

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:

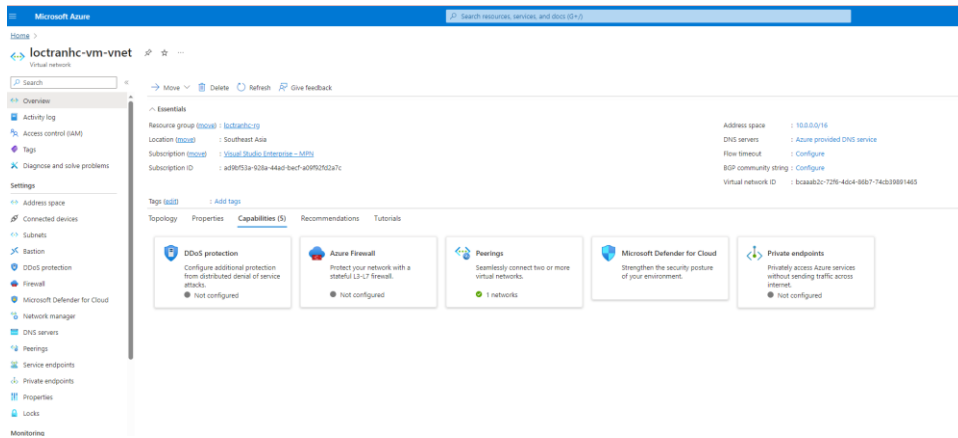
Region	Southeast Asia
Image	Ubuntu Server 22.04
Size	Standard B2s

The screenshot displays the Azure portal interface for a virtual machine named 'loctranhc-vm'. The left sidebar shows the navigation menu with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Networking, Connect, Disks, Size, Microsoft Defender for Cloud, Advisor recommendations, Extensions + applications, Availability + scaling, Configuration, Identity, Properties, Locks, Operations, Bastion, Auto-shutdown, Backup, and Virtual machine recommendations. The main content area is divided into several sections:

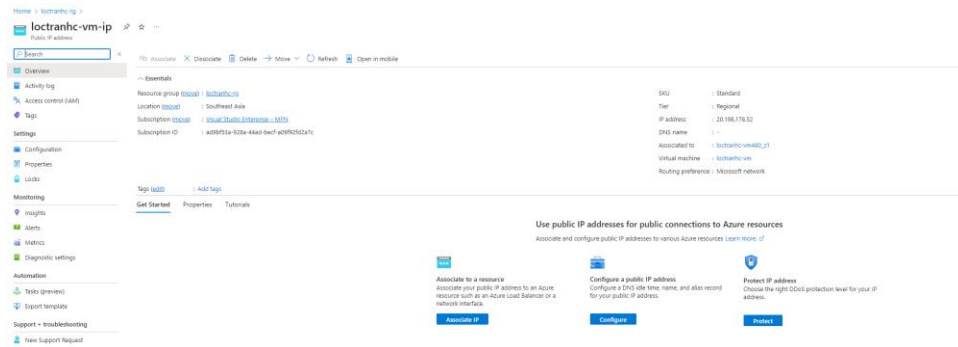
- Essentials:** Provides a quick overview of the VM's status and key details.
 - Resource group: loctranhc-rg
 - Status: Running
 - Location: Southeast Asia (Zone 1)
 - Subscription: Visual Studio Enterprise - M309
 - Subscription ID: ad9b52a-925b-44ad-beef-a096252a7c
 - Availability zone: 1
 - Tags: (add)
 - Operating system: Linux (Ubuntu 22.04)
 - Size: Standard B2s (2 vcpus, 4 GiB memory)
 - Public IP address: 20.198.176.52
 - Virtual network/subnet: loctranhc-vm-vnet/default
 - DNS name: loctranhc-vm
 - Health state: OK
- Properties:** Lists the VM's configuration details.
 - Computer name: loctranhc-vm
 - Operating system: Linux (Ubuntu 22.04)
 - Image publisher: canonical
 - Image offer: 0001-com-ubuntu-server-jammy
 - Image plan: 22_04-lts-gen2
 - VM generation: V2
 - VM architecture: x64
 - Agent status: Ready
 - Agent version: 2.9.1.1
 - Host group: None
 - Host: -
 - Proximity placement group: -
 - Colocation status: N/A
 - Capacity reservation group: -
- Networking:** Shows the network configuration.
 - Public IP address: 20.198.176.52 (Network interface: loctranhc-vm480_p1)
 - Private IP address (IPv4): 10.0.0.4
 - Private IP address (IPv6): -
 - Virtual network/subnet: loctranhc-vm-vnet/default
 - DNS name: Configure
- Size:** Displays the VM's size and resources.
 - Size: Standard B2s
 - Size: Standard B2s
 - vCPUs: 2
 - RAM: 4 GiB
- Disk:** Shows the disk configuration.
 - OS disk: loctranhc-vm_OsDisk_1_3820e1702c7148c38495d57e3dfa356b
 - Encryption at host: Disabled

When creating VM, I also create:

- Vnet for VM:



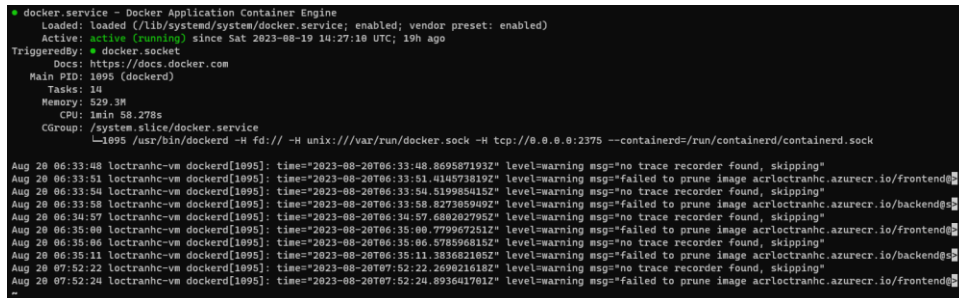
- Public Ip Address:



2. Install necessary tool on VM

Docker:

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



Jenkins:

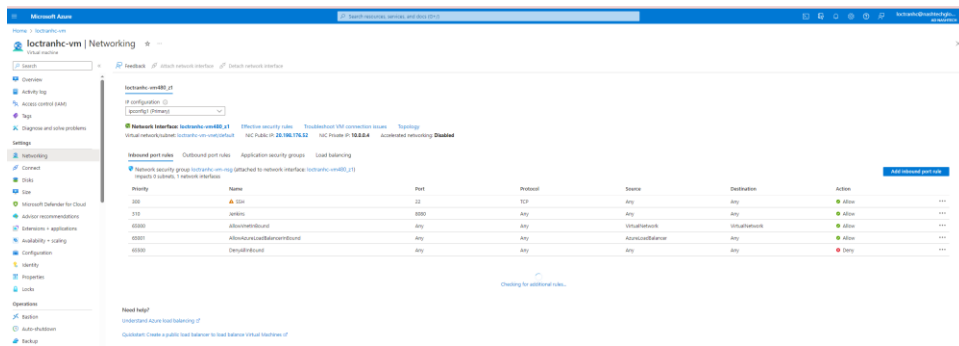
Ref to this link [Install Jenkins](#) for install Jenkins

```
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+0800 [id=1165] INFO o.j.p.g.webhook.WebhookManager$1:run: Github webhooks
Aug 20 06:04:17 loctranhc-vm jenkins[35727]: 2023-08-20 06:04:17.938+0800 [id=117] WARNING o.j.p.g.GitHubOrgWebHookRegister: Failed to register
Aug 20 06:04:19 loctranhc-vm jenkins[35727]: 2023-08-20 06:04:19.918+0800 [id=1174] INFO j.b.OrganizationFolder$OrganizationScanRun: sd4257-
Aug 20 06:04:49 loctranhc-vm jenkins[35727]: 2023-08-20 06:04:49.381+0800 [id=1177] INFO j.b.OrganizationFolder$OrganizationScanRun: sd4257-
Aug 20 06:07:37 loctranhc-vm jenkins[35727]: 2023-08-20 06:07:37.258+0800 [id=1192] INFO o.j.p.g.webhook.WebhookManager$1:run: Github webhooks
Aug 20 06:07:37 loctranhc-vm jenkins[35727]: 2023-08-20 06:07:37.571+0800 [id=1118] WARNING o.j.p.g.GitHubOrgWebHookRegister: Failed to register
Aug 20 06:07:39 loctranhc-vm jenkins[35727]: 2023-08-20 06:07:39.420+0800 [id=1197] INFO j.b.OrganizationFolder$OrganizationScanRun: sd4257-
Aug 20 06:10:45 loctranhc-vm jenkins[35727]: 2023-08-20 06:10:45.339+0800 [id=1235] WARNING o.j.p.w.flow.FlowExecutionListRunRegister: Owner[
Aug 20 06:11:12 loctranhc-vm jenkins[35727]: 2023-08-20 06:11:12.121+0800 [id=1274] WARNING o.j.p.w.flow.FlowExecutionListRunRegister: Owner[
Aug 20 06:30:00 loctranhc-vm jenkins[35727]: 2023-08-20 06:30:00.728+0800 [id=1809] INFO h.triggers.SCHTrigger$Runner$run: SCM changes detected
```

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

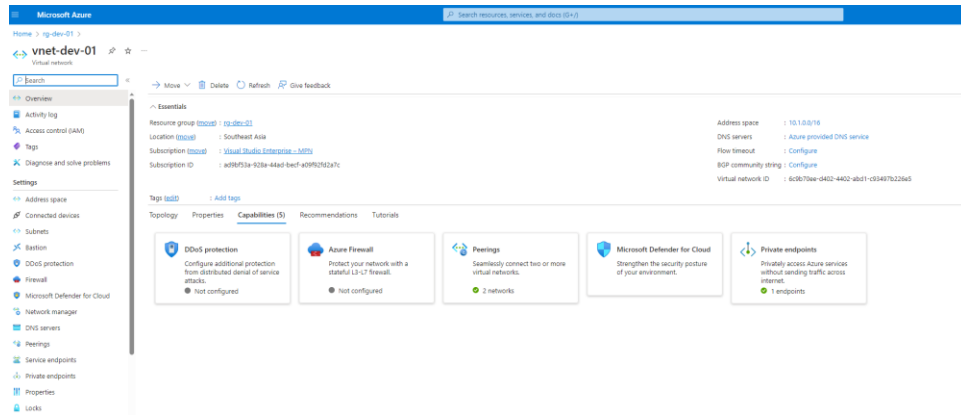
```
loctranhc@loctranhc-vm:~$ kubectl version
Client Version: v1.28.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Unable to connect to the server: dial tcp: lookup aksloctranhc-wga7f6ts.hcp.southeastasia.azmk8s.io on 127.0.0.53: no such host
```

3. Provision with terraform

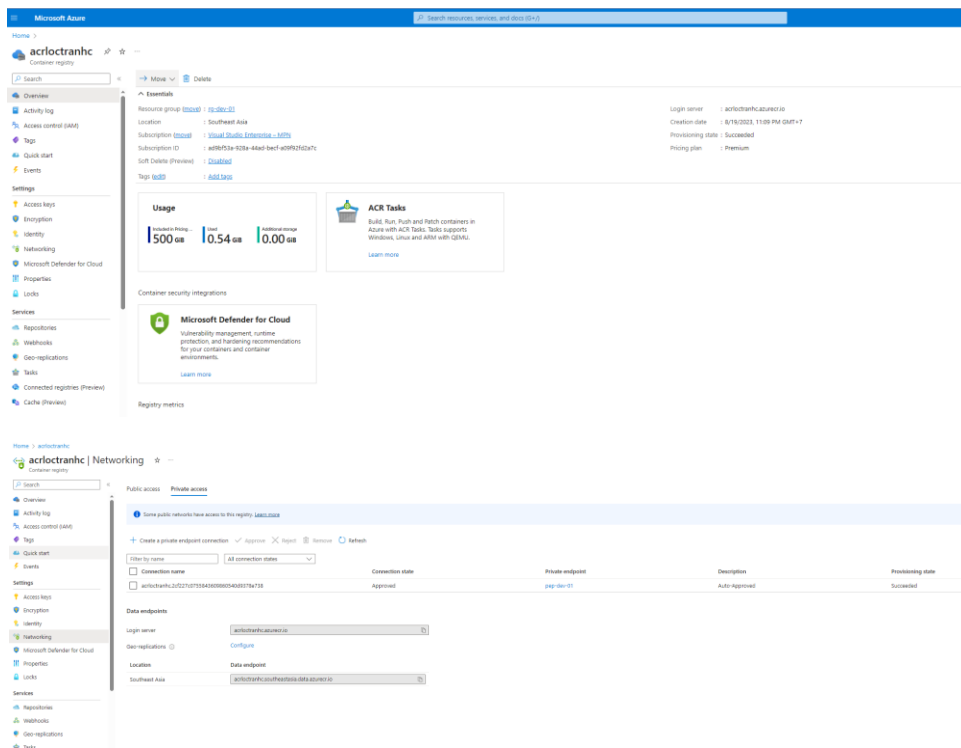
Please ref this repo here: [sd4257_azure_infrastructure](https://github.com/sd4257/azure_infrastructure) to more detail about structure.

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.

Home > **privatelink.azurecr.io** > Search

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Virtual network info

Properties

Links

Monitoring

Alerts

Network

Automation

Tools (preview)

Export template

Support > Troubleshooting

New Support Request

Resource group: **102000** > **10.10.0.0/24**

Subscription: **102000** > **10.10.0.0/24**

Top: **102000** > **10.10.0.0/24**

Not can search for record sets that have been loaded on this page. If you don't see what you're looking for, you can try scrolling to allow more record sets to load.

Name	Type	TTL	Value	Auto registered
privatelink.azurecr.io	A	300	10.10.0.0/24	True
privatelink.azurecr.io	A	300	10.10.0.0/24	True
privatelink.azurecr.io	A	300	10.10.0.0/24	True
privatelink.azurecr.io	A	300	10.10.0.0/24	True

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

Home > **loctranhc-vm-vnet** | Peerings >

peer-vm-acr

loctranhc-vm-vnet

This virtual network

Peering link name

peer-vm-acr

Peering status

Fully Synchronized

Peering state

Succeeded

Traffic to remote virtual network

☒ Allow (default)

☐ Block all traffic to the remote virtual network

Traffic forwarded from remote virtual network

☒ Allow (default)

☐ Block traffic that originates from outside the remote virtual network

Virtual network gateway or Route Server

☐ Use this virtual network's gateway or Route Server

☐ Use the remote virtual network's gateway or Route Server

☒ None (default)

Remote virtual network

Remote Vnet Id

/subscriptions/ad9bf53a-928a-44ad-becf-a09f92fd2a7c/resourceGroups/rg-dev-01/providers/Microsoft.Network/virtual...

Address space

10.1.0.0/16

Azure Kubernetes services

Home > **aks-dev-01** > Search

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Microsoft Defender for Cloud

Kubernetes resources

Namespaces

Workloads

Services and ingress

Storage

Configuration

Custom resources

Settings

Node pools

Cluster configuration

Networking

Extensions + applications

Backup (preview)

Open Service Mesh

CRDs

Automated deployments (preview)

Policies

Resource group: **10.10.0.0**

Status: **Succeeded (Running)**

Location: **Southeast Asia**

Subscription: **Visual Studio Enterprise - 3079**

Subscription ID: **ad9bf53a-928a-44ad-becf-a09f92fd2a7c**

Top: **102000** > **10.10.0.0/24**

Get started

Properties

Monitoring

Capabilities (6)

Recommendations

Tutorials

Kubernetes services

Encryption type

Virtual node pools

Node pools

Node pools

Kubernetes version

Node sizes

Configuration

Kubernetes version

Auto Upgrade Type

Authentication and Authorization

Local accounts

Extensions + applications

No extensions installed

Kubernetes version: **1.25.6**

API server address: **aks-dev-01-3920cyhnp.southeastasia.azureaks.io**

Network type (ipam): **Kubenet**

Node pools: **1 node pool**

API server address: **aks-dev-01-3920cyhnp.southeastasia.azureaks.io**

Network type (ipam): **Kubenet**

Service CIDR: **10.244.0.0/16**

DNS service IP: **10.244.0.10**

Cluster bridge CIDR: **-**

Network Policy: **None**

Load balancer: **Standard**

HTTP application routing: **Not enabled**

Private cluster: **Not enabled**

Authorized IP ranges: **Not enabled**

Application Gateway ingress controller: **Not enabled**

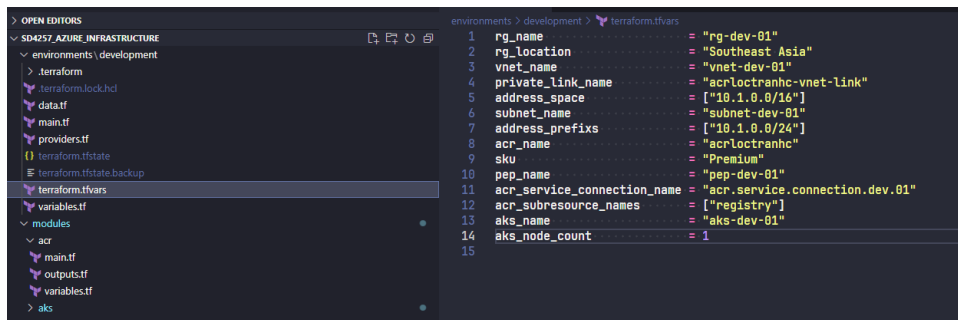
Integrations

Container insights: **Not enabled**

Workspace resource ID: **-**

Note

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



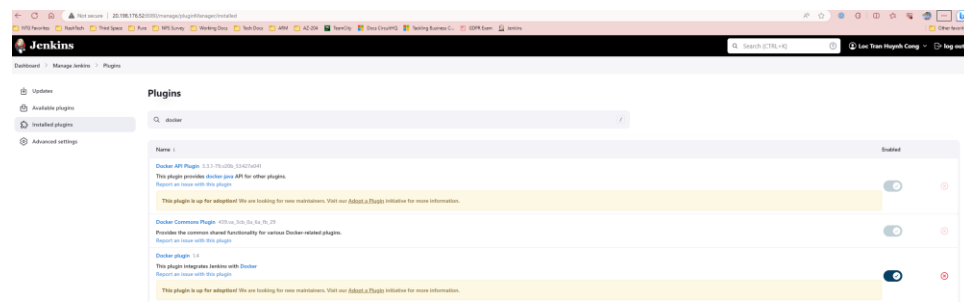
And run command “terraform apply” at /environments/development

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.

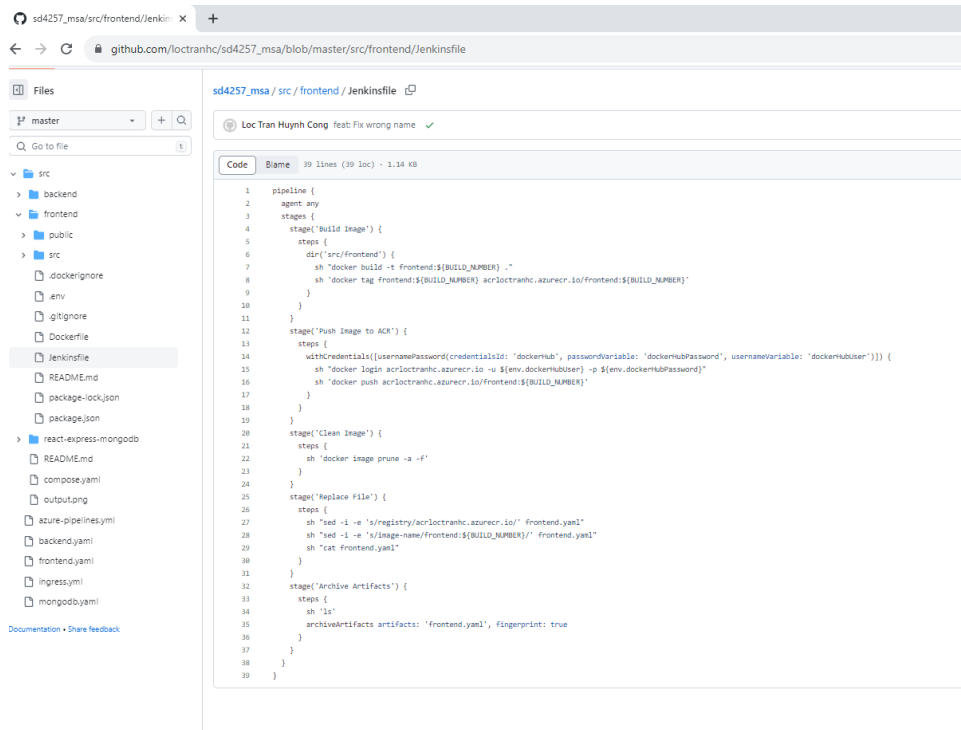


Stage 3: Set up CI

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

General Enabled 

Description

[Plain text] [Preview](#)

☐ Discard old builds [?](#)

☐ Do not allow concurrent builds

☐ Do not allow the pipeline to resume if the controller restarts

☐ GitHub project

☐ Pipeline speed/durability override [?](#)

☐ Preserve stashes from completed builds [?](#)

☐ This project is parameterized [?](#)

☐ Throttle builds [?](#)

Build Triggers

☐ Build after other projects are built [?](#)

☐ Build periodically [?](#)

☒ GitHub hook trigger for GITSCM polling [?](#)

☐ Poll SCM [?](#)

☐ Quiet period [?](#)

☐ Trigger builds remotely (e.g., from scripts) [?](#)

Advanced Project Options

Advanced ▼

Pipeline

Definition

Pipeline script from SCM ▼

SCM [?](#)

Git ▼ [?](#)

Repositories [?](#)

Repository URL [?](#) ✕

`https://github.com/octranhc/s4257_rsa.git`

Credentials [?](#)

octranhc@gmail.com/***** (git) ▼

Add ▼

Advanced ▼

Add Repository

Branches to build [?](#)

Branch Specifier (blank for 'any') [?](#) ✕

`*/master`

Add Branch

Script Path [?](#)

`src/frontend/jenkinsfile`

☒ Lightweight checkout [?](#)

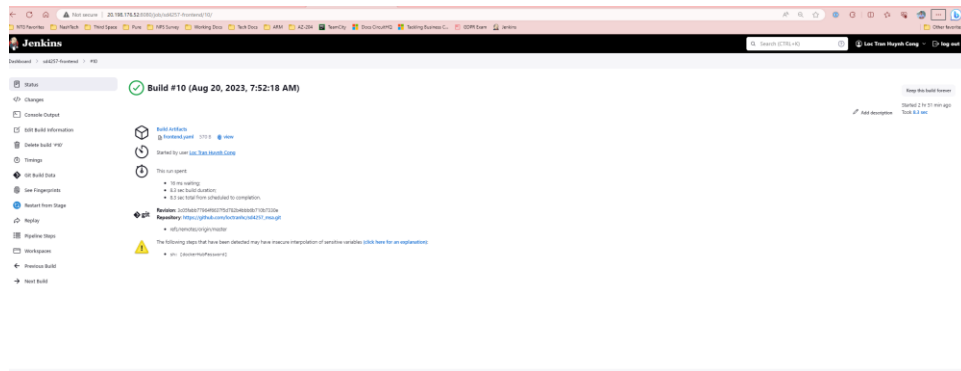
[Pipeline Syntax](#)

Save Apply

Note:

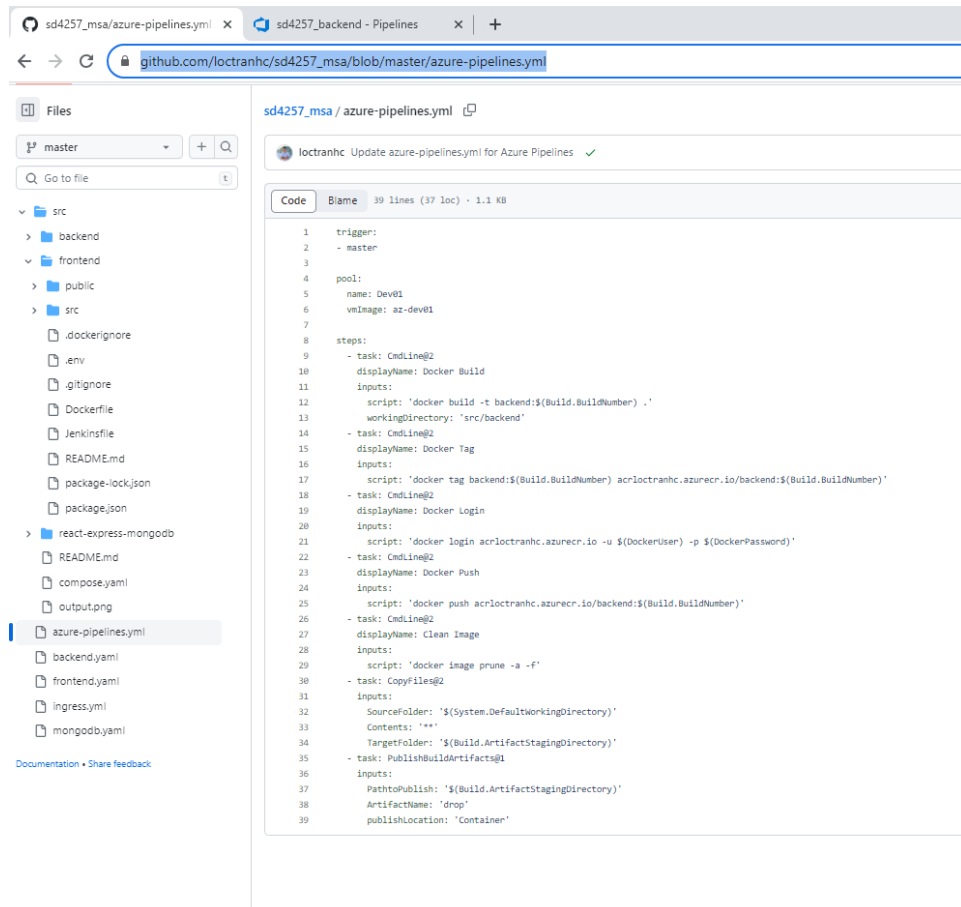
- At tab Credentials, select the git credential which I created at step 1.
- Script Path is filled for define the path to frontend Jenkinsfile.

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

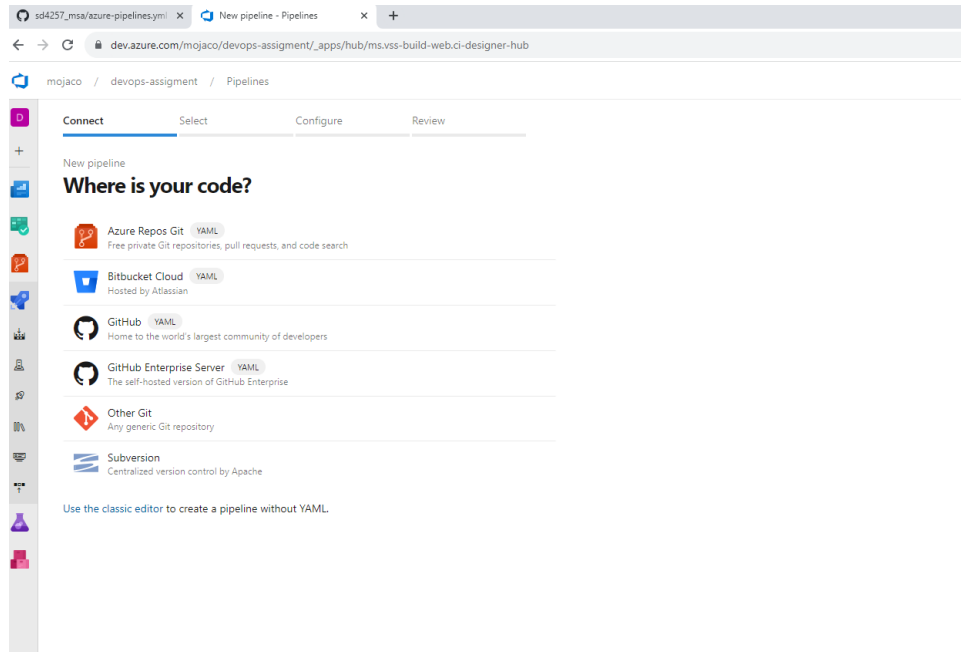


Azure DevOps

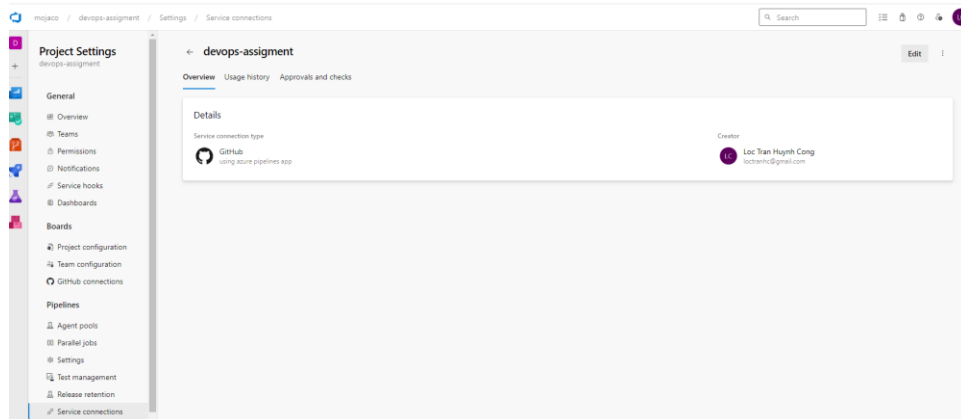
I created azure-pipelines.yml file at root of sd4257_msa repo. Please check direct file at here: [azure-pipelines.yml](https://github.com/loctranhc/sd4257_msa/blob/master/azure-pipelines.yml)



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.

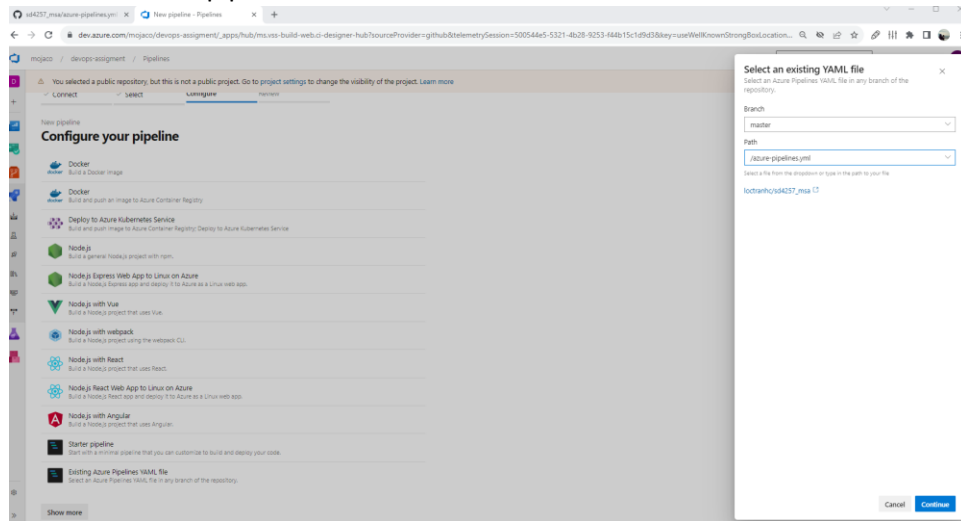


Select the sd4257_msa repo, and using the existing pipeline file.

The screenshot shows the 'New pipeline' wizard in Azure DevOps, specifically the 'Select' tab. The breadcrumb navigation at the top reads 'mojaco / devops-assignment / Pipelines'. The progress bar indicates the steps: 'Connect' (checked), 'Select' (active), 'Configure', and 'Review'. The main heading is 'New pipeline' followed by 'Select a repository'. Below this is a search bar with the placeholder 'Filter by keywords' and a dropdown menu set to 'My repositories'. A list of repositories is displayed, including 'loctranhc/sd4257_shared_library' (marked as private), 'loctranhc/sd4257_azure_infrastructure', and 'loctranhc/sd4257_msa' (highlighted with a red box). A note below the list states: 'Showing the most recently used repositories where you are a collaborator. If you can't find a repository, make sure you provide access. You may also select a specific connection.'

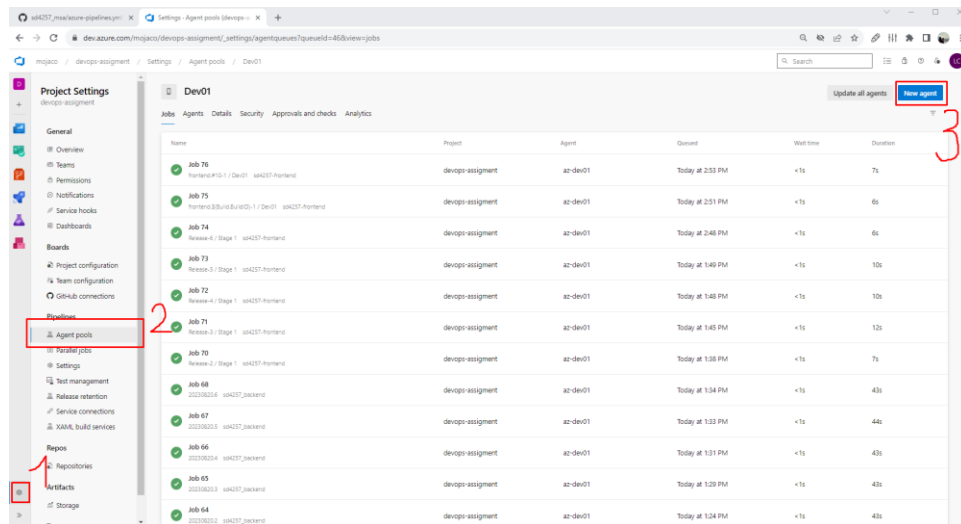
The screenshot shows the 'Configure your pipeline' step of the 'New pipeline' wizard. The breadcrumb navigation remains 'mojaco / devops-assignment / Pipelines'. The progress bar now shows 'Connect' (checked), 'Select' (checked), 'Configure' (active), and 'Review'. A warning banner at the top states: 'You selected a repository, but this is not a public project. Go to project settings to change the visibility of the project. Learn more'. The main heading is 'Configure your pipeline'. A list of pipeline templates is shown, including 'Docker', 'Node.js', and 'Node.js with Angular'. The 'Existing Azure Pipelines YAML file' option at the bottom is highlighted with a red box. A note below this option says: 'Select an Azure Pipelines YAML file in any branch of the repository.'

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

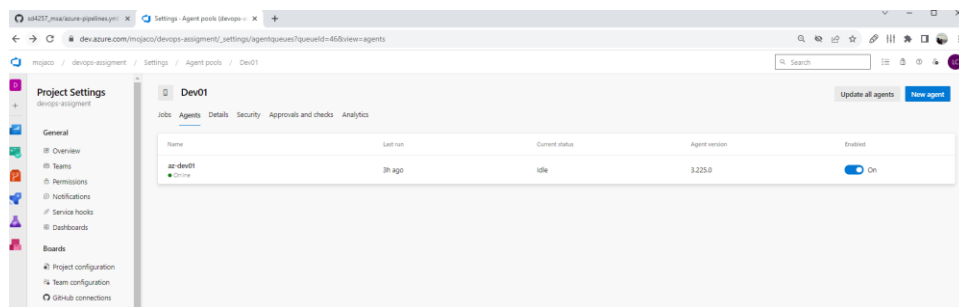
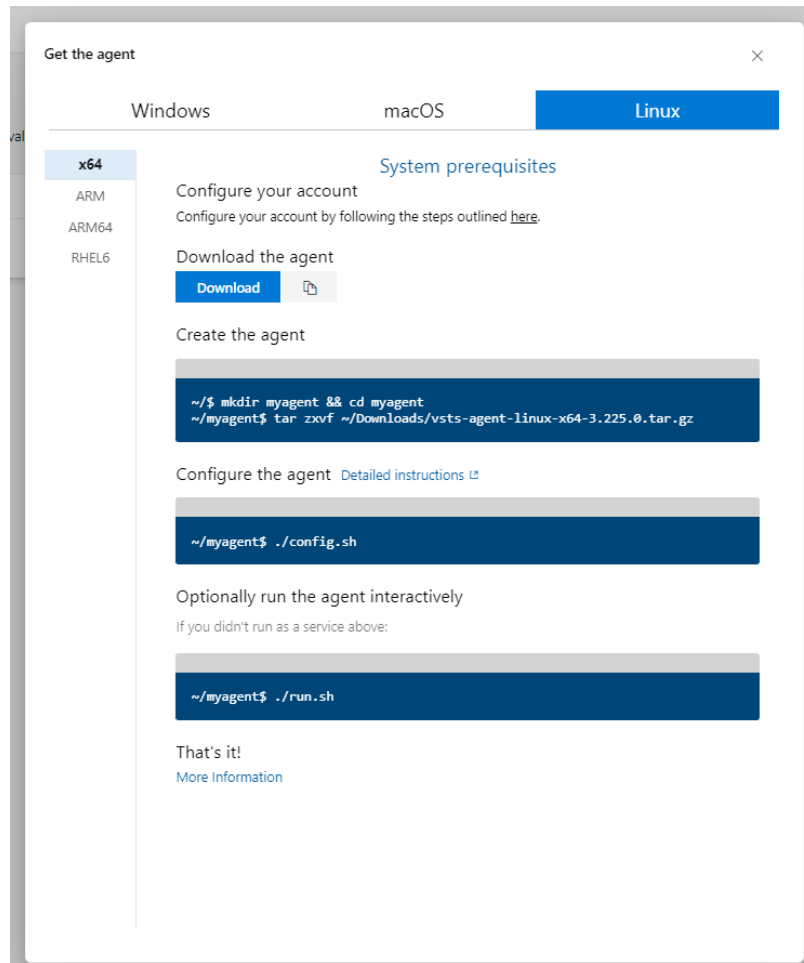


Note:

- 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.
 - o 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:

sd4257_msa / azure-pipelines.yml

loctranhc Update azure-pipelines.yml for Azure Pipelines ✓

Code Blame 39 lines (37 loc) · 1.1 KB

```
1 trigger:
2   - master
3
4   pool:
5     name: Dev01
6     vmImage: az-dev01
7
8   steps:
9     - task: CmdLine@2
10      displayName: Docker Build
```

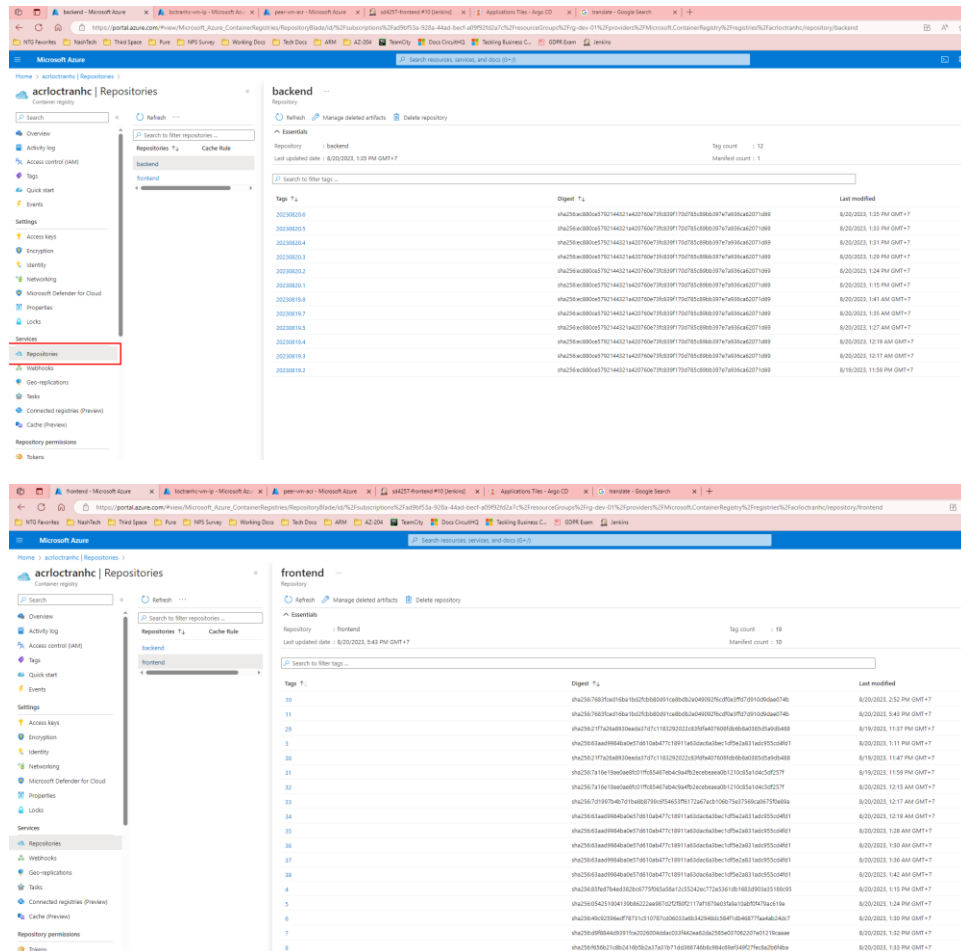
Pipelines - Runs for sd4257_backend

dev.azure.com/mojaco/devops-assignment/_build?definitionId=3

sd4257_backend

Description	Stages	
#20230820.6 • feat: Fix wrong name Individual CI for master 3c05fab0	✓	5h ago 46s
#20230820.5 • feat: Fix not archive Individual CI for master 7bac80de	✓	5h ago 47s
#20230820.4 • feat: Check current file Individual CI for master 8702e73e	✓	5h ago 46s
#20230820.3 • feat: Not find frontend.yml Individual CI for master 8468b237	✓	5h ago 46s
#20230820.2 • feat: Not find frontend.yml Individual CI for master 6563a038	✓	5h ago 46s
#20230820.1 • feat: Add ArchiveArtifacts stage Individual CI for master 320ef028	✓	5h ago 46s
#20230819.8 • Merge branch 'master' of https://github.com/loctranhc/sd4257_msa Individual CI for master 0898e82e	✓	17h ago 43s
#20230819.7 • Update azure-pipelines.yml for Azure Pipelines Individual CI for master 668b5b88	✓	17h ago 45s
#20230819.6 • Update azure-pipelines.yml for Azure Pipelines Individual CI for master 98a32a08	✗	17h ago 21s
#20230819.5 • Update azure-pipelines.yml for Azure Pipelines Individual CI for master a1408b03	✓	17h ago 25s

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

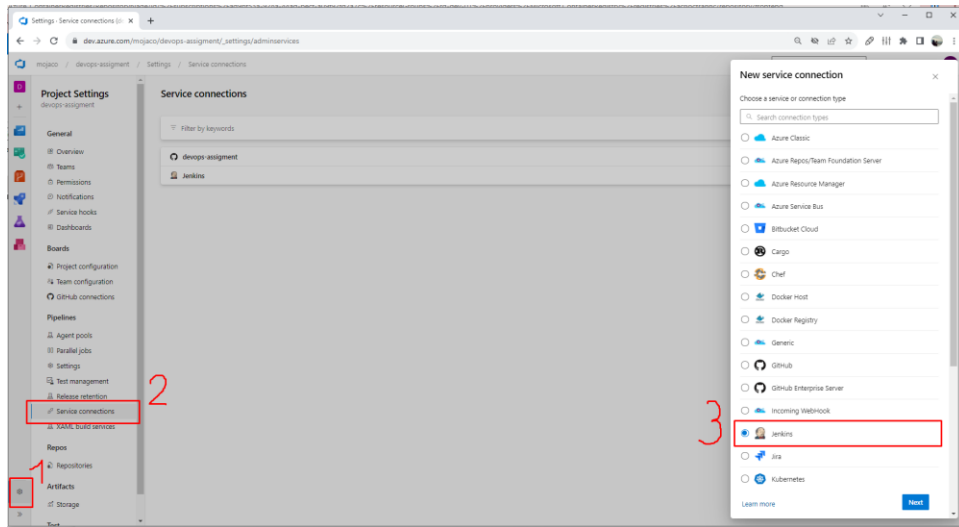


Stage 4: Set up CD

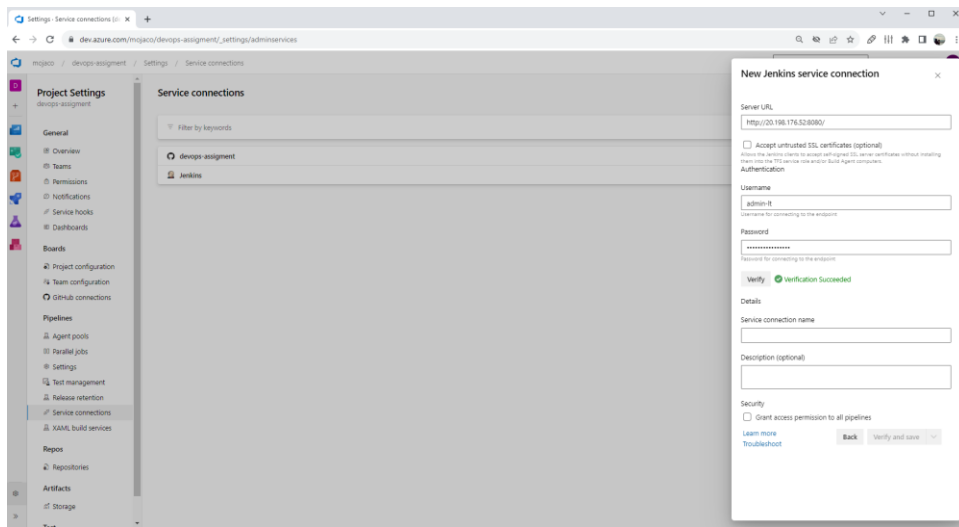
Azure DevOps

For frontend

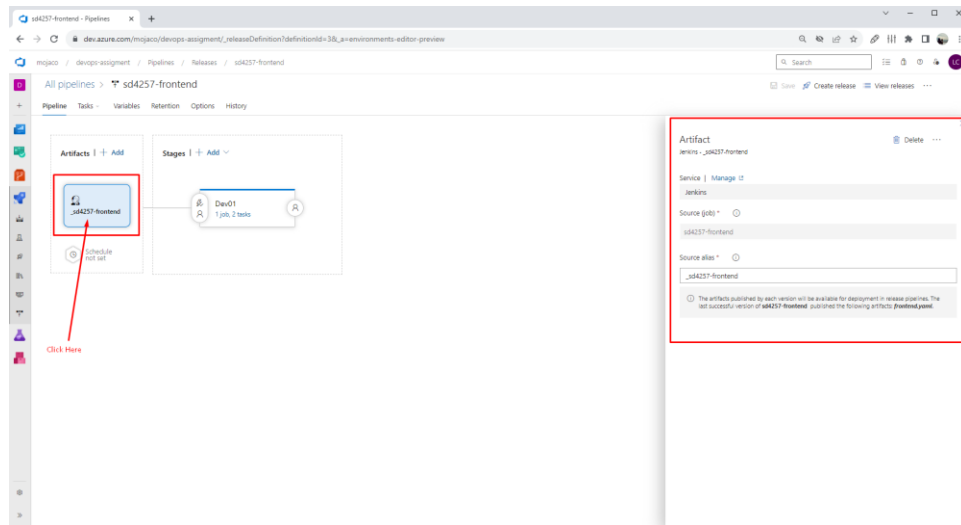
- As the below Jenkins CI step, I published frontend.yaml as an artifact.
- First I need create a service connection to Jenkins. To create, ref the below images.



Fill config my jenkins like the below image:

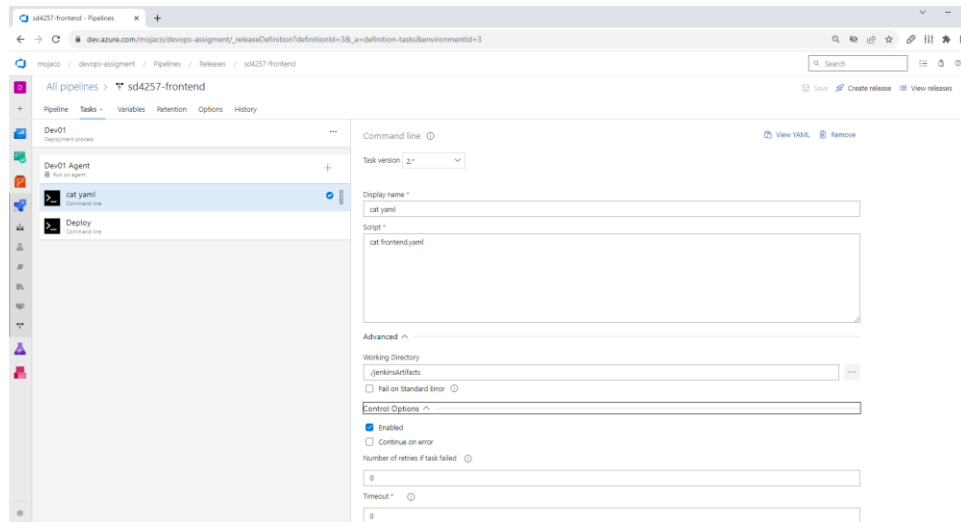


Create artifact for release pipeline, I choose my jenkins like this image

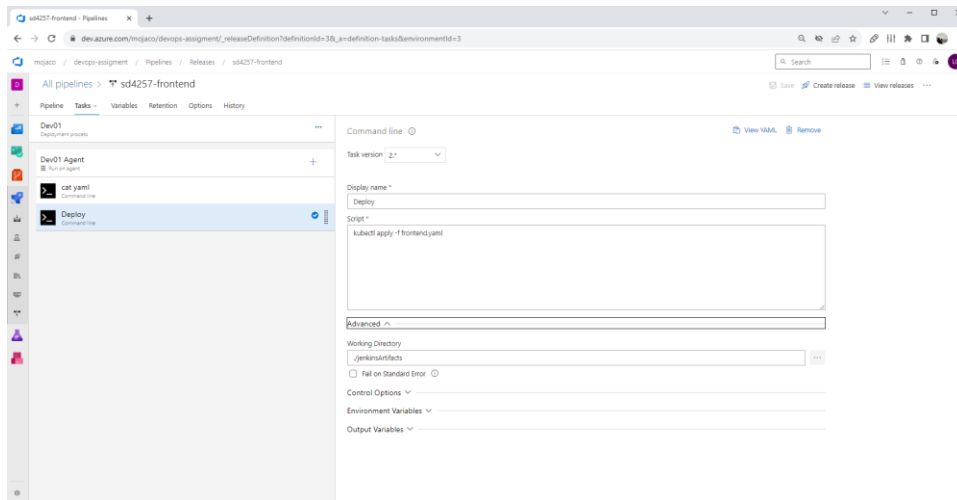


I set up 1 stage with 2 tasks

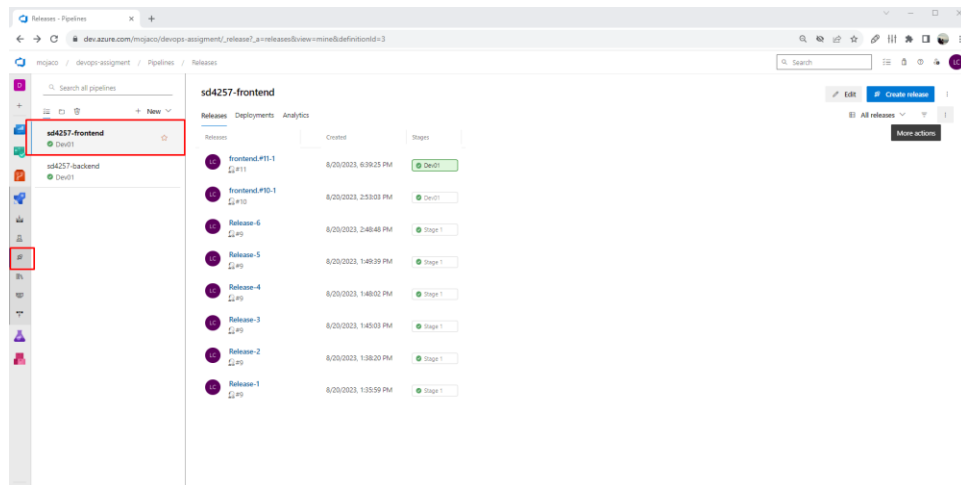
Task 1: Print content frontend.yaml to log. Check issue if frontend.yaml file is wrong.



Task 2: Apply frontend.yaml



I able to check release here:

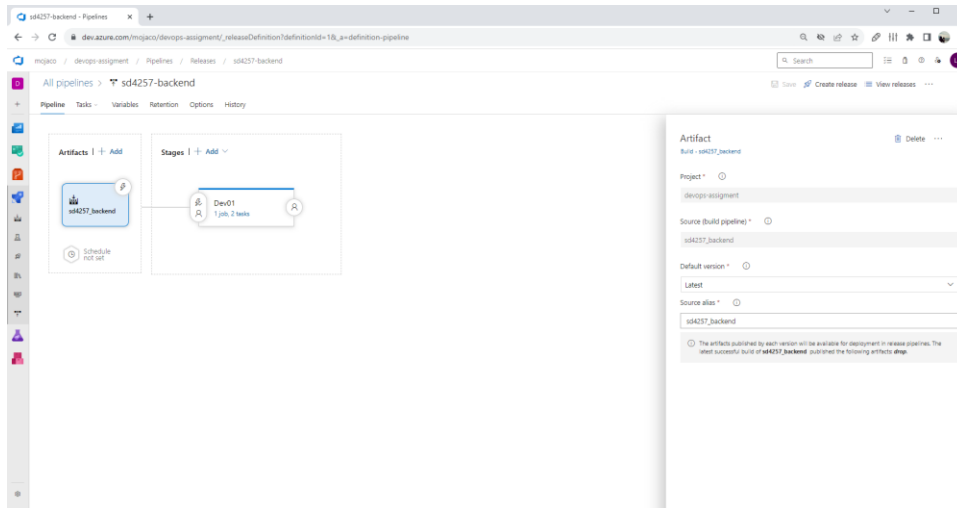


And check on cluster

```
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
```

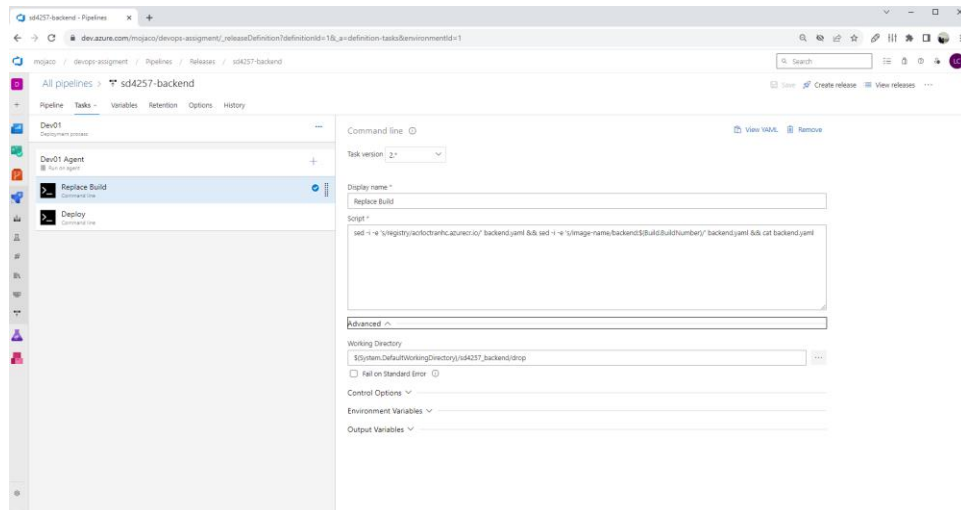
For backend

Using the artifact from azure devops pipeline for backend

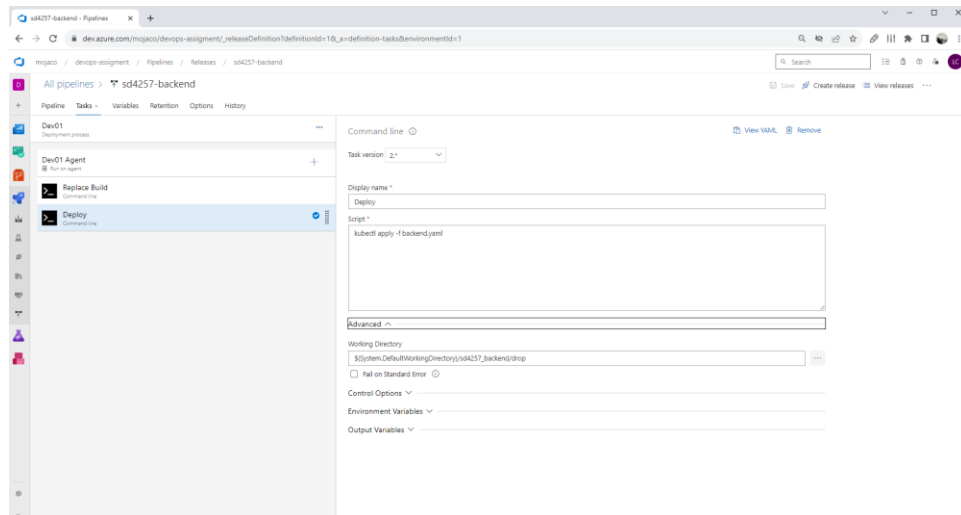


I set up 1 stage with 2 tasks

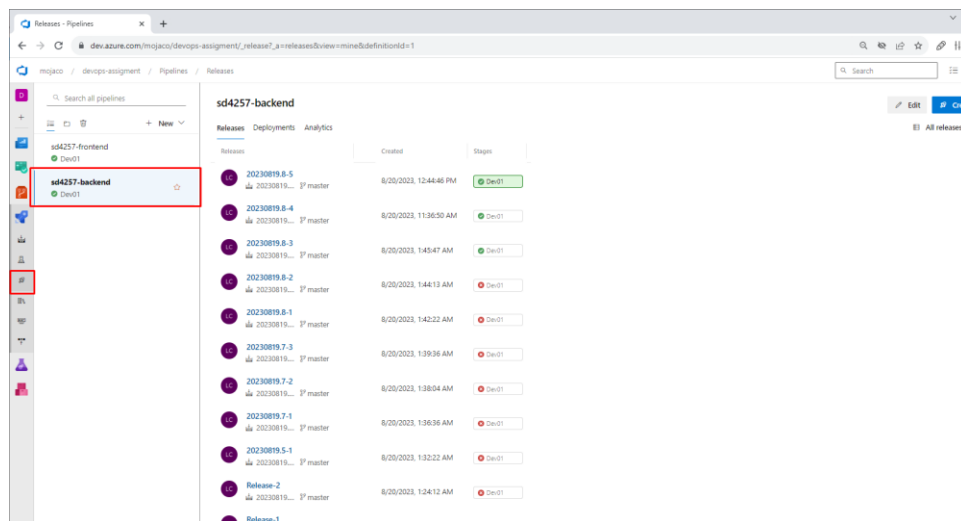
Task 1: Replace the image name of backend with current build from azure artifact.



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.

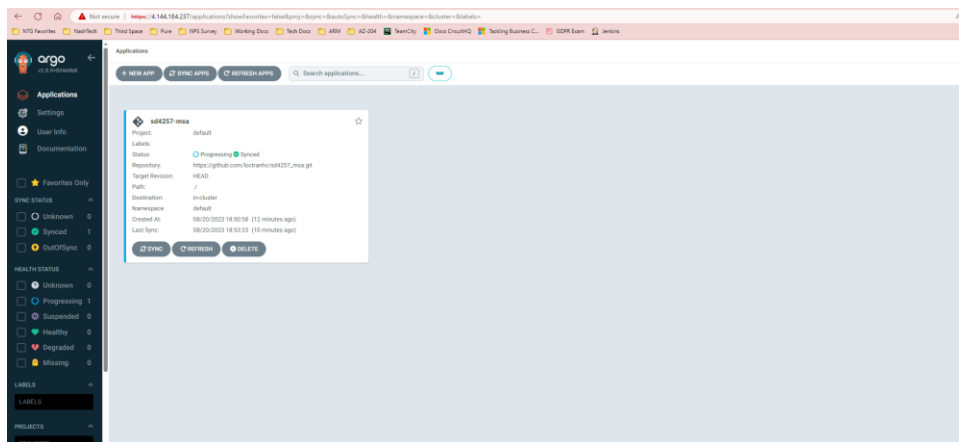
3. Expose argocd-server with node type = LoadBalancer by run command “kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}”

```
loctranhc@loctranhc-vm:~$ kubectl get svc -n argocd
NAME                                TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
one>
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:~$ |
```

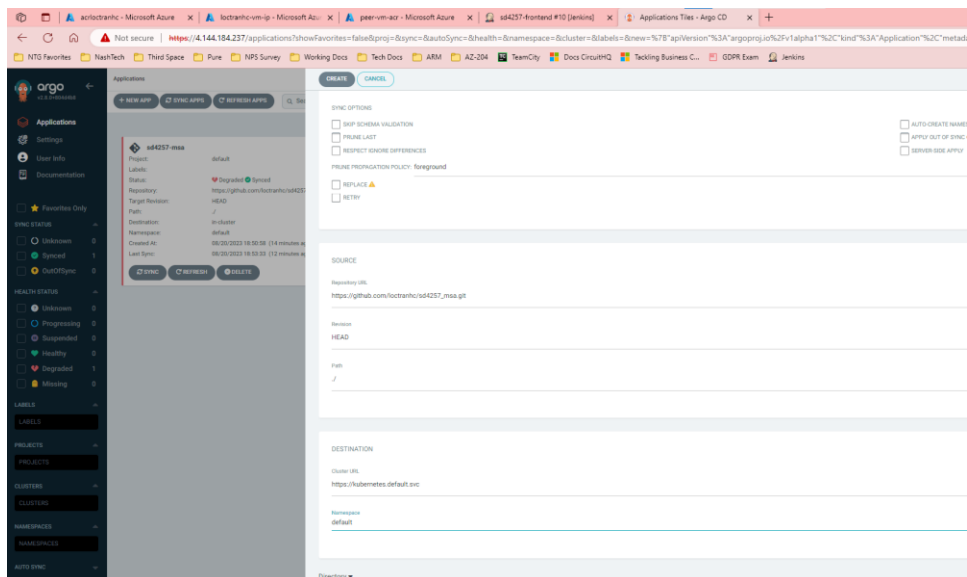
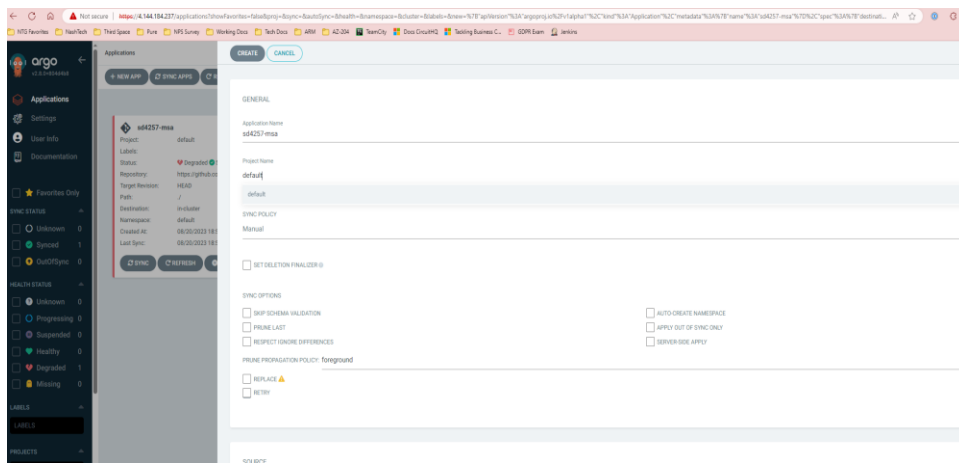
Login into ArgoCD with

UserName: admin

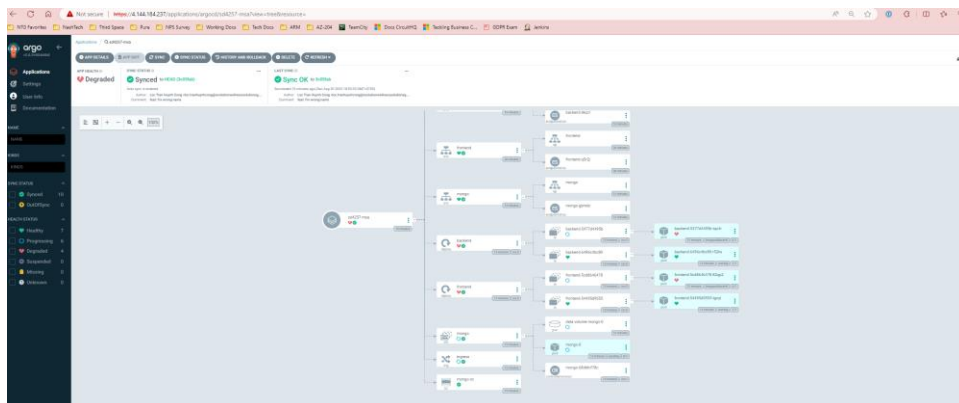
Password: Get from step 2.



Create new app as the below image



After completed 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

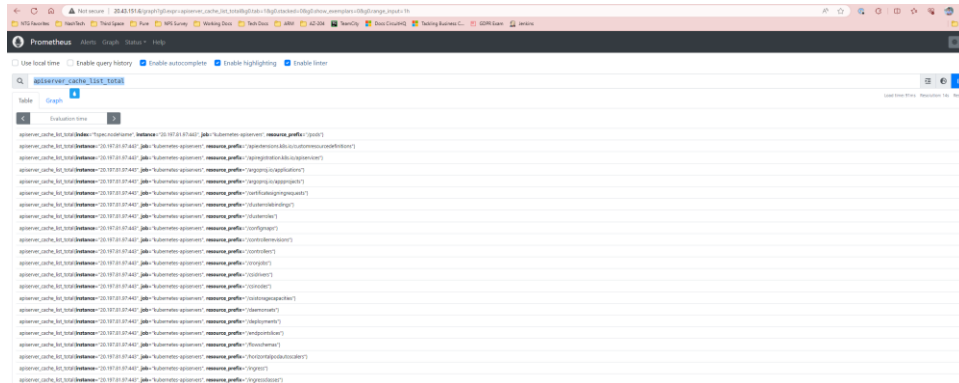
Helm now has an installer script that will automatically grab the latest version of Helm and **install it locally**.

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	None	<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

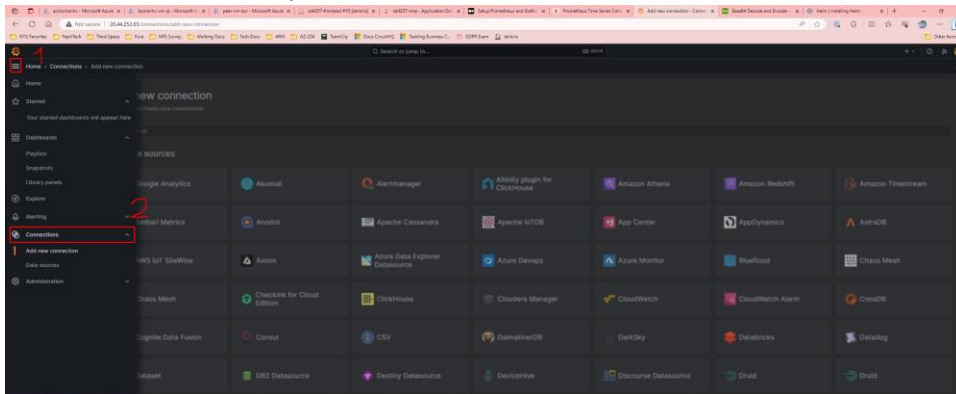
```
loctranhc@loctranhc-vm:~$
```

Setup DashBoard on Grafana

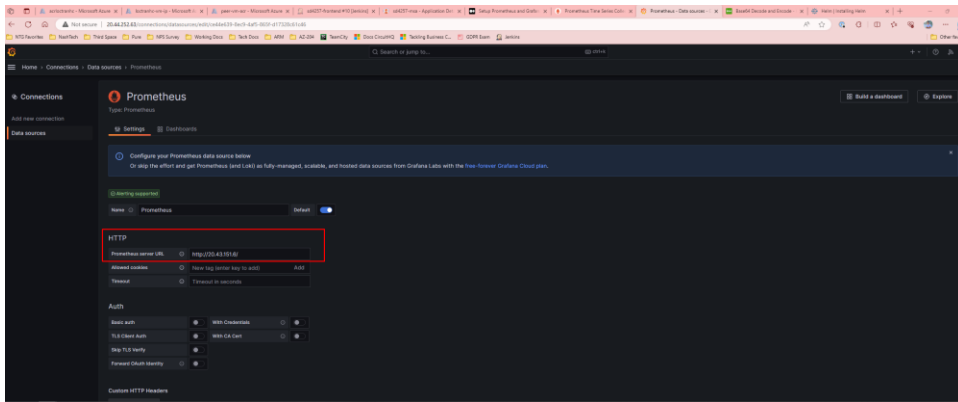
1. To login to Grafana, I need get account information by command: "kubectl get secret — namespace default grafana -o yaml"

```
loctranhc@loctranhc-vm:~$ kubectl get secret - namespace default grafana -o yaml
apiVersion: v1
items:
- apiVersion: v1
  data:
    admin-password: ZGVVBZ1pkRGJ5SHJZdHZtY242SktpNkVKcXM5aXBGWTZURG5MR3luVQ==
    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
```

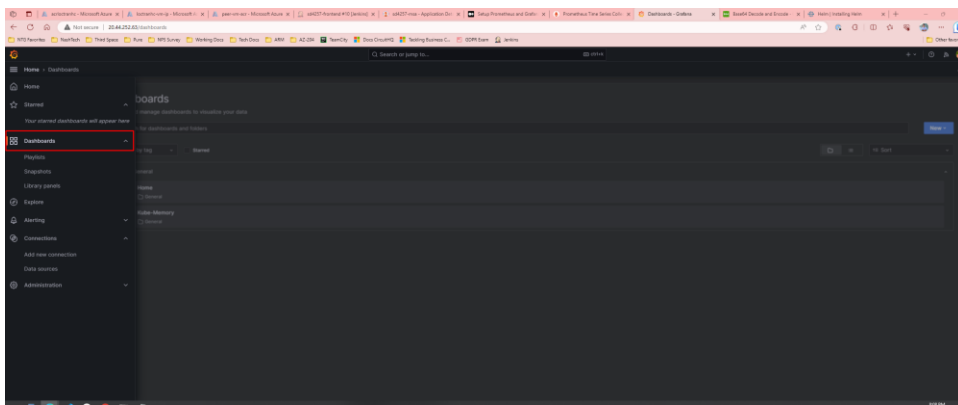
2. Take admin-user and admin-password and decode base64 to get exactly account login.
3. Add new connection to prometheus.



4. Set Prometheus Url server and click Save & Test



5. Create dashboard at here



6. Add a prometheus metric like this below

