README

SWE645 – Assignment 2

Contributions:

1) Dana Jamous:
- Built and applied the steps in part-1
- Documented part-1

2) Erman Sert:
- Built and applied the steps in part-2 + Part-3
- Documented part-2 + part-3
- Created videos to demonstrate part-2 + Part-3

3) Lubna Fatima:
- Created the video to demonstrate part-1

4) Asra Naseem:
- Contributed in part-2 + part-3 documentation

URls

- **Url for the application deployed on kubernetes:** http://a5aac318f2c7a4bb4b99b0ca14ace68d-1691372941.us-east-1.elb.amazonaws.com/StudentSurvey/
- **GitHub URL**:https://github.com/esert-gmu/SWE645-AEDL-HW2
- **AWS URL of your homepage**https://44.202.120.220/StudentSurvey/survey.html

- **PART-1:**
  - **Create a Github account and push eclipse application to it:**
    - Go to Github website https://github.com and create an account
    - Create new repository on Github and copy it's URL
    - Go to eclipse , commit and push your project to GitHub:
      - In Eclipse, right-click on your project in the Package Explorer or Project Explorer.
      - Select "Team" > "Commit" to open the Git staging view.
      - Select All.
      - Provide a commit message describing your changes and click "Commit and Push".
      - Choose the repository and branch you want to push to (origin/master is the default branch in a newly created repository).
      - Click "Finish" to commit and push your project to GitHub.
  - **Package your Eclipse application into a WAR**

- Right-click on the project Explorer view.
- Select Export.
- Select WAR File.

- **Build and run docker image**
  - Check if Docker is already installed: Open a terminal or command prompt and run the following command: `docker --version`
  - If Docker is installed, it will display the version number. If not, you will need to install Docker from (https://www.docker.com/products/docker-desktop)
  - Open the directory where your WAR file is located.
  - Create a new file in that directory and name it "Dockerfile" (without any file extension).
  - Open the Dockerfile.
  - Add the following content to the Dockerfile:

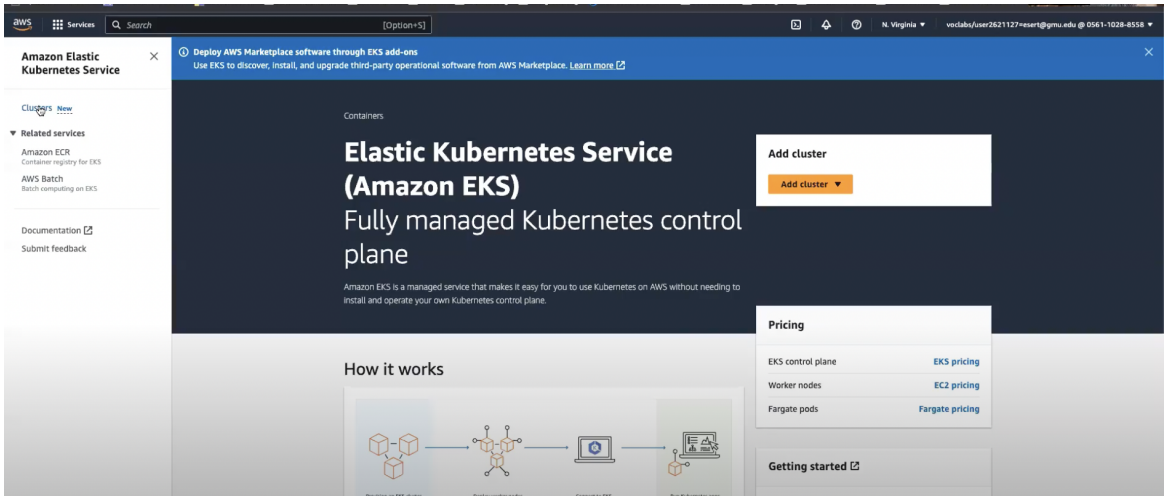
```
FROM tomcat:9.0-jdk15
COPY <warFile name>.war /usr/local/tomcat/webapps/
```

  - Open a terminal or command prompt.
  - Navigate to the directory where your Dockerfile and WAR file are located.
  - Run the following command to build the Docker image:
    `docker build -t <your-docker-image-name> .`
  - Once the image is successfully built, run the Docker container using the following command:
    `docker run -p 8182:8080 <your-docker-image-name>`
  - Docker will start the container and your application will be accessible at `http://localhost:8182/<war-file-na me>`/

- **To push your Docker image to Docker Hub**
  - Create account on Docker Hub (**https://hub.docker.com/**)
  - Open a terminal or command prompt.
  - Log in to Docker Hub using the `docker login` command.
  - Enter your Docker Hub username and password when prompted.
  - Tag your Docker image with your Docker Hub username and repository name.
    `docker tag <docker-image-name> <dockerhub-username>/<repo-name>:tag`
  - Push the tagged image to Docker Hub using the `docker push <dockerhub-username>/<new-image-name>:<tag>` command.
- **To upload war file on EC2 instance:**
  - Launch WinSCP and create a new session by providing the following details:

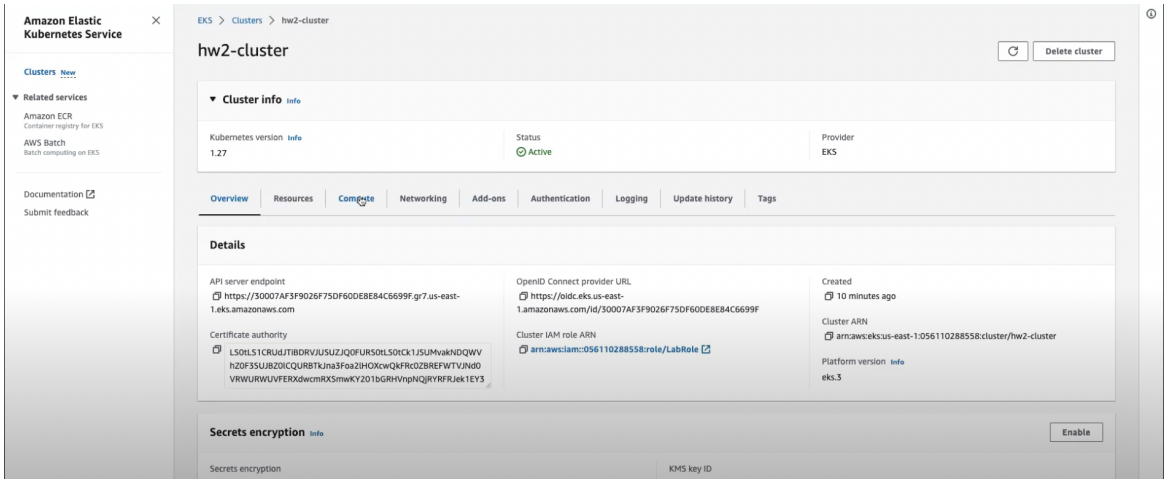
- Host name: Enter the public IP address or DNS name of your EC2 instance. (In our case is “44.202.120.220”).
- Port number: Set it to 22 (default for SSH).
- User name: Enter the username for your EC2 instance (In our case is “bitnami”).
- Click on "Advanced" and navigate to the "SSH" section.
- Private key file: Provide the path to your private key file (.pem) associated with the EC2 instance.
- Save the session settings and click on "Login" to establish a connection with the EC2 instance.
- Once connected, you'll see the local file system (your computer) on the left and the remote file system (EC2 instance) on the right.
- Navigate to the directory “/opt/bitnami/tomcat/webapps” on the remote file
- Drag and drop the WAR file from the local file system (left) to the remote file system (right) to initiate the upload.
- Now you can access the application using the following URL https://44.202.120.220/StudentSurvey/survey.html

- **PART-2**
  - Before the following steps, please make sure that you do `aws configure` to configure your account. These steps assumes that you have aws-cli installed on your environment and it should be configured using `aws configure`.
  - In order to deploy our application, we need a kubernetes cluster. In this homework, we are going to use the managed kubernetes cluster, EKS, provided by AWS. Here are the steps to configure/provision the kubernetes cluster:
    - Go to AWS Management Console and search for EKS
    - Once you are in the EKS service in AWS, click on add cluster.



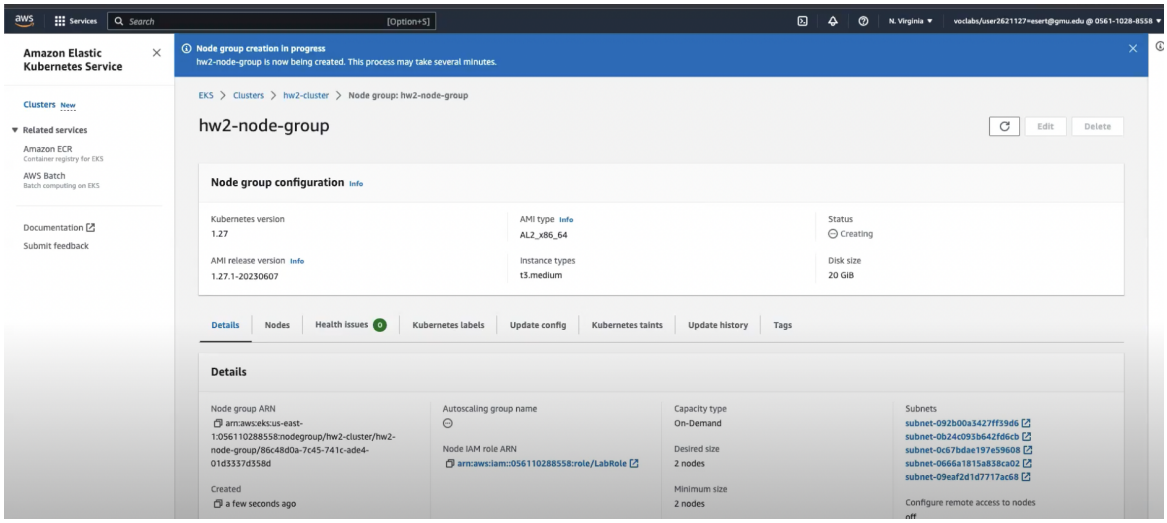
EKS service

- Type the cluster name and everything else should be left as default.
- Once you confirm, the cluster will be initializing. After the cluster is initialized, follow the steps below.
- Go to your clusters under EKS, and click on the cluster you just created.



hw2-cluster was created and is Active

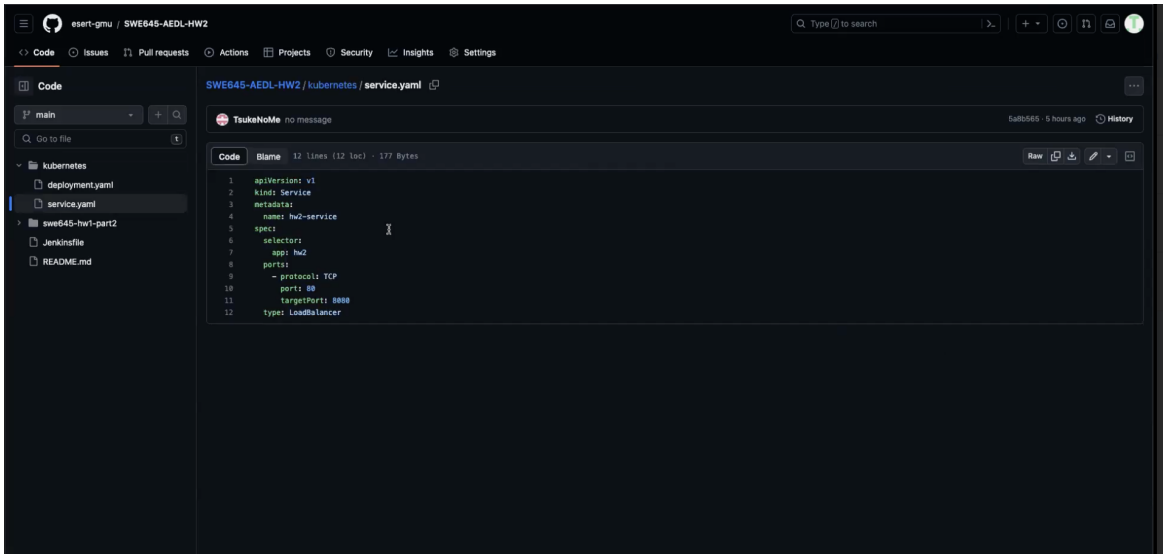
- Click on Compute and then click on Add Node Group under Node Groups.
- Give the node group a name and leave everything else as default and proceed.
- Wait for the group nodes to be active and at this point your cluster should be ready.



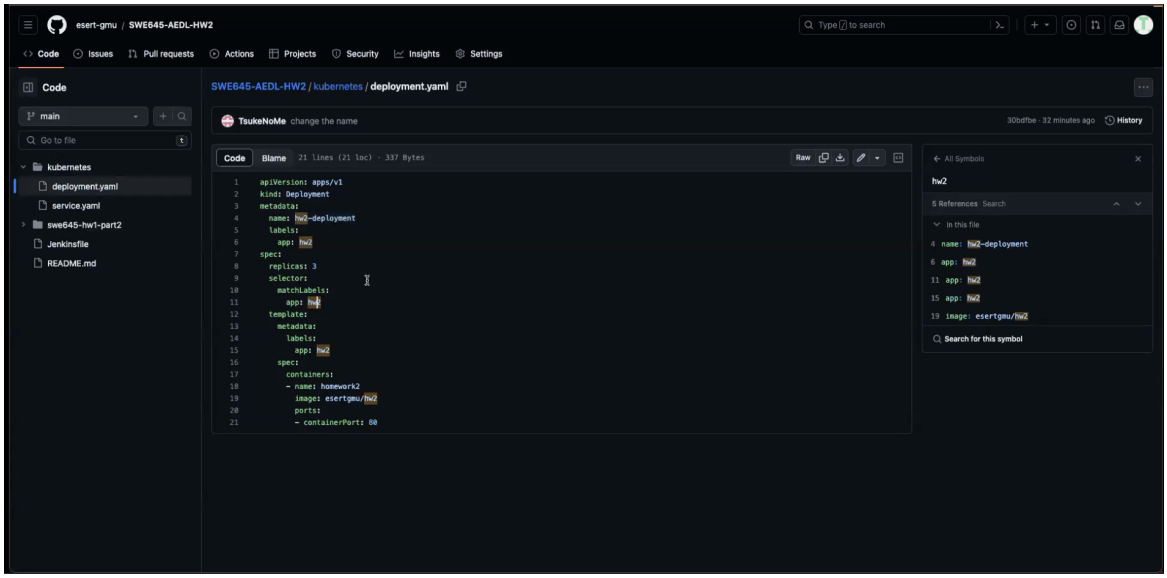
hw2-node-group is created

- The next step would be exporting the kubeconfig from your cluster. Running this command will add the kubeconfig to your kube file: `aws eks update-kubeconfig --region <Name of the region that was selected> --name <Name of the cluster that is created>`
- You can now view your kubeconfig file by running `cat ~/.kube/config`

- You also need to create certain resource files in order to run the application. For this particular application, a deployment and service file should be fine.
- The deployment file should contain replication as 3 as we want resiliency for this project and the service file can be a load balancer which will forward port 80 to 8080 of the container. The related files can be found at: https://github.com/esert-gmu/SWE645-AEDL-HW2/tree/main/kubernetes



service.yaml



deployment.yaml

- **PART-3**
  - The next step is setting up the Jenkins in order to establish the pipeline between your cluster and your code.
    - The first step for setting up Jenkins is launching an ubuntu EC2 instance. Please follow these steps to launch an EC2 ubuntu instance and make sure to save the keypair for authentication: https://docs.aws.amazon.com/efs/latest/ug/gs-step-one-create-ec2-resources.html
    - Once your EC2 instance is ready, go to the inbounds of your EC2 and open port 8080 for us to be able to communicate with Jenkins.
    - In order for us to be able to install Jenkins, we need JAVA installed into the machine. Please login to the machine using SSH and install JAVA with `sudo apt install default-jre`

- Then following that, please run the following commands in order per installation documentation located at: https://pkg.jenkins.io/debian/

  This is the Debian package repository of Jenkins to automate installation and upgrade. To use this repository, first add the key to your system:

```
curl -fsSLhttps://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

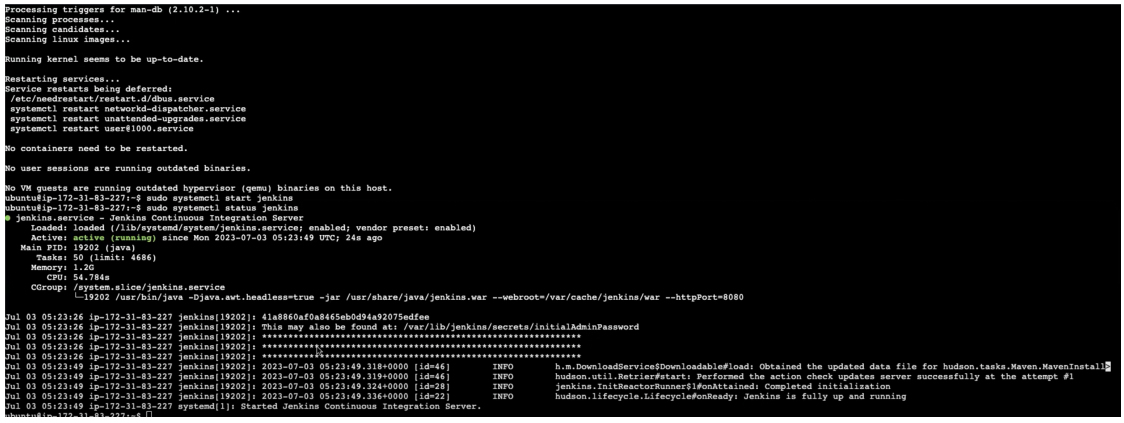
  Then add a Jenkins apt repository entry:

```
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
```

  Update your local package index, then finally install Jenkins:

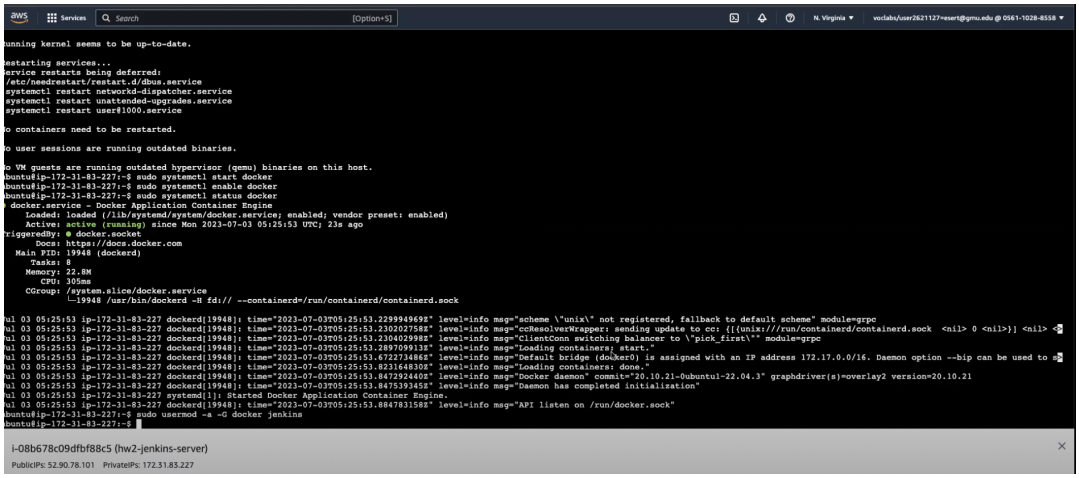
```
sudo apt-get update
sudo apt-get install fontconfig openjdk-11-jre
sudo apt-get install jenkins
```

- The steps above should install Jenkins on your ubuntu machine. But we also need to start the service using the following commands:
  - `sudo systemctl start jenkins`
  - `sudo systemctl status jenkins` → this command should show you whether your Jenkins service is running or not.

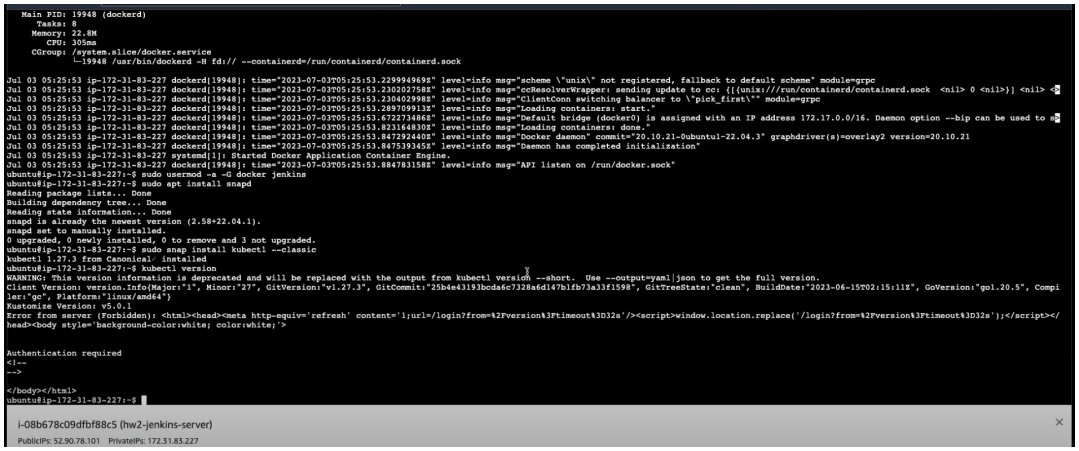


- Once you confirm that your Jenkins service is running, before we setup Jenkins, we should install docker, kubectl and aws-cli on the Jenkins machine for Jenkins integration.
  - To install and start docker, please run the following commands:
    - `sudo apt install docker.io`

- `sudo systemctl start docker`
- `sudo systemctl enable docker`
- `sudo systemctl status docker` → should show you if the docker service is running or not.
- `sudo usermod -a -G docker jenkins` → give permission to Jenkins for running docker

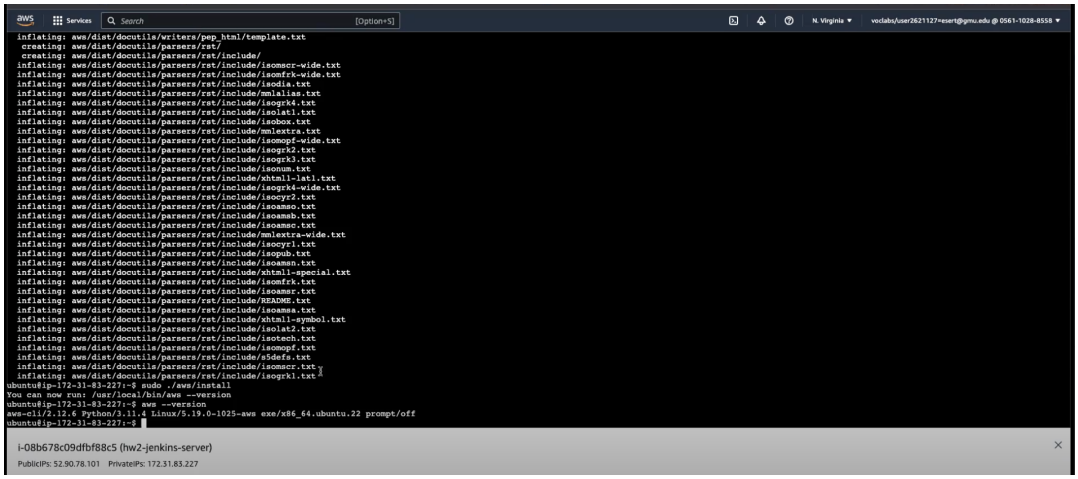


- Once you confirm that the docker service is running, now we can install kubectl on the ubuntu machine by using the following commands:
  - `sudo apt install snapd`
  - `sudo snap install kubectl --classic`

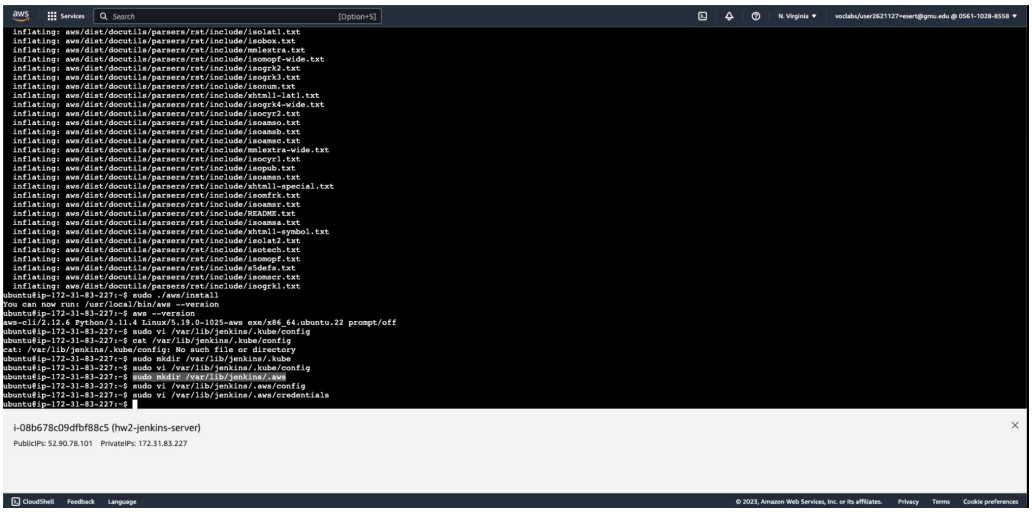


- Once you confirm that the `kubectl` is installed using `kubectl version` , you can proceed with the installation for aws cli which we will use alongside kubectl authentication. To install aws cli, follow these steps (the steps are located at https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html):
  - `curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"`
  - `sudo apt-get install unzip`

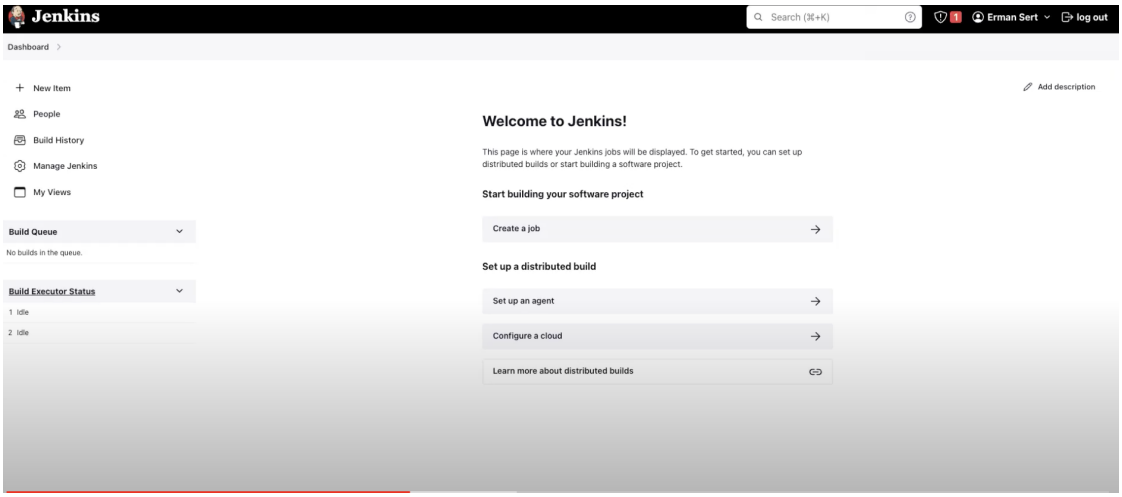
- `unzip awscliv2.zip`
- `sudo ./aws/install`



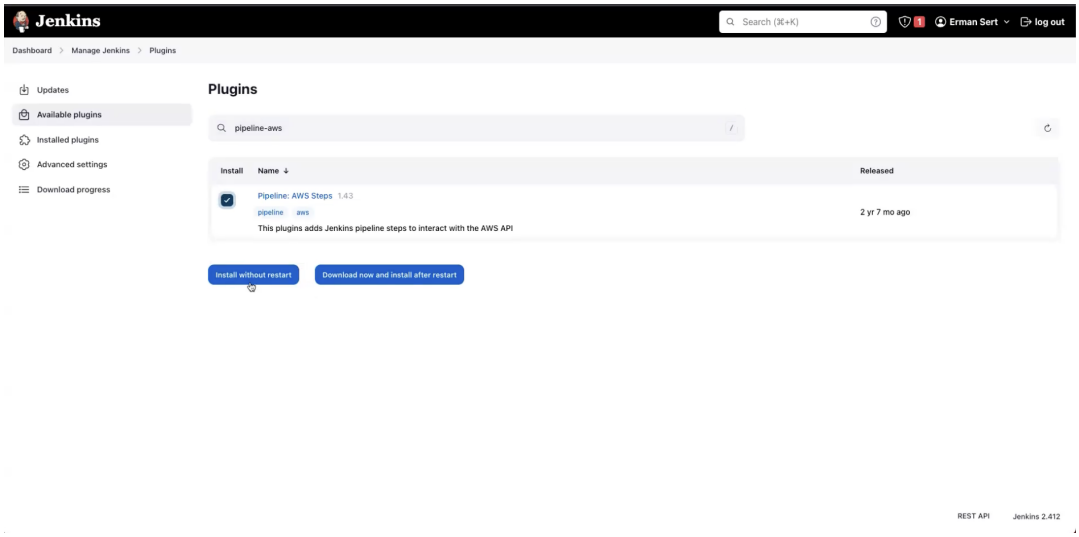
- Since this environment is an AWS Learner environment, we do not have access to manage the IAM resources. That is being the case, we cannot really assign the correct roles to the EC2 instance and have it access to the K8 cluster we created. Therefore, we need to add our AWS config and Kube Config to the Jenkins machine for connectivity. It is preferable that we would use IAM roles for this or a separate user but since we cannot do it, having the same AWS Config and Kube Config we use for our own AWS Learner user should be sufficient. To do that, please follow these steps:
  - It is assumed that you set up both kubectl and aws-cli on your local for retrieving the the kubeconfig. Please use `aws eks update-kubeconfig --region <Region Name> --name <Cluster Name>` for retrieving the kubeconfig from AWS (if you have not done it in the previous steps).
  - You should now be able to check your kubeconfig with `cat ~/.kube/config`
  - Since, you should have setup your aws cli and configured your aws account by now, you should also be able to see your aws config and credentials using: `cat ~/.aws/config` and `cat ~/.aws/config`
  - Please note the values you see in your kube config, aws config and aws credentials.
  - Now, please login to your Jenkins ubuntu machine and add your aws config, aws credential and kube config using the following commands:
    - `sudo mkdir /var/lib/jenkins/.kube && sudo vi /var/lib/jenkins/.kube/config` → and paste your kubeconfig file contents here
    - `sudo mkdir /var/lib/jenkins/.aws && sudo vi /var/lib/jenkins/.aws/config` → and paste your aws config file contents here
    - `sudo vi /var/lib/jenkins/.aws/credentials` → and paste your aws credentials file contents here



- Now you can login to Jenkins. Copy public DNS of your EC2 instance and then go to http://<EC2 Instance DNS>:8080 and then you should see the Jenkins setup page.
- You can grab the initial admin password by running: `sudo cat /var/lib/jenkins/secrets/initialAdminPassword`
- Enter the password and follow on screen instructions to setup a username and password for the admin user. Once the setup is complete, in order for Jenkins to pick up the latest changes, type /restart to the end of your Jenkins url and restart Jenkins.

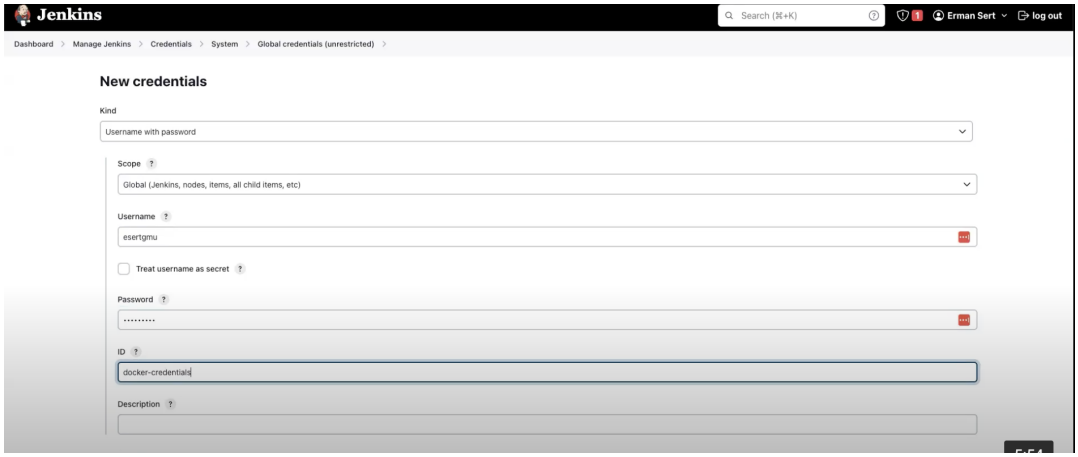


- Log back in and install the following plugins:
  - Go to Dashboard → Select Manage Jenkins → Click on Plugins → Search for aws-pipeline and install the plugin (https://github.com/jenkinsci/pipeline-aws-plugin) (pipeline-aws)
  - Go to Plugins page again → search for `docker-workflow` and install the plugin



aws-plugin

- Next, restart Jenkins again by going to the http://<jenkins url>:8080/restart
- Next, add your docker credentials that you created to the Jenkins by following these steps:
  - Click on your username at the right top
  - Click on credentials
  - Click on global next to System
  - Click on add credentials and add your docker username and password and have the id as docker-credentials



- Next, we need to set up a Jenkins job. To set up the job, please follow these steps:
  - Go to dashboard
  - Click on New Item
  - Type the name of your project
  - Select Pipeline and click on OK

- Select Poll SCM from the options and input `* * * * *` into the schedule box
- Then under pipeline, please select `pipeline script from scm`
- Then select git under SCM
- Input your repository URL (e.g. https://github.com/esert-gmu/SWE645-AEDL-HW2.git)
- Under branch specifier, replace master with main and click on Save
- Your Jenkins file should look like the following and should be in the root directory:

```
pipeline {
    agent any
    environment {
        DOCKERHUB_CREDS = credentials('docker-credentials')
    }
    stages {
        stage('Check the repository out') {
            steps {
                echo 'Getting the latest'
                checkout scm
            }
        }

        stage('Display Creds') {
            steps {
                echo "Docker Hub Username: ${DOCKERHUB_CREDS_USR}"
                echo "Docker Hub Password: ${DOCKERHUB_CREDS_PSW}"
            }
        }


        stage('Build and push the image') {
            steps {
                script {
                    dockerImage = docker.build('esertgmu/hw2', './swe645-hw1-part2')
                    docker.withRegistry('https://registry.hub.docker.com', 'docker-credentials') {
                        dockerImage.push('latest')
                    }
                }
            }
        }

        stage('Deploy using the deployment.yaml') {
            steps {
                script {
                    sh 'kubectl apply -f ./kubernetes/deployment.yaml'
                    sh 'kubectl apply -f ./kubernetes/service.yaml'
                }
            }
        }
    }
}
```

- And finally, we need to grab the public url for the load-balancer in EKS. We can do that by querying the services with `kubectl get services` and grabbing the url from the load balancer result. Once we have the URL, we can just load the url using http://eks_url/StudentSurvey/ which in our case was

http://a5aac318f2c7a4bb4b99b0ca14ace68d-1691372941.us-east-1.elb.amazonaws.com/StudentSurvey/

- Notes on Git Repository. Your git repository structure should be like the following: