

NashTech Training

**Hands on: Practical DevOps for DEV**

SD2350 – Hồ Phước Trúc

21 August 2023

[1 **Infrastructure as Code (IaC)**: Provision Azure resources with Terraform 2](#_Toc143464393)

[1.1 Authenticate using the Azure CLI 2](#_Toc143464394)

[1.2 Prepare IaC project folders and variables 3](#_Toc143464395)

[1.3 Script to create Virtual Network (VNET) and subnets using in the AKS cluster 4](#_Toc143464396)

[1.4 Script to create Azure Container Registry (ACR) 5](#_Toc143464397)

[1.5 Script to create Azure Kubernetes Service (AKS) 5](#_Toc143464398)

[1.6 Run Terraform scripts 6](#_Toc143464399)

[1.7 Verify resources created on Azure 7](#_Toc143464400)

[2 **CI:** Setting up an azure pipeline, build and push Docker image to ACR 9](#_Toc143464401)

[2.1 Setting up Azure Docker Registry service connection 10](#_Toc143464402)

[2.2 Create new a CI pipeline and connect to MSA GitHub repository 11](#_Toc143464403)

[2.3 Run CI build pipeline and verify results 13](#_Toc143464404)

[3 **CD**: Setting up an azure pipeline and deploying applications on AKS 14](#_Toc143464405)

[3.1 Create a Kubernetes namespace using for deployment 15](#_Toc143464406)

[3.2 Setting up AKS service connection 16](#_Toc143464407)

[3.3 Create new a CD pipeline and connect to Infrastructure GitHub repository 16](#_Toc143464408)

[3.4 Run CD build pipeline and verify results 18](#_Toc143464409)

[4 **GitOps**: Replace azure pipeline CD by Argo CD 19](#_Toc143464410)

[4.1 Install Argo CD on AKS 20](#_Toc143464411)

[4.2 Add new applications to Argo CD 22](#_Toc143464412)

[4.3 Test deployment with Argo CD 24](#_Toc143464413)

[5 **Monitoring**: Setup Prometheus and Grafana to monitor AKS resources 26](#_Toc143464414)

[5.1 Create a namespace for monitoring 26](#_Toc143464415)

[5.2 Install Prometheus and Grafana tools 27](#_Toc143464416)

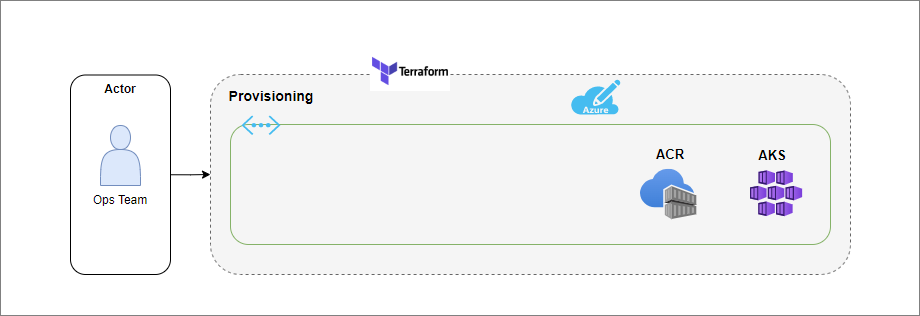
[5.3 Verify dashboard monitoring 28](#_Toc143464417)

# Infrastructure as Code (IaC): Provision Azure resources with Terraform

Using Terraform to provision resources include VNET, ACR and AKS on Azure cloud.

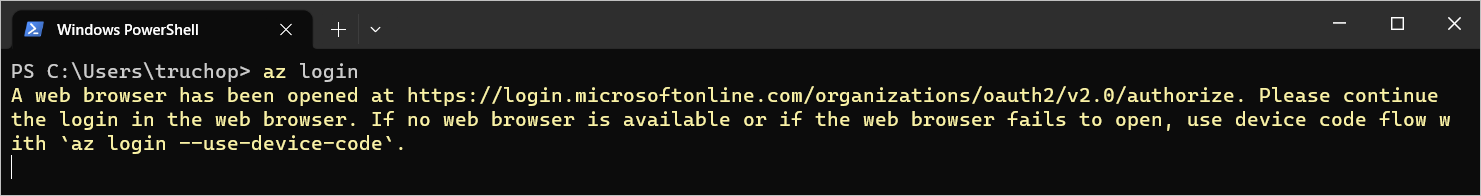
GitHub repository: [https://github.com/hptruc/sd2350\_azure\_infrastructure](https://github.com/hptruc/sd2350_azure_infrastructure/tree/main/iac)

Architecture overview:

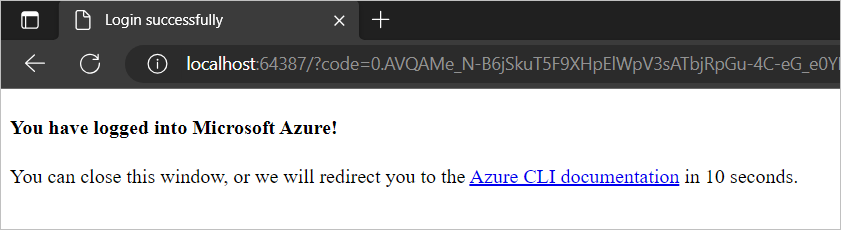


## Authenticate using the Azure CLI

* In your terminal, use the Azure CLI tool to setup your account permissions locally.

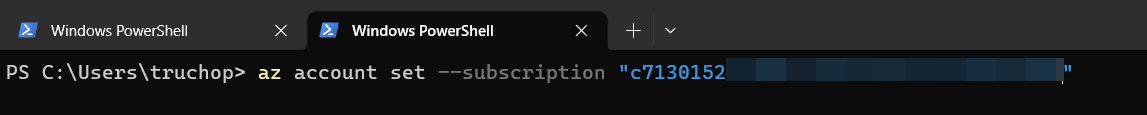


* Your browser will open and prompt you to enter your Azure login credentials.
* After successful authentication, your terminal will display your subscription information.



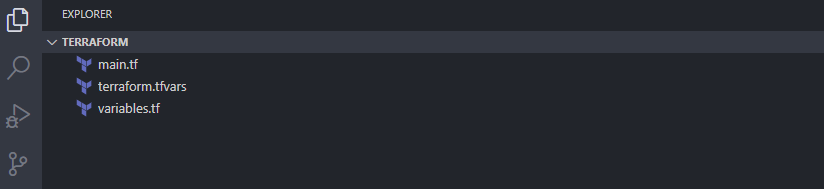


* Once you have chosen the account subscription ID, set the account with the Azure CLI.

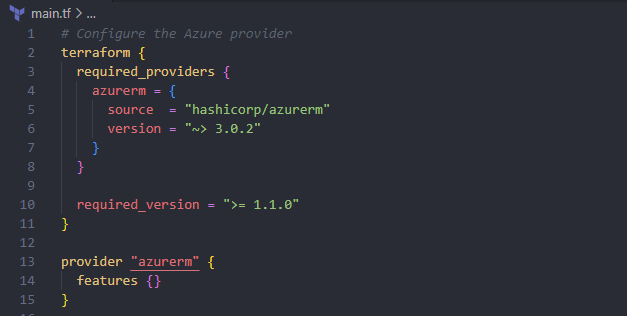


## Prepare IaC project folders and variables

* Create a folder to store IaC source code, e.g. Terraform
* In this folder, create 3 files as below



* In **main.tf** file, add terraform script to configure the Azure provider (Azure resource manager – Azure RM)



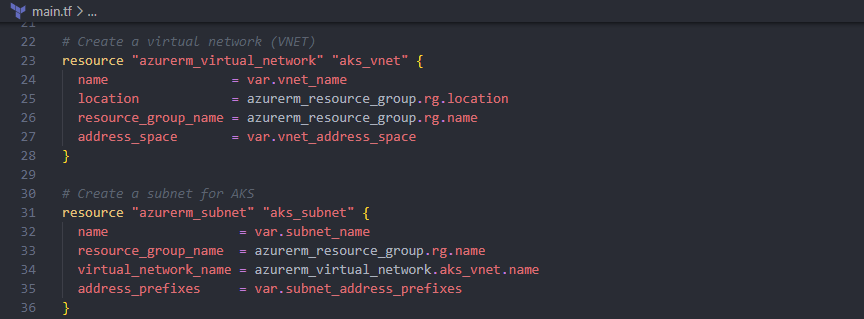
* In **variables.tfvars** file, define terraform variables as below



* Notes: variable name and type defined in **terraform.tf** file

## Script to create Virtual Network (VNET) and subnets using in the AKS cluster

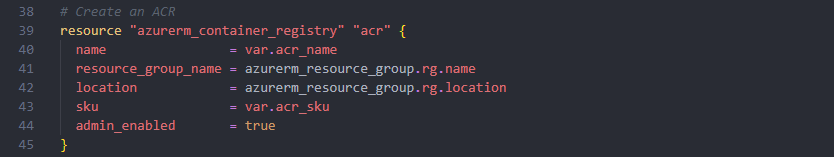
* Open the **main.tf** file and add script to create a virtual network (VNET) and a subnet using in AKS cluster.



* Notes: reference to **variables.tfvars** file to get value of variables

## Script to create Azure Container Registry (ACR)

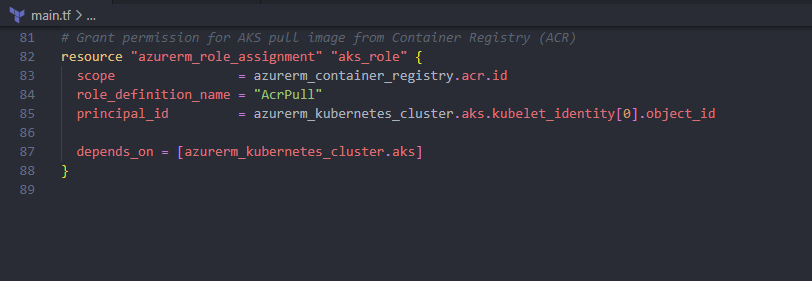
* At the end of **main.tf** file, add script below to create ACR resource



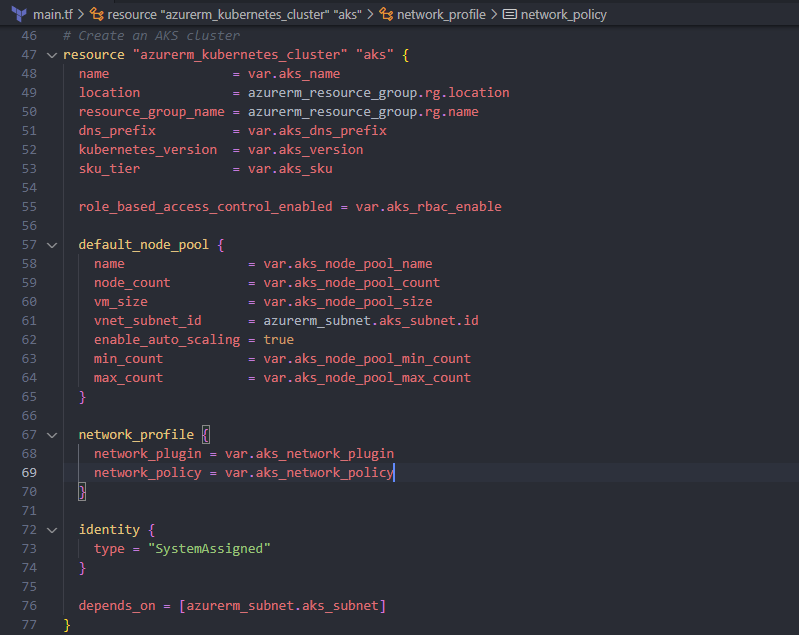
* Notes: reference to **variables.tfvars** file to get value of variables

## Script to create Azure Kubernetes Service (AKS)

* At the end of **main.tf** file, add script to create AKS resource and grant permission to AKS can pull images form container registry (ACR).
* Grant permission for AKS



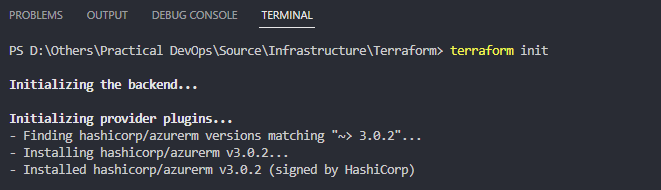
* Terraform script create the AKS resource



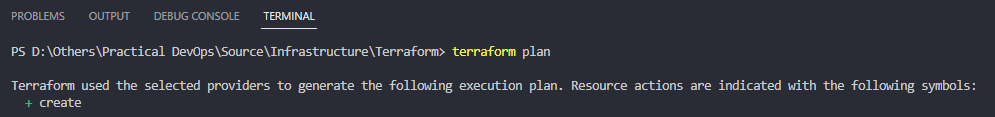
* Notes: reference to **variables.tfvars** file to get value of variables

## Run Terraform scripts

* In the root folder, open a Terminal and type command > terraform init

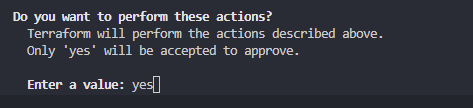


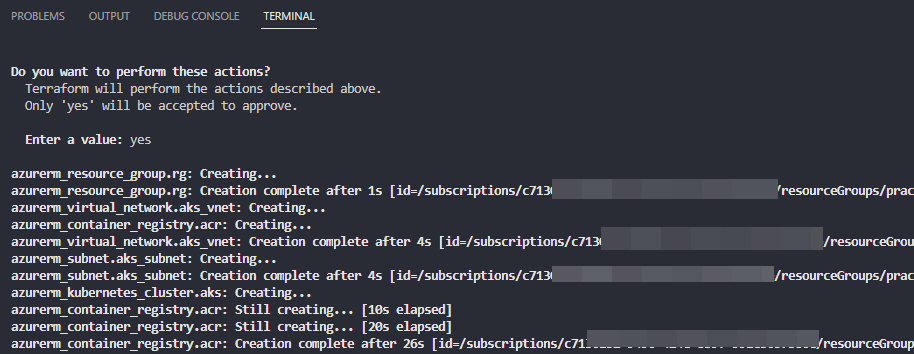
* Then, type command > terraform plan





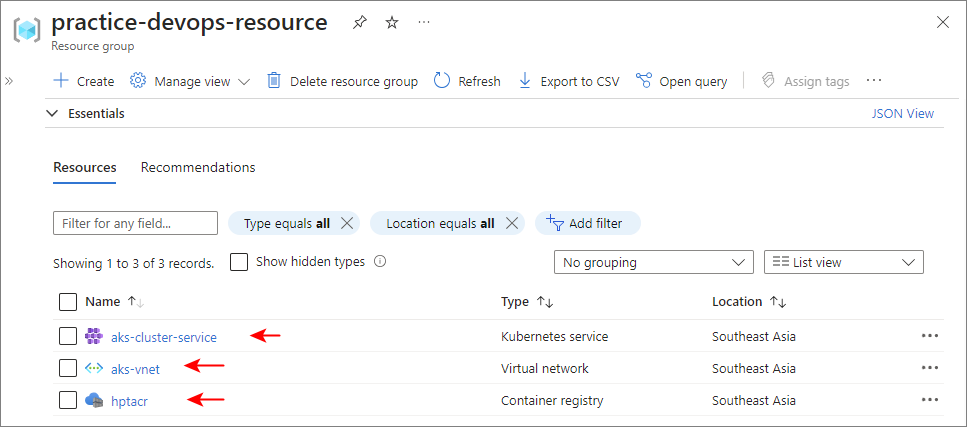
* Type commend > terraform apply



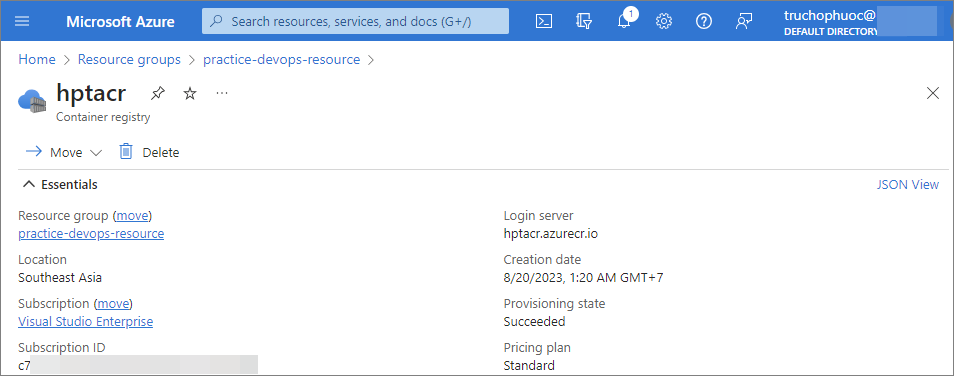


## Verify resources created on Azure

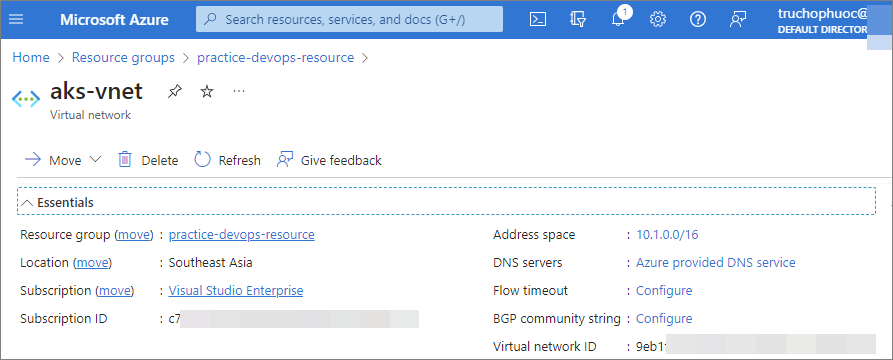
* After resources create successfully, login Azure portal to verify resources created
* Resource group

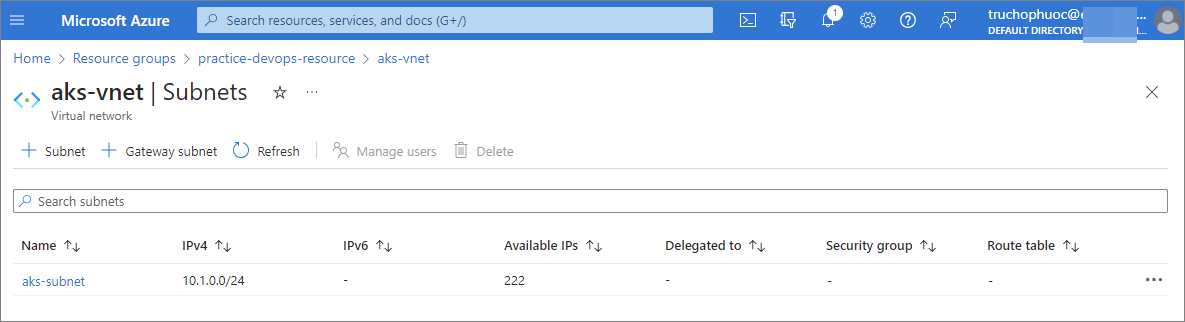


* Azure container registry (ACR)

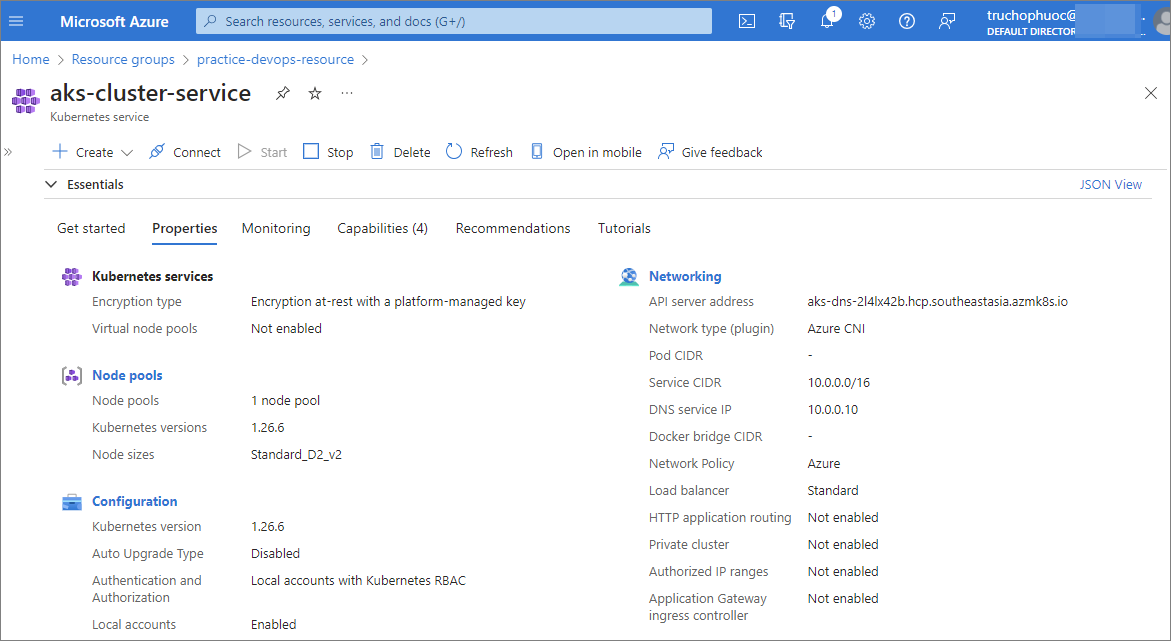


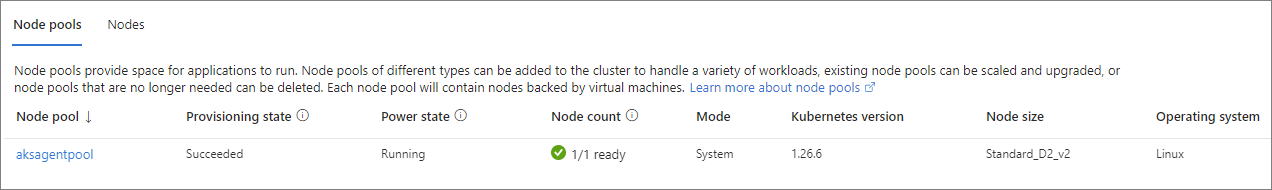
* Virtual network and subnets





* Azure Kubernetes service (AKS)





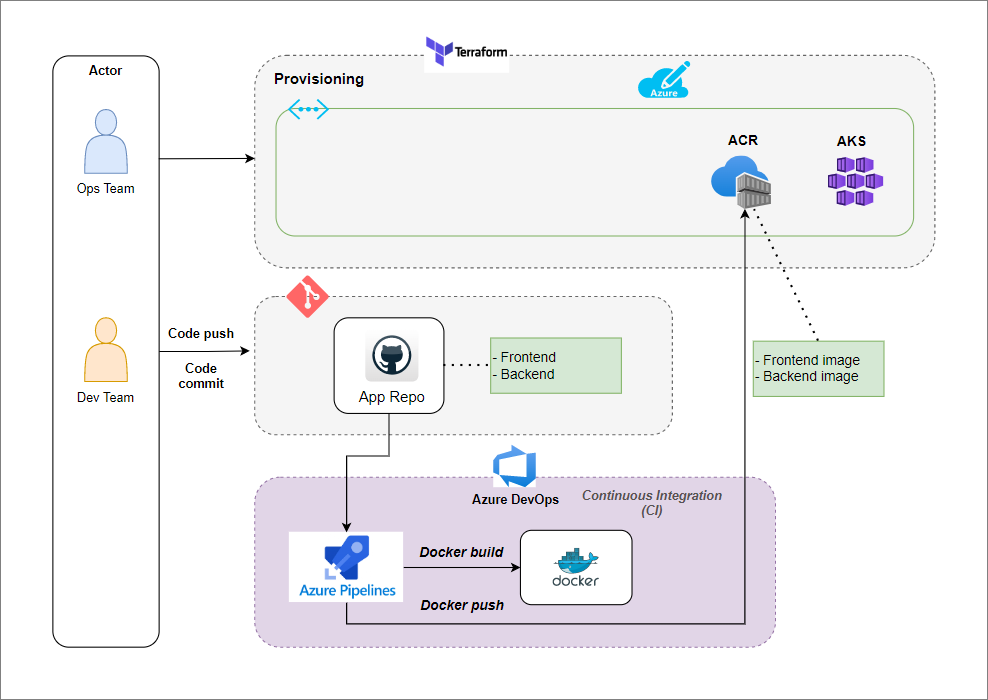
# CI: Setting up an azure pipeline, build and push Docker image to ACR

GitHub repository: [https://github.com/hptruc/sd2350\_msa](https://github.com/hptruc/sd2350_msa/blob/main/build%20pipelines/azure-ci-pipelines.yml)

Precondition:

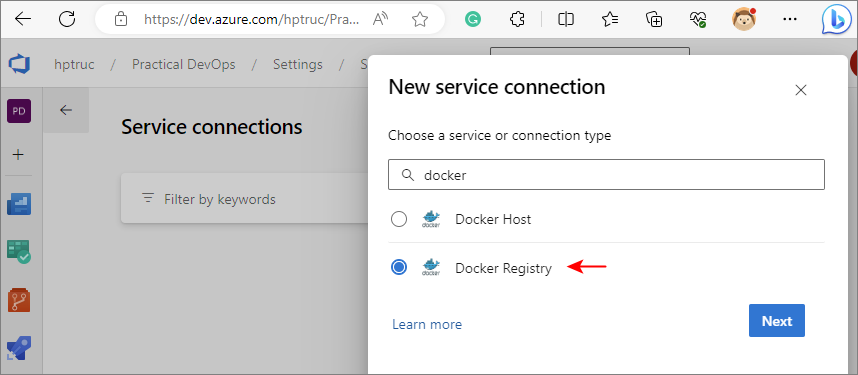
* Create an Azure DevOps instance and request to Microsoft ([fill form](https://aka.ms/azpipelines-parallelism-request)) to get a Microsoft-hosted agents for build pipelines.

Architecture overview:

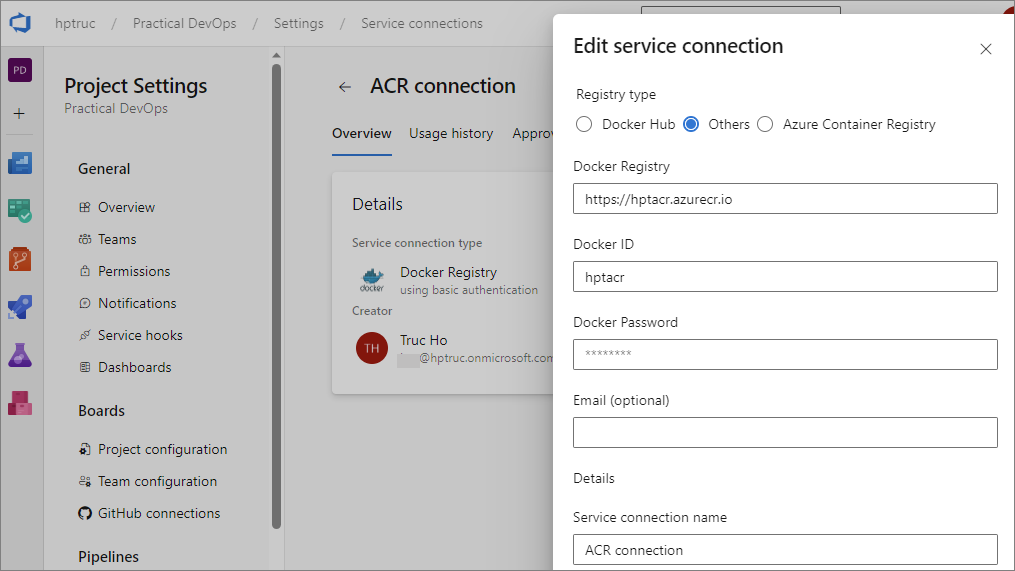


## Setting up Azure Docker Registry service connection

* Login to Azure DevOps instance and create new a project, e.g. Practical DevOps
* Go to Project settings page > Service connections > New service connection
* Select Docker Registry and click on Next button



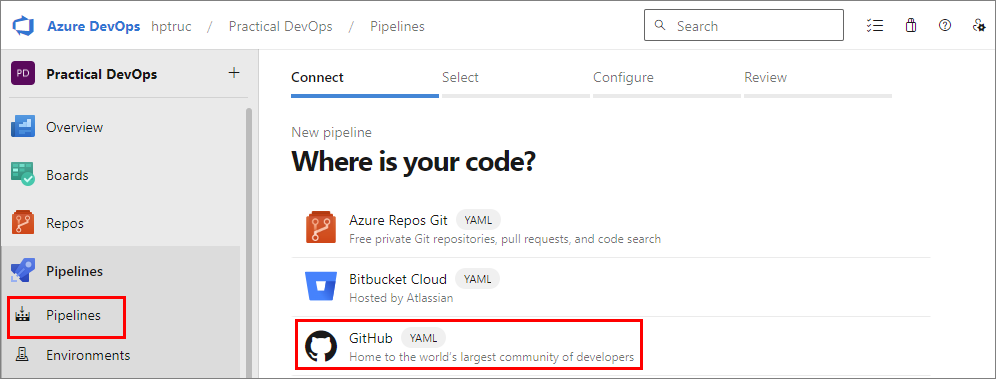
* Select **Others** option and fill required information



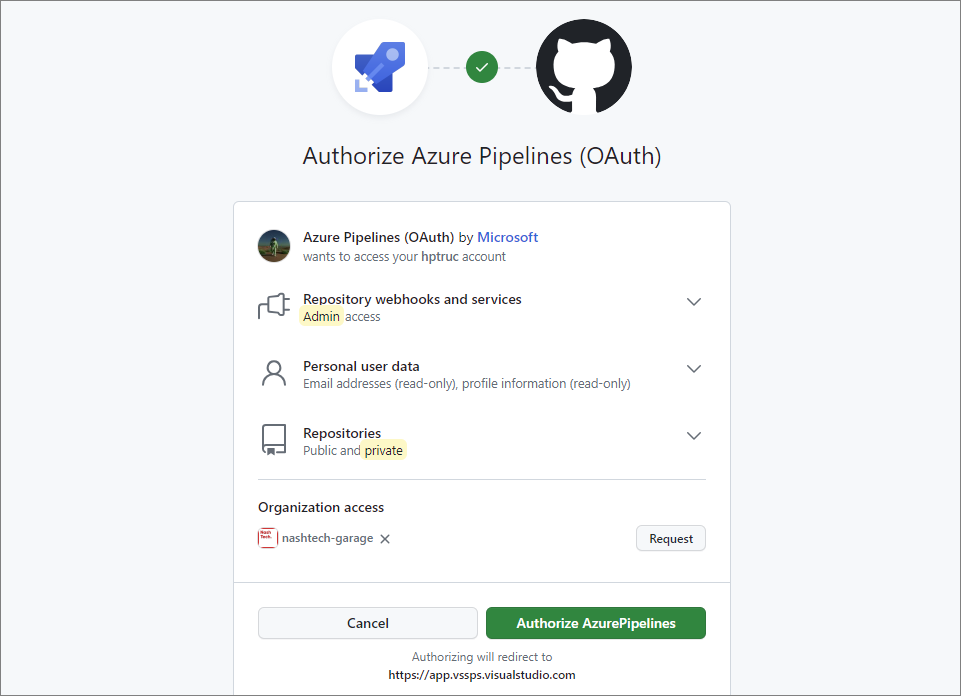
* The “**Azure container registry login server**” field: Link of ACR instance.

## Create new a CI pipeline and connect to MSA GitHub repository

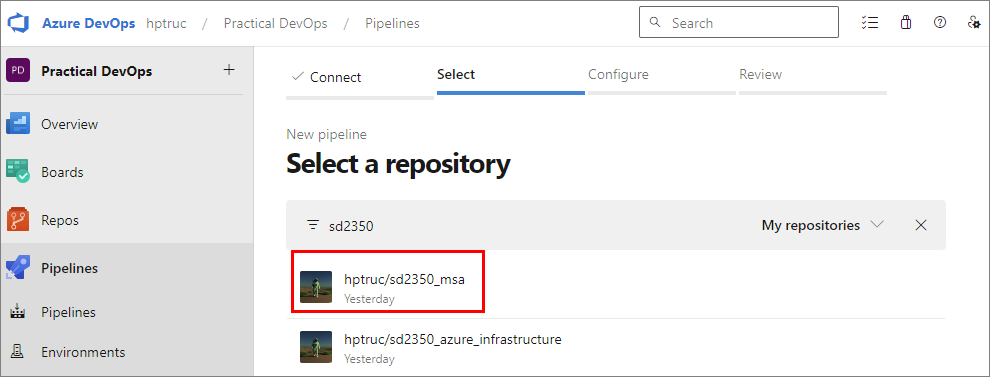
* Click on Pipelines item from left side bar > New pipeline
* Select GitHub method



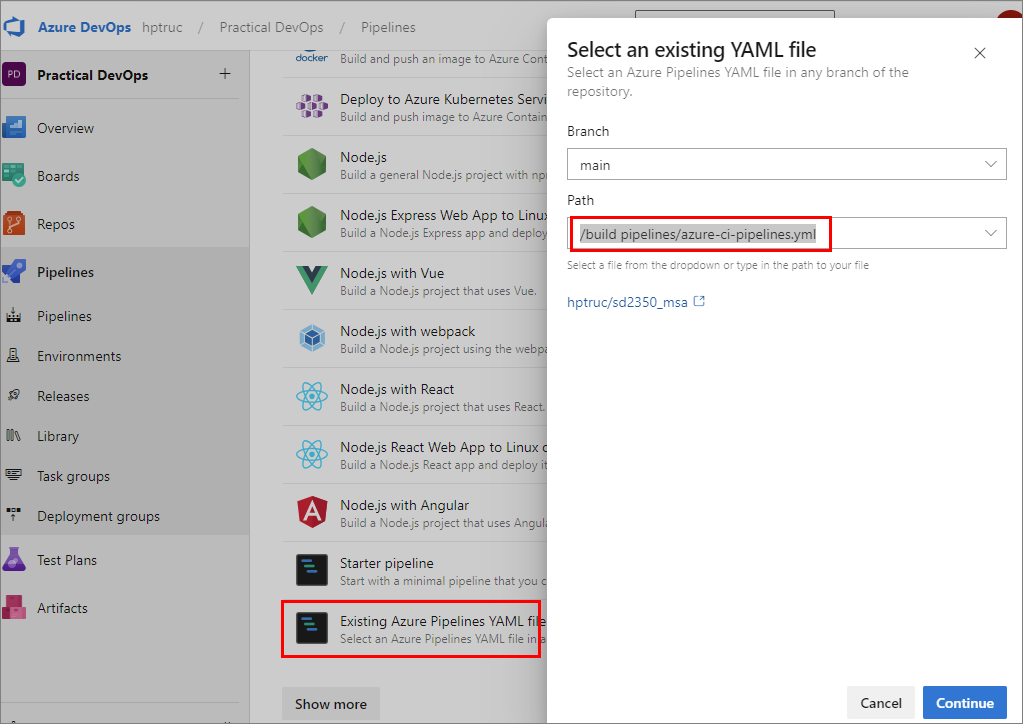
* Azure DevOps will redirect to GitHub authorize page



* Click on **Authorize Pipelines** button
* After authorize successfully, at the Azure DevOps page, select MSA repository



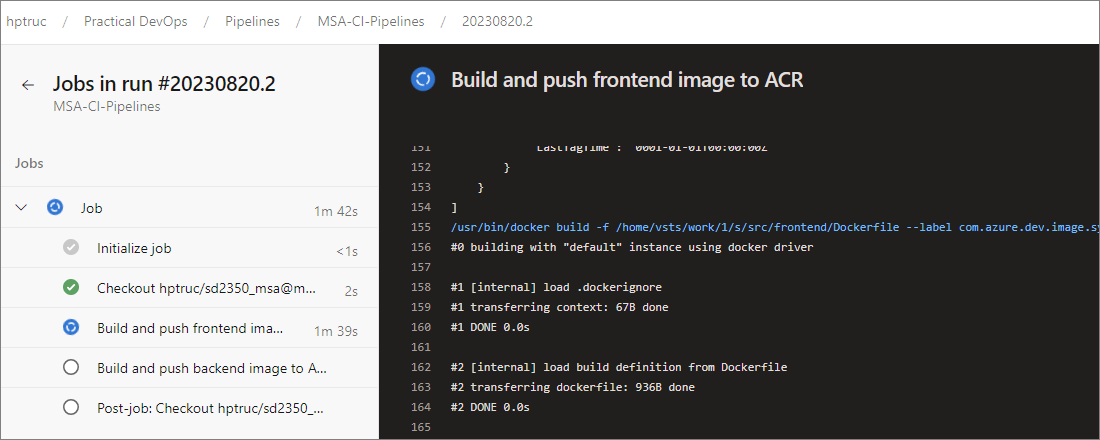
* Approve permission from GitHub page to access to this repository
* The next, in tab Configure > select item **Existing Azure Pipelines YAML file**
* In the select file popup > select [**ci pipeline** file](https://github.com/hptruc/sd2350_msa/blob/main/build%20pipelines/azure-ci-pipelines.yml) in Path dropdown



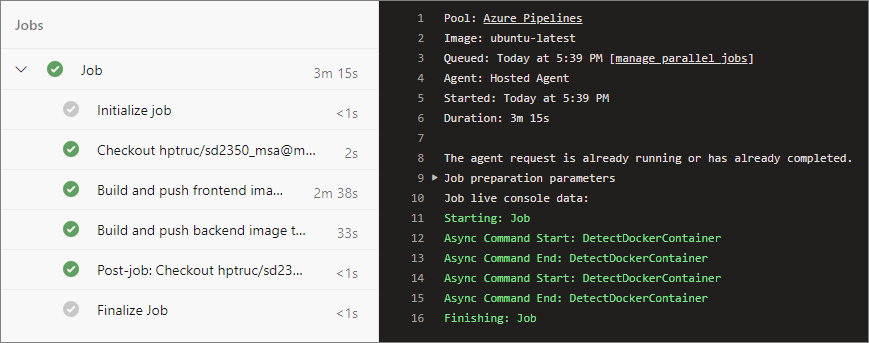
* Verify **ACR connection** to make sure that it work correctly.
* Save pipeline with name **MSA-CI-Pipelines**

## Run CI build pipeline and verify results

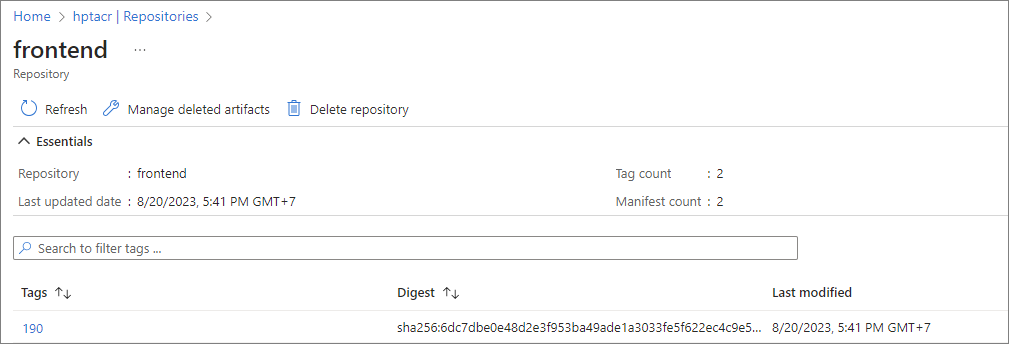
* Select pipeline **MSA-CI-Pipelines** and click on run button to start build pipeline



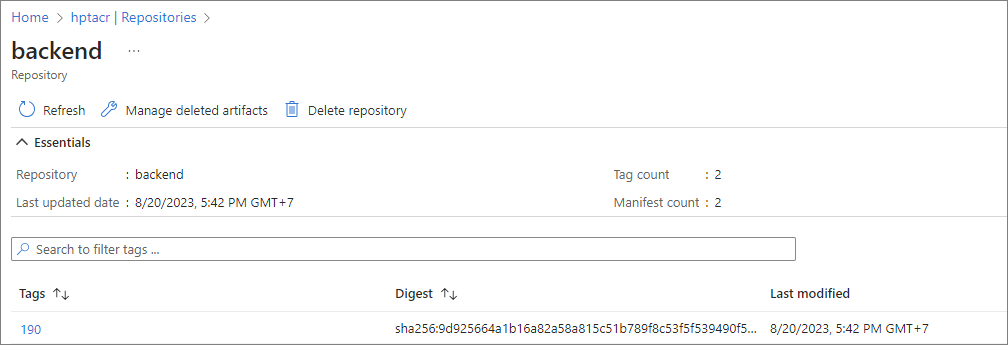
* Waiting for 3-5 minutes
* Check results from console to make sure that build pipeline complete without any error.



* Login to Azure portal to verify images
* Frontend image



* Backend image



* Complete Azure CI build pipeline.

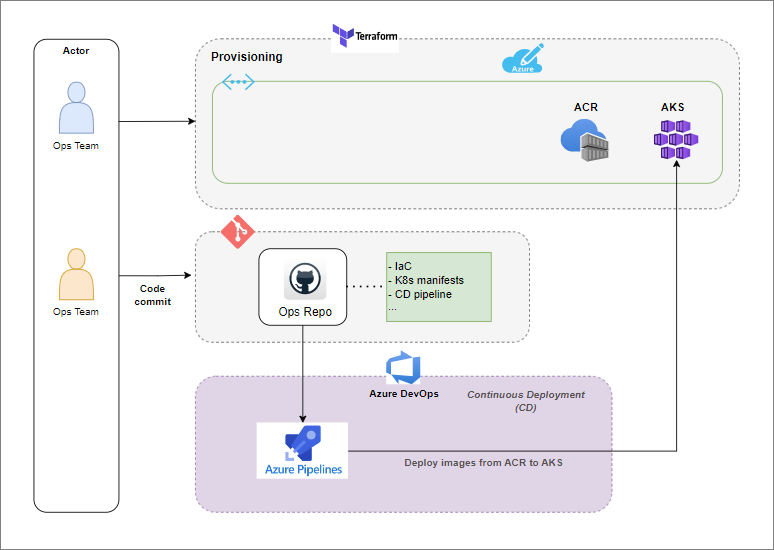
# CD: Setting up an azure pipeline and deploying applications on AKS

GitHub repository: [https://github.com/hptruc/sd2350\_azure\_infrastructure](https://github.com/hptruc/sd2350_azure_infrastructure/blob/main/k8s%20deploy%20strategies/azure%20pipeline/azure-cd-pipelines.yml)

Precondition:

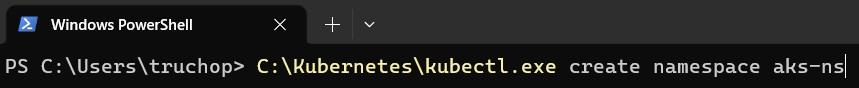
* CI build pipeline executed to push MSA images to Azure Container Registry

Architecture overview:

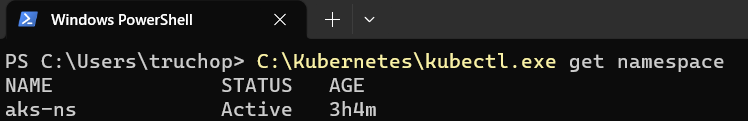


## Create a Kubernetes namespace using for deployment

* Open terminal and type this command to create a namespace
* Namespace: **aks-ns**

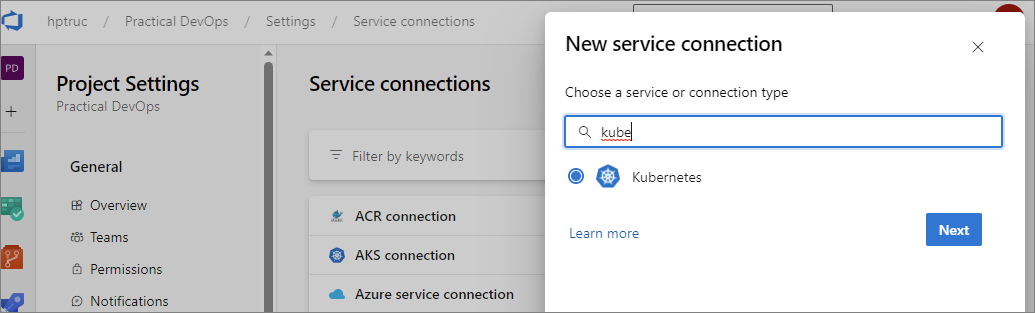


* Verify namespace created

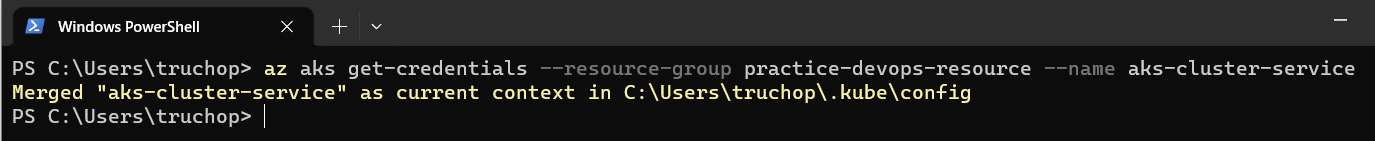


## Setting up AKS service connection

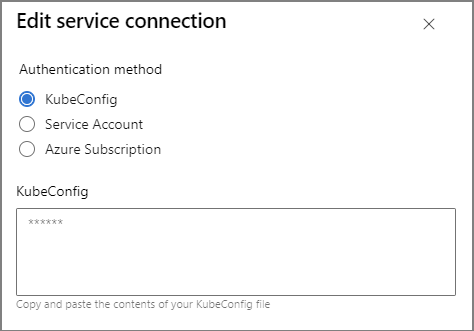
* Login to Azure DevOps instance > access **Practical DevOps** project
* Go to Project settings page > Service connections > New service connection
* Select **Kubernetes** and click on Next button



* You can choose one of three options depending on the Azure account you are using. In this demo, I choose the option KubeConfig.
* Open terminal and type this command to set Kubernetes credential to configure file

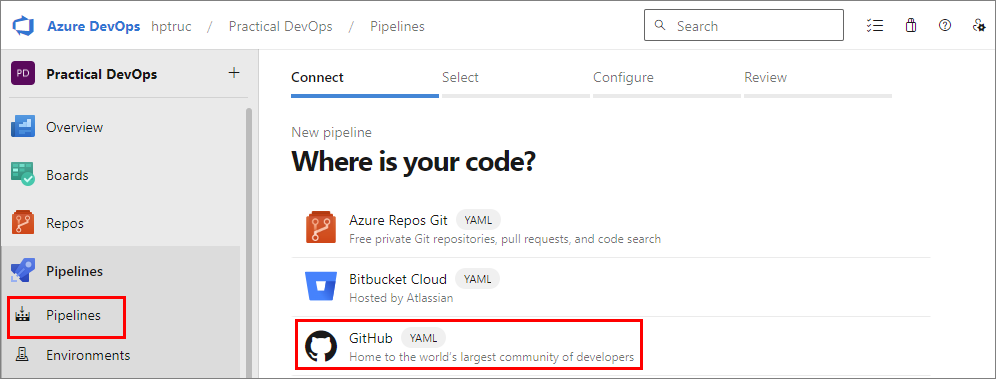


* Go to configure file with path in console (C:\Users\...)
* Copy all content of configure file and paste into textbox KubeConfig
* Click on verify button and save service connection with name **AKS connection**



## Create new a CD pipeline and connect to Infrastructure GitHub repository

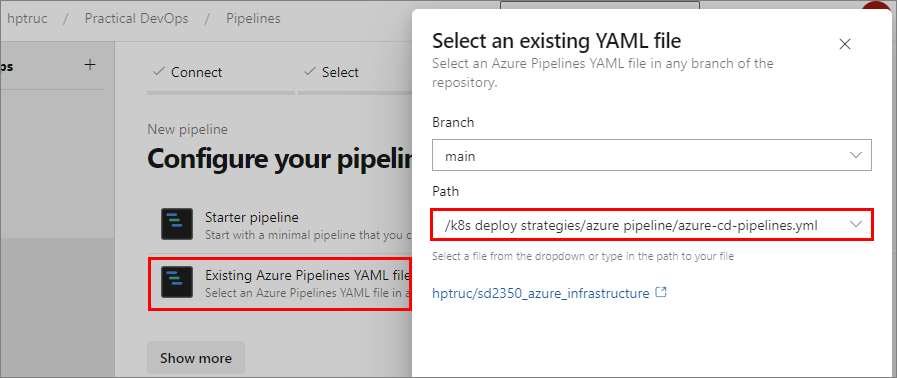
* Click on Pipelines item from left side bar > New pipeline
* Select GitHub method



* In the Azure DevOps page, select infrastructure repository



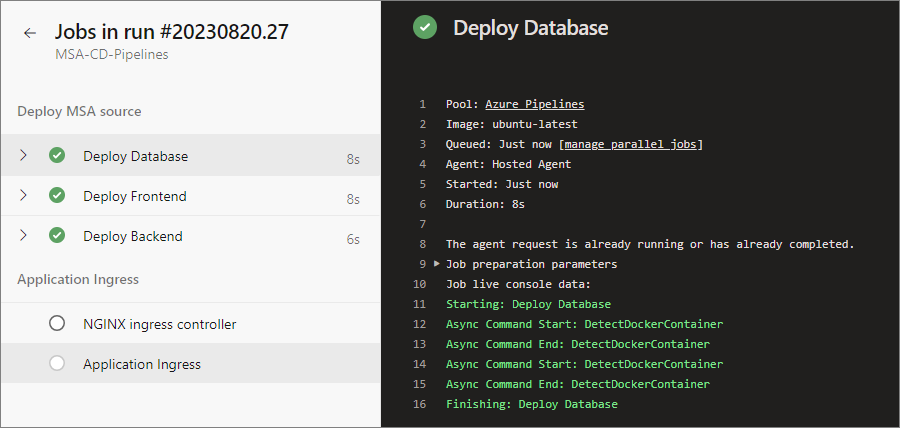
* Approve permission from GitHub page to access to this repository
* The next, in tab Configure > select **Existing Azure Pipelines YAML file** item
* In the select file popup > select [**cd pipeline** file](https://github.com/hptruc/sd2350_azure_infrastructure/blob/main/k8s%20deploy%20strategies/azure%20pipeline/azure-cd-pipelines.yml) in Path dropdown



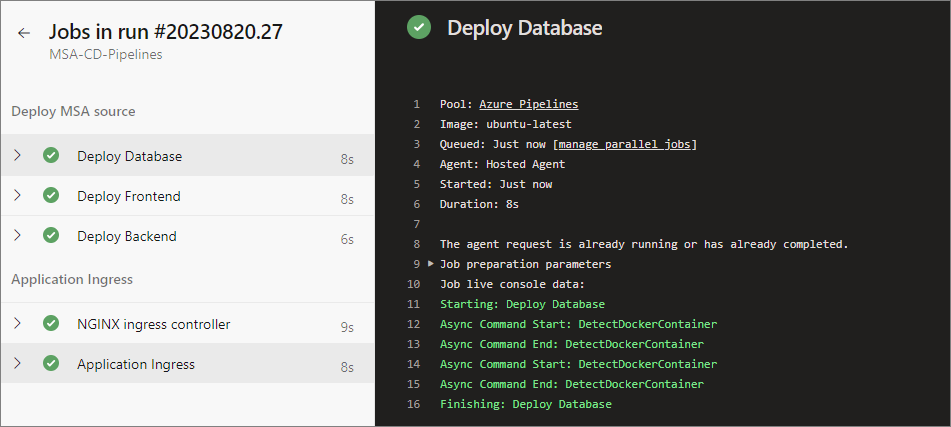
* Save pipeline with name **MSA-CD-Pipelines**

## Run CD build pipeline and verify results

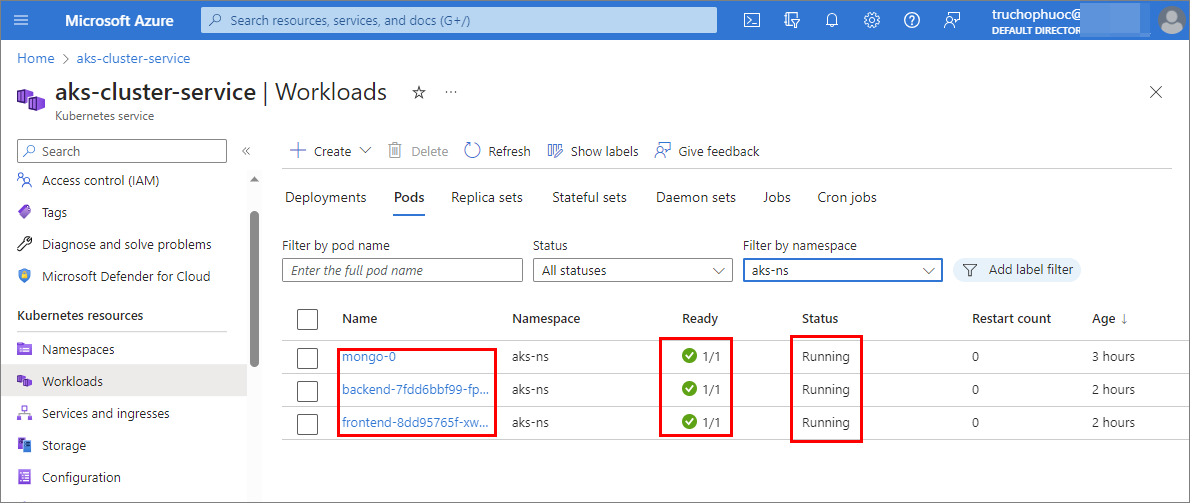
* Select pipeline **MSA-CD-Pipelines** and click on run to start build pipeline



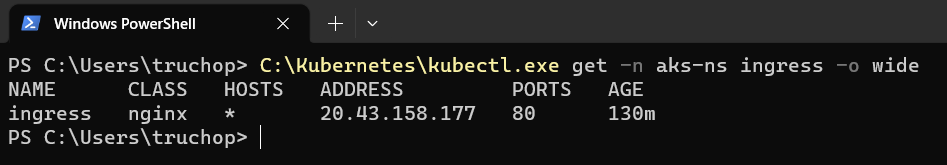
* Waiting for 3-5 minutes
* Check result from console logs to make sure build pipeline complete without any error



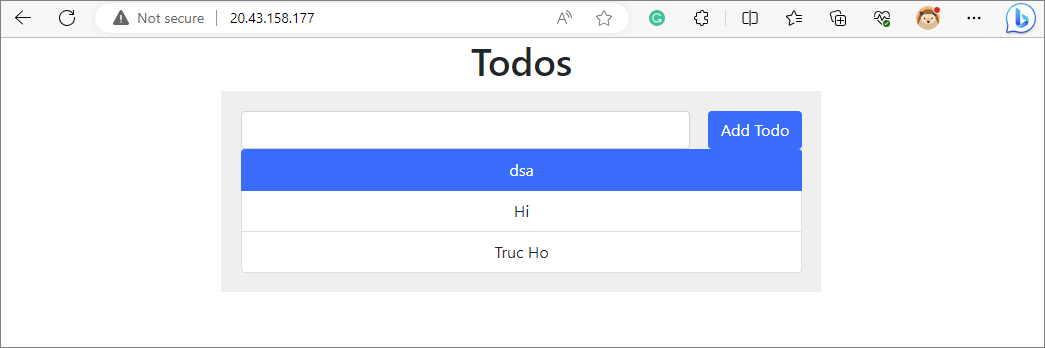
* Login to Azure portal to verify images



* Open terminal and type this command



* Copy ingress IP ADDRESS and paste to browser to go to web page



* Complete Azure CD deployment pipeline.

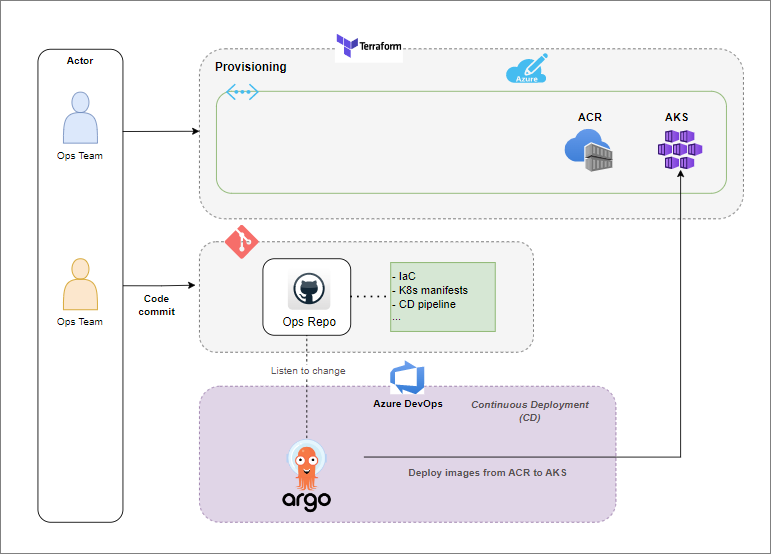
# GitOps: Replace azure pipeline CD by Argo CD

GitHub repository:

Precondition:

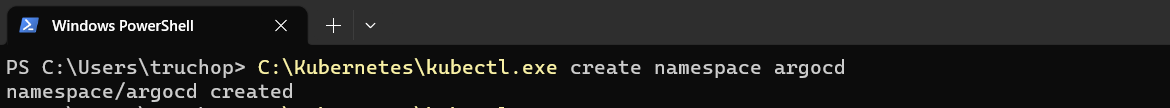
* CI build pipeline executed to push MSA images to Azure Container Registry.

Architecture overview:

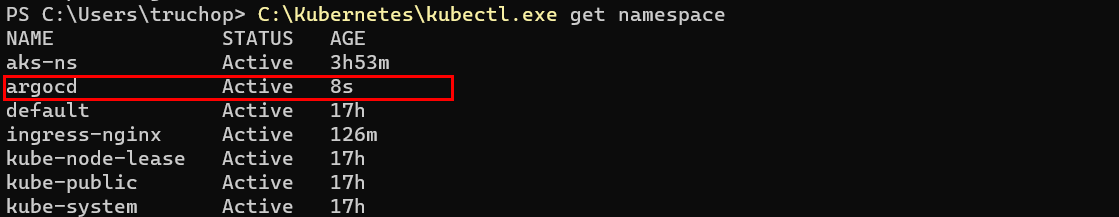


## Install Argo CD on AKS

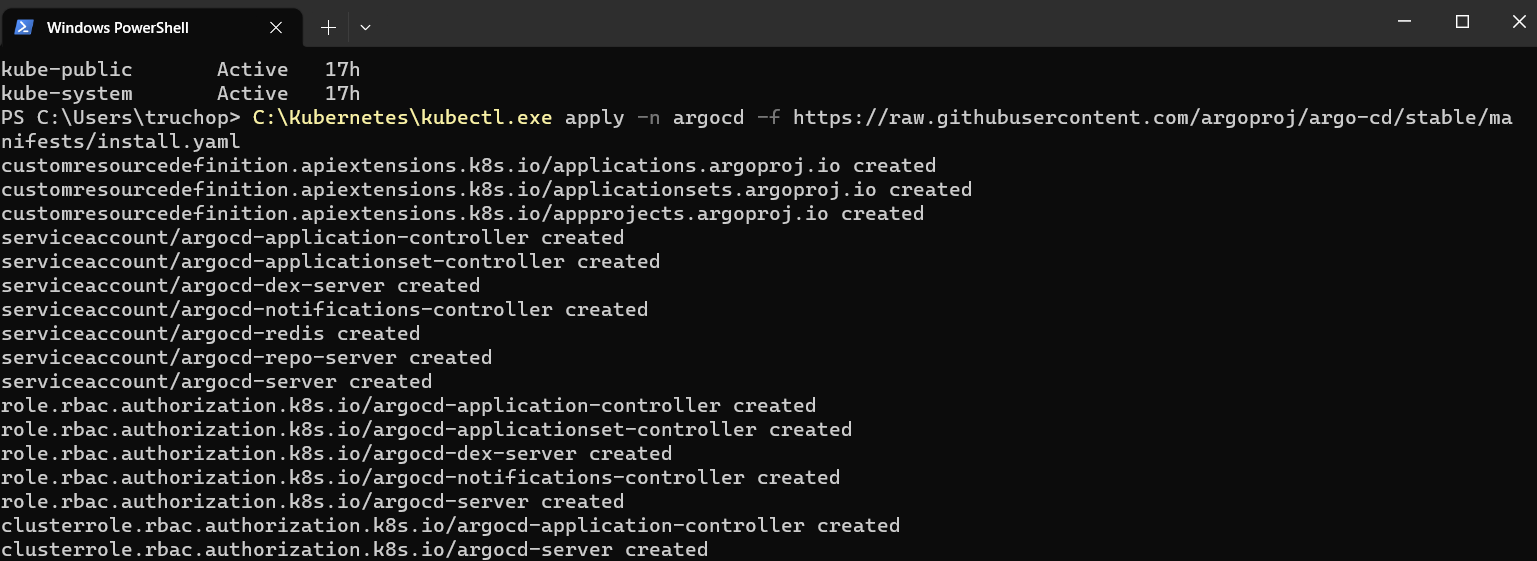
* Open terminal and create a namespace for Argo CD
* Namespace: argocd



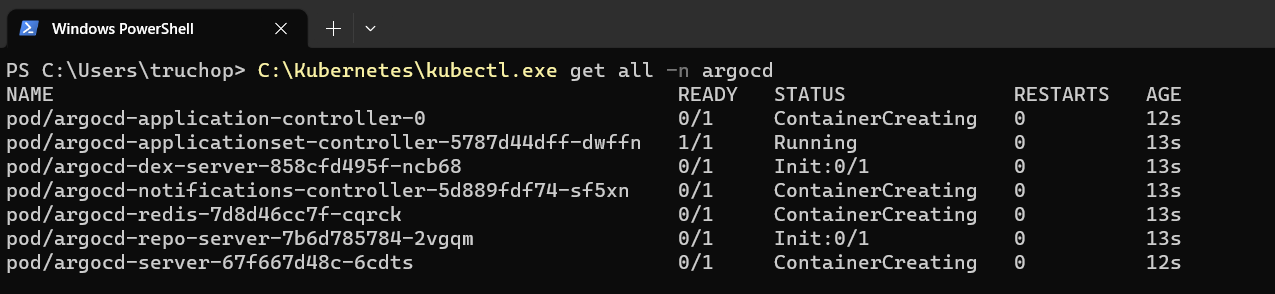
* Verify namespace created



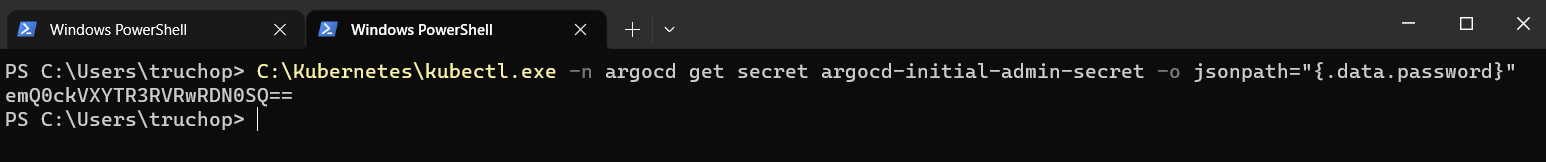
* Install Argo CD by script: <https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml>



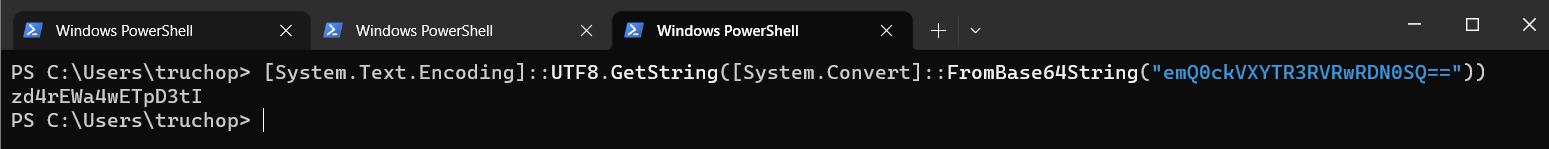
* Verify Argo CD installed successfully



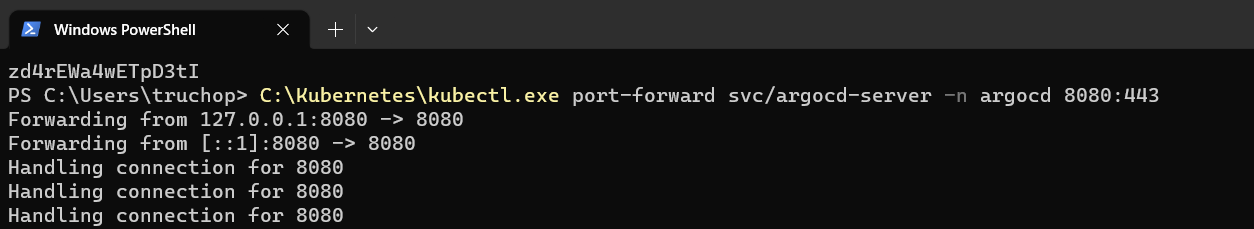
* Get raw password (base 64) login to Argo CD



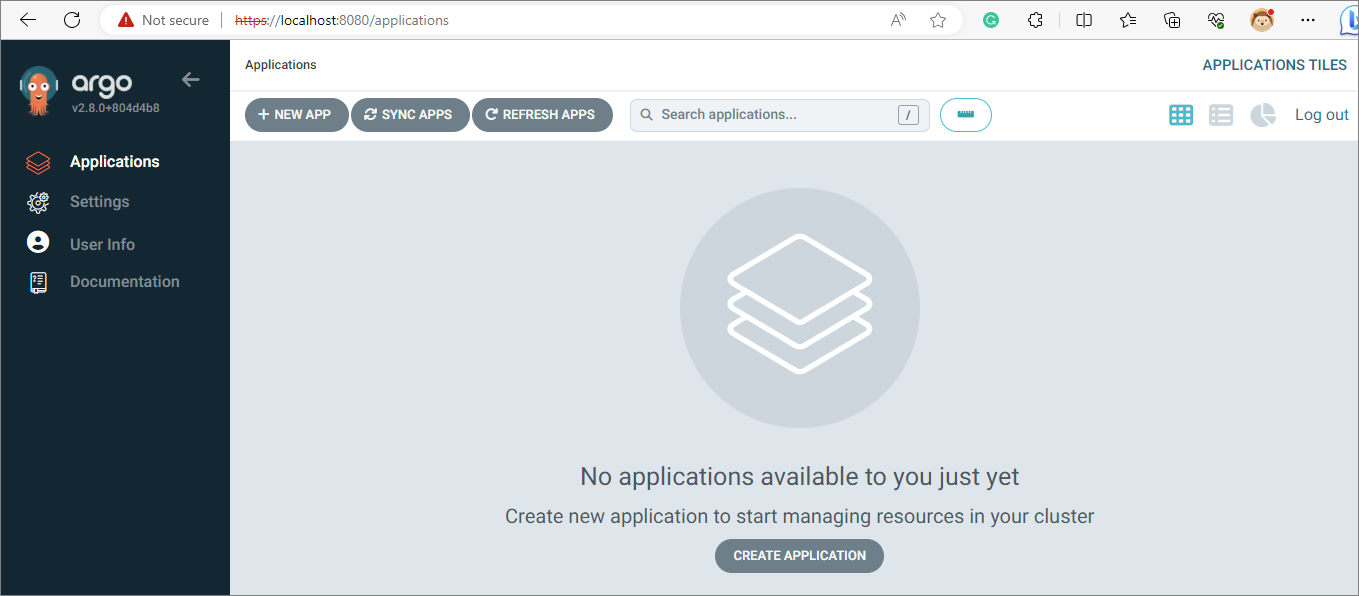
* Decode password by command



* Finally, use port forwarding to connect to the `argocd-server` service in AKS cluster.



* Open browser and go to address: <http://localhost:8080> with credential
  + Username: admin
  + Password: { decoded in previous step }



## Add new applications to Argo CD

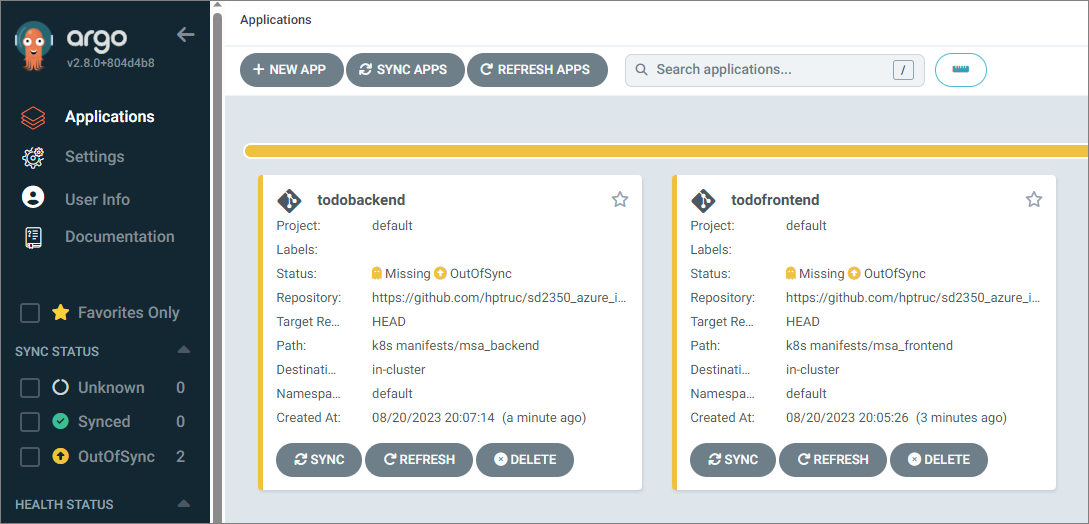
* In Argo CD portal, click on New App button to create new application
* For demo purpose, I will create 2 applications Frontend and Backend (skip database)
* Frontend

|  |  |
| --- | --- |
| Field Name | Value |
| Application Nam | todofrontend |
| Project Name | default |
| Sync Policy | Manual |
| Repository URL | <https://github.com/hptruc/sd2350_azure_infrastructure.git> |
| Path | k8s manifests/msa\_frontend |
| Cluster URL | <https://kubernetes.default.svc> |
| Namespace | default |

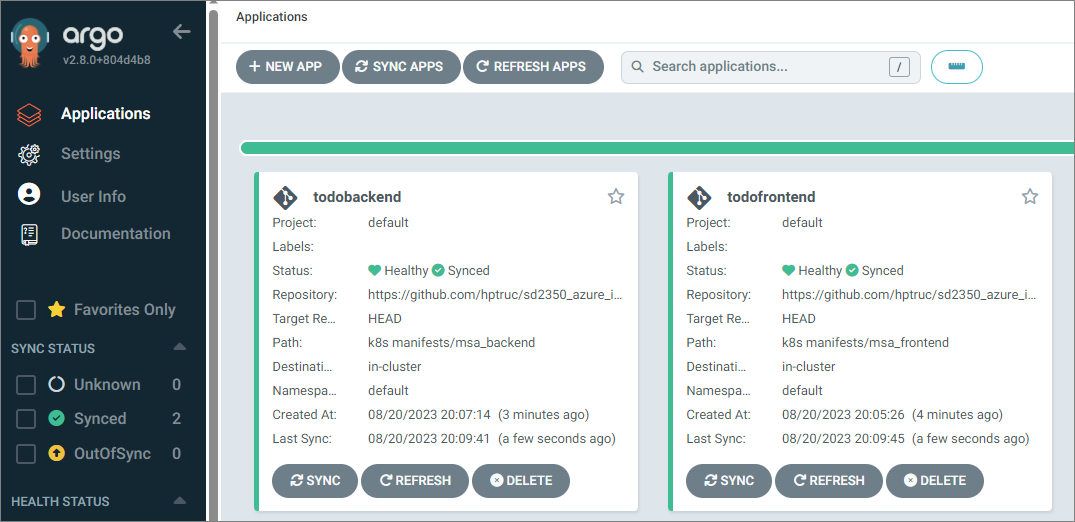
* Backend

|  |  |
| --- | --- |
| Field Name | Value |
| Application Nam | todobackend |
| Project Name | default |
| Sync Policy | Manual |
| Repository URL | <https://github.com/hptruc/sd2350_azure_infrastructure.git> |
| Path | k8s manifests/msa\_backend |
| Cluster URL | <https://kubernetes.default.svc> |
| Namespace | default |

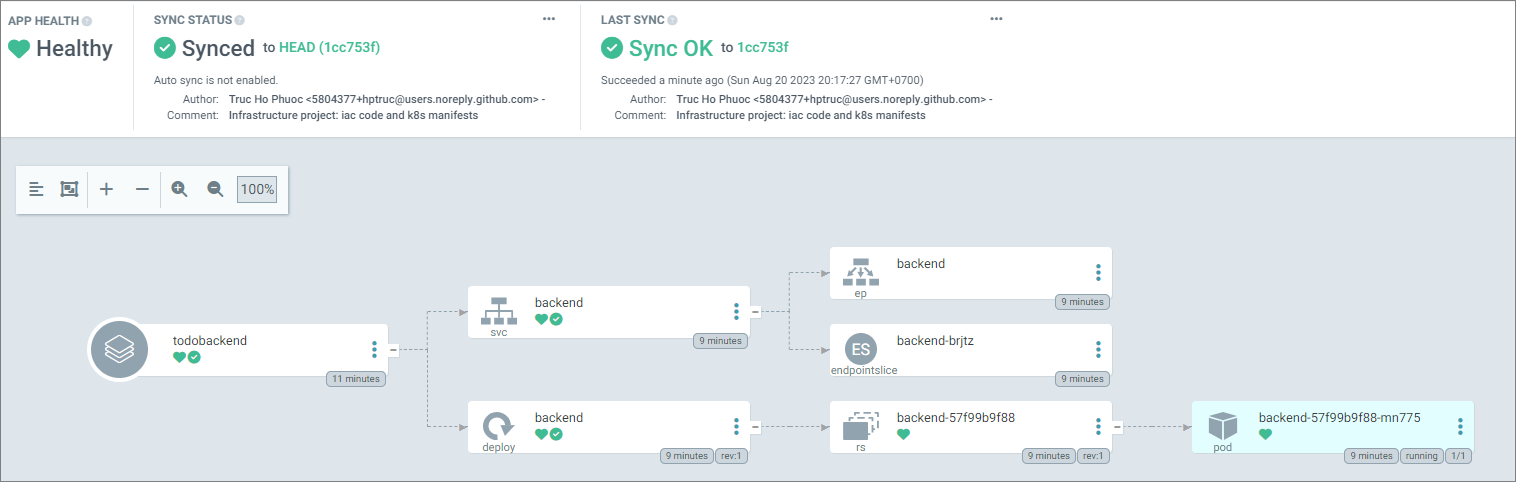
* Click on Create button to create application
* Initially, you will notice your application sync status is **OutOfSync** and your health status is **Missing**



* Click on SYNC button > SYNCHRONIZE

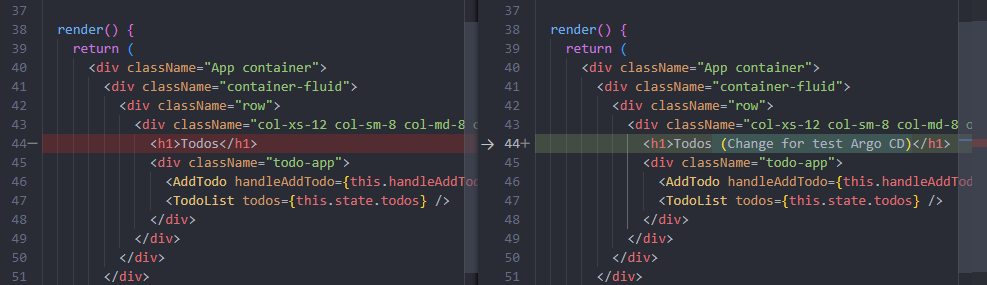


* Verify status of applications after sync

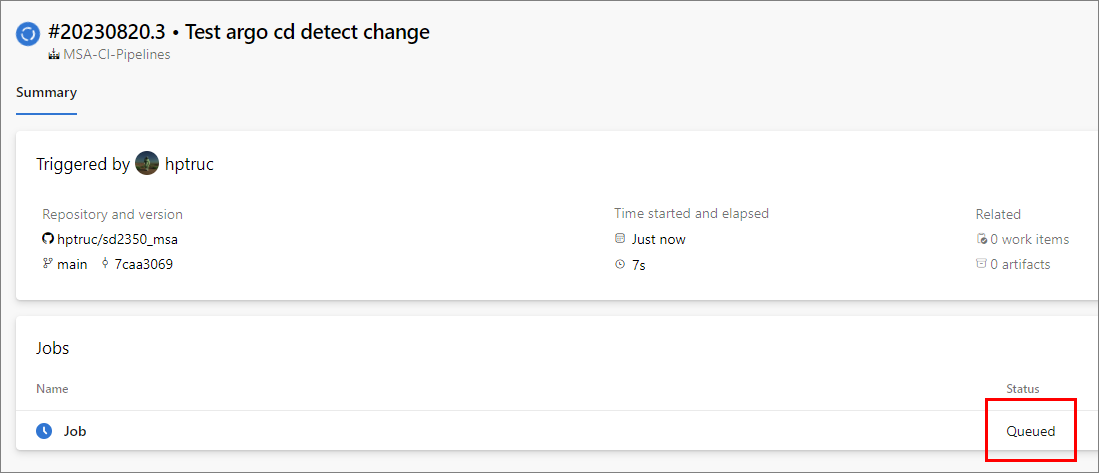


## Test deployment with Argo CD

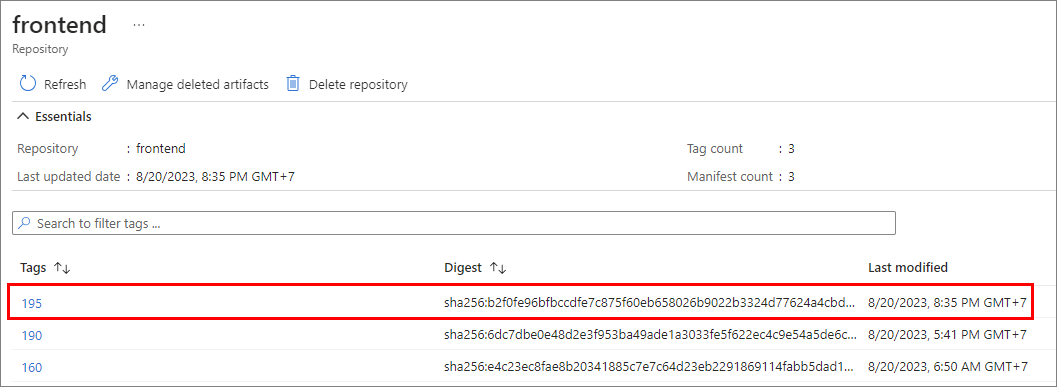
* Open MSA frontend source code and change something then commit change to MSA repo to build pipeline trigger create new image on ACR
* I will change as below



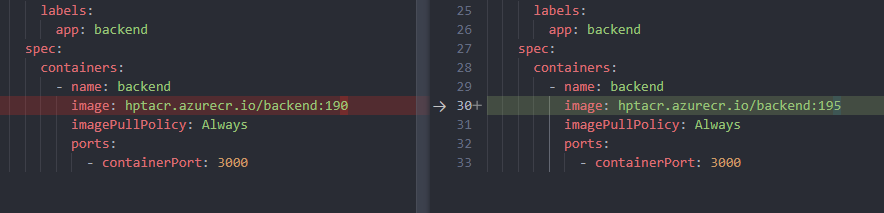
* Commit code change
* CI trigger build

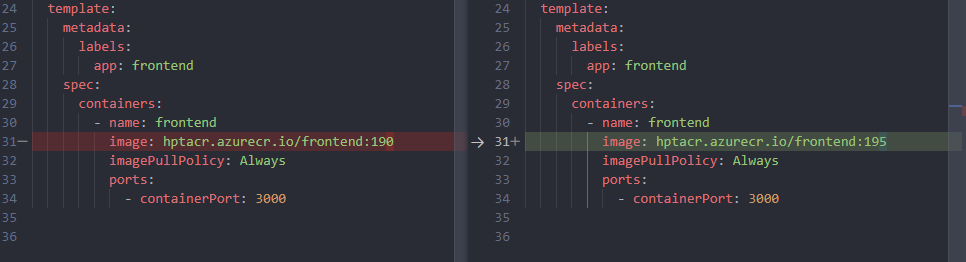


* Login to Azure portal to check new image tag

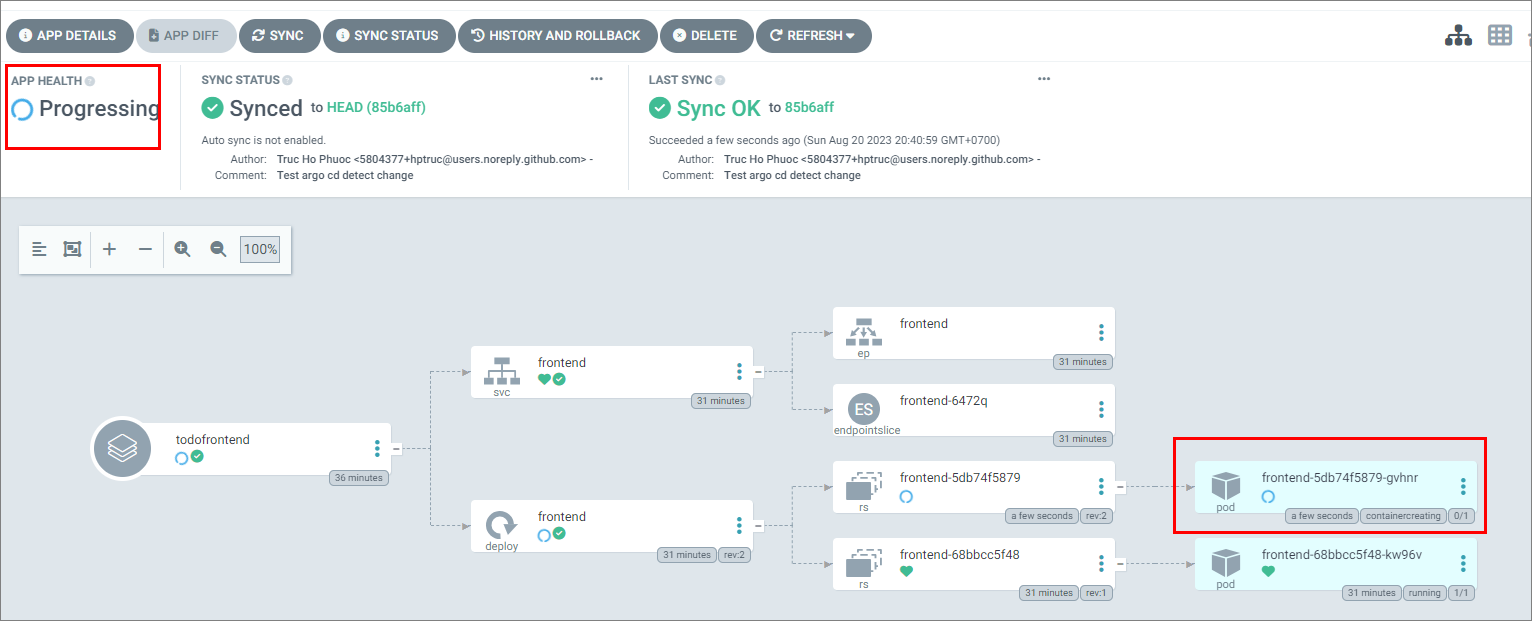


* New image tag: 195
* Update K8s manifest of frontend to get image by new tag 195



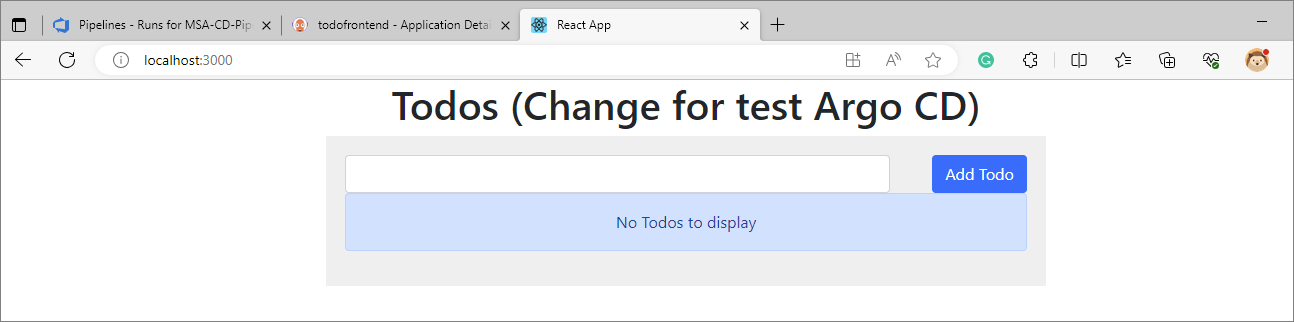


* Commit code change
* Login to Argo CD dashboard, status of repo changed to **OutOfSync**
* Click on SYNC button
* Argo CD triggered deploy to AKS automatically



* Forward port to localhost and try access to AKS



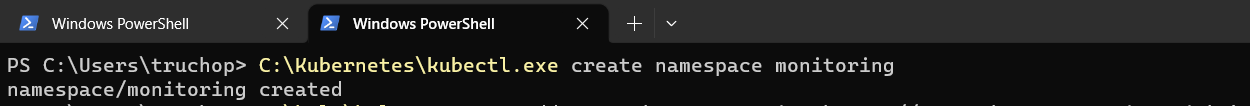


* Argo CD deployed new image successfully.

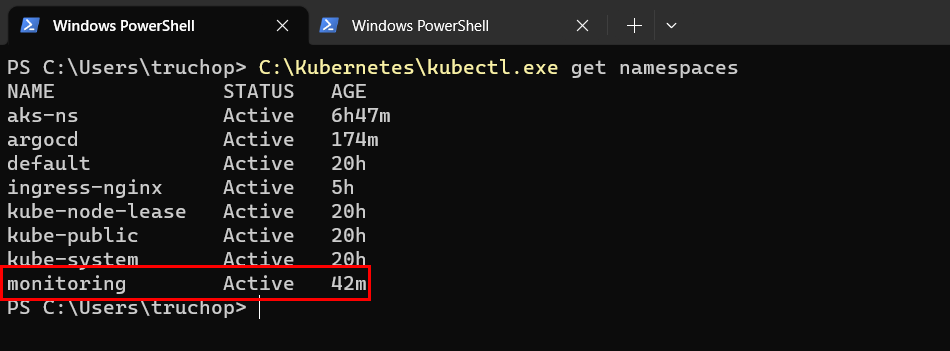
# Monitoring: Setup Prometheus and Grafana to monitor AKS resources

## Create a namespace for monitoring

* Open terminal and create a namespace with name **monitoring**

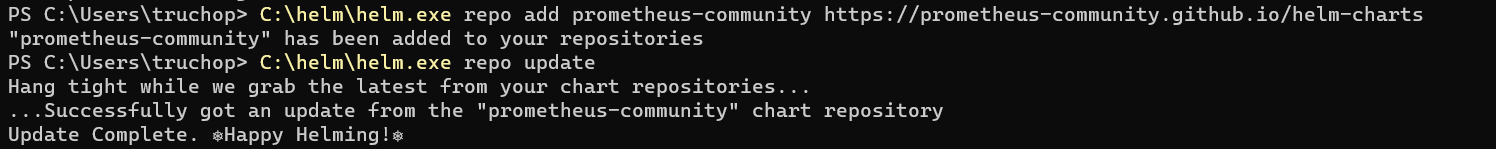


* Verify namespace created

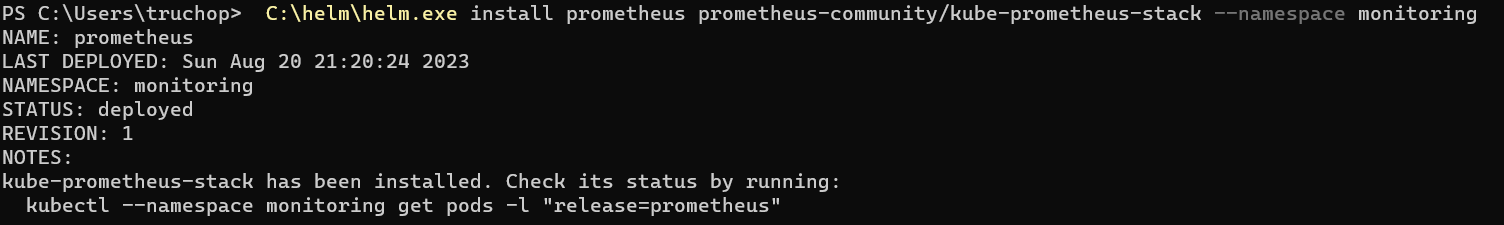


## Install Prometheus and Grafana tools

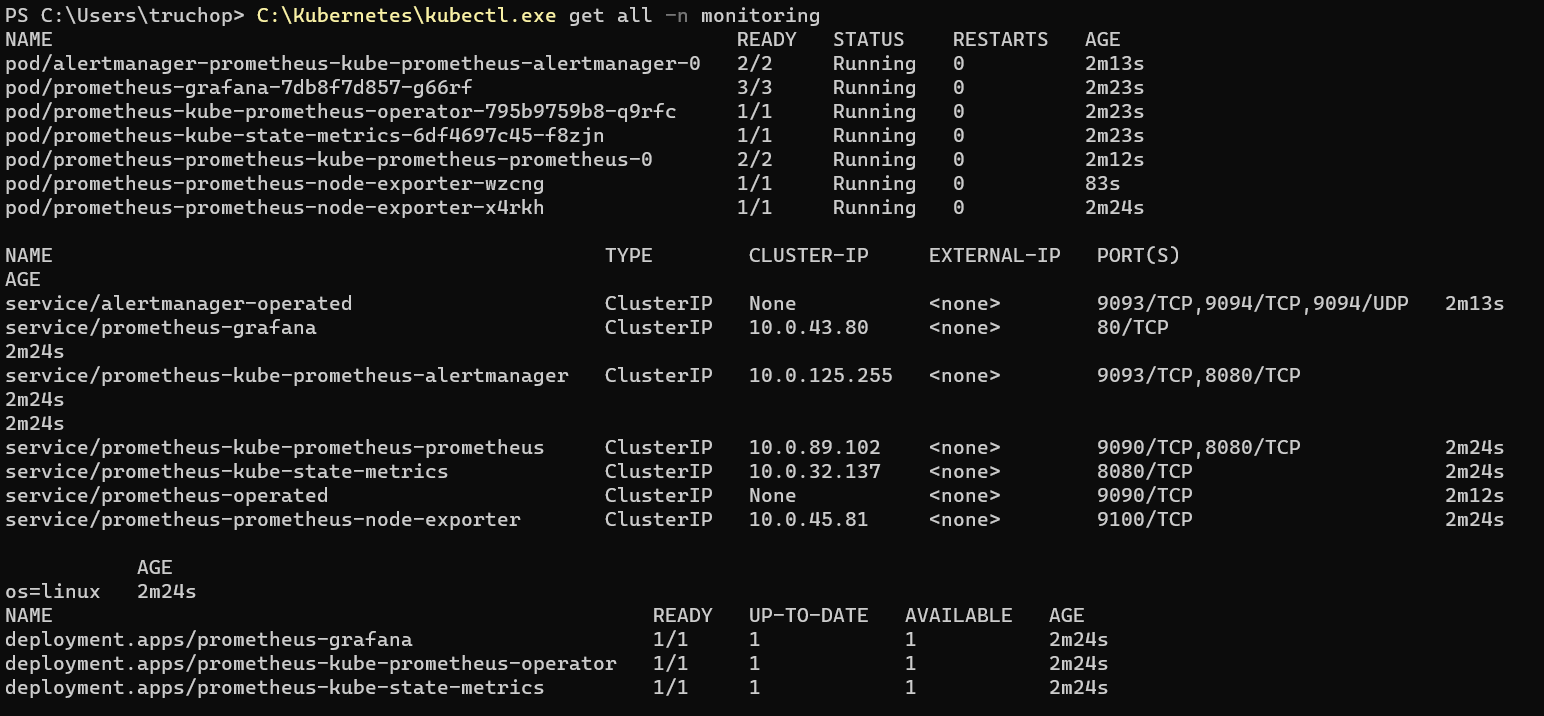
* Install helm charts



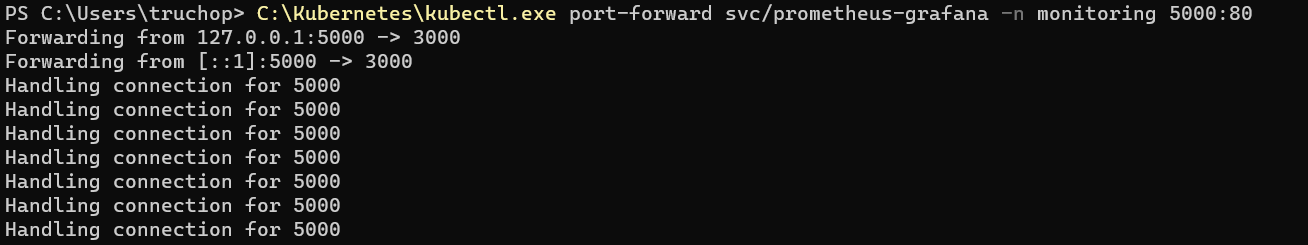
* Install Prometheus tool



* Verify Prometheus & Grafana installed

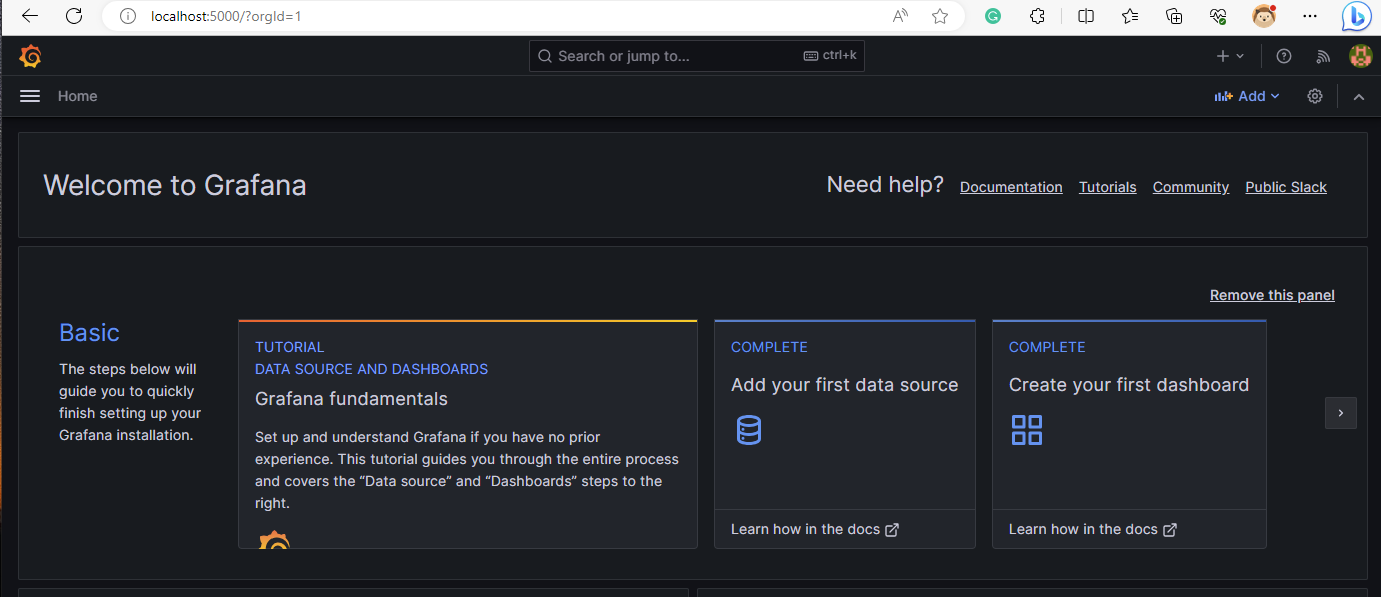


* Forward port to localhost to access dashboard

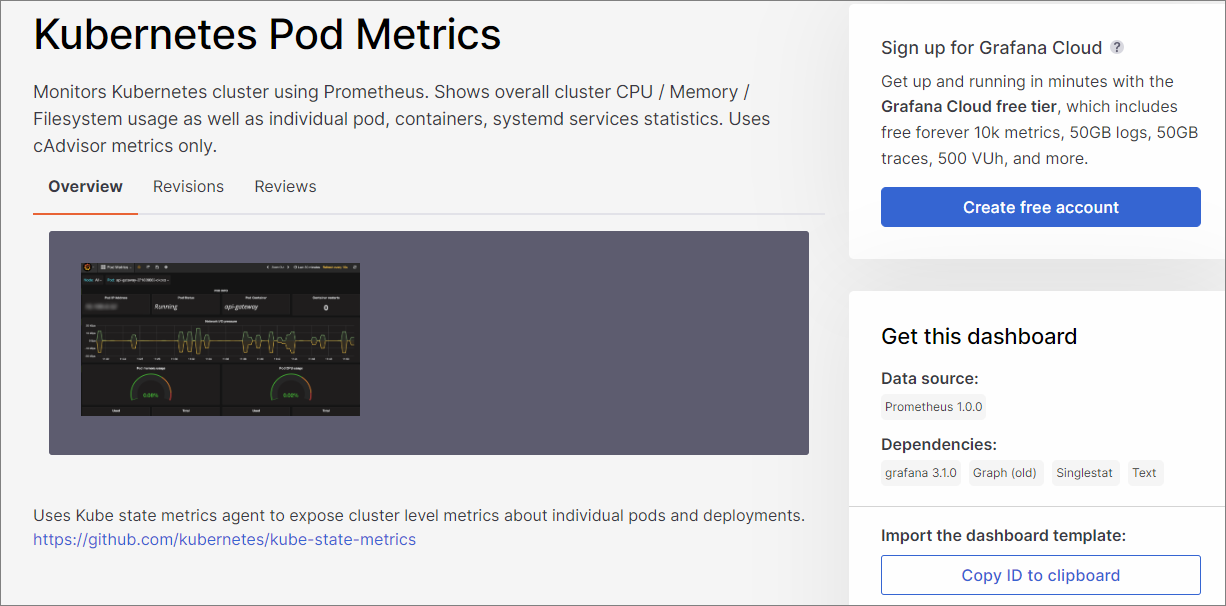


## Verify dashboard monitoring

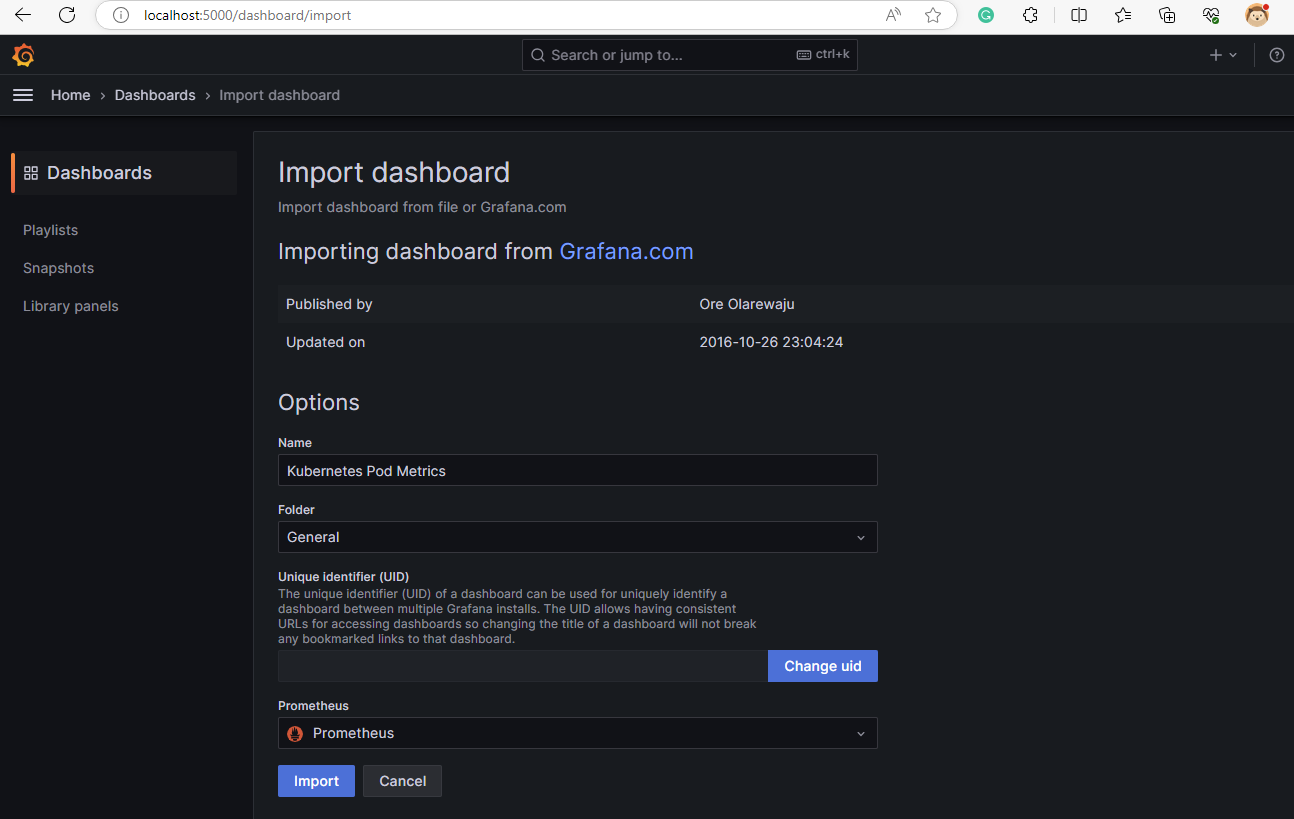
* Open browser and paste link: <http://localhost:5000>
* Credential default: admin/prom-operator

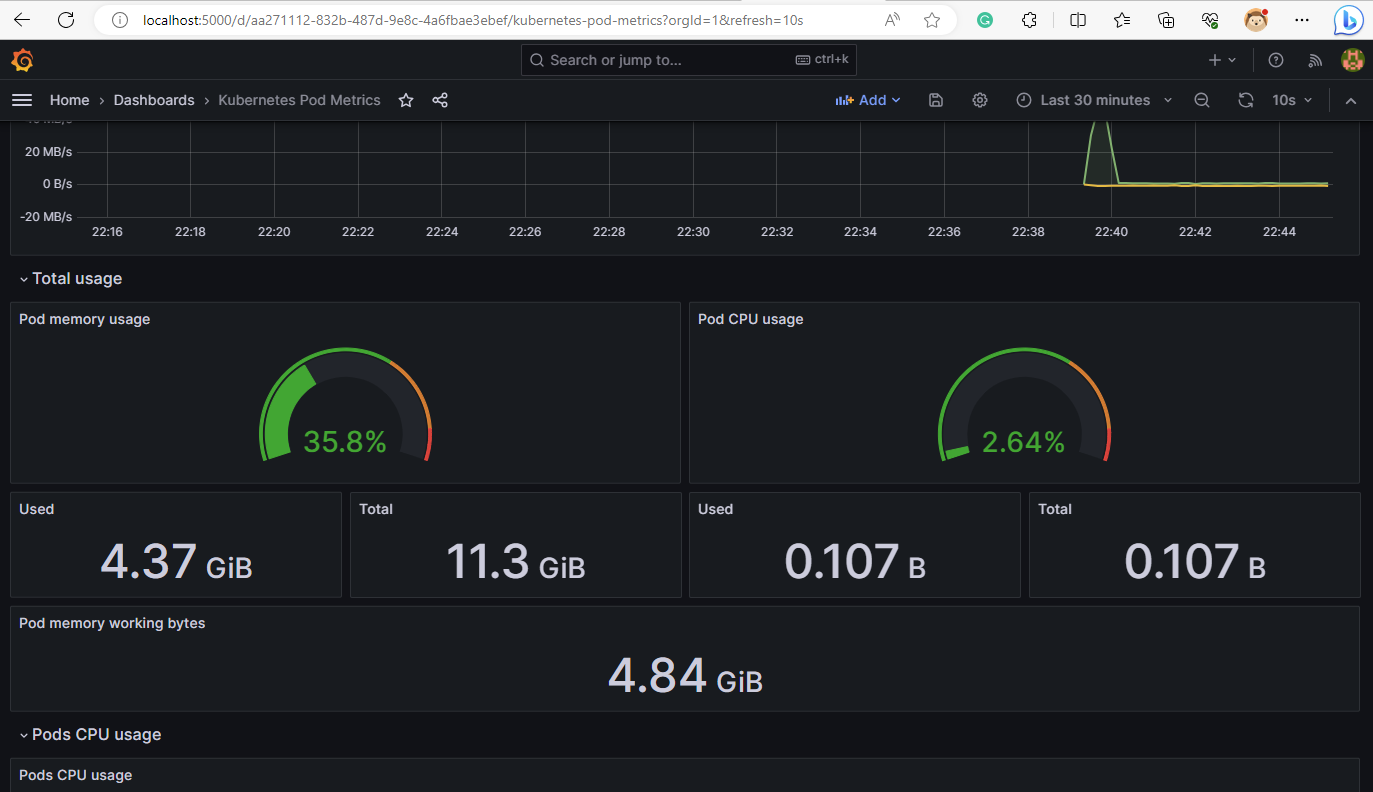


* Go to Grafana [home page](https://grafana.com/grafana/dashboards) and select a type dashboard to import (by ID). For example, I select this dashboard: Kubernetes Pod Metrics. ID 747



* Import to our dashboard





--- End documents ---