

Jenkins Infra Provisoining

Created Jenkins Server.

after provisoing of jenkins server > we will launch two new server one for Redis and second one for Postgres DB.

Once Redis Server we will provision like this :-

Install Redis on Ec2 Server -

```
sudo apt update
sudo apt install redis-server -y
sudo systemctl enable redis
sudo systemctl start redis
```

Edit Redis config: sudo vi /etc/redis/redis.conf set the following - bind 0.0.0.0

Then restart Redis: sudo systemctl restart redis

Ref link - <https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-redis-on-ubuntu-22-04>

Install Postgres on Ec2 Server -

```
sudo apt update
sudo apt install postgresql postgresql-contrib -y
sudo systemctl enable postgresql
sudo systemctl start postgresql
```

Create PostgreSQL User and Database

Imp - why we created user name and password and db name same because in code this has hard coded, but for the best practice is devloper can write these passwords on applicaion.properties or .env file, and we must use the same to communicate the db otherwise the communication will fall.

```
sudo -i -u postgres
psql
```

Inside psql, run:

```
CREATE USER postgres WITH PASSWORD 'postgres';
# CREATE DATABASE postgre OWNER postgres;
```

Allow Remote Access

Edit postgresql.conf: sudo nano /etc/postgresql/*/main/postgresql.conf

Set: listen_addresses = '*'

Then edit pg_hba.conf: sudo nano /etc/postgresql/*/main/pg_hba.conf

Add the following line:

```
host    all             all             0.0.0.0/0        md5
```

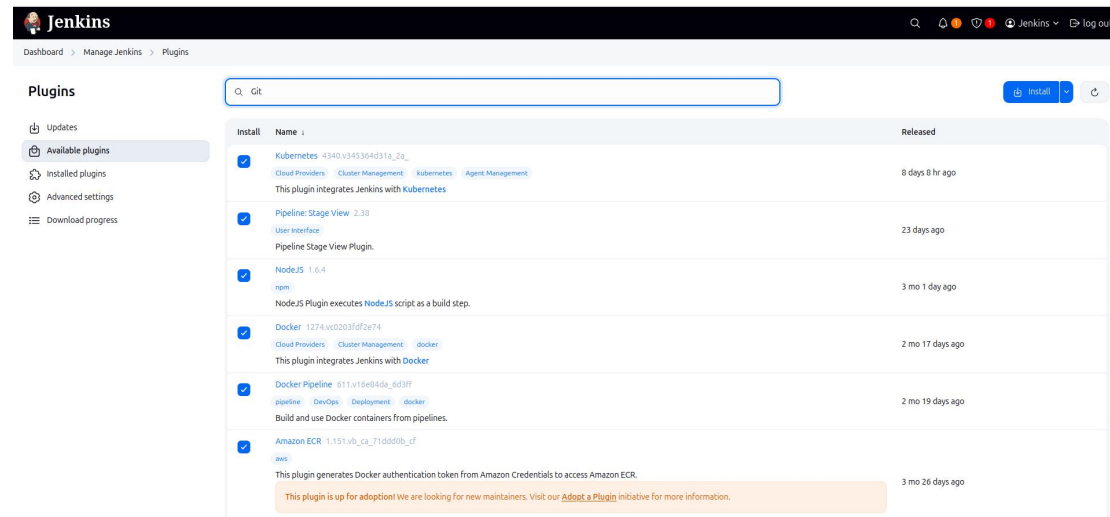
Restart PostgreSQL: sudo systemctl restart postgresql

TEST NETWORK CONNECTIVITY (From Jenkins)

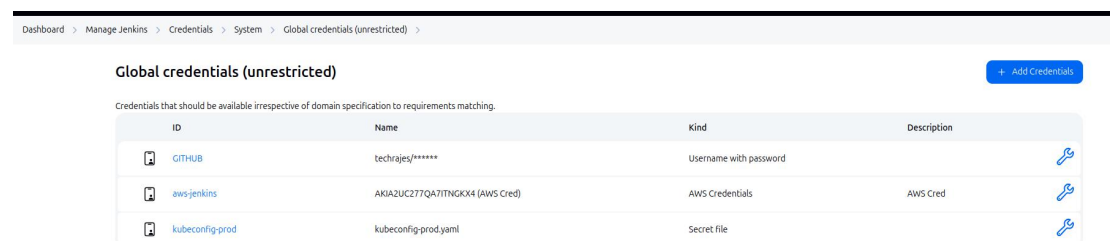
```
telnet <redis-private-ip> 6379
telnet <postgres-private-ip> 5432
```

After all these setup now, we will provision the Jenkins Server Setup and a vpn to access the private jenkins server.

We will install these required plugins -



Then these Credentials required :-



As per our Jenkins pipeline code, this pipeline does -

1. Clones source code from GitHub.
2. Logs into AWS ECR.
3. Builds a Docker image.
4. Pushes it to ECR.
5. Deploys the image to an EKS cluster using Helm.

Get the kubeconfig from EKS - This writes the kubeconfig to ~/.kube/config inside the Jenkins agent and copy the data and save a file on your local and use this file on Jenkins credentials secret.

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind
Secret file

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

File
Browse... kubeconfig-prod.yaml

ID ?
kubeconfig-prodNew

Description ?
This will help to deploy the helm charts on EKS

Create

After creating all these setups, create three repos like this.

Jenkins

Dashboard >

+ New Item

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

Build Queue
No builds in the queue.

Build Executor Status
0/2

Icon: S M L

| S | W | Name | Last Success | Last Failure | Last Duration |
|---|---|---------|-----------------|-----------------|---------------|
| ✓ | ☁ | result | 6 days 21 hr #4 | 6 days 23 hr #3 | 6.1 sec |
| ✓ | ☁ | Voicing | 6 days 21 hr #9 | 7 days 19 hr #5 | 6.6 sec |
| ✓ | ☁ | worker | 6 days 21 hr #8 | 6 days 22 hr #7 | 9.6 sec |

Add description

and in Github all these three pipeline codes are there, we will pull these codes and apply the pipeline.

Ingress we will create manually.