Internship Project

Kubernetes Cluster Setup for Scalable Microservices

Name: Devang Tyagi

Description:

The architecture of microservices has become increasingly famous for building scalable, resilient, and manageable applications. However, successful management of microservices at scale requires orchestration and infrastructure management capability. Kubernetes is one of the best-known open-source container orchestration platforms to deploy, scale, and manage containerized applications. This project consists of establishing a Kubernetes cluster, in which the scalable microservices will run and where Kubernetes can demonstrate its superior features like auto-scaling, service discovery, and load balancing. Students will learn by field experience in deploying microservices, configuring Kubernetes clusters, and assigning resources for optimizing infrastructure for scalability, reliability, and fault tolerance.

Objective:

1. Learn about Microservice Architecture: Understand the fundamentals of microservices, concentrating on how the services are constructed, deployed, and communicated with one another.

2. Introduction to Kubernetes Basics: Become well versed in the architecture of Kubernetes, and its huge components: clusters, pods, services, deployments, and scaling techniques.

3. Build Scalable Infrastructure: Design such Kubernetes architecture that enables the dynamic scaling of microservices depending on demand, while not compromising on high availability and fault tolerance.

4. Establish CI/CD Pipelines: Create a CI/CD pipeline to automate microservice deployments on the Kubernetes cluster.

5. Monitor and Maintain the Cluster: Install monitoring and logging tools that will be utilized to track the health and performance of microservices in a Kubernetes cluster.

6. Cost and Performance Optimization: You'll analyze and optimize resource usage in the Kubernetes cluster to strike the right balance between performance and cost.

Technologies used:

1. Azure (Azure Cloud Portal)
   - Kubernetes Services (creating Kubernetes cluster).
   - Container Registry
   - Prometheus
   - Grafana
2. Azure DevOps (Azure DevOps Portal)
   - Classic Editor (for building and pushing docker image to desired container registry)
   - Pipelines (one for building the project and another for deploying the project.)
3. Azure CLI (in local host)
4. GitHub link (to import files to Azure DevOps)

AKS: Azure Kubernetes Service, a container orchestration service that is a managed Kubernetes service and allows deploying, managing, and scaling containerized applications. We will set up AKS in Azure Cloud as part of the lab exercise.

Container Registry: Utilities and considered being for storage, manageability, and securing deployment services of the container image empowered by Docker.

Prometheus: Open-source instrumentation tool built for capturing and querying metrics in container environments.

Grafana: Visualization platform, an interactive dashboard tool for monitoring and analyzing different metrics.

Azure DevOps: An online service supported by Microsoft with features for source control, CI/CD, and project management.

Classic Editor (for building and pushing Docker images to desired container registry): A GUI-based pipeline editor in Azure DevOps that accepts pipeline definition that, in turn, automates Docker builds and pushes.

Pipelines: CI/CD automation tools for compiling, testing, and deploying applications.

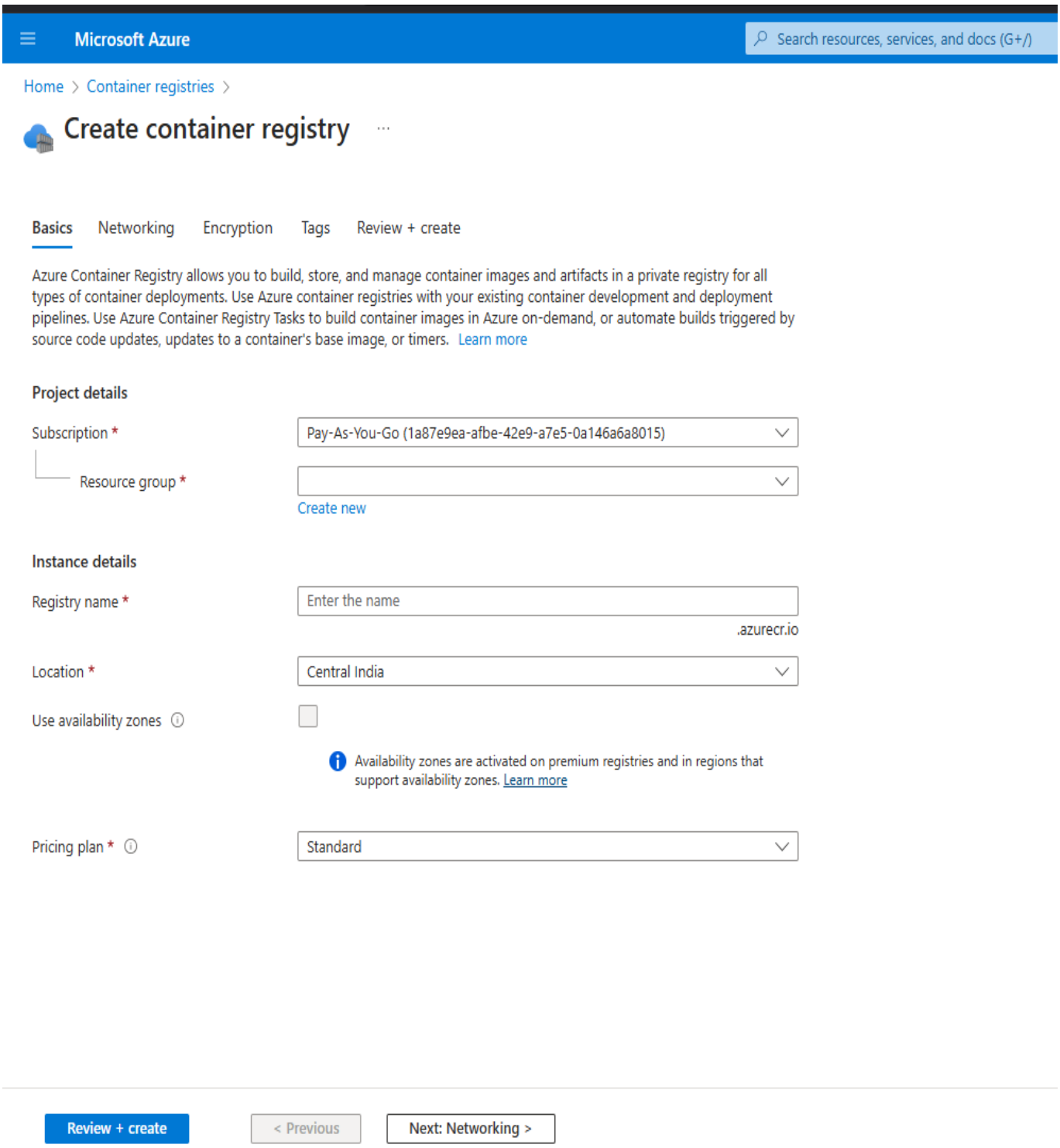
Azure CLI: A command-line tool meant for Azure resource management and going forth automates tasks for backend cloud service users.

GitHub link: A repository URL with which Azure DevOps can fetch source code for builds and deployments.

Create a Registry and Kubernetes Cluster in Azure Portal

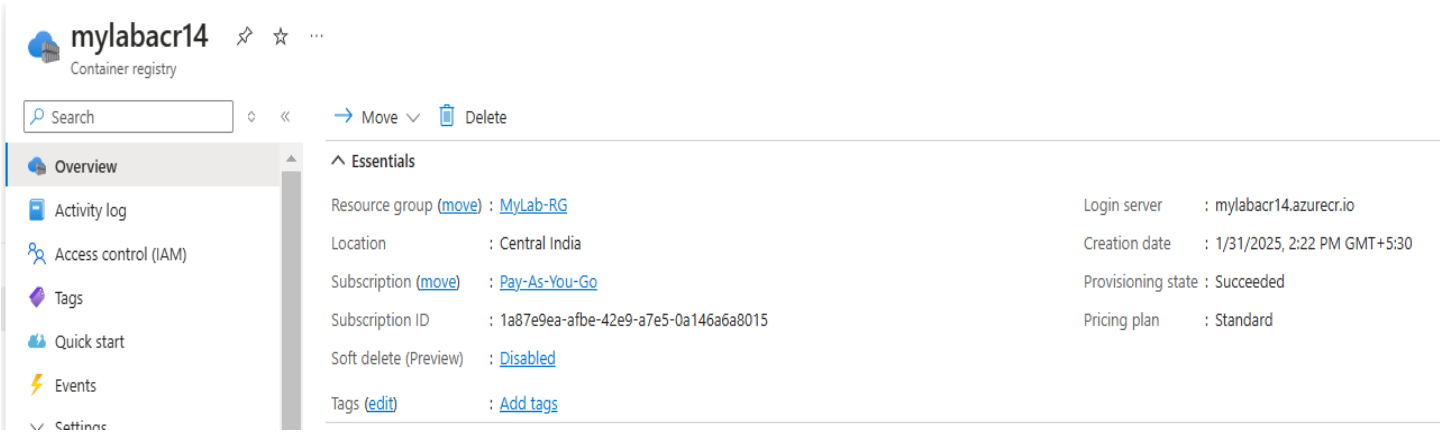
Create a Container registry:

Create a container registry where the docker image will be pushed.



Create a Container registry:

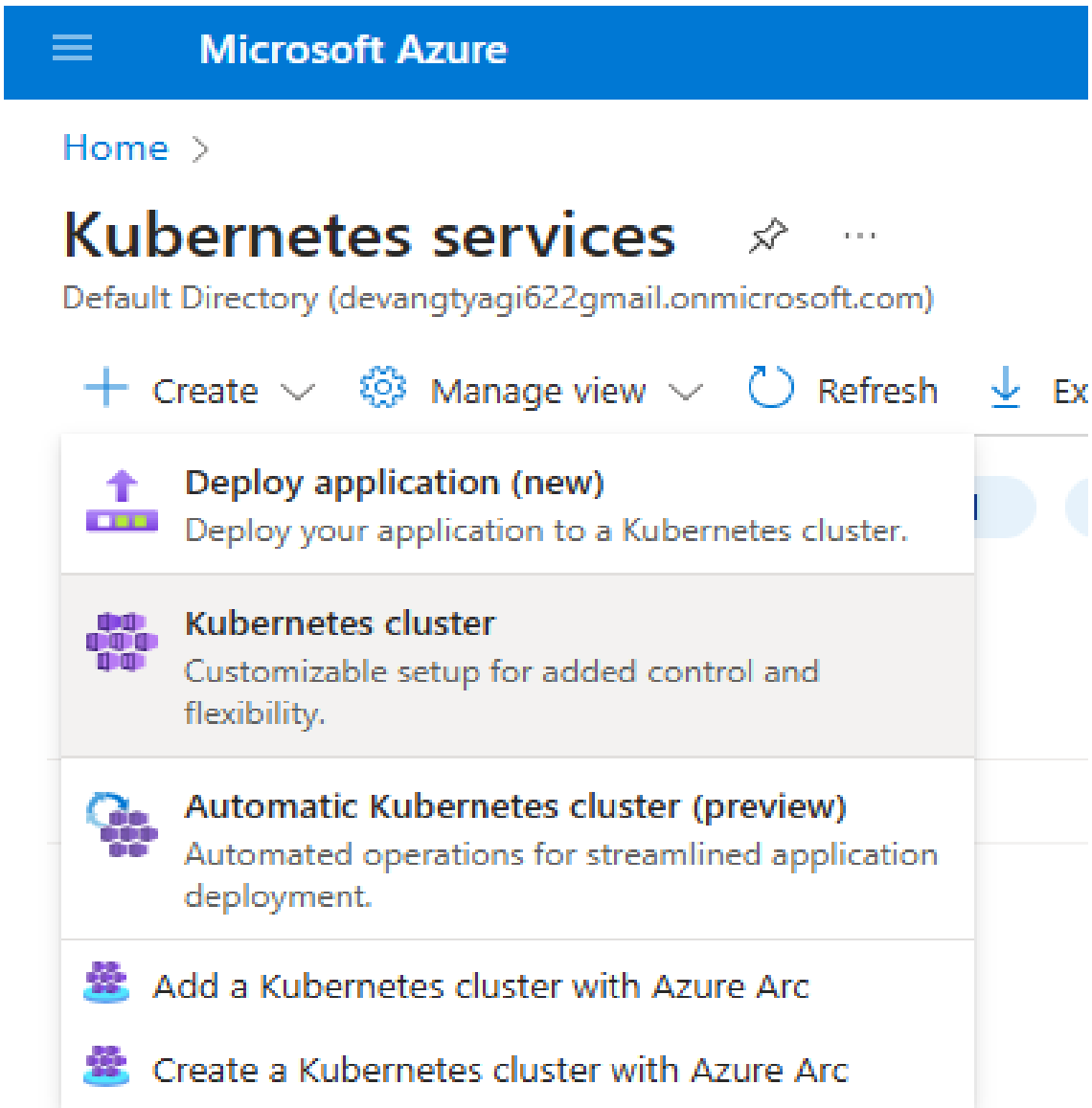
Once the registry is created successfully the ‘Provisioning State’ will show the state as ‘Succeeded’.



Create a Kubernetes Cluster:

1. The latest view is a bit different for creating a Kubernetes cluster so to create a cluster choose the desired option.

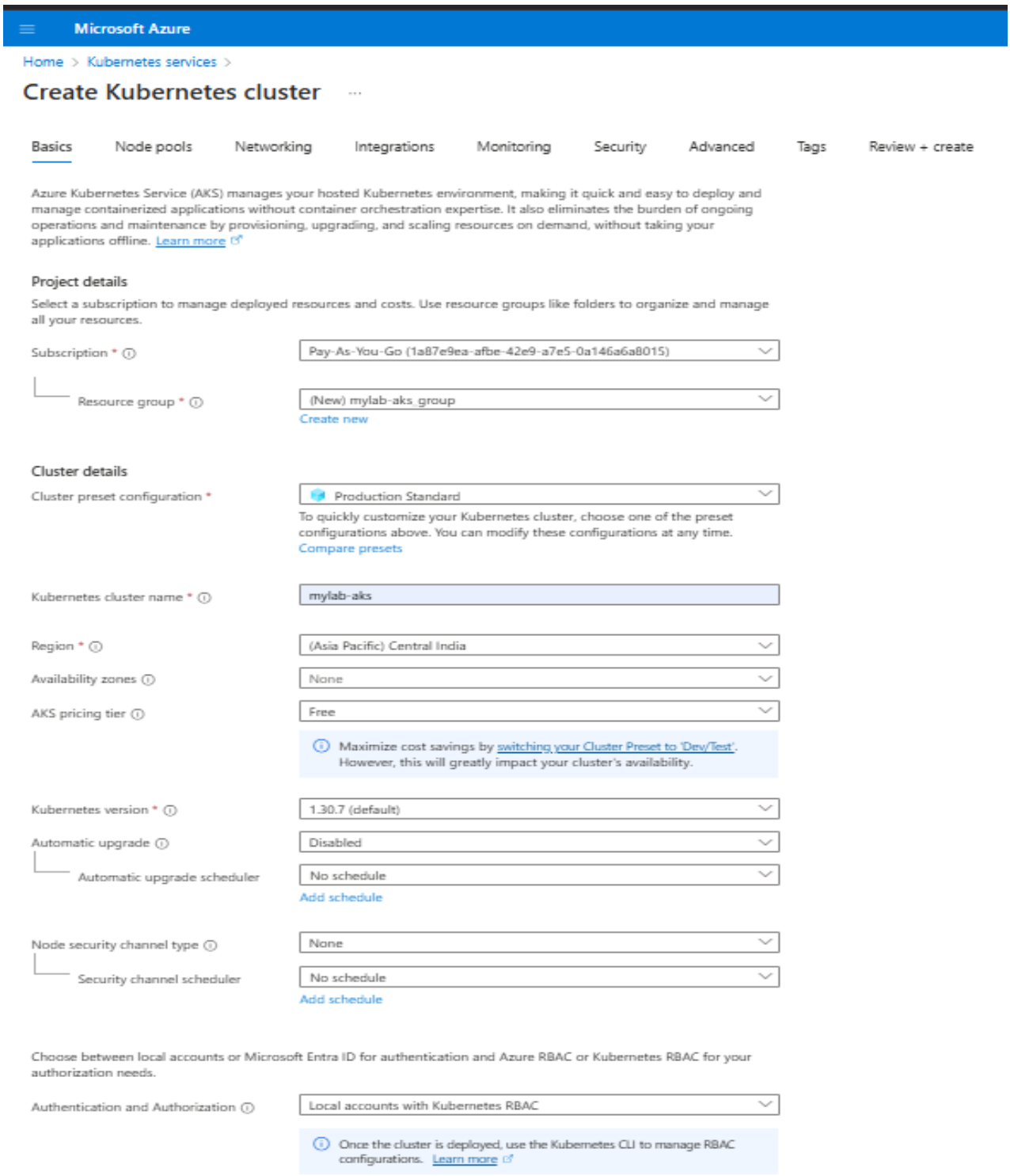
2. For this experiment, the shadowed option was used.



Create a Kubernetes Cluster:

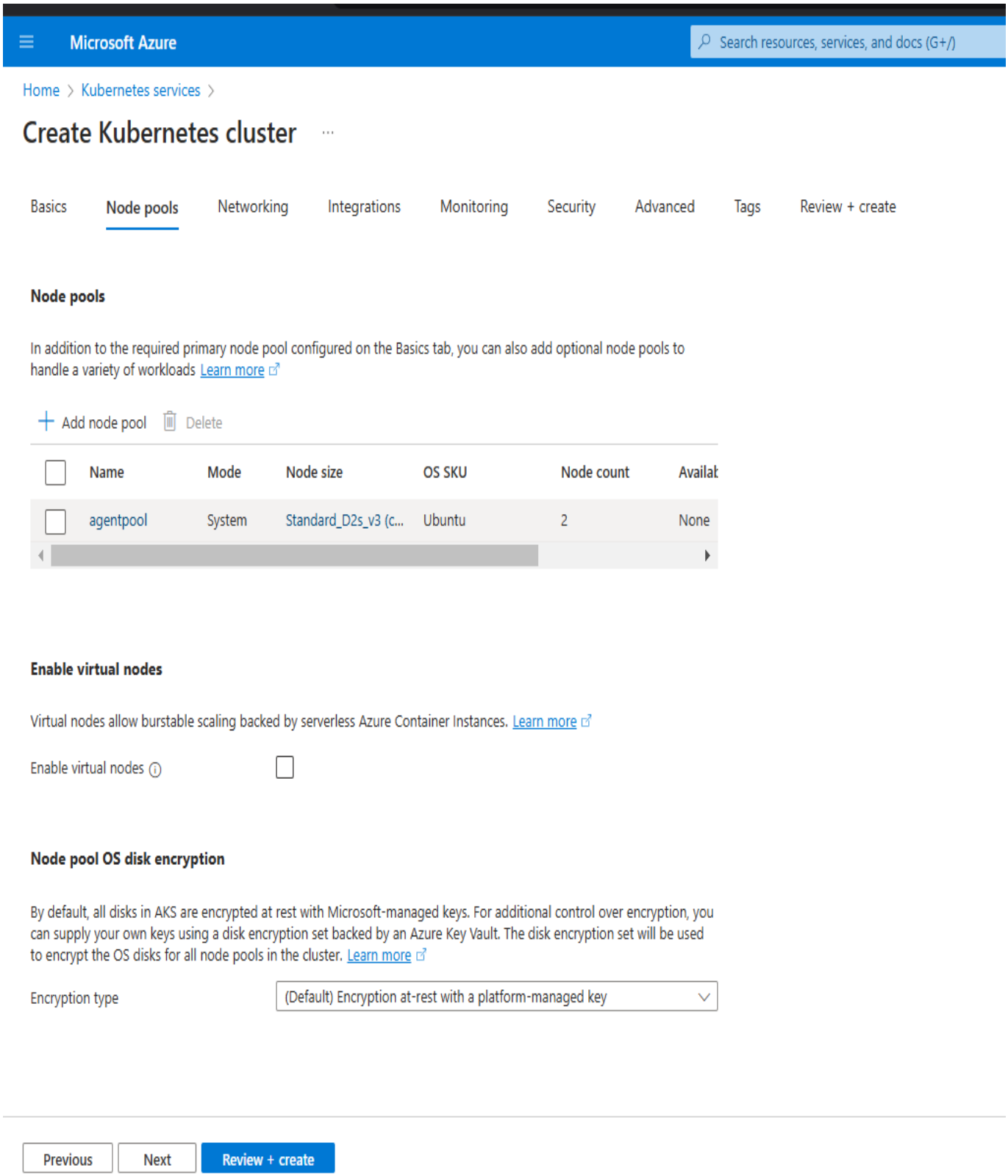
The options were selected for saving the cost but for desired option change only the following options:

1. Resource Group
2. Cluster preset configuration
3. Kubernetes cluster name



Create a Kubernetes Cluster:

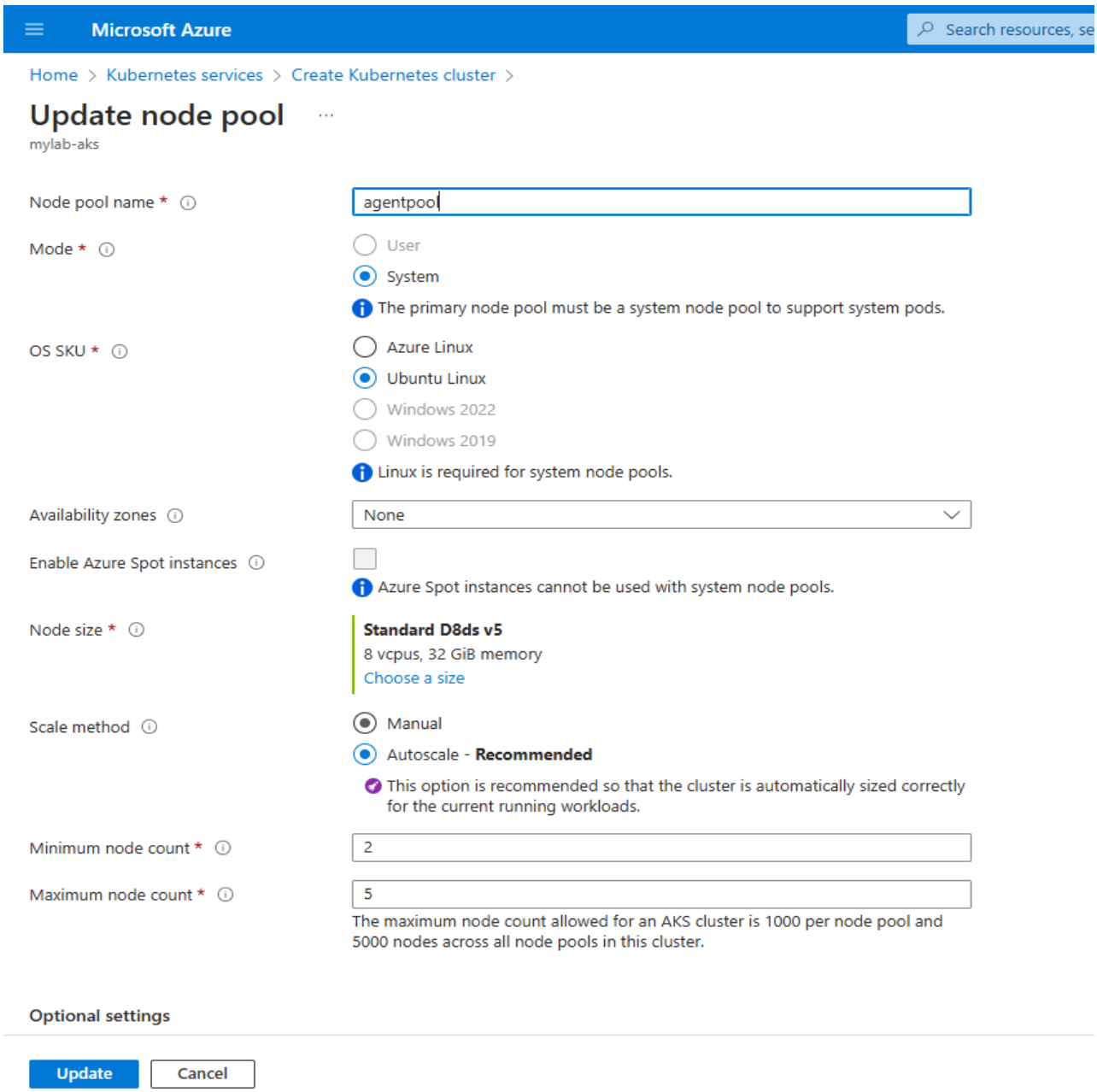
In the next section for node pool click on the node pool name and make the required changes



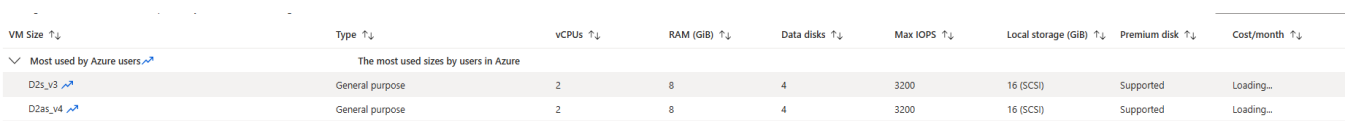
Create a Kubernetes Cluster:

For this experiment the desired changes are as follows:

1. The scaling method was ‘Manual’
2. The VM size was D2s_v3.



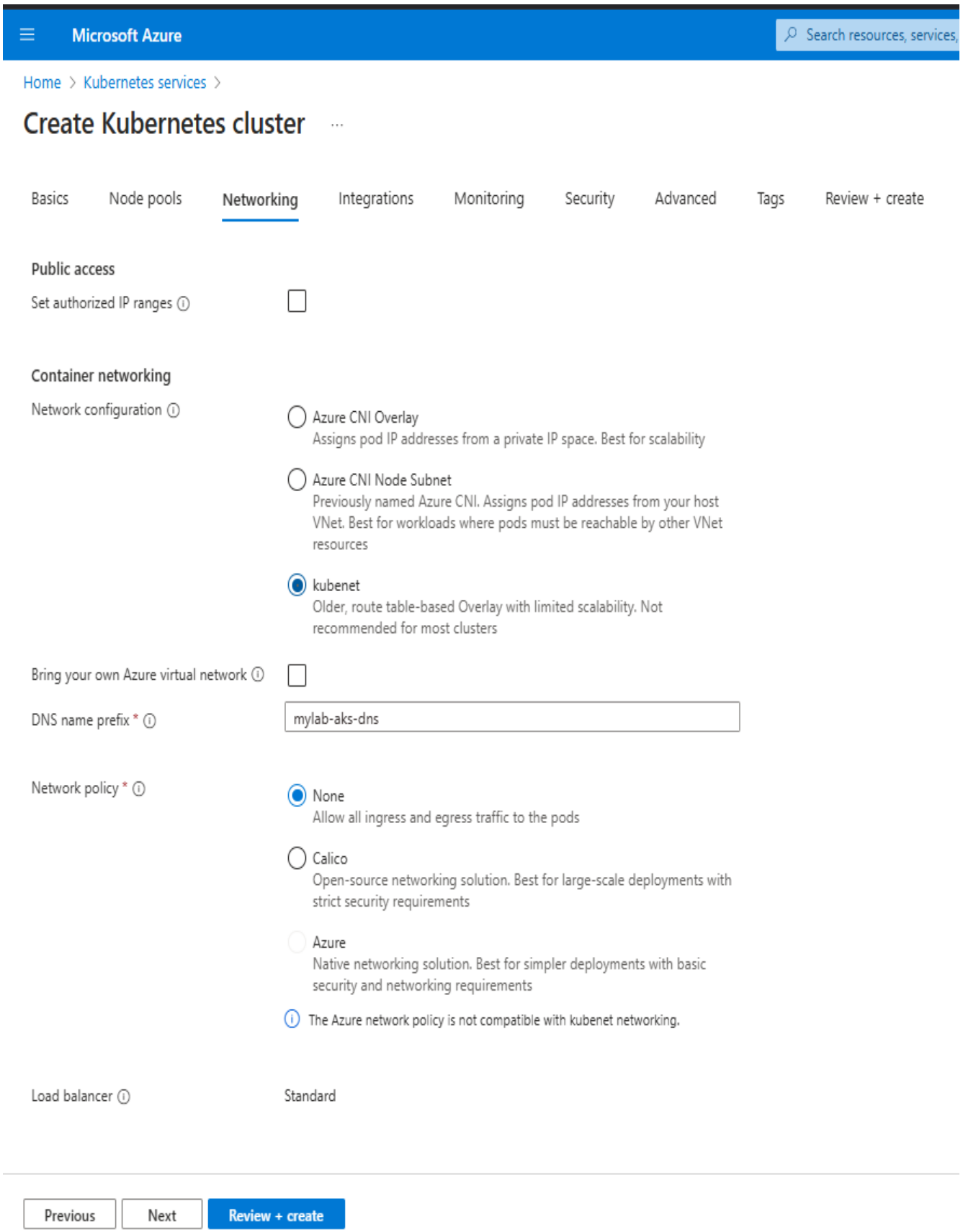
Node Size:



VM Size	Type	vCPUs	RAM (GiB)	Data disks	Max IOPS	Local storage (GiB)	Premium disk	Cost/month
Most used by Azure users	The most used sizes by users in Azure							
D2s_v3	General purpose	2	8	4	3200	16 (SCSI)	Supported	Loading...
D2as_v4	General purpose	2	8	4	3200	16 (SCSI)	Supported	Loading...

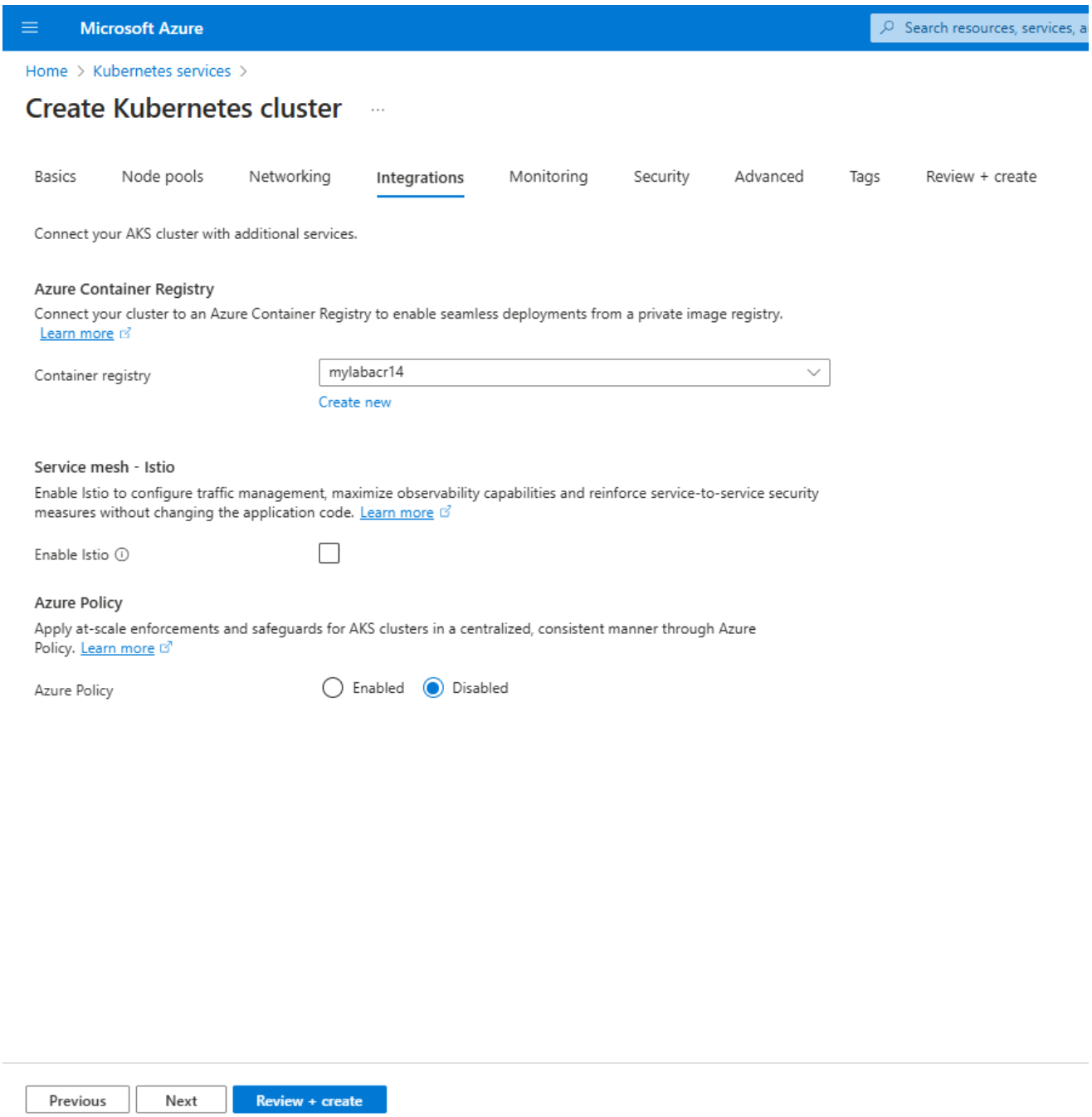
Create a Kubernetes Cluster:

In the container network select Kubenet



Create a Kubernetes Cluster:

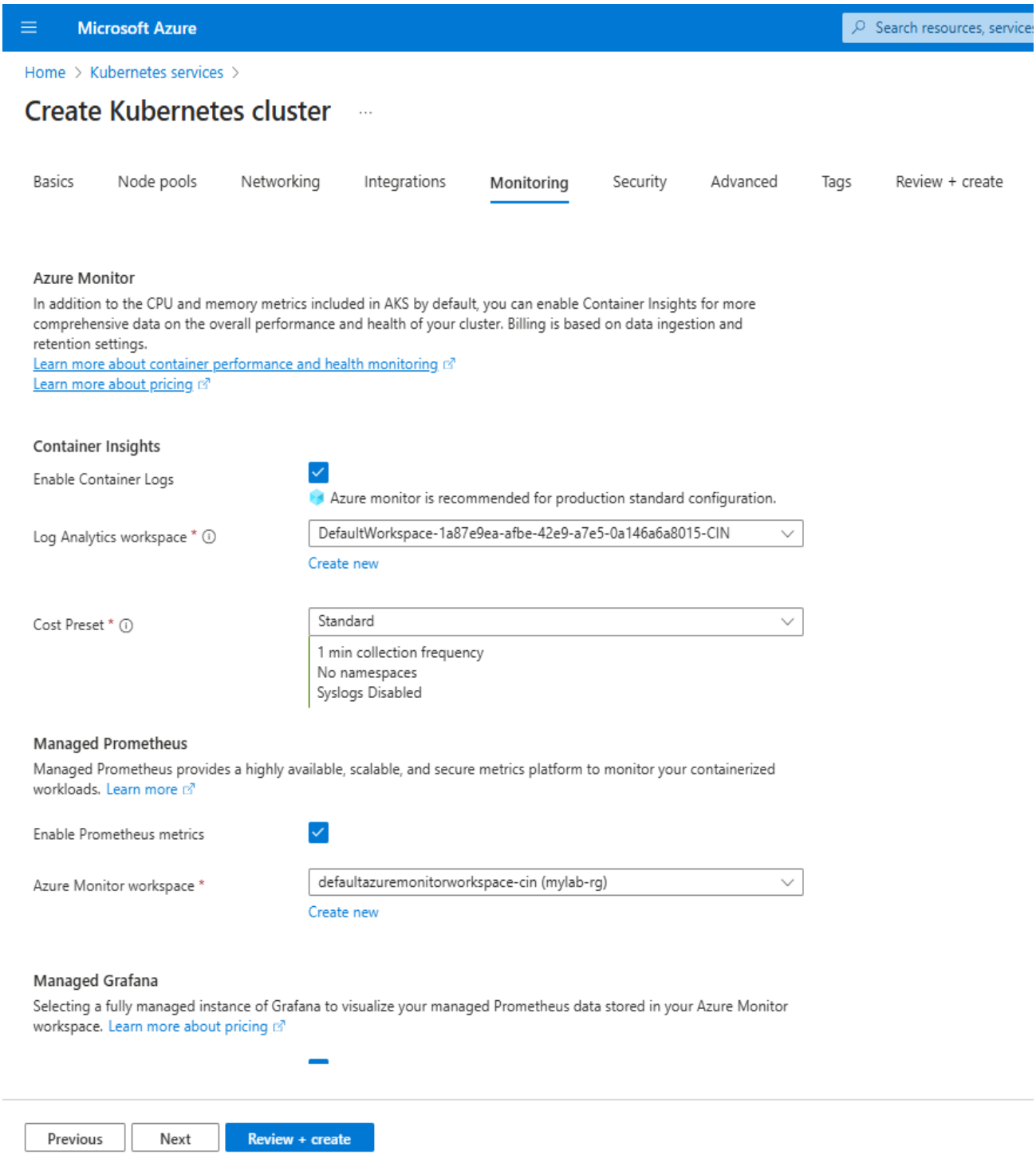
In Azure container registry select the required container registry



Create a Kubernetes Cluster:

Under Monitoring select the following options:

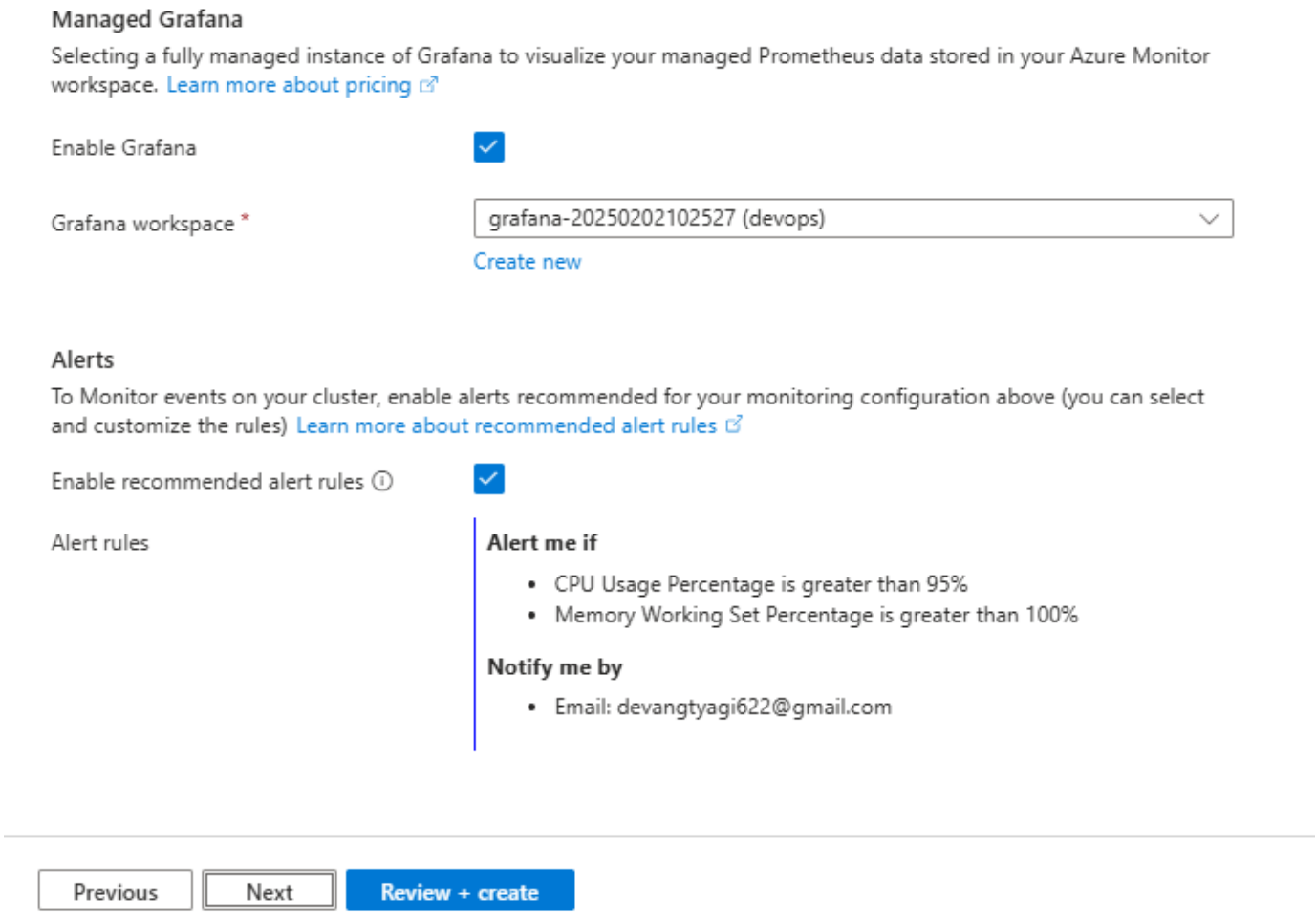
1. Alerts
2. Prometheus
3. Grafana



Create a Kubernetes Cluster:

Under Monitoring select the following options:

1. Alerts
2. Prometheus
3. Grafana

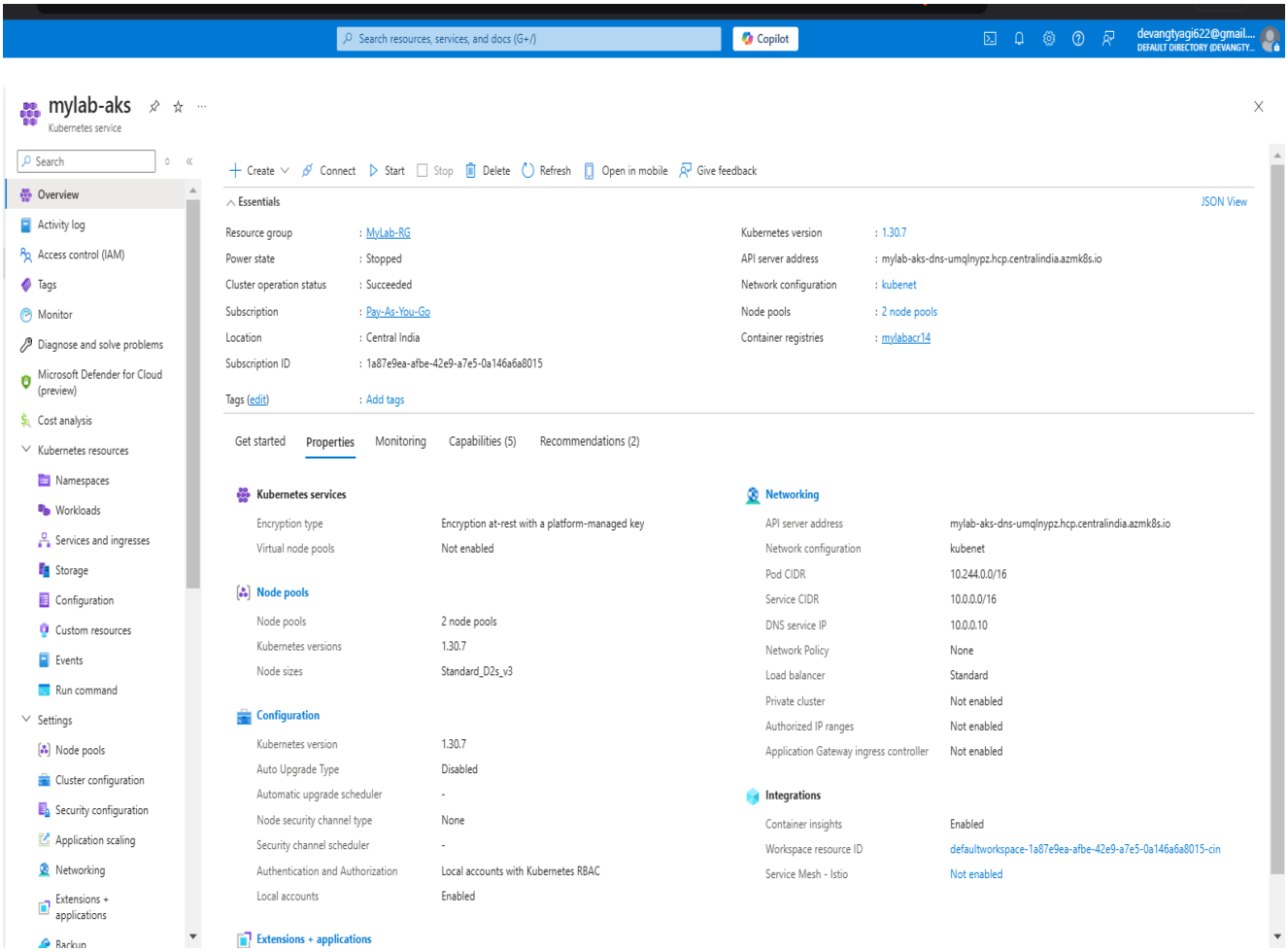


Create a Kubernetes Cluster:

After creating the Kubernetes cluster the operation status will show ‘Succeeded’

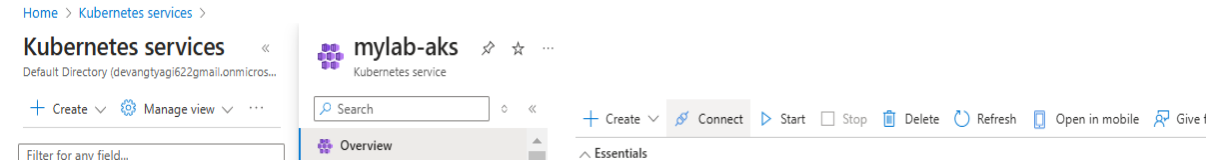
Note:

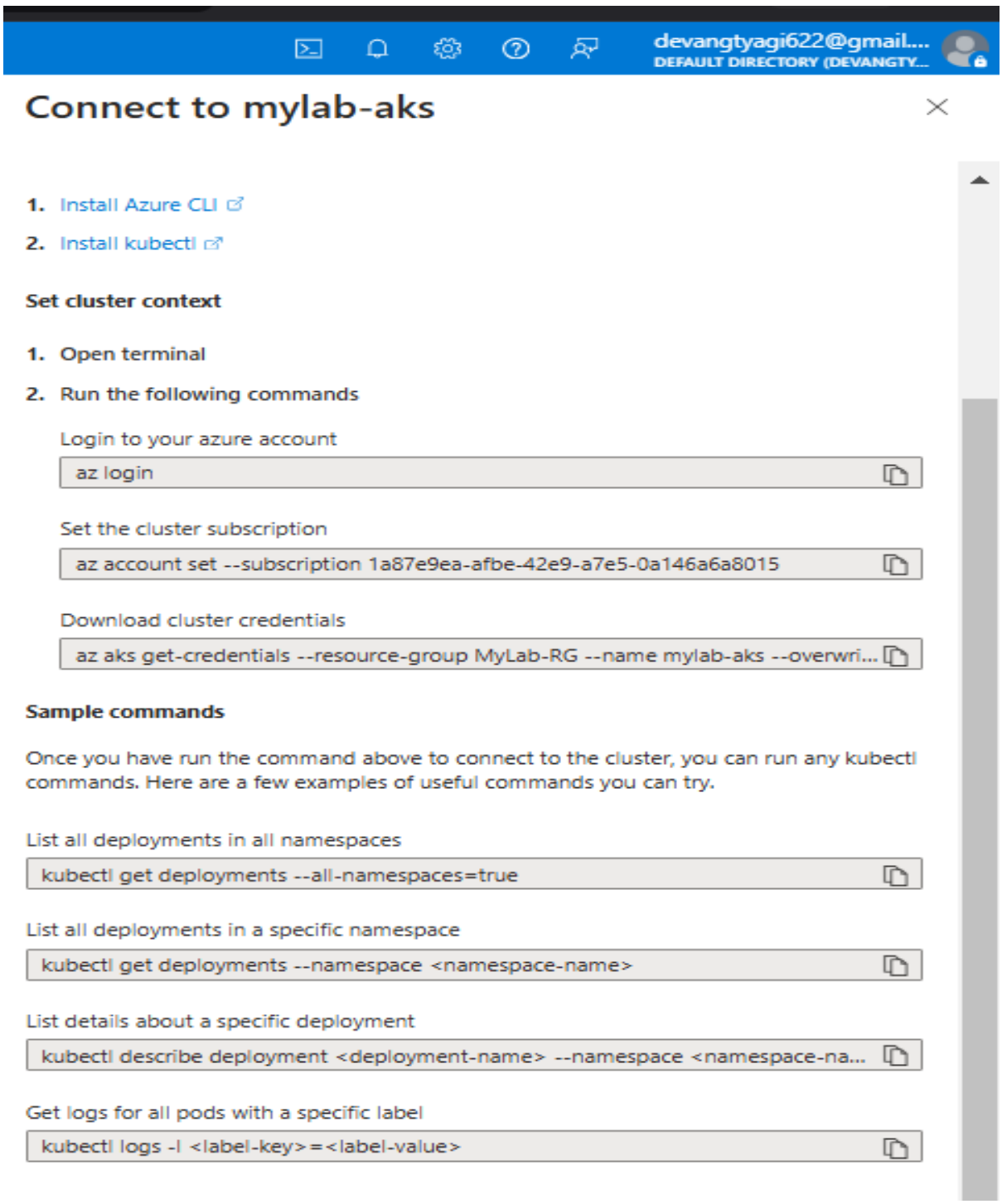
1. The screenshot was taken after the experiment was complete and to prevent further increase in cost the cluster was stopped.
2. It is always recommended to stop the cluster when it is used to reduce the cost, as this action reduces the need to delete the cluster and saves money.



Create a Kubernetes Cluster:

Here comes the use of the CLI. It is optional to use Cloud Shell or the Azure CLI but not in every region the Cloud Shell is available so Azure CLI can be used and all the necessary commands can be accessed using the ‘connect’ option as shown in the picture below.



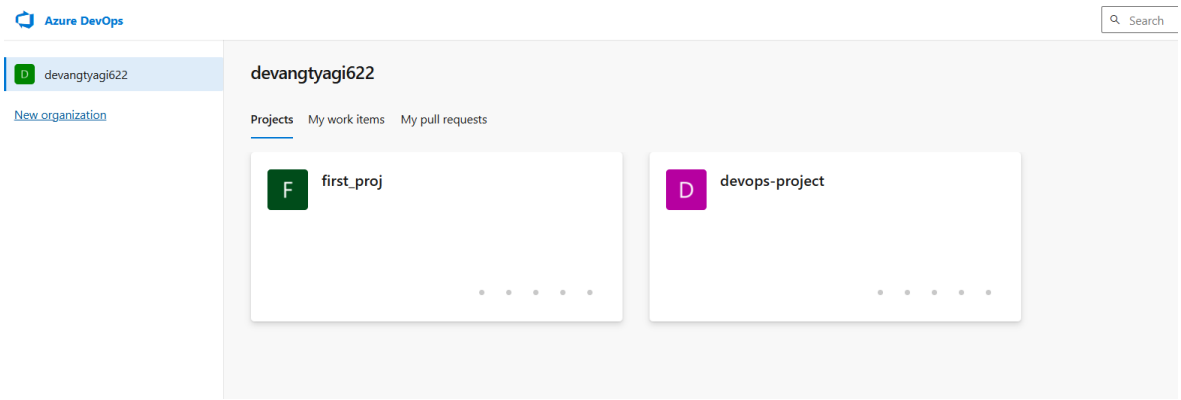


Create a CI/CD pipeline in Azure DevOps

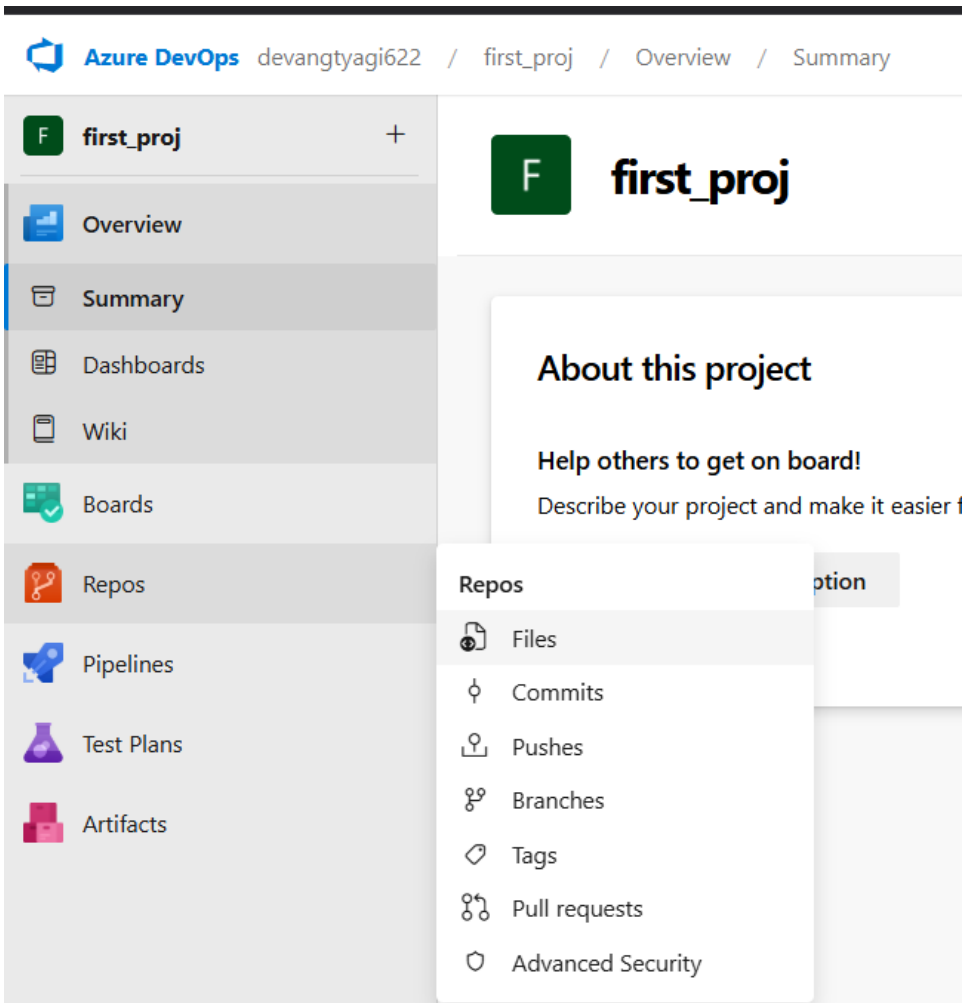
Create a Repo:

When you open the Azure DevOps Portal, sign-in using the email and password used in the Azure Portal sign-in.

1. Create the project for this experiment ‘first_proj’ was used

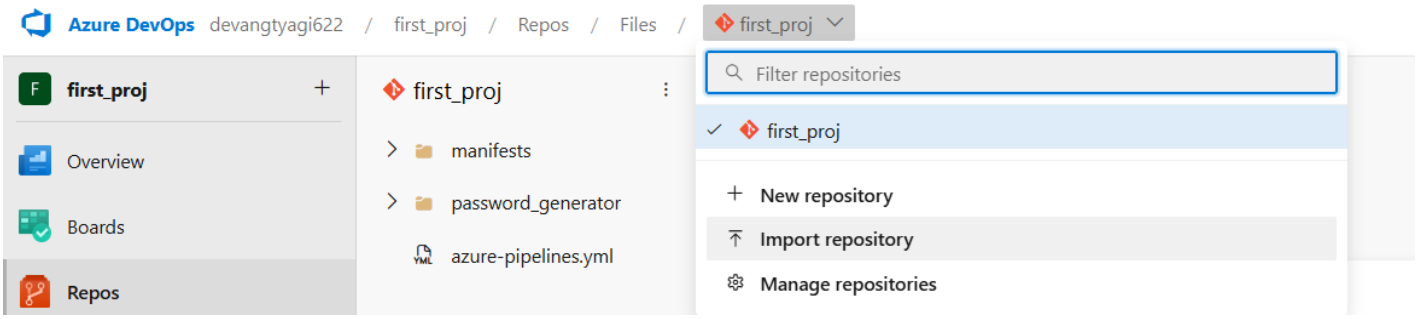


2. Then click on the project name and navigate to ‘repos’ to ‘Files’

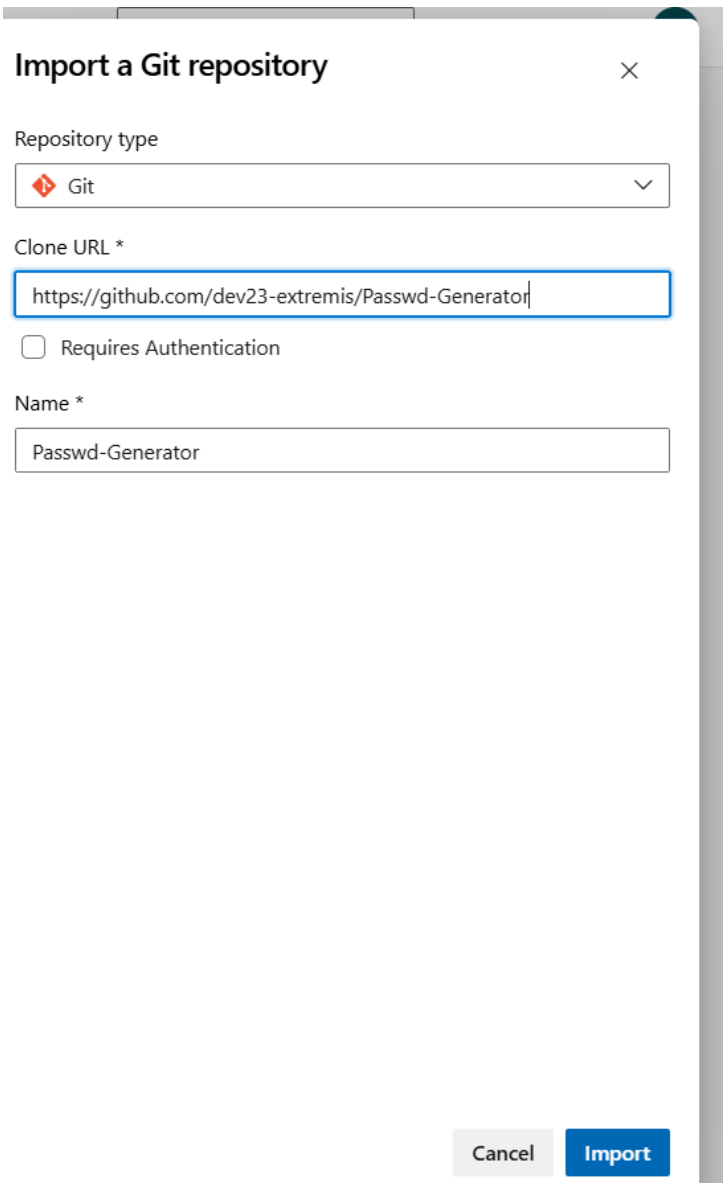


Create a Repo:

3. Then click on the project name and go to ‘import registry’

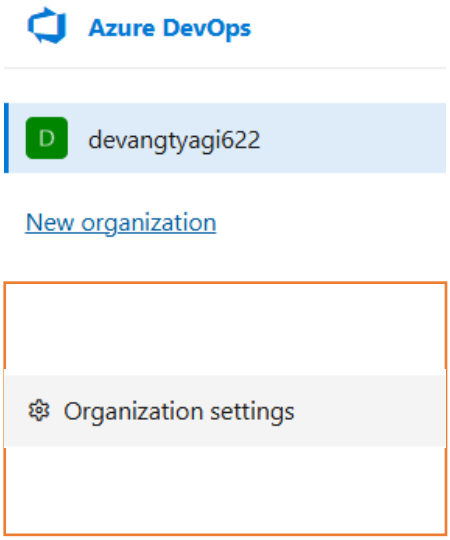


4. Their ‘Repository type’ is ‘Git’ by default.
5. Copy and paste the URL of the GitHub where the source is published. Then click Import.

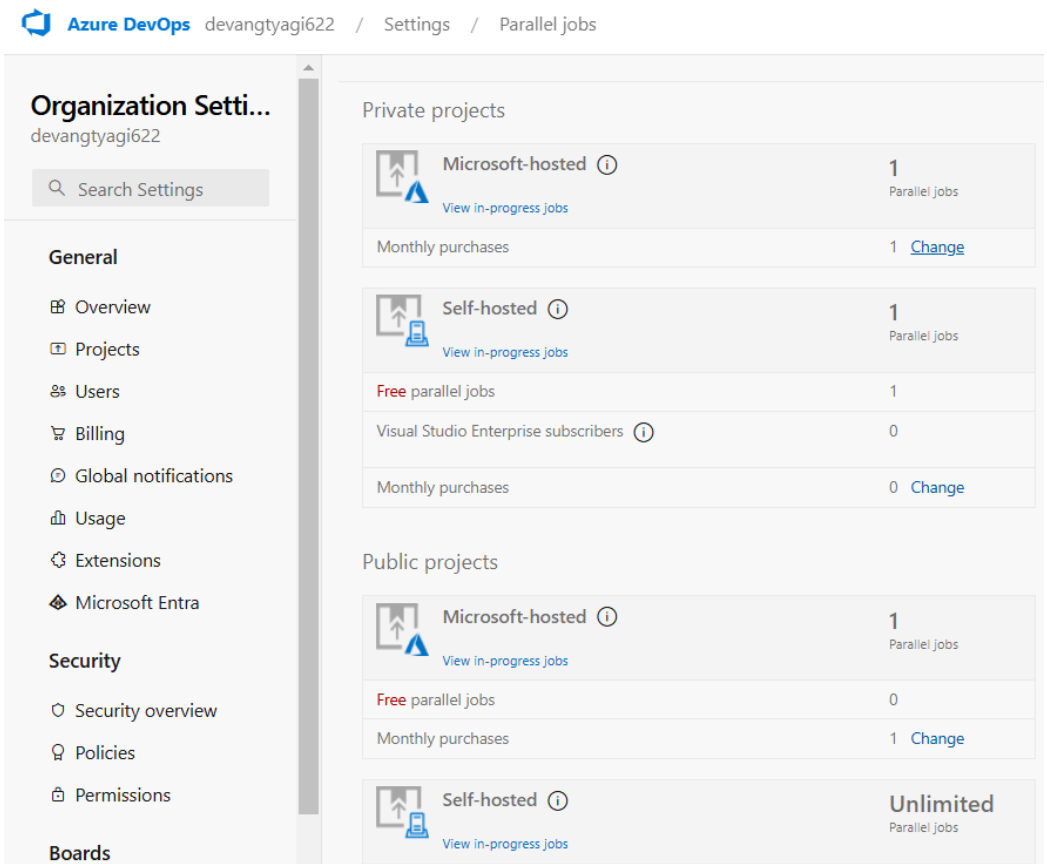


Create a Repo:

6. On the first landing page of Azure DevOps go to ‘Organization settings’.

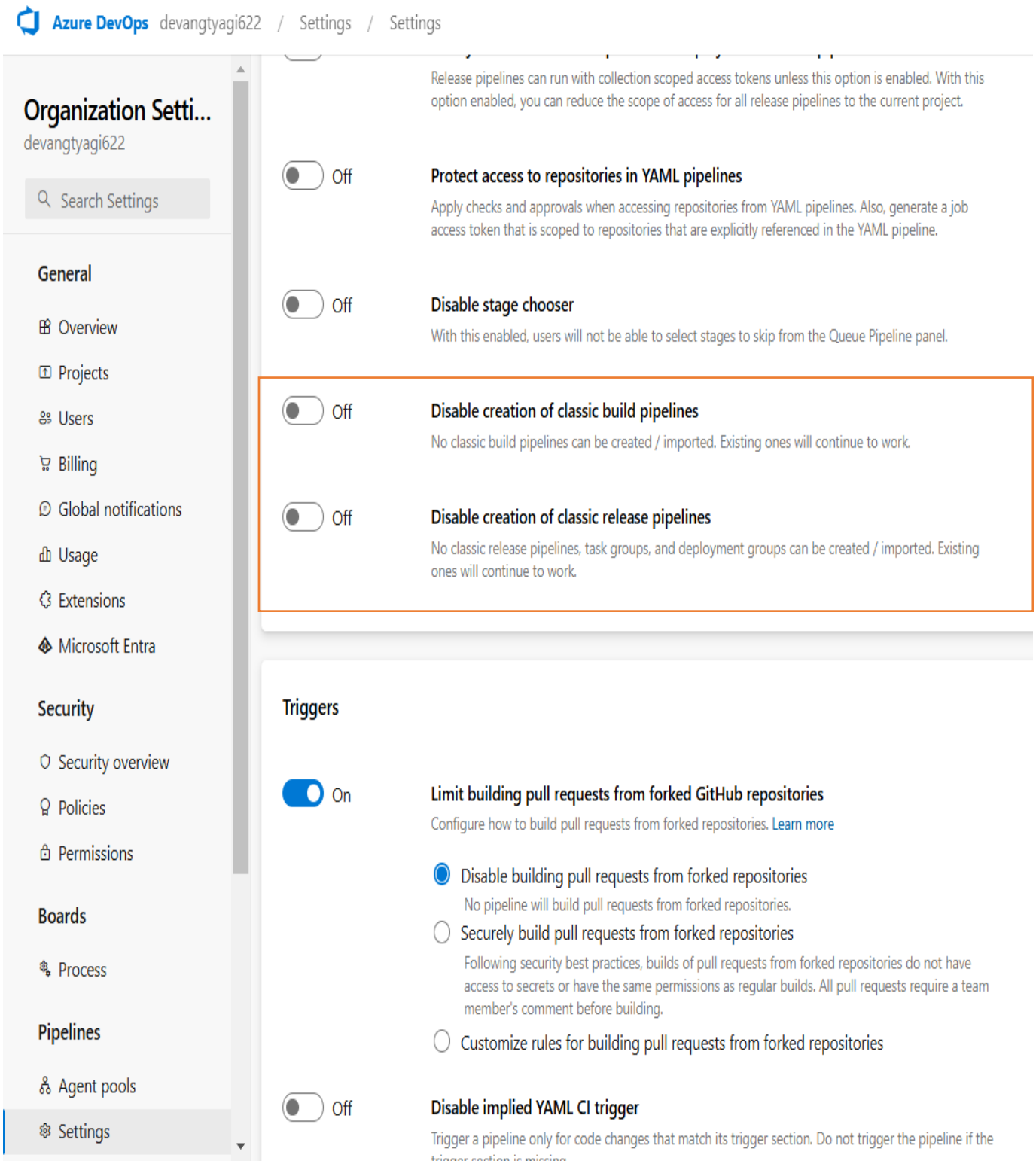


7. Navigate to ‘Parallel Jobs’ and change ‘Microsoft-hosted’ to 1. (If your project is private this will allow you to run the pipeline otherwise, the pipeline will not run).



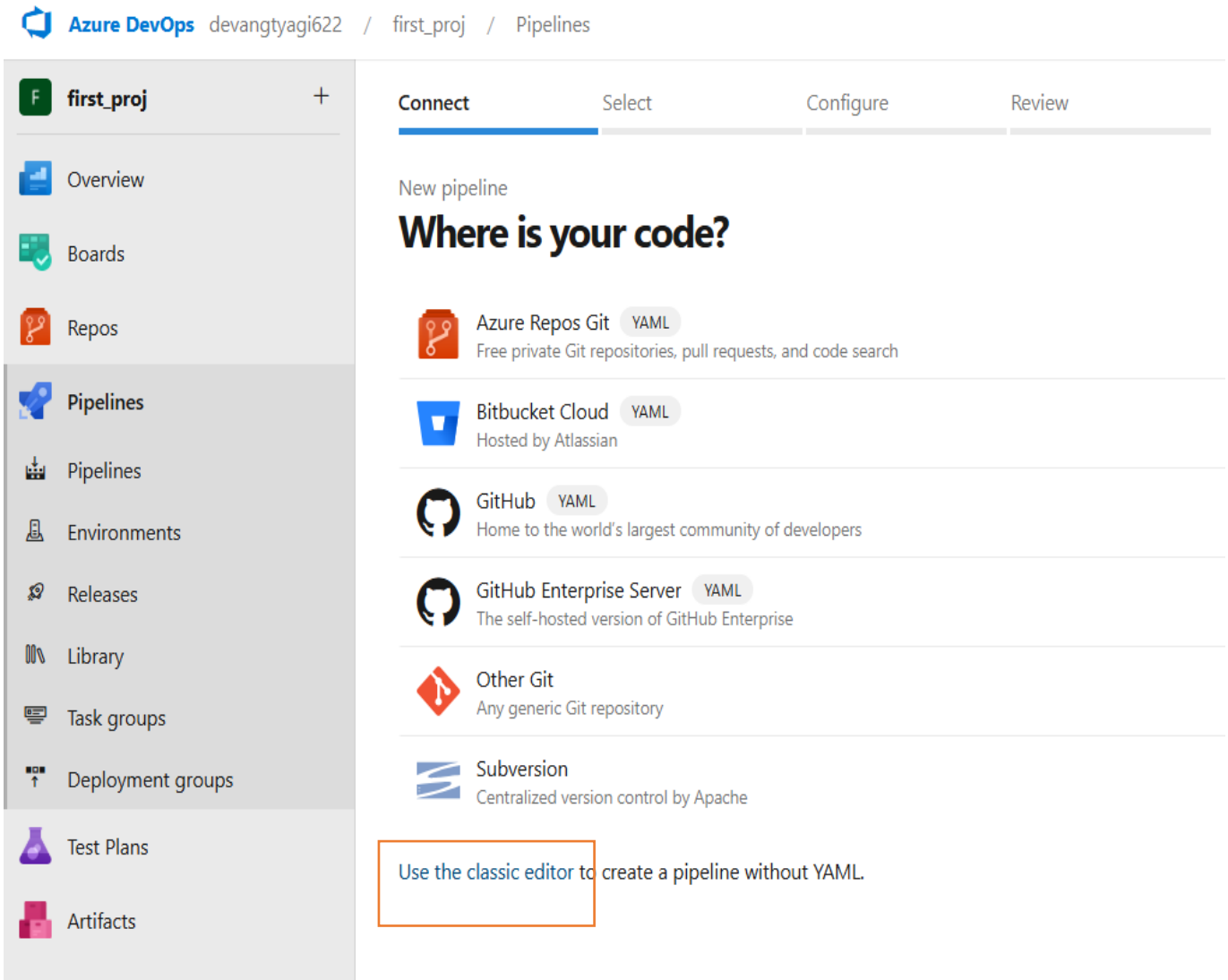
Create a Repo:

7. Navigate to Settings under Pipeline and turn-off the marked options since they are turned-on by default this will allow us to access the ‘Classic Editor’.



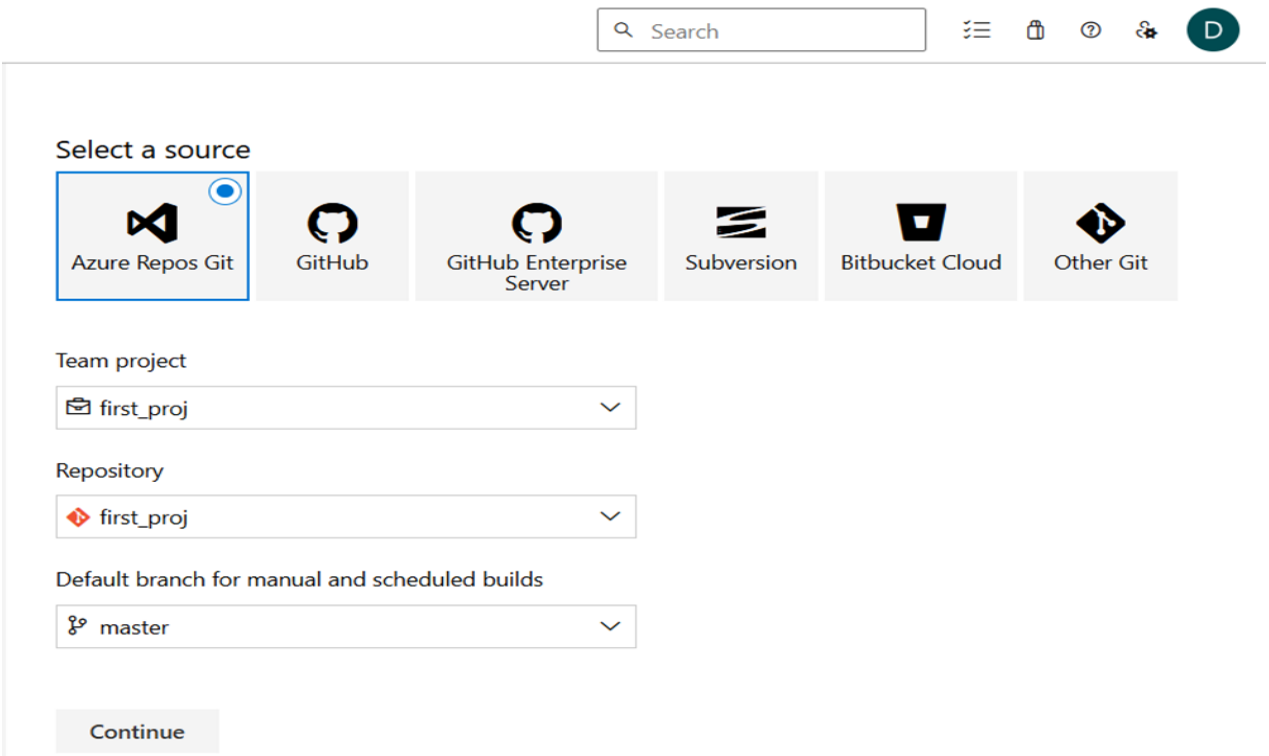
Create CI Pipeline:

1. Click on the project name, then click on the option ‘ Use the classic Editor’.

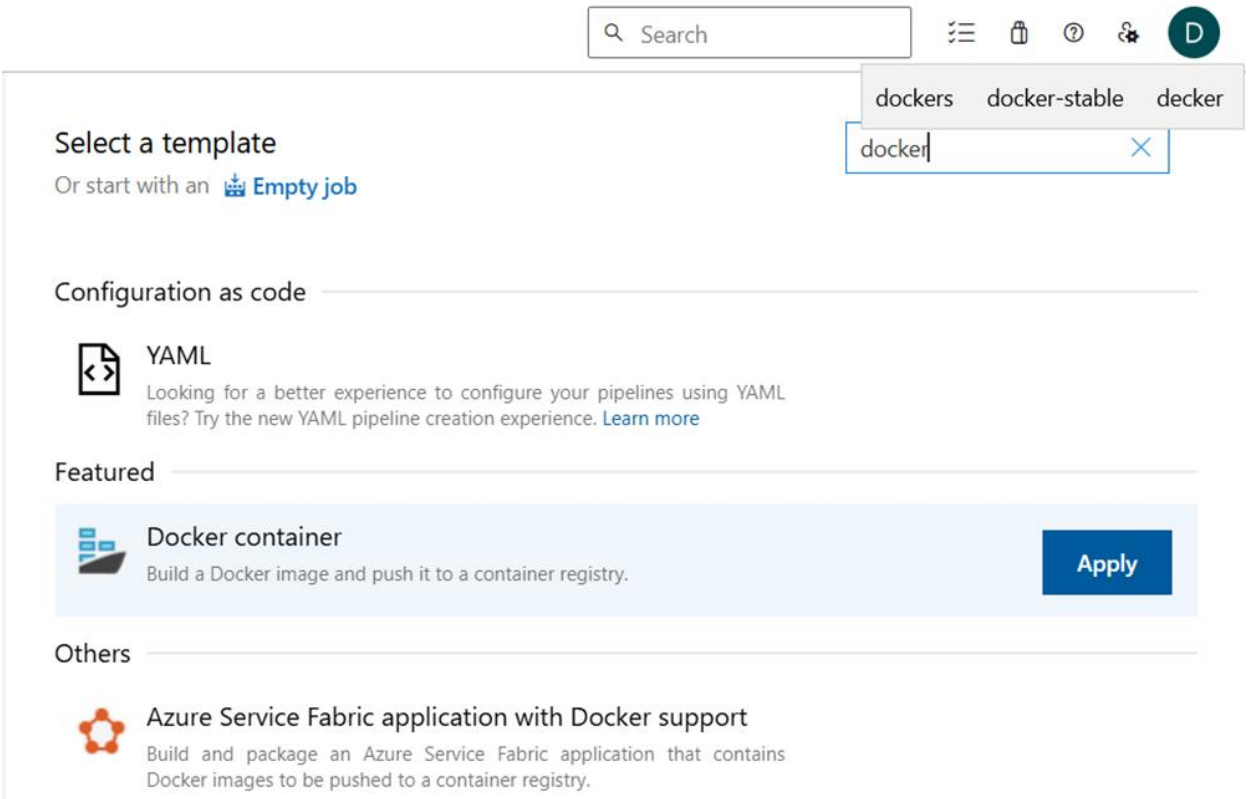


Create CI Pipeline:

2. Select the source, in this case, ‘Azure Repo Git’ and make the ‘Default branch’ to ‘master’.
3. Then click continue.

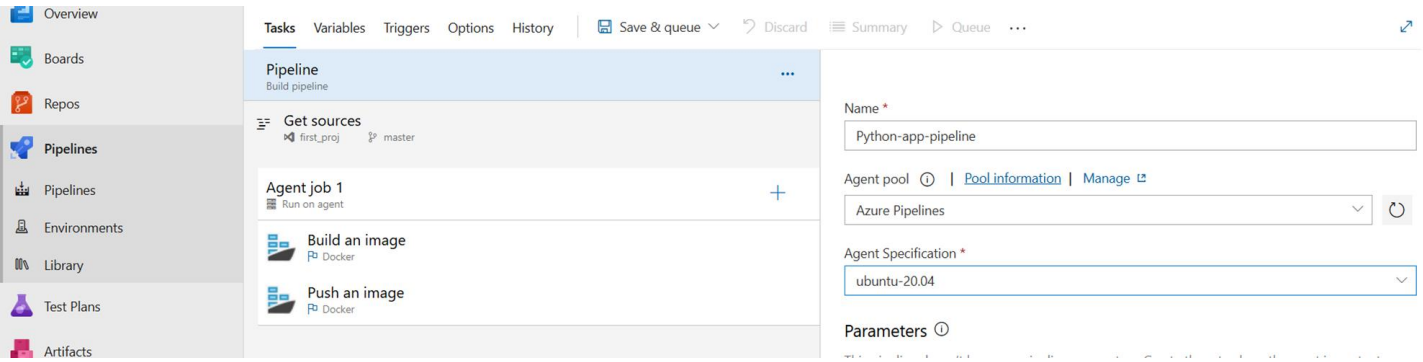


4. Search for a template and type ‘docker’ then apply.

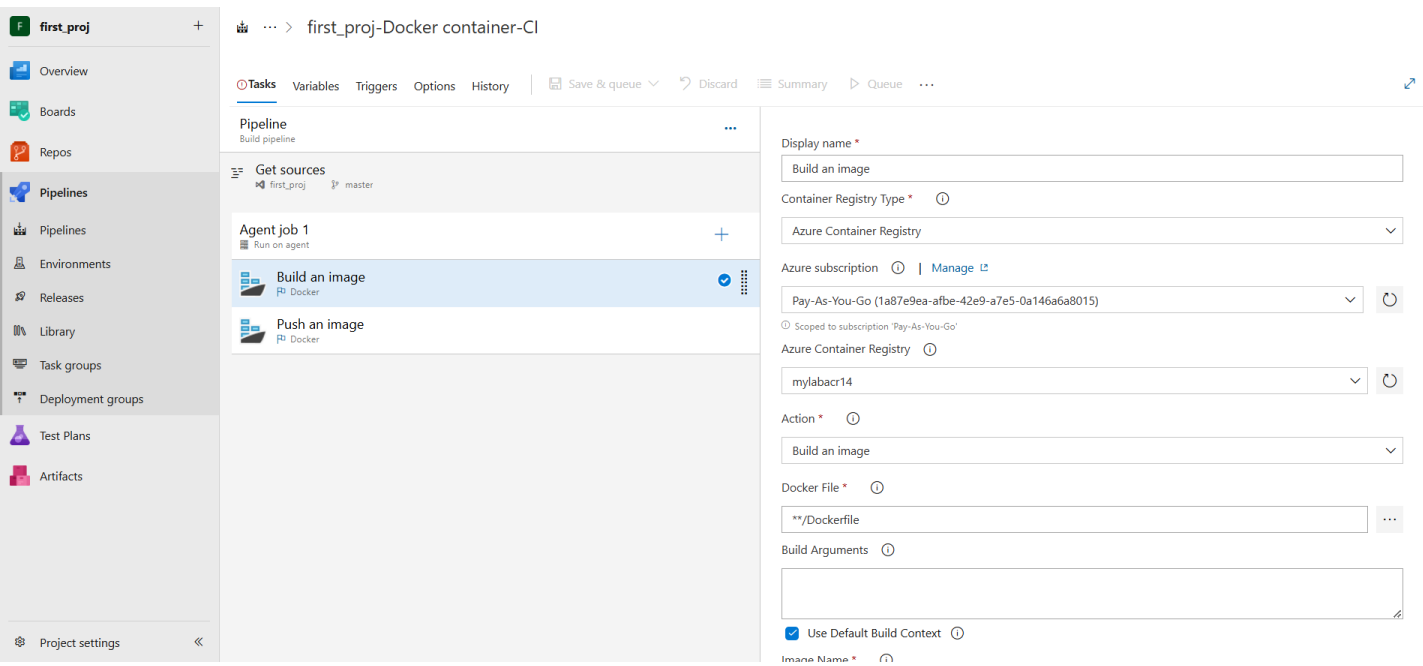


Create CI Pipeline:

5. Name the pipeline
6. In the ‘Agent pool’ select ‘Azure Pipelines’
7. In the ‘Agent Configuration’ select the desired version.



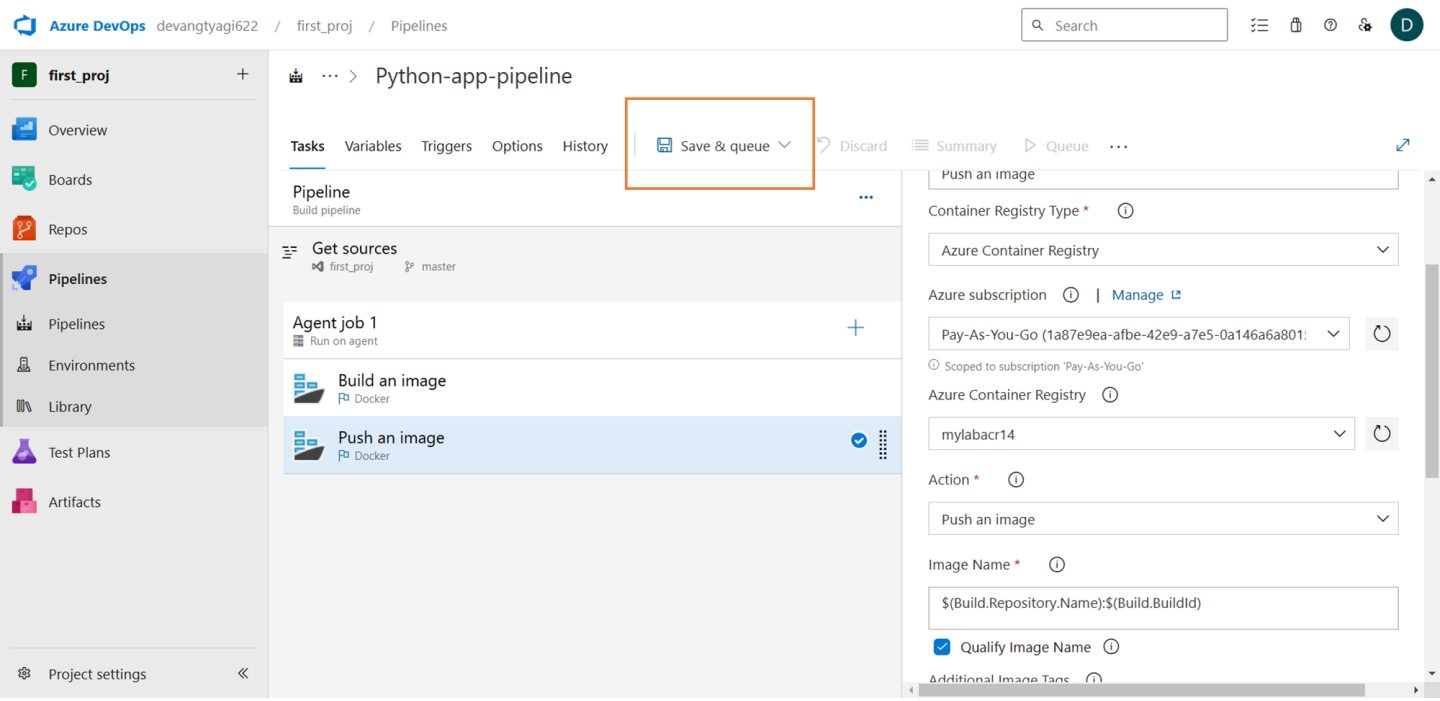
8. Under ‘Build an Image’ do the following:
   - Select the container registry that you created.
   - Select the respective Subscription using which all the resources were built.
   - Under the Docker file: leave it with the default name.
   - Tick the ‘Use Default Build Context’.



Create CI Pipeline:

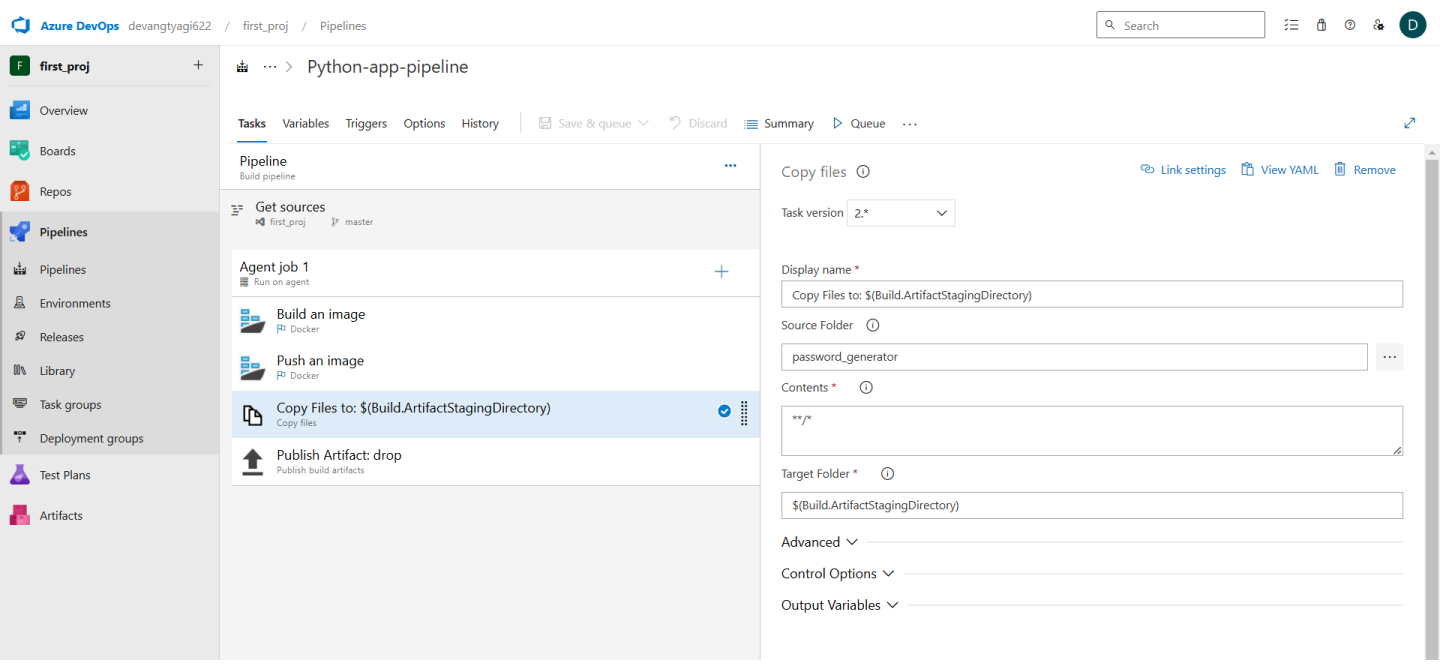
9. Under ‘Push an Image’ do the following:
   - Select the container registry that you created.
   - Select the respective Subscription using which all the resources were built.
   - Under Image Name hover over (i) icon and copy-paste the text
   - Tick ‘Qualify Image name’.

10. Once done with the configuration Click on ‘Save and queue’.

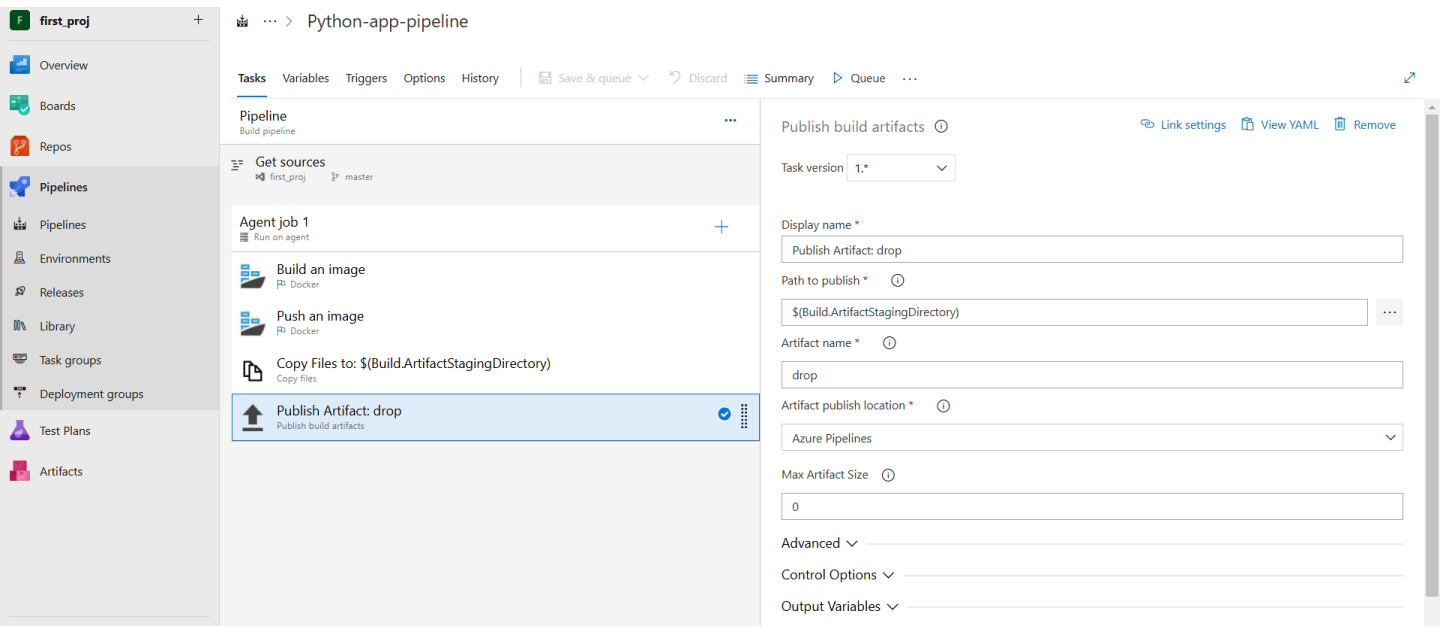


Create CI Pipeline:

9. Under ‘Copy to’, under ‘Contents’ you have to copy-paste the .yaml file name but sometimes it does not appear in the drop. So type ‘**/*’ this will copy all the files in drop.



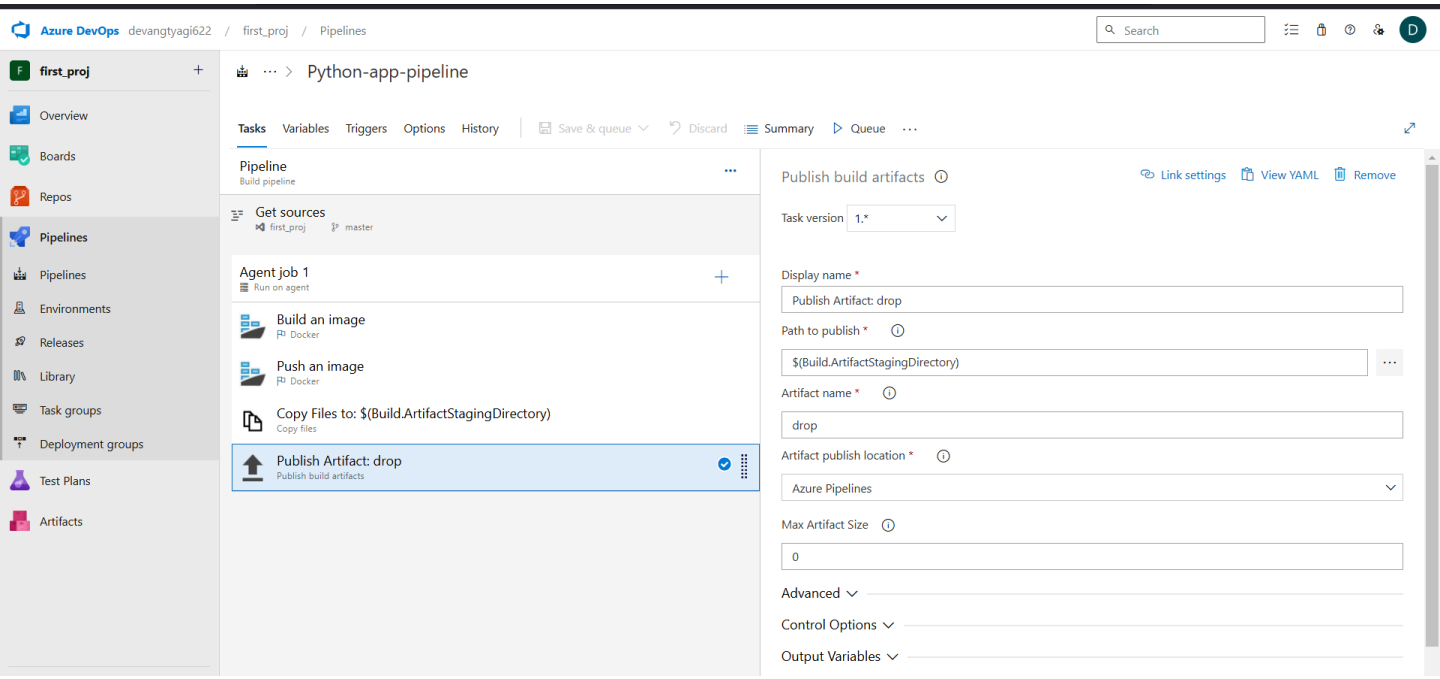
10. Under ‘Publish Artifact’, the default settings should not be changed.



Create CI Pipeline:

9. Under ‘Push an Image’ do the following:
   - Select the container registry that you created.
   - Select the respective Subscription using which all the resources were built.
   - Under Image Name hover over (i) icon and copy-paste the text
   - Tick ‘Qualify Image name’.

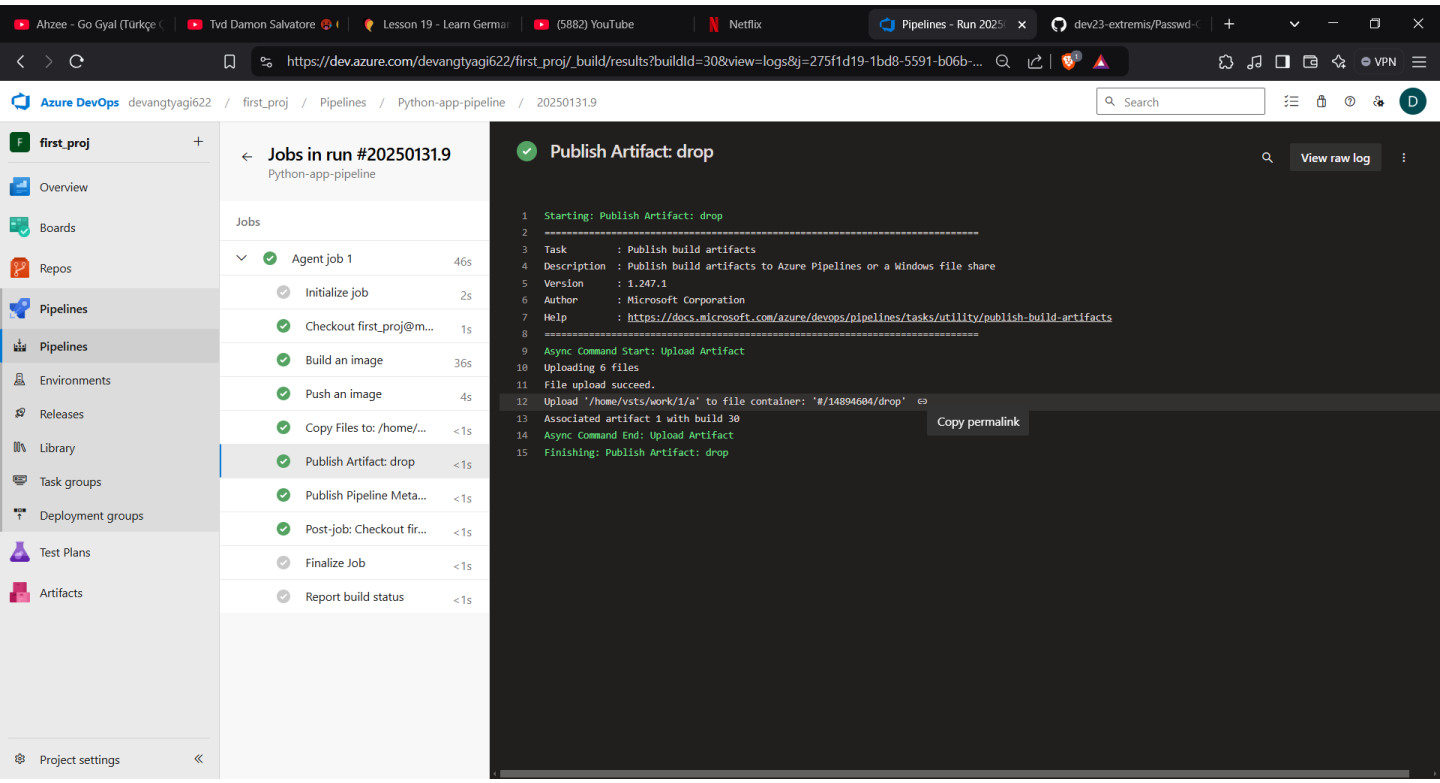
10. Once done with the configuration Click on ‘Save and queue’.



Create CI Pipeline:

Once done it will be shown as below made sure to check under Publish job in drop if the files were saved.

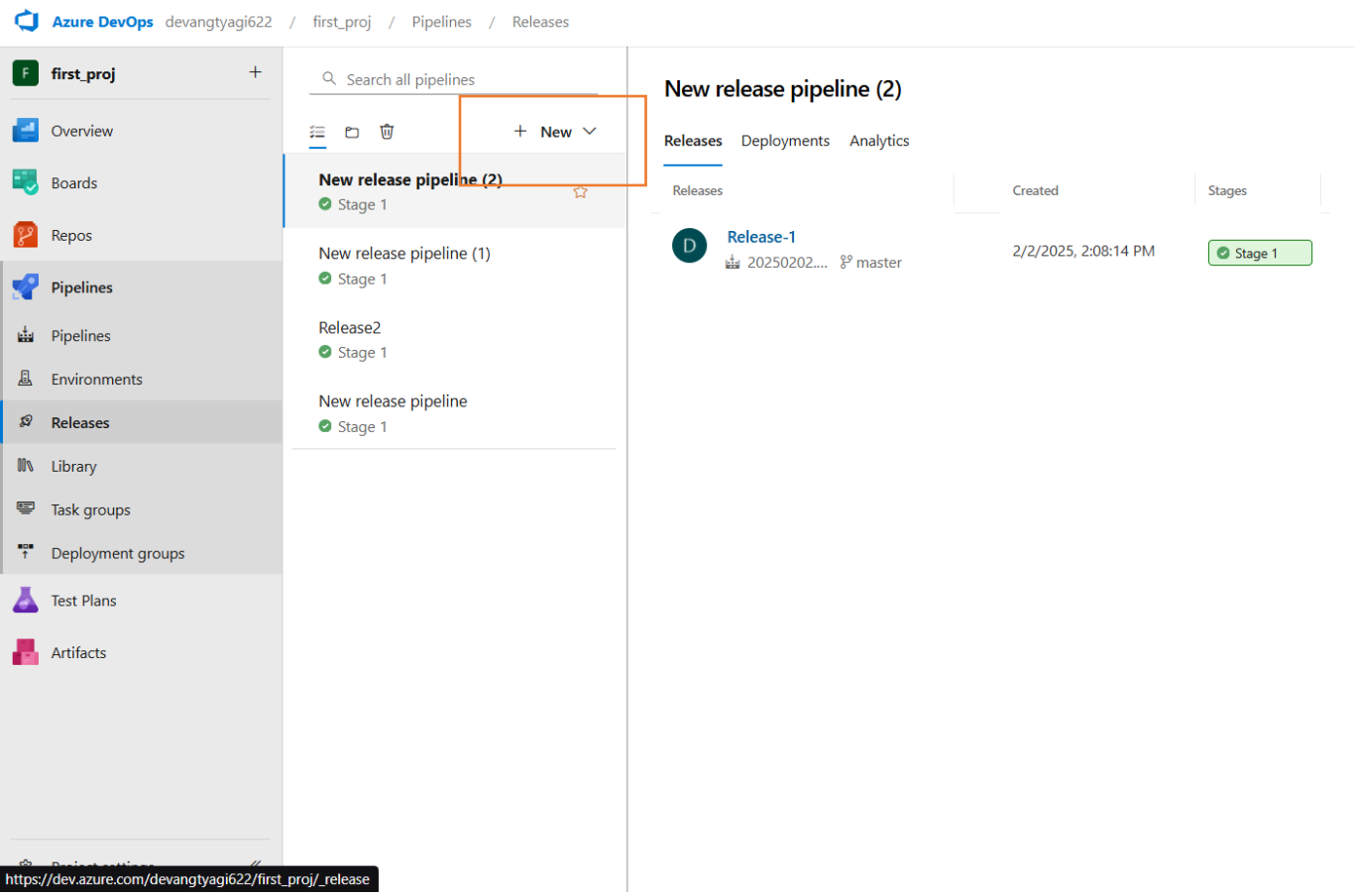
If not Under ‘Copy to’, under ‘Contents’ you have to copy-paste the .yaml file name but sometimes it does not appear in the drop.
So type ‘**/*’ this will copy all the files in drop. Then run again.



Create a CD or release Pipeline:

Here the CI pipeline will be exposed.

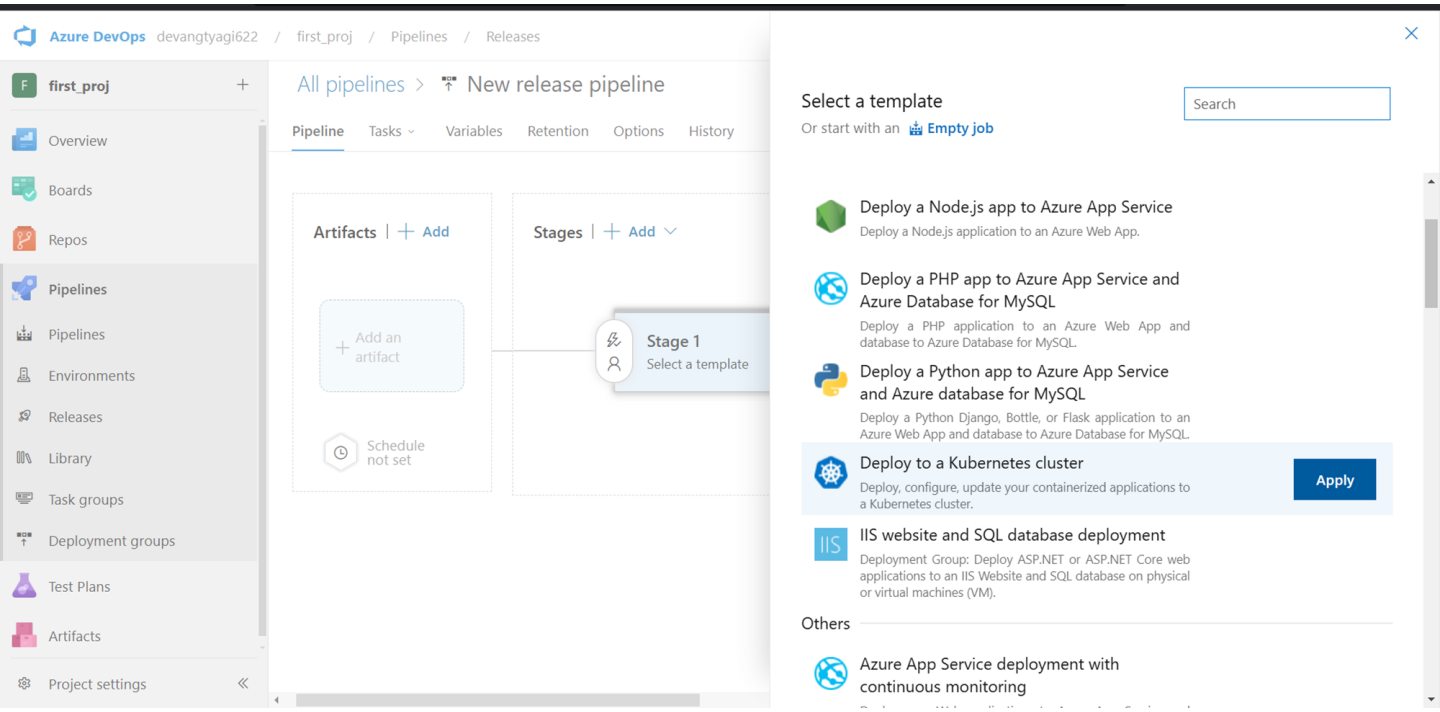
1. Click ‘New’ and choose ‘New release Pipeline’.



Create a CD or release Pipeline:

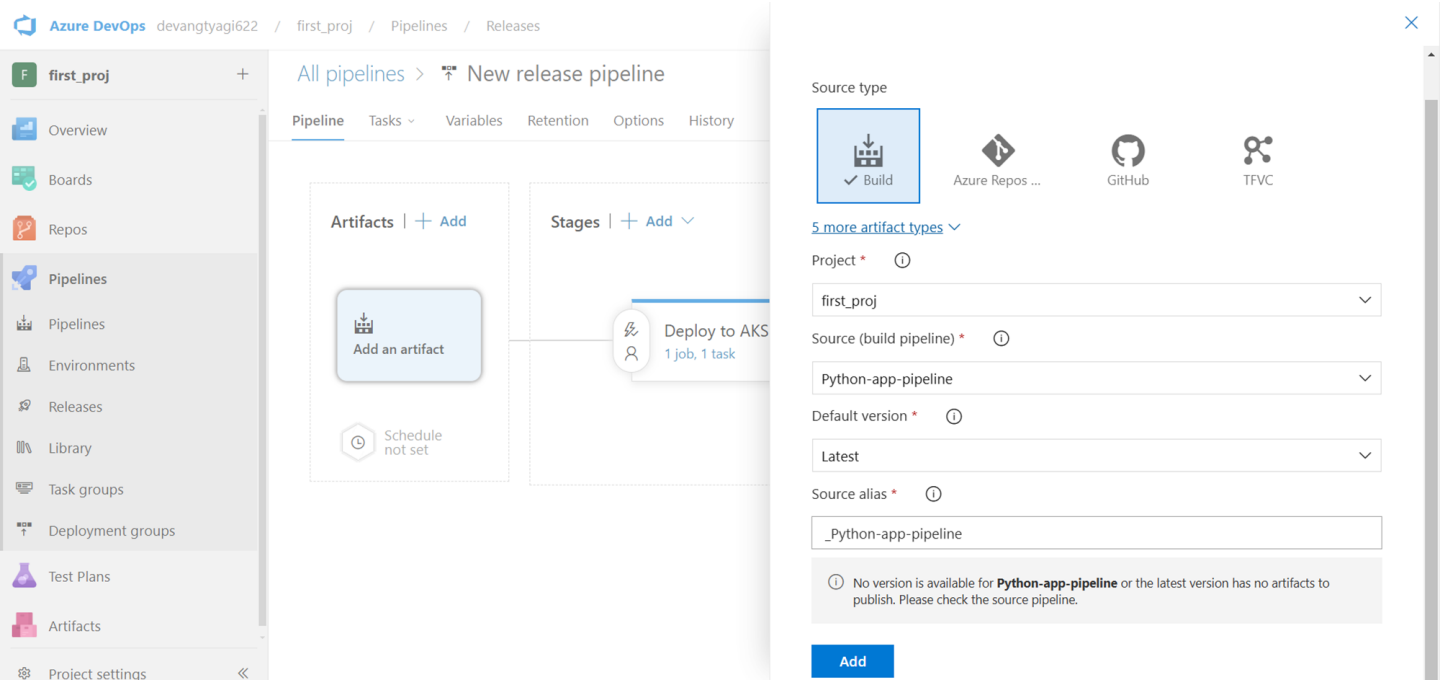
Here the CI pipeline will be exposed.

2 . Click On ‘Stage 1’ and then select the template ‘Deploy to a Kubernetes cluster’.

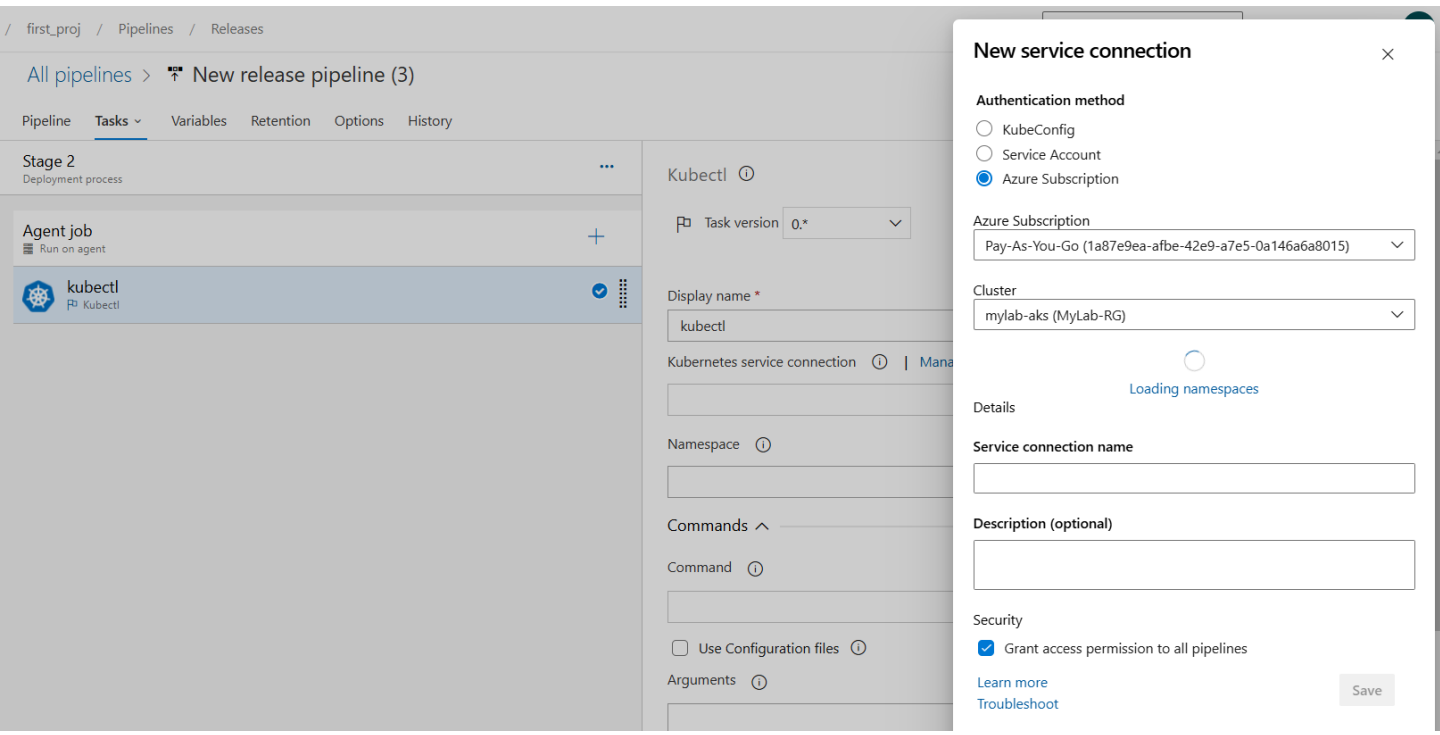


Create a CD or release Pipeline:

2 . Click On ‘Add an artifact’ and then select the source and click ‘Add’

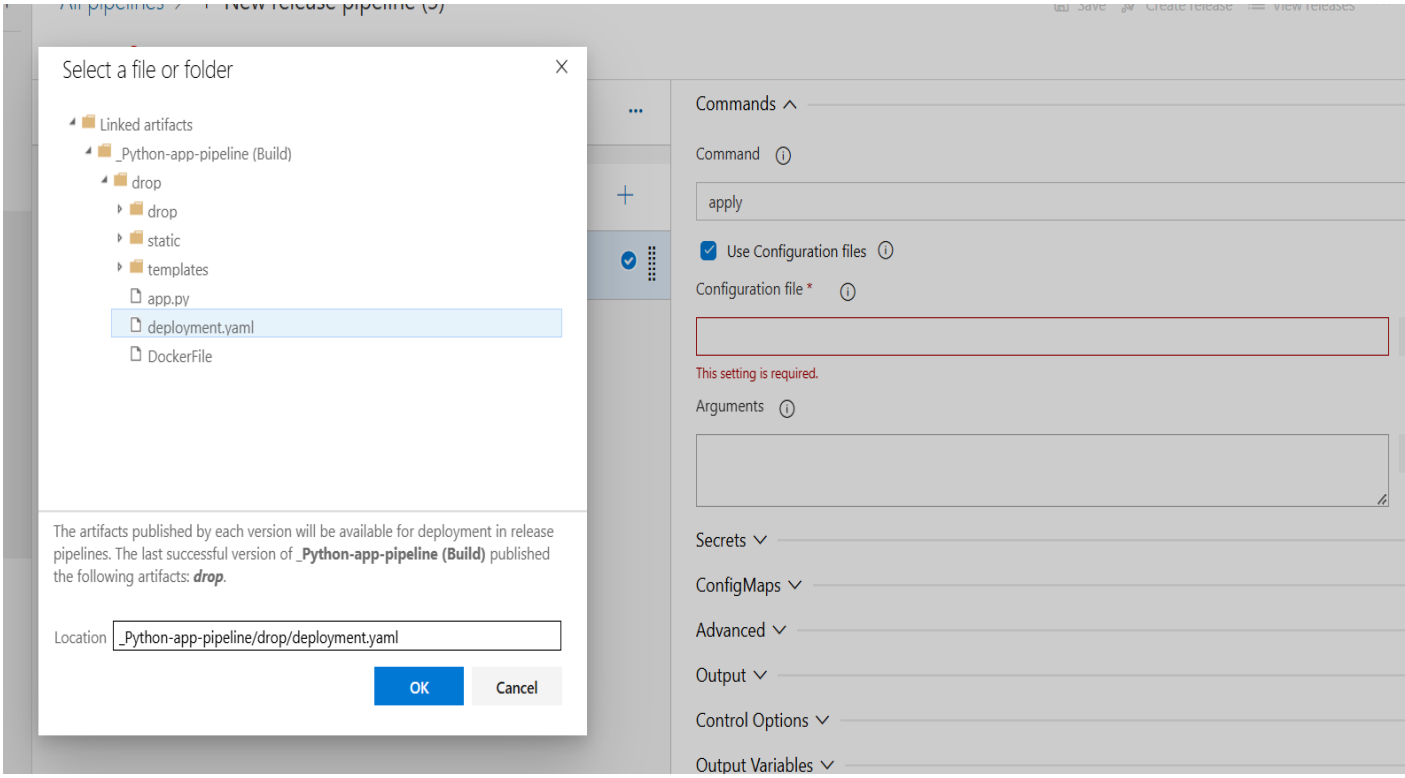


3. Click on ‘kubectl’, then select the ‘Azure subscription’ give ‘namespace’ name ‘default’ and in the service name give any name.

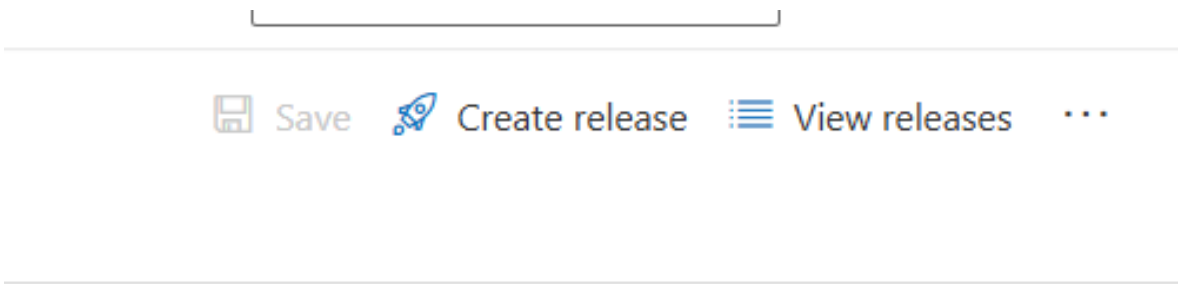


Create a CD or release Pipeline:

4 . Under ‘command’ select ‘apply’
5. Tick the configuration file.
6. Browse the .yaml file in drop as shown in the picture.

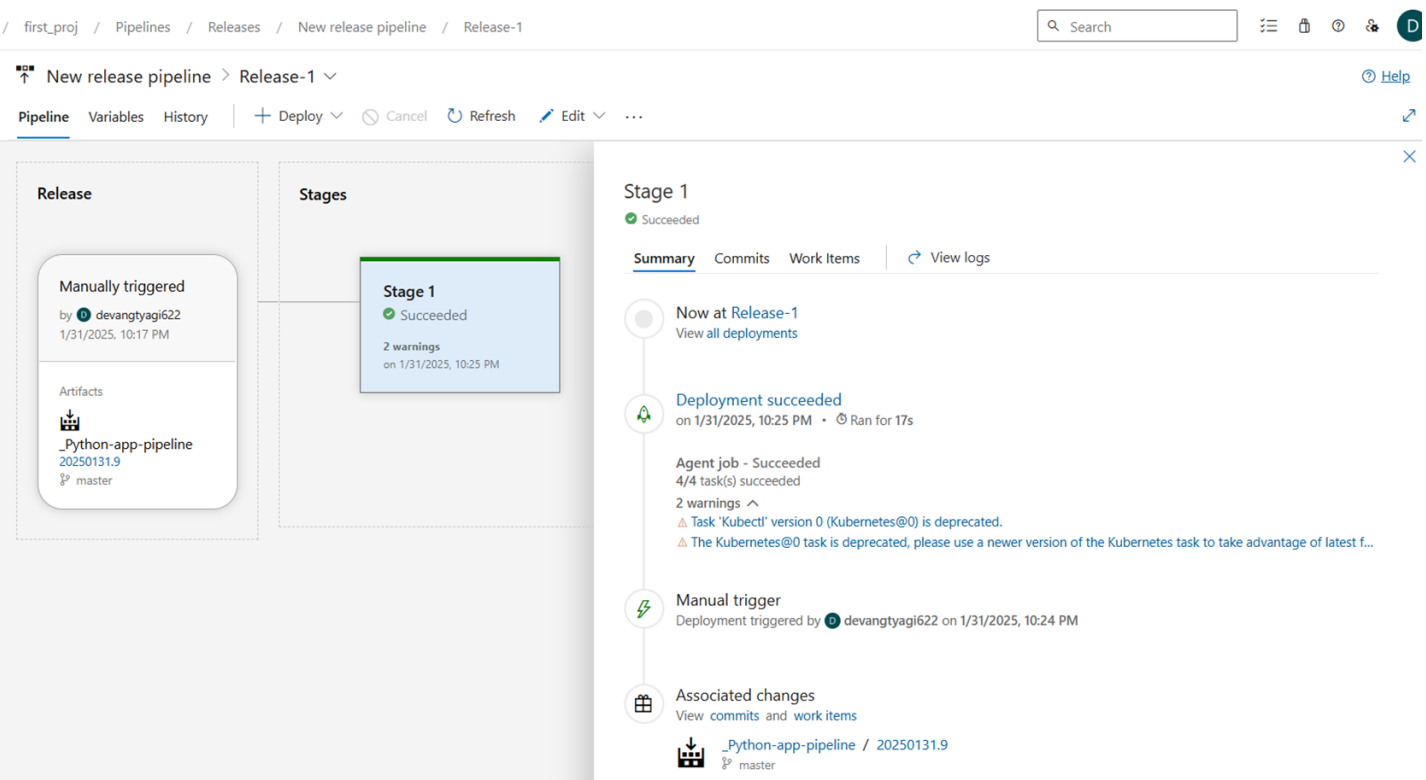


7. Click ‘Save’ and then click ‘create release’

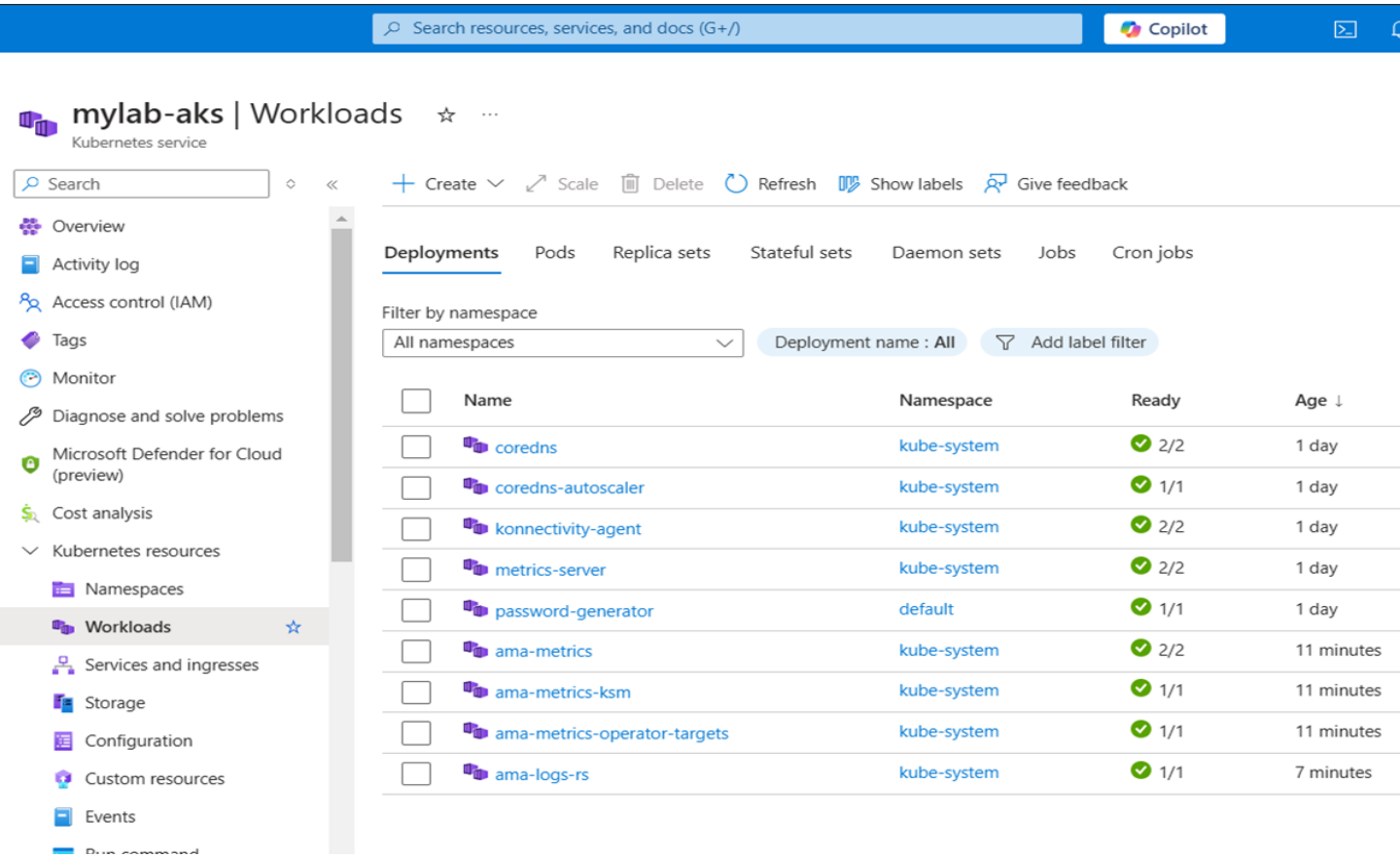


Create a CD or release Pipeline:

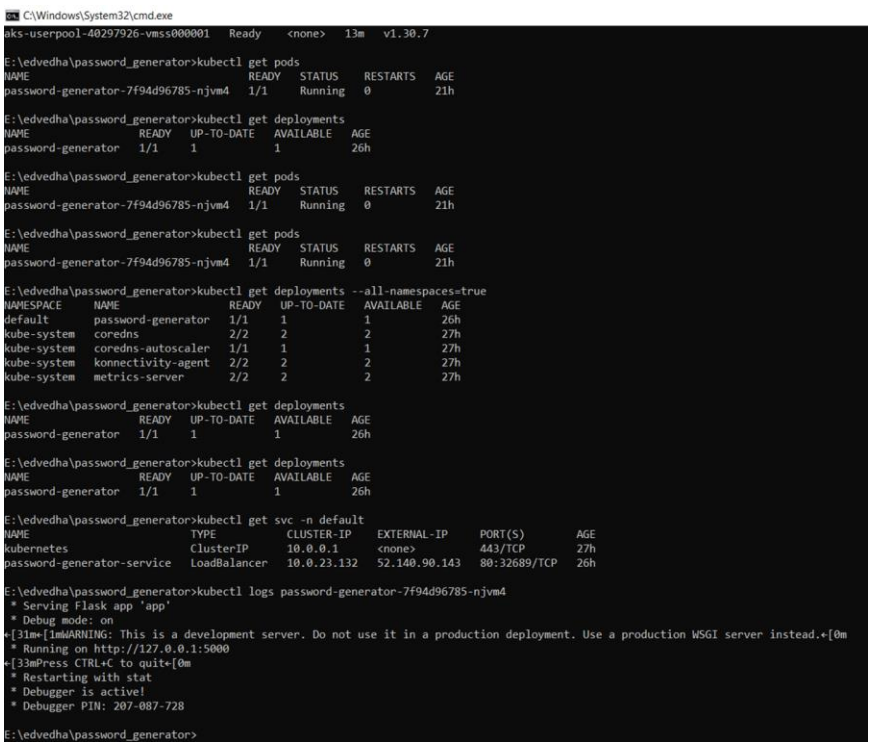
Once it is created successfully it will show the status succeeded and you check the logs also.



Once the deployment is successful in the Kubernetes cluster under ‘workloads’ the namespaces should look like the below.

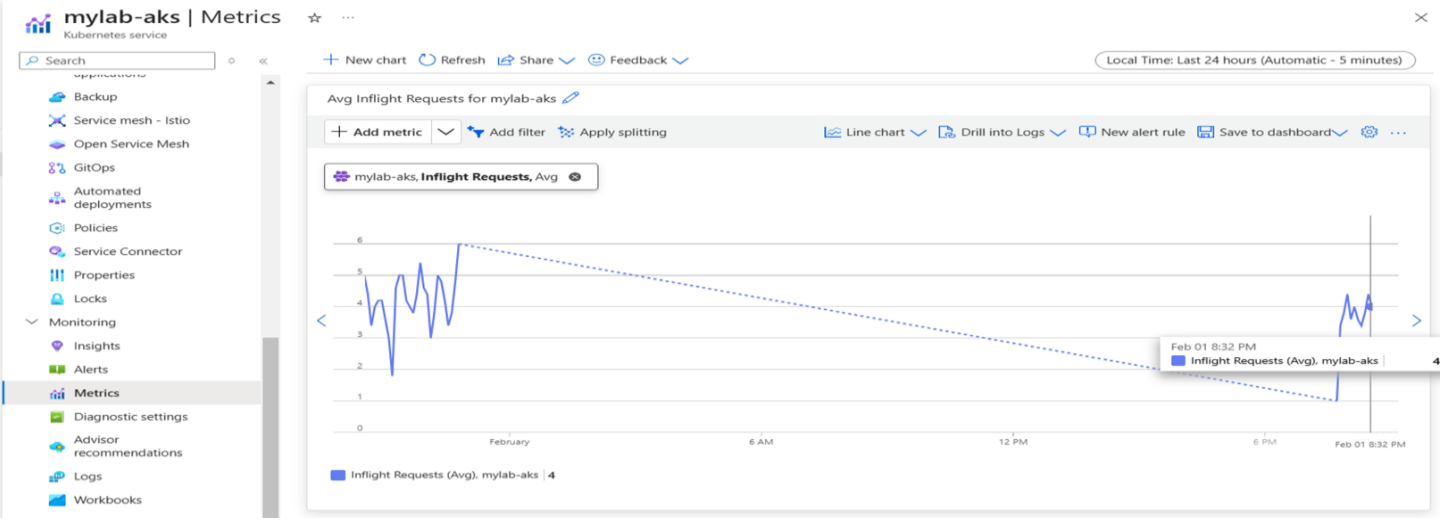


Using the Azure CLI the IP of the cluster is exposed or using Service and ingress just under workloads.

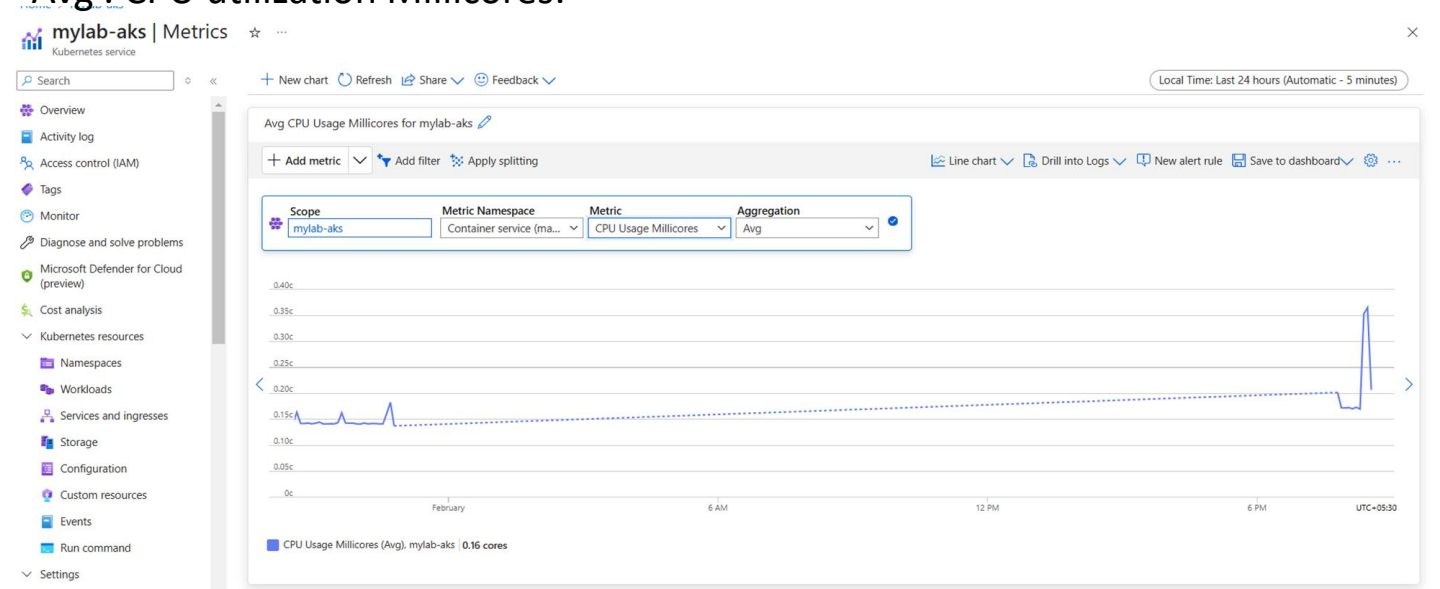


Metrics:

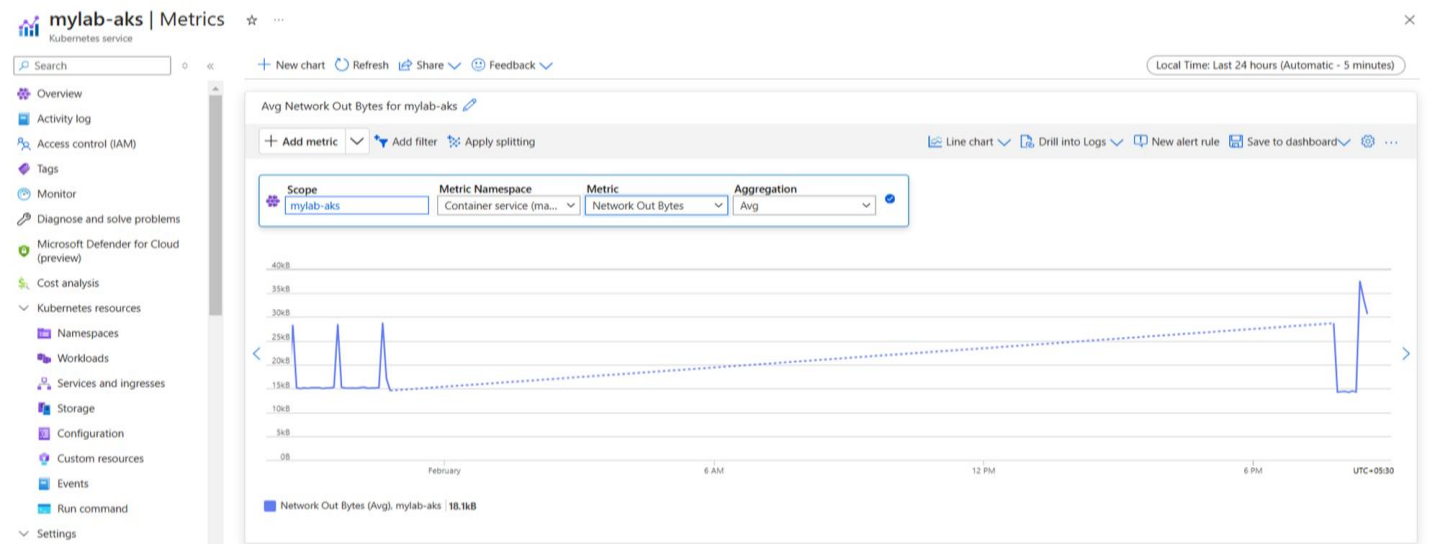
Avg. Inflight Requests:



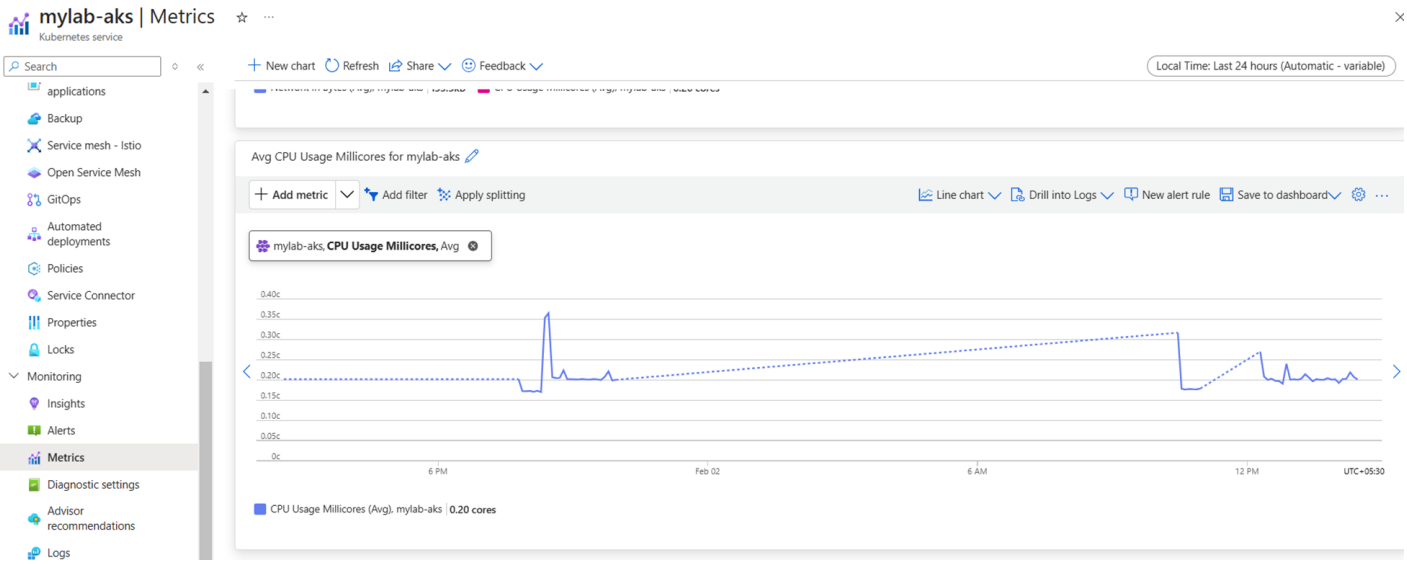
Avg . CPU utilization Millicores:

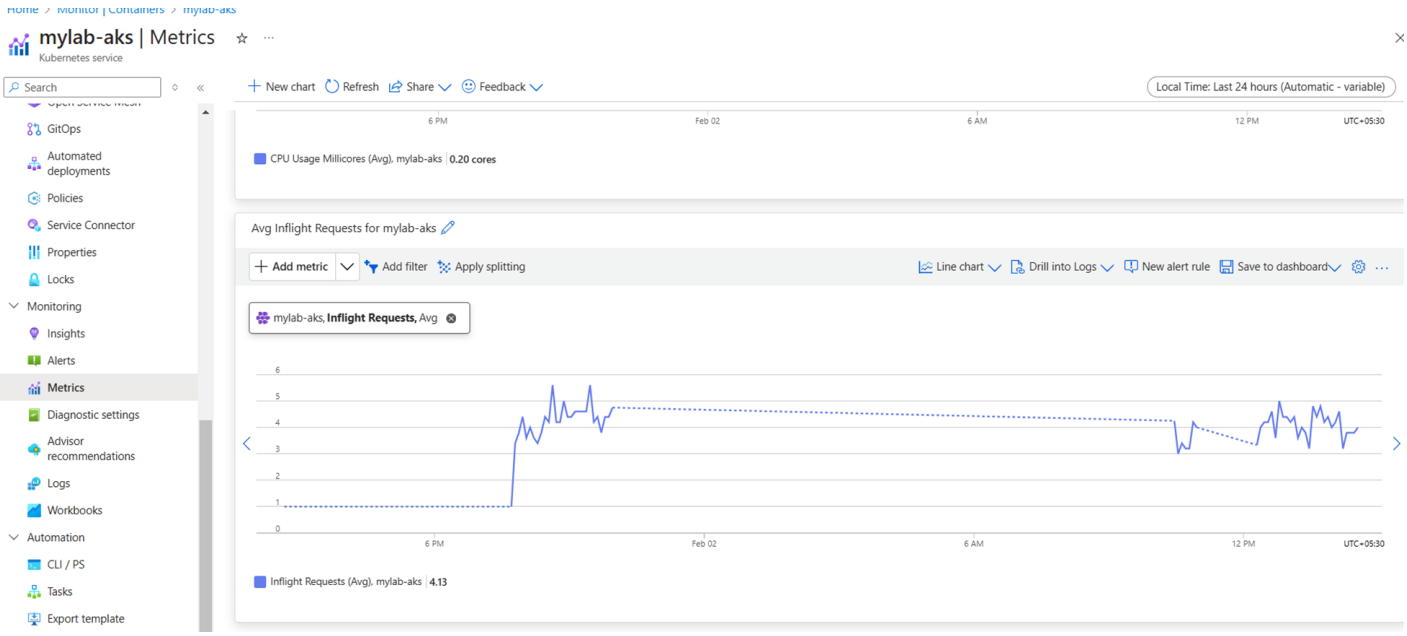


Avg. Network Out Bytes:

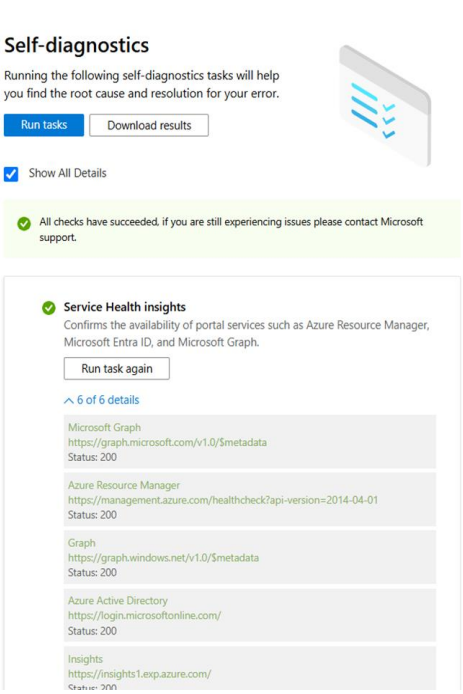


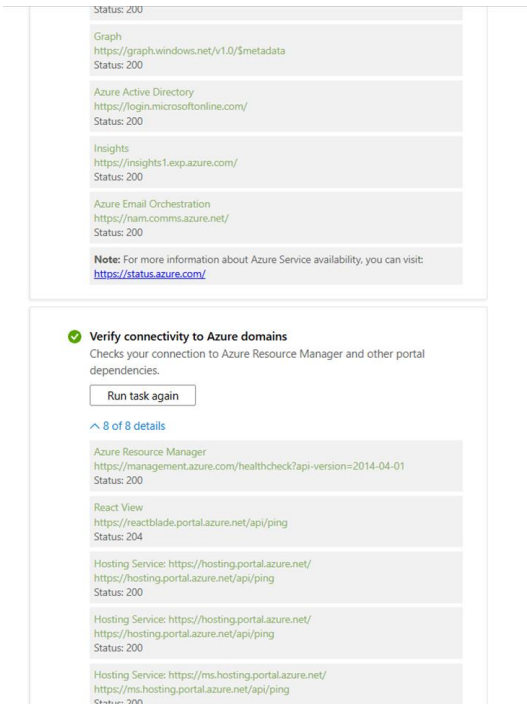
Metrics:

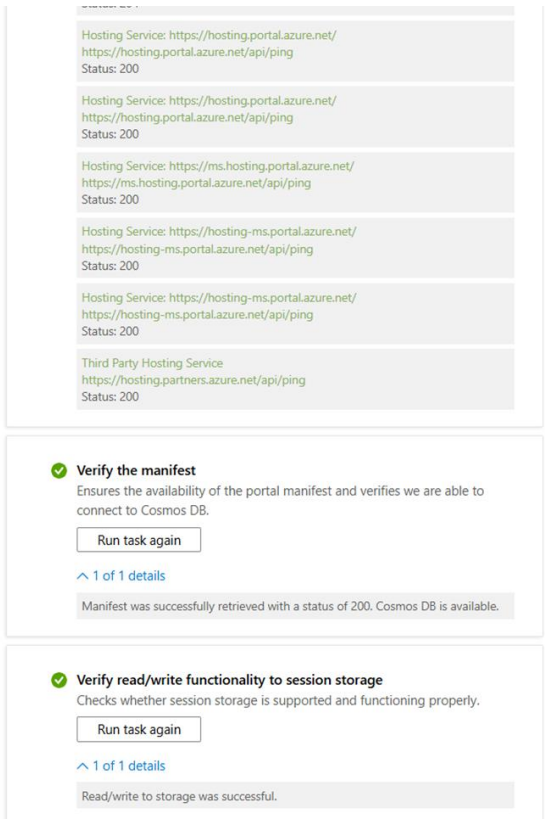

mylab-aks | Metrics
Kubernetes service
Search
New chart
Refresh
Share
Feedback
Local Time: Last 24 hours (Automatic - variable)
applications
Backup
Service mesh - Istio
Open Service Mesh
GitOps
Automated deployments
Policies
Service Connector
Properties
Locks
Monitoring
Insights
Alerts
Metrics
Diagnostic settings
Advisor recommendations
Logs
Avg CPU Usage Millicores for mylab-aks
Add metric
Add filter
Apply splitting
Line chart
Drill into Logs
New alert rule
Save to dashboard
mylab-aks, CPU Usage Millicores, Avg
0.40c
0.35c
0.30c
0.25c
0.20c
0.15c
0.10c
0.05c
0c
6 PM
Feb 02
6 AM
12 PM
UTC+05:30
CPU Usage Millicores (Avg), mylab-aks 0.20 cores

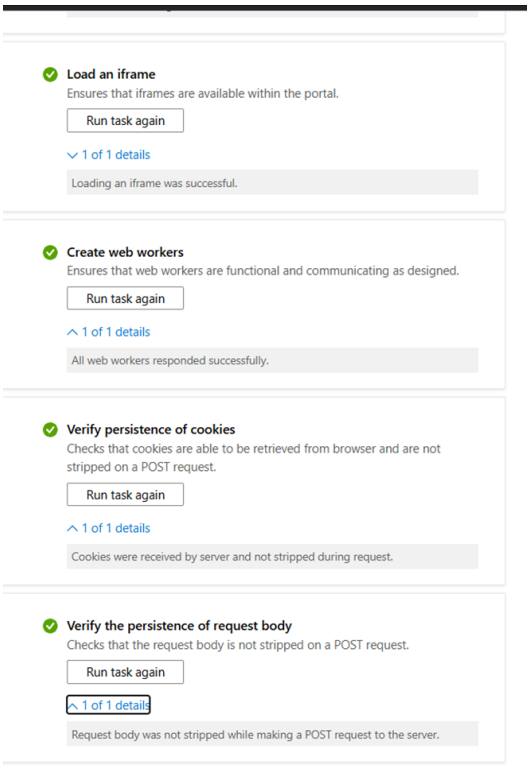

mylab-aks | Metrics
Kubernetes service
Search
New chart
Refresh
Share
Feedback
Local Time: Last 24 hours (Automatic - variable)
GitOps
Automated deployments
Policies
Service Connector
Properties
Locks
Monitoring
Insights
Alerts
Metrics
Diagnostic settings
Advisor recommendations
Logs
Workbooks
Automation
CLI / PS
Tasks
Export template
6 PM
Feb 02
6 AM
12 PM
UTC+05:30
CPU Usage Millicores (Avg), mylab-aks 0.20 cores
Avg Inflight Requests for mylab-aks
Add metric
Add filter
Apply splitting
Line chart
Drill into Logs
New alert rule
Save to dashboard
mylab-aks, Inflight Requests, Avg
6
5
4
3
2
1
0
6 PM
Feb 02
6 AM
12 PM
UTC+05:30
Inflight Requests (Avg), mylab-aks 4.13

Self Diagnostics Result: (Done by Azure itself)

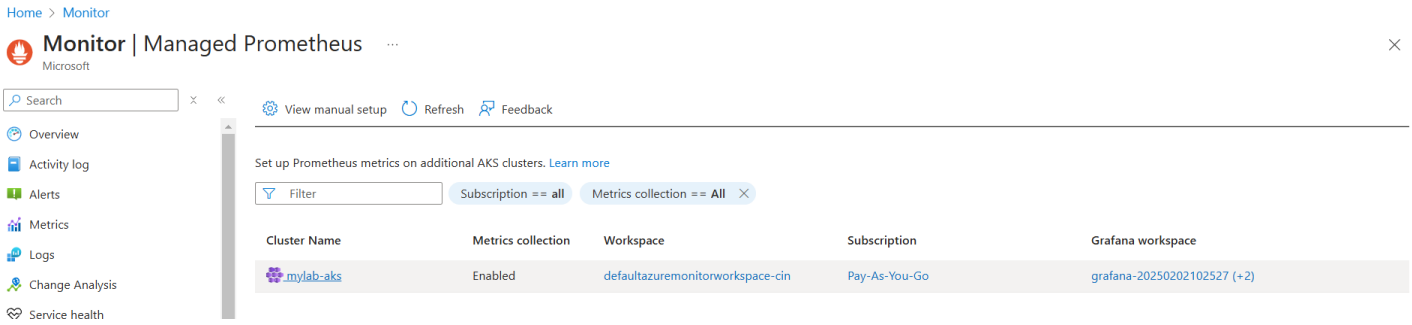




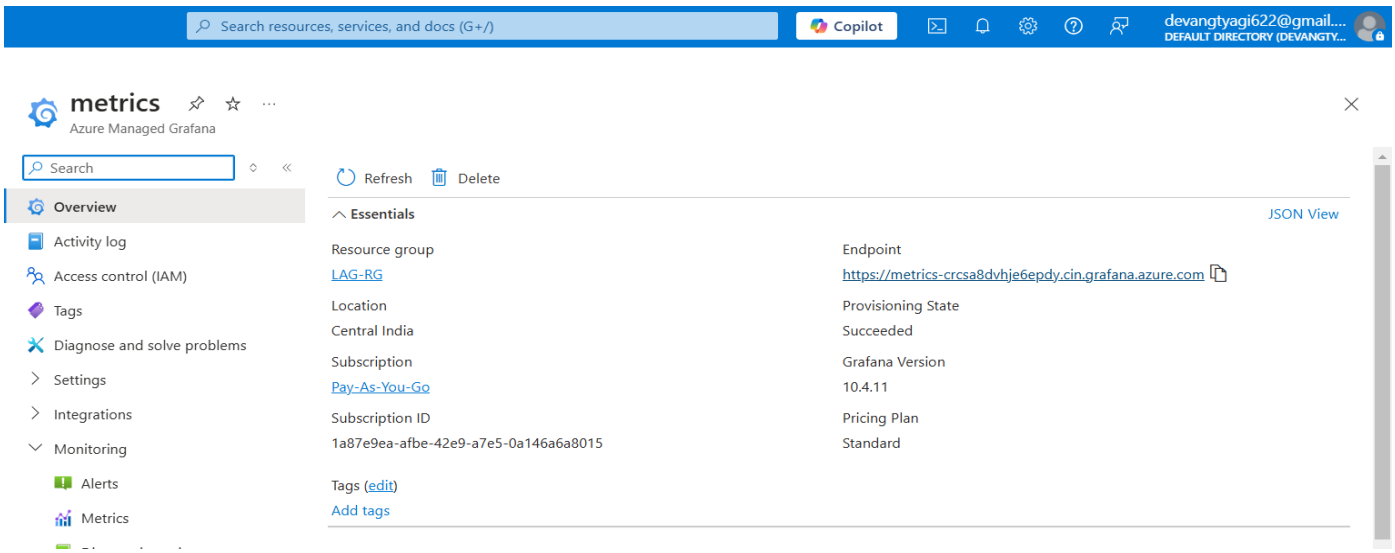




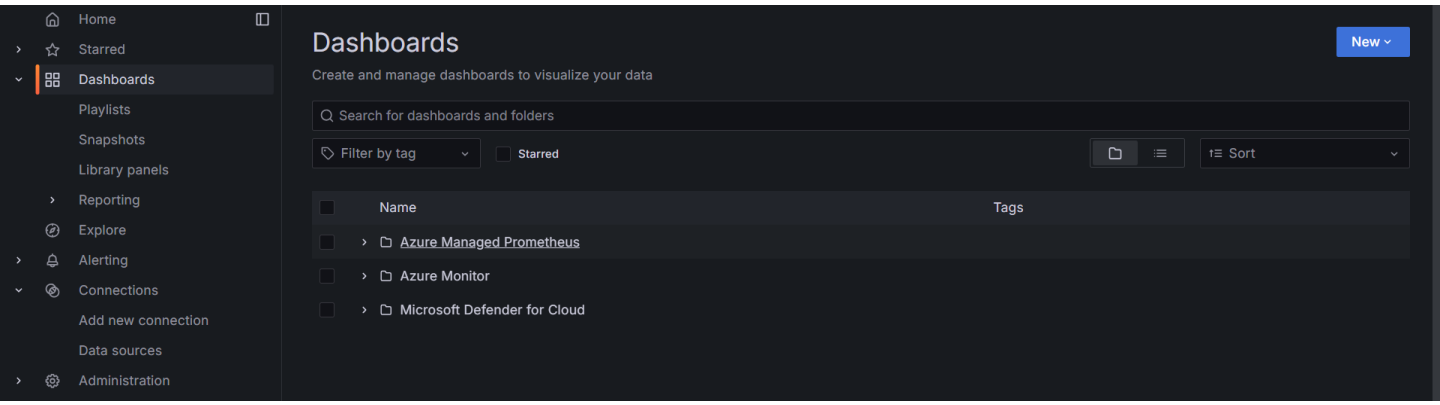
Apply Prometheus and Grafana to the Kubernetes cluster



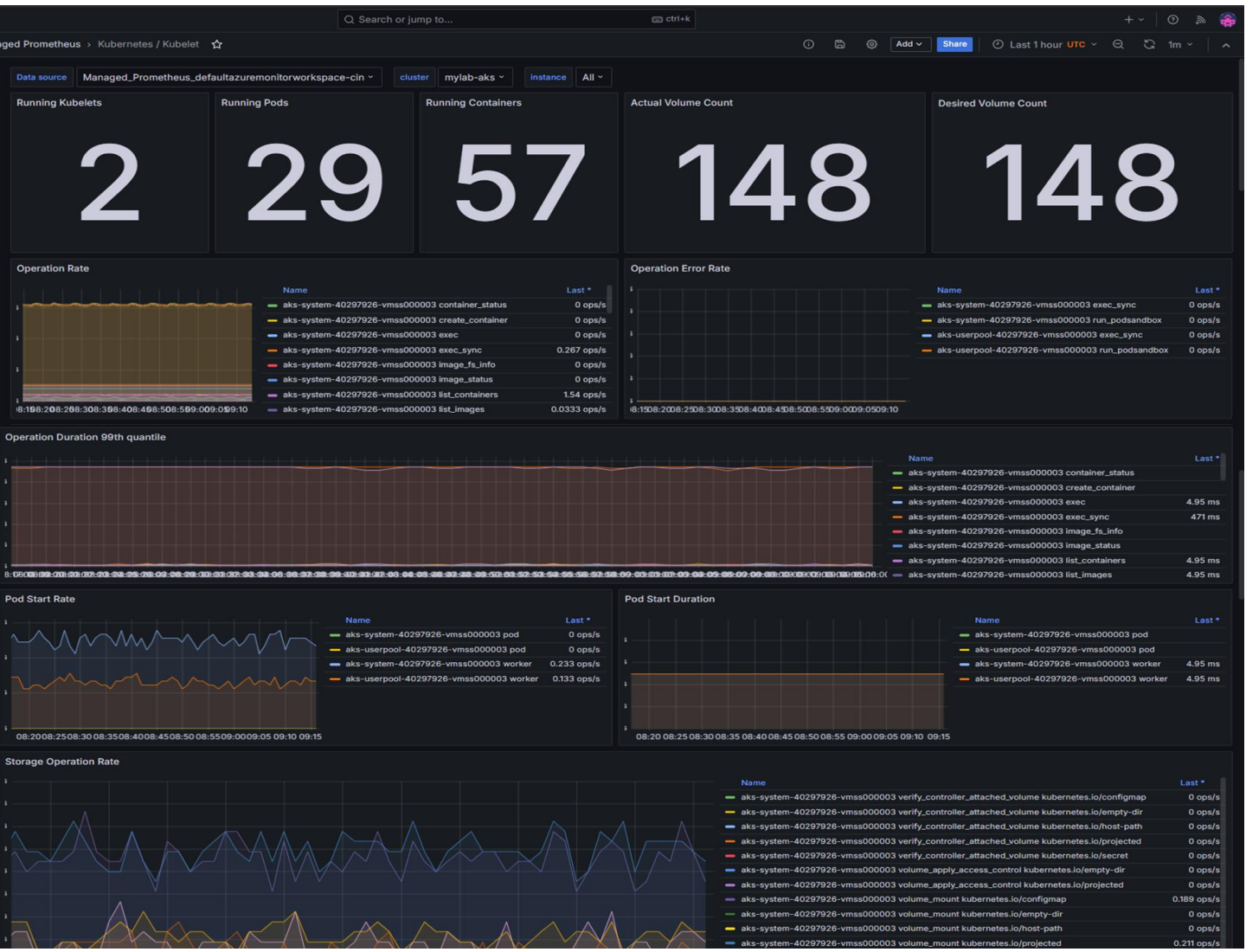
The name of the Grafana attached to Kubernetes cluster is ‘metrics’ .



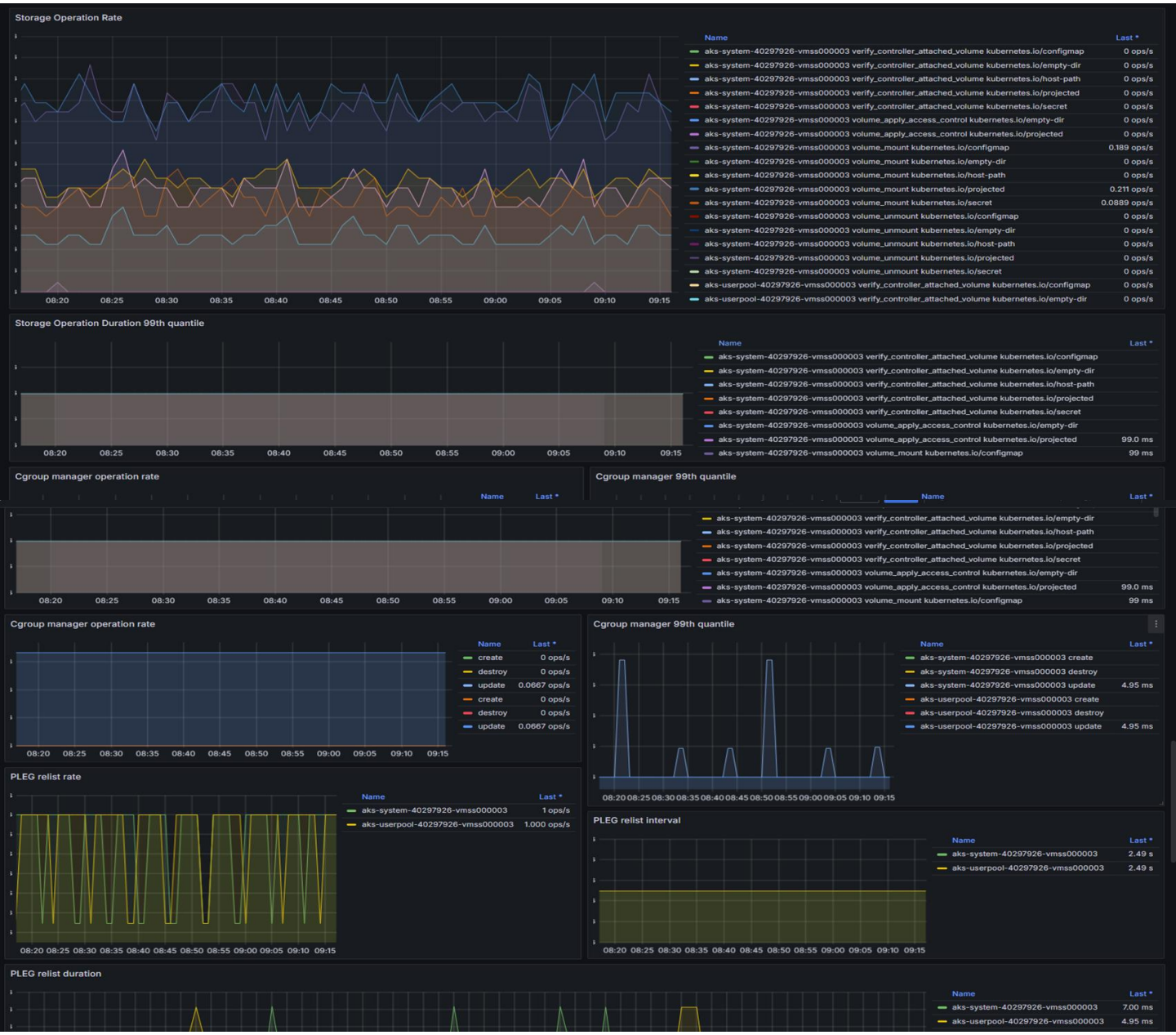
Grafana dashboards for the Kubernetes cluster



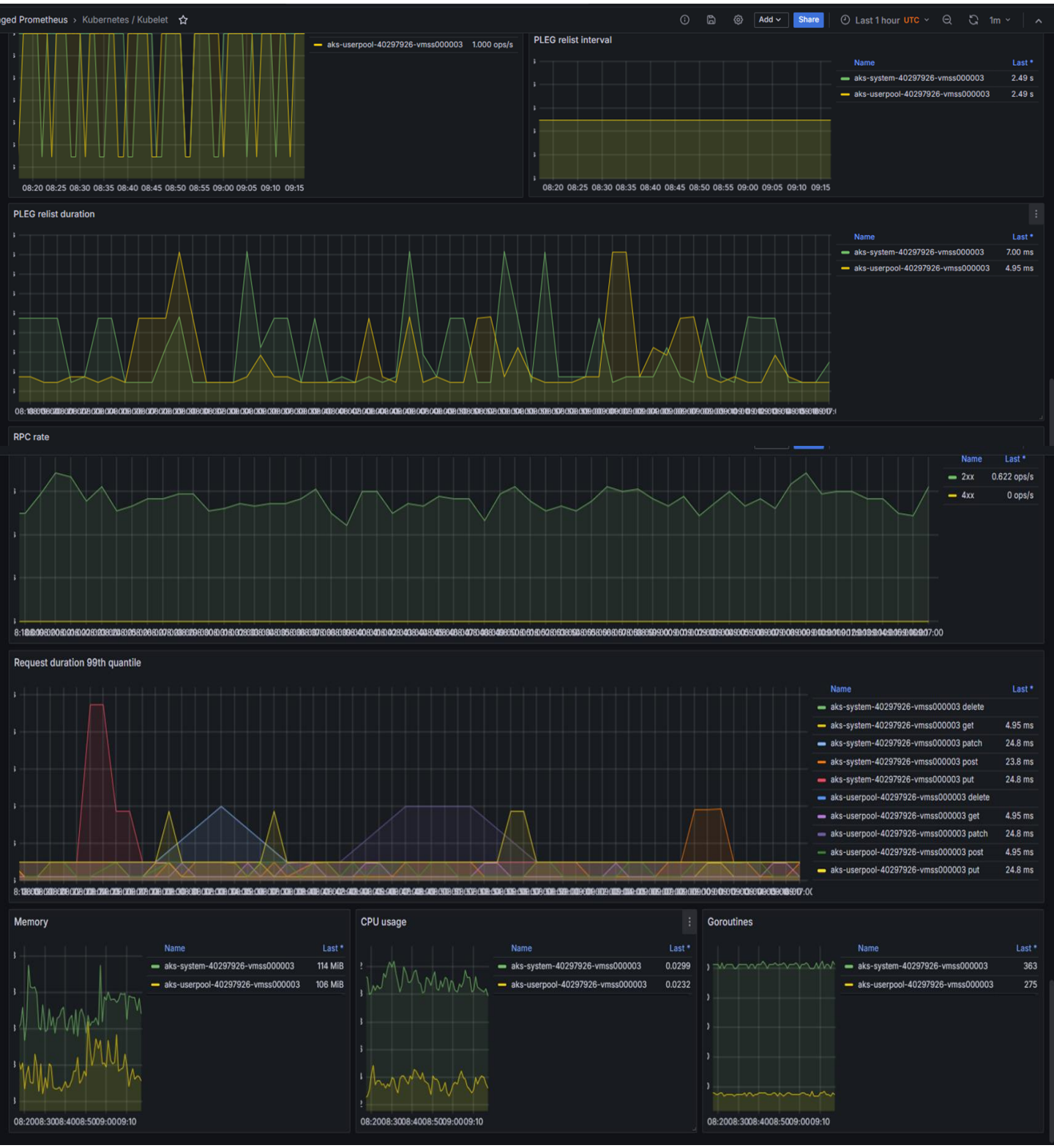
Azure Merged Prometheus> Kubernetes Dashboard



Azure Merged Prometheus> Kubernetes Dashboard



Azure Merged Prometheus> Kubernetes Dashboard



CPU Utilization Dashboard (for instance aks-system-40297926-vmss000003)

CPU Utilization Dashboard (for instance aks-userpool-40297926-vmss000003)