

Practical DevOps – CI/CD Pipeline and deploying applications on AWS EKS

1. Table of Contents

<i>I. Introduction</i>	1
<i>II. Prerequisites</i>	2
<i>III. Git Repositories</i>	2
1. Infrastructure as code	2
2. Micro-services application	2
<i>IV. Provisioning Infrastructure by Terraform</i>	2
1. AWS Configure.....	2
2. Provisioning.....	2
<i>V. Jenkins</i>	4
1. Setup	4
2. Install Plugins Pipeline: AWS Steps.....	6
3. Setup AWS Credential Global	6
4. Setup CI-CD.....	6
5. Build Image and Push to ECR.....	8
6. Config Webhook	11
<i>VI. ArgoCD</i>	12
1. Install.....	12
2. Port Forward and Access ArgoCD UI	13
3. Setup Application	13
<i>VII. Prometheus and Grafana</i>	15
1. Installation.....	15
2. Monitoring	16

I. Introduction

1. Setting up a CI/CD Pipeline and deploying applications on AWS EKS.
2. Reuse Jenkins CI and use GitOps for the CD pipeline.
3. Setting up Monitoring Prometheus and Grafana.

II. Prerequisites

- Have an AWS account.
- AWS CLI installed.
- Terraform installed.
- Kubectl installed.
- Helm installed.

III. Git Repositories

1. Infrastructure as code

[sd2793_aws_infrastructure](#)

2. Micro-services application

[sd2793_msa](#)

IV. Provisioning Infrastructure by Terraform

1. AWS Configure

```
+ sd2793_aws_infrastructure git:(main) ✘ aws configure
AWS Access Key ID [*****TGQL]: AKIAQ*****ETGQL
AWS Secret Access Key [*****xbNX]: lFGg2YYi*****QxbNX
Default region name [ap-southeast-1]: ap-southeast-1
Default output format [json]: json
```

2. Provisioning

1. Git clone from repository infrastructure as code

2. Go to folder sd2793_aws_infrastructure

2.1. Initialize terraform `terraform init`

```
+ sd2793_aws_infrastructure git:(main) ✘ terraform init
Initializing the backend...
Initializing modules...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/external from the dependency lock file
- Using previously-installed hashicorp/aws v5.61.0
- Using previously-installed hashicorp/external v2.3.3

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

2.1. Provisioning infra terraform apply -auto-approve --var-file "terraform.tfvars"

```

-> sd2793.aws_infrastructure git:(main) ✘ terraform apply -auto-approve --var-file "terraform.tfvars"
module.eks_cluster_and_worker_nodes.data.external.max_pods_calculator["one"]: Reading...
module.eks_iam_role.data.aws_iam_policy_document.assume_role_policy: Reading...
module.jenkins_iam_role.data.aws_iam_policy_document.assume_role_policy: Reading...
module.eks_iam_role.data.aws_iam_policy_document.eks_policy: Reading...
module.eks_iam_role.data.aws_iam_policy_document.assume_workers: Reading...
module.eks_iam_role.data.aws_iam_policy_document.assume_role_policy: Read complete after 0s [id=3552664922]
module.eks_iam_role.data.aws_iam_policy_document.assume_workers: Read complete after 0s [id=2851119427]
module.jenkins_iam_role.data.aws_iam_policy_document.ecr_policy: Read complete after 0s [id=3460616156]
module.jenkins_iam_role.data.aws_iam_policy_document.ecr_policy: Reading...
module.jenkins_iam_role.data.aws_iam_policy_document.assume_role_policy: Read complete after 0s [id=2217611055]
module.jenkins_iam_role.data.aws_iam_policy_document.ecr_policy: Read complete after 0s [id=401889238]
module.eks_cluster_and_worker_nodes.data.external.max_pods_calculator["one"]: Read complete after 0s [id=-]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# module.compute.aws_eip.jenkins_eip will be created
+ resource "aws_eip" "jenkins_eip" {
    + allocation_id      = (Known after apply)
    + arn                = (Known after apply)
    + association_id    = (Known after apply)
    + carrier_ip         = (Known after apply)
    + customer_owned_ip = (Known after apply)
    + domain             = (Known after apply)
    + id                 = (Known after apply)
    + instance            = (Known after apply)
    + network_border_group = (Known after apply)
    + network_interface   = (Known after apply)
    + private_dns        = (Known after apply)
    + private_ip          = (Known after apply)
    + ptr_record          = (Known after apply)
    + public_dns          = (Known after apply)
    + public_ip           = (Known after apply)
    + public_ipv4_pool    = (Known after apply)
    + tags_all            = (Known after apply)
    + vpc                = (Known after apply)
}

# module.compute.aws_instance.jenkins_docker_server will be created
+ resource "aws_instance" "jenkins_docker_server" {
    + ami                  = "ami-060e277c0d4cce53"
    + arn                = (Known after apply)
    + associate_public_ip_address = (Known after apply)
    + availability_zone   = (Known after apply)
    + cpu_core_count      = (Known after apply)
    + cpu_threads_per_core = (Known after apply)
    + disable_api_stop    = (Known after apply)
    + disable_api_termination = (Known after apply)
    + ebs_optimized       = (Known after apply)
    + get_password_data   = false
    + host_id             = (Known after apply)
    + host_resource_group_arn = (Known after apply)
    + iam_instance_profile = "jenkins"
    + id                 = (Known after apply)
    + instance_initiated_shutdown_behavior = (Known after apply)
    + instance_lifecycle   = (Known after apply)
    + instance_state       = "t3.small"
    + instance_type         = "t3.small"
    + ipv6_address_count   = (Known after apply)
    + ipv6_addresses        = (Known after apply)
}

```



```

module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [3m40s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [3m51s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [4m11s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [4m22s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [4m33s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [4m44s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [4m55s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [5m1s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [5m11s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [5m21s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [5m31s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [5m41s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [5m51s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [6m1s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [6m11s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [6m21s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [6m31s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [6m41s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [6m51s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [7m1s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Still creating... [7m21s elapsed]
module.eks_cluster_and_worker_nodes.aws_eks_cluster.eks_cluster: Creation complete after 7m25s [id=sd2793-devops-eks-cluster]
module.eks_cluster_and_worker_nodes.aws_launch_template.eks_cluster: Creation complete after 7m25s [id=sd2793-devops-eks-cluster]
module.eks_cluster_and_worker_nodes.aws_launch_template["one"]: Creating...
module.eks_cluster_and_worker_nodes.aws_launch_template["one"]: Creation complete after 0s [id=1-015aeac3186569f3e]
module.eks_cluster_and_worker_nodes.aws_node_group.node_groups["one"]: Creating...
module.eks_cluster_and_worker_nodes.aws_node_group.node_groups["one"]: Still creating... [10s elapsed]
module.eks_cluster_and_worker_nodes.aws_node_group.node_groups["one"]: Still creating... [20s elapsed]
module.eks_cluster_and_worker_nodes.aws_node_group.node_groups["one"]: Still creating... [30s elapsed]
module.eks_cluster_and_worker_nodes.aws_node_group.node_groups["one"]: Still creating... [40s elapsed]
module.eks_cluster_and_worker_nodes.aws_node_group.node_groups["one"]: Still creating... [50s elapsed]
module.eks_cluster_and_worker_nodes.aws_node_group.node_groups["one"]: Still creating... [1m0s elapsed]
module.eks_cluster_and_worker_nodes.aws_node_group.node_groups["one"]: Still creating... [1m10s elapsed]
module.eks_cluster_and_worker_nodes.aws_node_group.node_groups["one"]: Still creating... [1m20s elapsed]
module.eks_cluster_and_worker_nodes.aws_node_group.node_groups["one"]: Still creating... [1m30s elapsed]
module.eks_cluster_and_worker_nodes.aws_node_group.node_groups["one"]: Still creating... [1m40s elapsed]
module.eks_cluster_and_worker_nodes.aws_node_group.node_groups["one"]: Still creating... [1m50s elapsed]
module.eks_cluster_and_worker_nodes.aws_node_group.node_groups["one"]: Creation complete after 2m0s [id=sd2793-devops-eks-cluster:node-app]

Apply complete! Resources: 44 added, 0 changed, 0 destroyed.

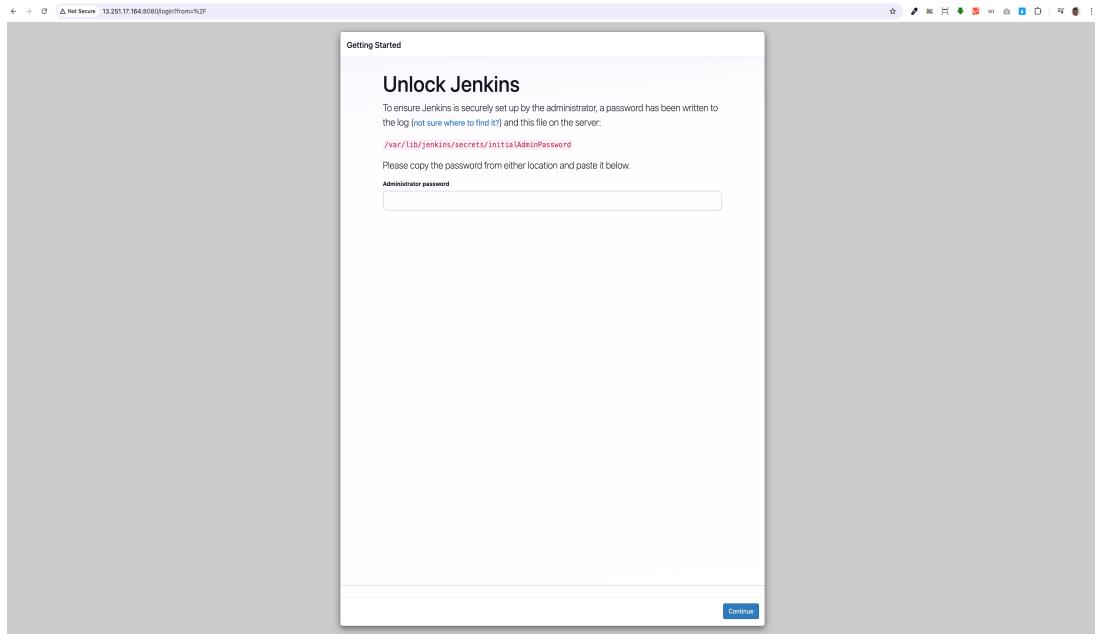
Outputs:

ecr_repository_urls = [
    "010438499500.dkr.ecr.ap-southeast-1.amazonaws.com/backend",
    "010438499500.dkr.ecr.ap-southeast-1.amazonaws.com/frontend",
]
] eks = {
    "cluster_endpoint" = "sd2793-devops-eks-cluster"
    "eks_cluster_sg_id" = "sg-04fe4565ae2466360"
    "max_pods_for_node_groups" = {
        "one" = tomap(
            "max_pods" = "11"
        )
    }
}
jenkins_instance_details = {
    "iam_profile" = "jenkins"
    "instance_id" = "i-042e2923a70652135"
    "jenkins_sg_id" = "sg-08ac63cabf1f60289"
    "public_ip" = "13.251.17.164"
}
```

V. Jenkins

1. Setup

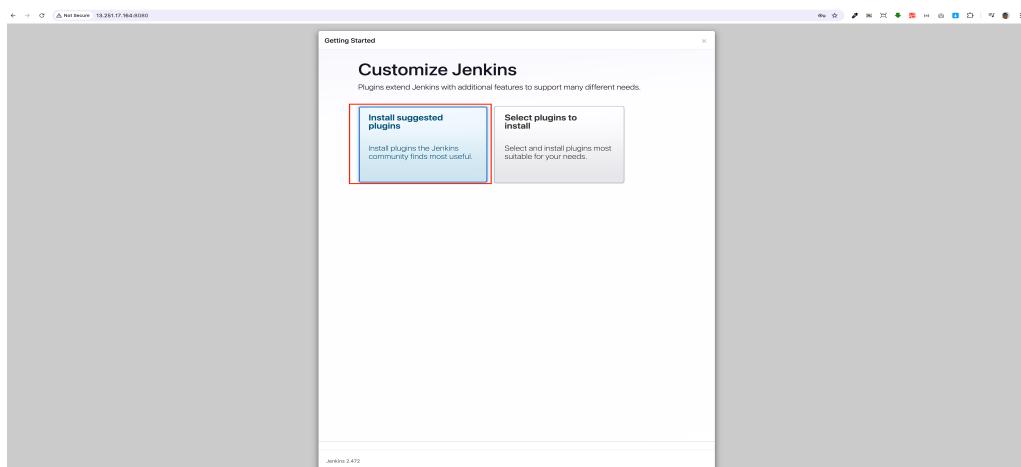
1. Get the **public_ip** in `jenkins_instance_details.public_ip` output from terraform
2. Open the browser http://jenkins_instance_details.public_ip:8080



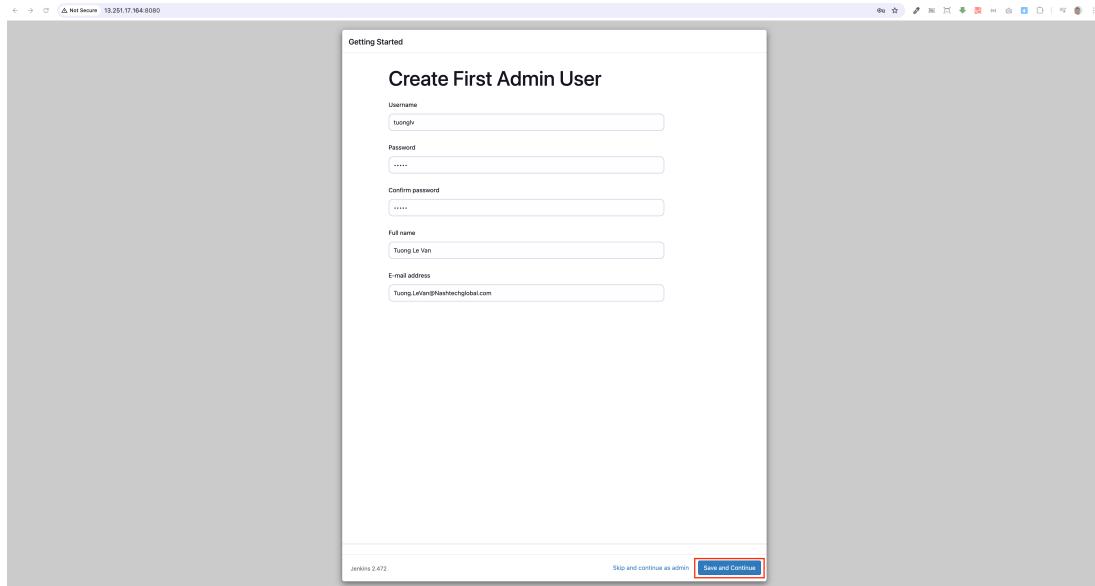
3. Access SSH to server and get the Admin password
`sudo cat /var/lib/jenkins/secrets/initialAdminPassword`

```
ubuntu@ip-10-0-4-146:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
666a07f314c344f59e174d954d7a21ca ←
```

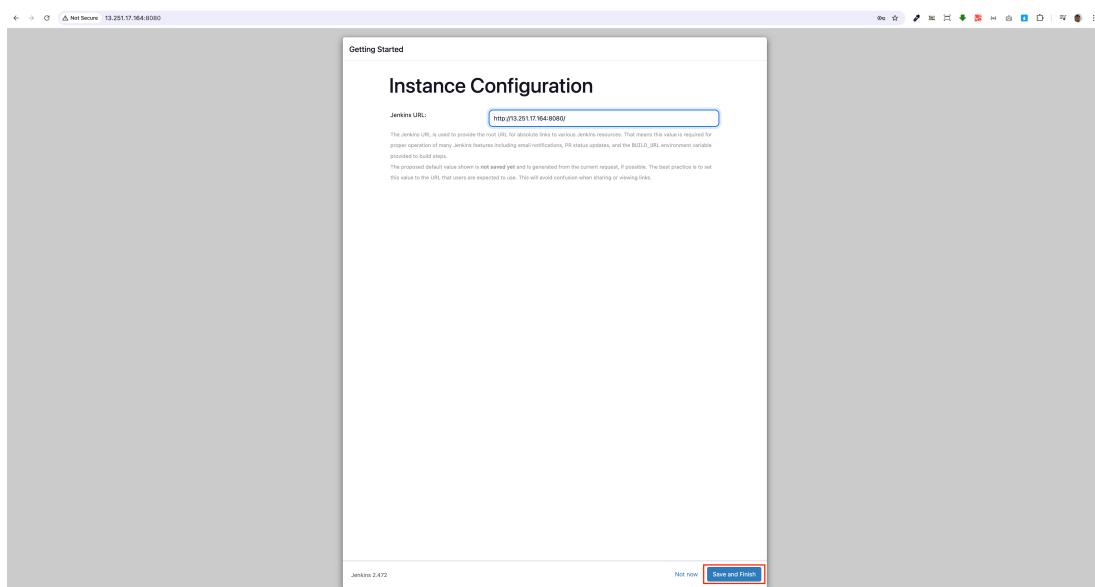
4. Copy the Admin Password output and enter to Administrator Password
> Click **Continue** button
5. Choose **Install suggested plugins**



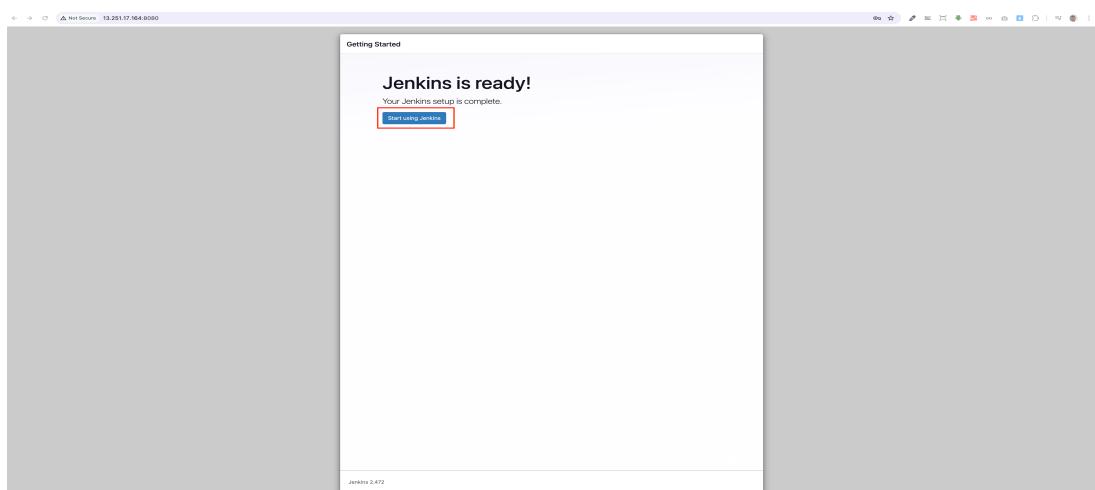
6. Create a new admin user and click **Save and Continue** button



7. Click **Save and Finish** button

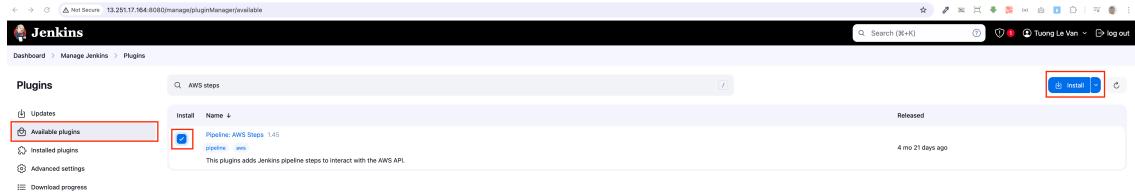


8. Click **Start using Jenkins** button



2. Install Plugins Pipeline: AWS Steps

1. Go to Dashboard > Manager Jenkins > Plugins
2. Available Plugins > Enter **AWS Steps** > Checked Install and click **Install** button

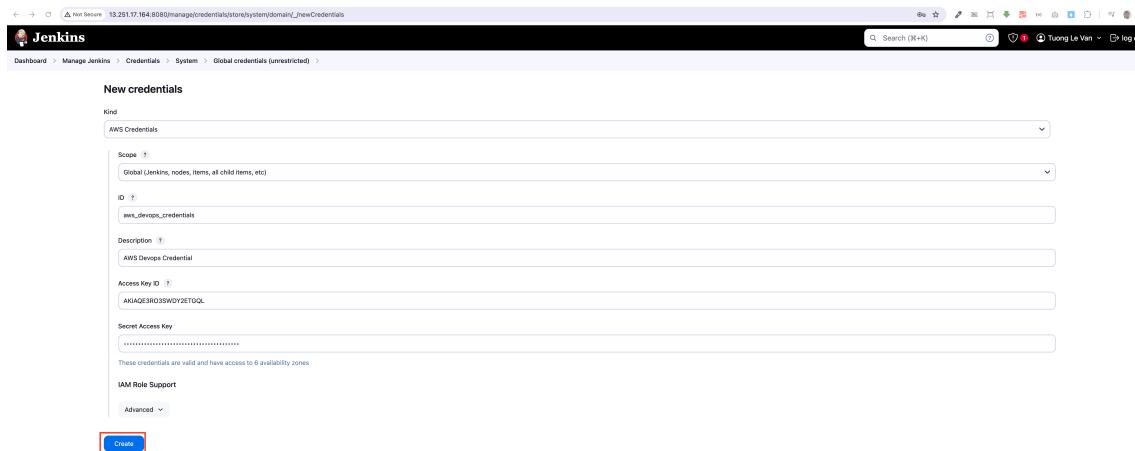


3. Setup AWS Credential Global

1. Go to Dashboard > Manager Jenkins > Credentials > System > Global Credentials > Click **Add credentials** button

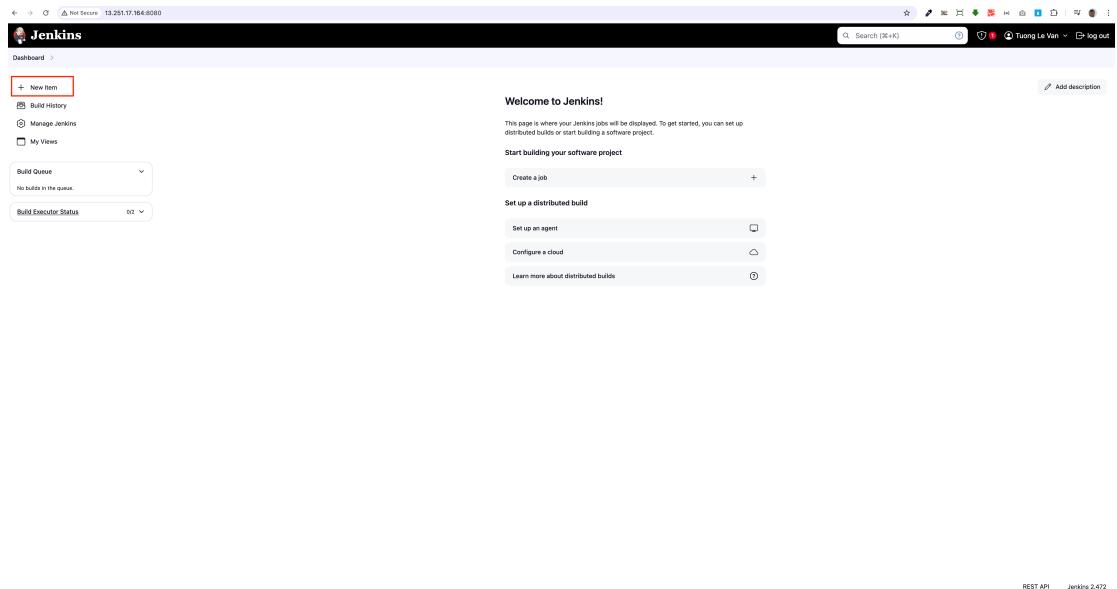


2. Enter the credentials and click **Create** button

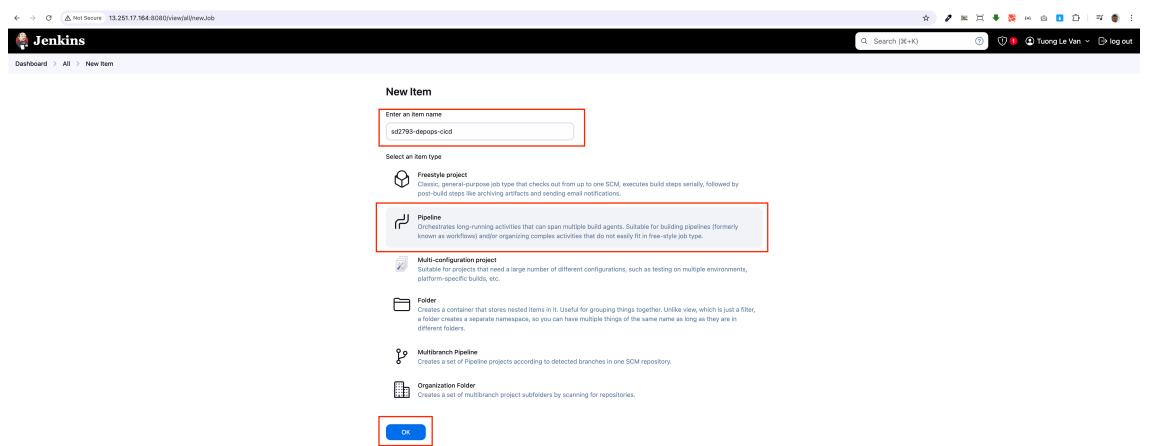


4. Setup CI-CD

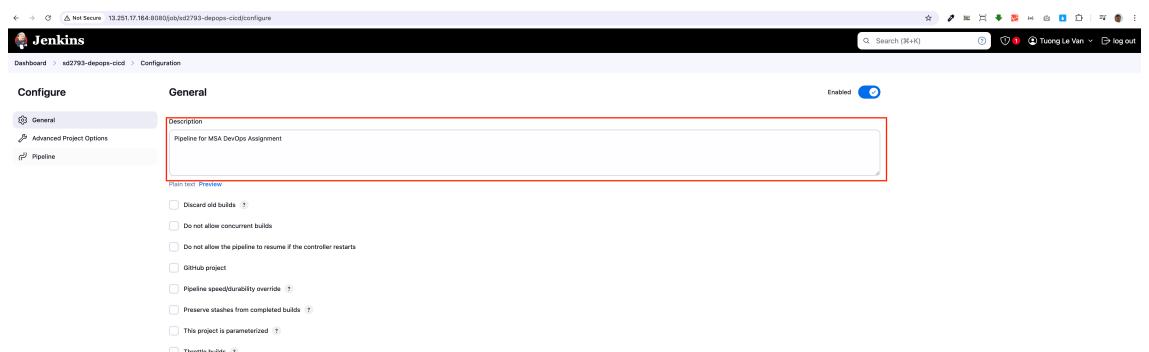
1. Go to Dashboard navigation and click **New Item** button



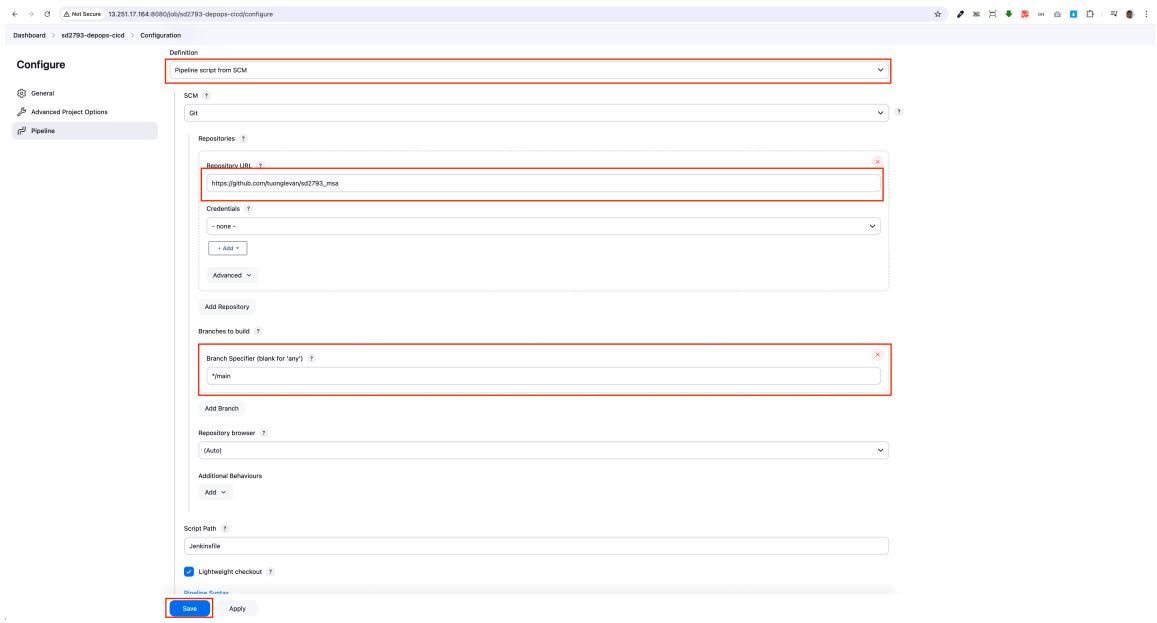
2. Enter the name of pipeline > Choose Pipeline and Click Ok button



3. Configure > General tab > Generate > Enter the Description name



4. Configure > Pipeline > Setup Pipeline SCM > Click **Save** button



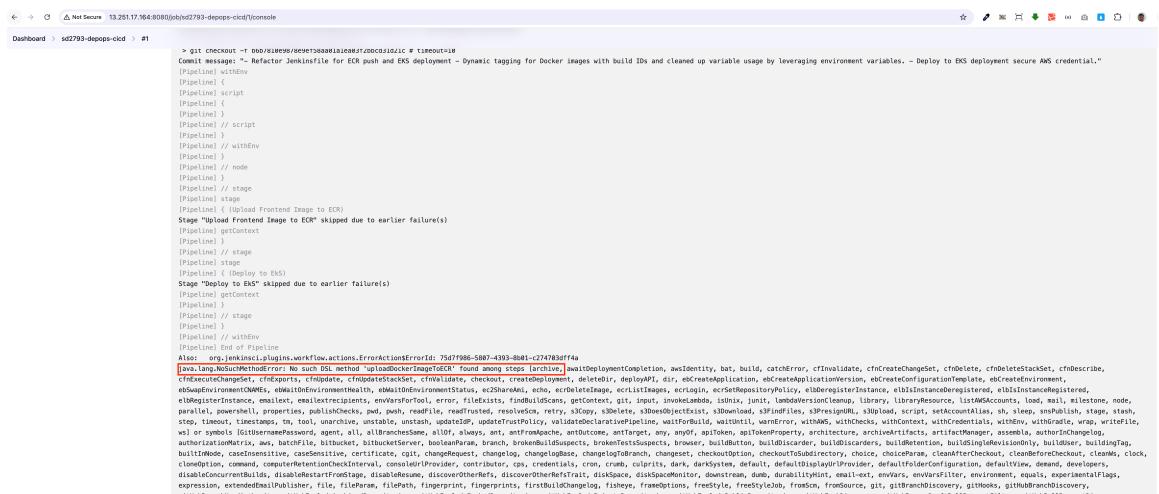
5. Build Image and Push to ECR

1. Go to Dashboard > Choose project name > Click button **Build Now**



2. Check Build > Read the **Console Output** of the build number

2.1. If have some issues, need to fixed and make another build



3. Check ECR > Login AWS console > Amazon ECR > Private Registry > Repositories

3.1. backend

The screenshot shows the AWS ECR console with the URL ap-southeast-1.console.aws.amazon.com/ecr/repositories/private/010438499500/backend?region=ap-southeast-1. The repository name is "backend". The "Images" tab is selected, showing one image entry:

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
latest, ver-2	Image	August 17, 2024, 18:39:48 (UTC+07)	89.53	Copy URI	sha256:ff8079dfba3d62a91de63dc7331f1...

3.2. frontend

The screenshot shows the AWS ECR console with the URL ap-southeast-1.console.aws.amazon.com/ecr/repositories/private/010438499500/frontend?region=ap-southeast-1. The repository name is "frontend". The "Images" tab is selected, showing one image entry:

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
ver-2, latest	Image	August 17, 2024, 18:41:02 (UTC+07)	485.33	Copy URI	sha256:4a30595246c6cd9dd20da59fd6d...

4. Verify Pipeline

The screenshot shows the Jenkins pipeline overview for job #8. The pipeline consists of five stages: Start, Build Backend, Build Frontend, Upload Backend, Upload Frontend, Deploy to EKS, and End. All stages are marked as green, indicating successful completion. The pipeline took 18 seconds to run.

5. Verify EKS

```
+ sd2793_aws_infrastructure git:(main) ✘ aws eks update-kubeconfig --name sd2793-devops-eks-cluster
Updated context arn:aws:eks:ap-southeast-1:010438499500:cluster/sd2793-devops-eks-cluster in /Users/tuonglv/.kube/config

[→] sd2793_aws_infrastructure git:(main) ✘ kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
backend-984f8b86d-zxwft   1/1     Running   0          6m43s
frontend-646f49fb59-9sqmr  1/1     Running   0          6m42s
mongo-0                 0/1     Pending   0          6m44s
```

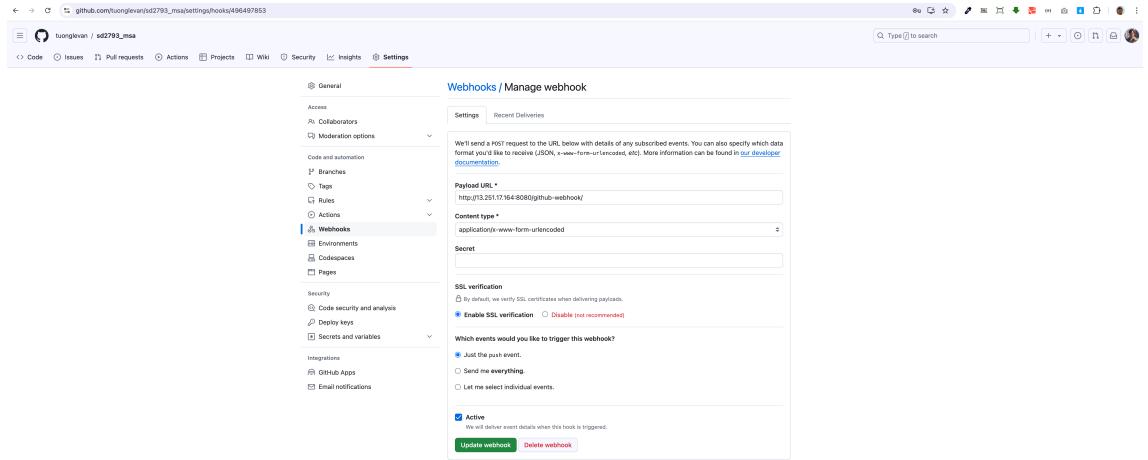
5.1. Verify Frontend App

```
→ sd2793_aws_infrastructure git:(main) ✘ kubectl get services
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
backend   ClusterIP 172.20.146.210 <none>        3000/TCP   12m
frontend  ClusterIP 172.20.195.210 <none>        3000/TCP   12m
kubernetes  ClusterIP 172.20.0.1   <none>        443/TCP    3h3m
mongo     ClusterIP 172.20.153.124 <none>        27017/TCP  12m
→ sd2793_aws_infrastructure git:(main) ✘ kubectl port-forward service/frontend 4000:3000
Forwarding from 127.0.0.1:4000 -> 3000
Forwarding from [::1]:4000 -> 3000
```

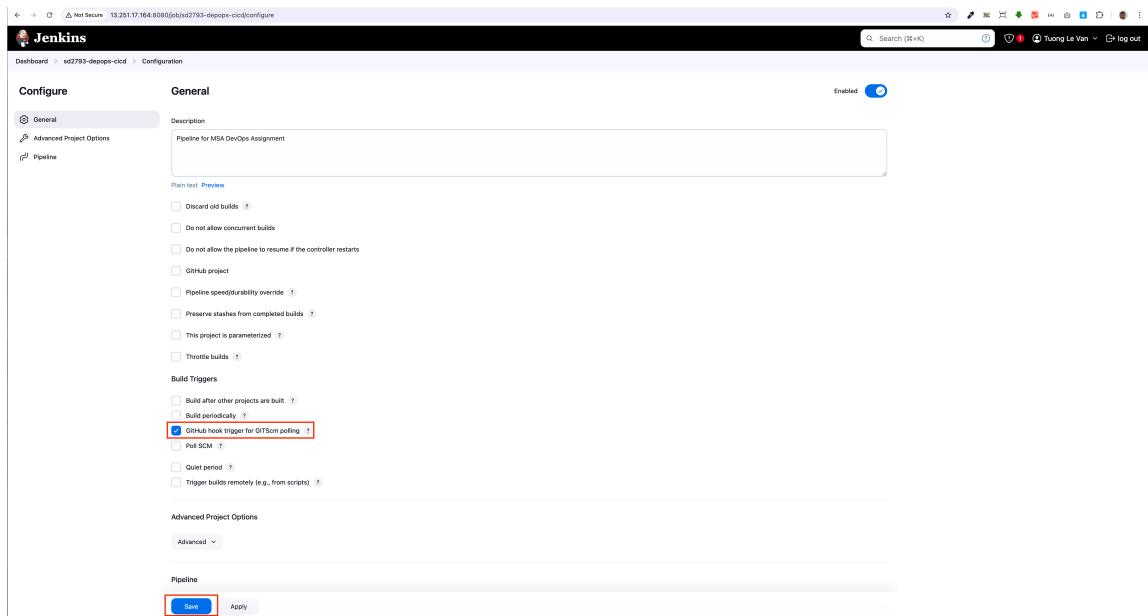


6. Config Webhook

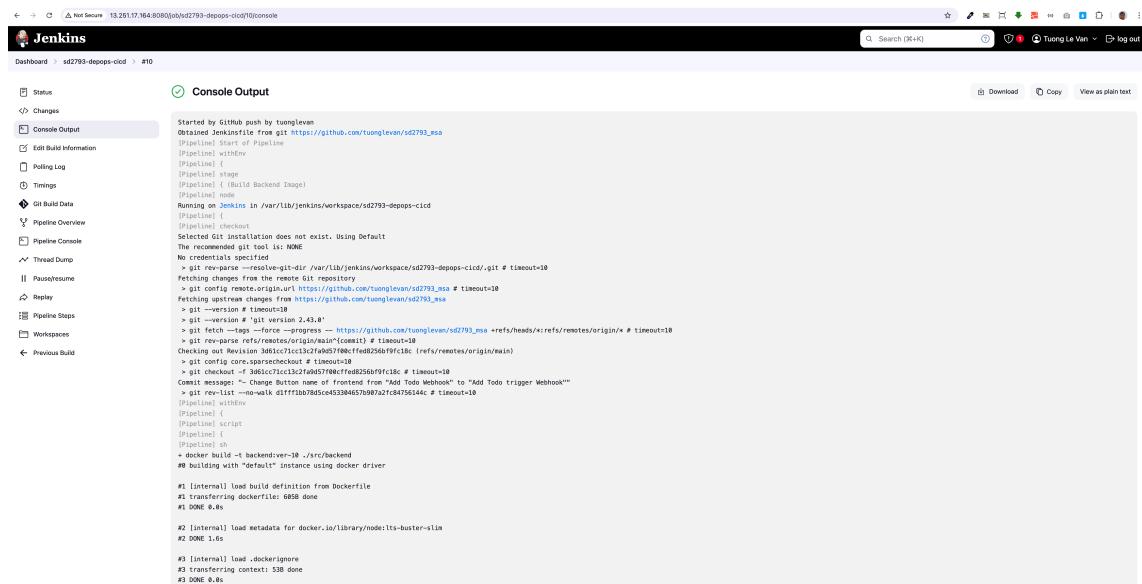
1. Go to repository [sd2793_msa](#) > Setting > Webhooks > click **Add webhook** button



2. Go to Dashboard in jenkins > Choose pipeline project > configure > general > checked **Github hook ...** in Build Triggers > click **Save** button



3. Changes code in MSA repo and checked Pipeline auto trigger



The screenshot shows the Jenkins Pipeline console output for a build named 'sd2793-deops-cld'. The output details the steps taken by the pipeline, including cloning the repository from GitHub, fetching upstream changes, and running a Docker build command. The build was triggered by a GitHub push from the user 'tuonglevan'.

```
Started by GitHub push by tuonglevan
  [Pipeline] $ git clone https://github.com/tuonglevan/sd2793_ms
  [Pipeline] > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/sd2793-deops-cld/.git # timeout=10
  [Pipeline] > git config remote.origin.url https://github.com/tuonglevan/sd2793_ms
  [Pipeline] > git fetch --tags --prune --progress -- https://github.com/tuonglevan/sd2793_ms
  [Pipeline] > git --version # timeout=10
  [Pipeline] > git --version # timeout=10
  [Pipeline] > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
  [Pipeline] > git rev-parse refs/heads/*{ref}^refs/remotes/origin/* # timeout=10
  Checking out Revision fedb256b9fc18c (refs/remotes/origin/main)
  > git config core.sparsecheckout true
  > git checkout -b 3d6cc7c1c32f6d579e0fcd856a9f918c # timeout=10
  Commit message: "Change button name of frontend From 'Add Todo Webhook' to 'Add Todo trigger Webhook'"
  > git rev-list --no-walk dff1bb78d5ce453394637b987a2fc84756144c # timeout=10
  [Pipeline] > git rev-parse --show-toplevel # timeout=10
  [Pipeline] > docker build -t backend:ver-18 ./src/backend
  # building with "default" instance using docker driver
  #1 [internal] load build definition from Dockerfile
  #1 transferring dockerfile: 6KB done
  #1 DONE 0.8s
  #2 [internal] load metadata for docker.io/library/node:8-buster-slim
  #2 DONE 1.6s
  #3 [internal] load dockerignore
  #3 transferring context: 53B done
  #3 DONE 0.4s
```

VI. ArgoCD

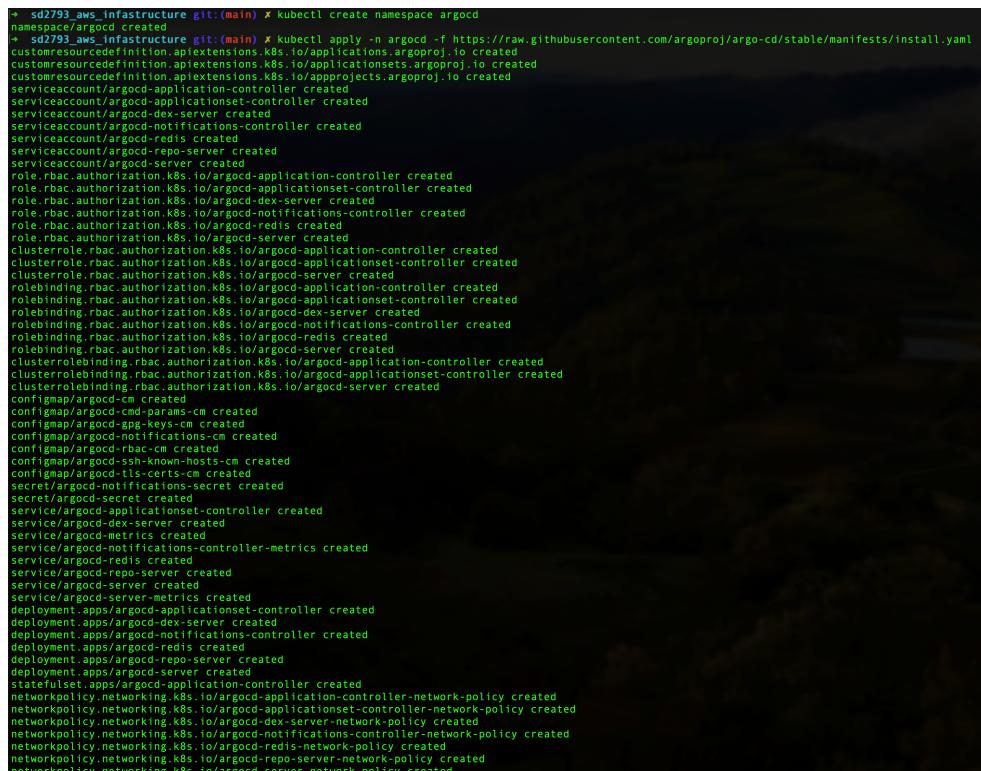
1. Install

1. Open terminal and run commands

1.1. `kubectl create namespace argo`

1.2. `kubectl apply -n argo -f`

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



The terminal window shows the execution of several kubectl commands to install ArgoCD. It includes creating a namespace, applying the ArgoCD manifest, and creating various roles, role bindings, and secrets required for the application's operation.

```
* sd2793_aws_infrastructure git:(main) ✘ kubectl create namespace argo
namespace/argo created
* sd2793_aws_infrastructure git:(main) ✘ kubectl apply -n argo -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
customresourcedefinition.apirextensions.k8s.io/applications.argoproj.io created
customresourcedefinition.apirextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apirextensions.k8s.io/approjects.argoproj.io created
serviceaccount/argocd-application-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-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-server created
configmap/argocd-cm created
configmap/argocd-cm-params-cm created
configmap/argocd-cm-pod-labels-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
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
networkpolicy.networking.k8s.io/argocd-applicationset-controller-network-policy created
networkpolicy.networking.k8s.io/argocd-dex-server-network-policy created
networkpolicy.networking.k8s.io/argocd-notifications-controller-network-policy created
networkpolicy.networking.k8s.io/argocd-redis-network-policy created
networkpolicy.networking.k8s.io/argocd-repo-server-network-policy created
networkpolicy.networking.k8s.io/argocd-server-network-policy created
```

2. Port Forward and Access ArgoCD UI

1. Port forward

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

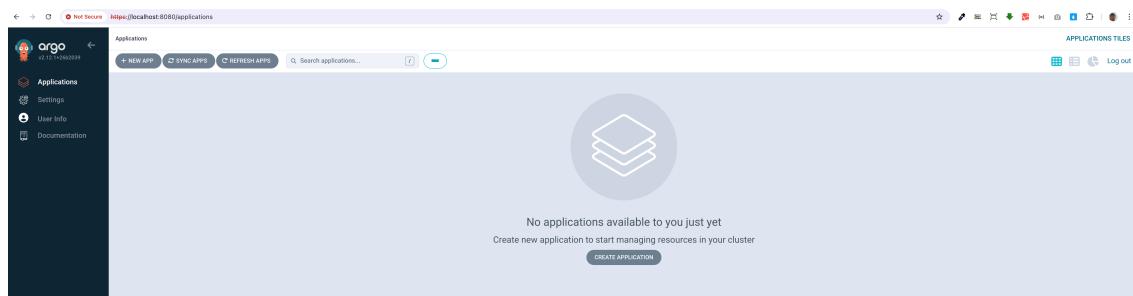
```
sd2793_aws_infrastructure git:(main) ✘ kubectl port-forward svc/argocd-server -n argocd 8080:443
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
```

2. Get password admin

2.1. `kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath={.data.password} | base64 --decode`

```
* sd2793_aws_infrastructure git:(main) ✘ kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath={.data.password} | base64 --decode
QpLaWYKkbP-rm75XG
```

3. Open browser and access to <https://localhost:8080>



3. Setup Application

1. Create a Project

The screenshot shows the ArgoCD UI for creating a project named 'devops-cicd'. The 'SUMMARY' tab is active, showing the following details:

- GENERAL**:
 - NAME: devops-cicd
 - DESCRIPTION: DevOps Practices
 - LINKS: (empty)
- SOURCE REPOSITORIES**: (empty)
- SCOPED REPOSITORIES**: Project has no source repositories

Below the summary, there are several sections:

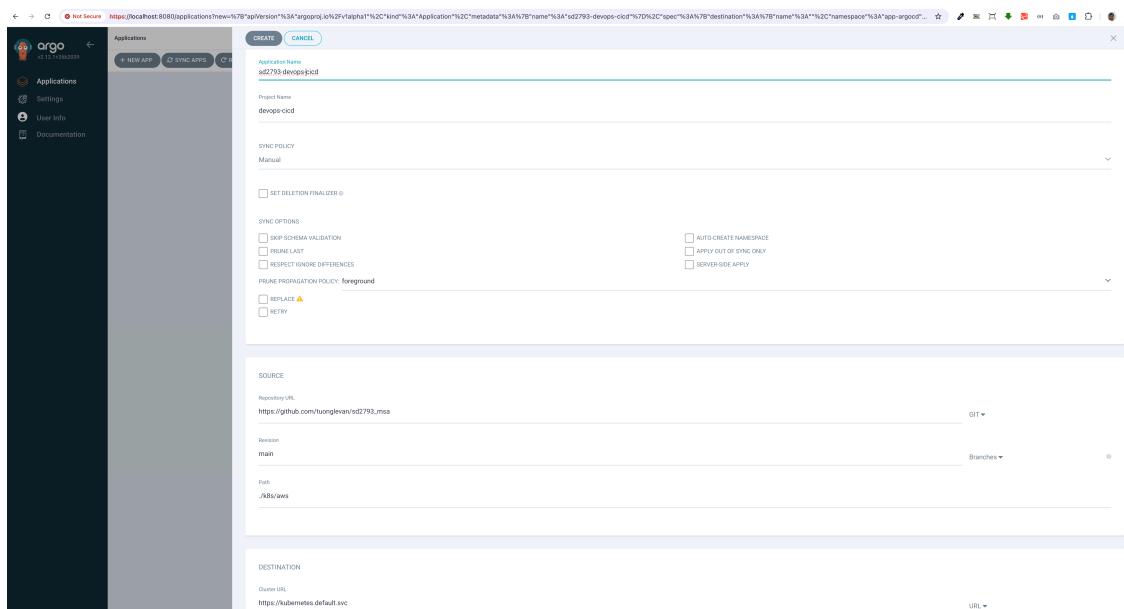
- DESTINATIONS**: (empty)
- SCOPED CLUSTERS**: Project has no destinations
- CLUSTER RESOURCE ALLOW LIST**: (empty)
- GPG SIGNATURE KEYS**: Project has no signature keys
- RESOURCE MONITORING**: (empty)

2. Create a new namespace for application

2.1. kubectl create namespace app-argocd

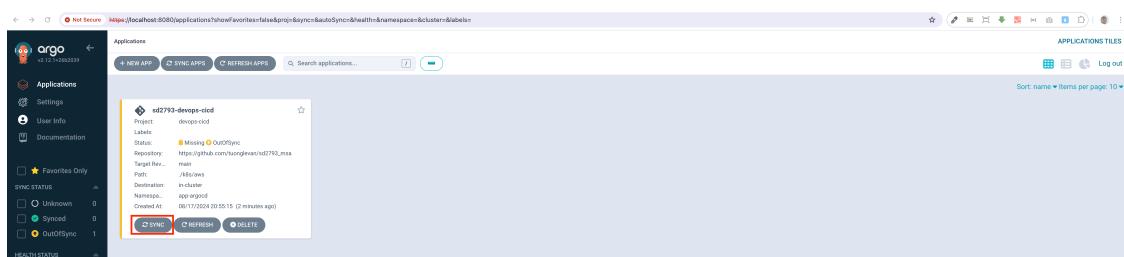
```
→ sd2793_aws_infrastructure git:(main) ✘ kubectl create namespace app-argocd
namespace/app-argocd created
```

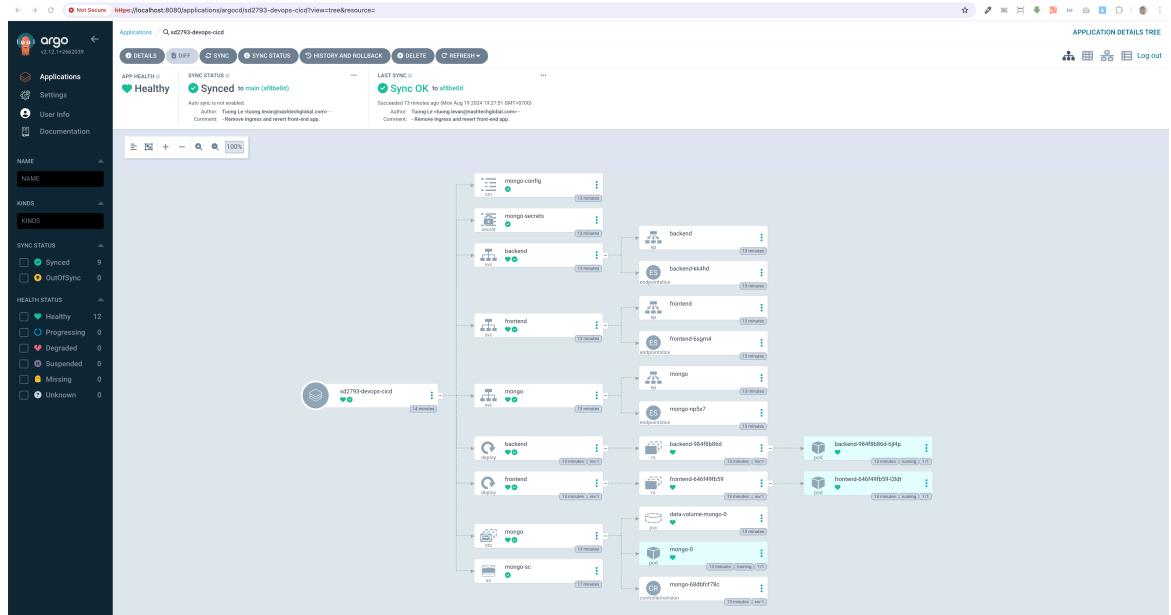
3. Create a new Application



4. Sync ArgoCD application with github

4.1. Go to Applications > choose application name > click SYNC button





4. Remove Stage CD in Jenkins and Update UI button

```
→ sd2793_aws_infrastructure git:(main) ✘ kubectl port-forward svc/frontend 4000:3000 -n app-argocd
Forwarding from 127.0.0.1:4000 -> 3000
Forwarding from [::1]:4000 -> 3000
Handling connection for 4000
Handling connection for 4000
Handling connection for 4000
```

VII. Prometheus and Grafana

1. Installation

```
+ sd2793_aws_infrastructure git:(main) ✘ helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
"prometheus-community" has been added to your repositories
+ sd2793_aws_infrastructure git:(main) ✘ helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
+ sd2793_aws_infrastructure git:(main) ✘ helm repo update
Hang tight while we grab the latest from your chart repositories...
... Ignoring package: prometheus from the 'prometheus-community' chart repository
... Successfully got an update from the 'stable' chart repository
Update Complete. *Happy Helm-ing!
+ sd2793_aws_infrastructure git:(main) ✘ helm install prometheus prometheus-community/kube-prometheus-stack
NAME: prometheus
LAST DEPLOYED: Sun Aug 18 16:24:46 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace default get pods -l "release=prometheus"
Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.
+ sd2793_aws_infrastructure git:(main) ✘
```

```
+ sd2793_aws_infrastructure git:(main) ✘ kubectl get all
NAME                                         READY   STATUS    RESTARTS   AGE
pod/alertmanager-prometheus-kube-prometheus-alertmanager-0   2/2     Running   0          2m13s
pod/backend-984f8b86d-pnc17                   1/1     Running   0          9m9s
pod/frontend-646f49fb59-hd8x9                 1/1     Running   0          9m8s
pod/mongo-0                                    0/1     Pending   0          9m10s
pod/prometheus-grafana-785b988595-5dmc7      3/3     Running   0          2m19s
pod/prometheus-kube-prometheus-operator-bb46f9945-pm2x2    1/1     Running   0          2m19s
pod/prometheus-kube-state-metrics-688d66b5b8-vg9d9       1/1     Running   0          2m19s
pod/prometheus-kube-prometheus-prometheus-0        2/2     Running   0          2m12s
pod/prometheus-prometheus-node-exporter-2ktkx      1/1     Running   0          2m19s
pod/prometheus-prometheus-node-exporter-c6nnpn     1/1     Running   0          2m19s

NAME                                         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
service/alertmanager-operated                ClusterIP  None         <none>        9093/TCP,9094/TCP,9094/UDP  2m13s
service/backend                            ClusterIP  172.20.202.229 <none>        3000/TCP          9m9s
service/frontend                           ClusterIP  172.20.254.127 <none>        3000/TCP          9m8s
service/kubernetes                         ClusterIP  172.20.0.1    <none>        443/TCP           39m
service/mongo                             ClusterIP  172.20.252.8  <none>        27017/TCP        9m10s
service/prometheus-grafana                 ClusterIP  172.20.60.203 <none>        80/TCP            2m19s
service/prometheus-kube-prometheus-alertmanager ClusterIP  172.20.76.128 <none>        9093/TCP,8080/TCP  2m19s
service/prometheus-kube-prometheus-operator  ClusterIP  172.20.135.197 <none>        443/TCP           2m19s
service/prometheus-kube-prometheus-prometheus ClusterIP  172.20.20.198 <none>        9090/TCP,8080/TCP  2m19s
service/prometheus-kube-state-metrics       ClusterIP  172.20.1.206  <none>        8080/TCP          2m19s
service/prometheus-operated                ClusterIP  None         <none>        9090/TCP          2m12s
service/prometheus-prometheus-node-exporter ClusterIP  172.20.64.106  <none>        9100/TCP          2m19s

NAME                                         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/prometheus-prometheus-node-exporter  2         2         2         2           2           kubernetes.io/os=linux  2m19s

NAME                                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/backend                      1/1     1           1           9m9s
deployment.apps/frontend                     1/1     1           1           9m8s
deployment.apps/prometheus-grafana          1/1     1           1           2m19s
deployment.apps/prometheus-kube-prometheus-operator 1/1     1           1           2m19s
deployment.apps/prometheus-kube-state-metrics 1/1     1           1           2m19s

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/backend-984f8b86d           1         1         1         9m9s
replicaset.apps/frontend-646f49fb59        1         1         1         9m8s
replicaset.apps/prometheus-grafana-785b988595 1         1         1         2m19s
replicaset.apps/prometheus-kube-prometheus-operator-bb46f9945 1         1         1         2m19s
replicaset.apps/prometheus-kube-state-metrics-688d66b5b8 1         1         1         2m19s

NAME                                         READY   AGE
statefulset.apps/alertmanager-prometheus-kube-prometheus-alertmanager 1/1   2m13s
statefulset.apps/mongo                       0/1   9m10s
statefulset.apps/prometheus-prometheus-prometheus      1/1   2m12s
```

2. Monitoring

1. Port Forward and Access UI

```
+ sd2793_aws_infrastructure git:(main) ✘ kubectl port-forward service/prometheus-grafana 8081:80
Forwarding from 127.0.0.1:8081 -> 3000
Forwarding from [::1]:8081 -> 3000
```

2. Open the Brow and access to <http://localhost:8081/> with username and password default

2.1. Username: admin

2.2. Password: prom-operator

The screenshot shows the Grafana home page at <http://localhost:8081/>. The main area displays the 'Welcome to Grafana' message and several introductory sections:

- Basic**: Steps to quickly set up Grafana.
- TUTORIAL**: 'DATA SOURCE AND DASHBOARDS' and 'Grafana fundamentals' sections.
- COMPLETE**: 'Add your first data source' and 'Create your first dashboard' buttons.
- Latest from the blog** sidebar:
 - Aug 16**: Behind the scenes of the OpenTelemetry Governance Committee.
 - Aug 15**: All about span events: what they are and how to query them.
 - Aug 14**: Grafana security release: Medium severity security fix for CVE-2024-8837.

3. Import Metrics to Dashboard

The screenshot shows the Grafana interface for importing a dashboard. On the left, there's a sidebar with navigation links like Home, Started, Dashboards, Explore, Alerting, Connections, and Administration. The main area is titled "Dashboards" and contains a search bar and a "Move" or "Delete" button. A large list of dashboards is displayed, each with a checkbox and a preview icon. Several checkboxes are checked, specifically for Kubernetes-related dashboards. To the right of the list, there's a column of "Tags" which include "alarmingmanager-metrics", "grafana", "dns", "etcd-metrics", "kubernetes-metrics", "kubernetes-node-metrics", "node-exporter-metrics", and "prometheus-metrics". At the top right, there are buttons for "New dashboard", "New folder", and "Import".

Dashboard ID: 1860 [Node Exporter Full](#)

This screenshot shows the "Import dashboard" dialog. It includes fields for "Published by" (rfmuz), "Updated on" (2024-05-22 23:07:35), and "Options" where the "Name" is set to "Node Exporter Full". Under "Dashboards", it says "Uniqueness (UID)" and provides a unique identifier. It also has sections for "Prometheus" and "Import". The "Import" button is highlighted.

4. Monitoring in the Dashboard UI.

