Full Name: Loc Tran Huynh Cong

Staff Code: SD4257

Link GitHub:

1. sd4257_msa
2. sd4257_azure_infrastructure

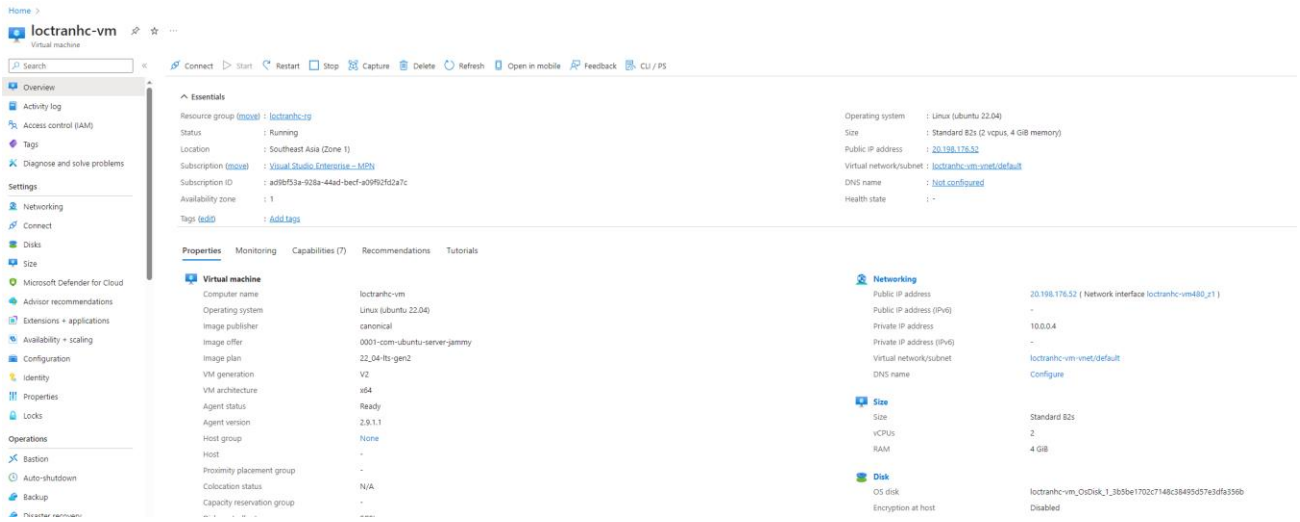
Assignment Steps

Azure is used as a cloud provider to perform this assignment.

Stage 1: Create infrastructure

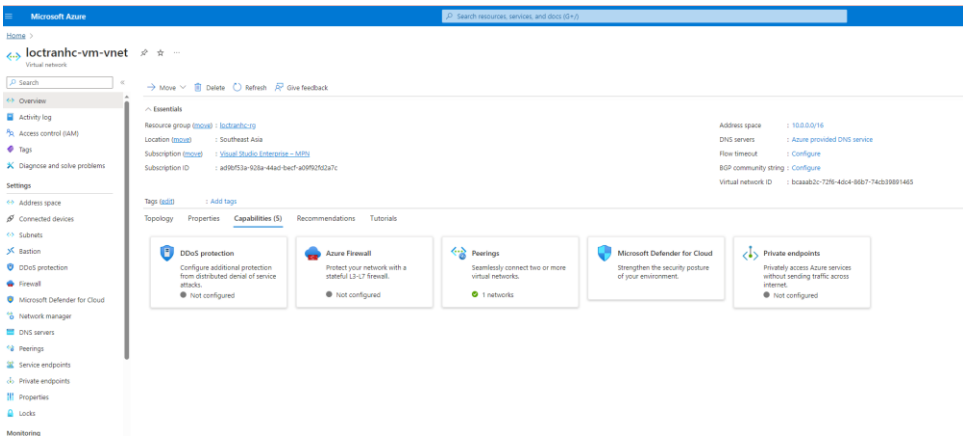
1. Set up a VM with:

Region	Southeast Asia
Image	Ubuntu Server 22.04
Size	Standard B2s

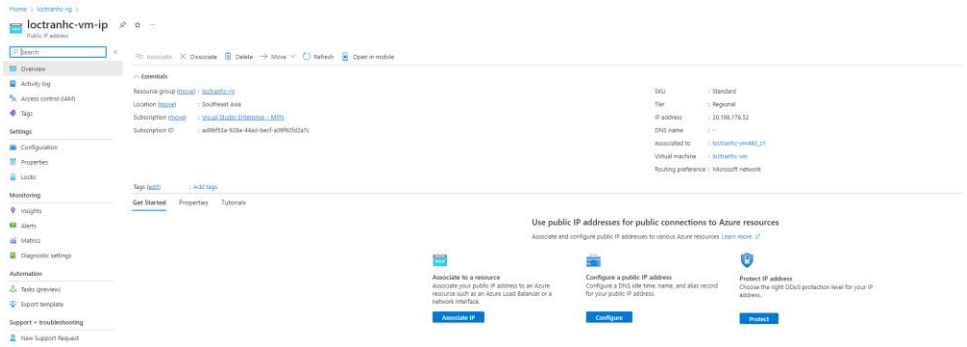


When creating VM, I also create:

- Vnet for VM:



- Public Ip Address:



2. Install necessary tool on VM

Docker:

Ref to this link [Install Docker]() for install docker

```
loctranhc@loctranhc-vm:~$ docker --version
Docker version 24.0.5, build ced0996
```

```
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2023-08-19 14:27:10 UTC; 19h ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 1095 (dockerd)
      Tasks: 14
     Memory: 529.3M
        CPU: 1min 58.278s
     CGroup: /system.slice/docker.service
             └─1095 /usr/bin/dockerd -H fd:// -H unix:///var/run/docker.sock -H tcp://0.0.0.0:2375 --containerd=/run/containerd/containerd.sock

Aug 20 06:33:48 loctranhc-vm dockerd[1095]: time="2023-08-20T06:33:48.869587193Z" level=warning msg="no trace recorder found, skipping"
Aug 20 06:33:51 loctranhc-vm dockerd[1095]: time="2023-08-20T06:33:51.414573819Z" level=warning msg="failed to prune image acrloctranhc.azurecr.io/frontende
Aug 20 06:33:54 loctranhc-vm dockerd[1095]: time="2023-08-20T06:33:54.519985415Z" level=warning msg="no trace recorder found, skipping"
Aug 20 06:33:58 loctranhc-vm dockerd[1095]: time="2023-08-20T06:33:58.827305949Z" level=warning msg="failed to prune image acrloctranhc.azurecr.io/backend@s
Aug 20 06:34:57 loctranhc-vm dockerd[1095]: time="2023-08-20T06:34:57.680202795Z" level=warning msg="no trace recorder found, skipping"
Aug 20 06:35:00 loctranhc-vm dockerd[1095]: time="2023-08-20T06:35:00.779967251Z" level=warning msg="failed to prune image acrloctranhc.azurecr.io/frontende
Aug 20 06:35:06 loctranhc-vm dockerd[1095]: time="2023-08-20T06:35:06.578596815Z" level=warning msg="no trace recorder found, skipping"
Aug 20 06:35:11 loctranhc-vm dockerd[1095]: time="2023-08-20T06:35:11.383682105Z" level=warning msg="failed to prune image acrloctranhc.azurecr.io/backend@s
Aug 20 07:52:22 loctranhc-vm dockerd[1095]: time="2023-08-20T07:52:22.269021618Z" level=warning msg="no trace recorder found, skipping"
Aug 20 07:52:24 loctranhc-vm dockerd[1095]: time="2023-08-20T07:52:24.893641701Z" level=warning msg="failed to prune image acrloctranhc.azurecr.io/frontende
~
```

Jenkins:

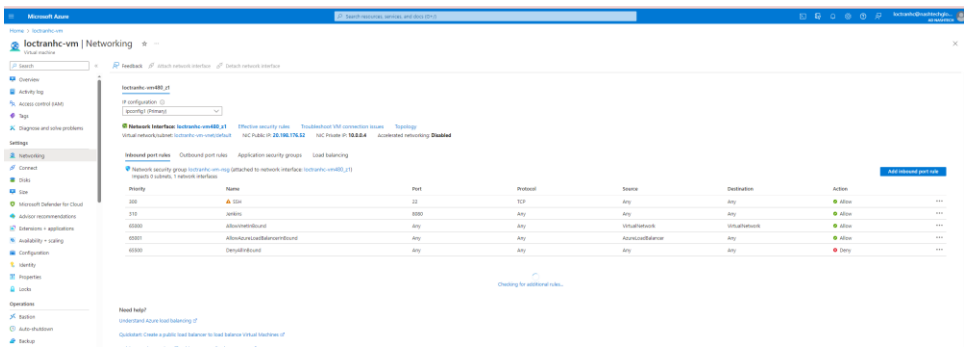
Ref to this link [Install Jenkins]() for install Jenkins

```
loctranhc@loctranhc-vm:~$ jenkins --version
2.401.3
```

```
loctranhc@loctranhc-vm:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2023-08-20 04:20:34 UTC; 5h 26min ago
   Main PID: 35727 (java)
      Tasks: 59 (limit: 4685)
     Memory: 1.4G
        CPU: 3min 35.477s
     CGroup: /system.slice/jenkins.service
             └─35727 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Aug 20 06:04:17 loctranhc-vm jenkins[35727]: 2023-08-20 06:04:17.367+0000 [id=1165]        INFO        o.j.p.g.webhook.WebhookManager$1#run: GitHub webhooks
Aug 20 06:04:17 loctranhc-vm jenkins[35727]: 2023-08-20 06:04:17.938+0000 [id=17]        WARNING        o.j.p.g.GitHubOrgWebHook#register: Failed to registe
Aug 20 06:04:19 loctranhc-vm jenkins[35727]: 2023-08-20 06:04:19.918+0000 [id=1174]        INFO        j.b.OrganizationFolder$OrganizationScan#run: sd4257-m
Aug 20 06:04:49 loctranhc-vm jenkins[35727]: 2023-08-20 06:04:49.381+0000 [id=1177]        INFO        j.b.OrganizationFolder$OrganizationScan#run: sd4257-m
Aug 20 06:07:37 loctranhc-vm jenkins[35727]: 2023-08-20 06:07:37.258+0000 [id=1192]        INFO        o.j.p.g.webhook.WebhookManager$1#run: GitHub webhooks
Aug 20 06:07:37 loctranhc-vm jenkins[35727]: 2023-08-20 06:07:37.571+0000 [id=1118]        WARNING        o.j.p.g.GitHubOrgWebHook#register: Failed to regis
Aug 20 06:07:39 loctranhc-vm jenkins[35727]: 2023-08-20 06:07:39.420+0000 [id=1197]        INFO        j.b.OrganizationFolder$OrganizationScan#run: sd4257-m
Aug 20 06:10:45 loctranhc-vm jenkins[35727]: 2023-08-20 06:10:45.339+0000 [id=1235]        WARNING        o.j.p.w.flow.FlowExecutionList#unregister: Owner[s
Aug 20 06:11:12 loctranhc-vm jenkins[35727]: 2023-08-20 06:11:12.121+0000 [id=1274]        WARNING        o.j.p.w.flow.FlowExecutionList#unregister: Owner[s
Aug 20 06:30:00 loctranhc-vm jenkins[35727]: 2023-08-20 06:30:00.728+0000 [id=109]        INFO        h.triggers.SCMTrigger$Runner#run: SCM changes detected
lines 1-20/20 (END)
```

To access Jenkins, I need to expose port = 8080 at tab Networking of VM. At here, we create inbound port rule with Destination port ranges = 8080.



Kubectl:

Run “snap install kubectl –classic" to install kubectl

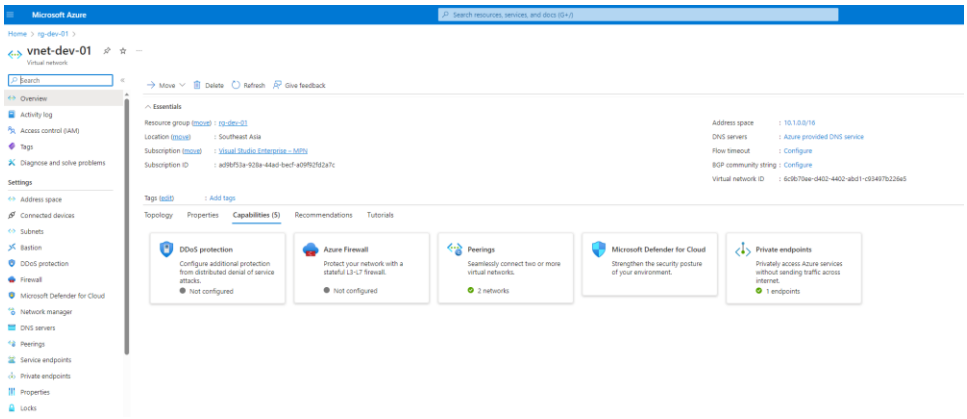
```
loctranhc@loctranhc-vm:~$ kubectl version
Client Version: v1.28.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Unable to connect to the server: dial tcp: lookup aksloctranhc-wga7f6ts.hcp.southeastasia.azmk8s.io on 127.0.0.53:53: no such host
```

3. Provision with terraform

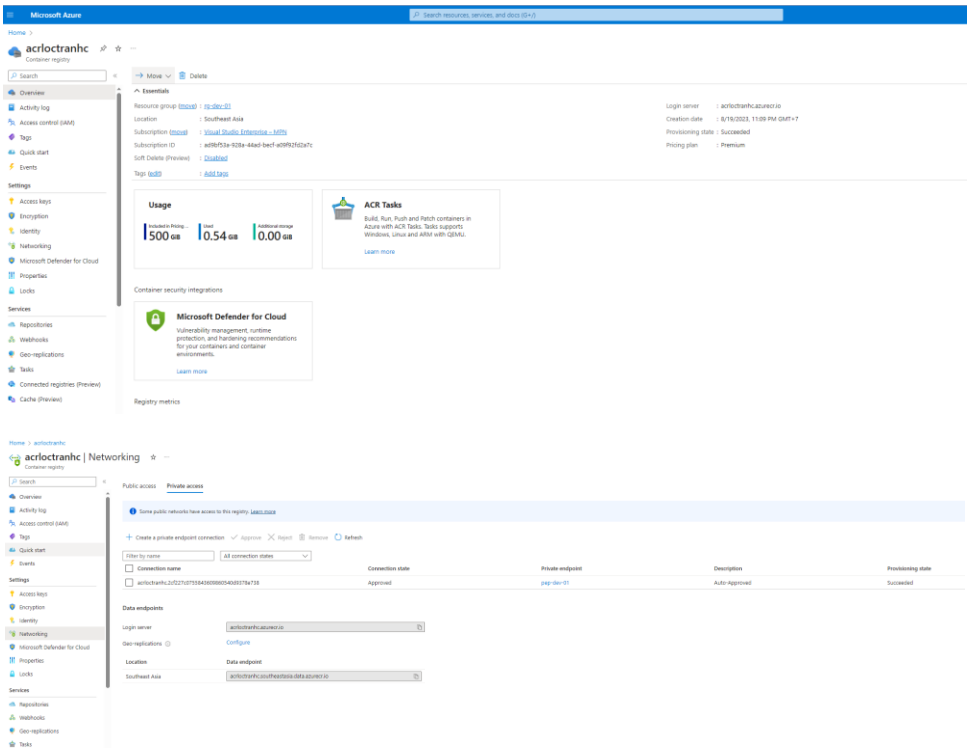
Please ref this repo here: [sd4257_azure_infrastructure]() to more detail about structure.

Basically, I created:

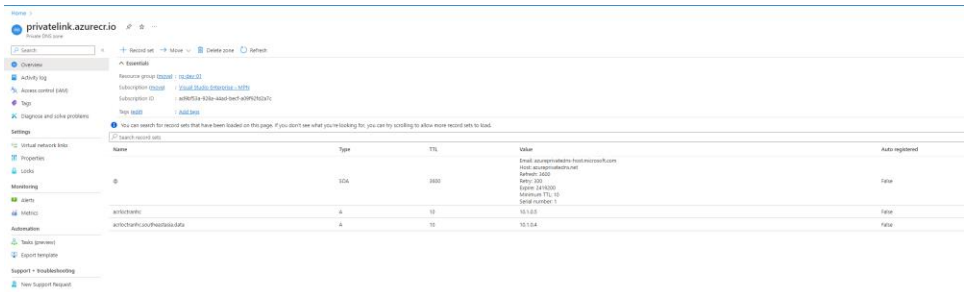
VNet for Azure Container Registry



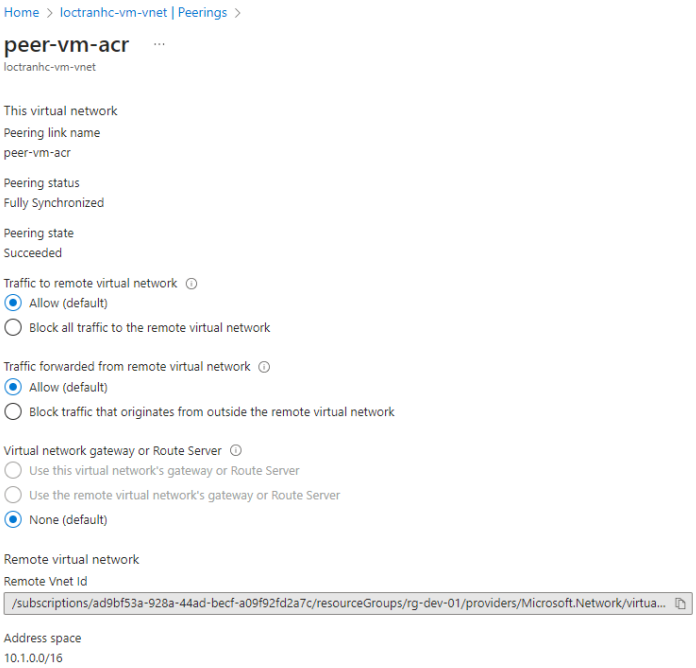
Azure Container Registry with private endpoint



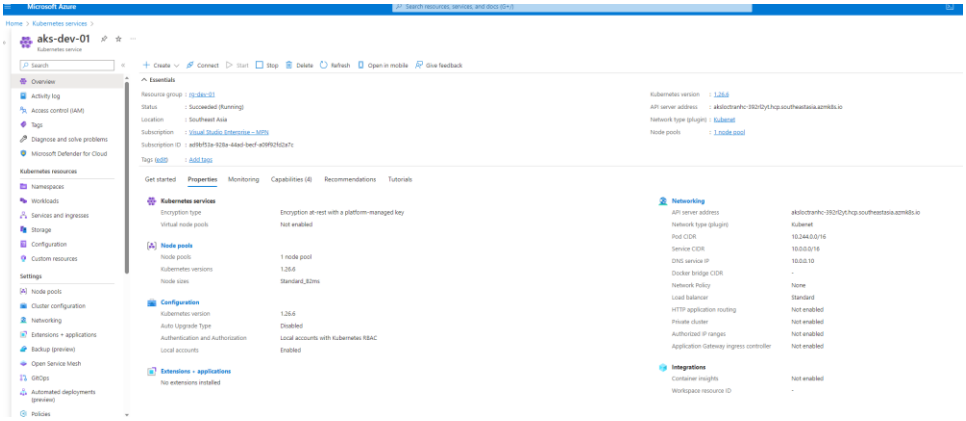
- I also created Private DNS Zone to able access registry from VM at the step 1.



- However, I need peering VM vnet and ACR vnet, so that they communicate together.

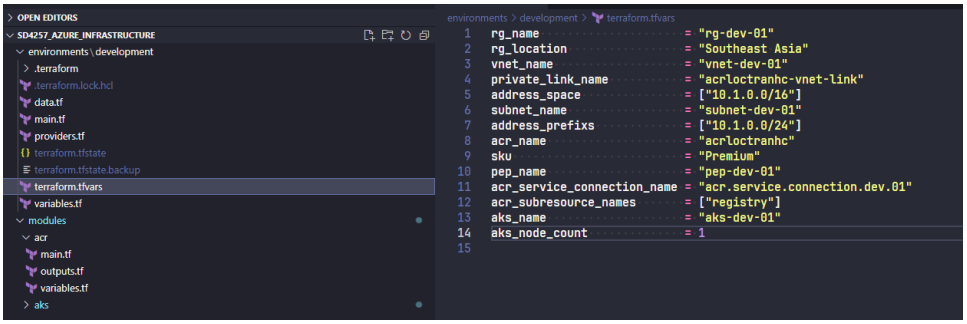


Azure Kubernetes services



Note

- To apply terraform to azure cloud, I need create a file *.tfvars like this:



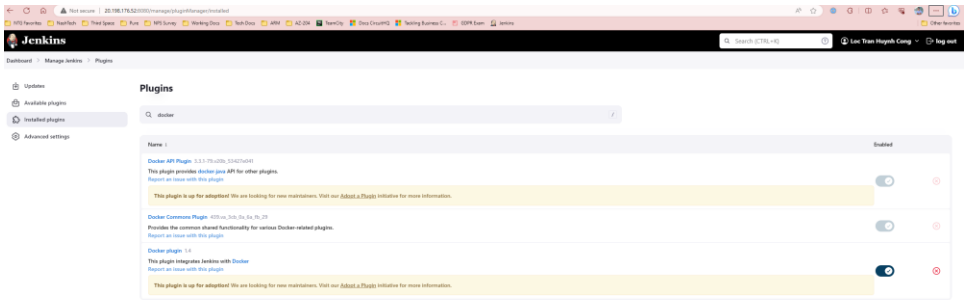
And run command “terraform apply” at /environments/development

Stage 2: Set up Jenkins

With the first time when access to Jenkins, I run this command to get initial admin password

“sudo cat /var/lib/jenkins/secrets/initialAdminPassword”

After that, I installed plugins like the bellow image.

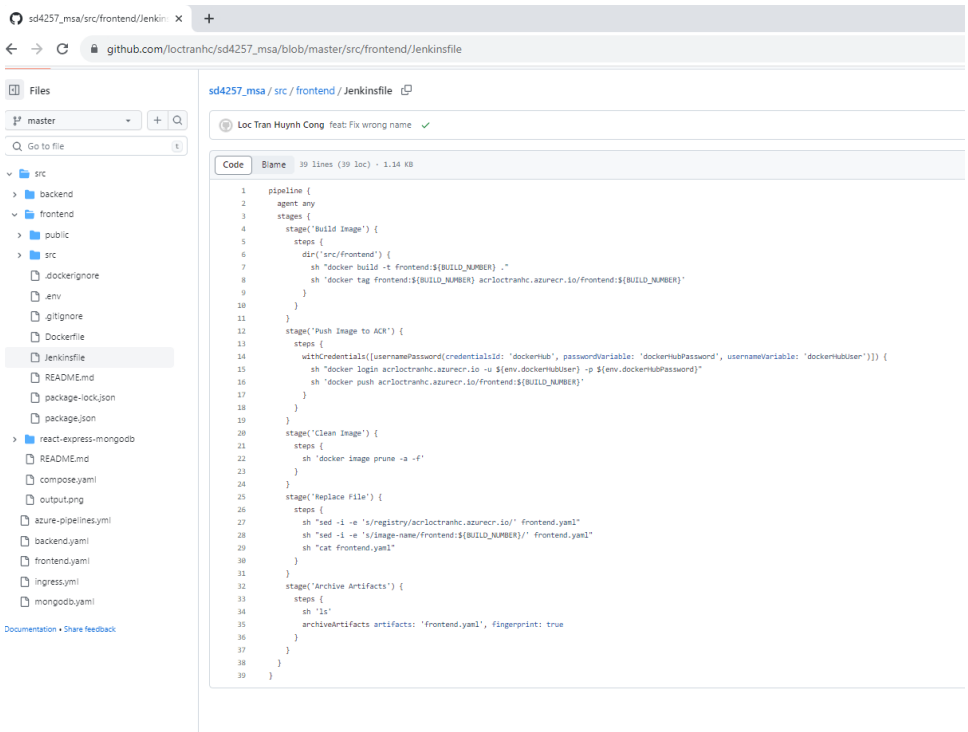


Stage 3: Set up CI

I used Jenkins to handle CI for *frontend* and Azure DevOps handle CI for *backend.*

Jenkins

I create a Jenkinsfile at /src/frontend like this below image. Alternatively, check direct file at here: Jenkinsfile



1. Create credential

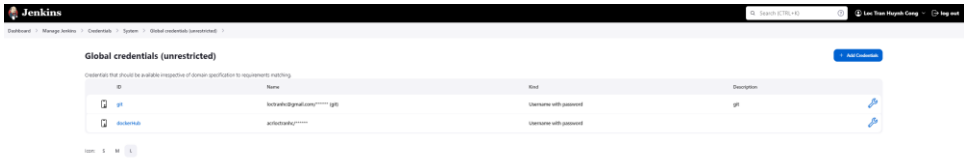
Go to Manage Jenkins > Credentials > System > Global credentials (unrestricted) to add Credentials I created credentials for:

Git:

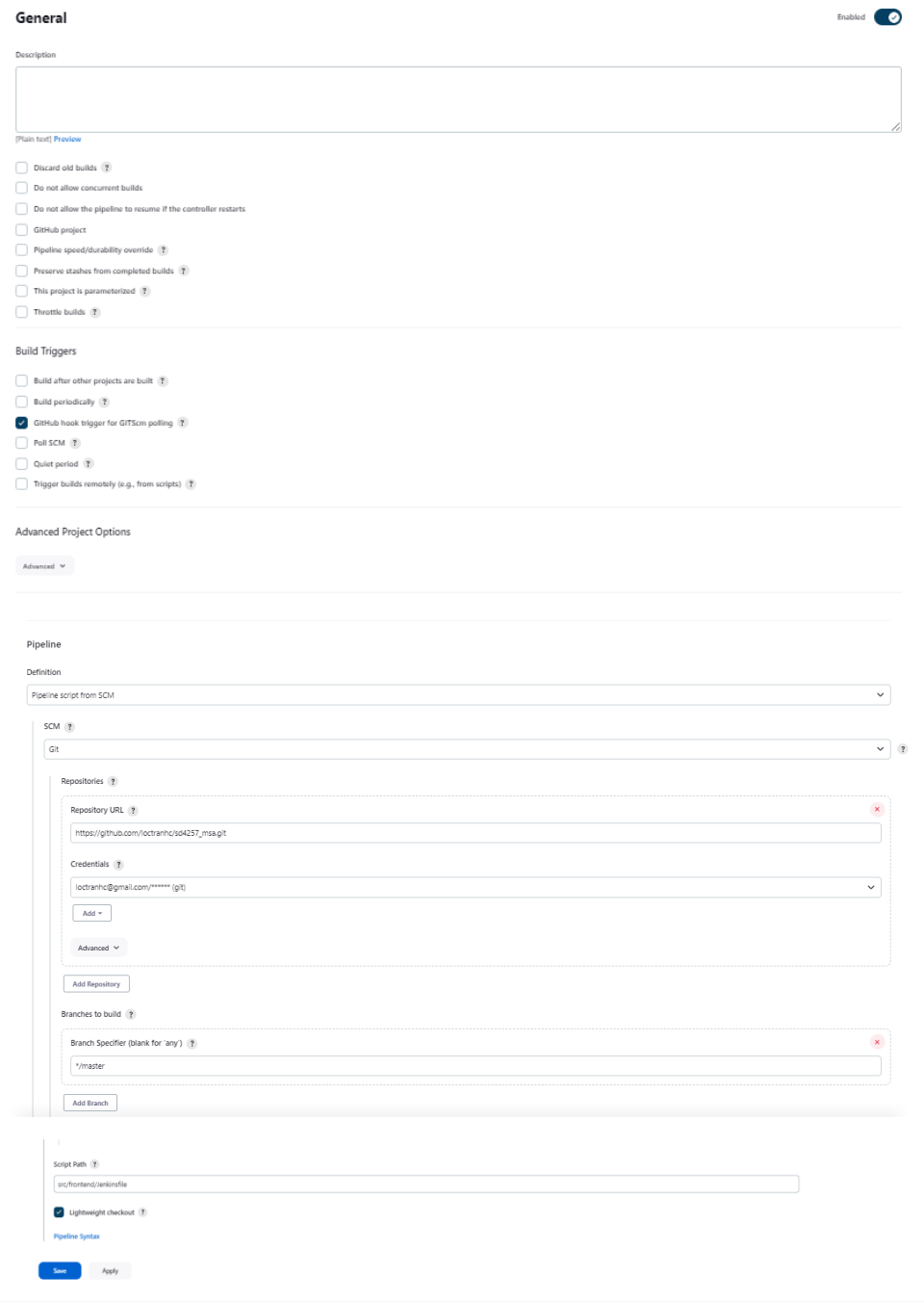
- Trigger build when have new commit.
- Check out source to build image.

Docker:

- Login to docker registry and push image



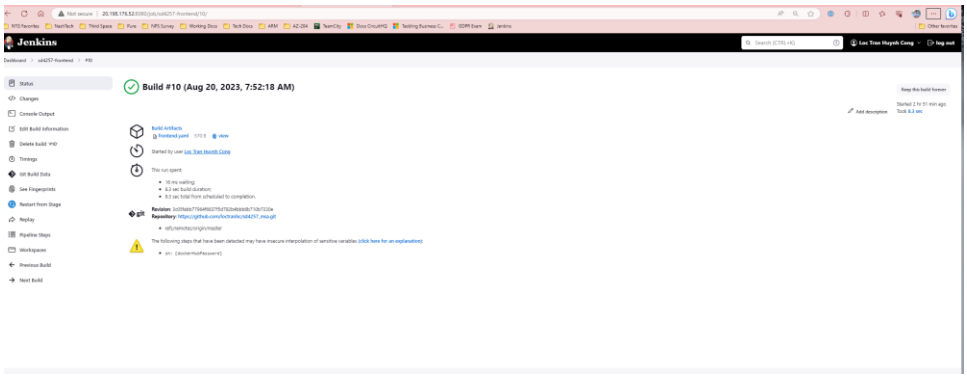
2. Create pipeline with config like this



Note:

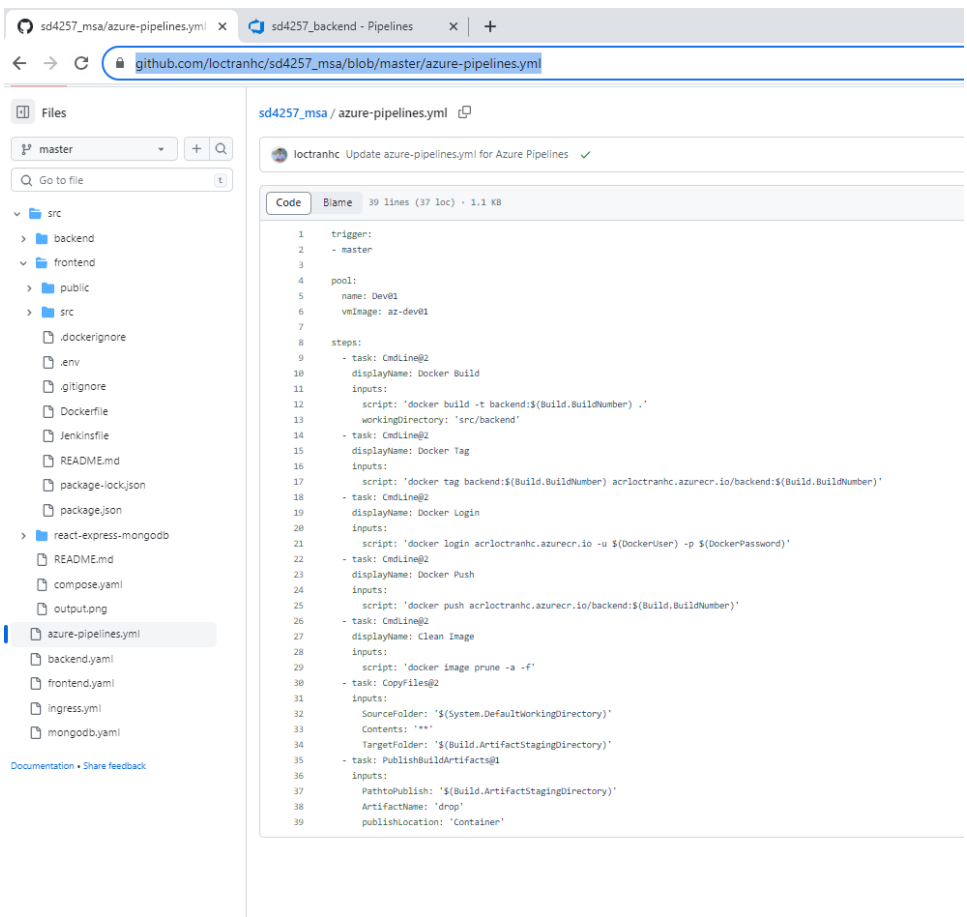
- At tab Credentials, select the git credential which I created at step 1.
- Script Path is filled for define the path to frontend Jenkinsfile.

3.After build and push image, I replace the build version and publish frontend.yaml as artifacts for CD reused.

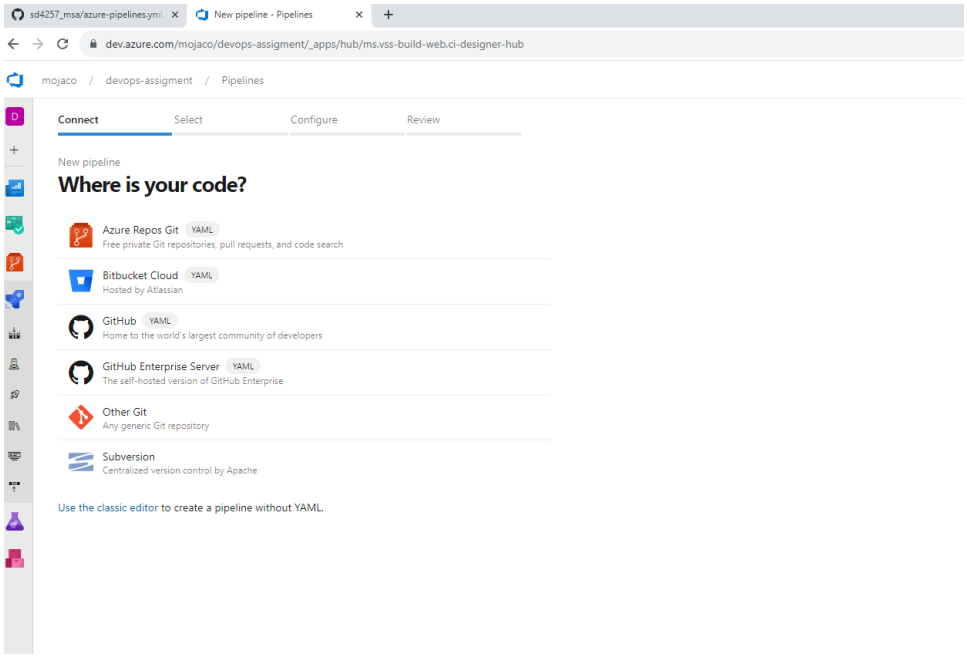


Azure DevOps

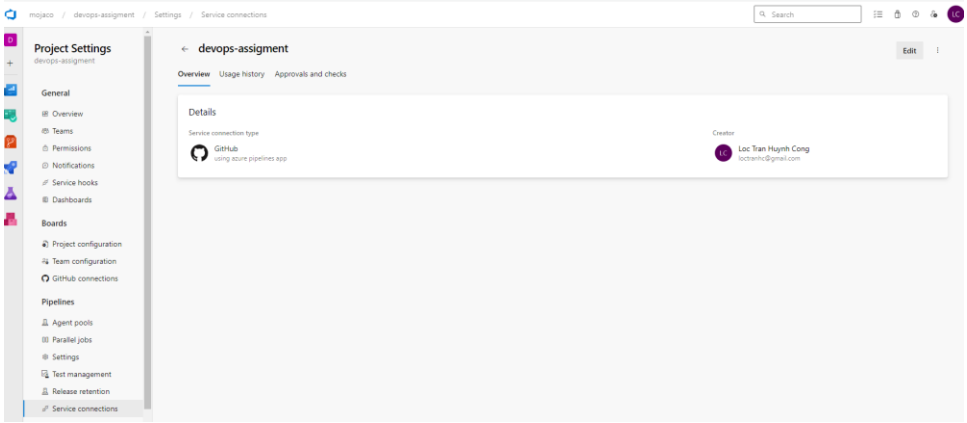
I created azure-pipelines.yml file at root of sd4257_msa repo. Please check direct file at here: [azure-pipelines.yml]



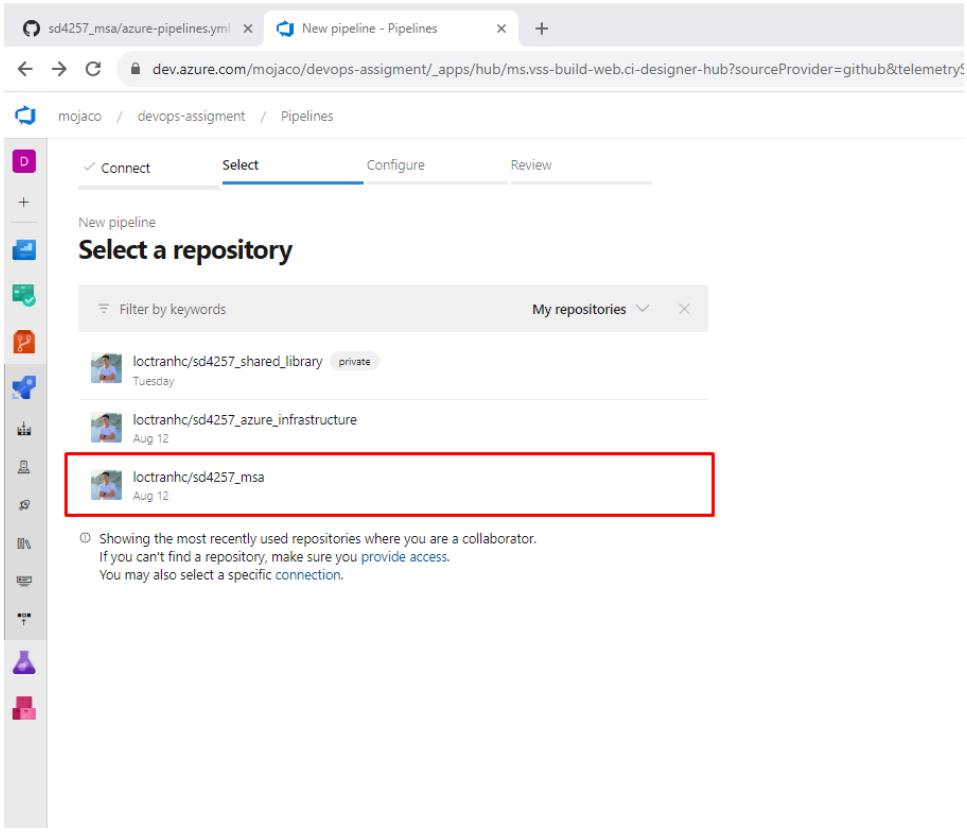
Next, I connect with git repo. Select GitHub and login my git account.

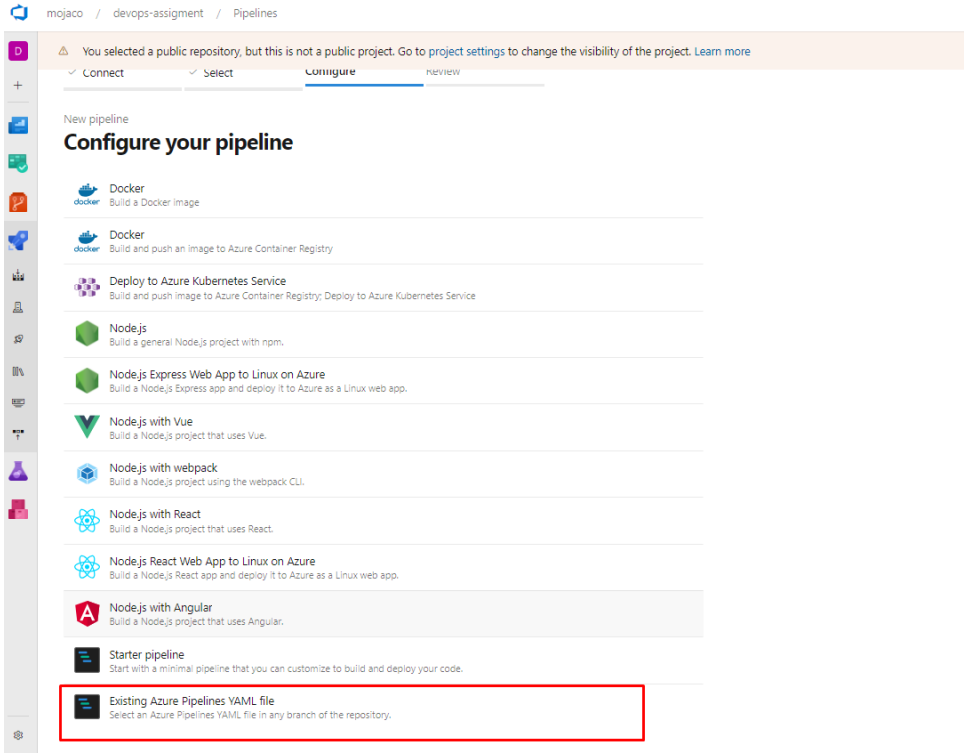


After login success, Azure created a service connection and I was able to check this like the below image.

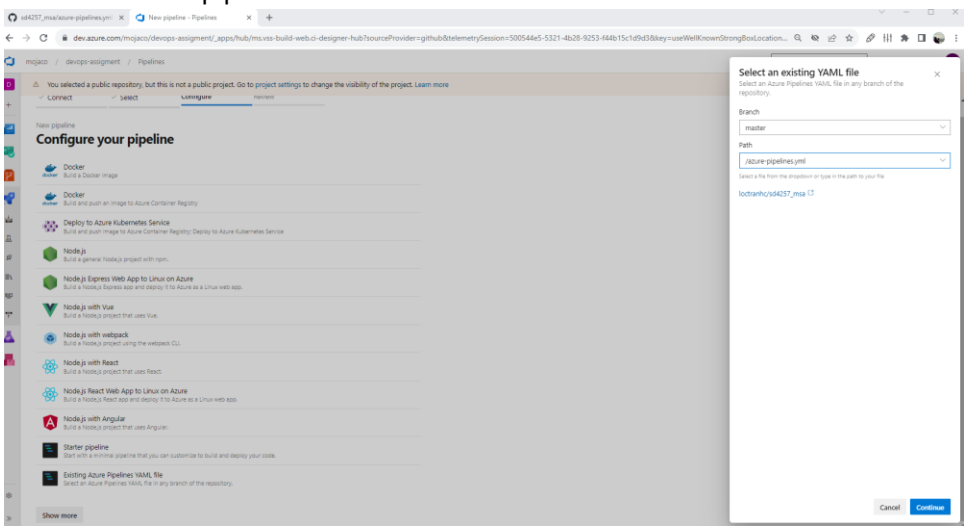


Select the sd4257_msa repo, and using the existing pipeline file.



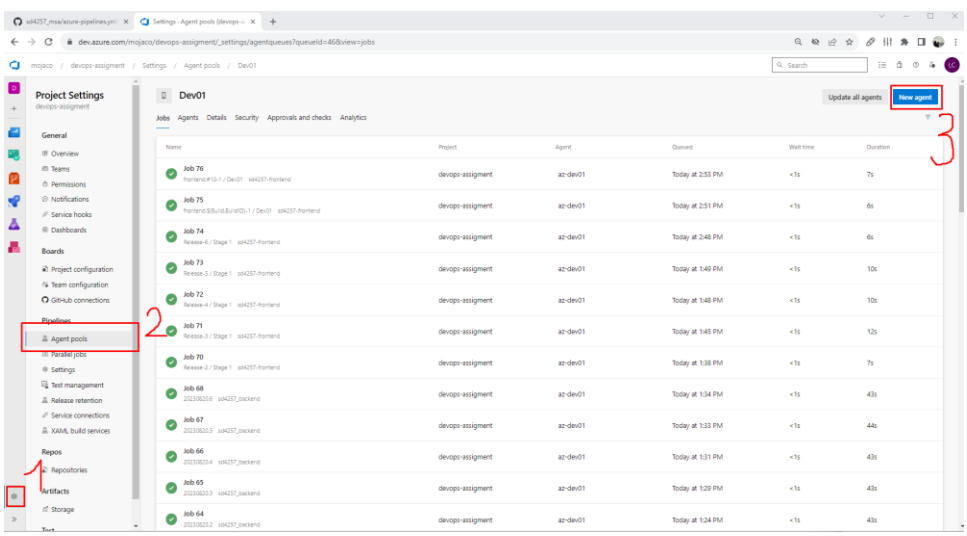


Select the Branch and azure-pipelines file that I want to use.

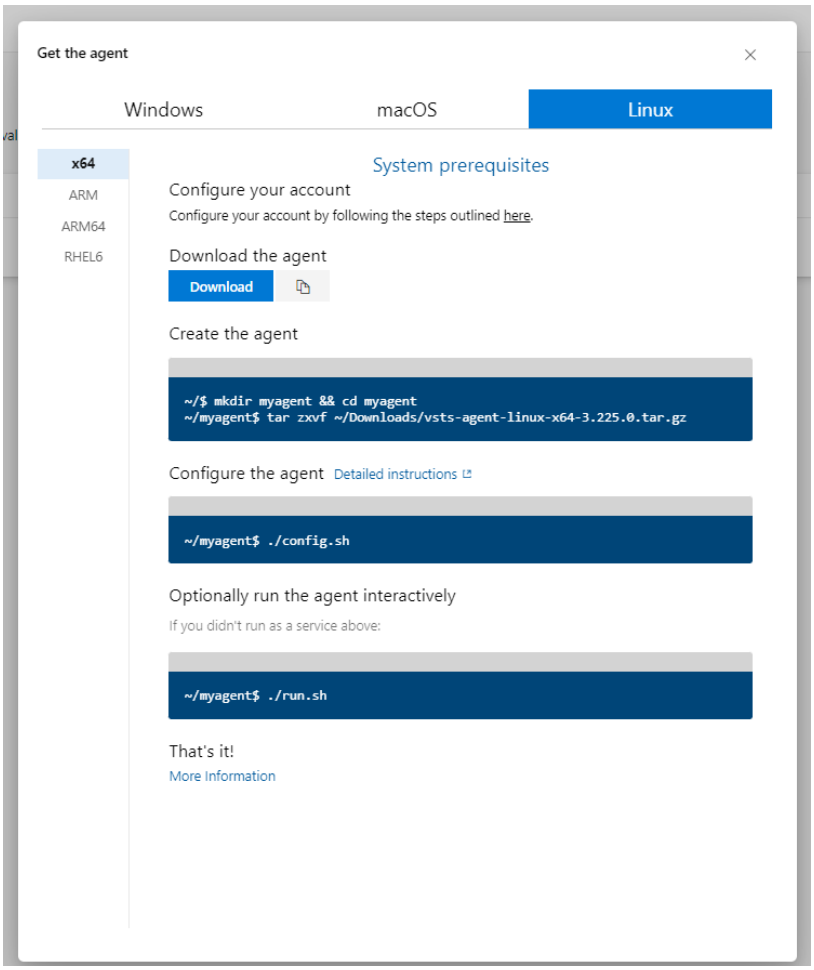


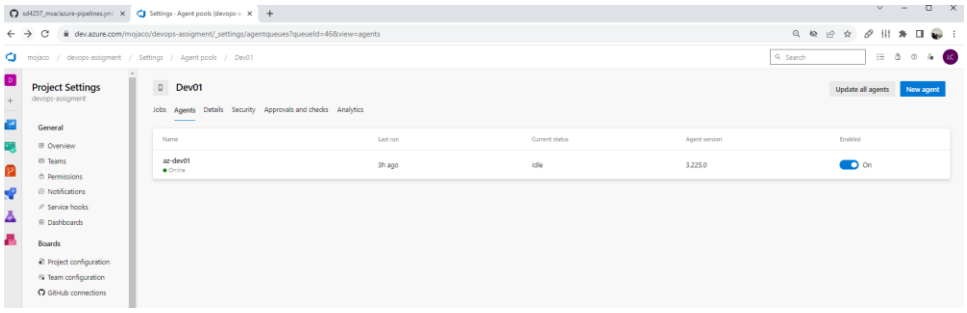
Note:

- I reused and setup VM at stage 1 for agent run CI and CD in Azure DevOps. When setting up agent, I need to create a Personal Token because I choose PAT authentication option when running script setup azure devops agent.
    - First, I click new agent at project settings like this image

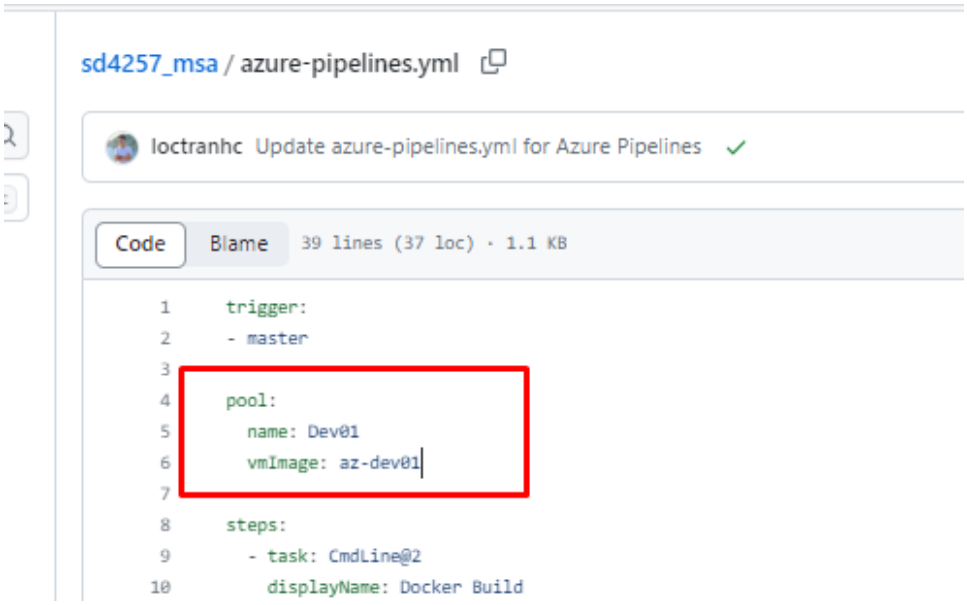


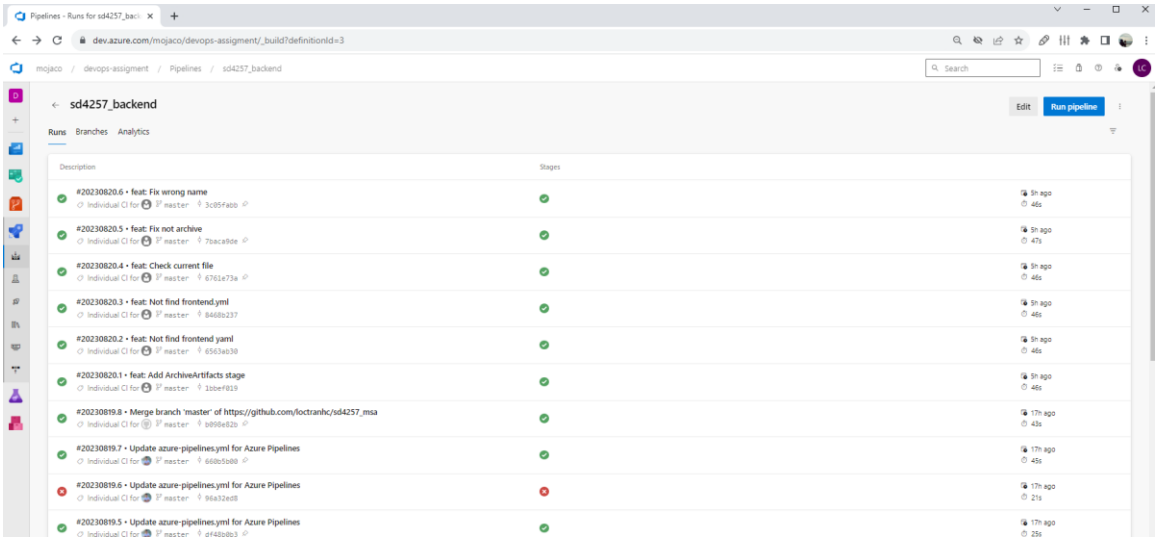
- And select the operating system of VM that I want to use as an agent. Follow this step in the below image to install.



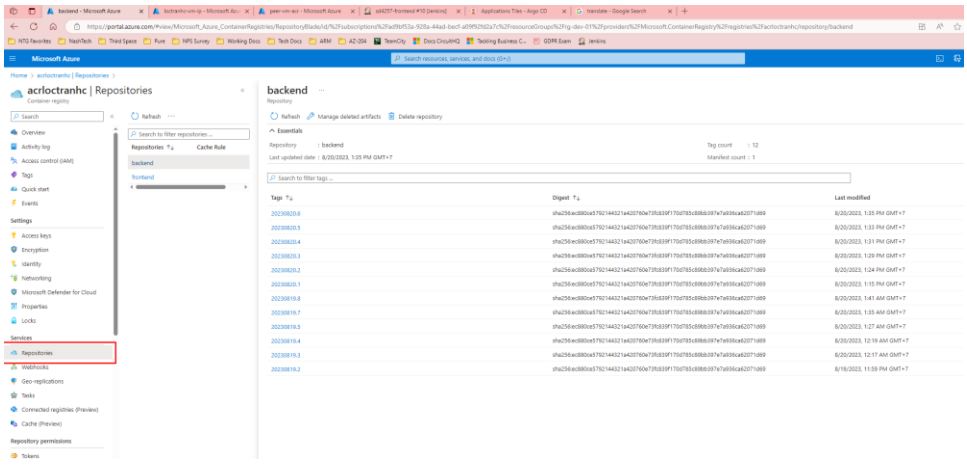


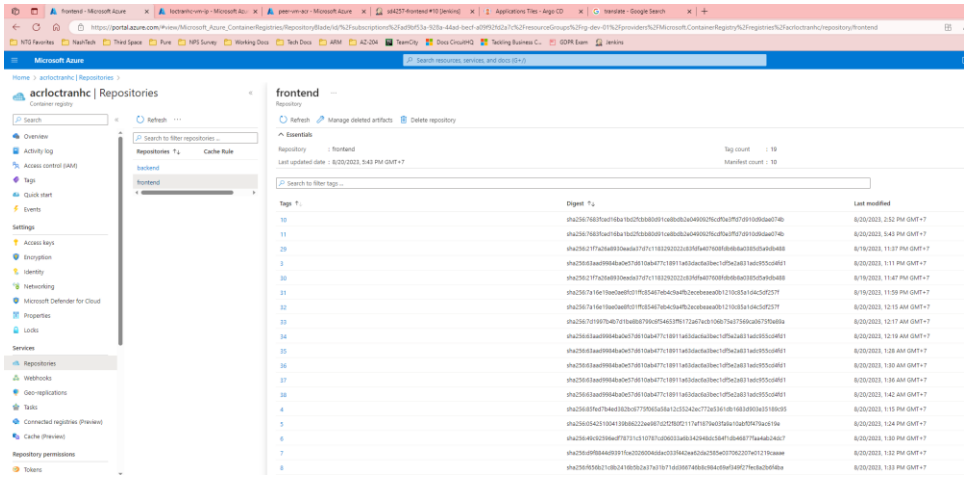
- After complete setup the self-host agent, I need to indicate at azure-pipeline files like this:





When CI runs successfully, we can check if the image has been pushed to the container registry by



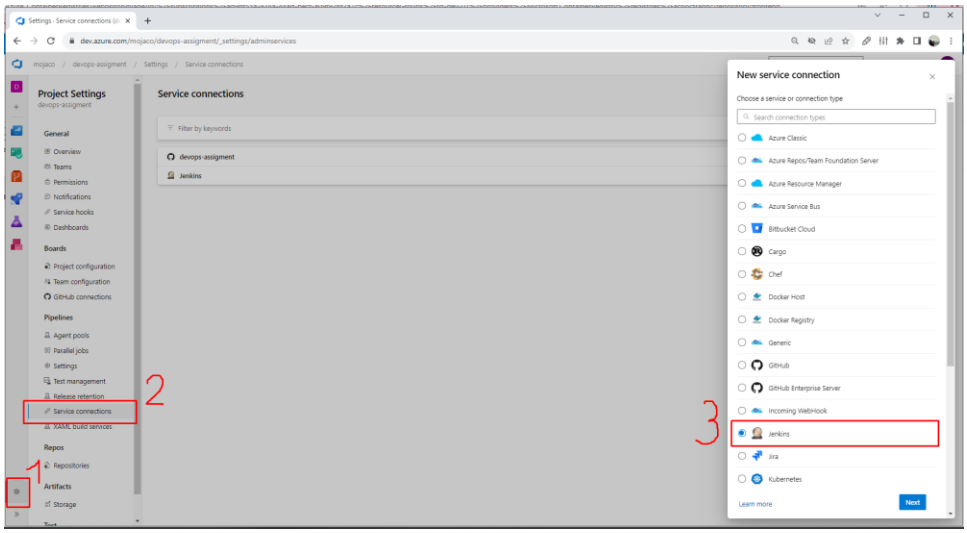


Stage 4: Set up CD

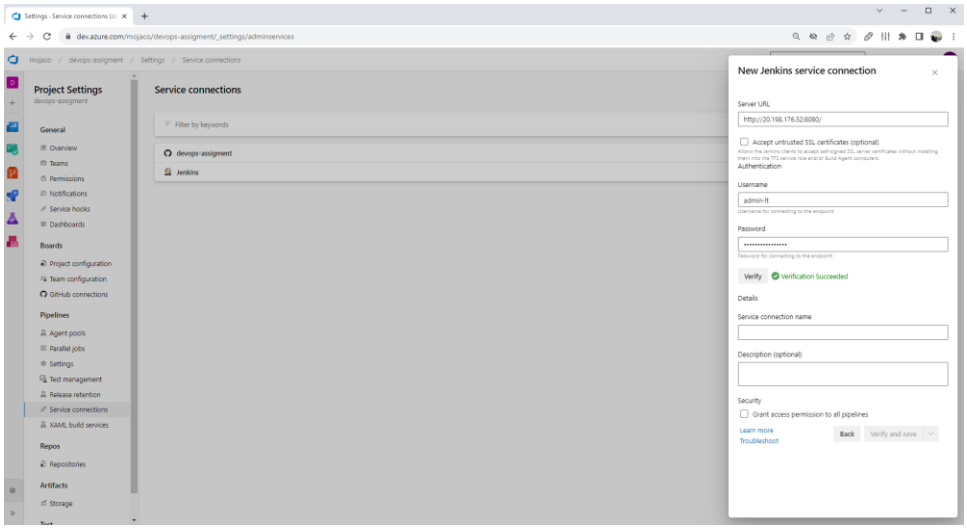
Azure DevOps

For frontend

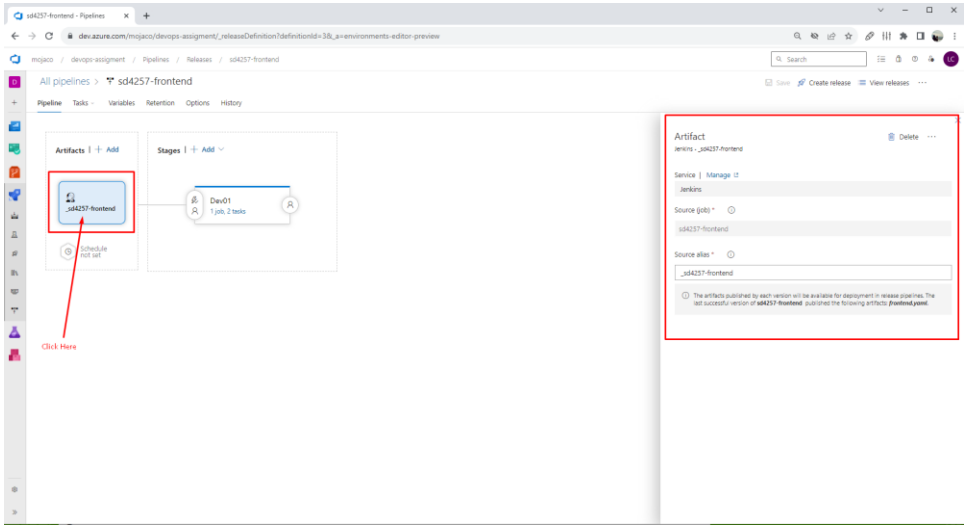
- As the below Jenkins CI step, I published frontend.yaml as an artifact.
- First I need create a service connection to Jenkins. To create, ref the below images.



Fill config my jenkins like the below image:

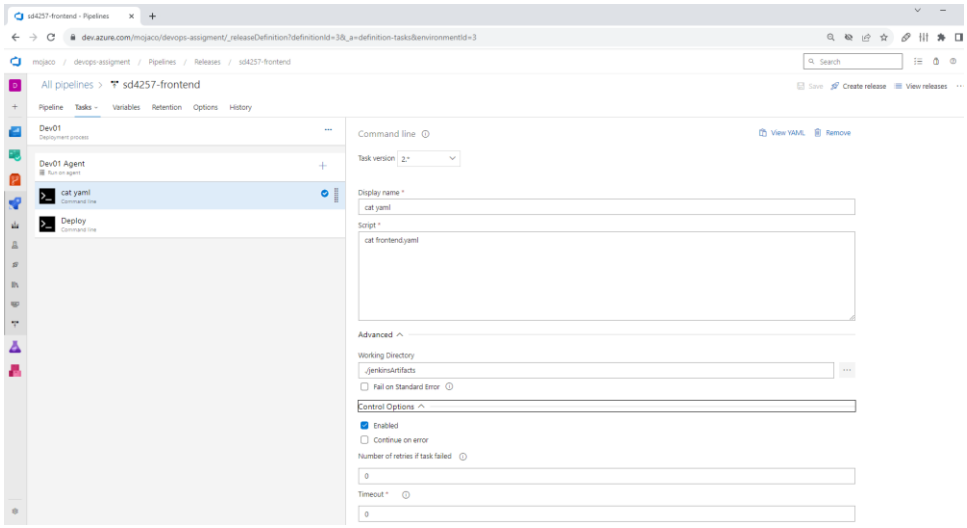


Create artifact for release pipeline, I choose my jenkins like this image

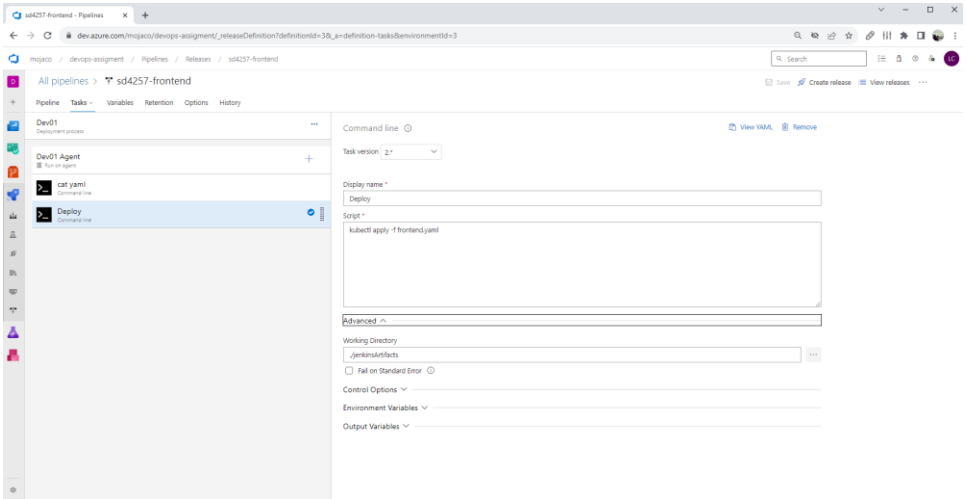


I set up 1 stage with 2 tasks

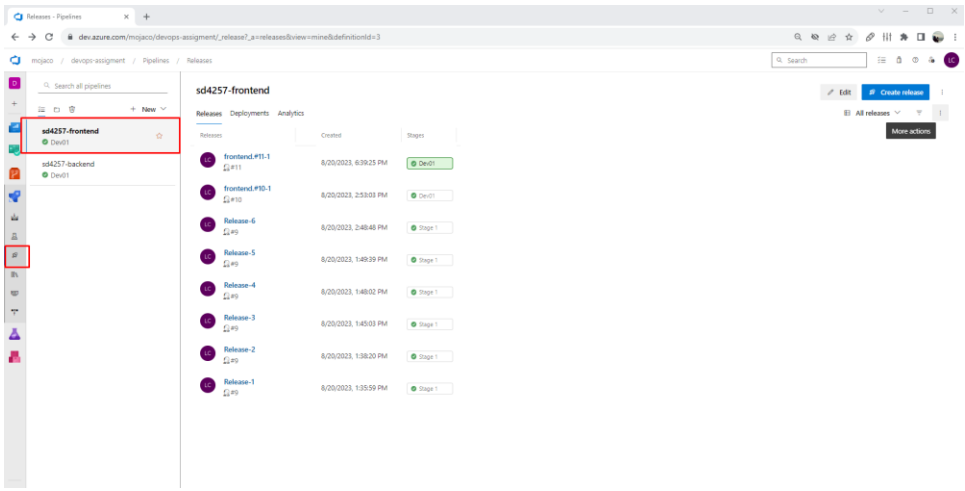
Task 1: Print content frontend.yaml to log. Check issue if frontend.yaml file is wrong.



Task 2: Apply frontend.yaml



I able to check release here:

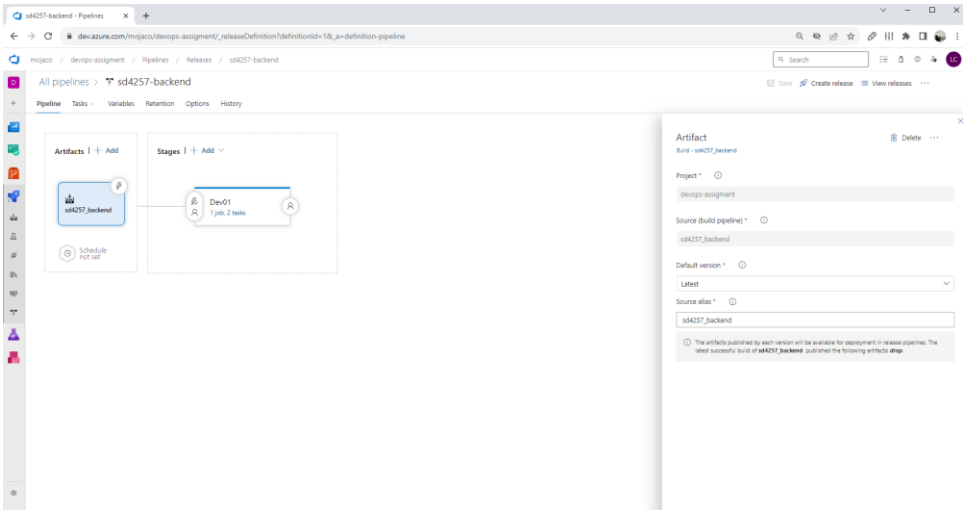


And check on cluster

```
loctranhc@loctranhc-vm:~$ kubectl get svc frontend
NAME       TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
frontend   ClusterIP   10.0.123.86   <none>        3000/TCP  16m
```

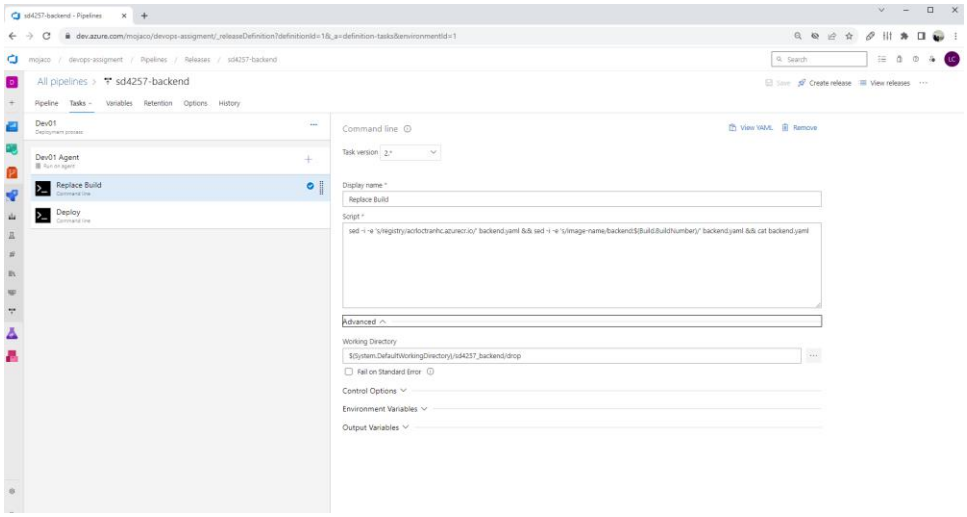
For backend

Using the artifact from azure devops pipeline for backend

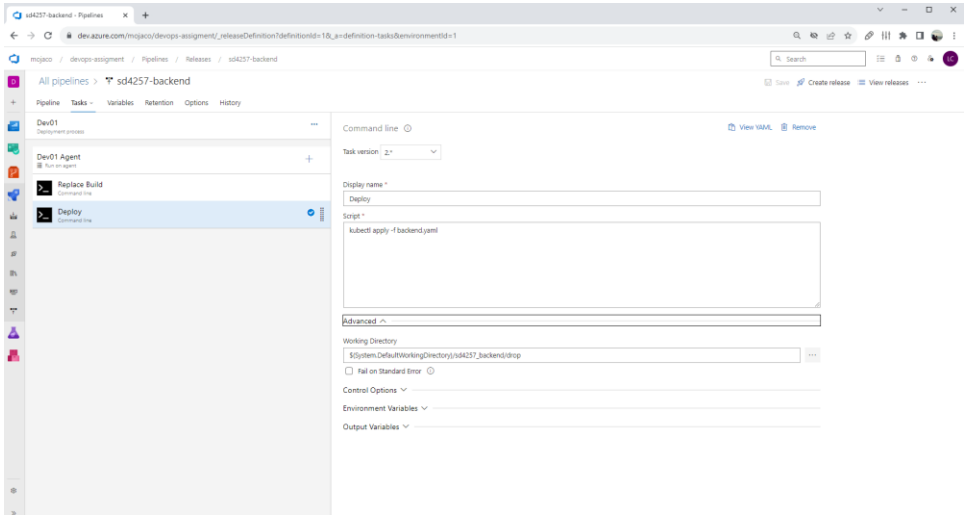


I set up 1 stage with 2 tasks

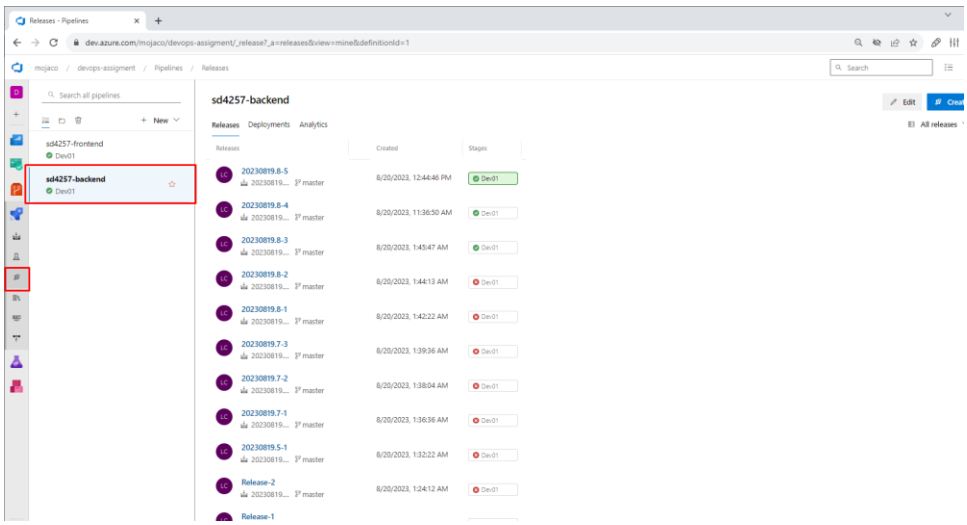
Task 1: Replace the image name of backend with current build from azure artifact.



Task 2: Apply backend.yaml



Check release here:



And check on cluster

```
loctranhc@loctranhc-vm:~$ kubectl get svc backend
NAME       TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
backend    ClusterIP   10.0.46.112   <none>        3000/TCP  5m13s
```

ArgoCD

1. Install ArgoCD into the current cluster.
   - Create namespace for argocd

```
loctranhc@loctranhc-vm:~$ kubectl get namespace
NAME              STATUS   AGE
argocd            Active   103m
default           Active   155m
kube-node-lease   Active   155m
kube-public       Active   155m
kube-system       Active   155m
```

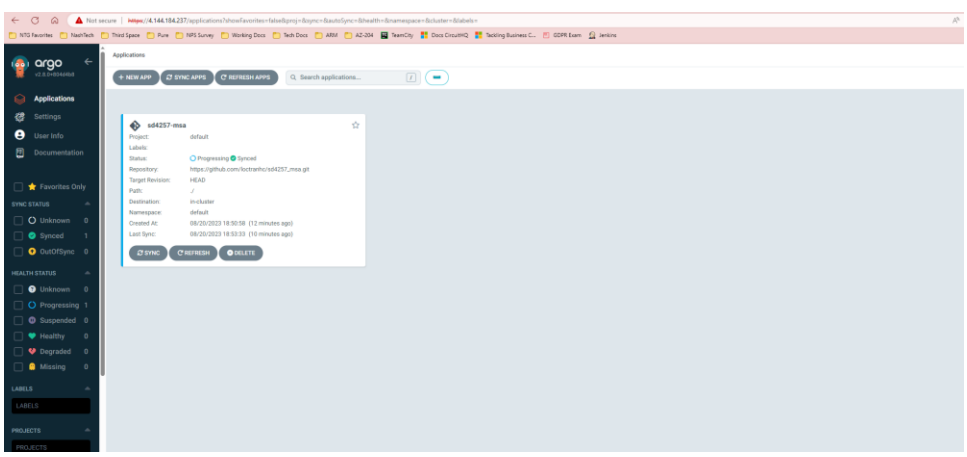
   - Run command “kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml” to deploy argocd application.
2. Login with initial admin password
   - Run command “kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d” to get initial admin password for login.
3. Expose argocd-server with node type = LoadBalancer by run command “kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'”

```
loctranhc@loctranhc-vm:~$ kubectl get svc -n argocd
NAME                                      TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGEargo
one>          7000/TCP,8080/TCP            107m
argocd-dex-server                         ClusterIP      10.0.141.179   <none>          5556/TCP,5557/TCP,5558/TCP   107m
argocd-metrics                            ClusterIP      10.0.190.11    <none>          8082/TCP                     107m
argocd-notifications-controller-metrics   ClusterIP      10.0.17.242    <none>          9001/TCP                     107m
argocd-redis                              ClusterIP      10.0.188.231   <none>          6379/TCP                     107m
argocd-repo-server                        ClusterIP      10.0.17.77     <none>          8081/TCP,8084/TCP            107m
argocd-server                             LoadBalancer   10.0.243.79    4.144.184.237   80:31285/TCP,443:30863/TCP   107m
argocd-server-metrics                     ClusterIP      10.0.137.189   <none>          8083/TCP                     107m
loctranhc@loctranhc-vm:~$
```

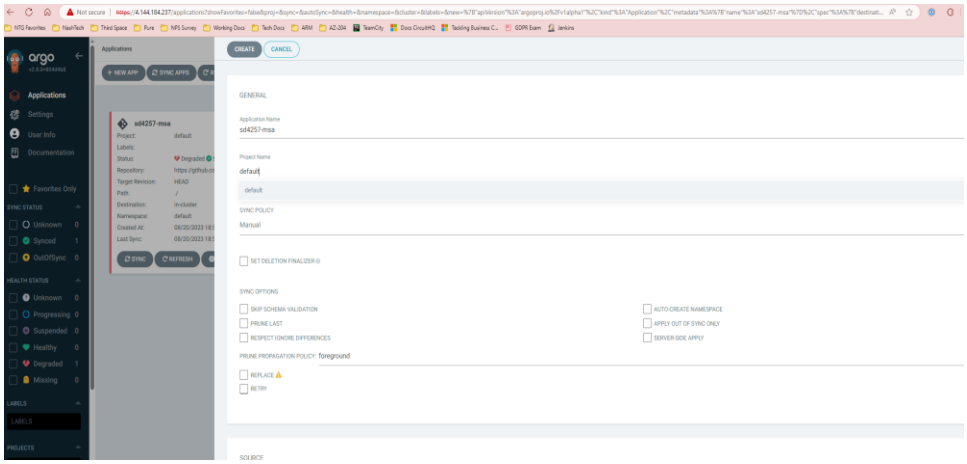
Login into ArgoCD with

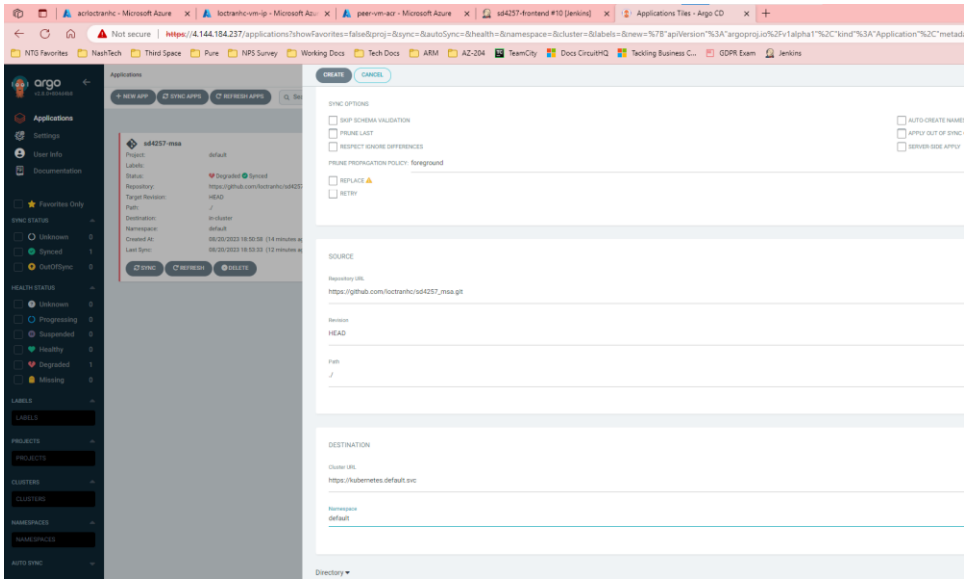
UserName: admin

Password: Get from step 2.

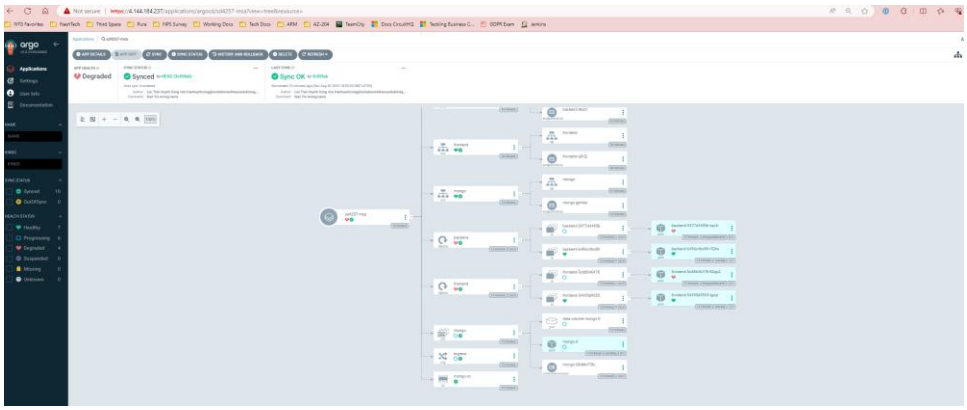


Create new app as the below image





After compled creation, I check the dashboard here



Stage 5: Setup Monitoring

I reused the AKS and VM to set up for this stage.

Firstly, I installed Helm, ref here [Helm Install]() to install

From Script

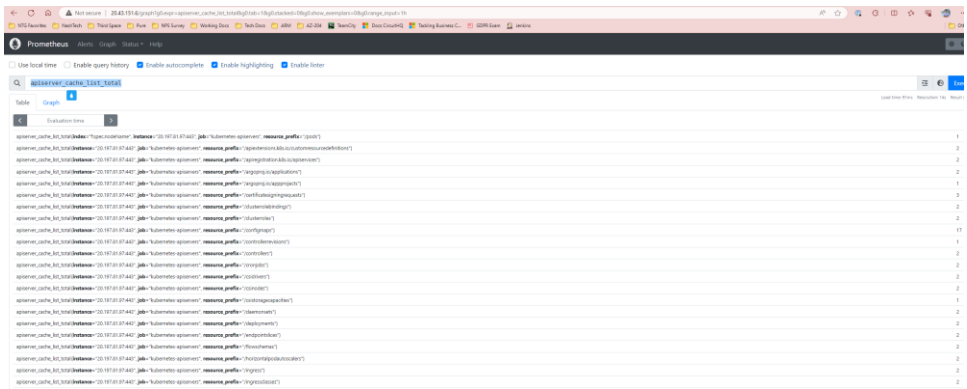
Helm now has an installer script that will automatically grab the latest version of Helm and install it locally .

You can fetch that script, and then execute it locally. It's well documented so that you can read through it and understand what it is doing before you run it.

```
$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
$ chmod 700 get_helm.sh
$ ./get_helm.sh
```

Install Prometheus

1. Add prometheus helm repo by run command: "helm repo add prometheus-community https://prometheus-community.github.io/helm-charts"
2. Install by run command "helm install prometheus prometheus-community/prometheus"
3. Expose prometheus-server with node type = "LoadBalancer"



Install Grafana

1. Add Grafana helm repo by run command: "helm repo add grafana https://grafana.github.io/helm-charts"
2. Install by run command "helm install grafana grafana/grafana"
3. Expose grafana with node type = "LoadBalancer"

After installed, I able to check grafana and prometheus on my cluster here

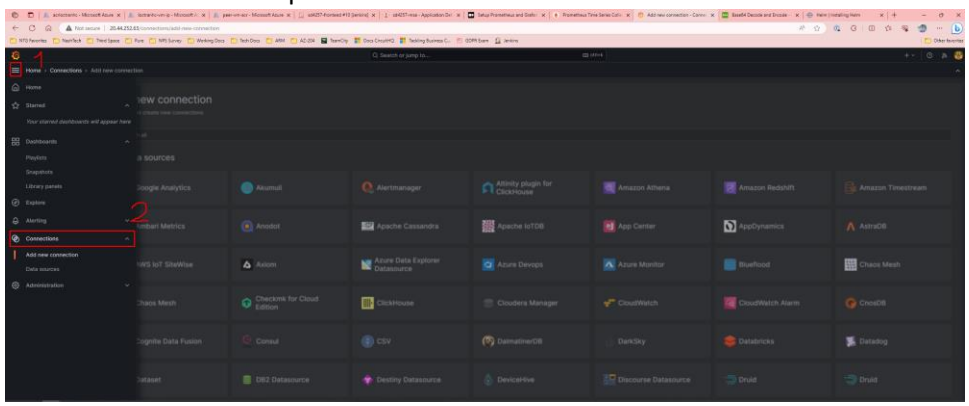
```
loctranhc@loctranhc-vm:~$ kubectl get svc -n default
NAME                                  TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)        AGE
backend                               ClusterIP      10.0.46.112    <none>         3000/TCP       66m
frontend                              ClusterIP      10.0.123.86    <none>         3000/TCP       77m
grafana                               LoadBalancer   10.0.191.47    20.44.252.63   80:30410/TCP   44m
kubernetes                            ClusterIP      10.0.0.1       <none>         443/TCP        3h34m
mongo                                 ClusterIP      10.0.232.105   <none>         27017/TCP      65m
prometheus-alertmanager               ClusterIP      10.0.21.148    <none>         9093/TCP       45m
prometheus-alertmanager-headless      ClusterIP      None           <none>         9093/TCP       45m
prometheus-kube-state-metrics         ClusterIP      10.0.172.195   <none>         8080/TCP       45m
prometheus-prometheus-node-exporter   ClusterIP      10.0.232.12    <none>         9100/TCP       45m
prometheus-prometheus-pushgateway     ClusterIP      10.0.139.41    <none>         9091/TCP       45m
prometheus-server                     LoadBalancer   10.0.155.247   20.43.151.6    80:32134/TCP   45m
loctranhc@loctranhc-vm:~$
```

Setup DashBoard on Grafana

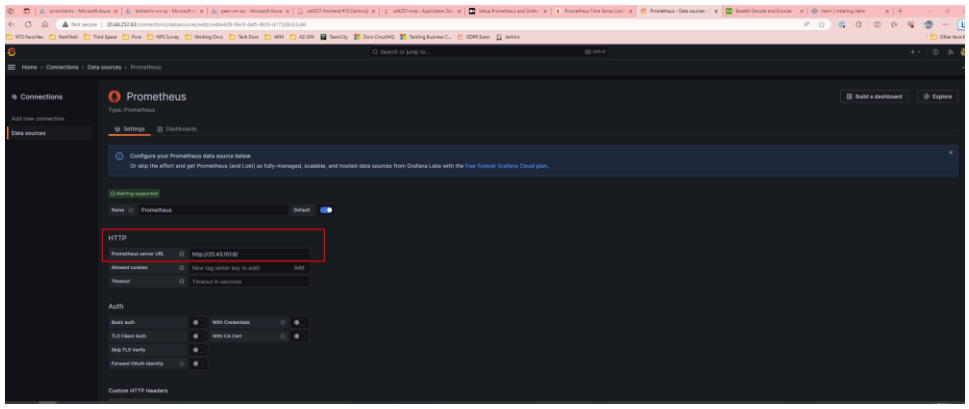
1. To login to Grafana, I need get account information by command: “kubectl get secret — namespace default grafana -o yaml”

```
loctranhc@loctranhc-vm:~$ kubectl get secret - namespace default grafana -o yaml
apiVersion: v1
items:
- apiVersion: v1
  data:
    admin-password: ZGVBZ1pkRGJ5SHJZdHZtY242SktpNkVKcXM5aXBGWTZURG5MR3luVQ==
    admin-user: YWRtaW4=
    ldap-toml: ""
  kind: Secret
  metadata:
    annotations:
      meta.helm.sh/release-name: grafana
      meta.helm.sh/release-namespace: default
    creationTimestamp: "2023-08-20T12:12:17Z"
    labels:
      app.kubernetes.io/instance: grafana
      app.kubernetes.io/managed-by: Helm
      app.kubernetes.io/name: grafana
      app.kubernetes.io/version: 10.0.3
      helm.sh/chart: grafana-6.58.9
    name: grafana
    namespace: default
    resourceVersion: "40509"
    uid: 5817740d-bad1-483f-95b5-404c99786985
  type: Opaque
kind: List
metadata:
  resourceVersion: ""
Error from server (NotFound): secrets "-" not found
Error from server (NotFound): secrets "namespace" not found
Error from server (NotFound): secrets "default" not found
```

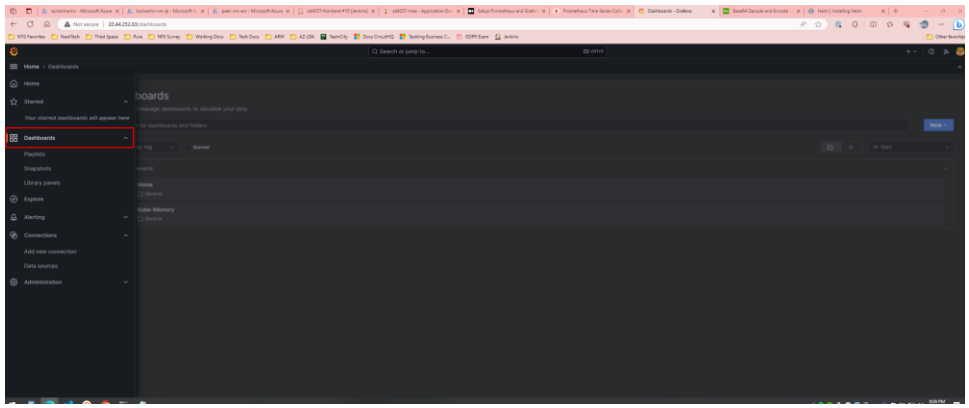
2. Take admin-user and admin-password and decode base64 to get exactly account login.
3. Add new connection to prometheus.



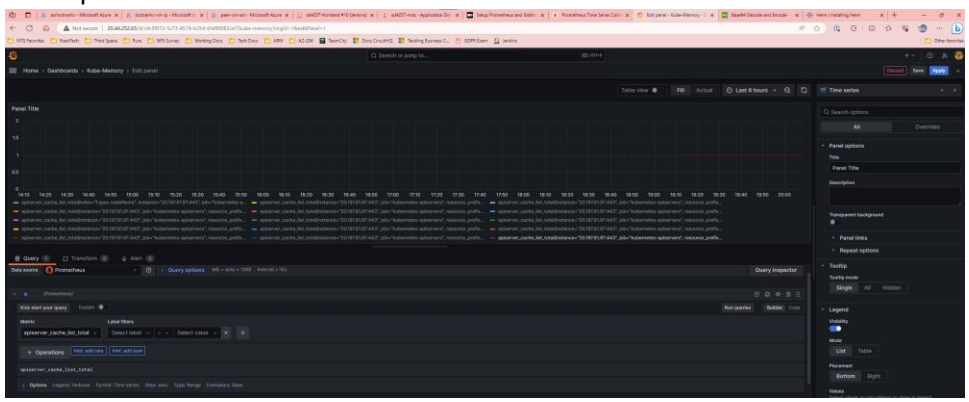
4. Set Prometheus Url server and click Save & Test

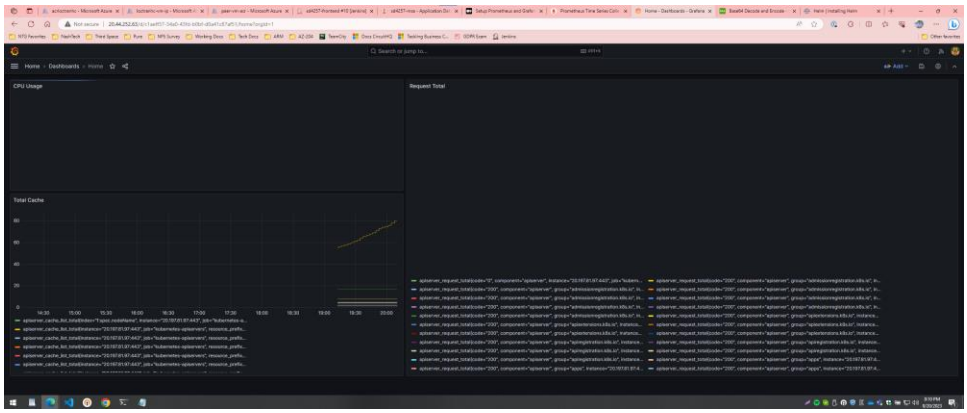


5. Create dashboard at here



6. Add a prometheus metric like this below




Home > Dashboards > Home
CPU Usage
Request Total
Total Cache