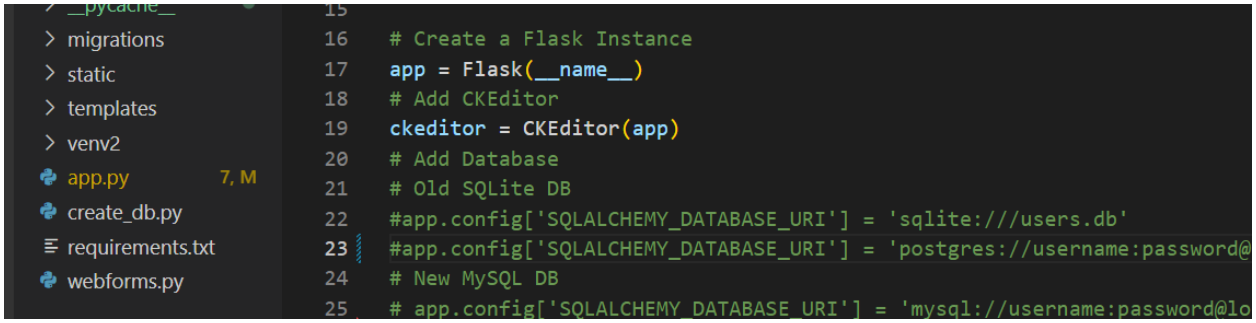
Deployment #5

Welcome to Deployment 5!! Time to deploy to your customized VPC. You will need to follow the steps below.

1. Before you start!!!

- Create a Postgres Database on AWS
- Enter database credentials on line 23 of app.py



2. Install Jenkins on an EC2 if you haven't already:

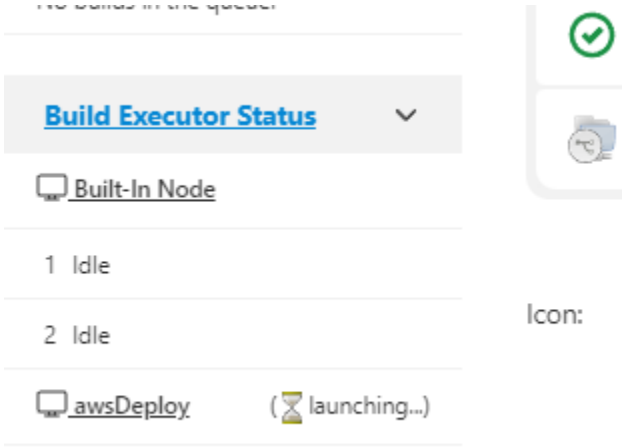
- You do not have to recreate a Jenkins server in your VPC. Highly recommend you use your Jenkins server from the default VPC!!

3. Create an EC2 in your Public Subnet of your VPC:

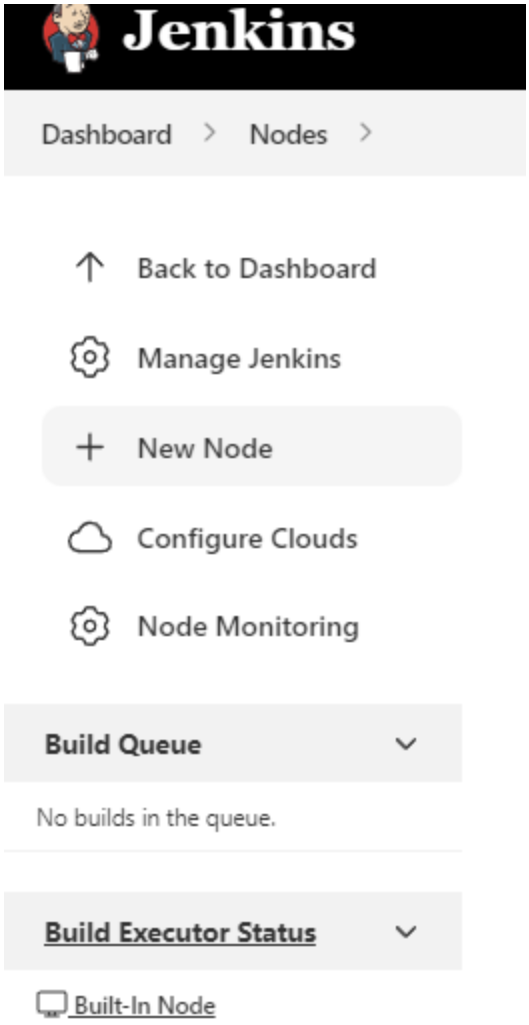
- The Ubuntu EC2 will need ports number: 22 and 5000 open.
- Install packages: **default-jre, python3-pip, python3.10-venv and nginx.**

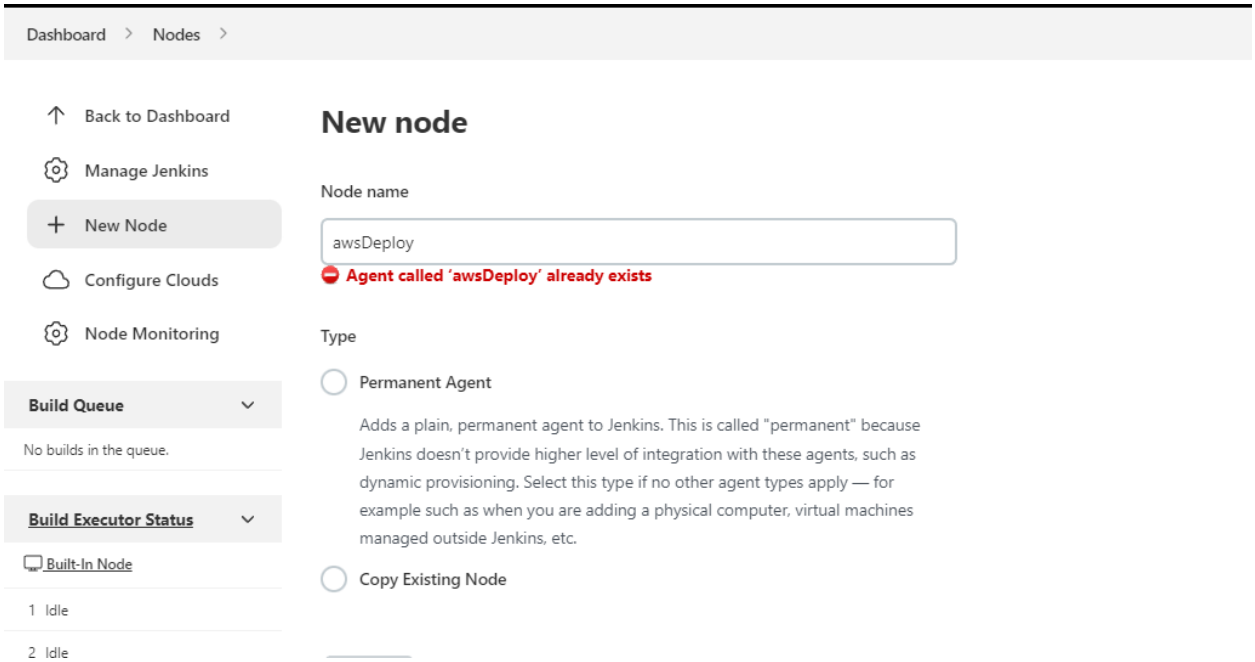
4. Configure and connect a Jenkins agent to Jenkins:

- Enter your Jenkins server and Select the Build Executor Status:



- Next Select “+ New Node” to configure and add the agent. Enter the node name “awsDeploy” select “Permanent Agent” and then create.


Jenkins
Dashboard
Nodes
Back to Dashboard
Manage Jenkins
New Node
Configure Clouds
Node Monitoring
Build Queue
No builds in the queue.
Build Executor Status
Built-In Node



- Now enter the configurations below:
  Name: **awsDeploy**
  Description: **Deployment server**
  Number of executors: **1**
  Remote root directory: **/home/ubuntu/agent**
  Labels: **aweDeploy**
  Usage: **only build jobs with label….**
  Launch method: **launch agents via ssh**
  Host: {Enter the public IP of your EC2 in the Public subnet and not this text}
  Credentials: see below
  Host key verification strategy: **non verifying verification strategy**

Availability: **keep this agent online as much as possible**

awsDeploy

Name ?

awsDeploy

Description ?

Deployment server

Number of executors ?

1

Remote root directory ?

/home/ubuntu/agent

Labels ?

awsDeploy

Usage ?

Only build jobs with label expressions matching this node

Launch method ?

Launch agents via SSH

Host ?

54.163.30.187

Credentials ?

ubuntu (SSH-CALI)

+ Add

Host Key Verification Strategy ?

Non verifying Verification Strategy ?

Advanced...

- Credential steps:
  Select “Add” => “Jenkins”=>Kind:”SSH username with private key”
  Enter the ID, Description, username
  To add the key, select “Enter Directly” => select “add” => paste the private key into the white box and save.

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Global credentials (unrestricted)

Kind

Username with password

Username with password
AWS Credentials
GitHub App
SSH Username with private key
Secret file
Secret text
Certificate

Username ?

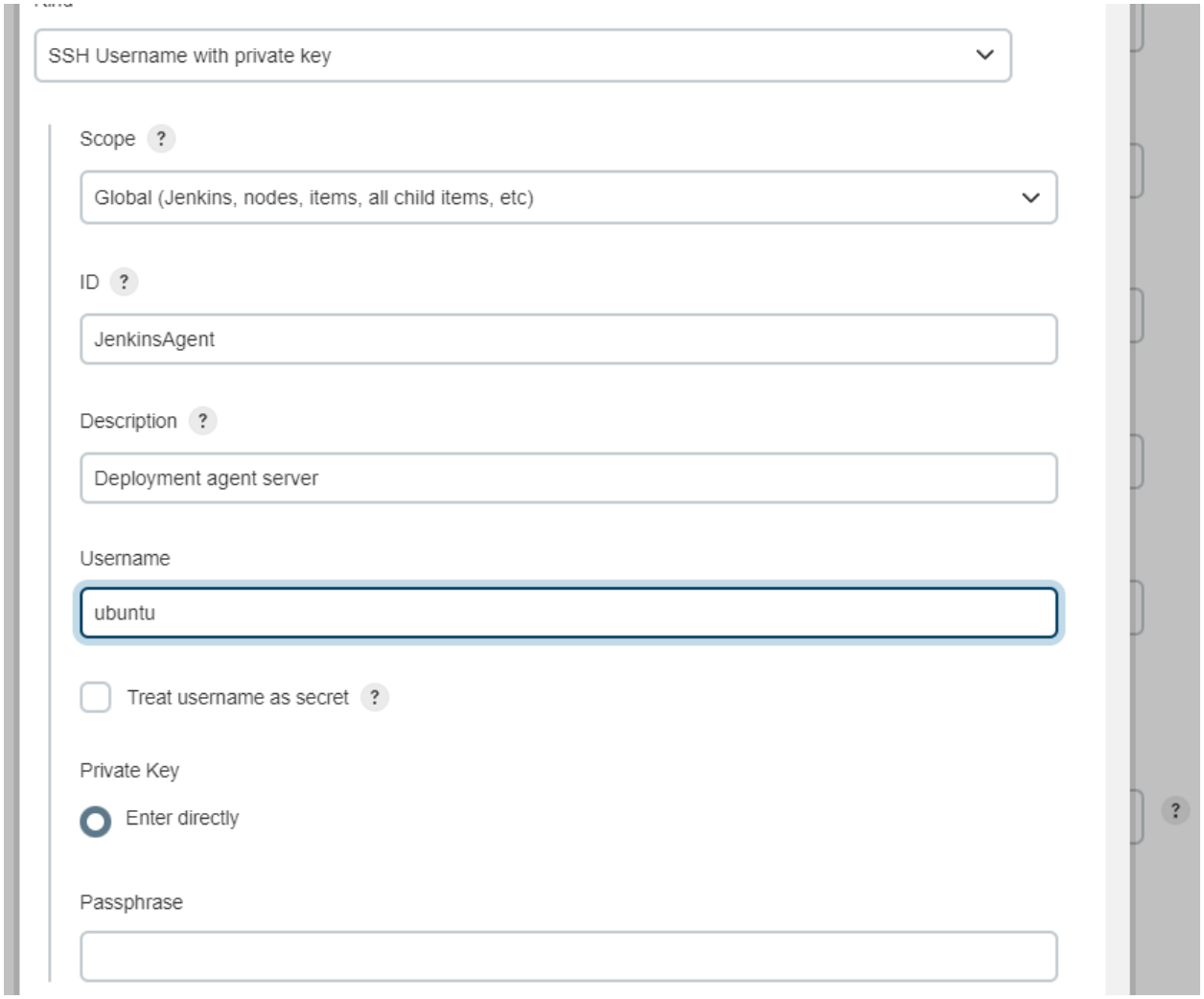
Treat username as secret ?

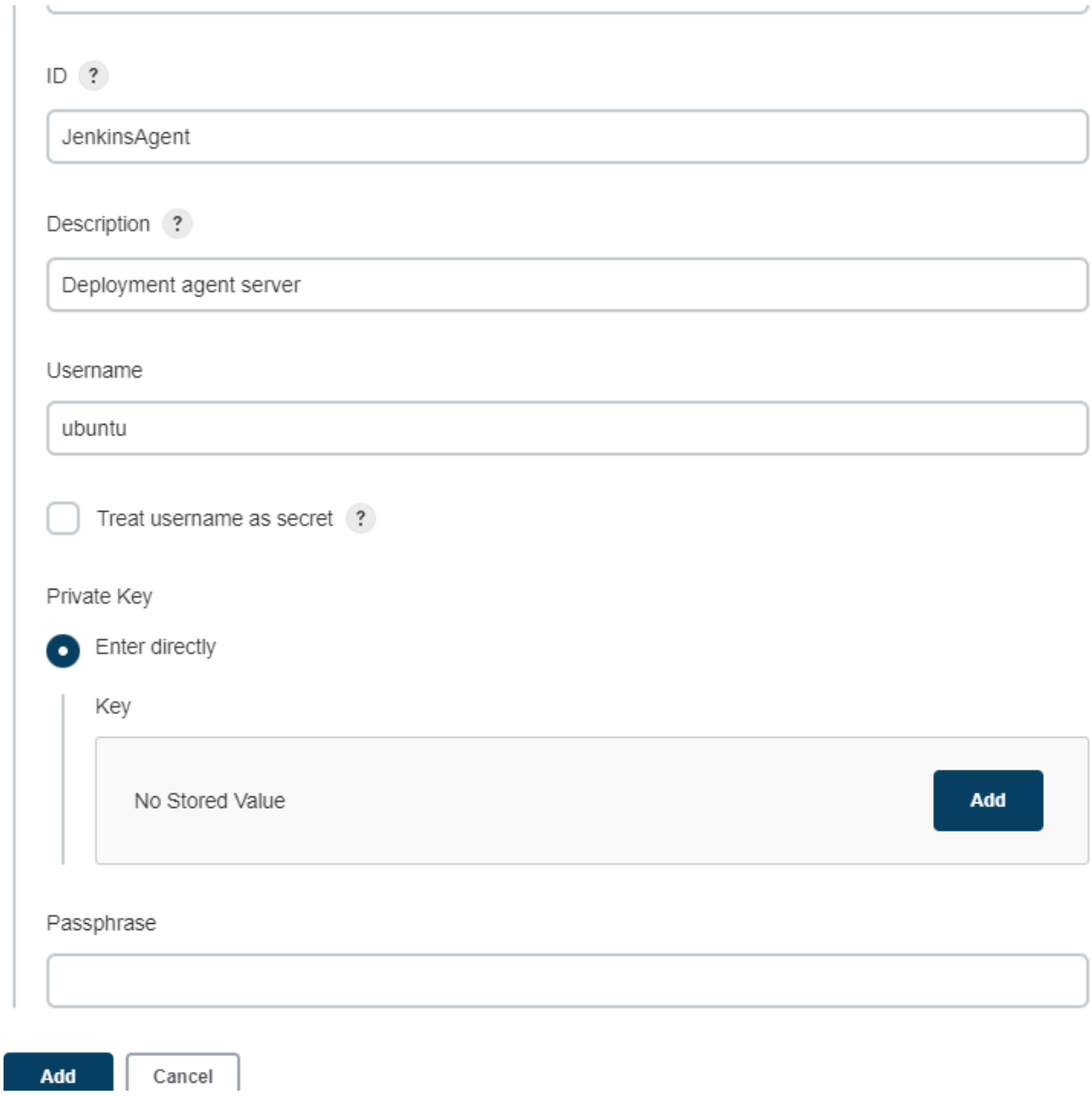
Password ?

ID ?

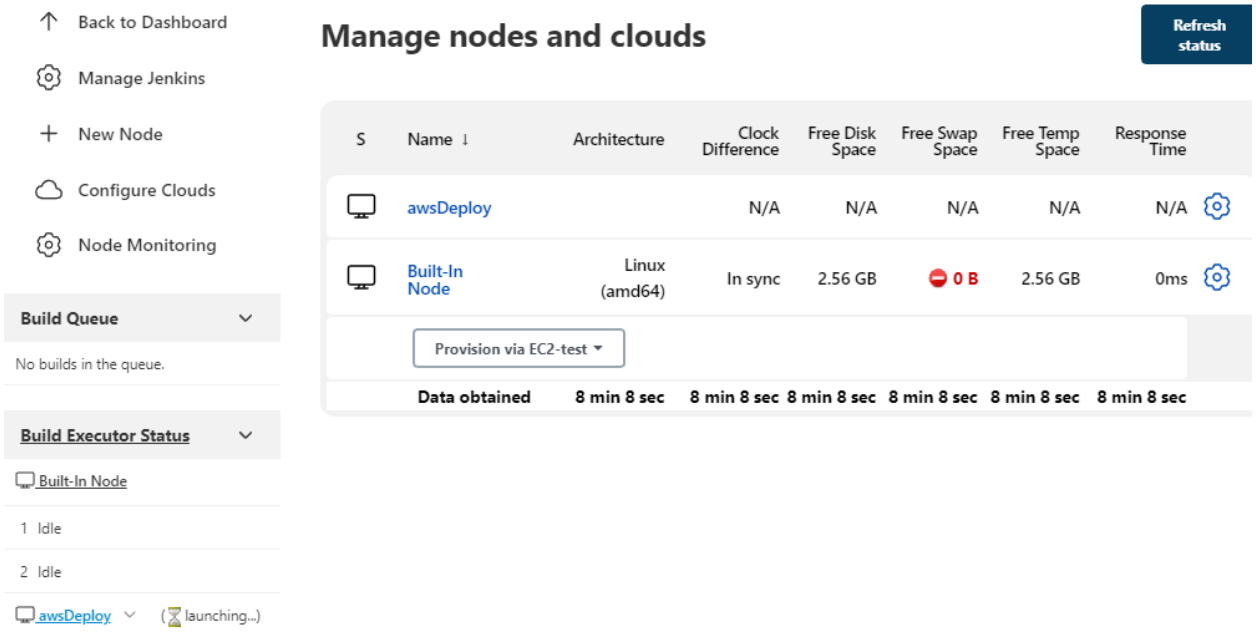
Description ?

Add Cancel


SSH Username with private key
Scope
Global (Jenkins, nodes, items, all child items, etc)
ID
JenkinsAgent
Description
Deployment agent server
Username
ubuntu
Treat username as secret
Private Key
Enter directly
Passphrase



- Save the configurations and wait for Jenkins to connect to the agent. It should look like what you see below:



5. Create a Pipeline build in Jenkins:

- Before you build your pipeline, SSH into the EC2 in your VPC and then nano into the “/etc/nginx/sites-enabled/default” file.
- First change the port from 80 to 5000:

```
server {
        listen 5000 default_server;
        listen [::]:5000 default_server;
```

- Scroll down to where you see “location” and replace it with the text below:

```
location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
```

- Now edit the Jenkinsfile in your repo to the script below:

You will need to download the Jenkins plugin “Pipeline Keep Running Step”

```
pipeline {
  agent any
   stages {
    stage ('Build') {
      steps {
        sh '''#!/bin/bash
        python3 -m venv test3
        source test3/bin/activate
        pip install pip --upgrade
        pip install -r requirements.txt
        export FLASK_APP=app
        flask run &
        '''
     }
   }
    stage ('test') {
      steps {
```

```
        sh '''#!/bin/bash
        source test3/bin/activate
        py.test --verbose --junit-xml test-reports/results.xml
        '''
      }

      post{
        always {
          junit 'test-reports/results.xml'
        }

      }
    }
   }
   stage ('Clean') {
      agent{label 'awsDeploy'}
      steps {
        sh '''#!/bin/bash
        if [[ $(ps aux | grep -i "gunicorn" | tr -s " " | head -n 1 | cut  -d " " -f 2) != 0 ]]
        then
          ps aux | grep -i "gunicorn" | tr -s " " | head -n 1 | cut  -d " " -f 2 > pid.txt
          kill $(cat pid.txt)
          exit 0
        fi
        '''
      }
    }
    stage ('Deploy') {
      agent{label 'awsDeploy'}
      steps {
      keepRunning {
        sh '''#!/bin/bash
        pip install -r requirements.txt
        pip install gunicorn
        python3 -m gunicorn -w 4 application:app -b 0.0.0.0 --daemon
        '''
       }
      }
    }
  }
}
```

- Log back into Jenkins and configure a multi branch pipeline or just a single pipeline build. Make sure you connect Jenkins to your GitHub Repo and then start your build!!

6. Repeat the steps above for a second build:

- Make sure you create the second instance in another AZ
- You can name this agent **awsDeploy2**
- **Change the color of the navigation bar for the second deployment for awsDeploy2 agent**
- **Now create an application load balancer and connect the two deployed applications to the ALB**

Create documentation:

Document your process for adding an application load balancer. Also, include a summary of what you configured in

each part of your deployment. Lastly, include any issues you ran into and what you did to fix them.

> **Note:** Submit your work by uploading your work to a repo or the forked repo. Then submit the link to the repo via LMS.