

Deployment Guide

Version 1.0

Presented by

Stefon Williams

Caleb Crickette

Thomas Barton

Eugene Kim

Nicholas Ballo

Abdul Kamara

Sohail Sobhani

Geoffrey Dean

Jean-nae Dedrick

Anthony Cedeno

SWEN 670

University of Maryland Global Campus

Project Plan Approvals

|  |  |  |
| --- | --- | --- |
| Name | Signature | Date |
| Approved by:  Dr. Mir Assadullah |  |  |
| Approved by:  "Stakeholder" |  |  |
| Approved by:  Project Manager, Stefon |  |  |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Name | Description of Change(s) |
| 1.0 | 2/28/2021 | Eugene Kim | Initial Release |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Purpose 4](#_Toc66020761)

[2. Intended Audience 4](#_Toc66020762)

[3. Software Overview 6](#_Toc66020763)

[4. Configurations 7](#_Toc66020764)

[4.1. REST API 7](#_Toc66020765)

[Prerequisites 7](#_Toc66020766)

[Features 7](#_Toc66020767)

[4.2. GCP Dialogflow Engine 8](#_Toc66020768)

[Prerequisites 8](#_Toc66020769)

[Features 8](#_Toc66020770)

[5. Software Installation 8](#_Toc66020771)

[5.1. REST API 8](#_Toc66020772)

[Prerequisites 8](#_Toc66020773)

[Installations Steps 8](#_Toc66020774)

[Configuration 10](#_Toc66020775)

[Configured Values 10](#_Toc66020776)

[Administration 10](#_Toc66020777)

[5.2. GCP Dialogflow Engine 10](#_Toc66020778)

[Prerequisites 10](#_Toc66020779)

[Installation Steps 10](#_Toc66020780)

[Configuration 11](#_Toc66020781)

[Configured Values 11](#_Toc66020782)

[Administration 11](#_Toc66020783)

[6. Known Bugs 11](#_Toc66020784)

[7. Testing Installation 11](#_Toc66020785)

# Purpose

This deployment guide provides the technological information needed to install and initialize the Formscriber application when standing up a fresh Form Scriber instance. This only includes the installation and deployment of the Dialogflow engine, the webhook service, and the Docs API.

# Intended Audience

Technical personnel are the primary expected audience for this document in the form of stakeholders that provide planning, setup, and installation as well as deployment of the Formscriber application. Stakeholder rolls may include system administrators, analysts, and developers.

The information contained here should allow stakeholders to coordinate the deployment of a Formscriber application to integrate with the Dialogflow service

Table 1 - Project Stakeholders

|  |  |  |
| --- | --- | --- |
| Name | E-mail address | Role |
| Professor Mir Assadullah | mir.assadullah@faculty.umgc.edu | Stakeholder |
| Roy Gordon | [uspsrgordon@aol.com](mailto:uspsrgordon@aol.com) | PM Mentor |
| Johnny Lockhart | janthony@umgc.dev | Dev Mentor |
| Stefon Williams | swilliams538@student.umgc.edu | Project Manager |
| Caleb Crickette | ccrickette@@student.umgc.edu | Lead Developer |
| Thomas Barton | [tbarton7@student.umgc.edu](mailto:tbarton7@student.umgc.edu) | Developer |
| Sohail Sobhani | [ssobhani@student.umgc.edu](mailto:ssobhani@student.umgc.edu) | Developer |
| Abdul Kamara | [akamara35@student.umgc.edu](mailto:akamara35@student.umgc.edu) | Developer |
| Eugene Kim | [ekim55@student.umgc.edu](mailto:ekim55@student.umgc.edu) | Business Analyst |
| Nicholas Ballo | [nbsockem@gmail.com](mailto:nbsockem@gmail.com) | Test Engineer |
| Jean-nae’ Dedrick | [jcdedrick@gmail.com](mailto:jcdedrick@gmail.com) | Project Manager |
| Geoff Dean | [gdean5@student.umgc.edu](mailto:gdean5@student.umgc.edu) | Developer |
| Anthony Cedeno | [anthony.cedeno@gmail.com](mailto:anthony.cedeno@gmail.com) | Test Engineer |
| Ivy Pham | [ipham@student.umgc.edu](mailto:ipham@student.umgc.edu) | DevSecOps Engineer |
| Vincent Leung | [vleung1@studeng.umgc.edu](mailto:vleung1@studeng.umgc.edu) | DevSecOps Engineer |

Table 2 - Acronyms and Abbreviations

|  |  |
| --- | --- |
| Abbreviation | Definition |
| GCP | Google Cloud Platform |
| UI | User Interface |
| VM | Virtual Machine |
| AKS | Azure Kubernetes Service |
| ACR | Azure Container Registry |

# Software Overview

Dialogflow is an application that interfaces between the end-user (professional) and records their voice. The service does additional processing to help the end-user fill out their report. The use of a mobile UI for recording and Google’s Dialogflow engine that interprets the audio shall form the final product.

At a high level, the mobile team is focusing on building out a UI that can identify the user and allow them to record their voice using an API that our team will provide. The Dialogflow team will focus on building out the backend services by receiving the recorded audio in real time, extracting the relevant fields and data, and populating a custom report on Google Docs/Sheets. DevSecOps will provide the necessary infrastructure to deploy our software using Kubernetes and create the appropriate logging and debugging tools to audit system applications.

The Dialogflow service architecture will consist of three different components:

1. Dialogflow Engine
   1. GCP’s Dialogflow engine allows developers to program intents based on speech or text. These intents are used to extract data from the user’s speech. Our service will enable the user to create intents based on the data they want to capture, which will allow the application to collect that data.
2. Web Application Layer
   1. This is the Google Assistant UI that the user will interact with. There will be guided prompts to assist the user in using the application as well as provide feedback for any captured data.
3. Webhook Service
   1. The webhook service will integrate with the Dialogflow engine to create the dynamic intents based on the user’s report fields and process the extracted data coming from the engine during user interaction. This service will help to do the final ETL processes for the application.
4. Google Docs/Sheets Integration
   1. Reports will be stored in Google Drive using Google Docs and Google Sheets. These are both Google’s document and spreadsheet applications. Our service will need to integrate with those to allow for read and write permissions. This will enable the final report to be generated for the user.

# Configurations

## REST API

### Prerequisites

The REST API is a Go server that runs a frontend Vue application for users to access help documents and a backend Go service for Dialogflow webhook requests. The REST API will act as the webhook service and the Google Docs authentication agent. It will be used to read custom report fields, generate dynamic intents, process any incoming data coming from the Dialogflow engine, and fill out the final report. There will also be a web application layer that shares a privacy policy, terms of service, and information about the application.

The system must have the following software and tools installed before deploying the REST API

* make – version 4.2.1 or higher
* git – version 2.25.1 or higher
* docker – version 19.03.8 or higher
* Azure CLI – version 2.0.81 or higher
* Go – version 1.16 or higher
* kubectl – version 1.20.4 or higher
* helm – version 3.5.2 or higher

An Azure account is needed in order to be able to deploy on Azure. An Azure account can be created at <https://azure.microsoft.com/en-us/free/>

A Google Drive account is necessary to run the application and read/write permissions must be given to the application through an access token. The steps to retrieve the token is outlined in the Software Installation section. A Google account can be created for free at <https://accounts.google.com/signup>.

### Features

The REST API makes use of the following features:

* RESTful interface able to receive GET and POST requests
  + POST requests are used by Dialogflow webhook requests
  + GET requests are used for the web application layer
* Sending POST requests to Google APIs

## GCP Dialogflow Engine

The Google Dialogflow engine was created to be the primary driving component of the application that allows for the interpretation of the various intents as well as the voice of the professional that uses the application.

### Prerequisites

* GCP account is needed to access the Dialogflow service
* The Dialogflow console can be accessed at <https://dialogflow.cloud.google.com/>

### Features

GCP Dialogflow Engine makes use of the following features:

* Recognize intents and capture variables from user’s input via voice or text.
* Generate intents that look for key words in user’s input
* Generate responses based on intents.

# Software Installation

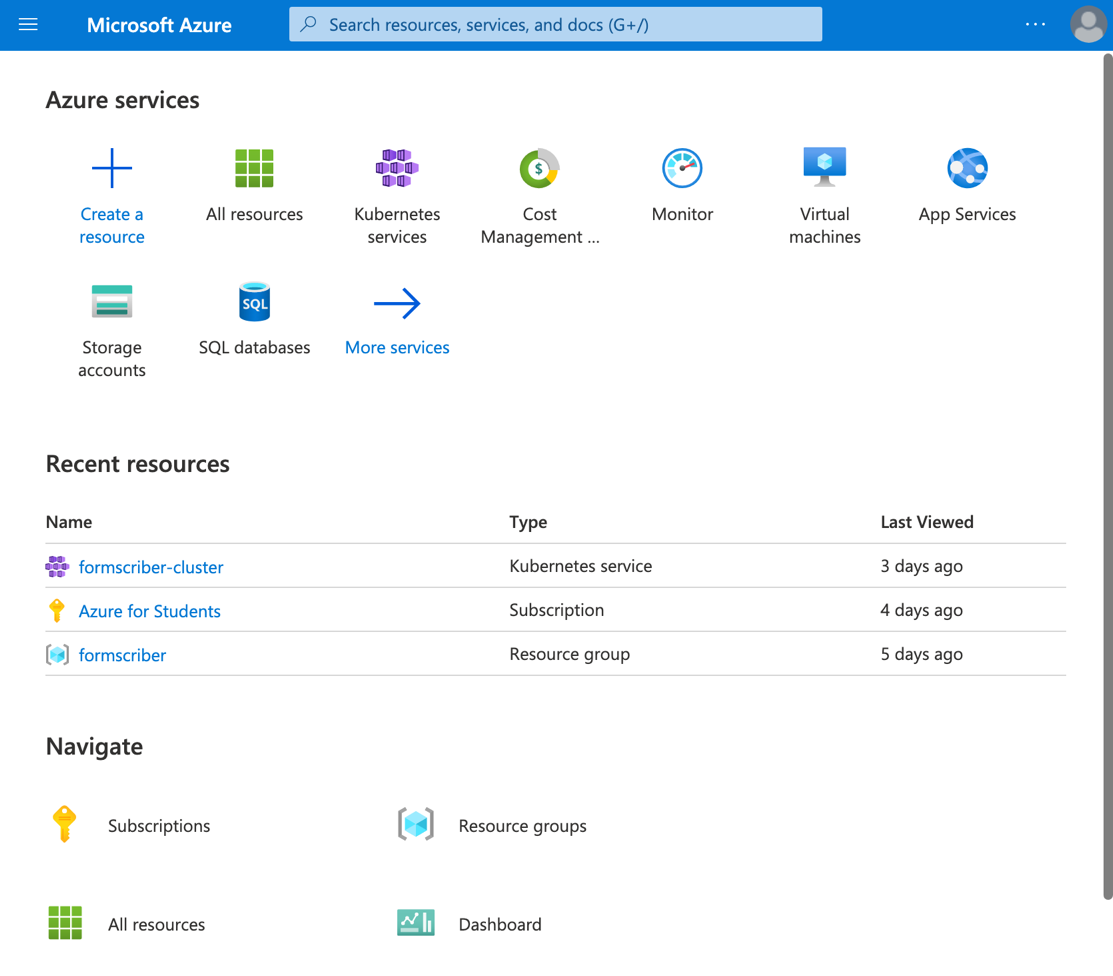
## REST API

### Prerequisites

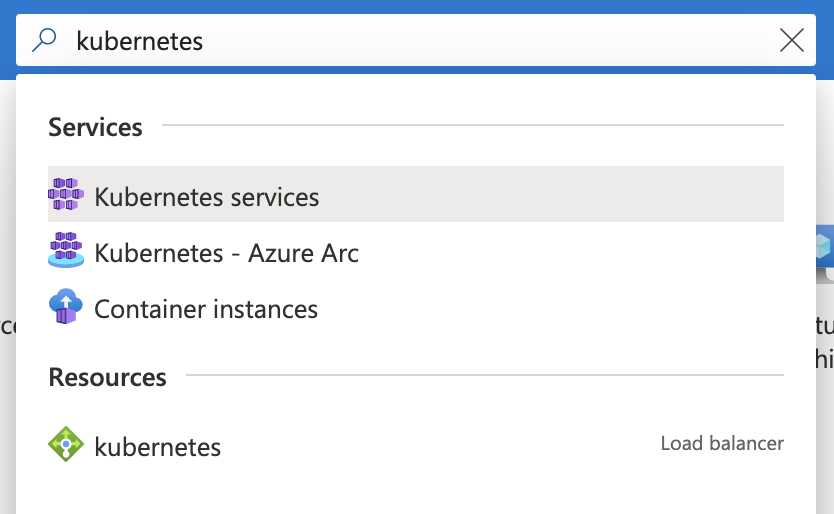
* Azure account needed to access Azure resources.
  + An Azure account can be created at <https://azure.microsoft.com/>
* Google account needed to access Drive sharing permissions for reports.
  + A Google account can be created at <https://www.google.com/intl/en/drive/>
* A Github account is needed to grab the code repository.
  + A Github account can be created at <https://github.com/join>

### Installations Steps

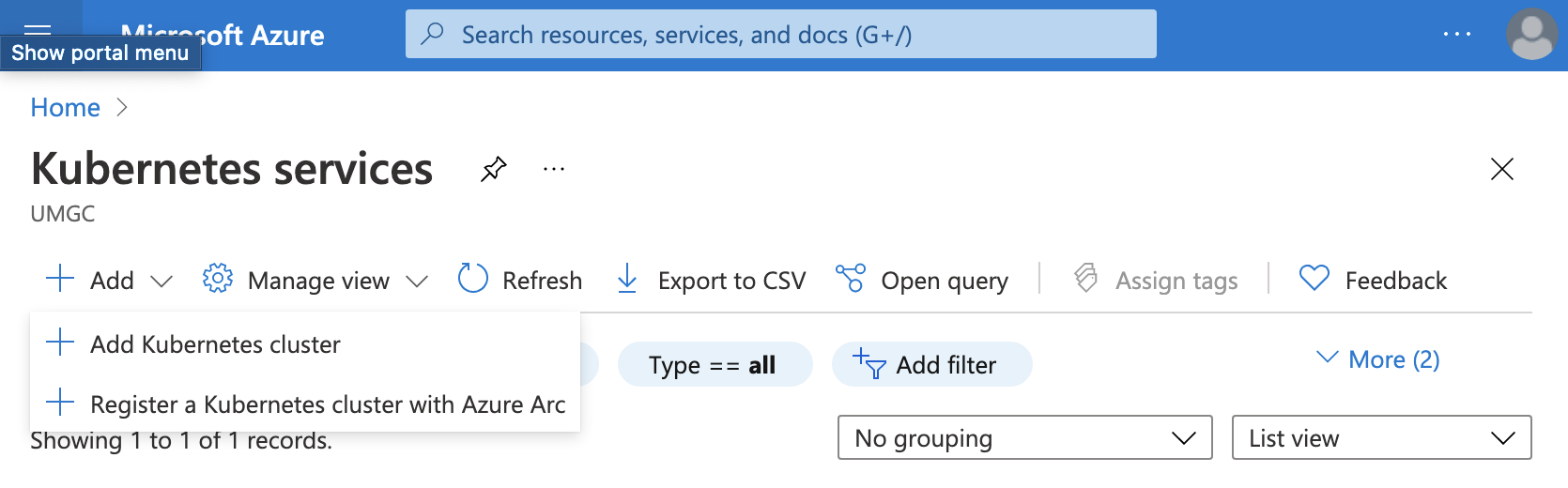
1. Install the packages mention in section 5.2.1.
   1. make comes preinstalled in most linux distributions. Information to install on your OS can be found at <https://www.gnu.org/software/software.html#getgnu>
   2. git can be installed from <https://git-scm.com/downloads>
   3. docker can be installed for your OS at <https://docs.docker.com/get-docker/>
   4. Azure CLI installation guides can be found on <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli>
   5. Go can be installed at <https://golang.org/doc/install>
   6. kubectl can be installed by running the following Azure CLI command:  
      **az aks install-cli**
   7. Helm can be installed by following the documentation at <https://helm.sh/docs/intro/install/>
2. Create a single node Kubernetes cluster on AKS.
   1. Navigate to portal.azure.com and authenticate with your Azure account details



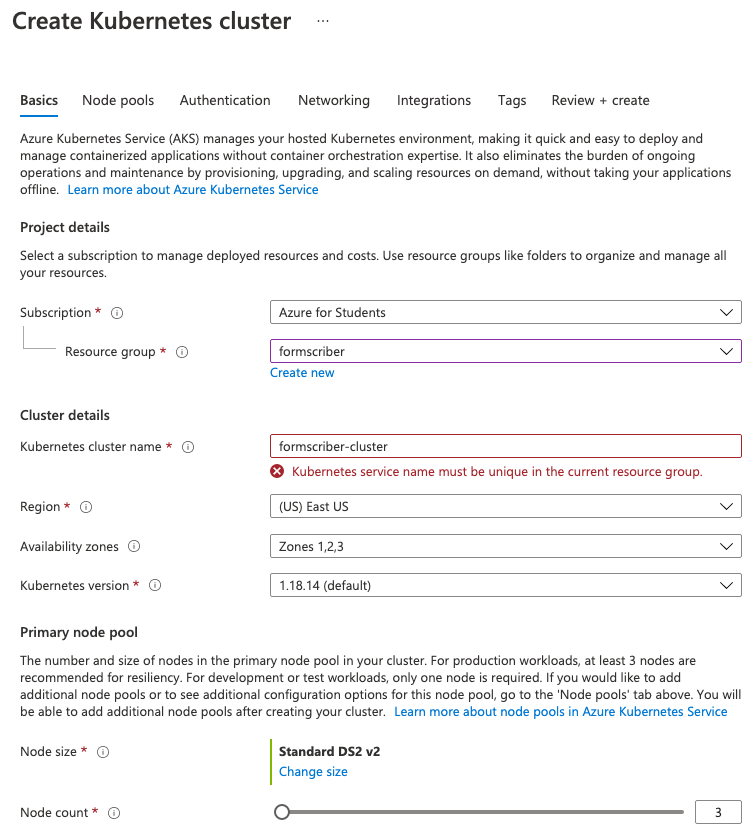
* 1. Search for Kubernetes Service in the search bar at the top and select Kubernetes services



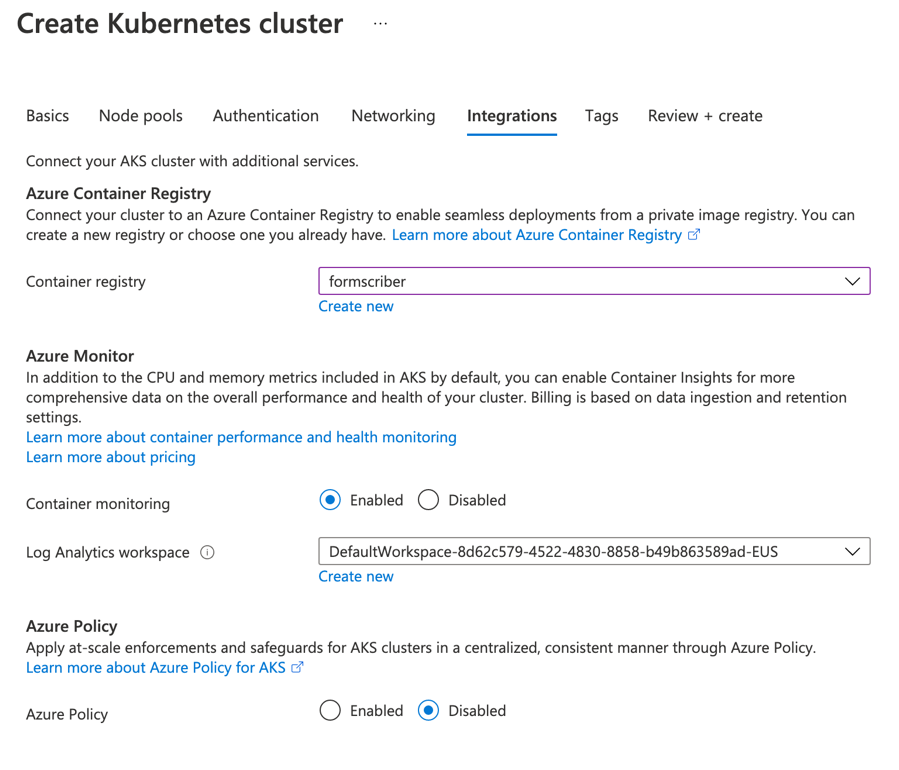
* 1. Select the ‘+ Add’ dropdown button and select ‘+ Add Kubernetes Cluster’.



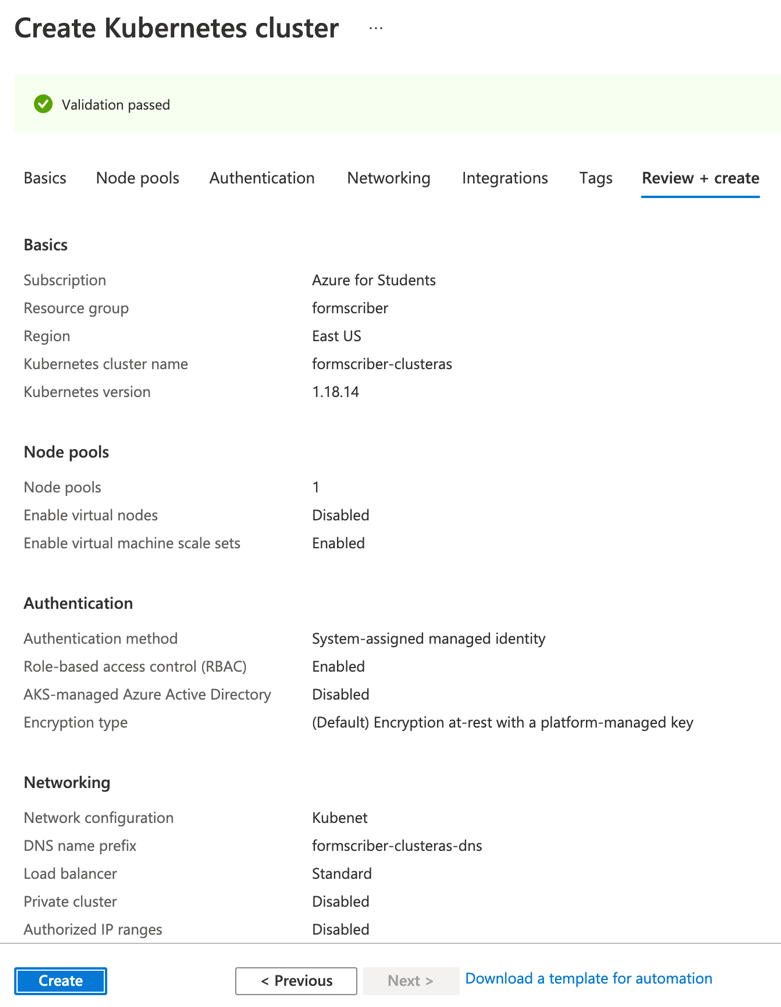
* 1. Create a new resource group called formscriber and set the Kubernetes cluster name to be formscriber-cluster.



* + 1. The node size and count can be updated and is dependent on expected usage and load of the application but the recommended size for cost efficiency is B2s with a node count of 1.
    2. More information on VMs can be found at <https://docs.microsoft.com/en-us/azure/virtual-machines/sizes>
  1. Select Integrations and create a new Container registry called Formscriber



* 1. Select Review + Create, review configuration details, and select the ‘Create’ button to finish setting up the AKS cluster.



1. Clone the code repository at <https://github.com/umgc/umgc.formbot.dialogflow> by running the following command in a terminal window:  
   **git clone** [**https://github.com/umgc/umgc.formbot.dialogflow**](https://github.com/umgc/umgc.formbot.dialogflow)
   1. If you run into permission errors, ensure you have access to the repository by contacting the DevSecOps team.
2. Connect to the AKS cluster by running the following command in the root directory of the code repository:  
   **make aks-login**
3. Configure HTTPS in AKS cluster
   1. Create a static public IP by running the following command:  
      **az network public-ip create --resource-group MC\_formscriber\_formscriber-cluster\_eastus --name formscriberPublicIp --sku Standard --allocation-method static --query publicIp.ipAddress -o tsv**
      1. Keep track of the output which will be the static Ip assigned to our AKS resource group.
   2. Create a nginx ingress controller by running the following commands:

**# Create a namespace for your ingress resources**

**kubectl create namespace cert-manager**

**# Add the ingress-nginx repository**

**helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx**

**# Use Helm to deploy a NGINX ingress controller**

**helm install nginx-ingress ingress-nginx/ingress-nginx \**

**--namespace cert-manager \**

**--set controller.replicaCount=2 \**

**--set controller.nodeSelector."beta\.kubernetes\.io/os"=linux \**

**--set defaultBackend.nodeSelector."beta\.kubernetes\.io/os"=linux \**

**--set controller.admissionWebhooks.patch.nodeSelector."beta\.kubernetes\.io/os"=linux \**

**--set controller.service.loadBalancerIP="STATIC\_IP" \**

**--set controller.service.annotations."service\.beta\.kubernetes\.io/azure-dns-label-name"=”formscriber-umgc"**

* + 1. Make sure to replace STATIC\_IP with the IP created from the last step.
  1. Install cert-manager by running the following commands

**# Label the cert-manager namespace to disable resource validation**

**kubectl label namespace cert-manager cert-manager.io/disable-validation=true**

**# Add the Jetstack Helm repository**

**helm repo add jetstack https://charts.jetstack.io**

**# Update your local Helm chart repository cache**

**helm repo update**

**# Install the cert-manager Helm chart**

**helm install \**

**cert-manager \**

**--namespace cert-manager \**

**--version v0.16.1 \**

**--set installCRDs=true \**

**--set nodeSelector."beta\.kubernetes\.io/os"=linux \**

**jetstack/cert-manager**

* 1. Install the CA Cluster issuer by running the following command in the root directory of the code repository:  
     **kubectl apply -f deploy/ssl/cluster-issuer-prod.yaml**

1. Run the deployment script by running the following command in the root directory of the code repository:   
   **make deploy**
   1. On the first time you run this script, it will ask you authenticate your azure account via the web browser.
2. Access webapp at <https://formscriber-umgc.eastus.cloudapp.azure.com/>.

### Configuration

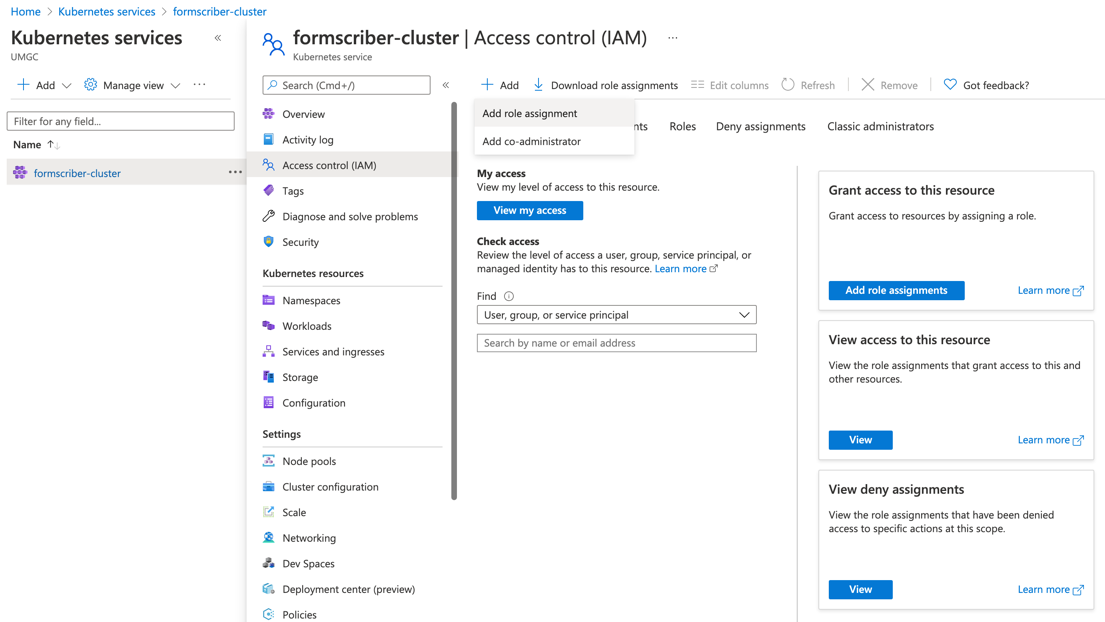
All configured values can be found at the top of the **Makefile** in the root directory of the code repository.

### Configured Values

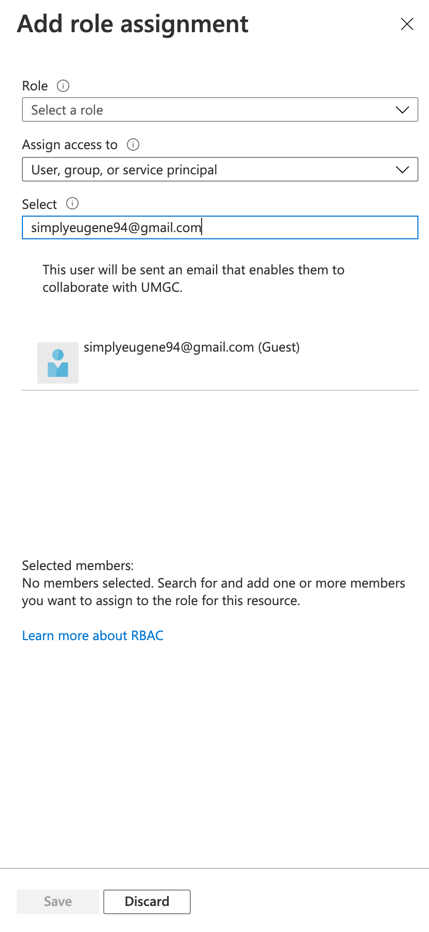
|  |  |  |
| --- | --- | --- |
| Value | Definition | Default Value |
| SERVICE | Name of the application docker image. | formscriberapi |
| VERSION | Version and tag of the application docker image. | 0.2 |
| RESOURCE\_GROUP | Azure resource group that defines a set of cloud resources. | formscriber |
| REGISTRY | Name of the ACR used for the service | formscriber |
| CLUSTER | Name of the AKS cluster used for the service. | formscriber-cluster |
| NAMESPACE | Name of the Kubernetes namespace that services are deployed in. | cert-manager |

### Administration

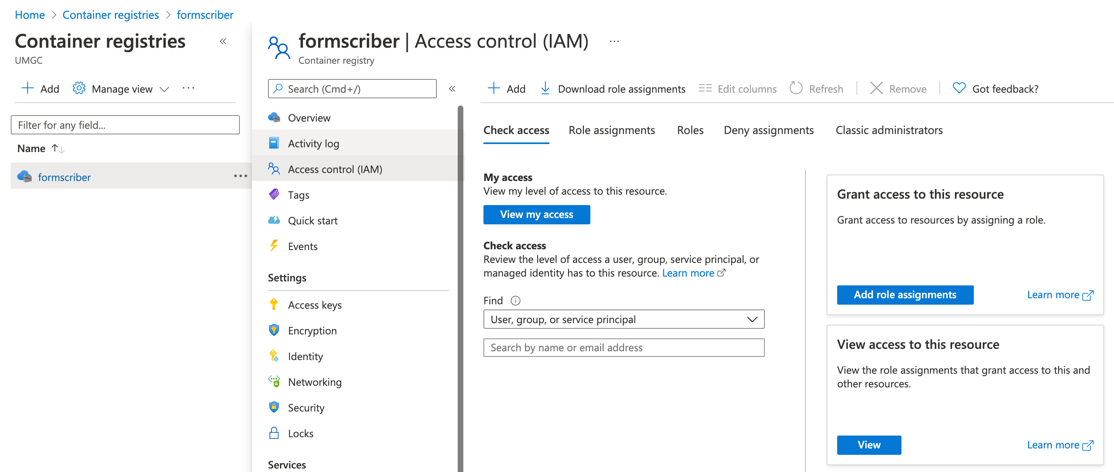
1. Deleting the cluster and all related resources when not in use to save costs
   1. To delete the cluster, run the following command:  
      **az group delete --name formscriber --yes --no-wait**
2. Adding permissions to allow other users to deploy and view services on AKS
   1. Navigate to the Azure portal at <https://portal.azure.com>
   2. Navigate to the **Kubernetes services** resource and select **formscriber-cluster**
   3. Select the **Access control (IAM)** menu and select the **Add role assignments** button.



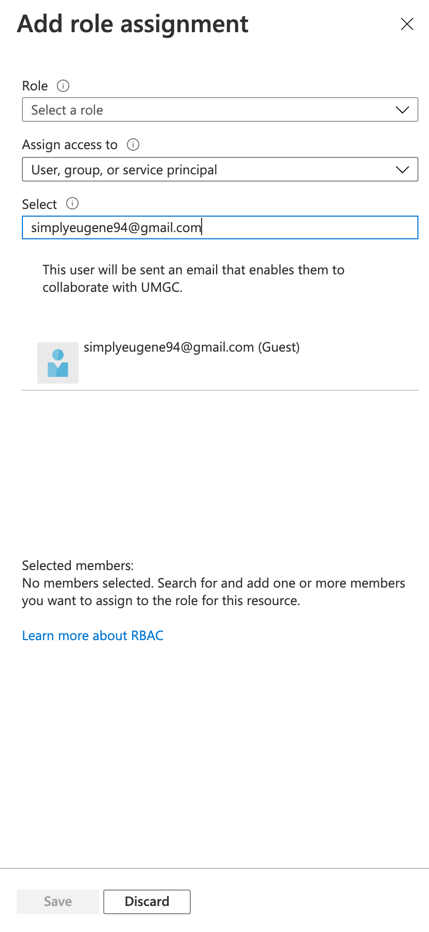
* 1. Select an appropriate role for the user and type in their email in the **Select** box



1. Adding permissions to allow other users to push and pull images from ACR
   1. Navigate to the Azure portal at <https://portal.azure.com>
   2. Navigate to the **Container registries** resource and select **formscriber**
   3. Select the **Access control (IAM)** menu and select the **Add role assignments** button.



* 1. Select an appropriate role for the user and type in their email in the **Select** box



1. Running the service on a local server or machine
   1. Run the following command in the root directory of the code repository:  
      **make run-docker**
   2. Navigate to <http://localhost:8080> to view the application.

## GCP Dialogflow Engine

### Prerequisites

* GCP account needed to access Dialogflow console.
  + GCP account can be created at <https://cloud.google.com>
* Verify REST API is installed and accessible at <https://www.formscriber.com>

### Installation Steps

1. Create an agent to be used for the application
2. Clone the code repository at <https://github.com/umgc/umgc.formbot.dialogflow>
3. Navigate to <https://dialogflow.cloud.google.com>.
4. Select your agent from the dropdown menu.
5. Click the settings button next to the agent name.
6. Select the **Export and Import** tab.
7. Select **Restore From Zip** and follow instructions to restore the zip file located in the code repository at dialogflow/form-bot-agent.zip.

### Administration

1. Export Dialogflow agent to save agent configurations.
   1. Navigate to <https://dialogflow.cloud.google.com>.
   2. Select your agent from the dropdown menu.
   3. Click the settings button next to the agent name.
   4. Select the **Export and Import** tab.
   5. Select **Export As Zip** and follow instructions to restore the zip file located in the code repository at dialogflow/form-bot-agent.zip.
2. Adding permissions for users to configure the Dialogflow service.

# Known Bugs

|  |  |  |
| --- | --- | --- |
| Bug | Description | Proposed Solutions |
| Hardcoded DNS | There is no way to dynamically configure the DNS name. It is fixed on formscriber.com | 1. Enable Domain Forwarding from the DNS provider 2. Provide a configuration for users to input their own DNS entry when deploying. |
| No Automated Deployment of Infrastructure | There is no code in place to automatically build the infrastructure programmatically. | 1. Develop automated deployment pipeline alongside DevSecOps |

# Testing Installation

The following tests can be performed to ensure Dialogflow components are working properly  
  
1. Verify if the web application loads and HTTPS configuration in AKS cluster is implemented.

1. From any browser, load the URL: <https://formscriber-umgc.eastus.cloudapp.azure.com/>
2. Ensure that the FormScriber homepage loads and there are no SSL-related errors.

2. Verify if the FormScriber REST API calls work.

1. Submit a voice or text message that will trigger a call to one of the team’s customized REST APIs.  
   Performing the test case below, and getting the expected result as indicated by *Post Condition*, shall make an assumption that all custom REST APIs, or test cases, will be accessible.  
     
   TEST CASE: Dialogflow will correctly identify a specified Google Document

|  |  |
| --- | --- |
| **Description:** | Dialogflow identifies a user-defined form created as a Google Document |
| **Requirements:** | A user-defined form created as a Google document URL, created according to instructions on the home page of formscriber.com. |
| **Actor:** | User shall paste type the URL in Google Assistant, or chat window. |
| **Trigger** | The user will press Enter after pasting the URL of the Google Document. |
| **Precondition:** | User has google assistant open and has a proper device such a smartphone or computer. Additionally, they need to be interfaced with Form Scriber by invoking the “Talk to Formscriber.com” invocation. |
| **System Response** | The system will respond that it has recognized the specified Google Document form and prompts for the next field to fill. |
| **Post Condition** | The user receives the system response indicated above and the system waits for the this user’s response. |
| **Alternate Path** | If the user specifies a non-existing Google Document, the system will notify the user regarding the issue. |