Name: **Đỗ Minh Trí**

Staff Code: **SD5976**

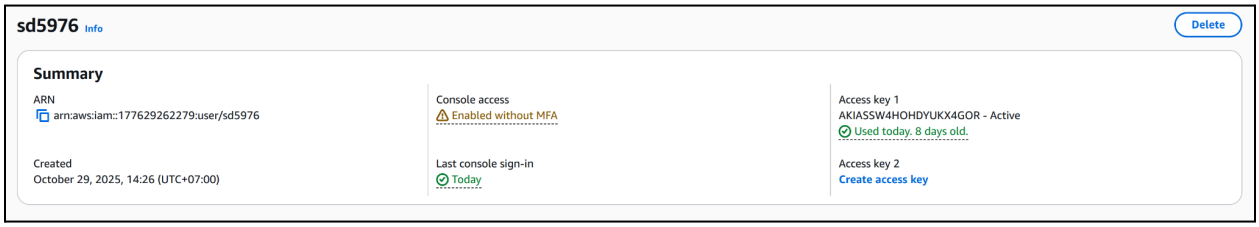
AWS Infrastructure With ArgoCD

I. Setting Up AWS Infrastructure with Terraform

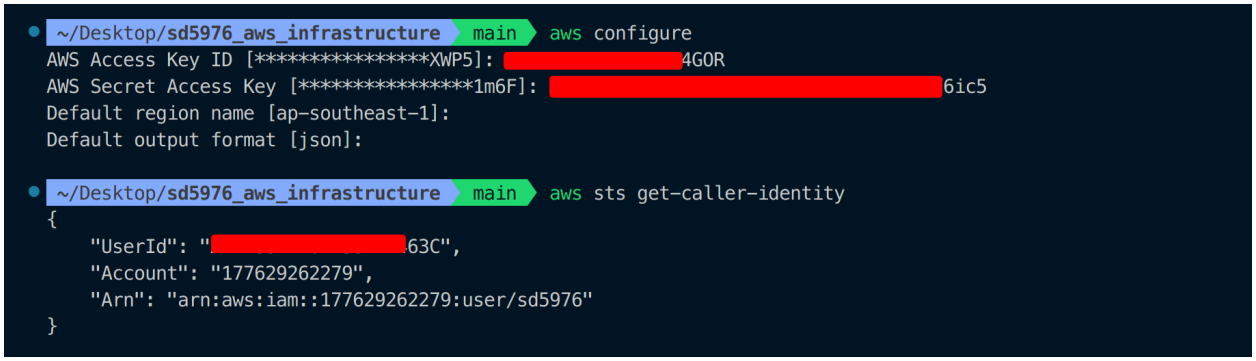
We will deploy AWS infrastructure using Terraform through this repository:

https://github.com/tri-dominh/sd5976_aws_infrastructure

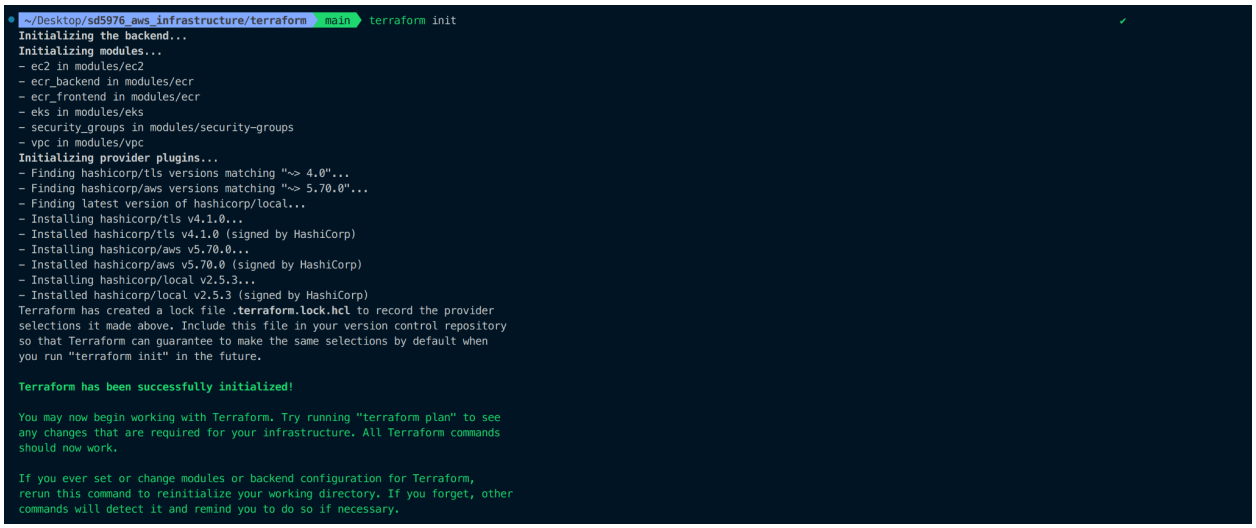
First, I created an IAM user for practical DevOps training: **sd5976**.



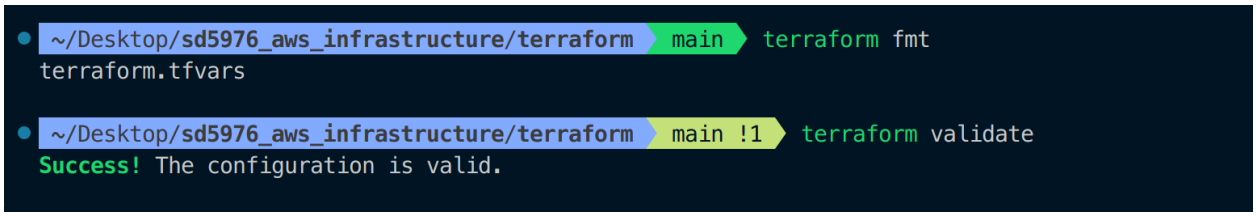
Configure this IAM user on local via **AWS CLI** with Access Key and Secret Access Key.



Go to **sd5976_aws_infrastructure/terraform**, run **terraform init** to initialize the infrastructure.



Use **terraform fmt** and **terraform validate** to format and check your code.



Run **terraform plan** to preview resources that will be created.

Execute **terraform apply** to deploy the resources.

Type **yes** to confirm.

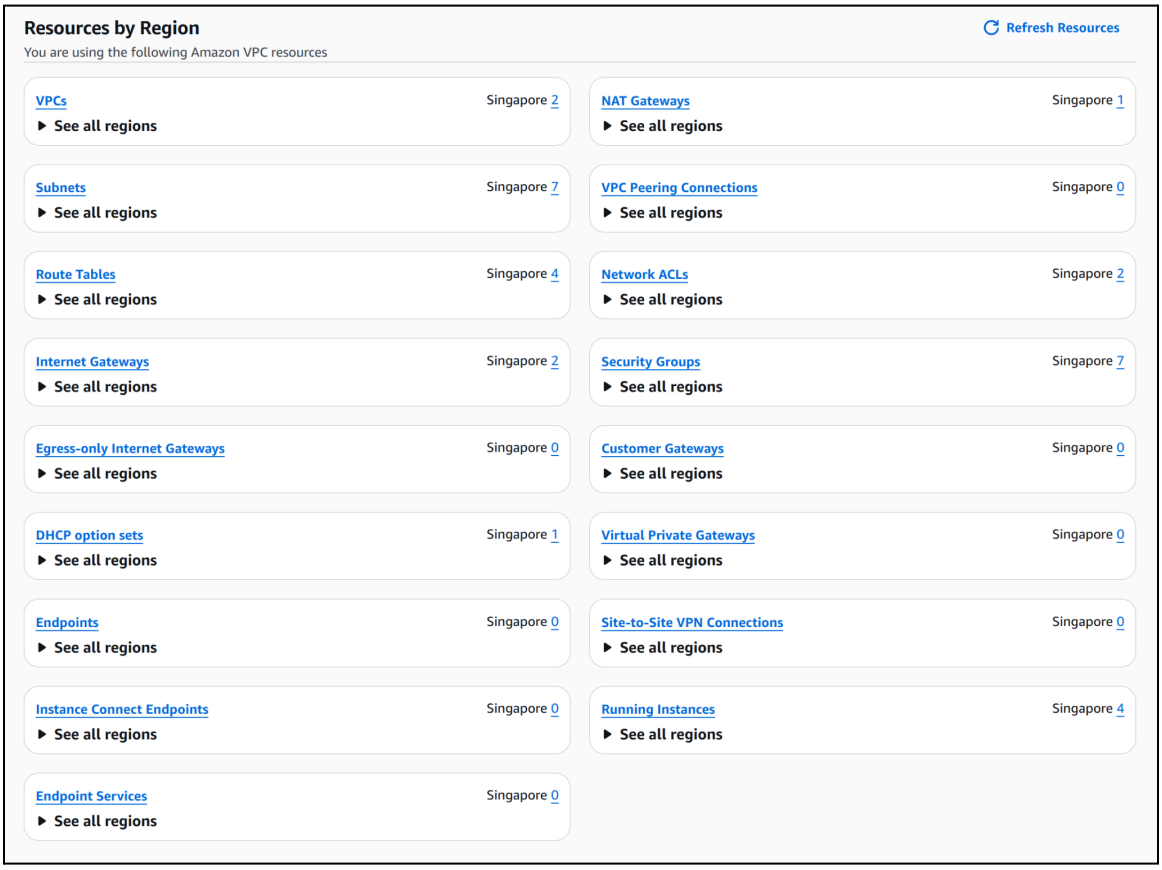
```
Plan: 38 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + configure_kubectl               = "aws eks update-kubeconfig --region ap-southeast-1 --name cicd-pipeline-cluster"
  + docker_ip                       = (known after apply)
  + ecr_backend_repository_name     = "sd5976-backend"
  + ecr_backend_repository_url      = (known after apply)
  + ecr_frontend_repository_name    = "sd5976-frontend"
  + ecr_frontend_repository_url     = (known after apply)
  + eks_cluster_endpoint            = (known after apply)
  + eks_cluster_name                = "cicd-pipeline-cluster"
  + jenkins_initial_password_command = (known after apply)
  + jenkins_ip                      = (known after apply)
  + jenkins_url                     = (known after apply)
  + ssh_connection_docker           = (known after apply)
  + ssh_connection_jenkins          = (known after apply)
  + vpc_id                          = (known after apply)

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

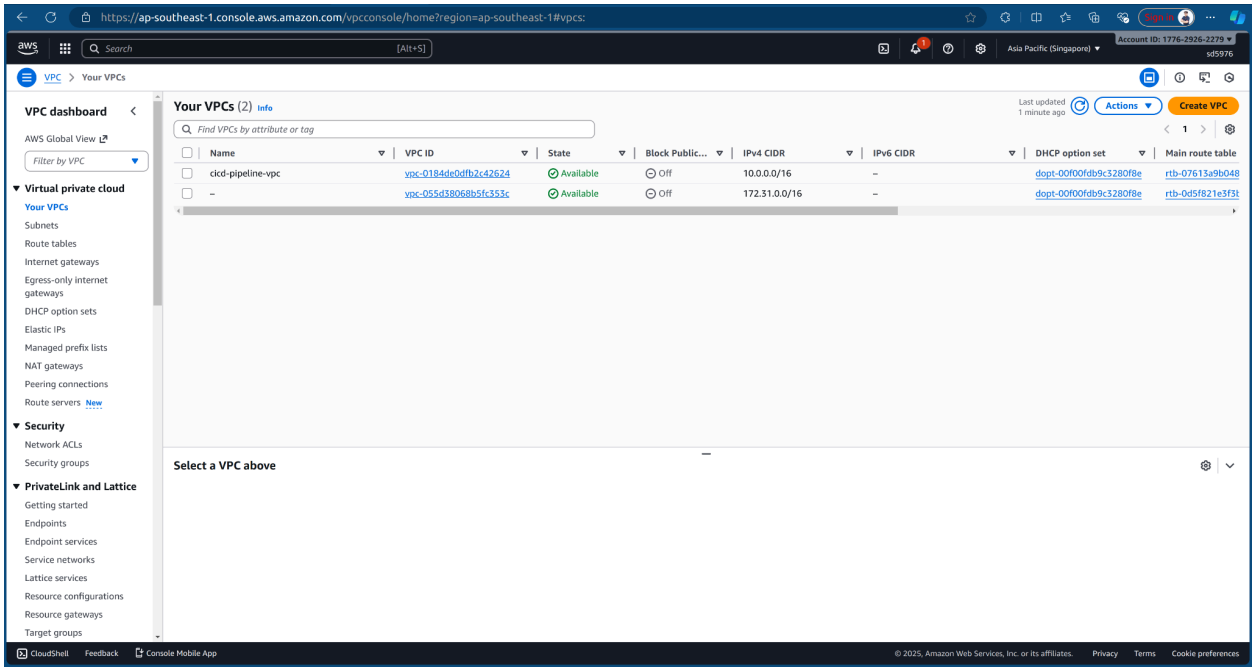
Resource creation may take around 20 minutes or more.

```
Apply complete! Resources: 38 added, 0 changed, 0 destroyed.

Outputs:

configure_kubectl = "aws eks update-kubeconfig --region ap-southeast-1 --name cicd-pipeline-cluster"
docker_ip = "47.128.237.198"
ecr_backend_repository_name = "sd5976-backend"
ecr_backend_repository_url = "177629262279.dkr.ecr.ap-southeast-1.amazonaws.com/sd5976-backend"
ecr_frontend_repository_name = "sd5976-frontend"
ecr_frontend_repository_url = "177629262279.dkr.ecr.ap-southeast-1.amazonaws.com/sd5976-frontend"
eks_cluster_endpoint = "https://9A770A8B363B32838F392C1018289B7E.gr7.ap-southeast-1.eks.amazonaws.com"
eks_cluster_name = "cicd-pipeline-cluster"
jenkins_initial_password_command = "ssh -i cicd-pipeline-key.pem ubuntu@13.228.28.131 'sudo cat /var/lib/jenkins/secrets/initialAdminPassword'"
jenkins_ip = "13.228.28.131"
jenkins_url = "http://13.228.28.131:8080"
ssh_connection_docker = "ssh -i cicd-pipeline-key.pem ubuntu@47.128.237.198"
ssh_connection_jenkins = "ssh -i cicd-pipeline-key.pem ubuntu@13.228.28.131"
vpc_id = "vpc-0184de0dfb2c42624"

○ ~/Desktop/sd5976_aws_infrastructure/terraform main
```

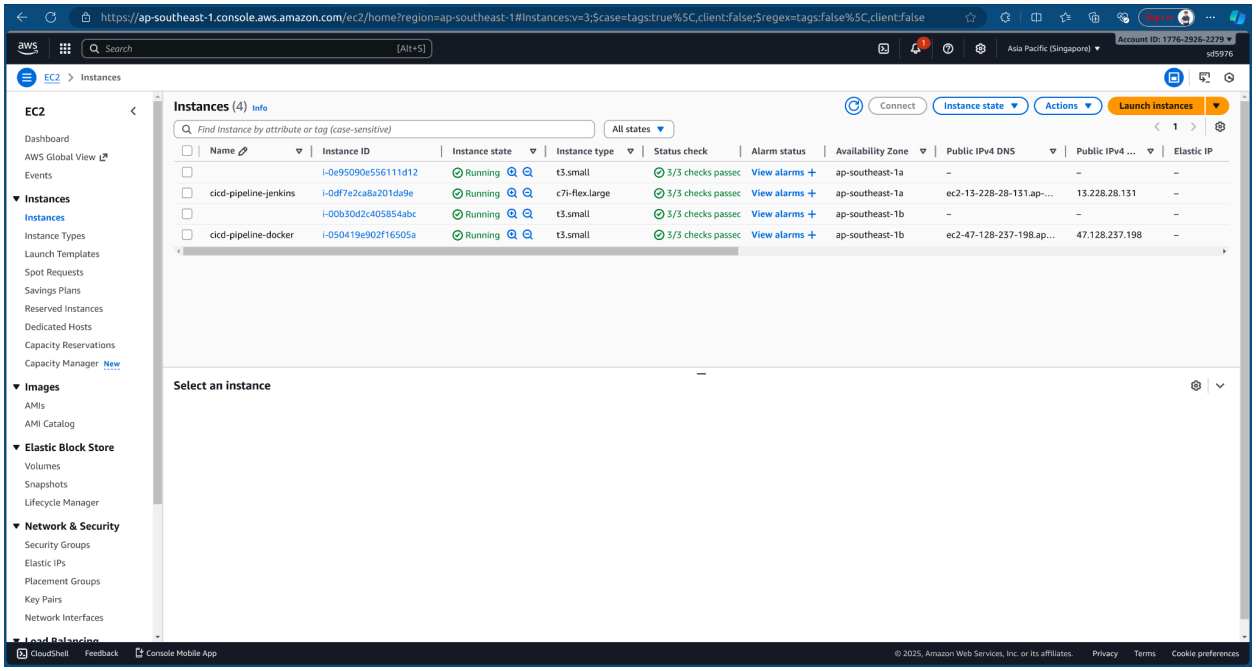
After applying, you can see the following resources on AWS:

VPC

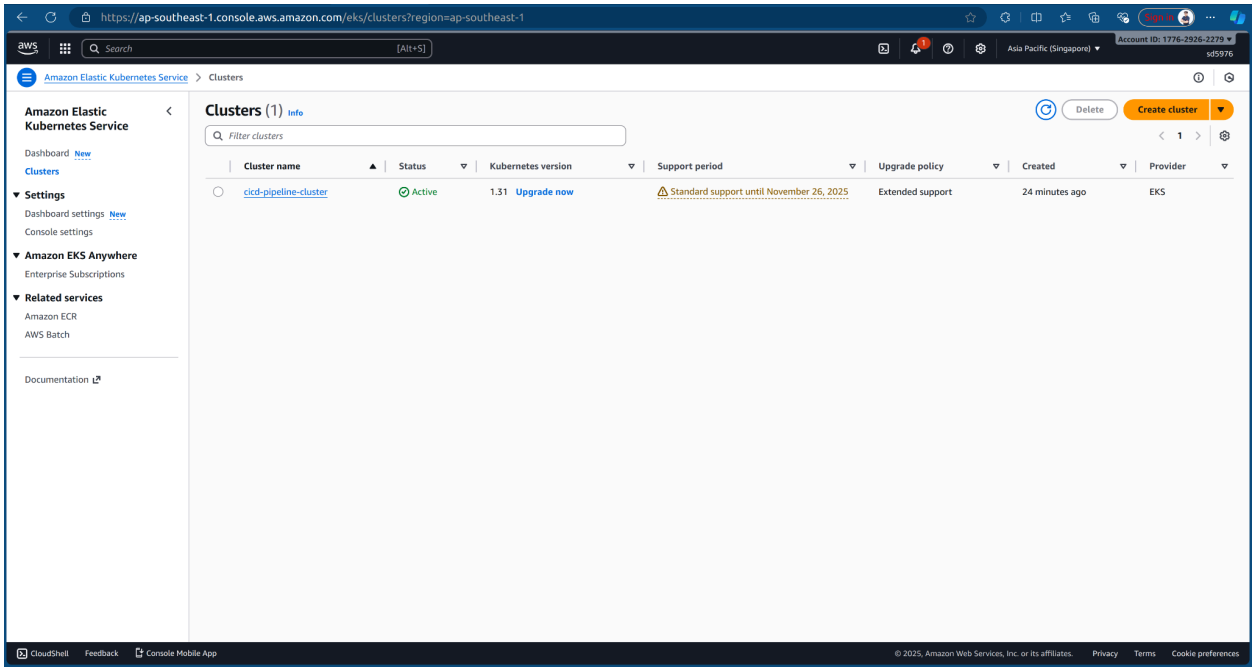


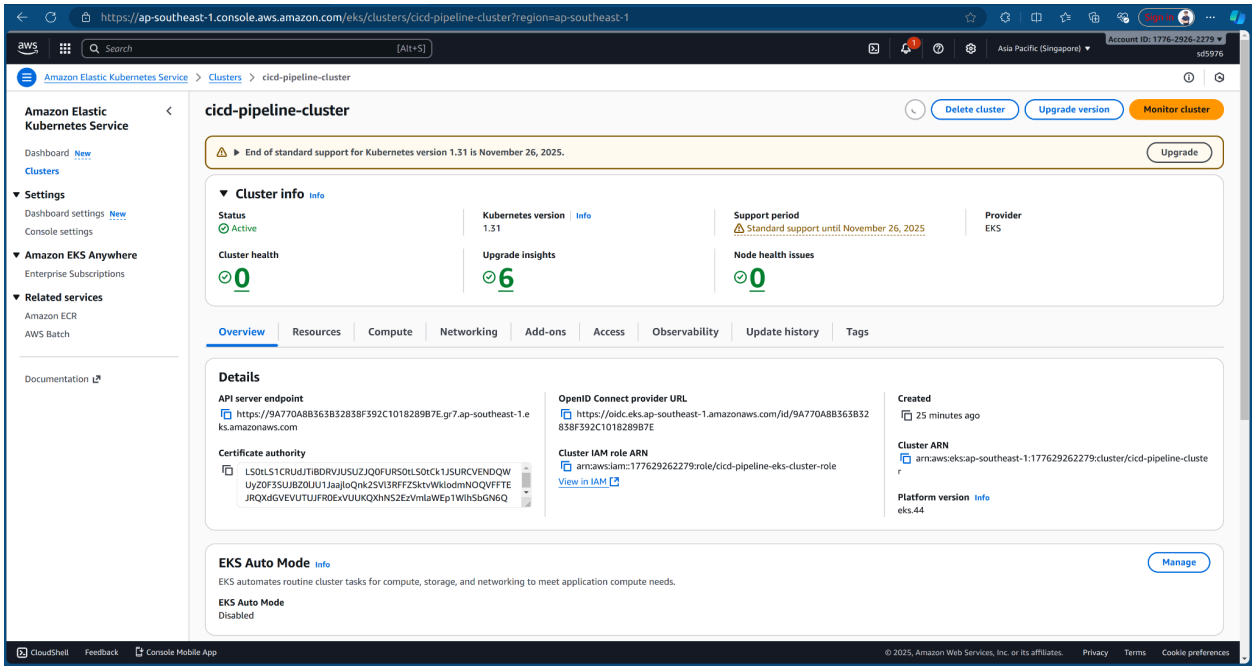


EC2


https://ap-southeast-1.console.aws.amazon.com/ec2/home?region=ap-southeast-1#Instances:v=3;$case=tags:true%5C,client:false;$regex=tags:false%5C,client:false
Search
[Alt+S]
Asia Pacific (Singapore)
Account ID: 1776-2926-2279
sd5976
EC2 > Instances
EC2
Dashboard
AWS Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Capacity Manager New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Instances (4) Info
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
Public IPv4 ...
Elastic IP
i-0e95090e556111d12 Running t3.small 3/3 checks passed View alarms ap-southeast-1a – – –
cicd-pipeline-jenkins i-0df7e2ca8a201da9e Running c7i-flex.large 3/3 checks passed View alarms ap-southeast-1a ec2-13-228-28-131.ap-... 13.228.28.131 –
i-00b30d2c405854abc Running t3.small 3/3 checks passed View alarms ap-southeast-1b – – –
cicd-pipeline-docker i-050419e902f16505a Running t3.small 3/3 checks passed View alarms ap-southeast-1b ec2-47-128-237-198.ap... 47.128.237.198 –
Select an instance
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

EKS

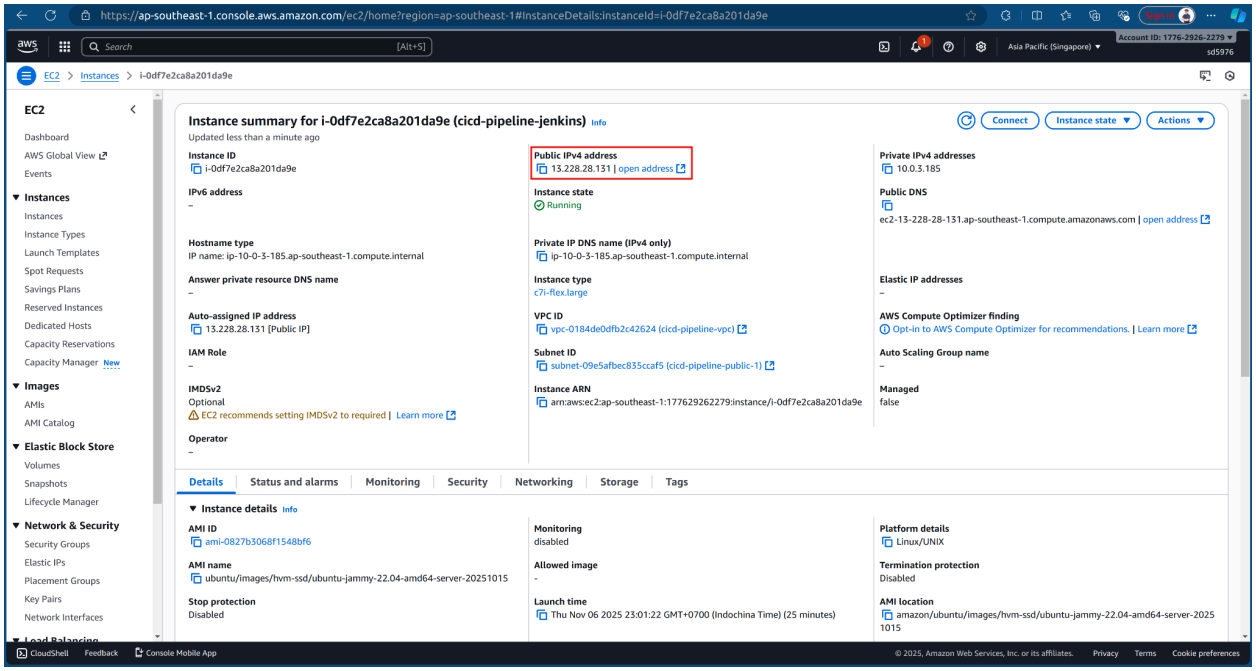

https://ap-southeast-1.console.aws.amazon.com/eks/clusters?region=ap-southeast-1
Search
[Alt+S]
Asia Pacific (Singapore)
Account ID: 1776-2926-2279
sd5976
Amazon Elastic Kubernetes Service > Clusters
Amazon Elastic Kubernetes Service
Dashboard New
Clusters
Settings
Dashboard settings New
Console settings
Amazon EKS Anywhere
Enterprise Subscriptions
Related services
Amazon ECR
AWS Batch
Documentation
Clusters (1) Info
Delete
Create cluster
Filter clusters
Cluster name
Status
Kubernetes version
Support period
Upgrade policy
Created
Provider
cicd-pipeline-cluster Active 1.31 Upgrade now Standard support until November 26, 2025 Extended support 24 minutes ago EKS
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

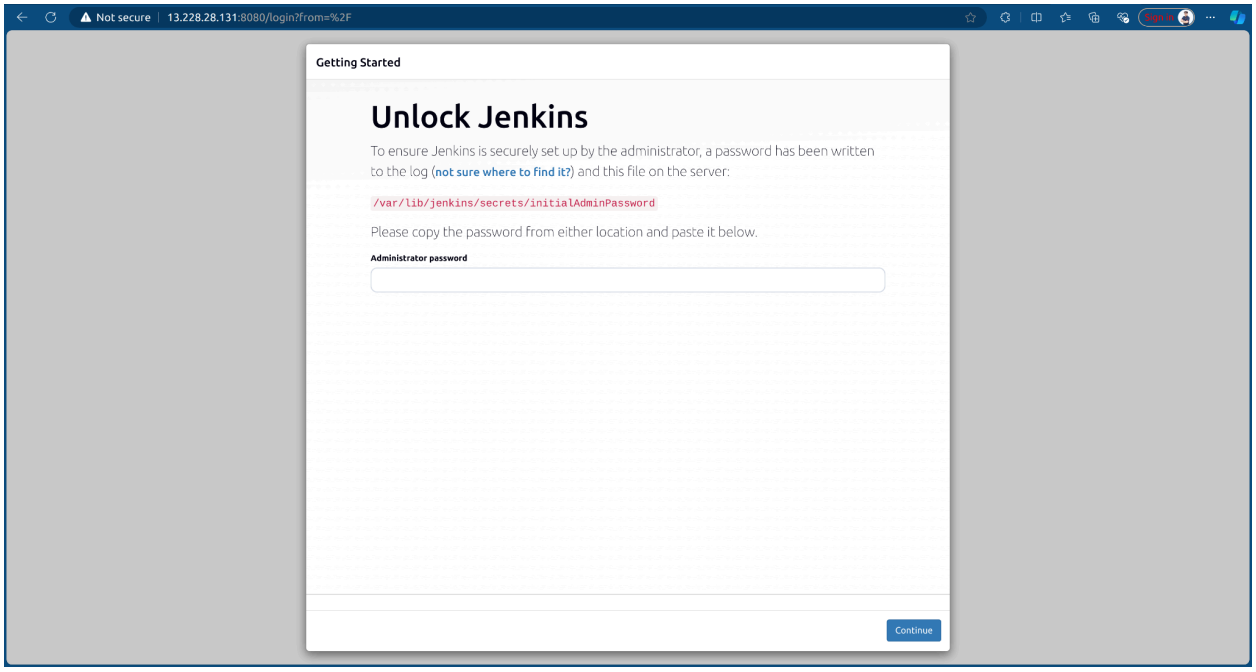

https://ap-southeast-1.console.aws.amazon.com/eks/clusters/cicd-pipeline-cluster?region=ap-southeast-1
Search
[Alt+S]
Asia Pacific (Singapore)
Account ID: 1776-2926-2279
sd5976
Amazon Elastic Kubernetes Service
Clusters
cicd-pipeline-cluster
Amazon Elastic Kubernetes Service
Dashboard New
Clusters
Settings
Dashboard settings New
Console settings
Amazon EKS Anywhere
Enterprise Subscriptions
Related services
Amazon ECR
AWS Batch
Documentation
cicd-pipeline-cluster
Delete cluster
Upgrade version
Monitor cluster
End of standard support for Kubernetes version 1.31 is November 26, 2025.
Upgrade
Cluster info Info
Status
Active
Kubernetes version Info
1.31
Support period
Standard support until November 26, 2025
Provider
EKS
Cluster health
0
Upgrade insights
6
Node health issues
0
Overview
Resources
Compute
Networking
Add-ons
Access
Observability
Update history
Tags
Details
API server endpoint
https://9A770A8B363B32838F392C1018289B7E.gr7.ap-southeast-1.eks.amazonaws.com
Certificate authority
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJU1JaajloQnk2SVl3RFFZSktvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQXhNS2EzVmlaWEp1WlhSbGN6Q
OpenID Connect provider URL
https://oidc.eks.ap-southeast-1.amazonaws.com/id/9A770A8B363B32838F392C1018289B7E
Cluster IAM role ARN
arn:aws:iam::177629262279:role/cicd-pipeline-eks-cluster-role
View in IAM
Created
25 minutes ago
Cluster ARN
arn:aws:eks:ap-southeast-1:177629262279:cluster/cicd-pipeline-cluster
Platform version Info
eks.44
EKS Auto Mode Info
Manage
EKS automates routine cluster tasks for compute, storage, and networking to meet application compute needs.
EKS Auto Mode
Disabled
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

II. Access Jenkins

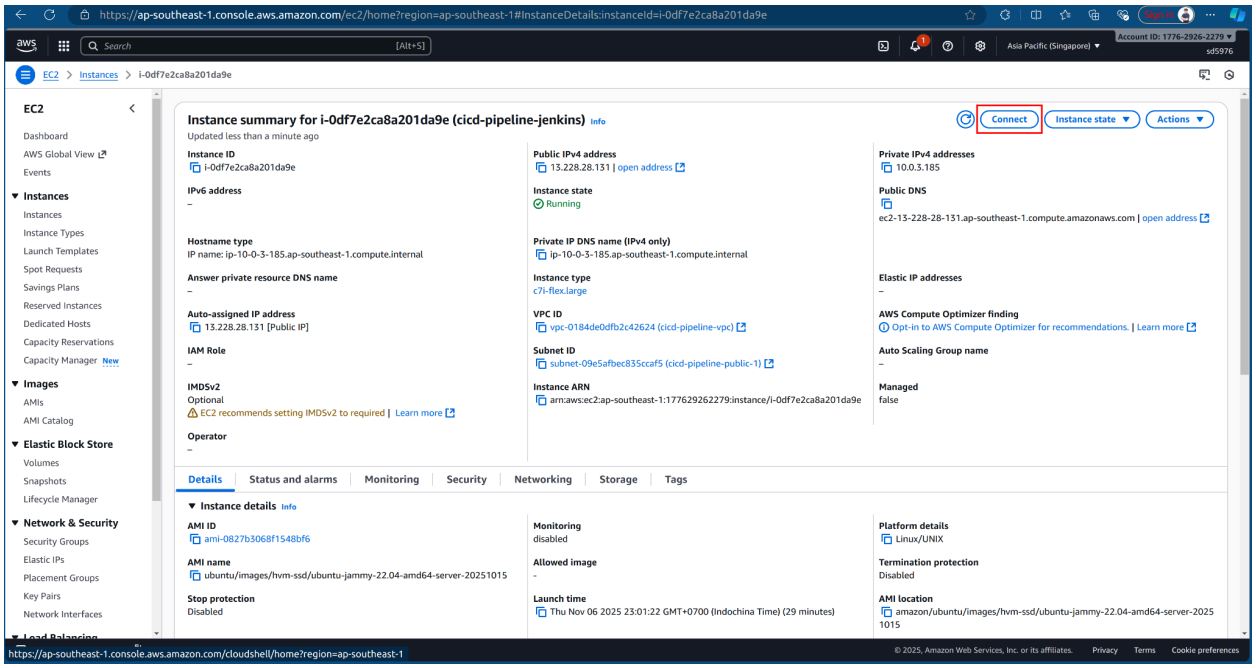
Terraform has already created an EC2 instance with Jenkins installed. Access Jenkins at:

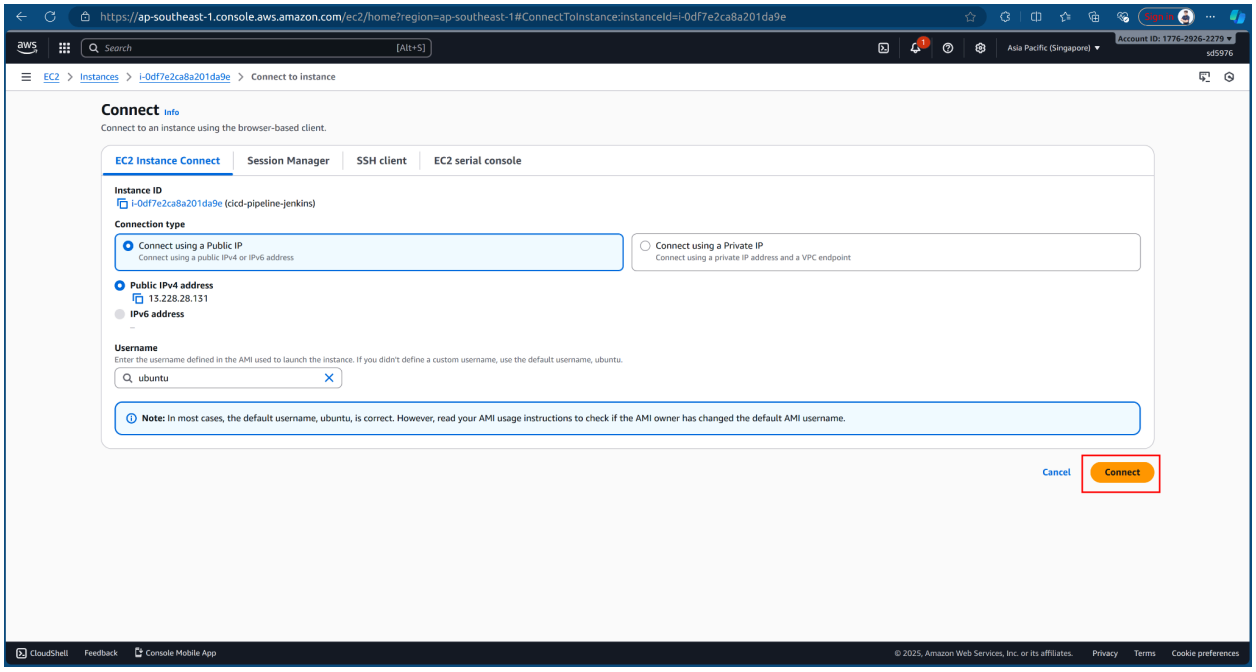
http://13.228.28.131:8080/





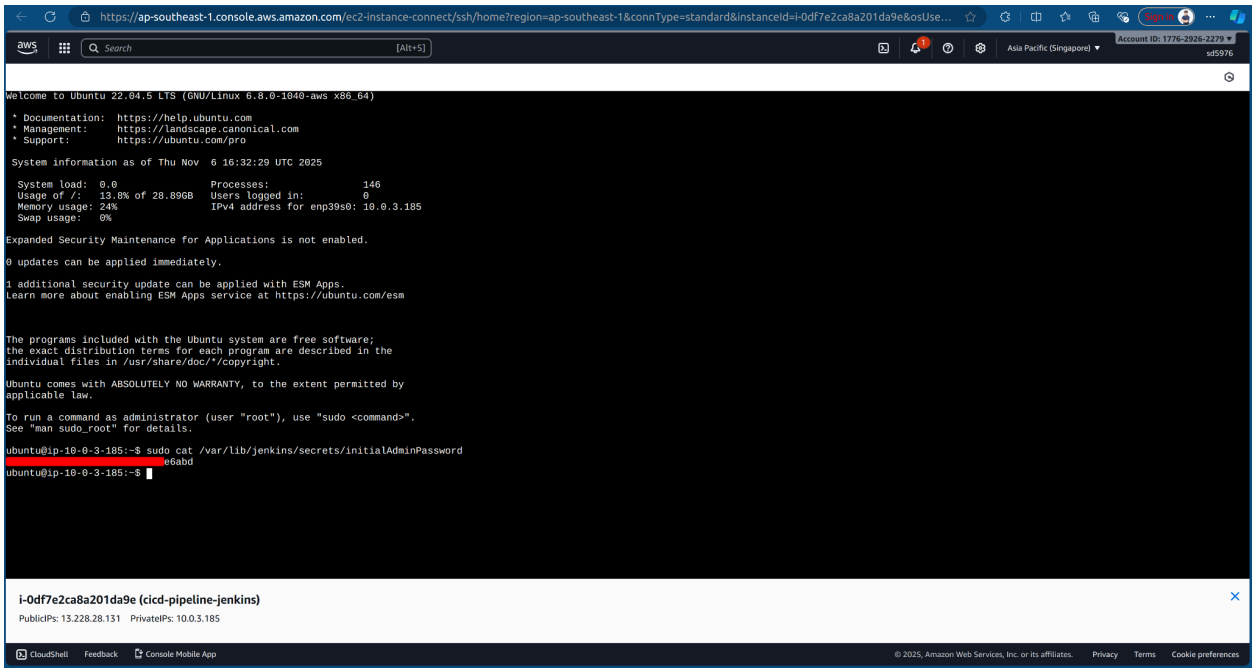
To get the initialAdminPassword, we need to connect to the EC2 instance **cicd-pipeline-jenkins**.



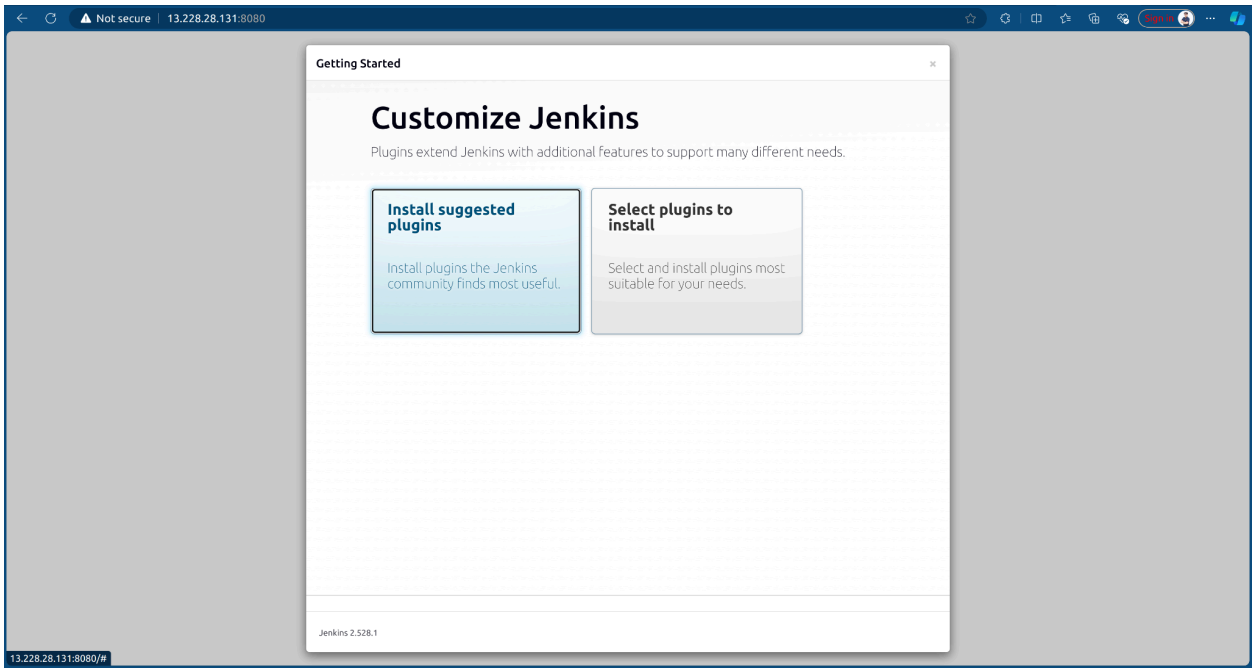


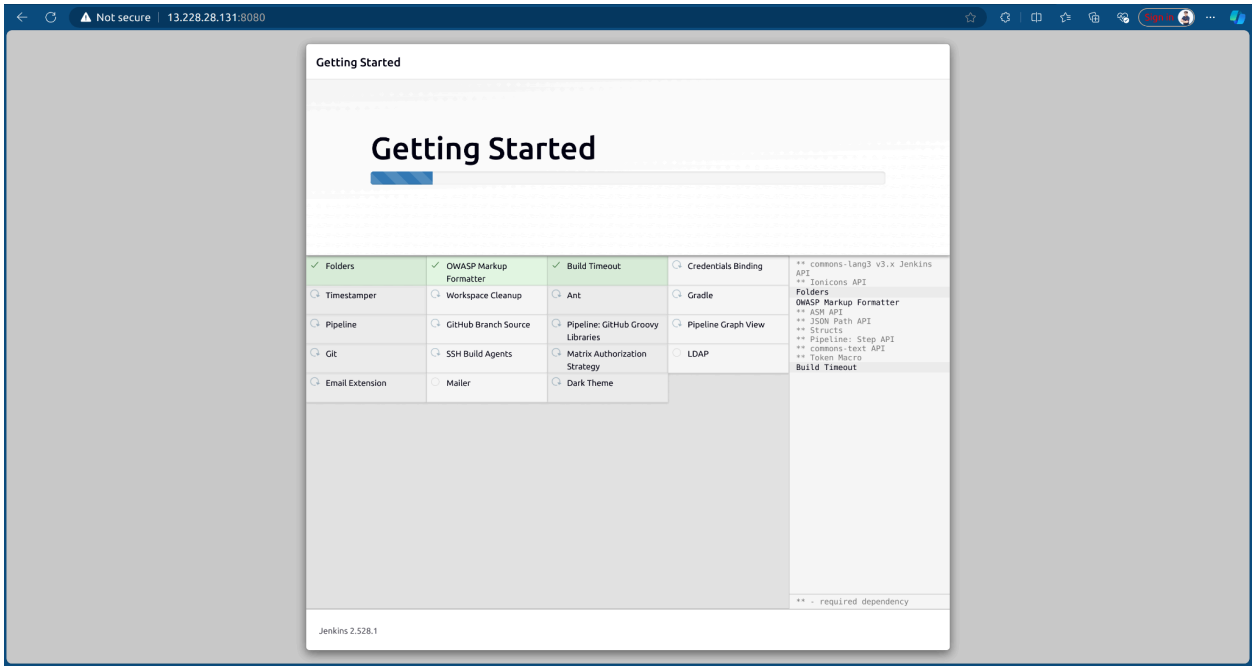
Retrieve the initial admin password:

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

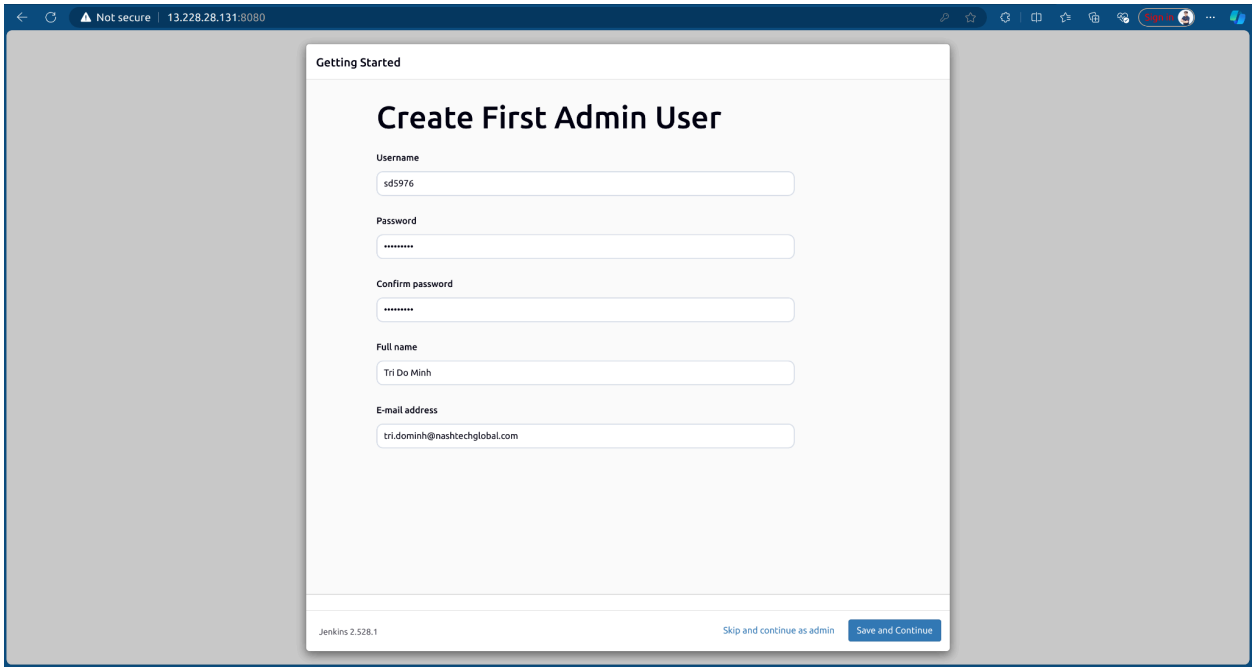


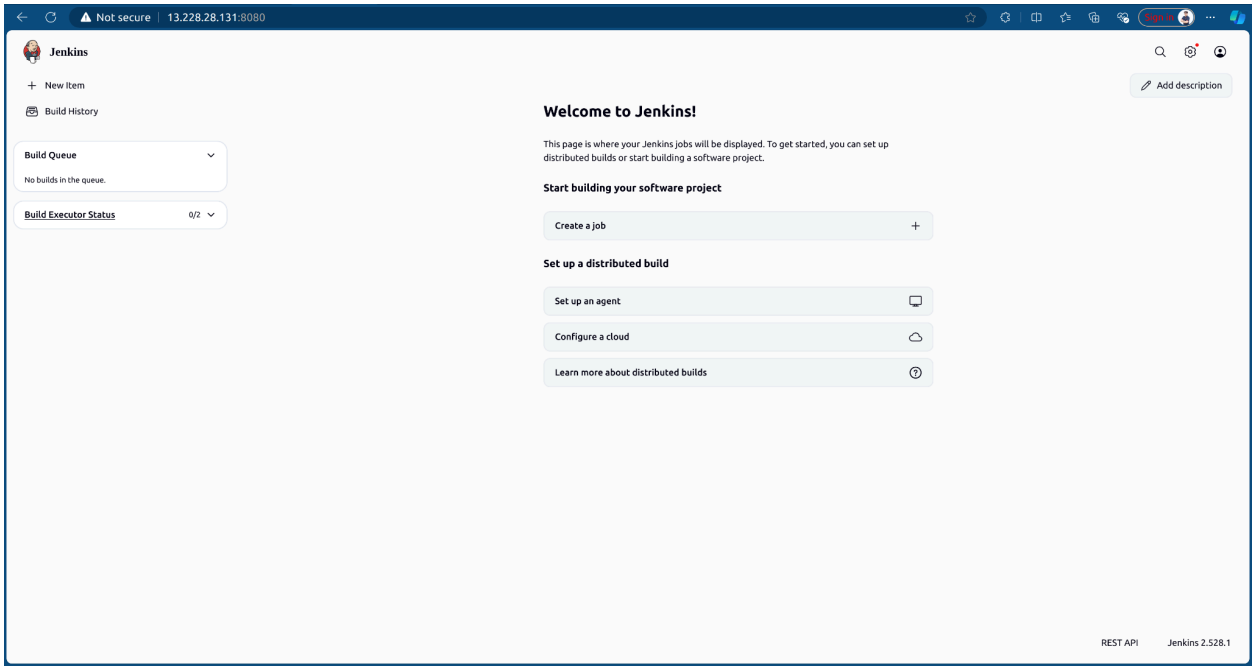
Copy the password and paste it into Jenkins UI, then select **Install suggested plugins**.





After installation, create an Admin User to log in to the dashboard.

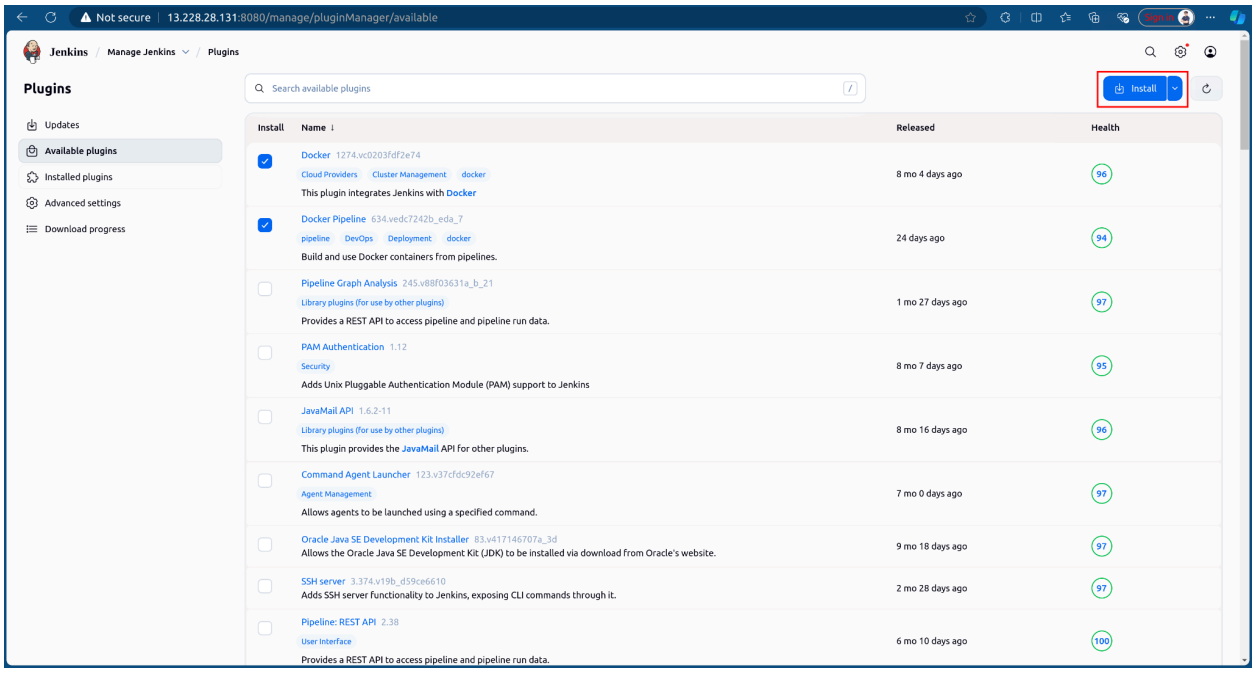




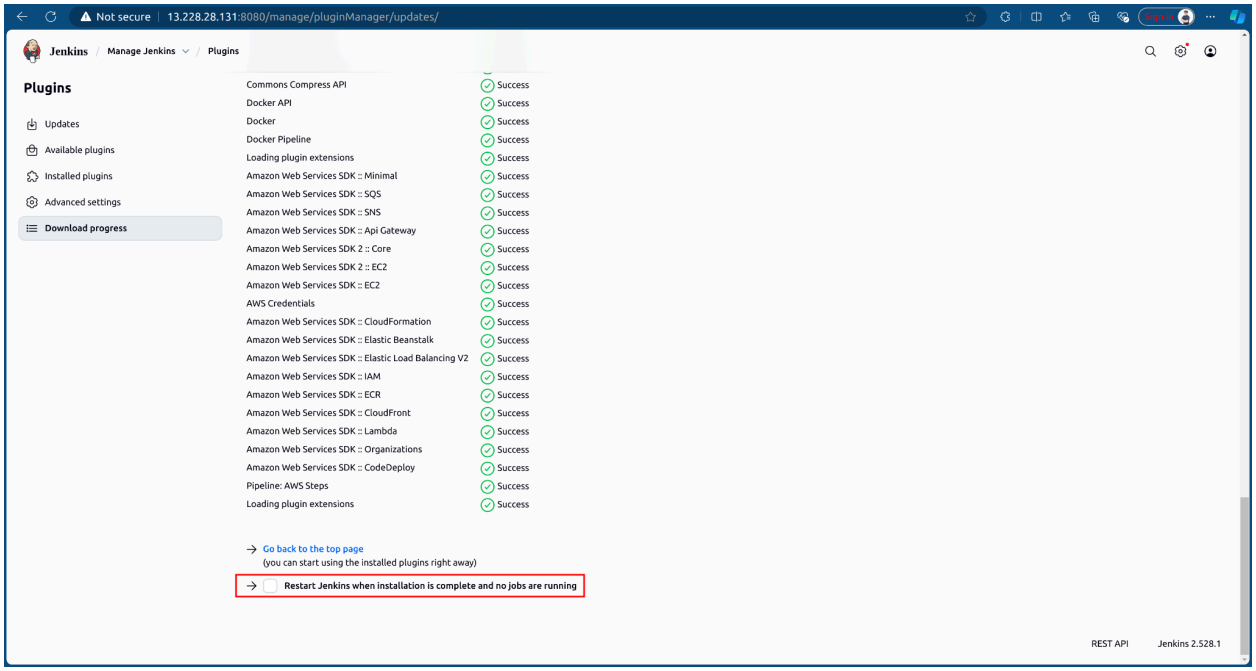
To run CI/CD, we need to install some plugins:

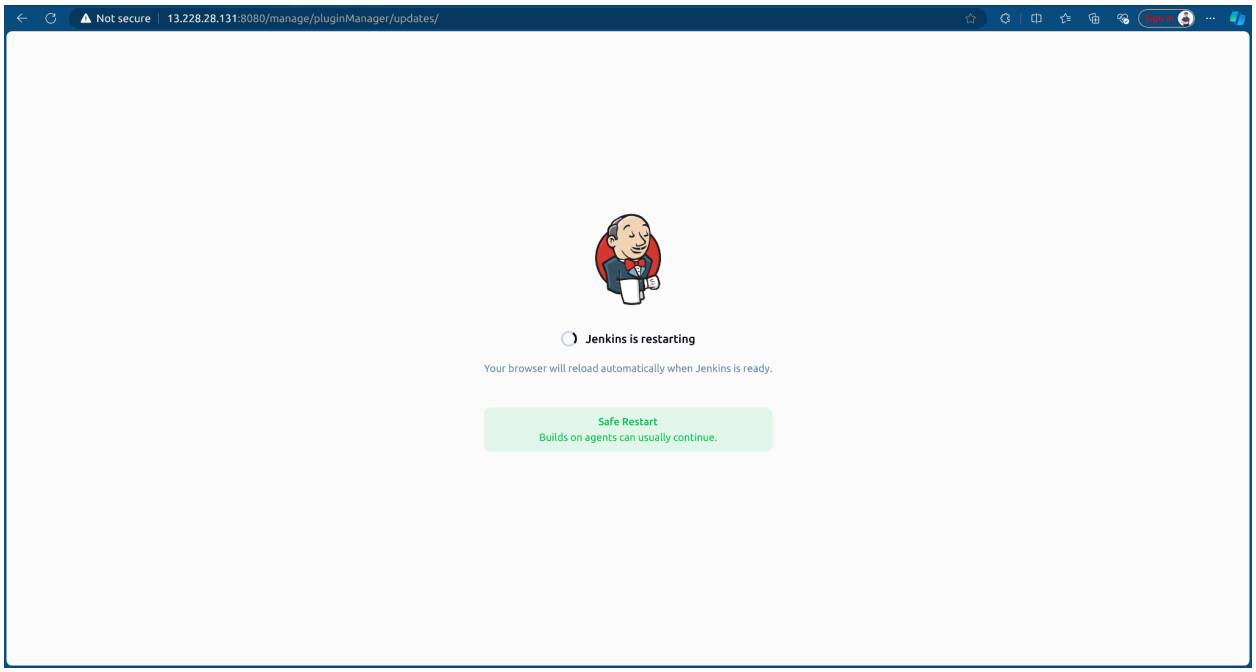
- Docker
- Docker Pipeline
- Pipeline: AWS Steps

Go to **Manage Jenkins → Plugins → Available plugins** tab.



Restart Jenkins after plugin installation.

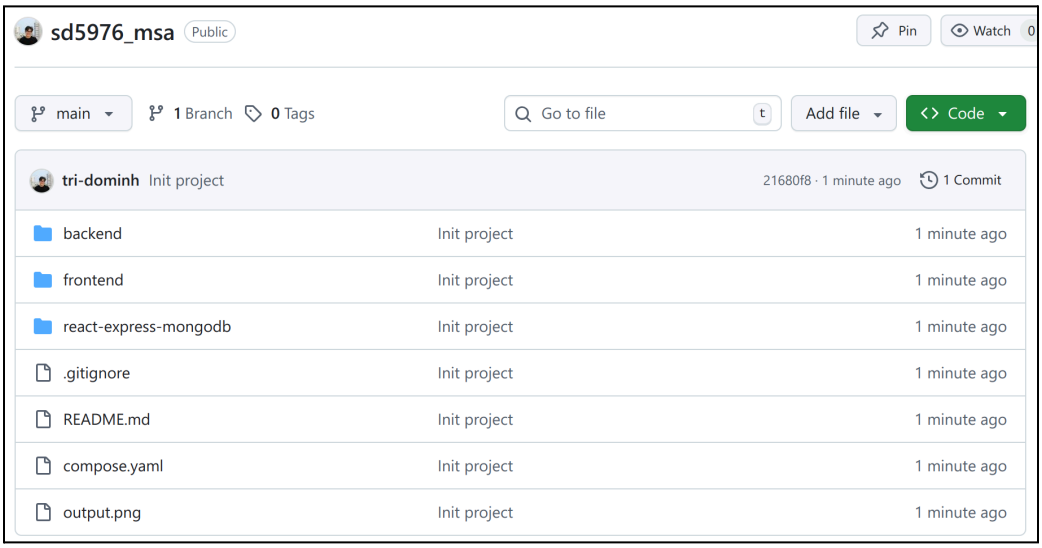




III. Deploying the MSA Application to EKS with Jenkins CI/CD

Prepare the MSA repository containing *frontend* and *backend*:

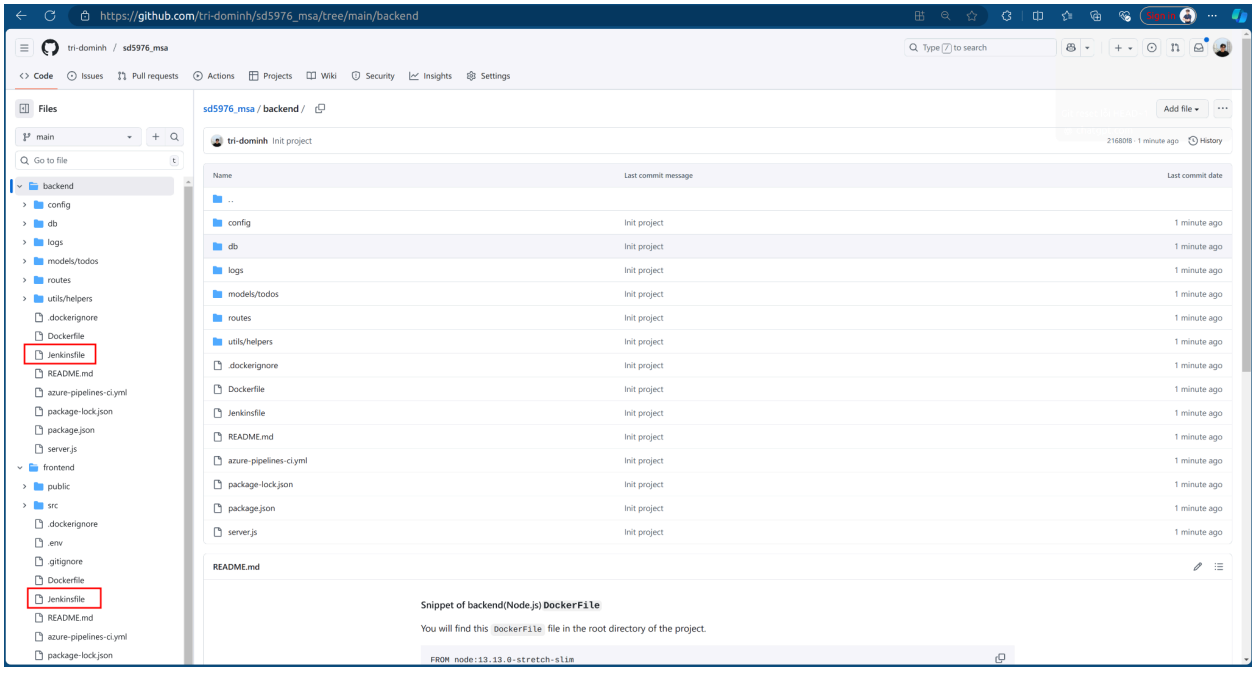
https://github.com/tri-dominh/sd5976_msa



Each source (*frontend*/*backend*) should have a **Jenkinsfile** defining agent and stages: install, build, validate, test, and push to ECR.

https://github.com/tri-dominh/sd5976_msa/blob/main/backend/Jenkinsfile

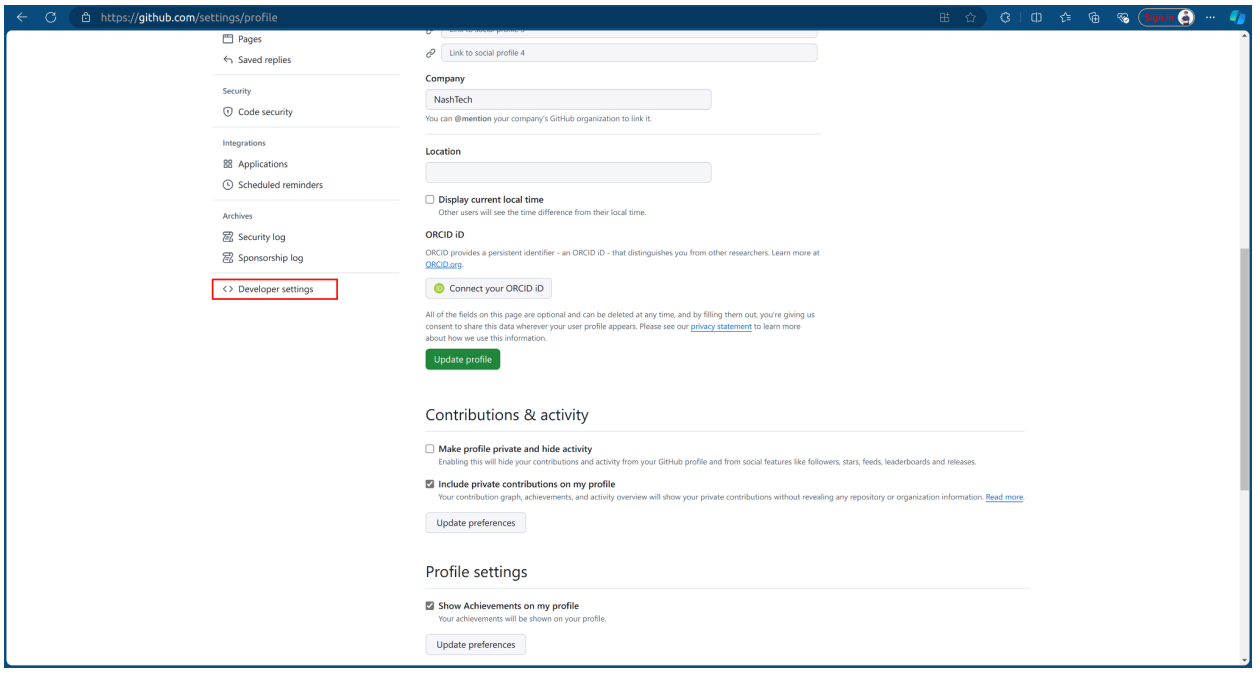
https://github.com/tri-dominh/sd5976_msa/blob/main/frontend/Jenkinsfile

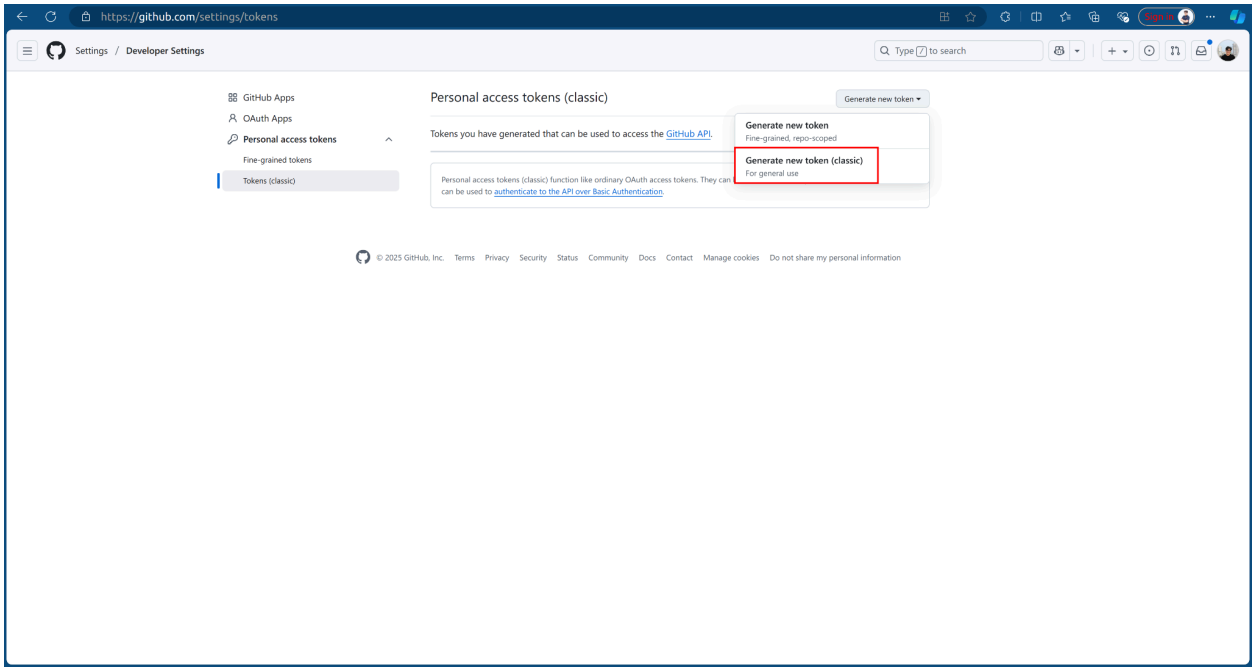


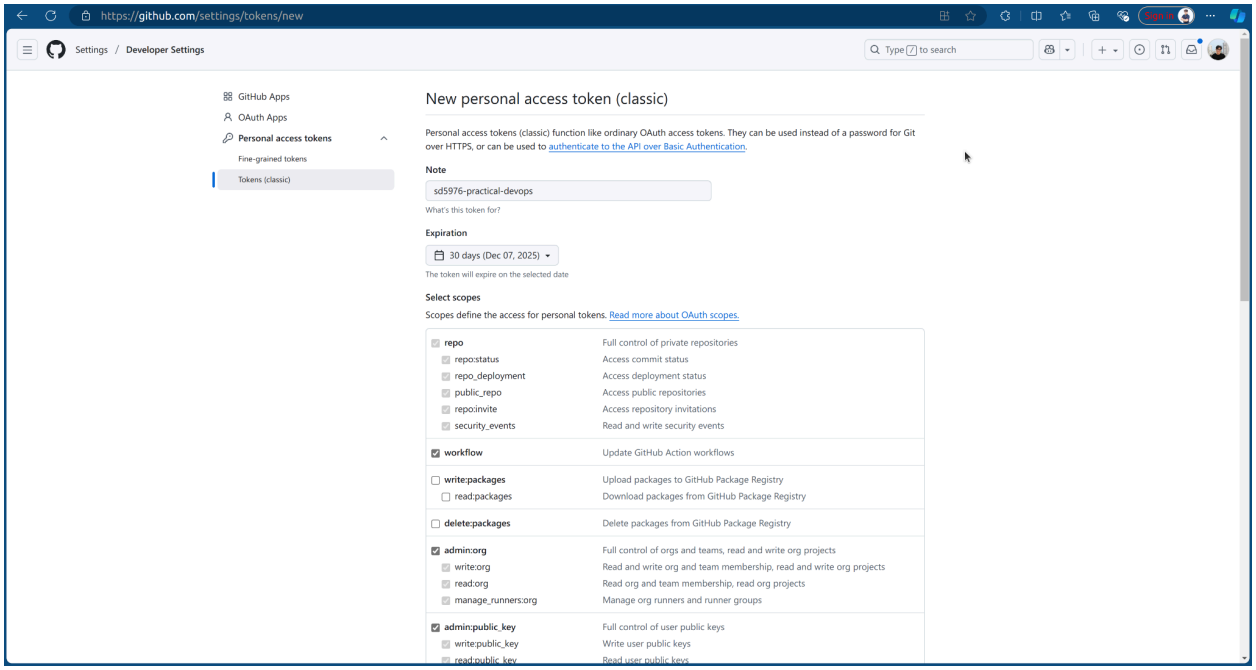
Next, we need to configure Jenkins to access GitHub and AWS resources.

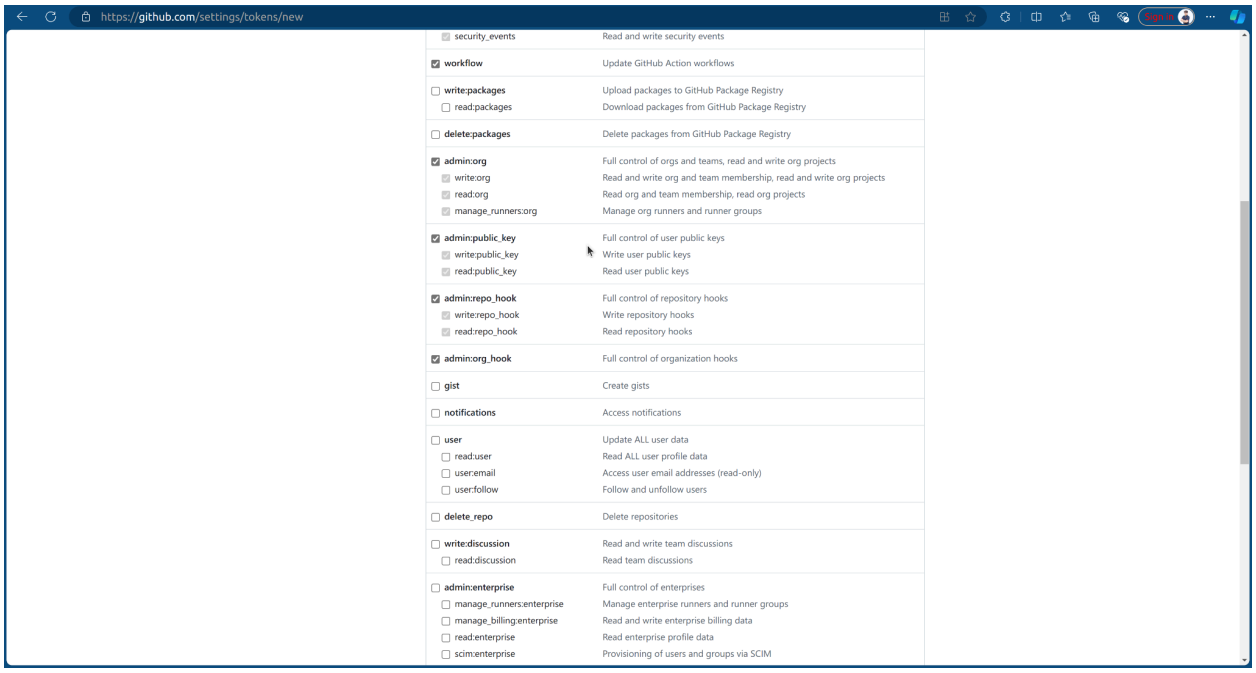
Create a GitHub Access Token for Jenkins to access the MSA repository.

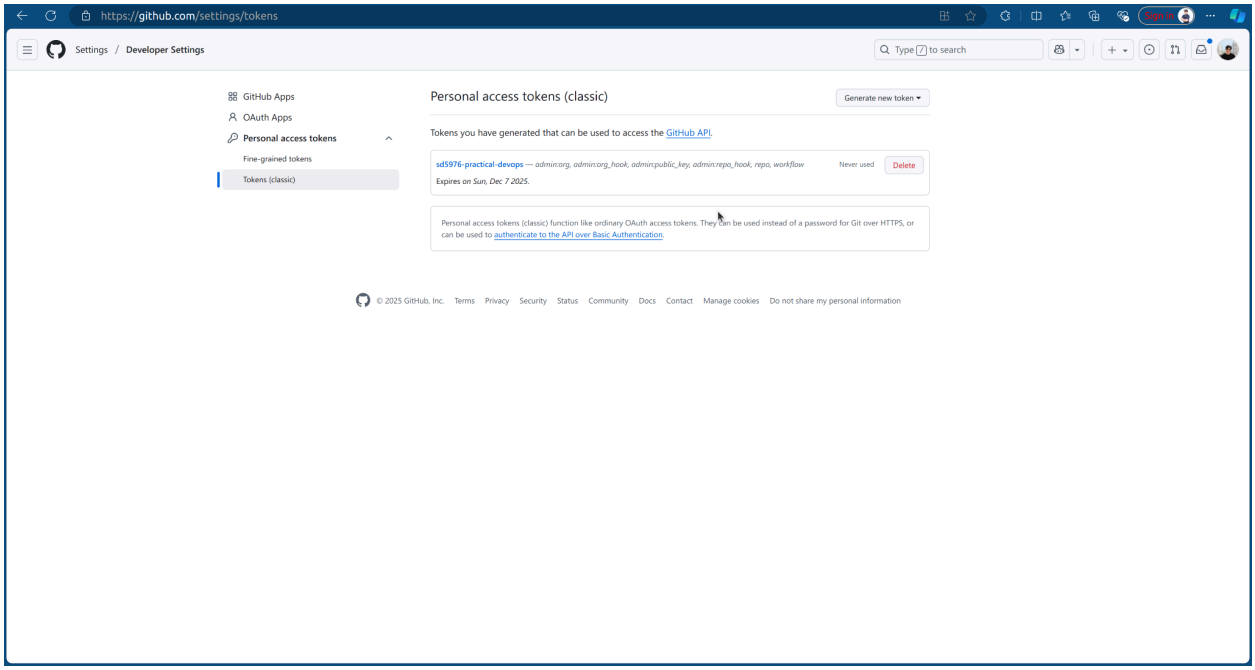
Go to **Settings → Developer settings → Personal access tokens → Tokens (classic) → Generate new token (classic)** → Enter token name and select scopes to allow Jenkins access to GitHub → Create.

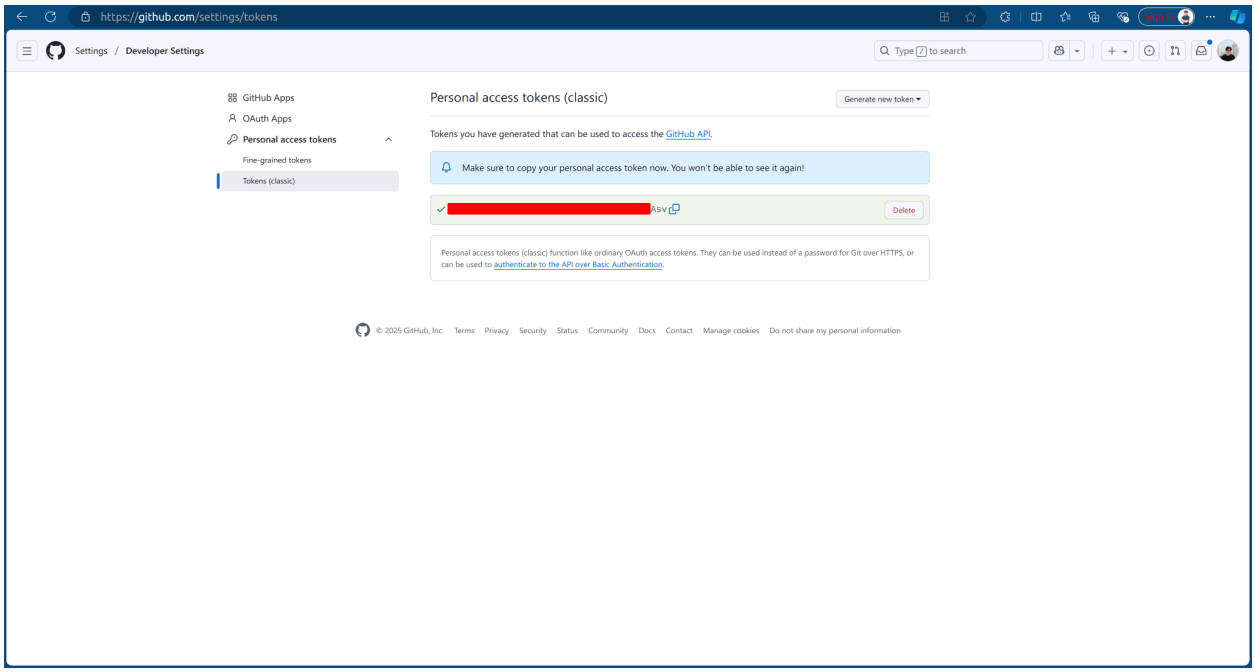




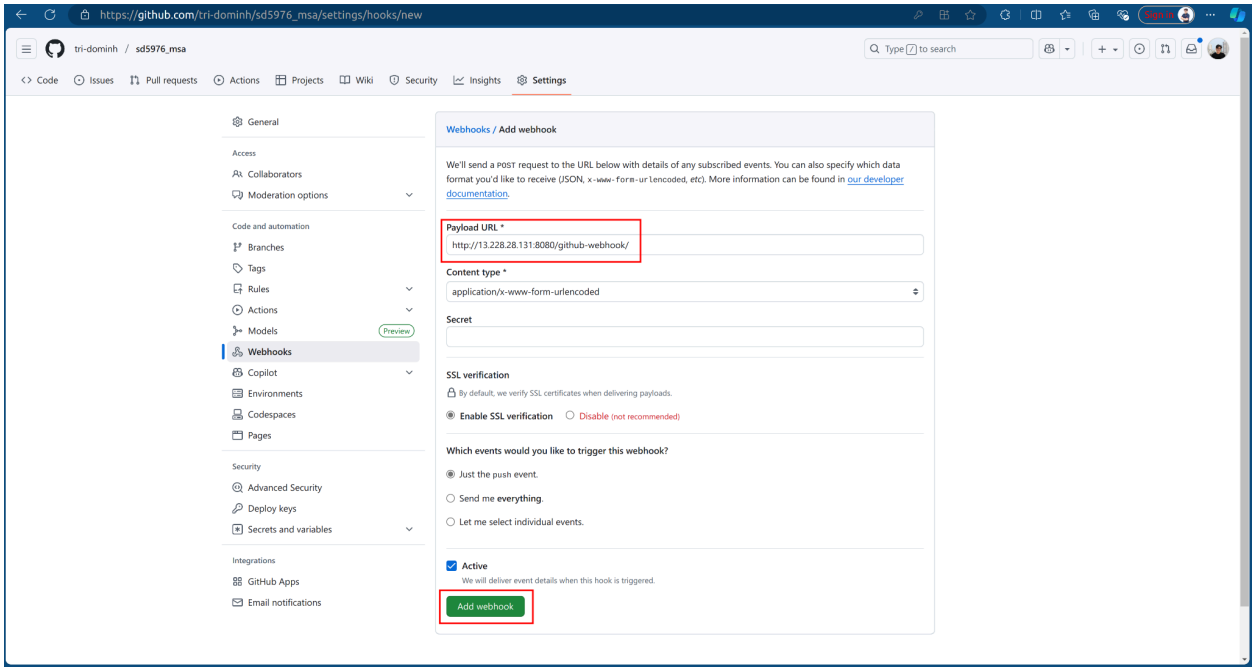

https://github.com/settings/tokens/new
Settings / Developer Settings
Type / to search
GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens
Tokens (classic)
New personal access token (classic)
Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.
Note
sd5976-practical-devops
What's this token for?
Expiration
30 days (Dec 07, 2025)
The token will expire on the selected date
Select scopes
Scopes define the access for personal tokens. Read more about OAuth scopes.
repo Full control of private repositories
repo:status Access commit status
repo_deployment Access deployment status
public_repo Access public repositories
repo:invite Access repository invitations
security_events Read and write security events
workflow Update GitHub Action workflows
write:packages Upload packages to GitHub Package Registry
read:packages Download packages from GitHub Package Registry
delete:packages Delete packages from GitHub Package Registry
admin:org Full control of orgs and teams, read and write org projects
write:org Read and write org and team membership, read and write org projects
read:org Read org and team membership, read org projects
manage_runners:org Manage org runners and runner groups
admin:public_key Full control of user public keys
write:public_key Write user public keys
read:public_key Read user public keys


https://github.com/settings/tokens/new
security_events Read and write security events
workflow Update GitHub Action workflows
write:packages Upload packages to GitHub Package Registry
read:packages Download packages from GitHub Package Registry
delete:packages Delete packages from GitHub Package Registry
admin:org Full control of orgs and teams, read and write org projects
write:org Read and write org and team membership, read and write org projects
read:org Read org and team membership, read org projects
manage_runners:org Manage org runners and runner groups
admin:public_key Full control of user public keys
write:public_key Write user public keys
read:public_key Read user public keys
admin:repo_hook Full control of repository hooks
write:repo_hook Write repository hooks
read:repo_hook Read repository hooks
admin:org_hook Full control of organization hooks
gist Create gists
notifications Access notifications
user Update ALL user data
read:user Read ALL user profile data
user:email Access user email addresses (read-only)
user:follow Follow and unfollow users
delete_repo Delete repositories
write:discussion Read and write team discussions
read:discussion Read team discussions
admin:enterprise Full control of enterprises
manage_runners:enterprise Manage enterprise runners and runner groups
manage_billing:enterprise Read and write enterprise billing data
read:enterprise Read enterprise profile data
scim:enterprise Provisioning of users and groups via SCIM


https://github.com/settings/tokens
Settings / Developer Settings
Type / to search
GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens
Tokens (classic)
Personal access tokens (classic)
Generate new token
Tokens you have generated that can be used to access the GitHub API.
sd5976-practical-devops — admin:org, admin:org_hook, admin:public_key, admin:repo_hook, repo, workflow
Never used
Delete
Expires on Sun, Dec 7 2025.
Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.
© 2025 GitHub, Inc. Terms Privacy Security Status Community Docs Contact Manage cookies Do not share my personal information

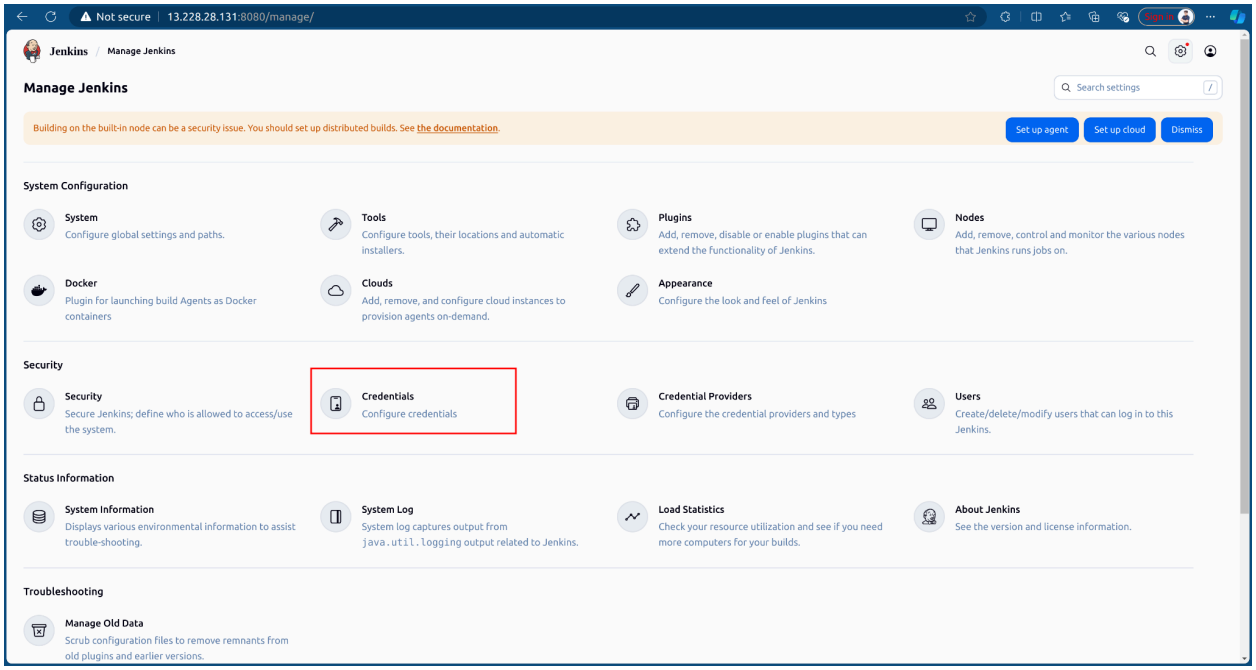

https://github.com/settings/tokens
Settings / Developer Settings
Type / to search
GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens
Tokens (classic)
Personal access tokens (classic)
Generate new token
Tokens you have generated that can be used to access the GitHub API.
Make sure to copy your personal access token now. You won't be able to see it again!
[redacted]Asv
Delete
Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.
© 2025 GitHub, Inc. Terms Privacy Security Status Community Docs Contact Manage cookies Do not share my personal information

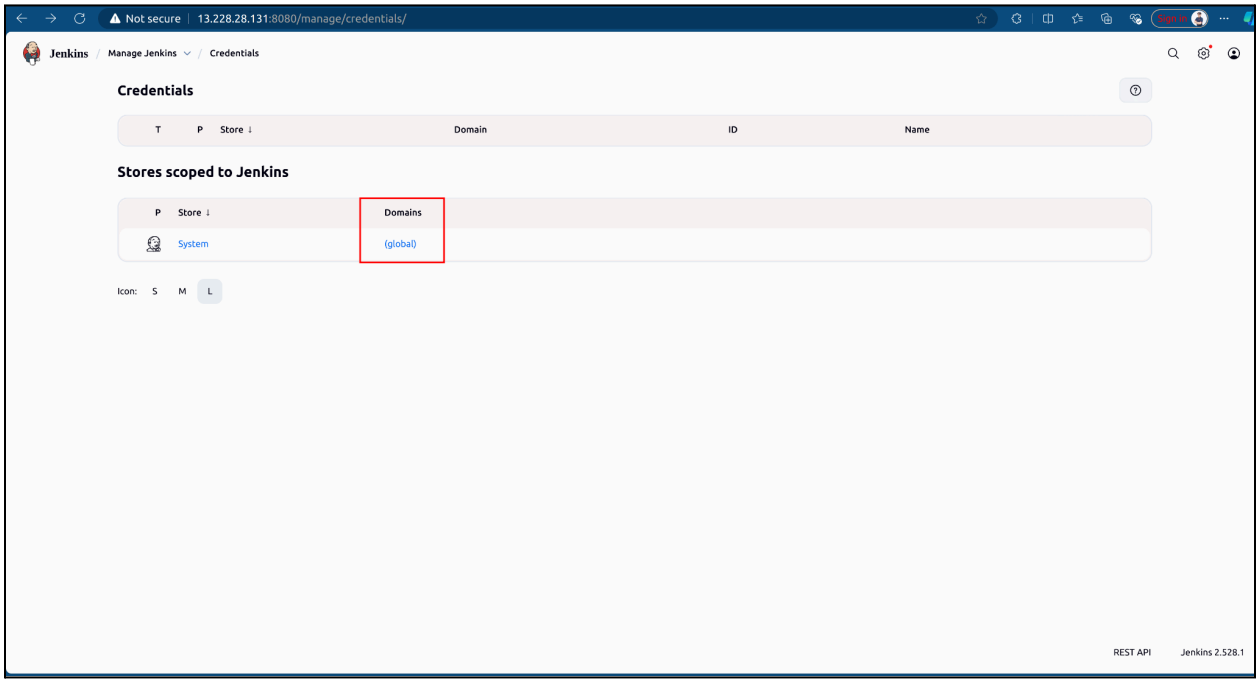
Then, we create a Github webhook to trigger Jenkins whenever we push new code. In the **sd5976-msa** repo, go to **Settings → Webhooks → Add webhook**, enter the Jenkins URL with */github-webhook/*

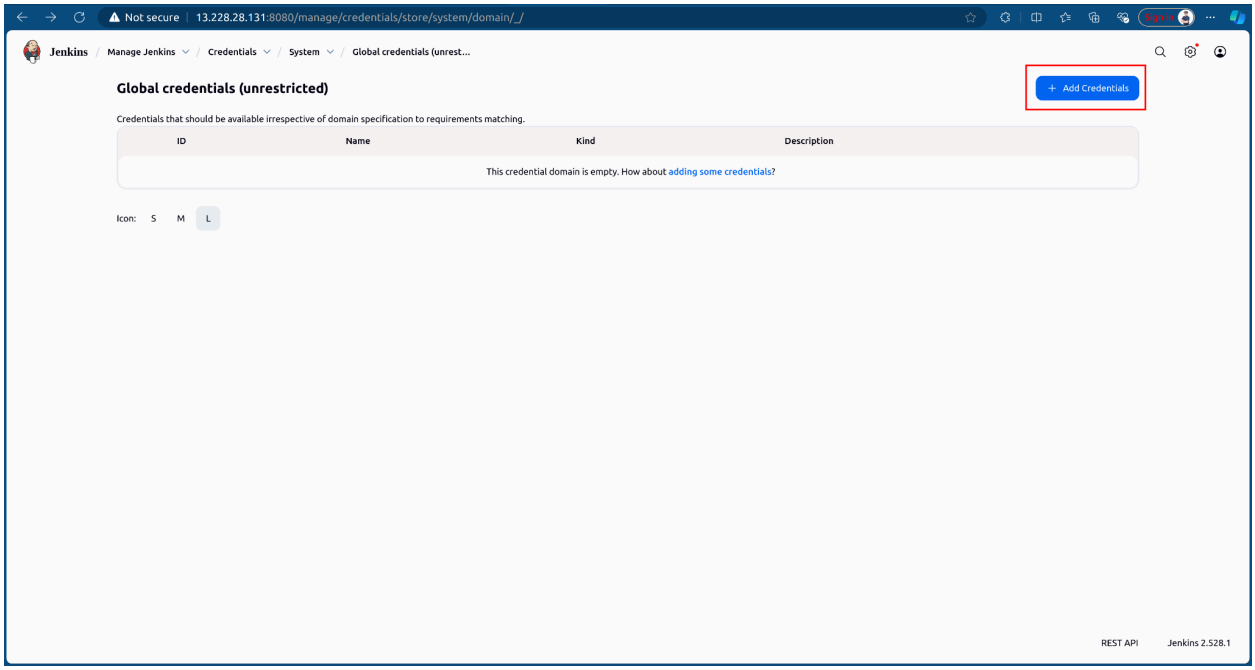


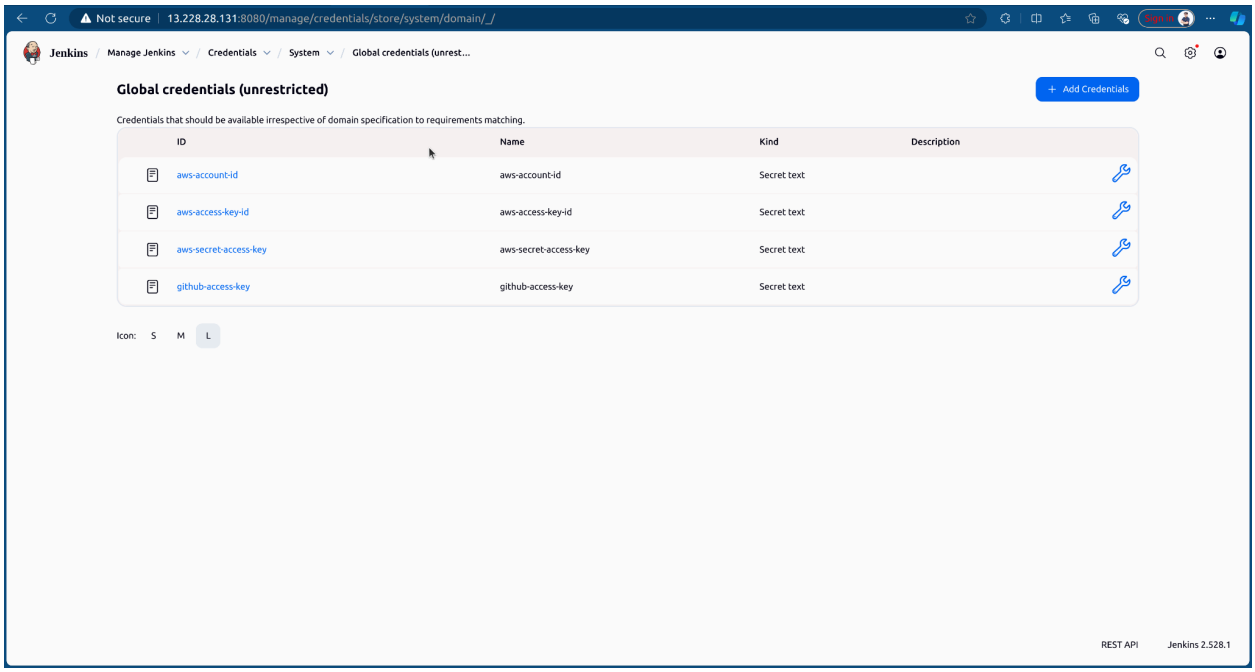
After that, we create the credentials in Jenkins which allow Jenkins access to Resources. Go to **Jenkins Credentials** → Click on **(global)** → **Add Credentials** (*Secret text* kind).

- AWS Account ID, AWS Access Key ID, AWS Secret Access Key.
- GitHub Token.

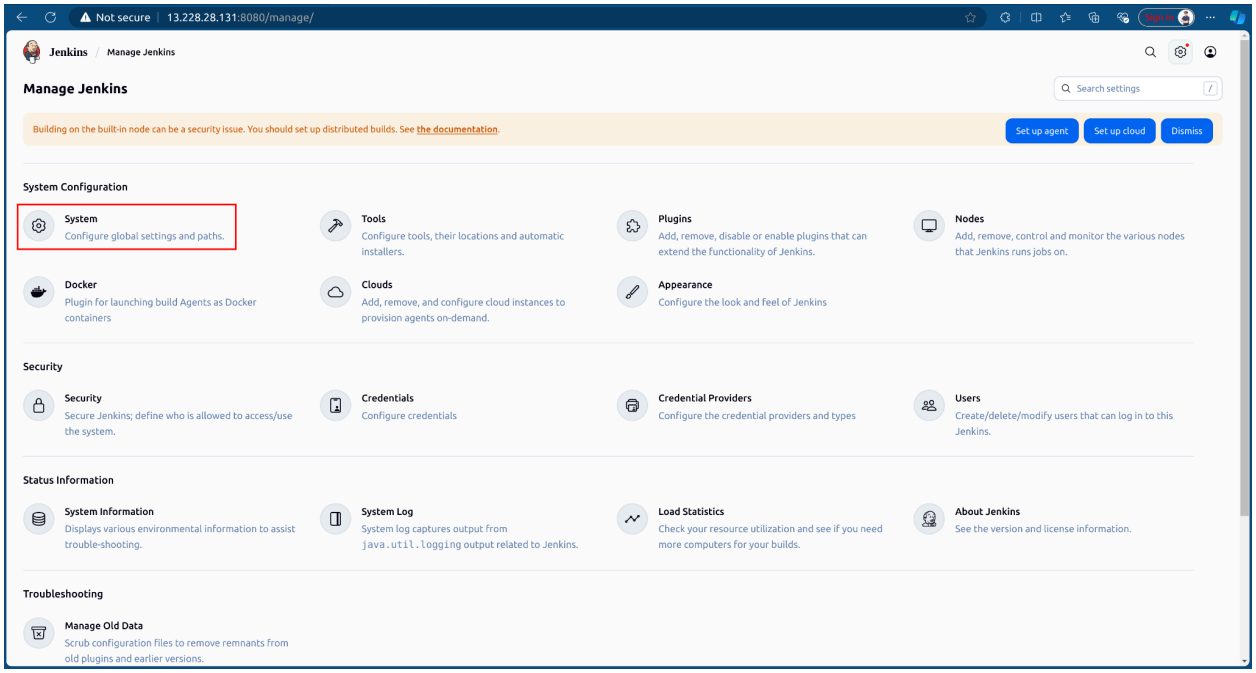

Not secure | 13.228.28.131:8080/manage/
Jenkins / Manage Jenkins
Manage Jenkins
Search settings
Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.
Set up agent
Set up cloud
Dismiss
System Configuration
System
Configure global settings and paths.
Tools
Configure tools, their locations and automatic installers.
Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
Docker
Plugin for launching build Agents as Docker containers
Clouds
Add, remove, and configure cloud instances to provision agents on-demand.
Appearance
Configure the look and feel of Jenkins
Security
Security
Secure Jenkins; define who is allowed to access/use the system.
Credentials
Configure credentials
Credential Providers
Configure the credential providers and types
Users
Create/delete/modify users that can log in to this Jenkins.
Status Information
System Information
Displays various environmental information to assist trouble-shooting.
System Log
System log captures output from java.util.logging output related to Jenkins.
Load Statistics
Check your resource utilization and see if you need more computers for your builds.
About Jenkins
See the version and license information.
Troubleshooting
Manage Old Data
Scrub configuration files to remove remnants from old plugins and earlier versions.

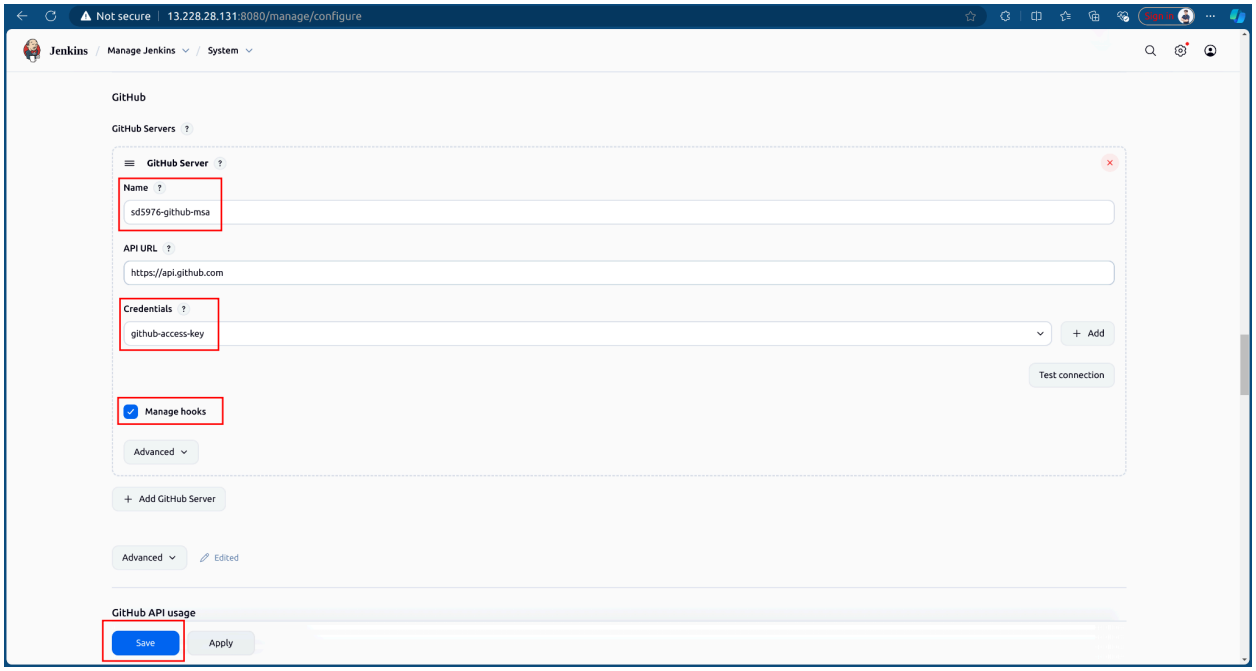

Not secure | 13.228.28.131:8080/manage/credentials/
Jenkins / Manage Jenkins / Credentials
Credentials
T
P
Store ↓
Domain
ID
Name
Stores scoped to Jenkins
P
Store ↓
Domains
System
(global)
Icon: S M L
REST API
Jenkins 2.528.1


Not secure | 13.228.28.131:8080/manage/credentials/store/system/domain/_/
Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrest...
Global credentials (unrestricted)
+ Add Credentials
Credentials that should be available irrespective of domain specification to requirements matching.
ID
Name
Kind
Description
This credential domain is empty. How about adding some credentials?
Icon: S M L
REST API
Jenkins 2.528.1


Not secure | 13.228.28.131:8080/manage/credentials/store/system/domain/_/
Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrest...
Global credentials (unrestricted)
+ Add Credentials
Credentials that should be available irrespective of domain specification to requirements matching.
ID
Name
Kind
Description
aws-account-id
aws-account-id
Secret text
aws-access-key-id
aws-access-key-id
Secret text
aws-secret-access-key
aws-secret-access-key
Secret text
github-access-key
github-access-key
Secret text
Icon: S M L
REST API
Jenkins 2.528.1

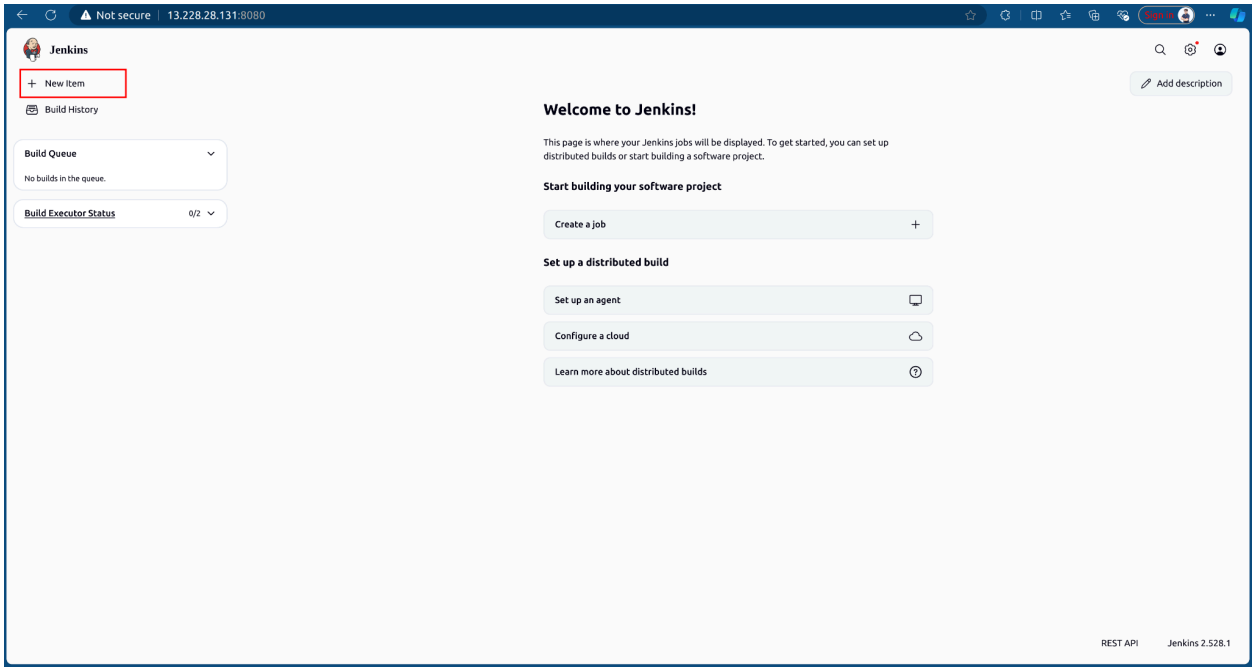
Add GitHub Server for Jenkins, go to **Settings** → **System** → In **GitHub** section → **Add GitHub Server** → Enter name, select credentials **github-access-key** and tick **Manage hooks** → Save.

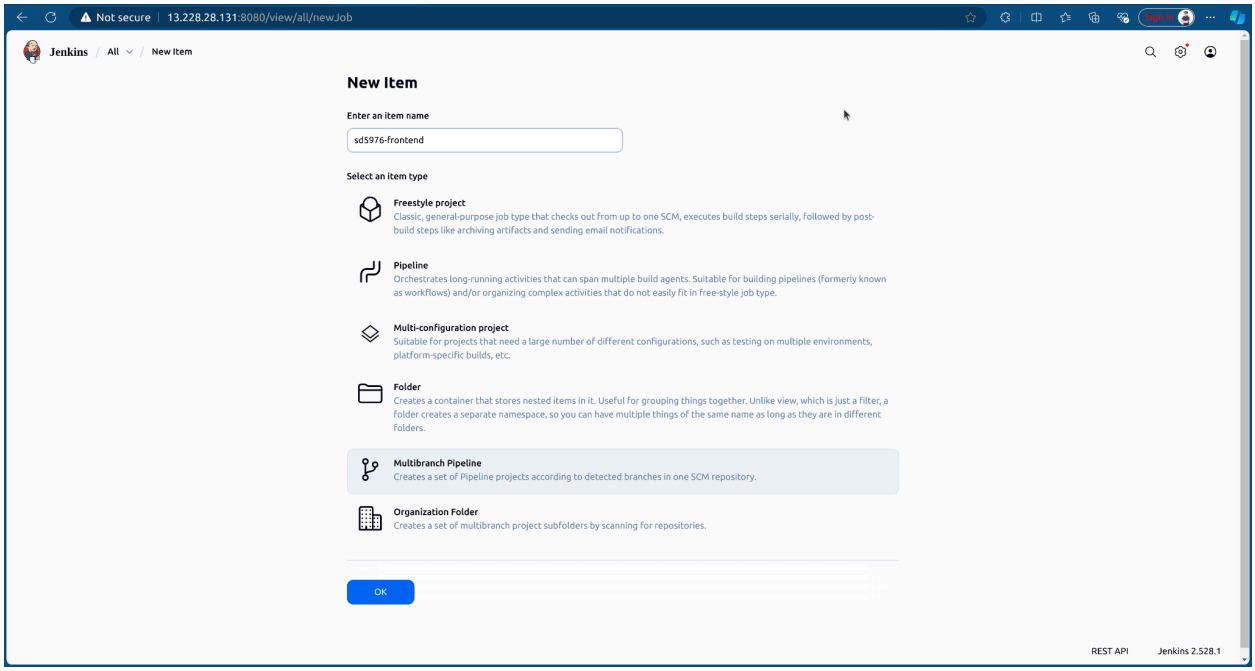




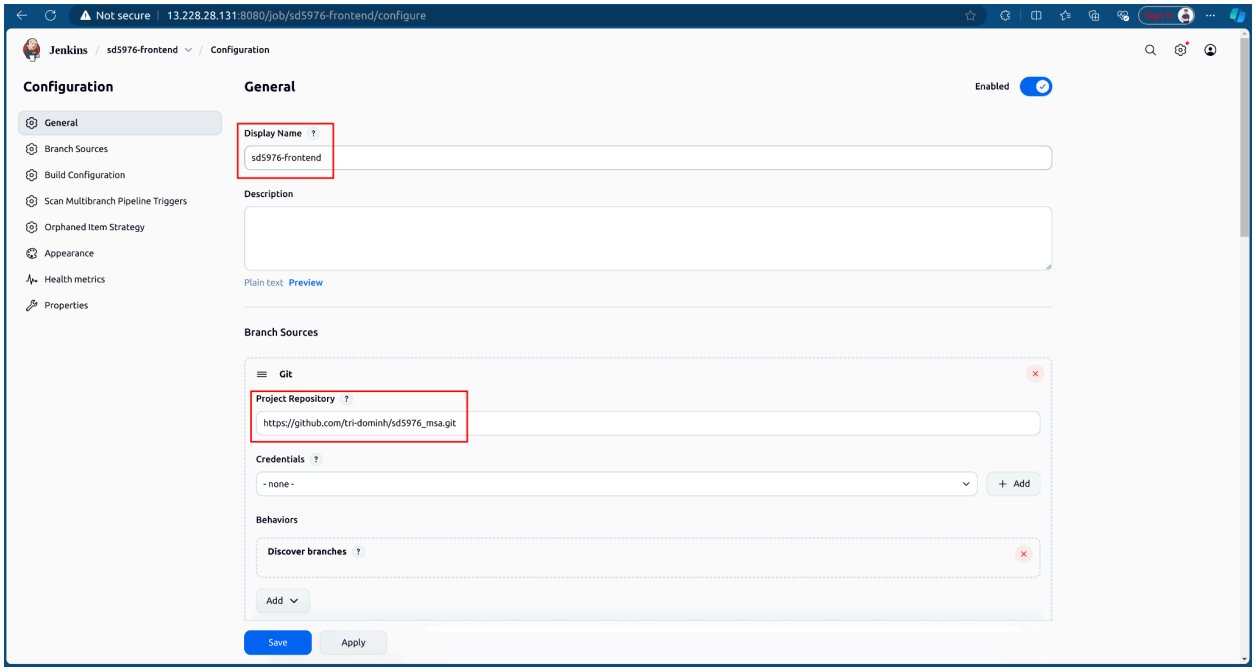
IV. Creating CI jobs for MSA Application

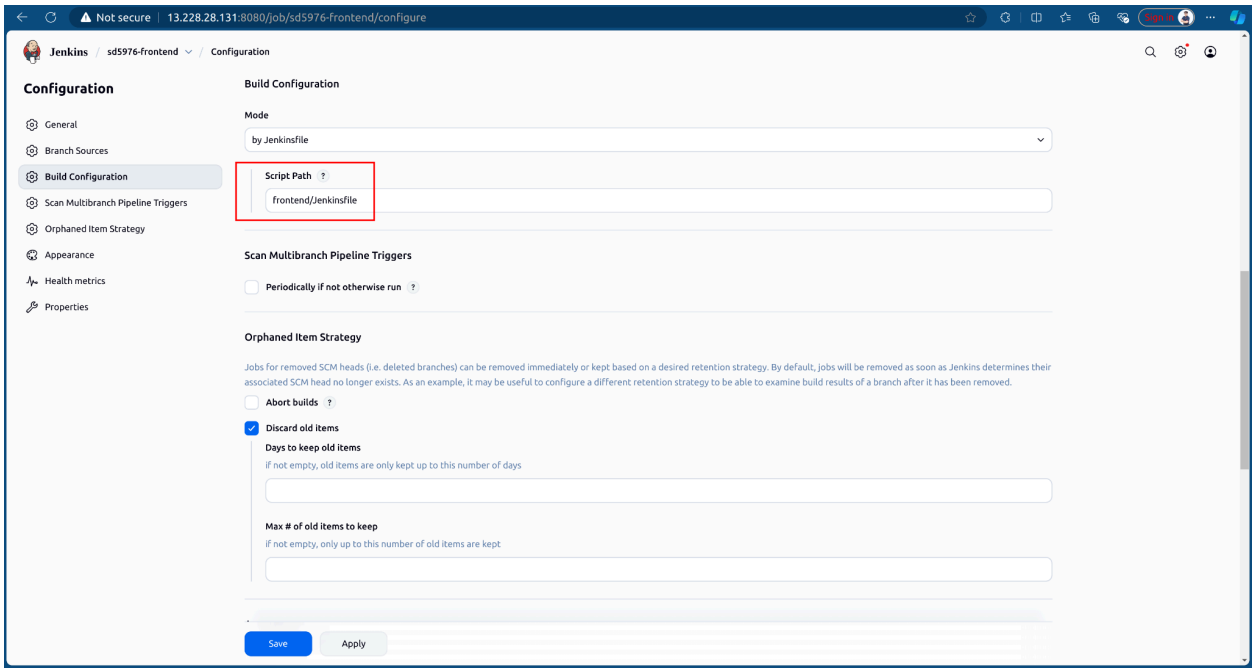
First, we build a job for the *Frontend*, click **New item** → Enter name and select **Multibranch pipeline** job.



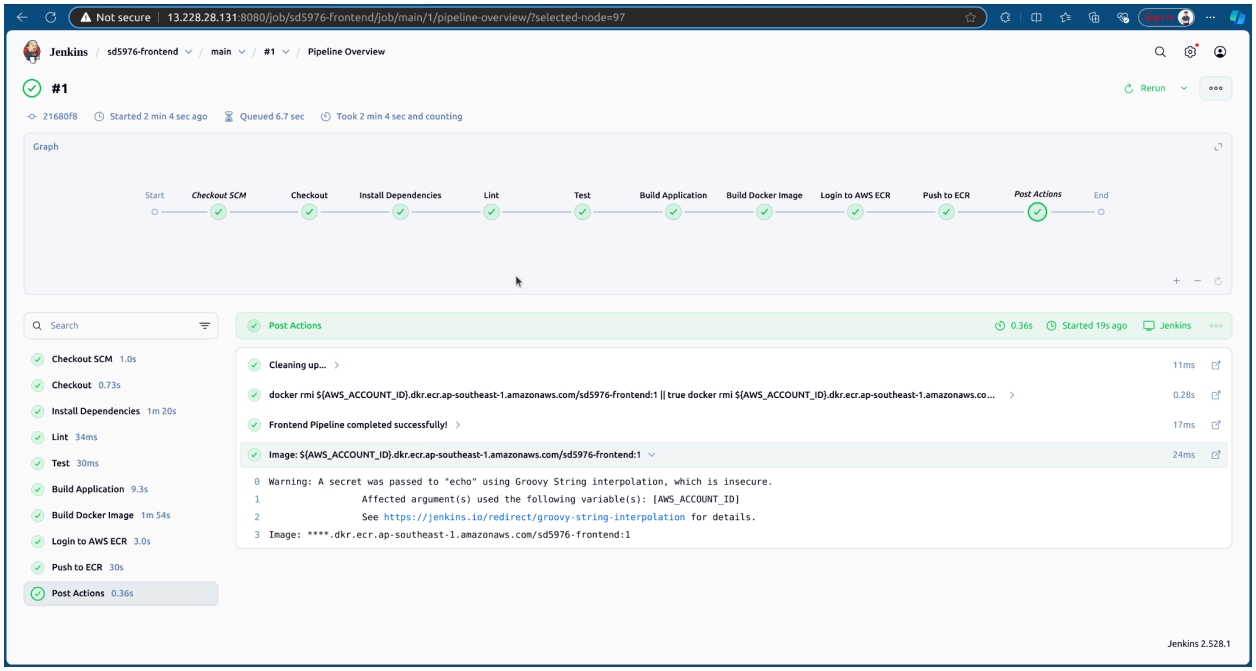


Configure Jenkins job to point to the *Frontend* branch and Jenkinsfile of *Frontend*.

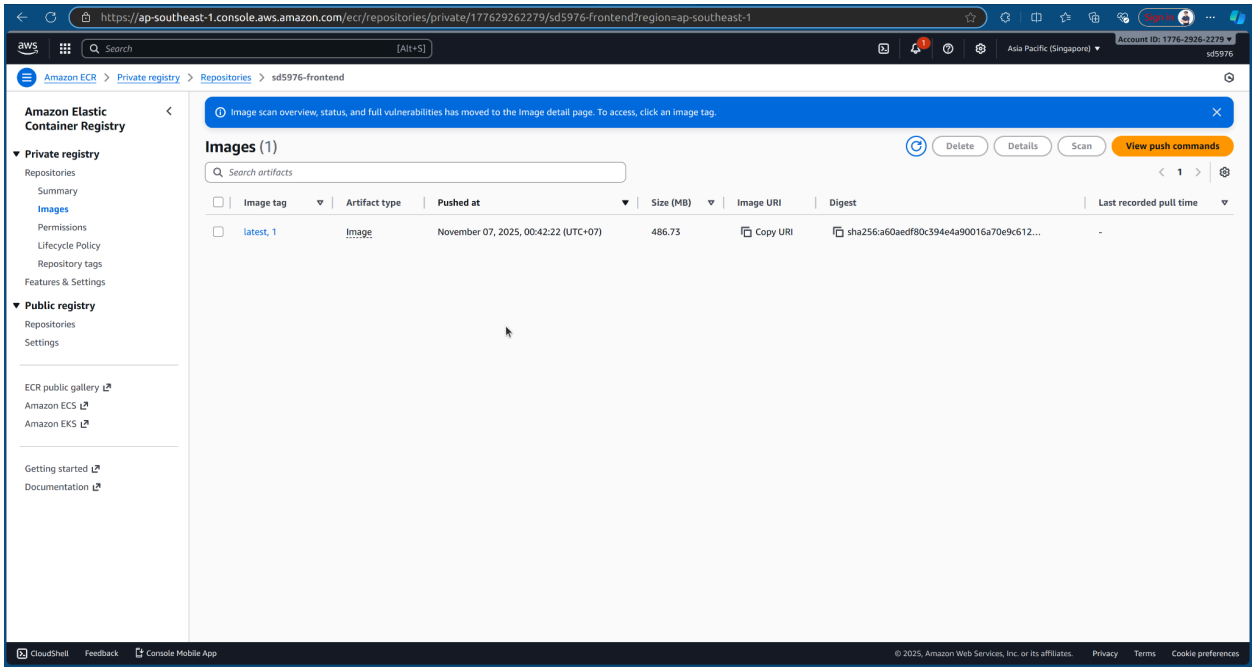




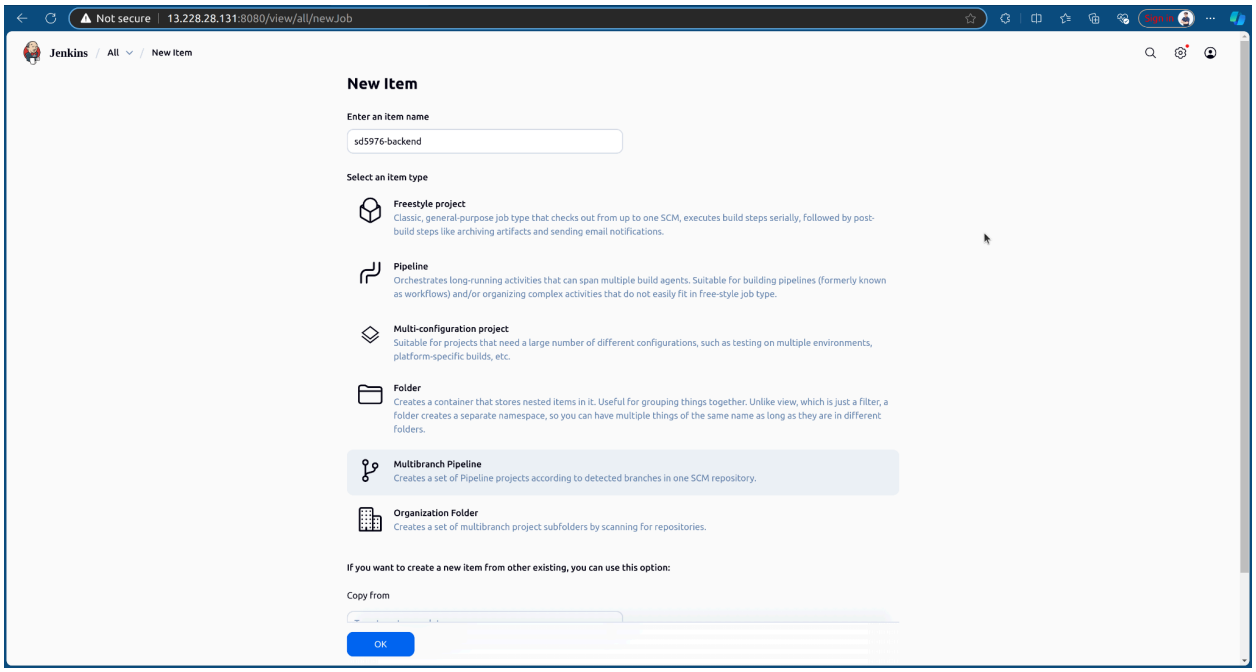
After creating, the job will run Jenkins based on the Jenkinsfile and source.



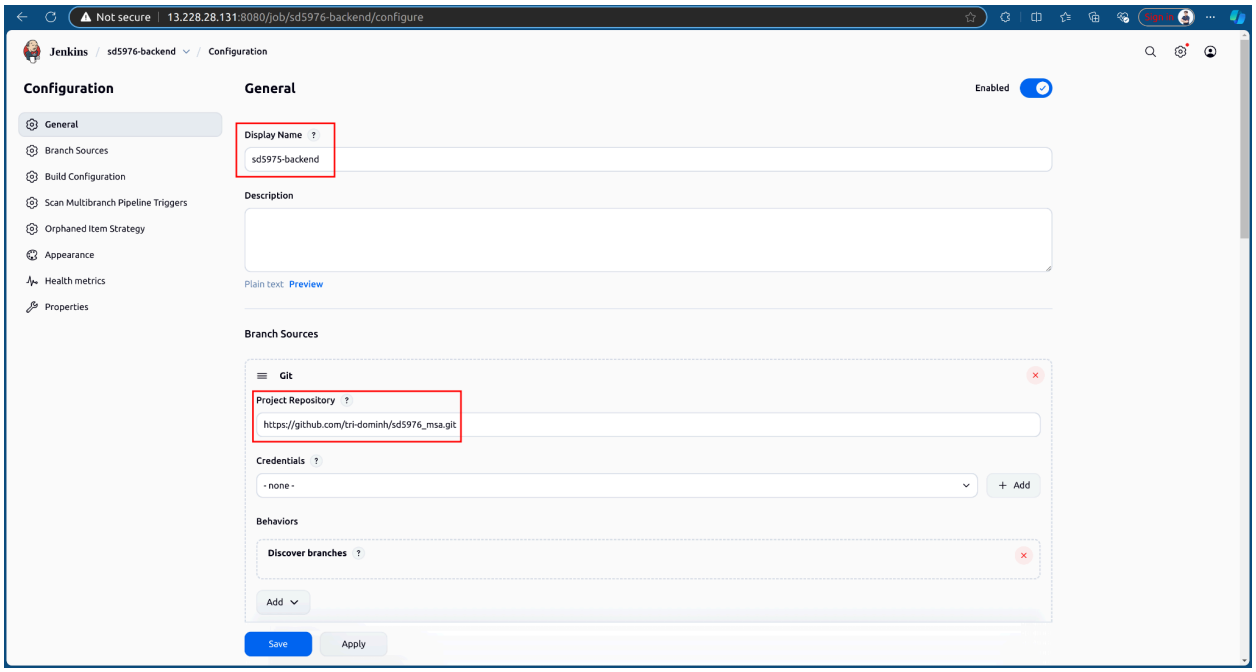
CI will run and push the *Frontend* image to ECR.

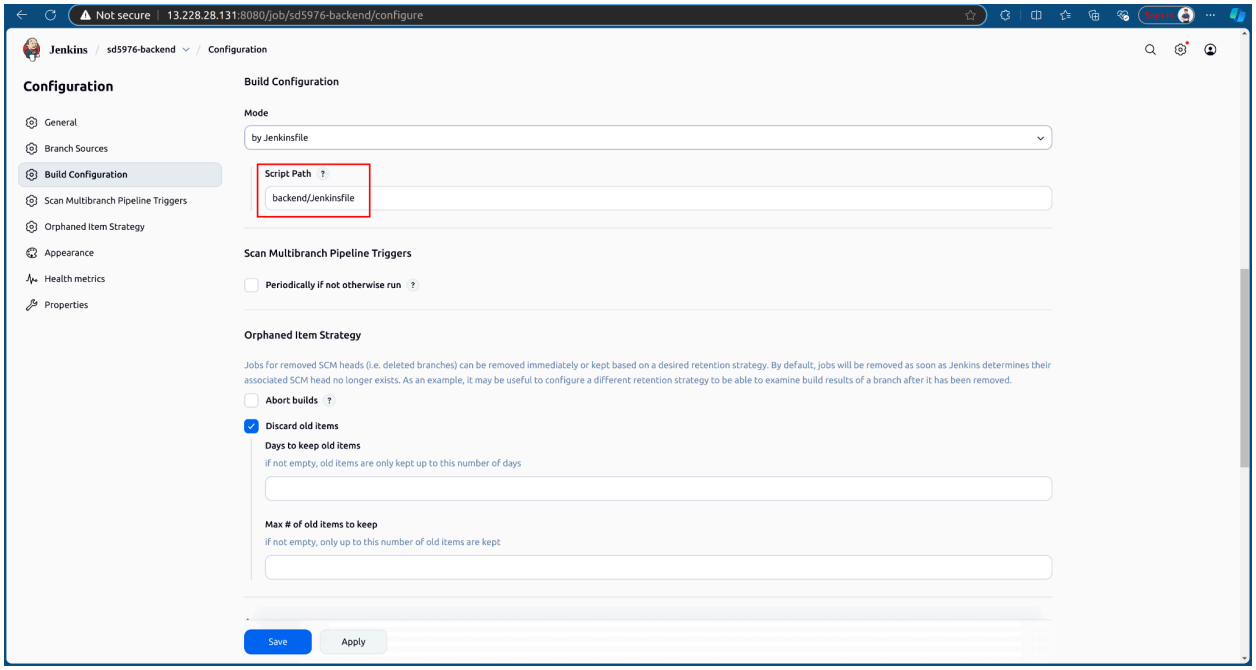


Next, we will create a similar job for the *Backend*, click **New item** → Enter name and select **Multibranch pipeline** job.

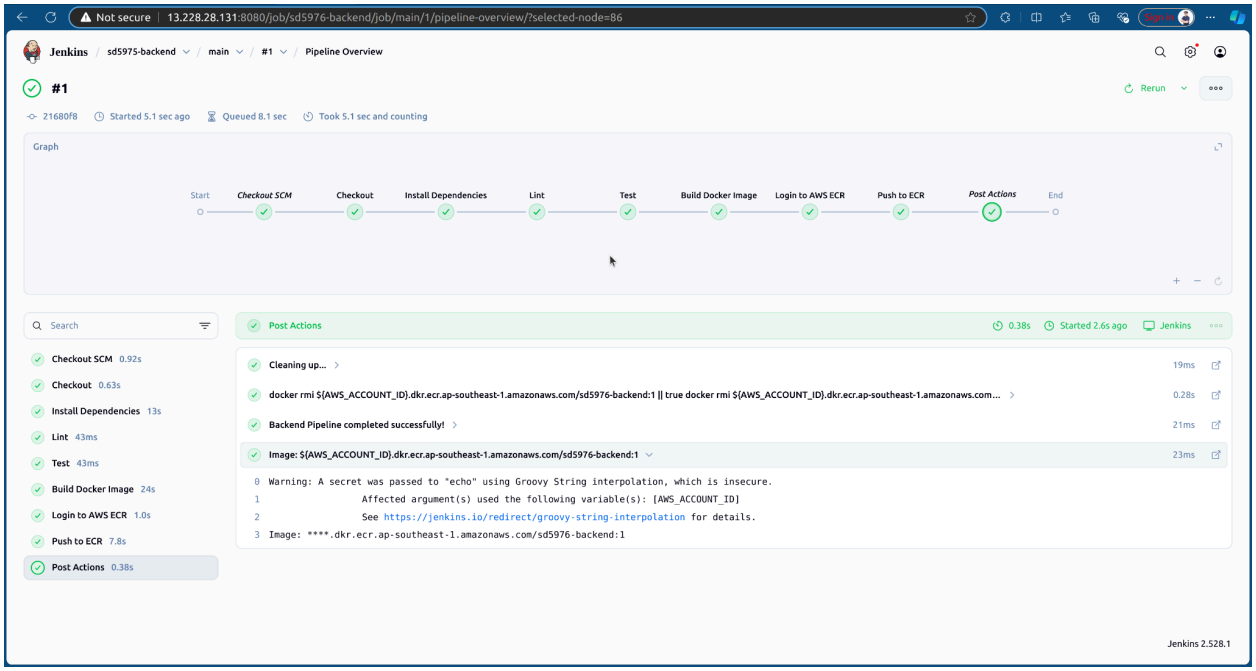


Configure Jenkins job to point to *Backend* branch and Jenkinsfile of *Backend*.

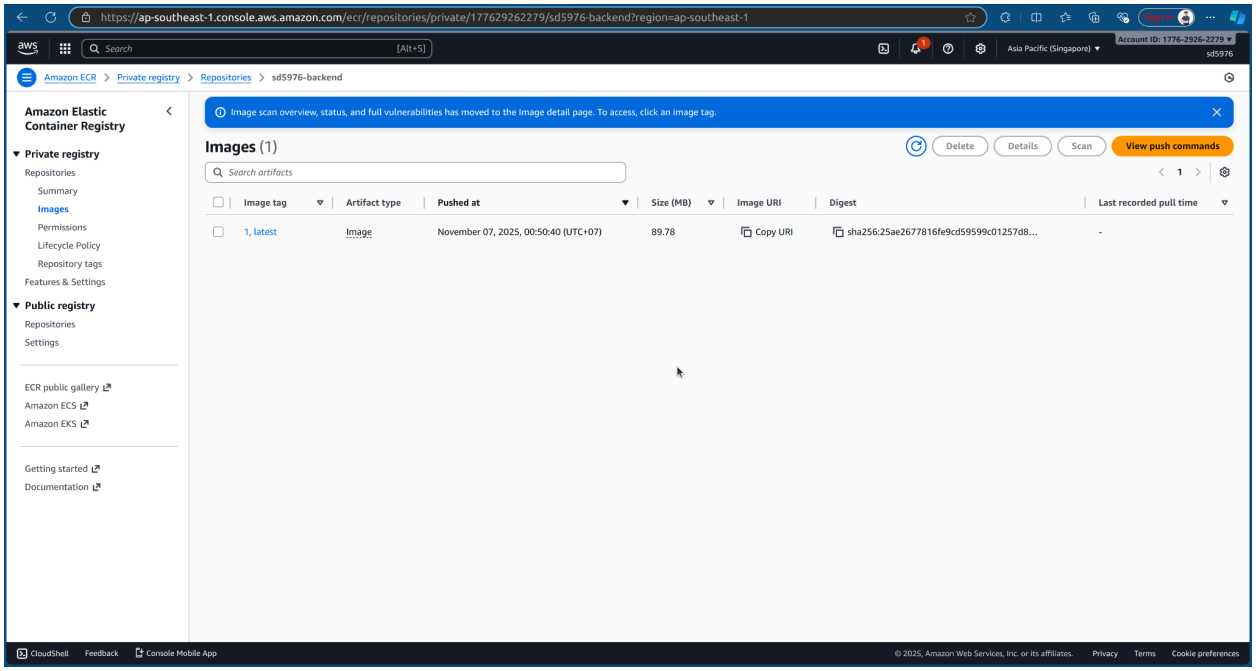




After creating, the job will run Jenkins based on the Jenkinsfile and source.



CI will run and push the *Backend* image to ECR.



V. Deploy with ArgoCD

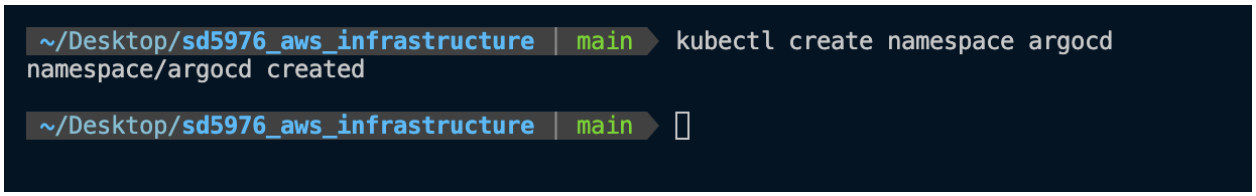
Refer to source code:

https://github.com/tri-dominh/sd5976_devops_cicd_deployment/tree/main/argocd-deployment

Before starting, you need to install the ArgoCD CLI on your local machine.

Connect to your EKS cluster and create a namespace for ArgoCD:

kubectl create namespace argocd



Install ArgoCD to the **argocd** namespace in EKS:

kubectl apply -n argocd -f

https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

```
~/De/sd5976_aws_infrastructure | main  kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
customresourcedefinition.apiextensions.k8s.io/applications.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/appprojects.argoproj.io created
serviceaccount/argocd-application-controller created
serviceaccount/argocd-applicationset-controller created
serviceaccount/argocd-dex-server created
serviceaccount/argocd-notifications-controller created
serviceaccount/argocd-redis created
serviceaccount/argocd-repo-server created
serviceaccount/argocd-server created
role.rbac.authorization.k8s.io/argocd-application-controller created
role.rbac.authorization.k8s.io/argocd-applicationset-controller created
role.rbac.authorization.k8s.io/argocd-dex-server created
role.rbac.authorization.k8s.io/argocd-notifications-controller created
role.rbac.authorization.k8s.io/argocd-redis created
role.rbac.authorization.k8s.io/argocd-server created
clusterrole.rbac.authorization.k8s.io/argocd-application-controller created
clusterrole.rbac.authorization.k8s.io/argocd-applicationset-controller created
clusterrole.rbac.authorization.k8s.io/argocd-server created
rolebinding.rbac.authorization.k8s.io/argocd-application-controller created
rolebinding.rbac.authorization.k8s.io/argocd-applicationset-controller created
rolebinding.rbac.authorization.k8s.io/argocd-dex-server created
rolebinding.rbac.authorization.k8s.io/argocd-notifications-controller created
rolebinding.rbac.authorization.k8s.io/argocd-redis created
rolebinding.rbac.authorization.k8s.io/argocd-server created
clusterrolebinding.rbac.authorization.k8s.io/argocd-application-controller created
clusterrolebinding.rbac.authorization.k8s.io/argocd-applicationset-controller created
clusterrolebinding.rbac.authorization.k8s.io/argocd-server created
configmap/argocd-cm created
configmap/argocd-cmd-params-cm created
configmap/argocd-gpg-keys-cm created
configmap/argocd-notifications-cm created
configmap/argocd-rbac-cm created
configmap/argocd-ssh-known-hosts-cm created
configmap/argocd-tls-certs-cm created
secret/argocd-notifications-secret created
secret/argocd-secret created
service/argocd-applicationset-controller created
service/argocd-dex-server created
service/argocd-metrics created
service/argocd-notifications-controller-metrics created
service/argocd-redis created
service/argocd-repo-server created
service/argocd-server created
service/argocd-server-metrics created
deployment.apps/argocd-applicationset-controller created
deployment.apps/argocd-dex-server created
deployment.apps/argocd-notifications-controller created
deployment.apps/argocd-redis created
deployment.apps/argocd-repo-server created
deployment.apps/argocd-server created
statefulset.apps/argocd-application-controller created
networkpolicy.networking.k8s.io/argocd-application-controller-network-policy created
```

Wait for a few minutes while Kubernetes creates the pods, then check their status:

kubectl get pods -n argocd

```
~/De/sd5976_aws_infrastructure | main  kubectl get pods -n argocd
NAME                                                READY   STATUS    RESTARTS   AGE
argocd-application-controller-0                     1/1     Running   0          50s
argocd-applicationset-controller-5c8d57cb-twmxx     1/1     Running   0          51s
argocd-dex-server-77978c8bbc-8r8n7                  1/1     Running   0          51s
argocd-notifications-controller-59db9f4c79-ddj84    1/1     Running   0          51s
argocd-redis-c9bdd6f47-rdzbn                        1/1     Running   0          51s
argocd-repo-server-6c499f5448-8g5l8                 1/1     Running   0          51s
argocd-server-58b9cbff8d-tq29t                      1/1     Running   0          51s
```

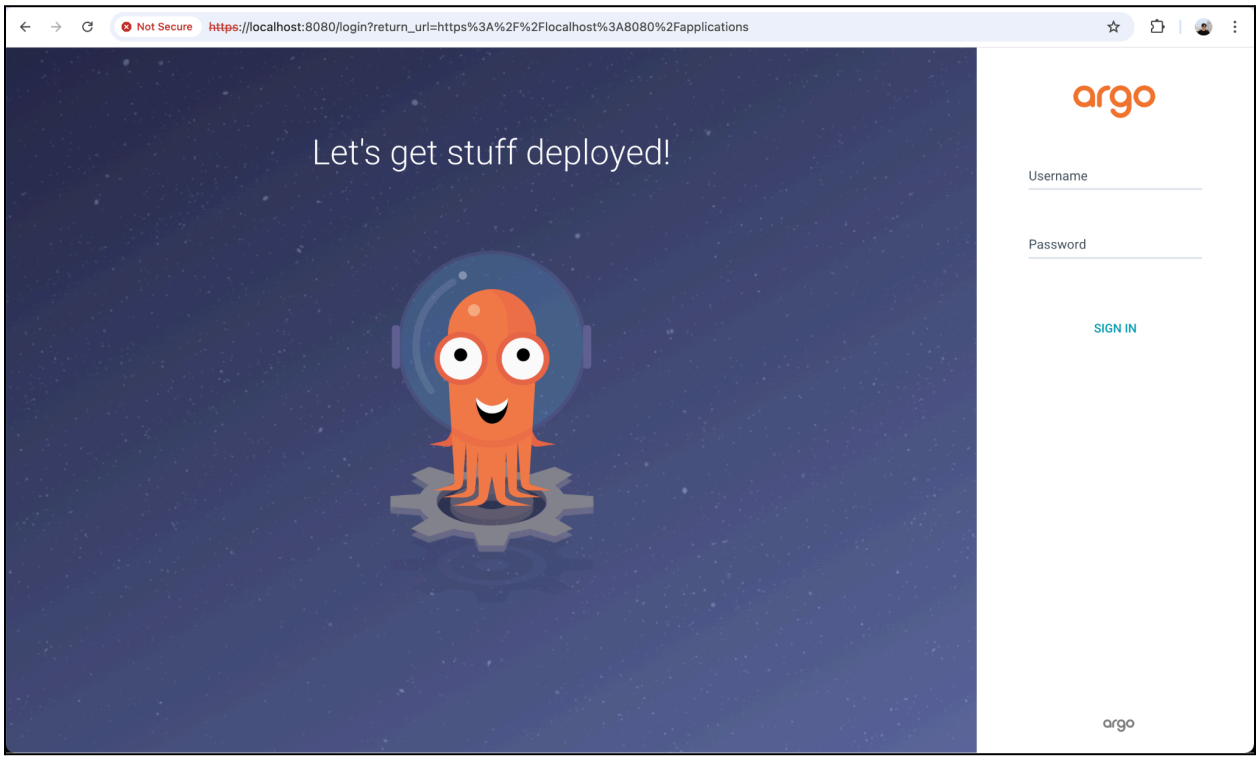
We need to get the initial password and decode it from the base64-encoded secret:

kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 --decode; echo

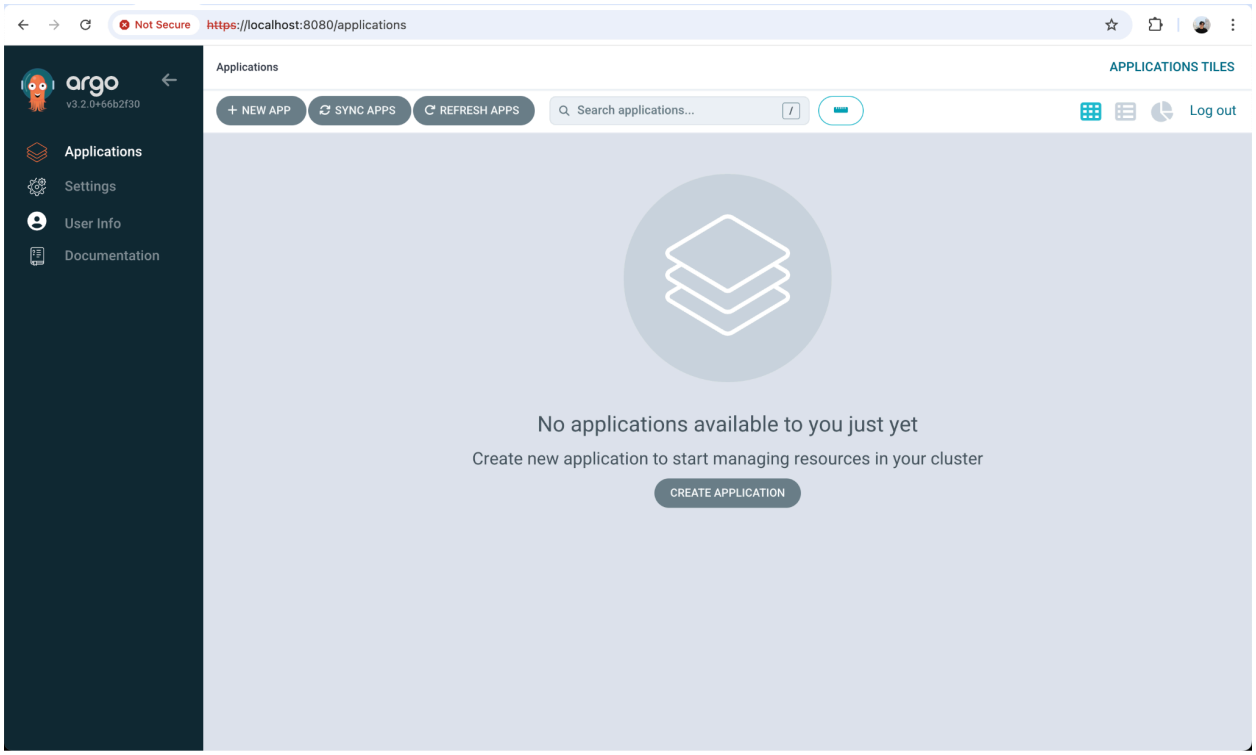
Forward the ArgoCD service port from EKS to your local machine:

kubectl port-forward svc/argocd-server -n argocd 8080:443

Then open your browser and go to https://localhost:8080



Login to ArgoCD with default username **admin** and the decode password.



Once you can access the ArgoCD web interface, log in through the CLI:

argocd login localhost:8080 --insecure

```
~/Desktop/sd5976_aws_infrastructure | main  argocd login localhost:8080 --insecure
Username: admin
Password:
'admin:login' logged in successfully
Context 'localhost:8080' updated

~/Desktop/sd5976_aws_infrastructure | main
```

Go to **sd5976_devops_cicd_deployment/argocd-deployment**, run ecr-secret script to create ECR in EKS. This help ArgoCD can pull images from ECR inside EKS.

```
~/De/sd5976_devops_cicd_deployment/argocd-deployment | main  bash ecr-secret.sh -l app.kubernetes.io/name=argocd-image-updater -f
Creating ECR pull secret...
namespace/sd5976-msa configured
secret/ecr-secret created
ECR secret 'ecr-secret' created in namespace 'sd5976-msa'

Note: ECR tokens expire after 12 hours.
For production, set up a CronJob to refresh this secret.
```

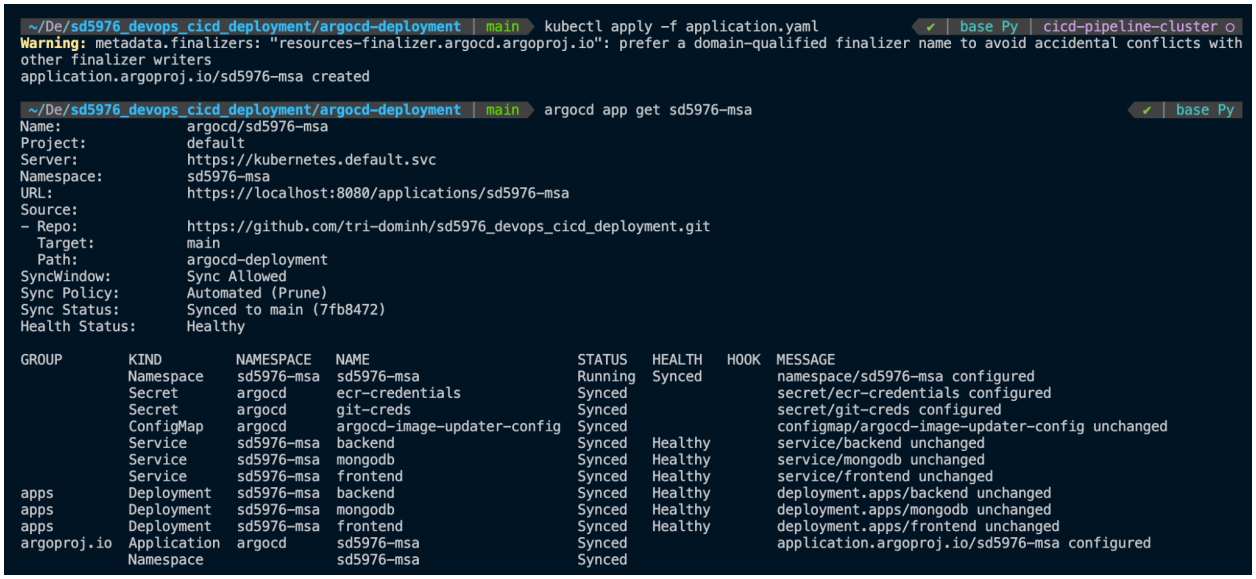
Connect your deployment repository to ArgoCD so it can monitor and deploy automatically:

argocd repo add https://github.com/tri-dominh/sd5976_devops_cicd_deployment --username <your_username> --password <your_password>

Use **application.yaml** file to create **sd5976_msa** app in **argocd**:

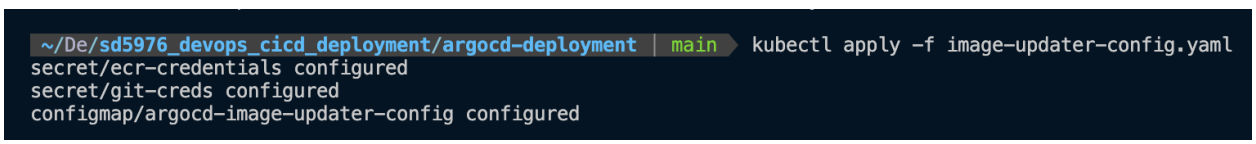
kubectl apply -f application.yaml

argocd app get sd5976-msa



Apply the **image-updater-config.yaml** file so that ArgoCD can automatically update images from ECR:

kubectl apply -f image-updater-config.yaml



Once the deployment starts, you can monitor the pods running in the **sd5976-msa** namespace:

kubectl get pods -n sd5976-msa -w

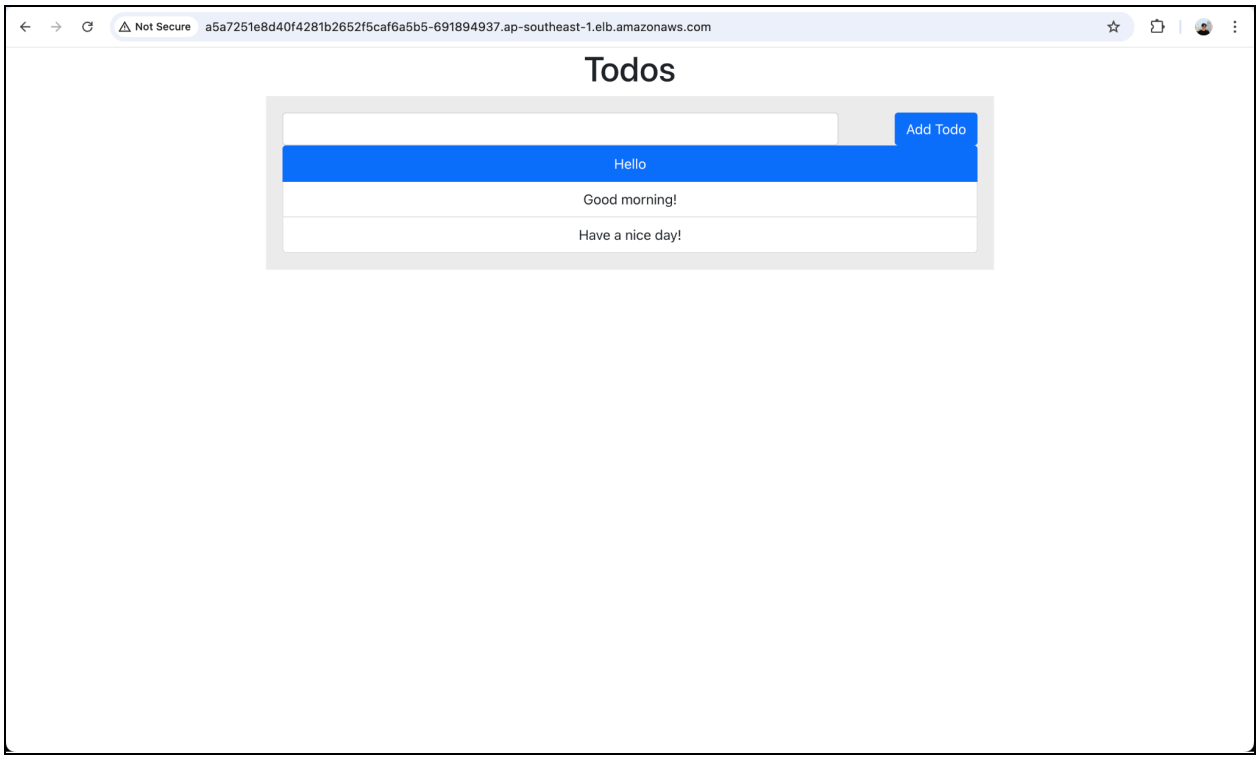
```
~/De/sd5976_devops_cicd_deployment/argocd-deployment | main !1 kubectl get pods -n sd5976-msa -w
NAME                        READY   STATUS    RESTARTS   AGE
backend-8684865f6f-gsswz    1/1     Running   0          21m
backend-8684865f6f-hrm92    1/1     Running   0          21m
frontend-5687bcc76d-b7qvf   1/1     Running   0          21m
frontend-5687bcc76d-c465t   1/1     Running   0          21m
mongodb-78547887ff-t68k6    1/1     Running   0          21m
```

After all pods are running, get the external URL of your frontend service:

kubectl get svc frontend -n sd5976-msa

```
~/Desktop/sd5976_devops_cicd_deployment | main !1 kubectl get svc frontend -n sd5976-msa
NAME       TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)        AGE
frontend   LoadBalancer   172.20.97.235   a5a7251e8d40f4281b2652f5caf6a5b5-691894937.ap-southeast-1.elb.amazonaws.com   80:32120/TCP   22m

~/Desktop/sd5976_devops_cicd_deployment | main !1
```

The frontend application is now up and running.


Not Secure
a5a7251e8d40f4281b2652f5caf6a5b5-691894937.ap-southeast-1.elb.amazonaws.com
Todos
Add Todo
Hello
Good morning!
Have a nice day!