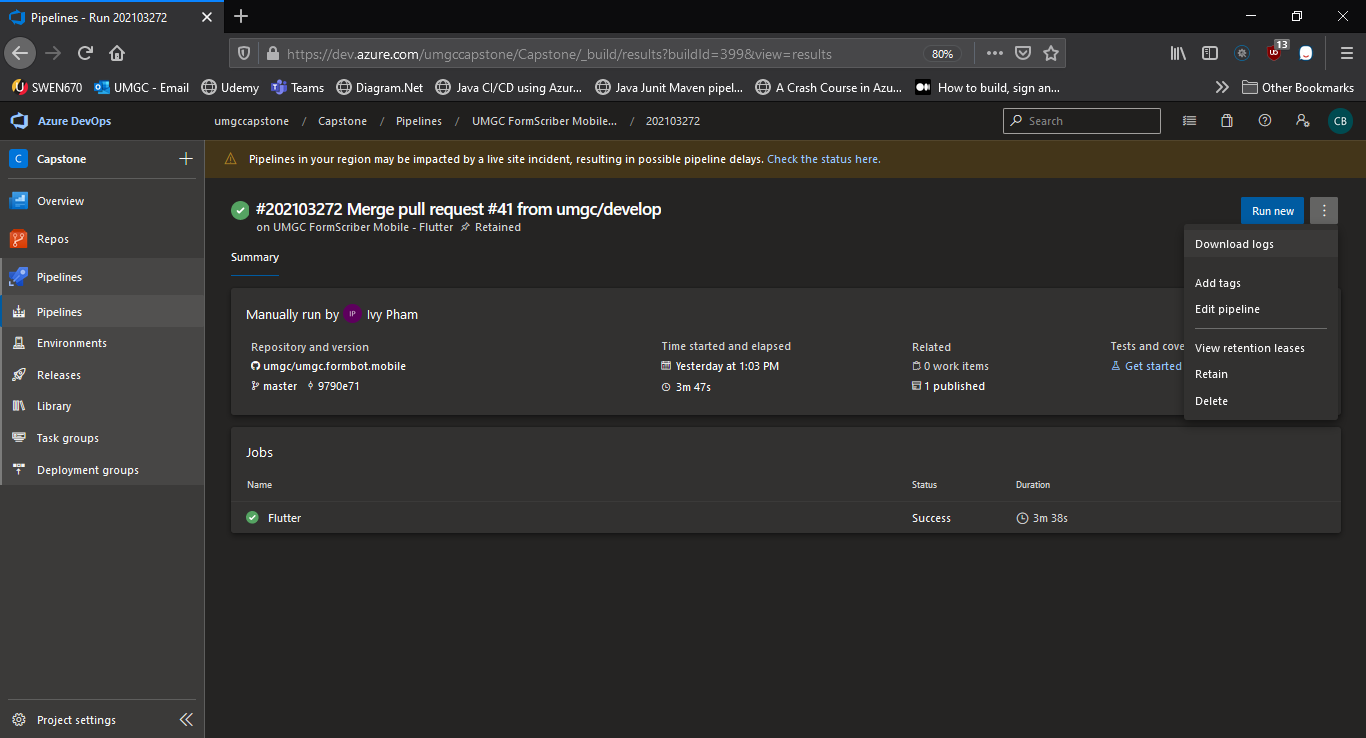


Figure 33. Azure DevOps Pipeline Logs

1. When modifying **azure-pipelines.yml** directly on the Azure DevOps UI, please make sure to save to a new feature branch and follow GitHub pull request process to merge to Master.

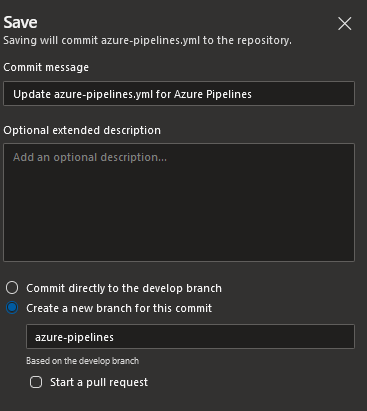


Figure 34. Azure DevOps YML Mod

# 8. Azure Build Pipeline Dialogflow (Golang)

As mentioned before, Form Scriber is comprised of two major components which include Mobile and Dialogflow. Dialogflow team has decided to build the component with Golang and small part of JavaScript. Golang and JavaScript are supported natively in Azure DevOps. Therefore, no plugin is required.

1. Click on **New pipeline** from **Pipelines** tab.
2. Select **GitHub** as the code repository.
3. As the last semester has already authorized the connection from Azure DevOps to UMGC GitHub repository, a list of all repositories is shown.
4. Change the drop down to **All repositories** and select **umgc/umgc.formbot.dialoglow.**

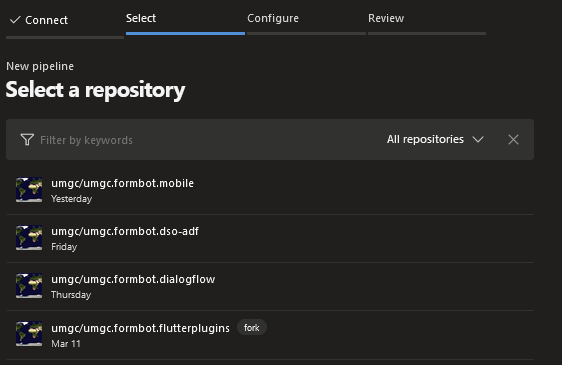


Figure 35. Azure DevOps GitHub Repositories

1. Select Starter pipeline.
2. Insert appropriate tasks

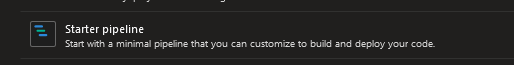


Figure 36. Azure DevOps Starter Pipeline

1. Insert the appropriate tasks (Install Go & JS Tools, Build, Test and Publish).

Note: At this point of writing, developers are working on the automatic testing. The Go Test Task can be activated by uncommenting the section when automatic testing is ready. Figure 37 shows all tasks for Golang. Figure 38 shows tasks for JavaScript.

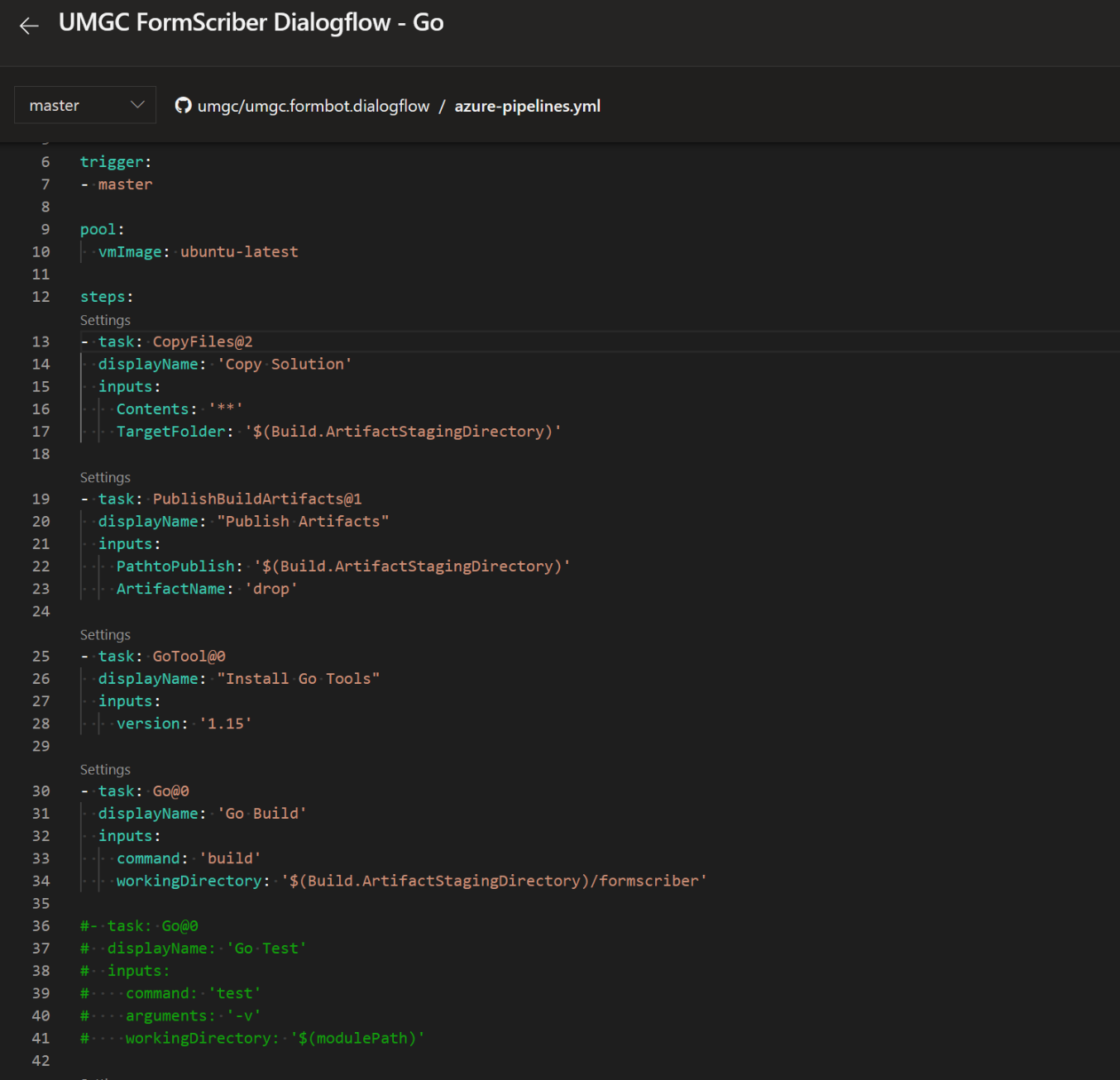


Figure 37. Azure DevOps Form Scriber (Golang) YML Config File

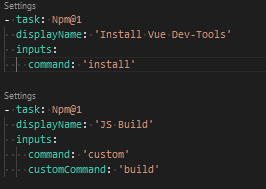


Figure 38. Azure DevOps Form Scriber (JavaScript) YML Config File

1. The pipeline is configured in Azure to automatically trigger when a pull is requested on the GitHub mobile repository. GitHub is also configured to trigger the pipeline when a pull is requested on any branch in the repository. Please see Figure 39 for a successful run.

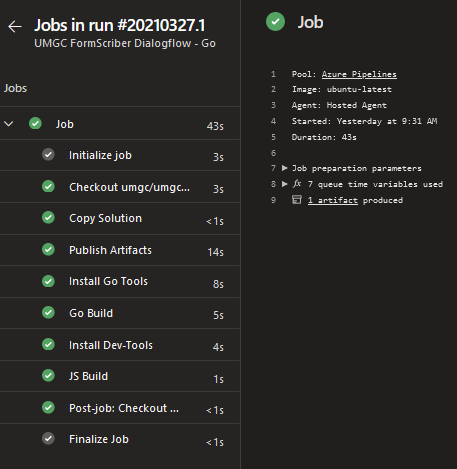


Figure 39. Azure DevOps Go/JS Job

1. The pipeline can be triggered manually by clicking on **Run pipeline** in Mobile Flutter pipeline. Pipeline can be triggered against any branches in the repository.

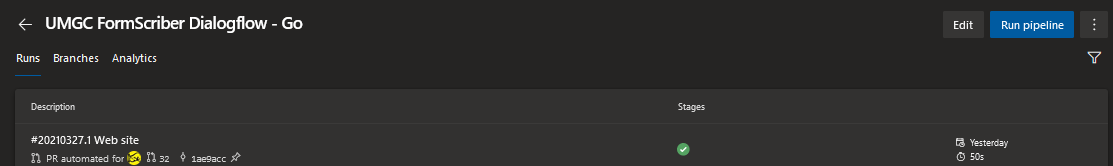


Figure 40. Azure DevOps Go/JS Manual Trigger

1. Logs can be downloaded to troubleshoot any issues in the pipeline. Please see Figure 41.

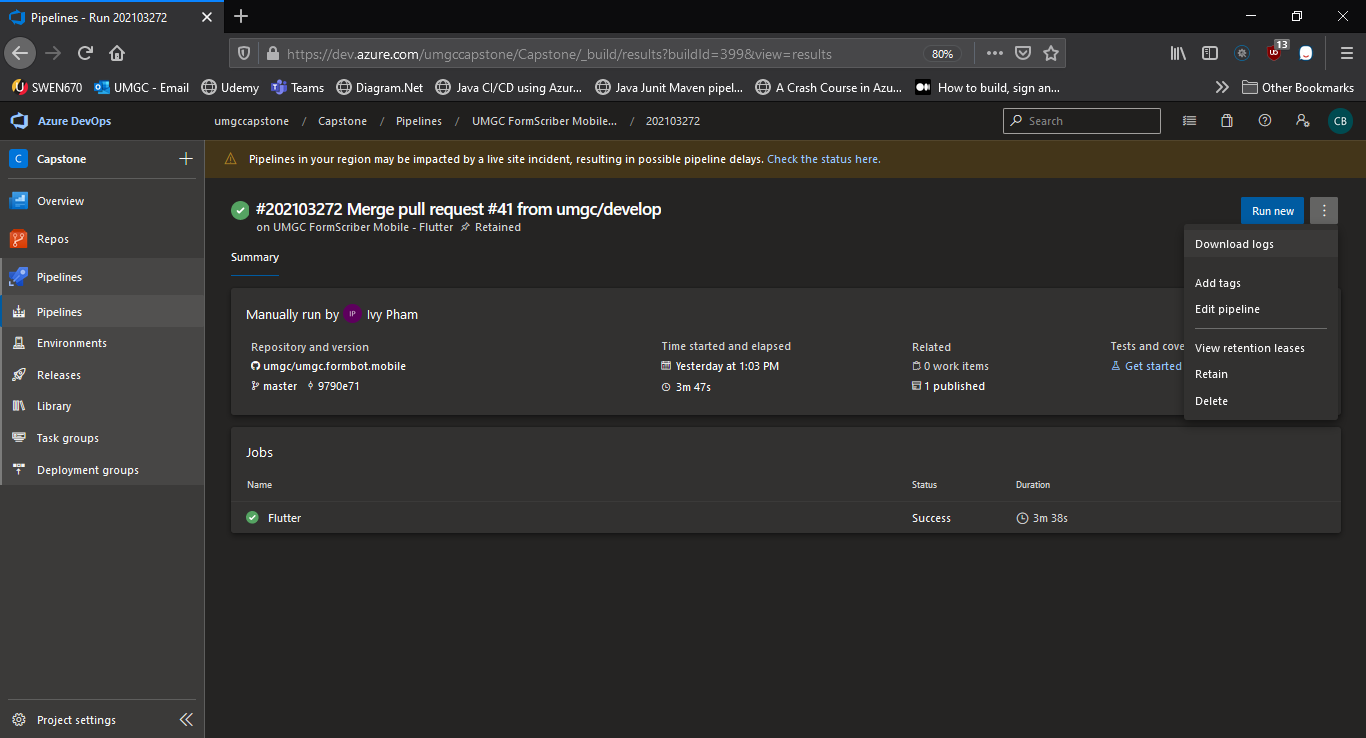


Figure 41. Azure DevOps Pipeline Logs

1. When modifying **azure-pipelines.yml** directly on the Azure DevOps UI, please make sure to save to a new feature branch and follow GitHub pull request process to merge to Master.

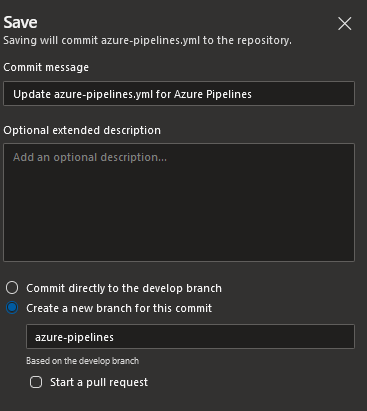


Figure 42. Azure DevOps YML Mod

# 9. Advance Development Factory (ADF) – Dialogflow/AKS Deployment

ADF is being deployed to provide standard tools and support for the development team. The ADF docker container is updated and pushed regularly to **docker.io**. The docker image can be pulled to developer’s desktop to support their development tasks such as build, test, configure and push the app to AKS. As mentioned previously, the developers will only need to install GIT and Docker to be able to use the docker container to perform those tasks mentioned above. Azure account is also required to deploy the app to Azure AKS. For the steps below, assuming Linux is being used.

1. Clone the GitHub repository (https://github.com/umgc/umgc.formbot.dialogflow) using the preferred tool.
2. DSO leverages the existing Makefile from Dialogflow application to write additional commands to make it easier, faster and more automated.
3. Please see figure 43 and 44 for the Makefile.

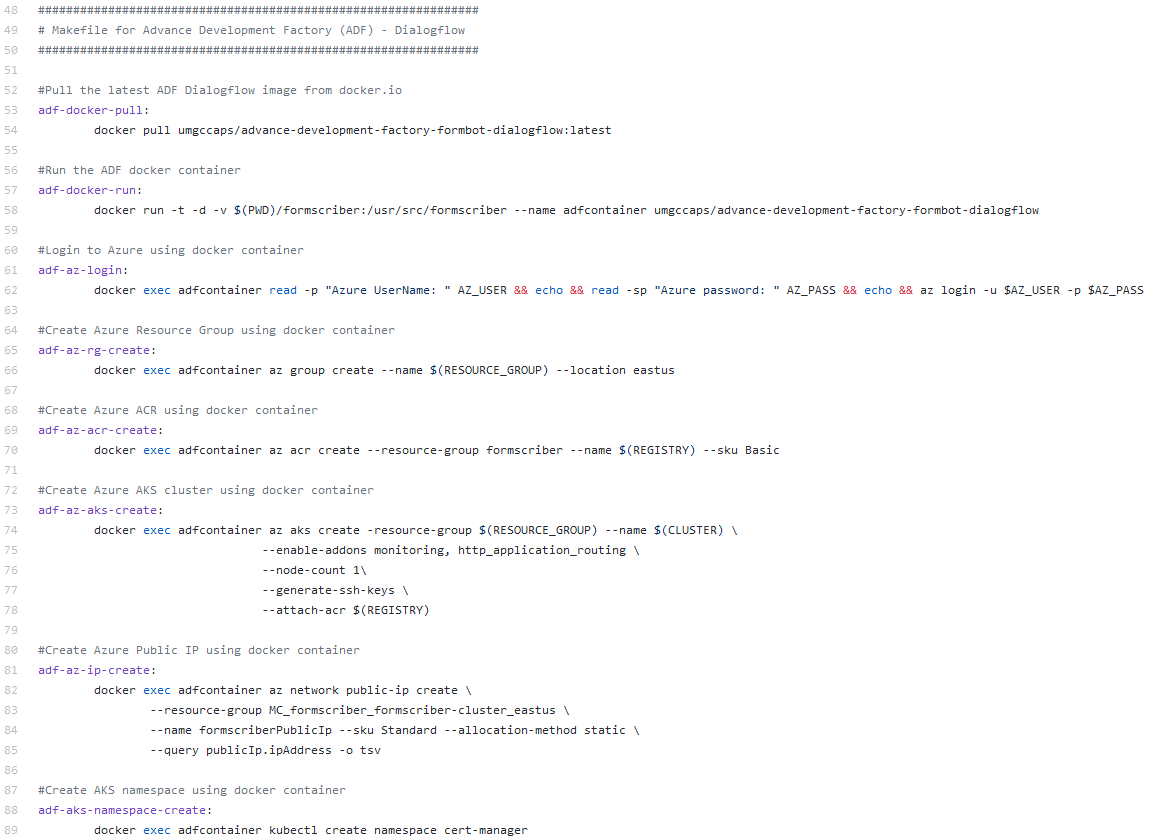


Figure 43. DSO/AKS Deployment Makefile

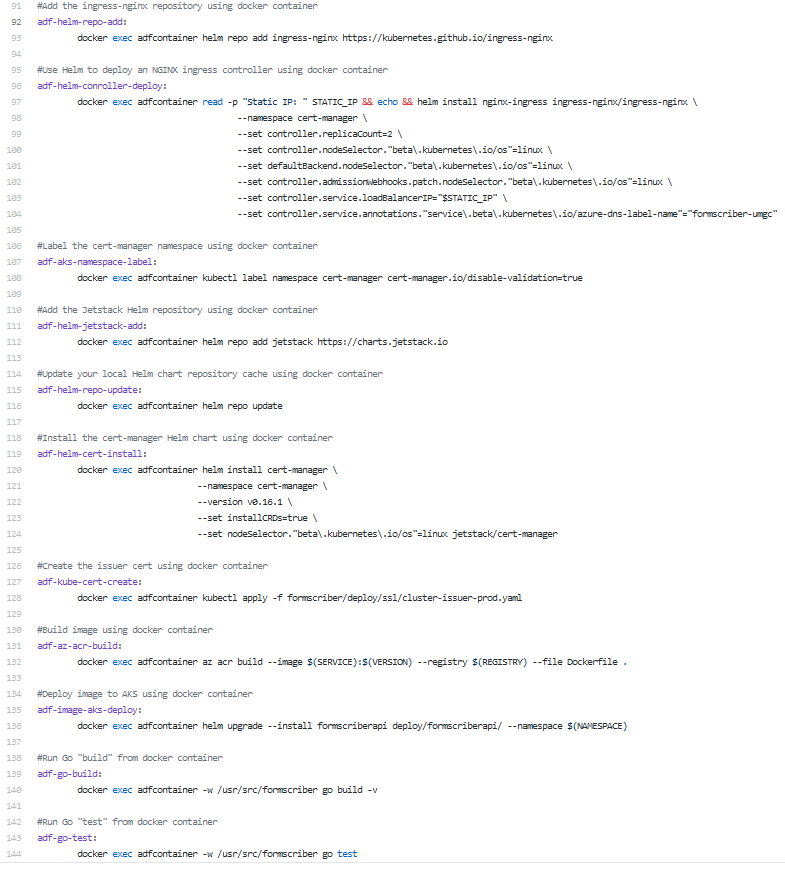


Figure 44. DSO/AKS Deployment Makefile

Note: All commands can be combined to run all at one time. For the purpose of documentation each step is executed individually to provide the explanation of what it does.

1. To pull the ADF docker image from **docker.io**:

make adf-docker-pull

1. To run the ADF docker container and map the application directory, run below command from the root of the project directory:

make adf-docker-run

1. To run the build task on the application:

make adf-go-build

1. To run the test task on the application:

make adf-go-test

1. To login to Azure:

make adf-az-login

1. To create Azure Resource Group:

make adf-rg-create

1. To create Azure Container Registry:

make adf-az-acr-create

1. To create Azure Kubernetes Cluster:

make adf-az-aks-cluster

1. To create Azure public IP:

make adf-az-ip-create

1. To create AKS namespace:

make adf-namespace-create

1. To add the ingress-nginx:

make adf-helm-repo-add

1. To use Helm to deploy a nginx ingress controller:

make adf-helm-controller-deploy

1. To label the cert-manager namespace:

make adf-aks-namespace-lable

1. To add the Jetstack Helm repository:

make adf-helm-jetstack-add

1. To update local Helm chart repository:

make adf-helm-repo-update

1. To install the cert-manager Helm chart:

make adf-helm-cert-install

1. To create the issuer certificate:

make adf-kube-cert-create

1. To deploy docker image to AKS:

make adf-az-acr-build

1. To delete AKS cluster

make adf-az-aks-delete

1. To delete the ADF docker container

make adf-docker-delete

# 10. Azure Release Pipeline – Google Play Store Push

At the time of this document is written, developers have not provided the JSON authorization file. DSO provides the instruction below for future works.

1. Create the JSON authorization file from Google Play:
   * Navigate to **Google Play Console** with owner role and login
   * Select **Settings** from the menu.
   * From Developer account menu choose **API access Sub Menu**.
   * At the bottom section **Service Accounts** choose **Create Service Account**.
   * Navigate to **Google Cloud – Service accounts**.
   * Press on **+Create service account**.
   * Add service account name.
   * To grant full access, choose for role: **Project -> Owner**.
   * At the bottom under Key section, choose **Add Key**.
   * Choose **JSON** which is downloaded automatically to local computer.
2. Install the Google Play plugin for Azure DevOps:
   * To install the Google Play plug-in, please login to https://dev.azure.com with the organizational administrative account (umgc.capton.bot) and go to **Organization Settings**.
   * Click on **Extensions** and **Browse marketplace**.

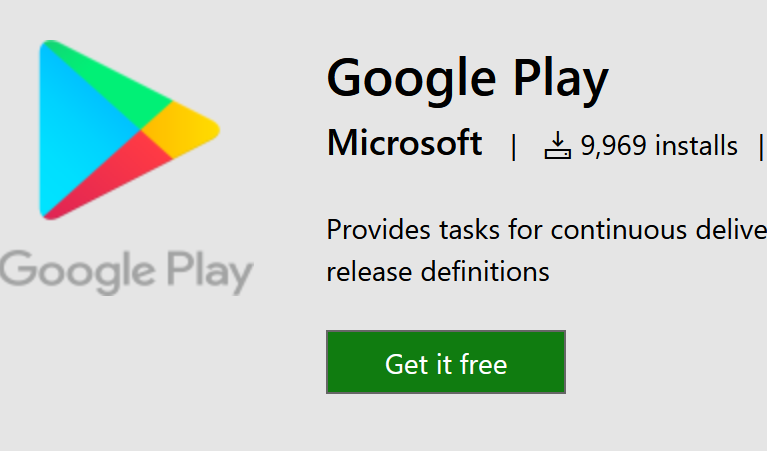


Figure 44. Azure DevOps Google Play Plug-in

* + Select **Get it free** and **Install.**

1. Create Azure Release Pipeline with Google Play push:
   * From the Azure DevOps project, select **Pipelines -> Releases -> New Release Pipeline**
   * Select **Add an artifact**.

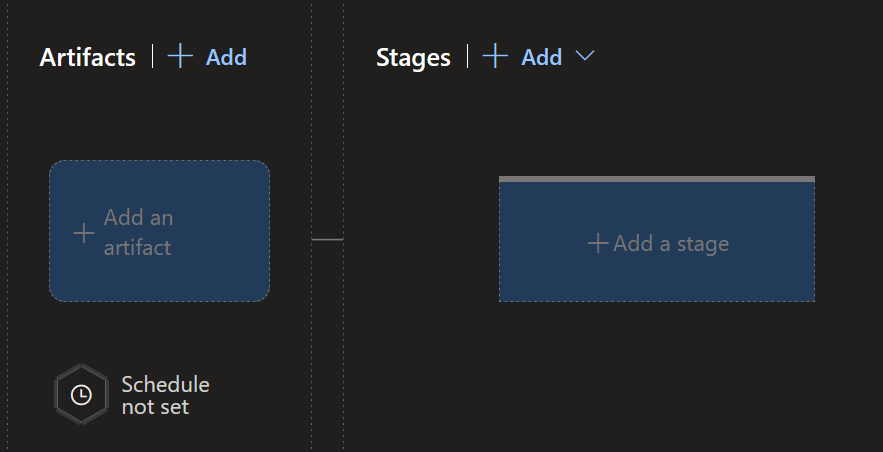


Figure 45. Azure DevOps Release Pipeline – Artifact

* + Select **Build** with Capstone for **Project** and UMGC FormScriber Dialogflow – Go as **Source**.

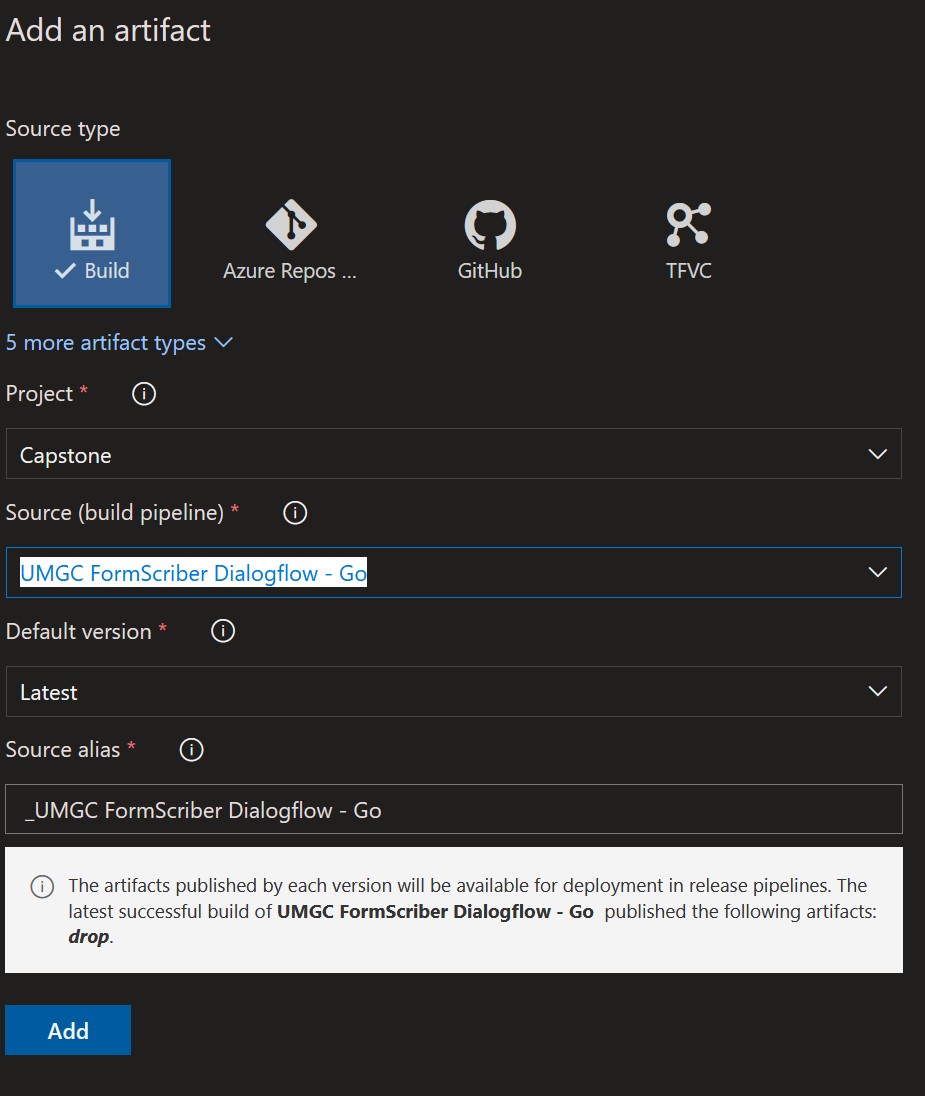


Figure 46. Azure DevOps Release Pipeline – Select

* + Select **Add Stage** and **Empty Job**

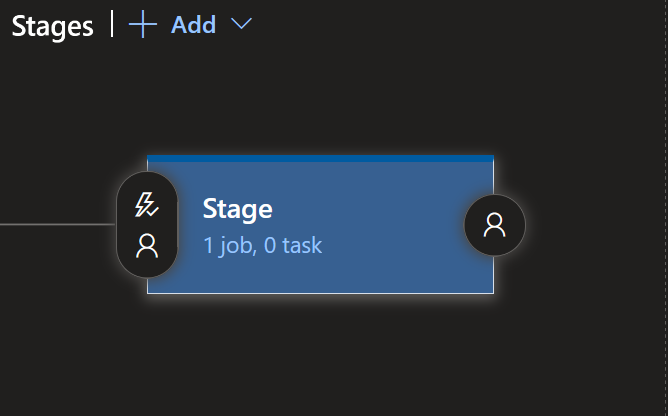


Figure 47. Azure DevOps Release Pipeline - Stage

* + Select **1 job**, search for **Google Play** and select **Add**

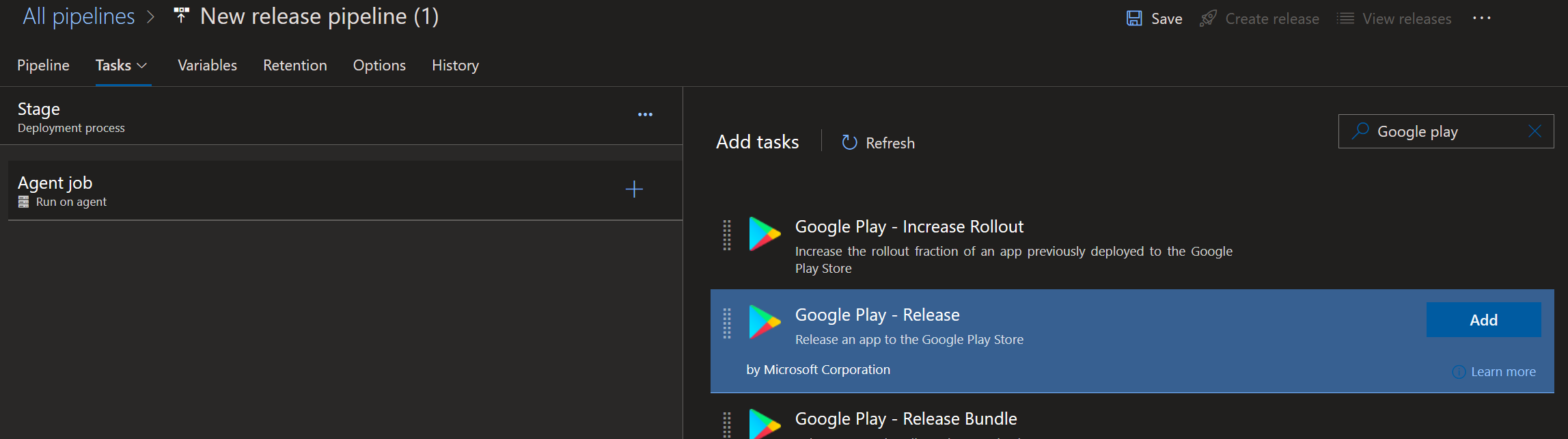
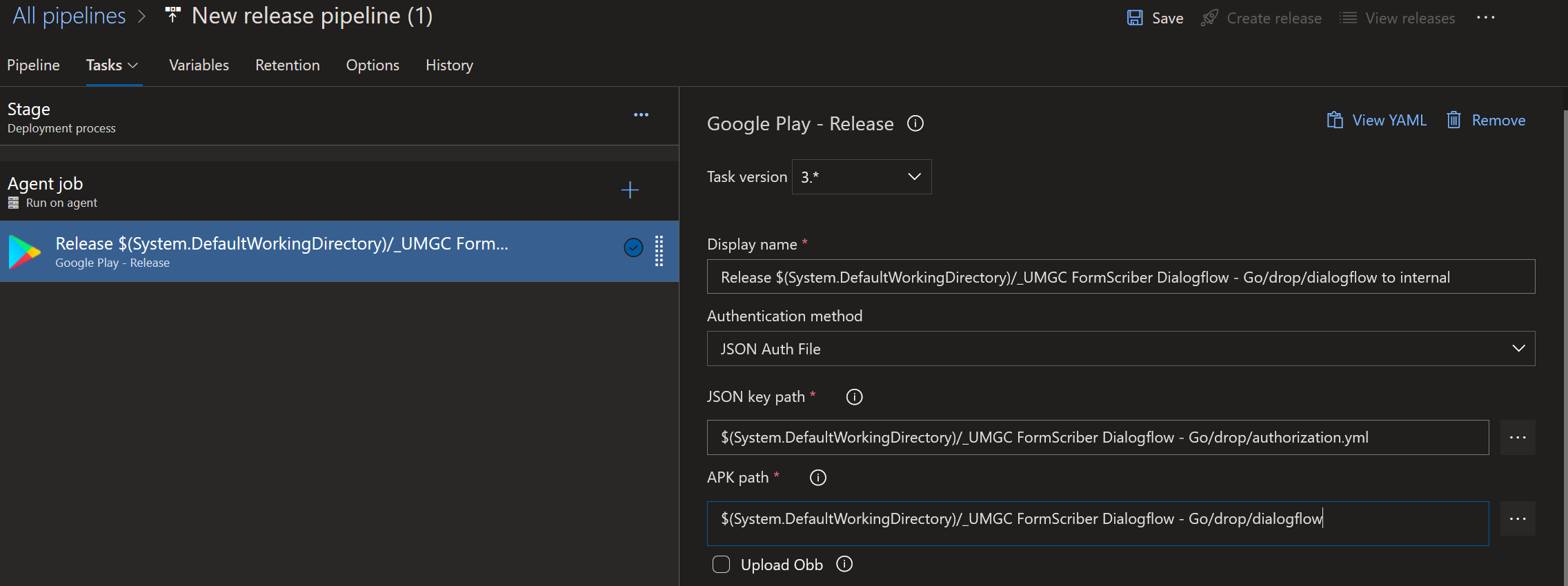


Figure 48. Azure DevOps Release Pipeline – Agent Job

* + Select appropriate path and file for the JSON authentication file and the APK.



* + Select **Save** to complete.