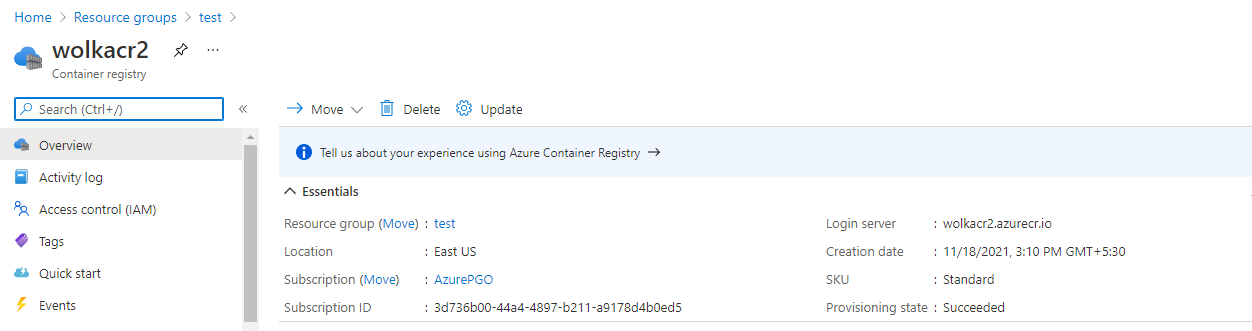
Project 2 - Azure DevOps Pipelines | Aashiq Ahamed SP | Sa00570532 | Azure DevOps – CFS

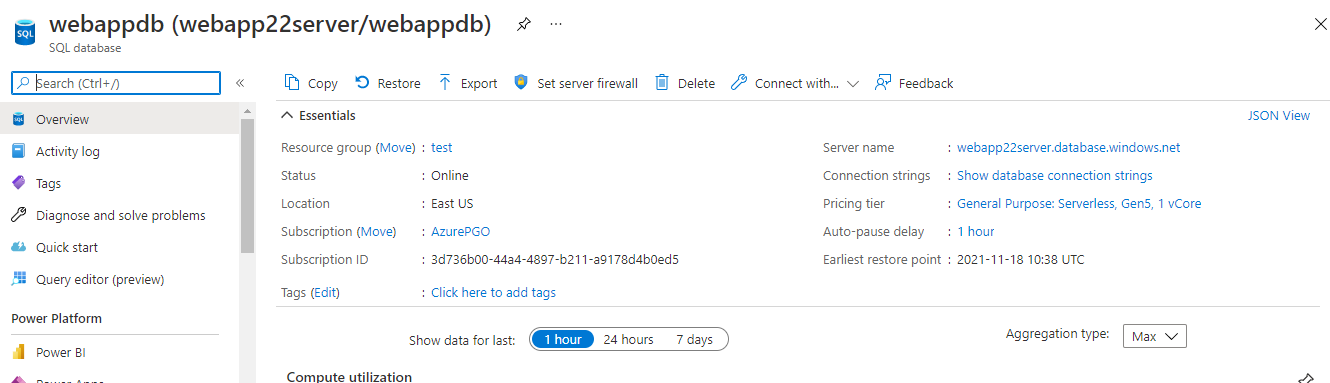
Task 1:

1. Create Azure Container Registry for storing .net core web application images.

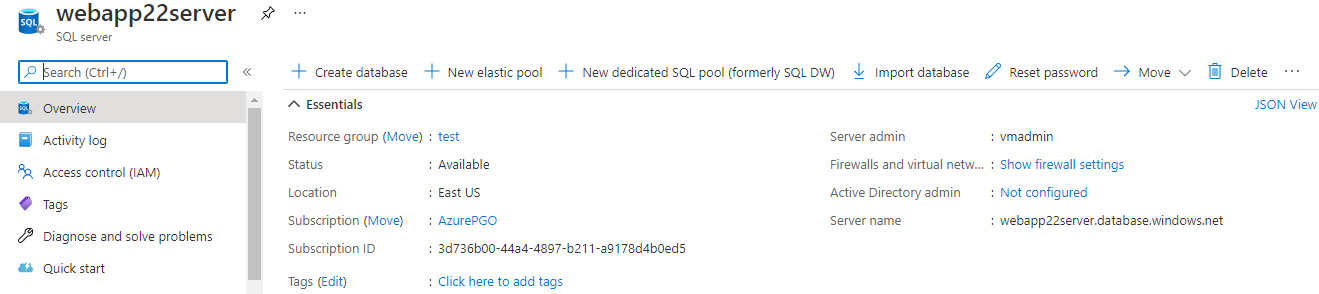
* Image created in azure pipeline Docker Build and Push job will be stored in this Registry.



1. Create Azure SQL DB to work as backend for .net core webapp

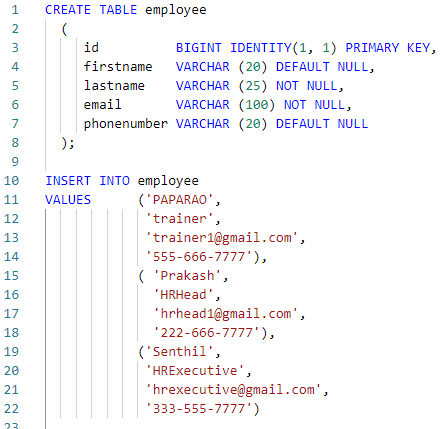


* Azure SQL DB with server price model (server less) has been created



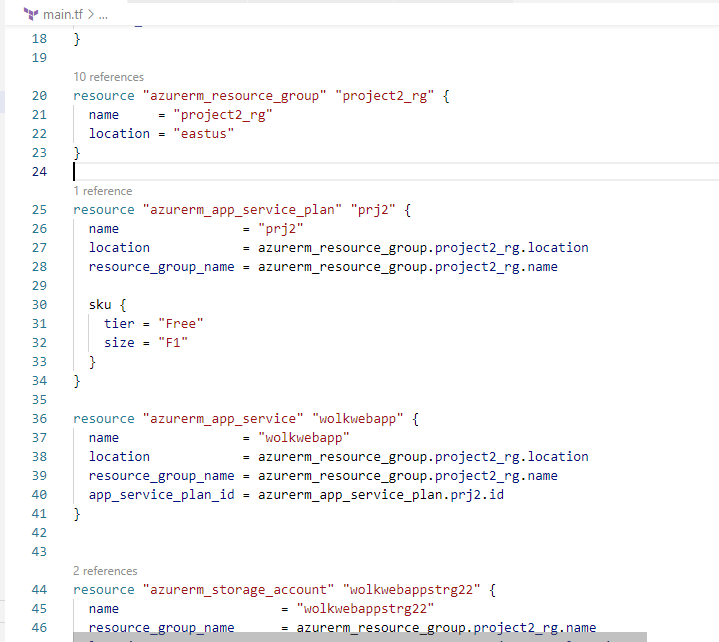
1. Creating table and inserting data into the SQL DB using VS 2019 SQL server explorer.

* Connect through SQL server explorer to webappdb and create data and view .



Task 2:

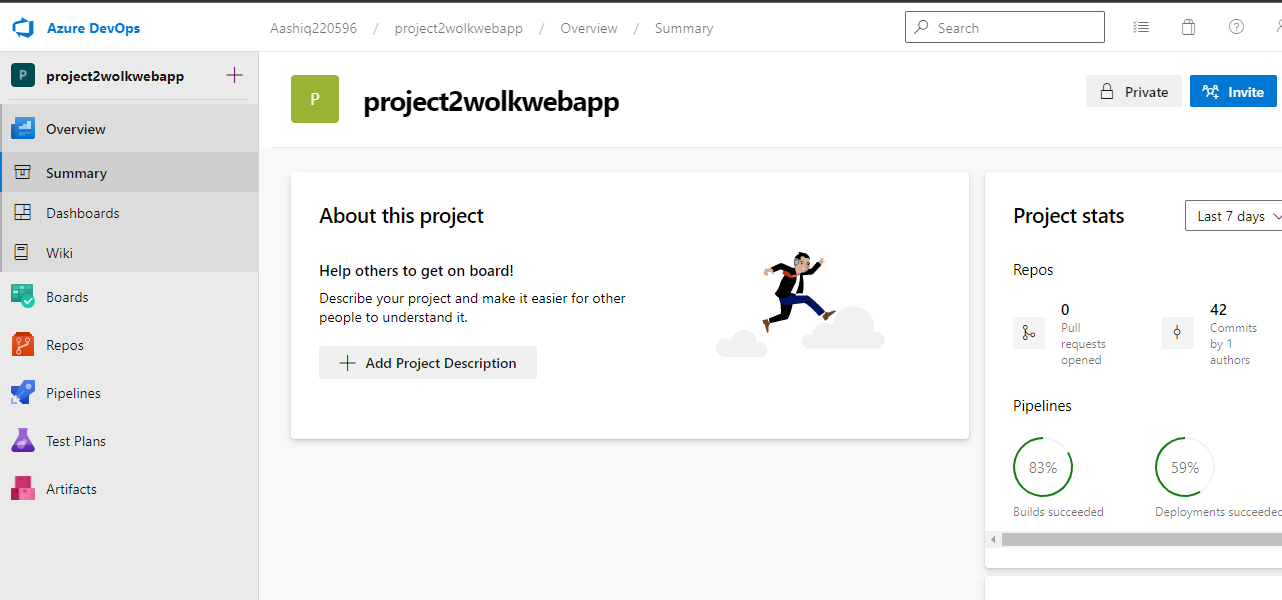
1. Creating Terraform script to deploy below resources:
   1. Resource Group
   2. Azure App service plan
   3. Azure app service
   4. Azure Storage account, Container and Blob to store logs
   5. AKS cluster



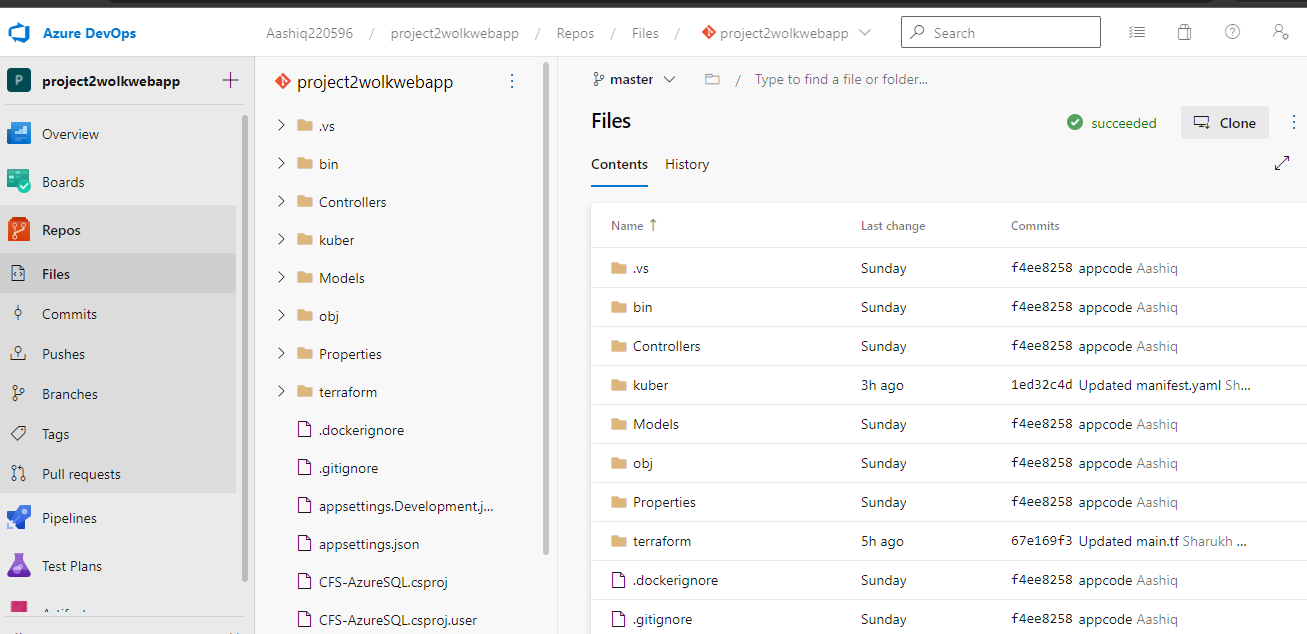
* This terraform script will access already available ACR service in azure.
* Using Role assignment block the user defined managed Identity of AKS cluster will be given Contributor role under ACR to form integration between AKS and ACR.
* Using Azure pipelines this terraform script will be deployed in CD release pipeline.

Task 3:

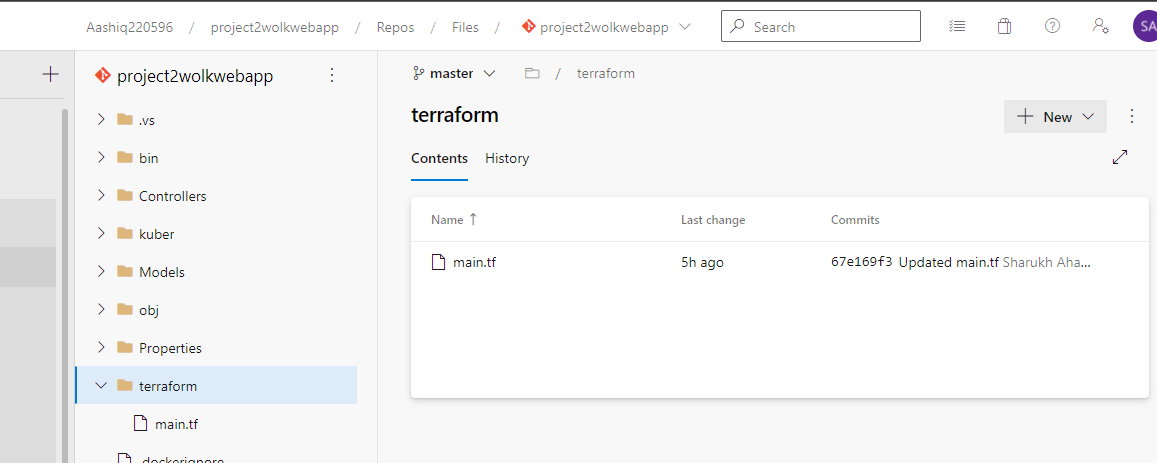
1. Create new project under Azure devops organization for this project



1. Push the .net core webapp code into Azure repos using VS code

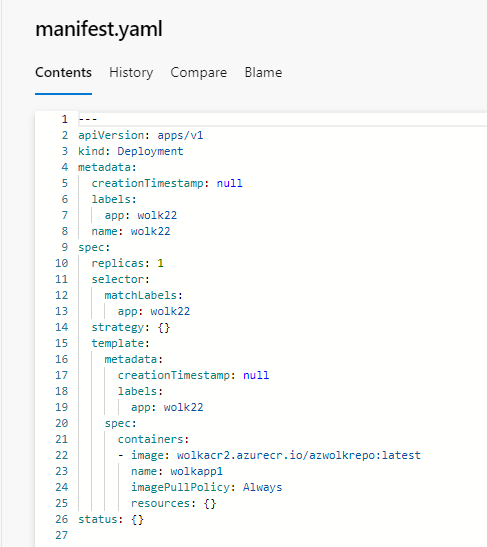


1. Create a folder under master for terraform .tf file

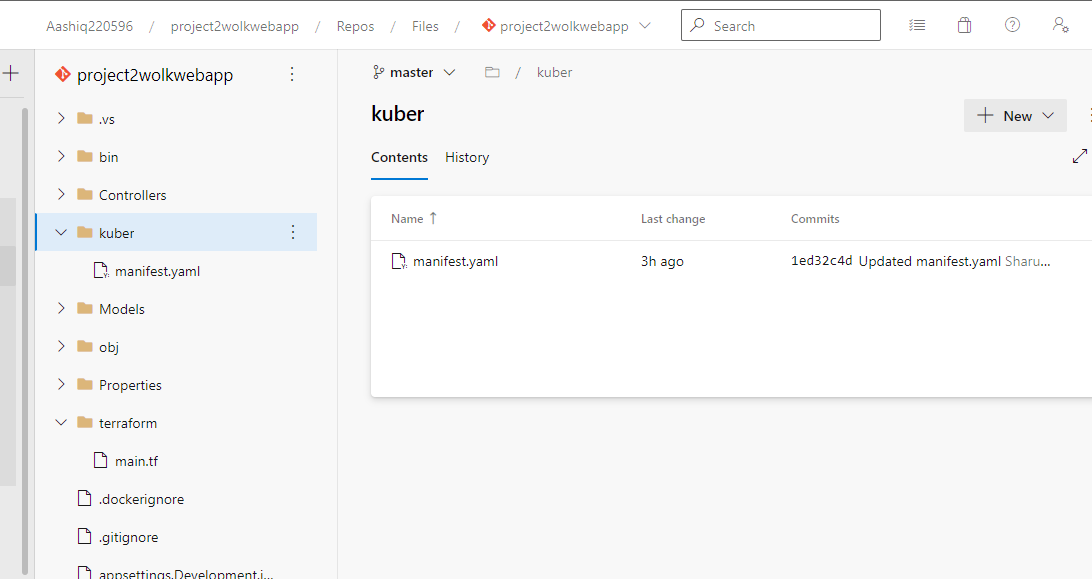


1. Create a deployment and service yaml manifest file to deploy containerized webapp image into AKS

Image is taken from ACR pushed in the same azure build pipline

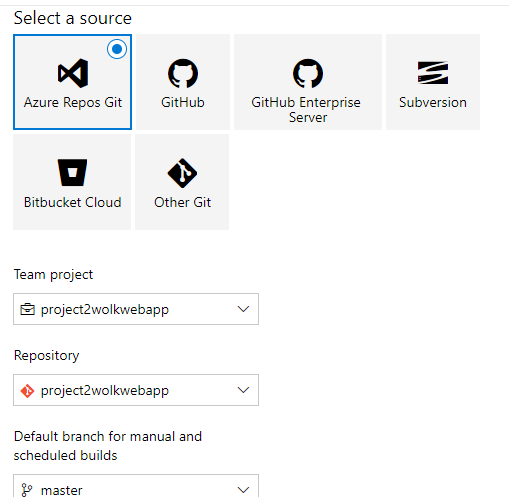


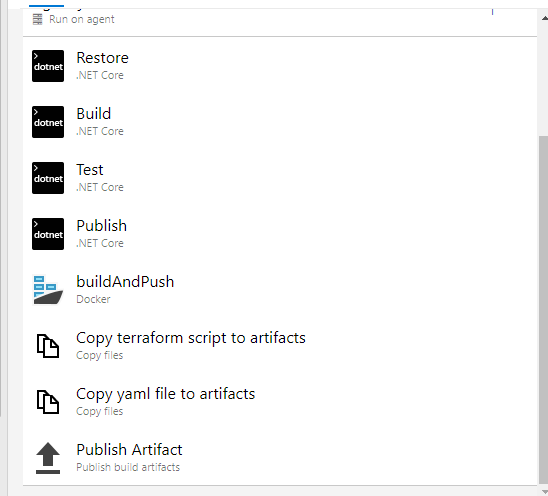
1. Create a folder kuber to store kubernetes yaml file manifests



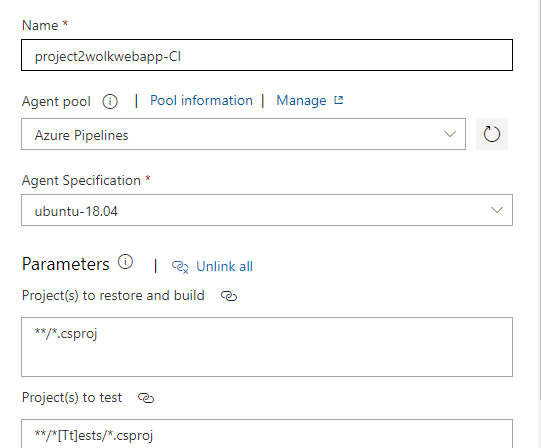
1. Create Build pipe line in azure devops to:

* Fetch app code from repos and build application
* Use Dockerfile in the repo and to build docker image and push it to ACR
* Copy the terraform .tf, Kubernetes .yaml and app code .zip to drop folder in artifacts
* Publish the artifacts for usage in release pipelines



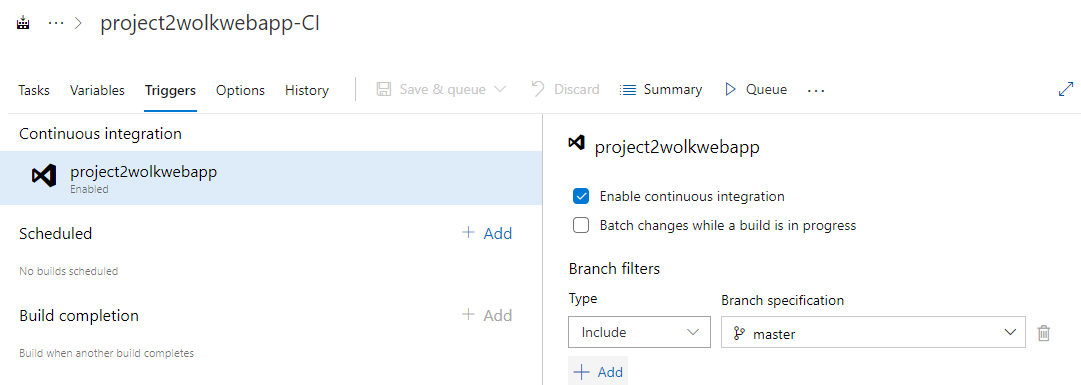


Using azure hosted agent pool for this project

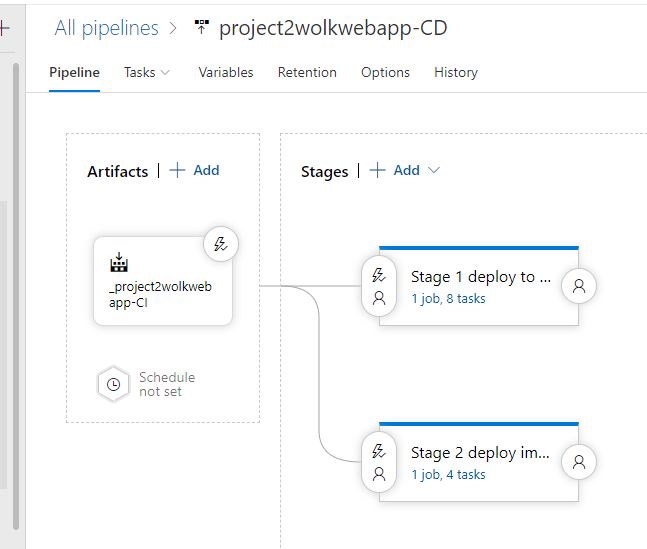


Task 4:

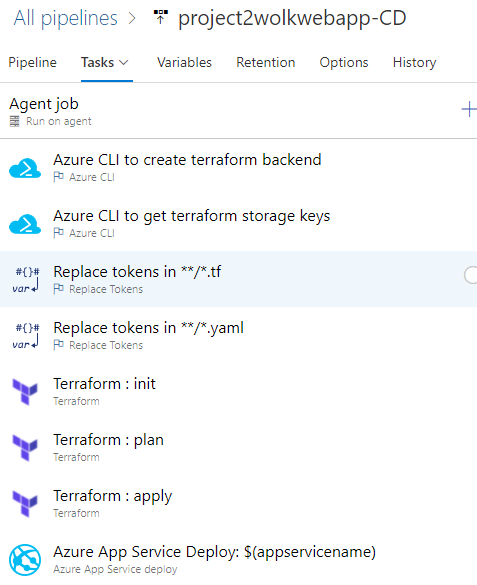
1. Enables continuous Integration between Repos and build pipeline in triggers



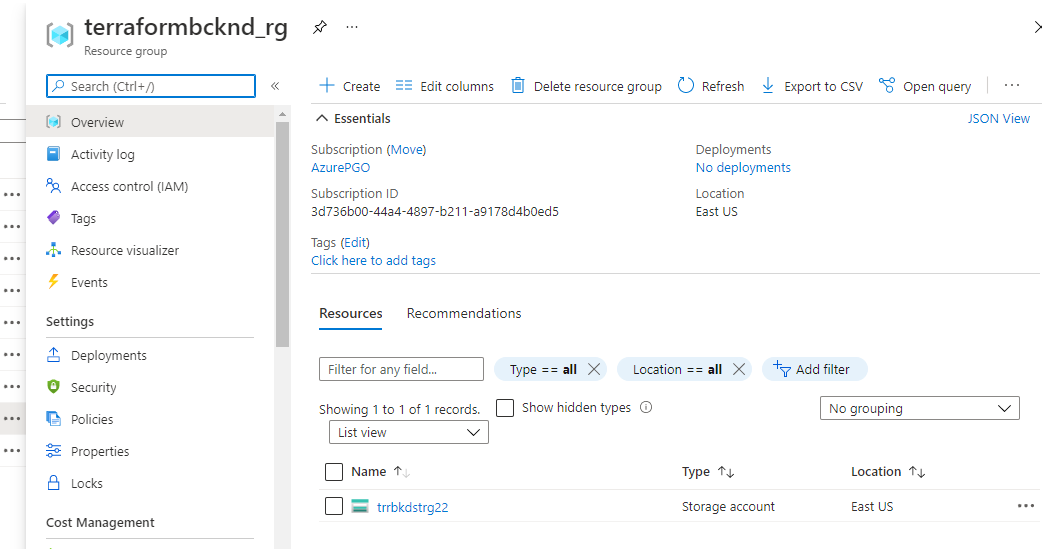
1. Creating two staged release pipeline for CD in the same project

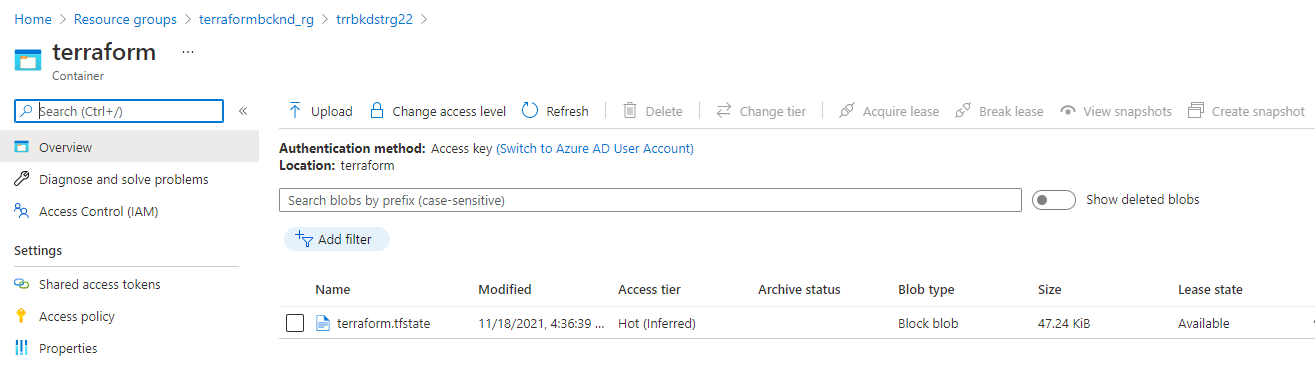


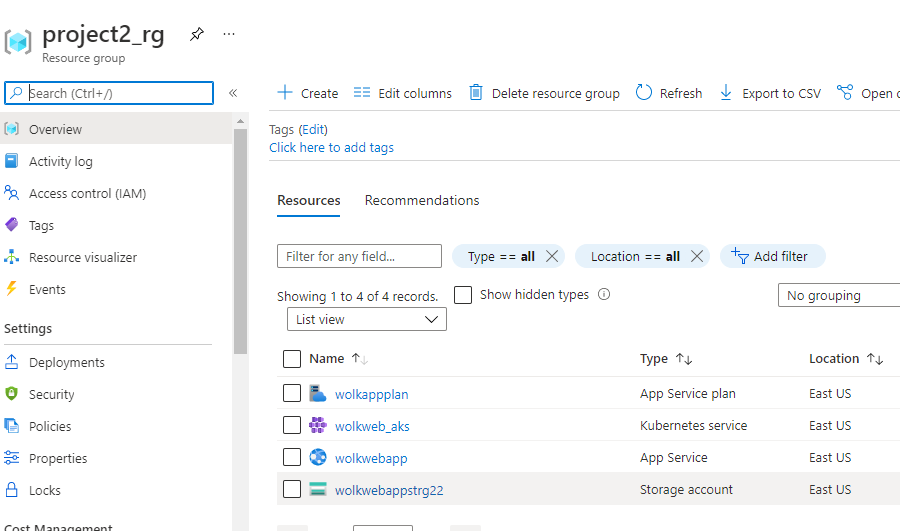
* Stage 1 deploy the application to azure web app
* Stage 2 deploy the application image to AKS

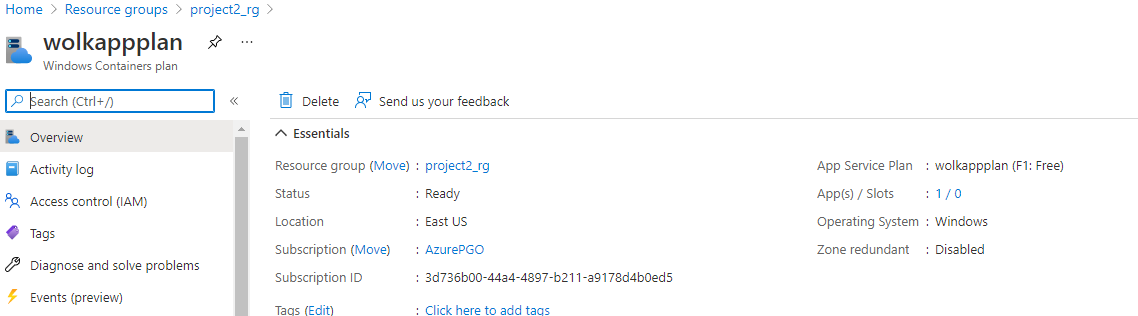


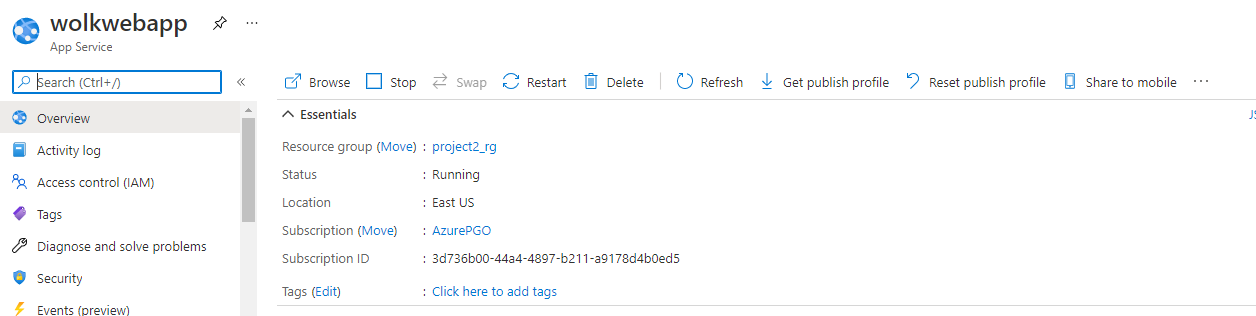
* The stage 1 does following actions:
  + Create a new Azure resource Group, Azure storage account and container with terraform.tfstate with Az CLI to serve as backend for our terraform script deployment.
  + Fetch storage service keys using AZ CLI.
  + Replace all variables in the tf file and yaml file to match the live creating resources from the variables created in pipeline.
  + Terraform init, plan and apply to deploy resources into azure.
  + Deploy application to azure webapp service

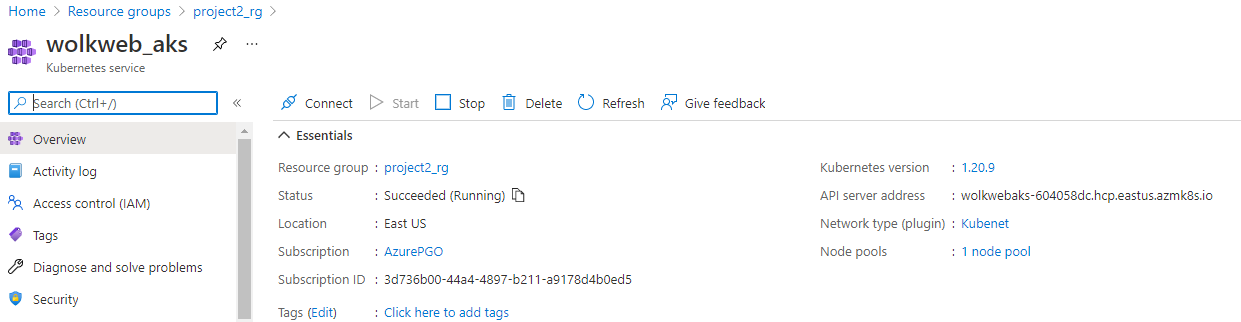


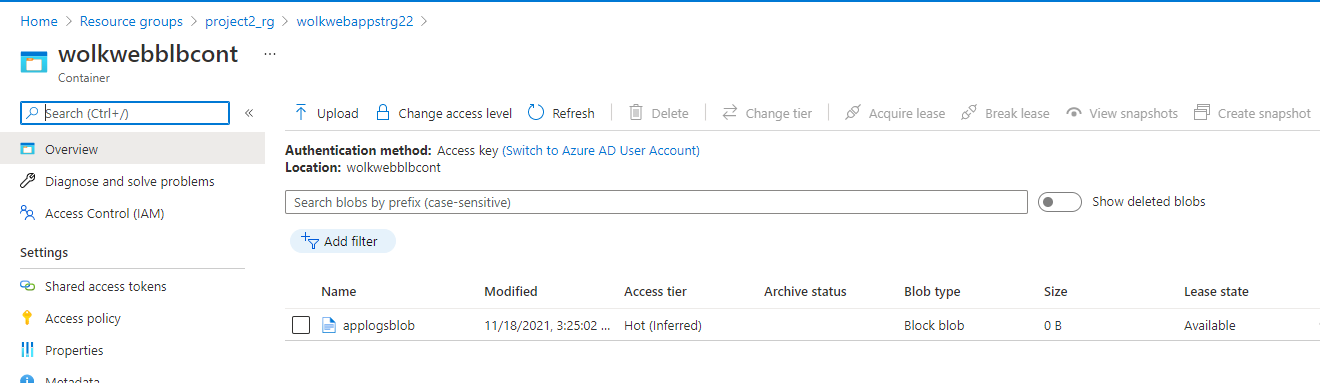




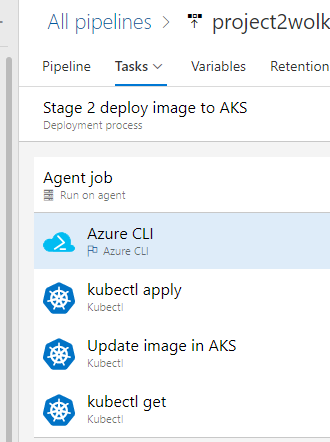








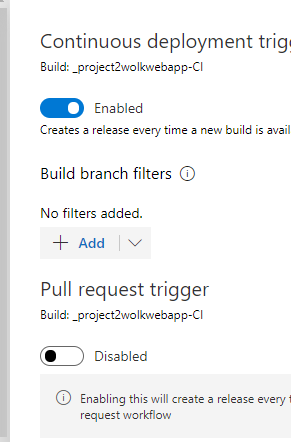
1. Stage 2 deploy below app image from ACR to AKS



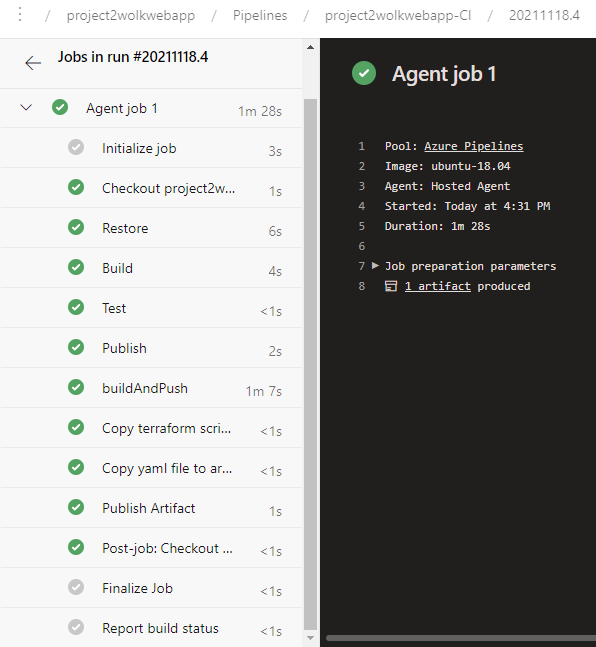
* Azure CLI will connect to the AKS cluster to run kubectl command
* Kubectl Apply will deploy yaml manifest to run a deployment and service to run the app image.

Task 5:

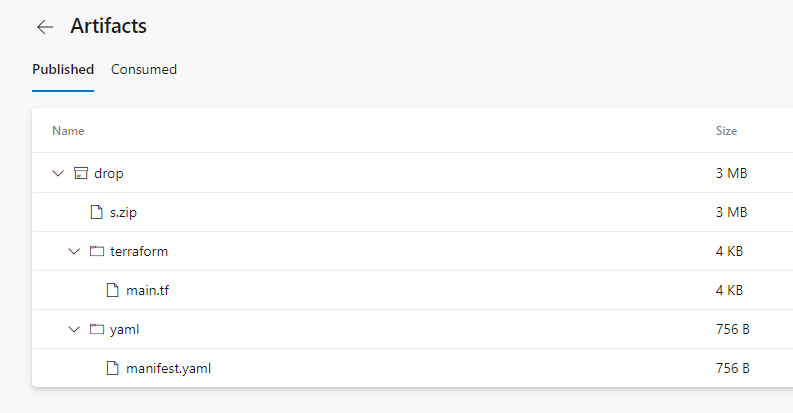
1. Trigger created in between build pipeline and release pipeline



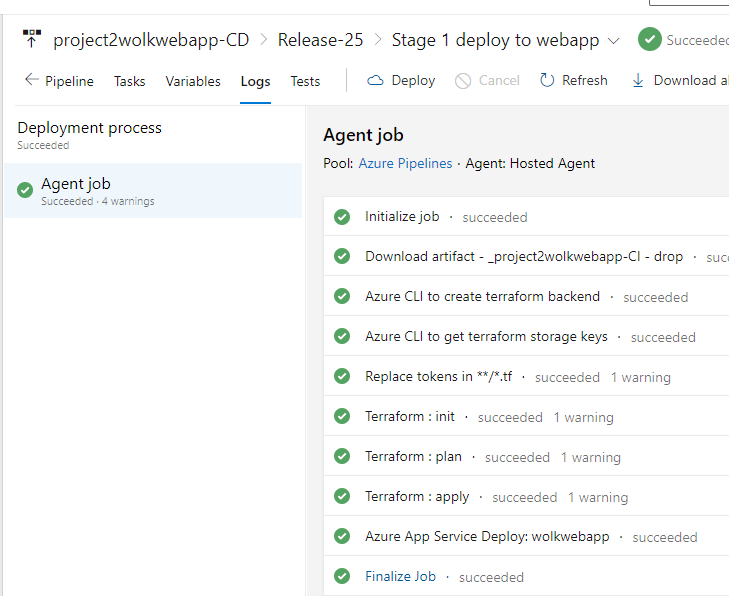
All logs of running build and release pipelines below.

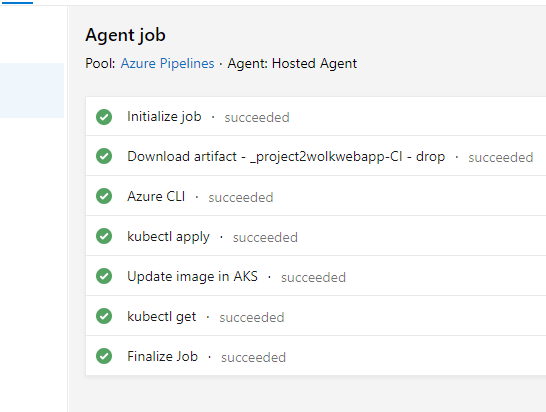


Build pipeline artifacts;



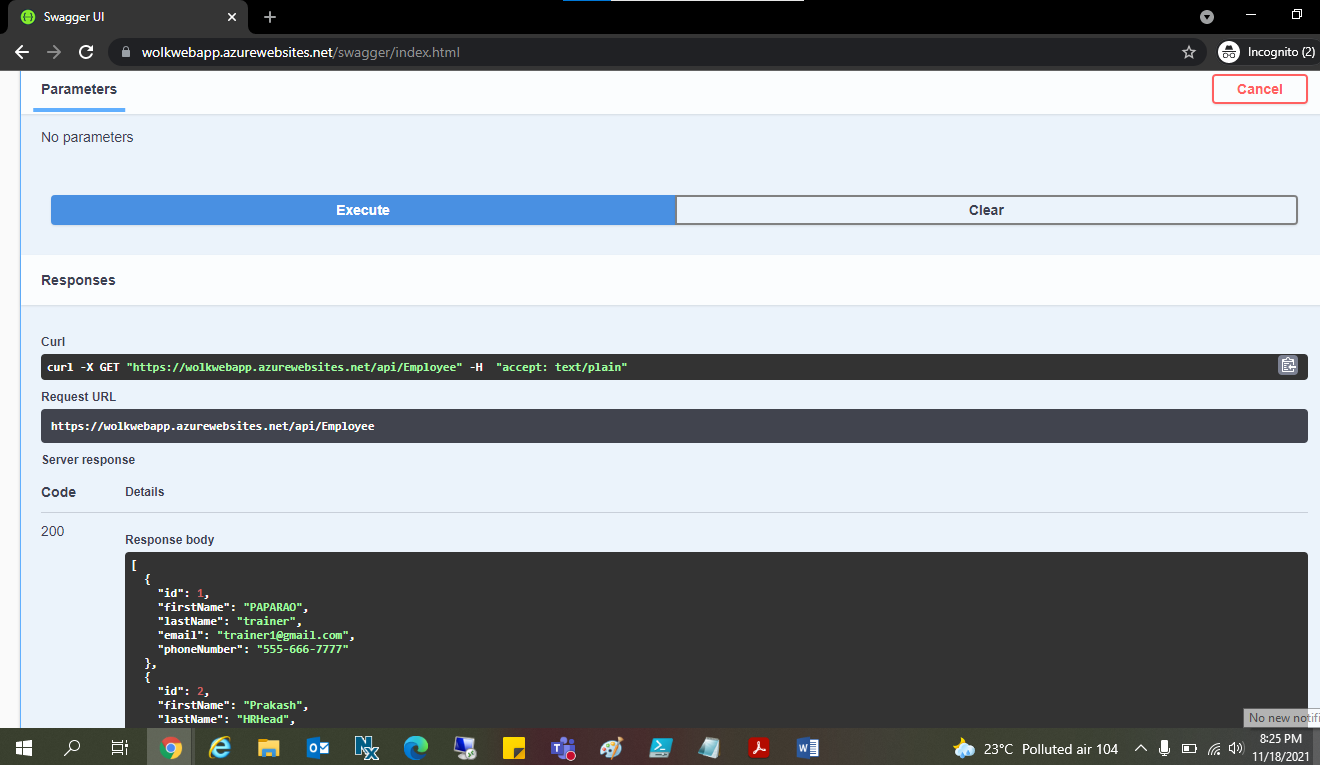
Release pipeline logs:



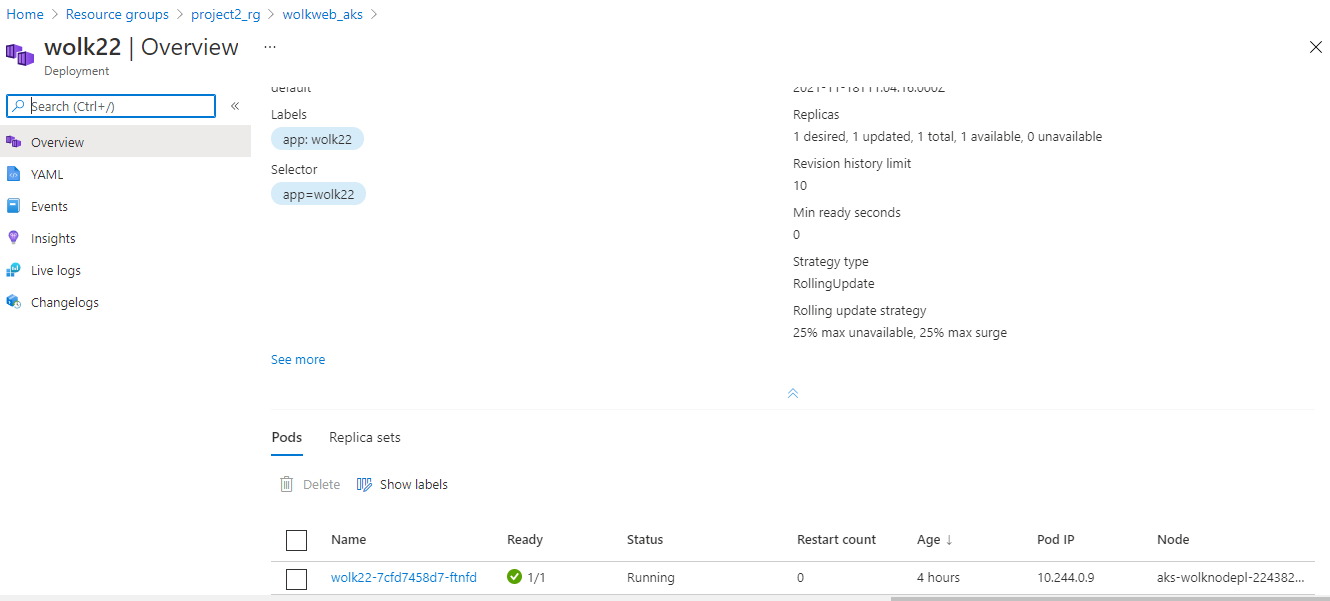


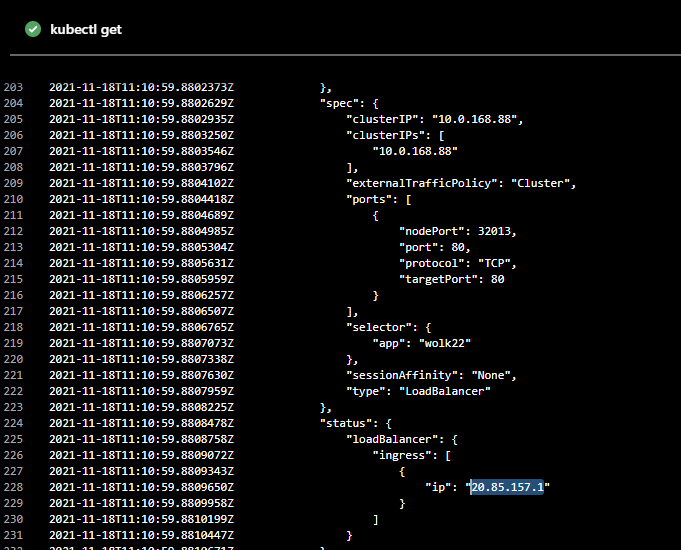
After successful running both the pipelines we have the application running in both

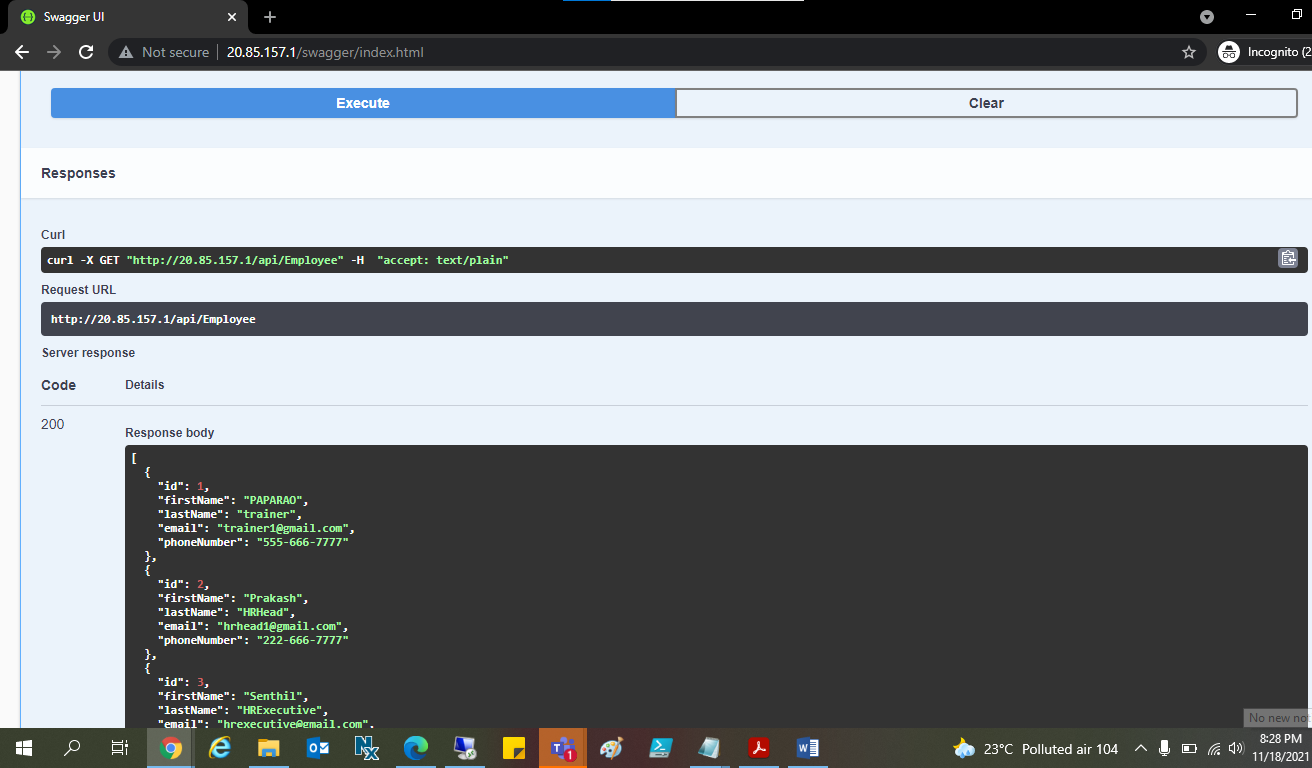
1. Azure Web App:



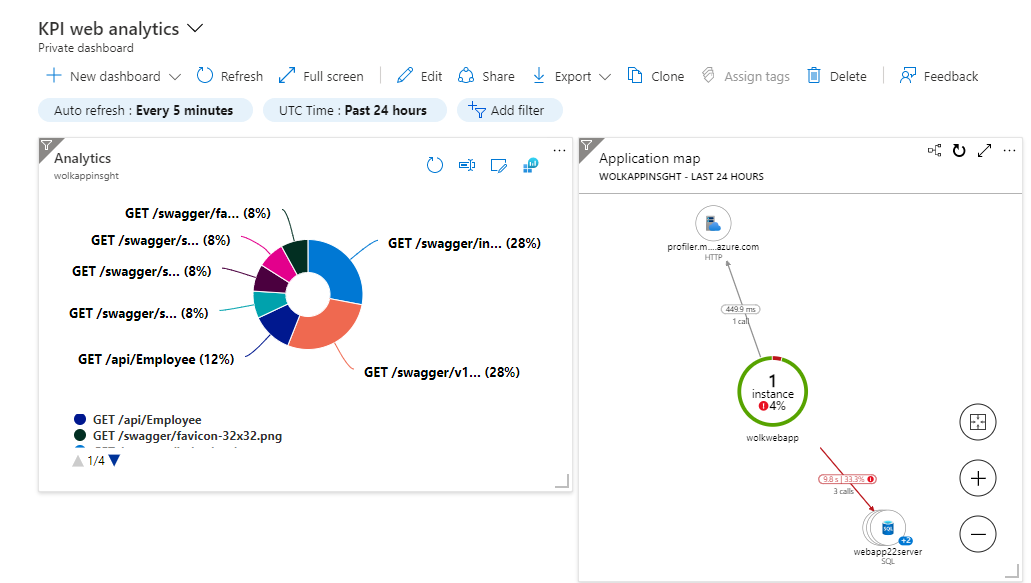
1. Application deployed as image in AKS:



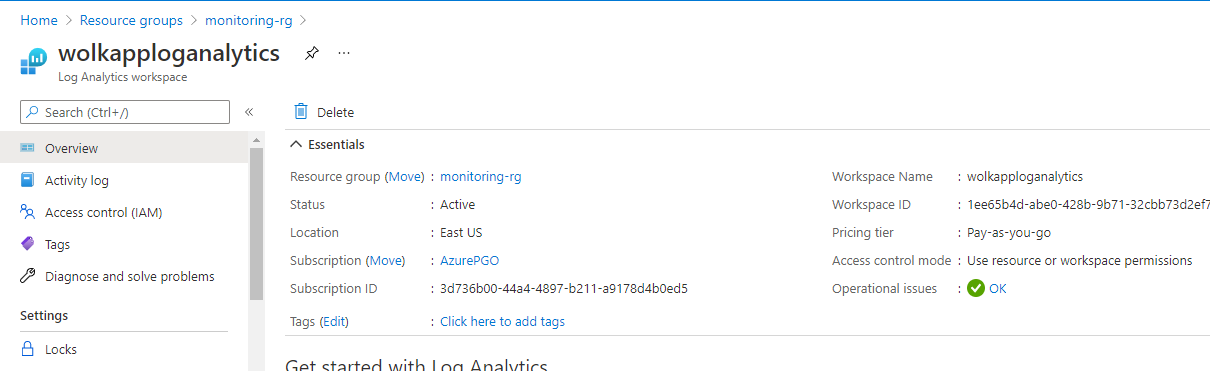




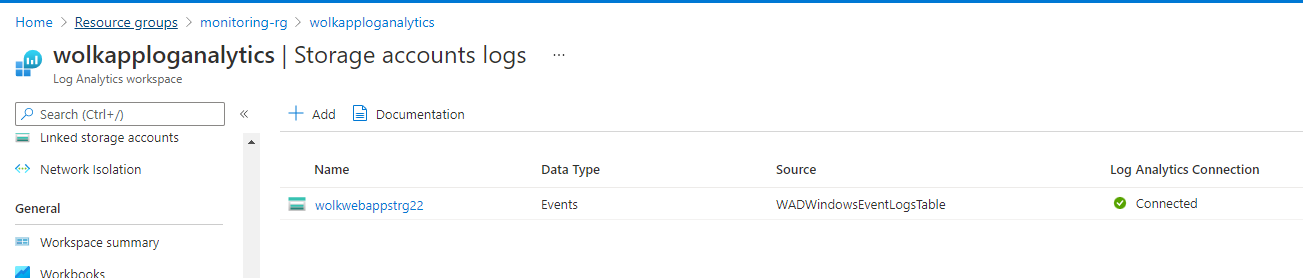
* Azure Dashboard with KPI data for monitoring



* Log analytics workspace for application metrics view:



* Logs are storages in storage account:



* Application insights configured to collect metrics for web application:

