Suresh Thirunavukkarasu

Batch | D011

**CAPSTONE PROJECT**

Application Deployment

**Table of Contents**

**Submission Appendix1**

**Application Steps3**

**Docker4**

**Bash Scripting8**

**Version Control12**

**Docker Hub17**

**Jenkins19**

**AWS45**

**Monitoring52**

**Submission Appendix:**

Github repo URL: <https://github.com/tsuresh218/capstone_project.git>

Deployed Site URL: <http://13.210.167.184:80/>

Docker images name:

<https://hub.docker.com/repository/docker/tsuresh218/dev/general> - tsuresh218/dev:latest

<https://hub.docker.com/repository/docker/tsuresh218/prod/general> - tsuresh218/prod:latest

Screenshots of below can be found in the word document (D011\_Capstone\_Suresh Thirunavukkarasu.docx)

<https://github.com/tsuresh218/capstone_project/blob/master/D011_Capstone_Suresh_Thirunavukkarasu.docx>

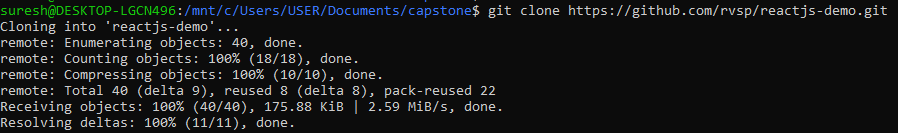
* Jenkins (Login Page, Configuration settings, execute step commands)
* AWS (EC2-Console, SG Configs)
* Docker hub repo with image tags
* Deployed Site Page
* Monitoring Health Check Status

**Application Steps:**

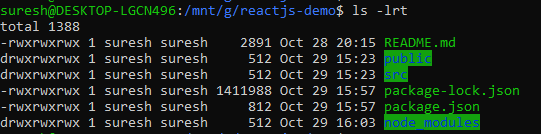
Clone the below mentioned repo and deploy the application (Run the application in port 80 [HTTP] )

<https://github.com/rvsp/reactjs-demo.git>

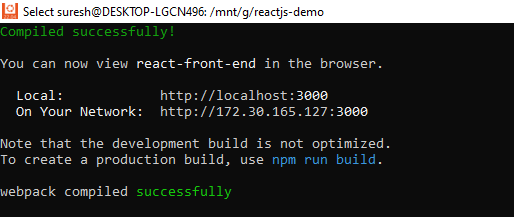
* Above repo has been cloned to the local machine as below.



**Application tested successfully in local system before building the docker image:**



* **Using npm start command, compiled and start-up the application successfully**



<http://localhost:3000/>



**Steps to Dockerize the application**



**Docker**

***Dockerize the application by creating a Dockerfile***

* Dockerfile has been created to build a node image



# Start your image with a node base image

FROM node:14

# The /app directory should act as the main application directory

WORKDIR /app

# Copy the app package and package-lock.json file

COPY package\*.json ./

# Copy local directories to the current local directory of our docker image (/app)

COPY . .

COPY ./src ./src

COPY ./public ./public

# Install node packages, install serve, build the app, and remove dependencies at the end

RUN npm install \

    && npm install -g serve \

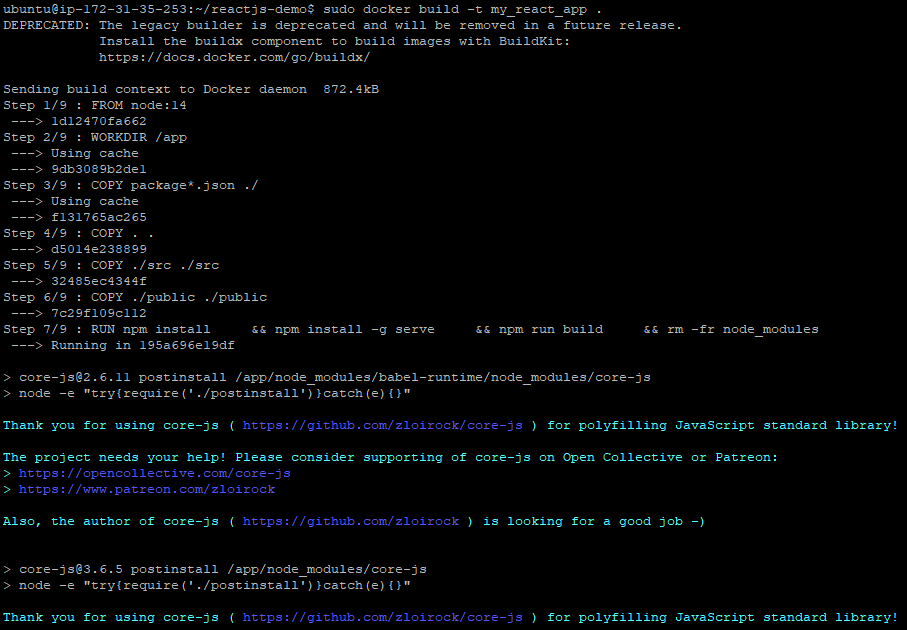
    && npm run build \

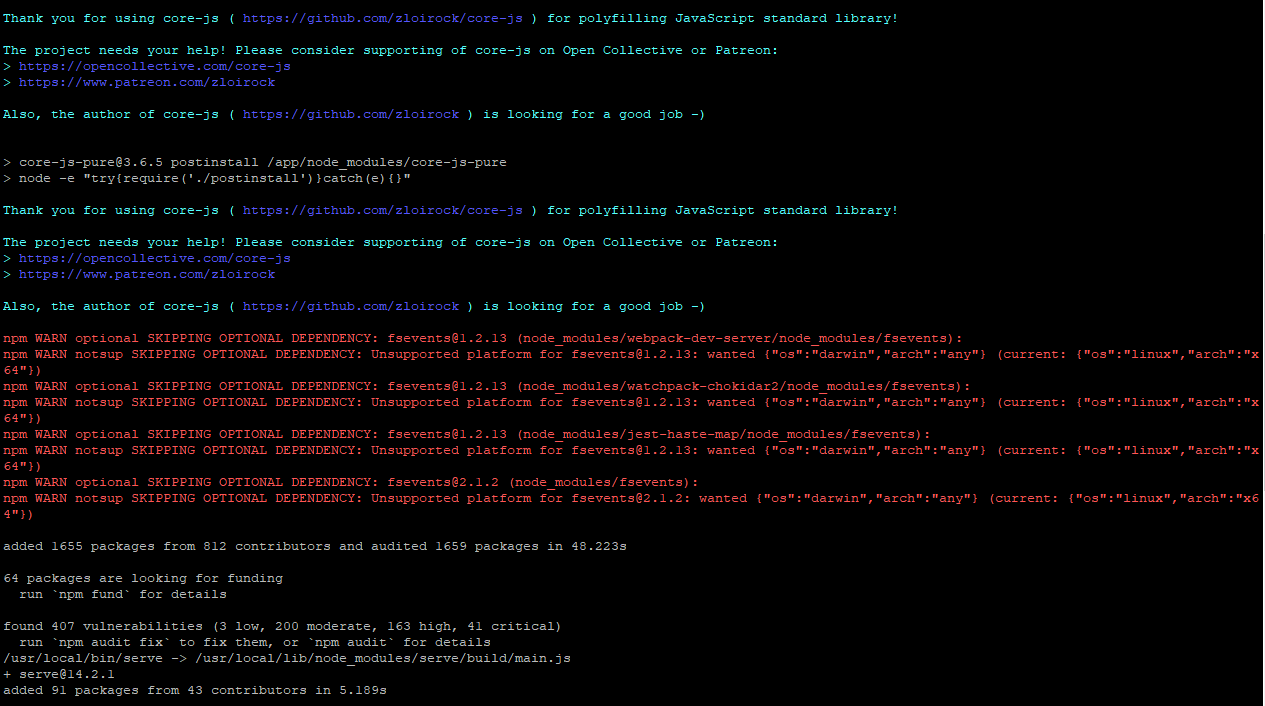
    && rm -fr node\_modules

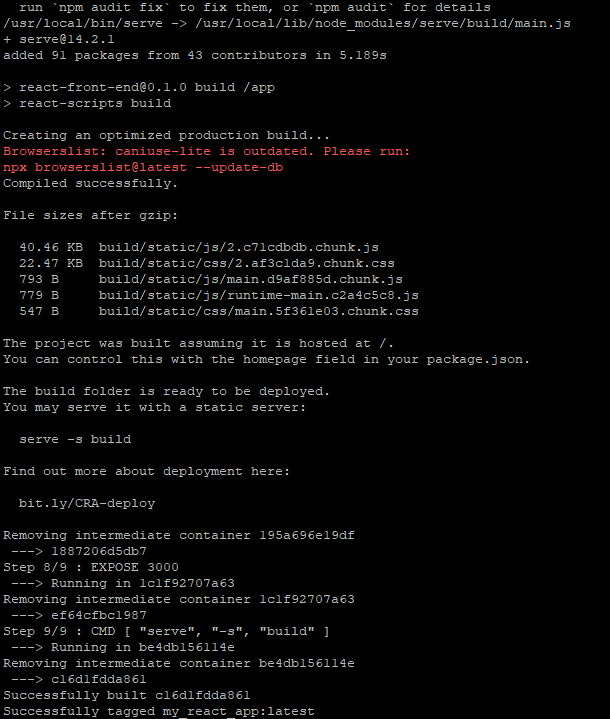
EXPOSE 3000

# Start the app using serve command

CMD [ "serve", "-s", "build" ]







***Create a docker-compose file to use the above image***

* docker-compose file has been created to use the above image



version: '3'

services:

  reactjs:

    image: my-react-app:latest

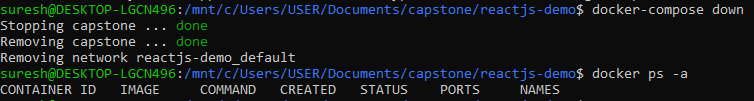
    container\_name: capstone

    ports:

      - "80:3000"

* Using docker-compose commands able to start and stop the container as below





**Bash Scripting**

***build.sh – for building docker images***

* build.sh script helps to build the docker image and run using the docker-compose file and it checks if docker container is running



#!/bin/bash

# Define the variables

IMAGE\_NAME='my\_react\_app'

TAG='latest'

CONTAINER\_NAME='capstone'

DOCKERFILE\_PATH='/mnt/c/Users/USER/Documents/capstone/reactjs-demo'

# Check if Docker is running

if ! docker info > /dev/null 2>&1; then

echo "Docker does not seems to be running, run it first and retry"

exit 1

fi

#Build Docker image

docker build -t $IMAGE\_NAME:$TAG .

# Check if Docker image creation was successful

if [ $? -eq 0 ]; then

    echo "Docker image '$IMAGE\_NAME:$TAG' has been built successfully."

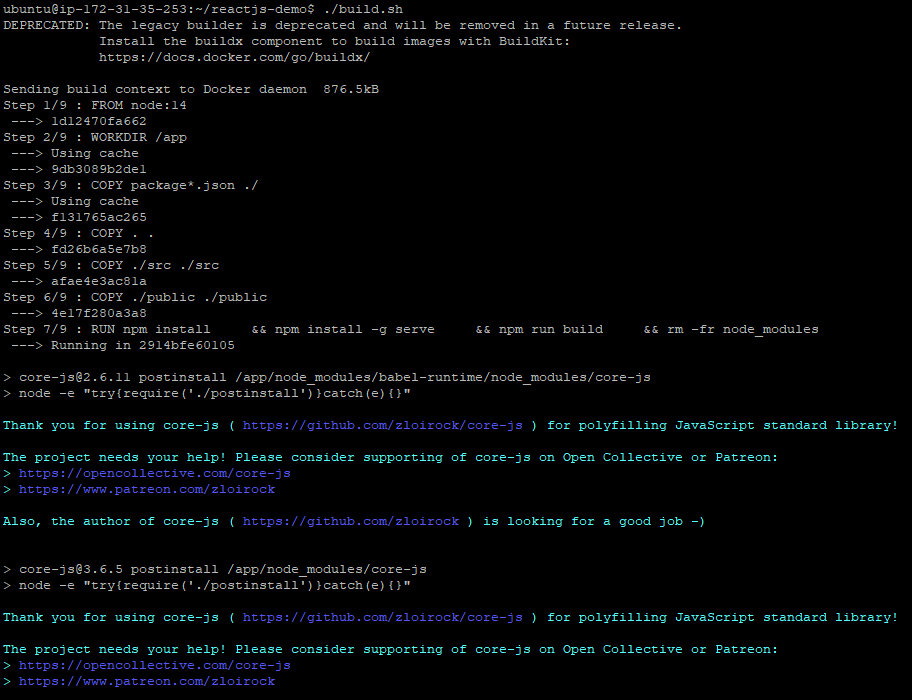
else

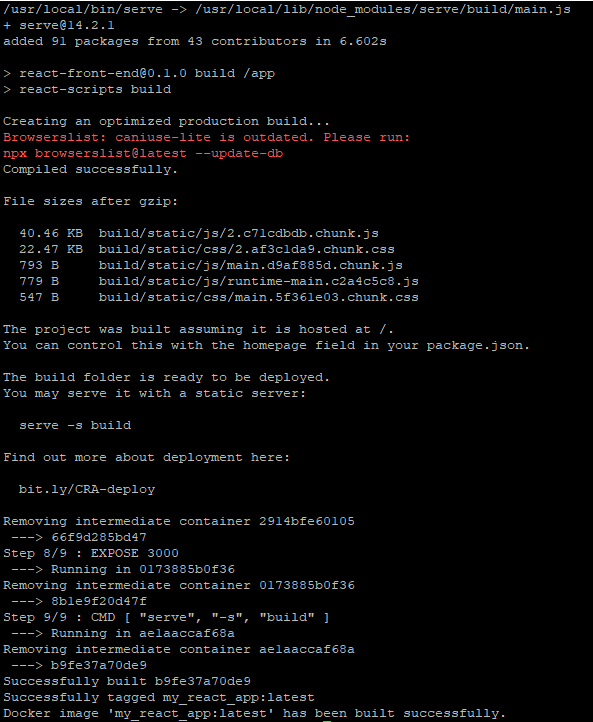
    echo "Failed to build the Docker image '$IMAGE\_NAME:$TAG'."

    exit 1

fi

**build.sh script output**





***deploy.sh – to deploy the image using docker compose file***

* deploy.sh script helps to deploy the build docker image and run using the docker-compose file and it checks if docker container is running fine

#!/bin/bash

# Path to the directory containing docker-compose.yml

DOCKERCOMPOSE\_PATH='/mnt/c/Users/USER/Documents/capstone/reactjs-demo'

CONTAINER\_NAME='capstone'

# Navigate to the path

cd $DOCKERCOMPOSE\_PATH

#Run the Docker image

docker-compose up -d

# Check if Docker image is running

if [ $(docker inspect -f '{{.State.Running}}' $CONTAINER\_NAME) = "true" ]; then

    echo "Docker container '$CONTAINER\_NAME is deployed and running"

else

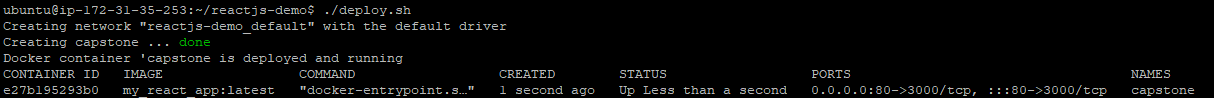
    echo "Docker container '$CONTAINER\_NAME is not deployed and running"

fi

# Check the docker container is running

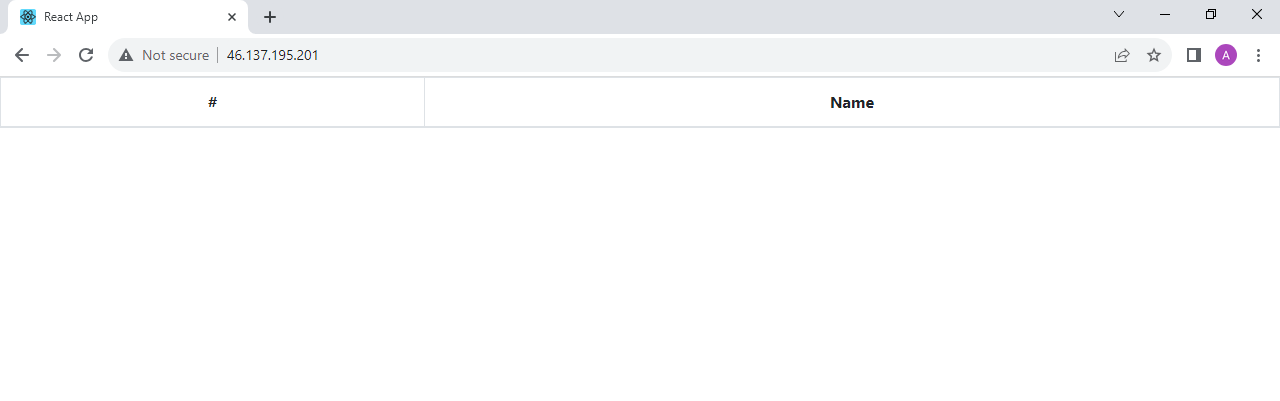
docker ps -a

**deploy.sh script output**



**Here’s the application output which is running locally:**

<http://46.137.195.201:8080/>



**Version Control**

***git\_dev\_push.sh – for pushing the code to git repository***

* git\_dev\_push.sh script helps to push the code via CLI commands from local directory to my github dev branch repository as shown below

#!/bin/bash

# Variables

REPO\_URL='https://github.com/tsuresh218/capstone\_reactjs.git'

LOCAL\_DIRECTORY='/mnt/c/Users/USER/Documents/capstone/reactjs-demo'

REPO\_DIRECTORY='/mnt/c/Users/USER/Documents/capstone/capstone\_reactjs'

NEW\_BRANCH\_NAME\_1='dev'

DOCKERIGNORE\_FILE='.dockerignore'

GITIGNORE\_FILE='.gitignore'

# Clone the repo into your local directory

 git clone $REPO\_URL

# Navigate to local directory

cd $LOCAL\_DIRECTORY

# Copy the contents to local git repo

cp -R . $REPO\_DIRECTORY

# Change to the cloned repository folder

cd $REPO\_DIRECTORY

# Create a dev branch

git checkout -b $NEW\_BRANCH\_NAME\_1

echo "New branch $NEW\_BRANCH\_NAME\_1 was created"

# Check if .dockerignore exists

if [ -f "$DOCKERIGNORE\_FILE" ]; then

    echo "$DOCKERIGNORE\_FILE found"

else

    echo ".git

    \*.md

    " > $DOCKERIGNORE\_FILE

fi

# Check if .gitignore exists

if [ -f "$GITIGNORE\_FILE" ]; then

    echo "$GITIGNORE\_FILE found"

else

    echo ".DS\_Store

    node\_modules/

    dist/

    \*.log

    " > $GITIGNORE\_FILE

fi

# Add files to the local repository

git add .

# Commit your changes

git commit -m "Initial commit to include all the files to dev branch"

# Add remote repository

git remote add origin $REPO\_URL

# Push the changes to dev branch on Github

if git push origin $NEW\_BRANCH\_NAME\_1; then

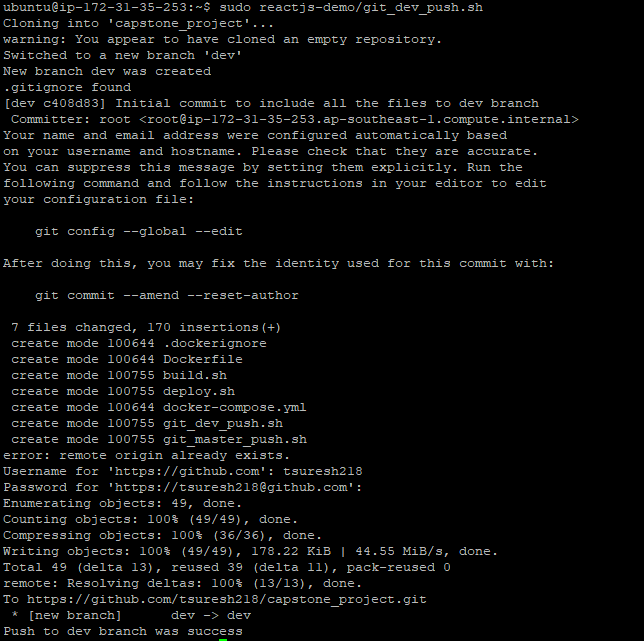
    echo "Push to $NEW\_BRANCH\_NAME\_1 branch was success"

else

    echo "Push to $NEW\_BRANCH\_NAME\_1 branch was failed"

fi

**git\_push.sh output**



***git\_master\_push.sh – to create a master branch***

* git\_master\_push.sh script helps to create master branch via CLI as shown below

#!/bin/bash

# Variables

REPO\_DIRECTORY='/mnt/c/Users/USER/Documents/capstone/capstone\_reactjs'

NEW\_BRANCH\_NAME\_1='master'

# Change to the cloned repository folder

cd $REPO\_DIRECTORY

# Create a dev branch

git checkout --orphan $NEW\_BRANCH\_NAME\_1

echo "New branch $NEW\_BRANCH\_NAME\_1 was created"

# Remove the files

git rm -rf . > /dev/null 2>&1

# Commit your changes

git commit --allow-empty -m "Initial commit to master branch"

# Push the master branch on Github

if git push origin $NEW\_BRANCH\_NAME\_1; then

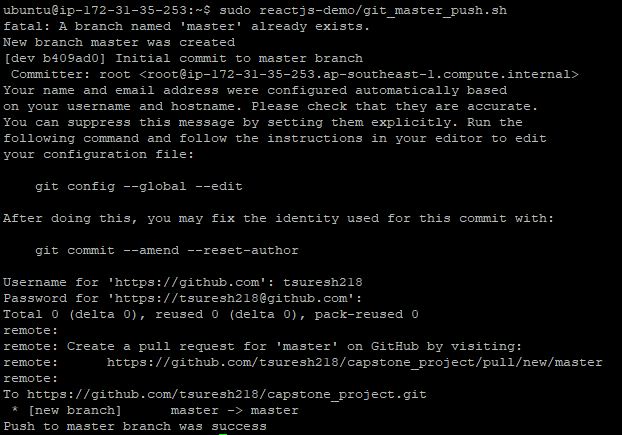
    echo "Push to $NEW\_BRANCH\_NAME\_1 branch was success"

else

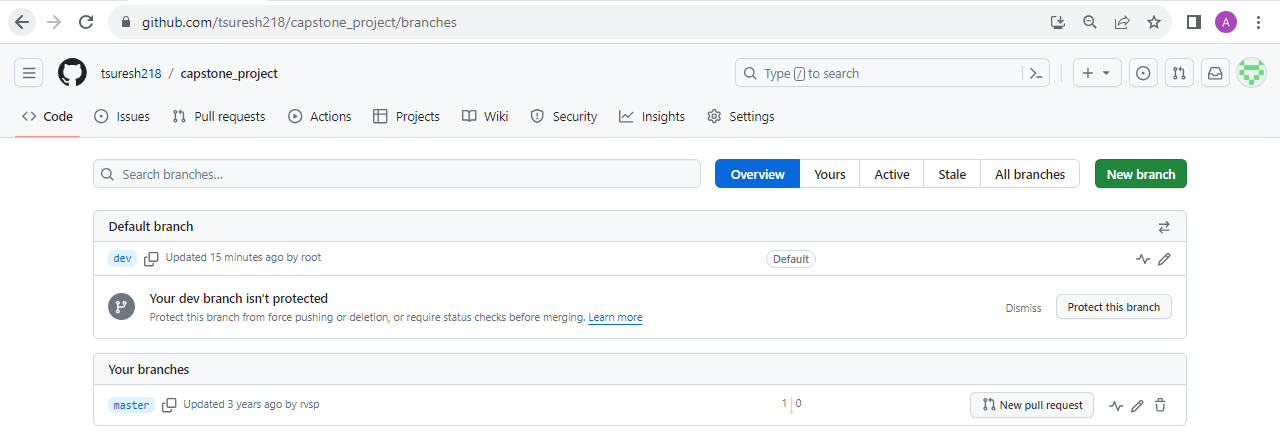
    echo "Push to $NEW\_BRANCH\_NAME\_1 branch was failed"

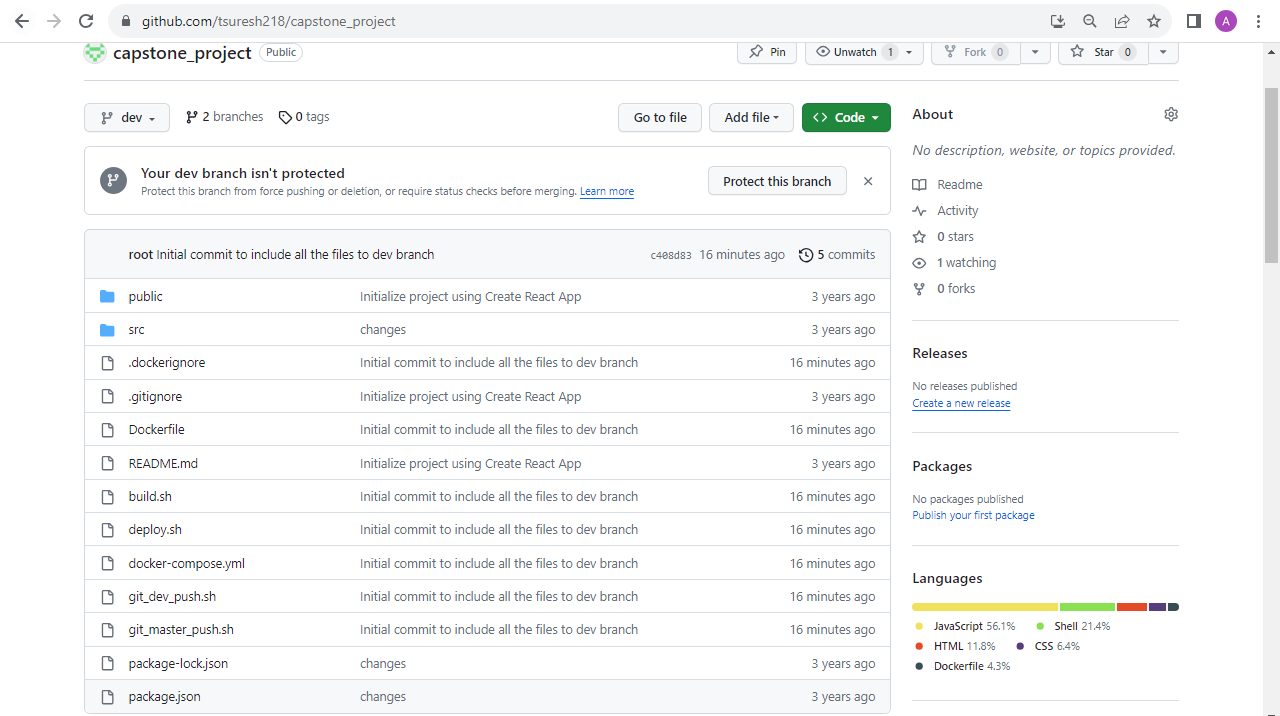
fi

**git\_master\_push.sh output**



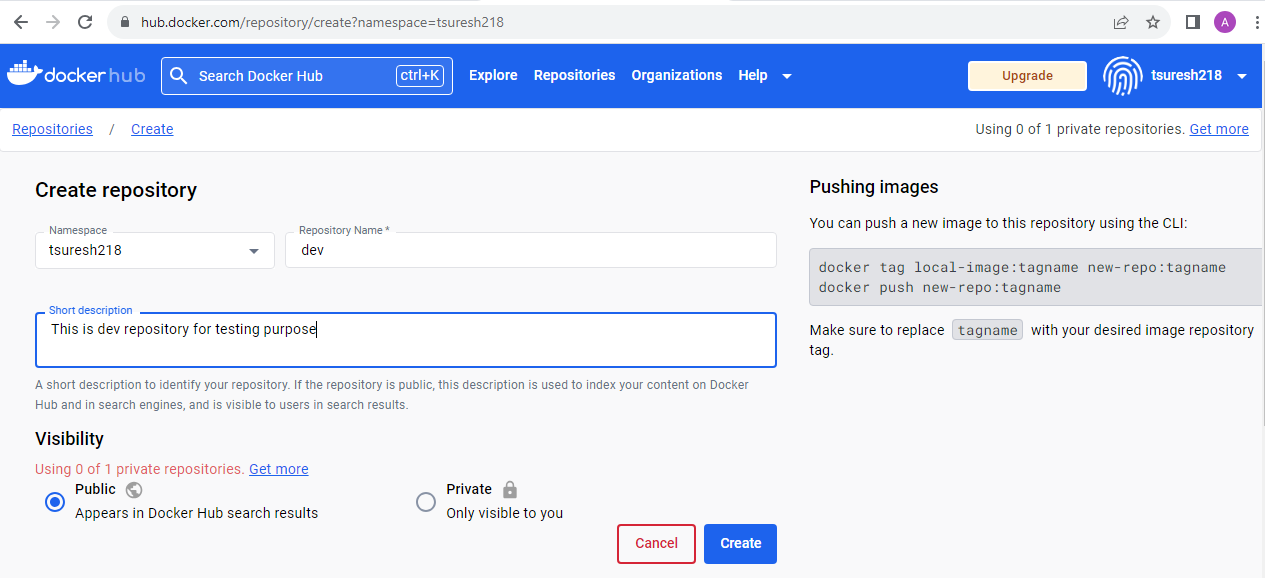
* Two branches have been created successfully in Github GUI as shown below

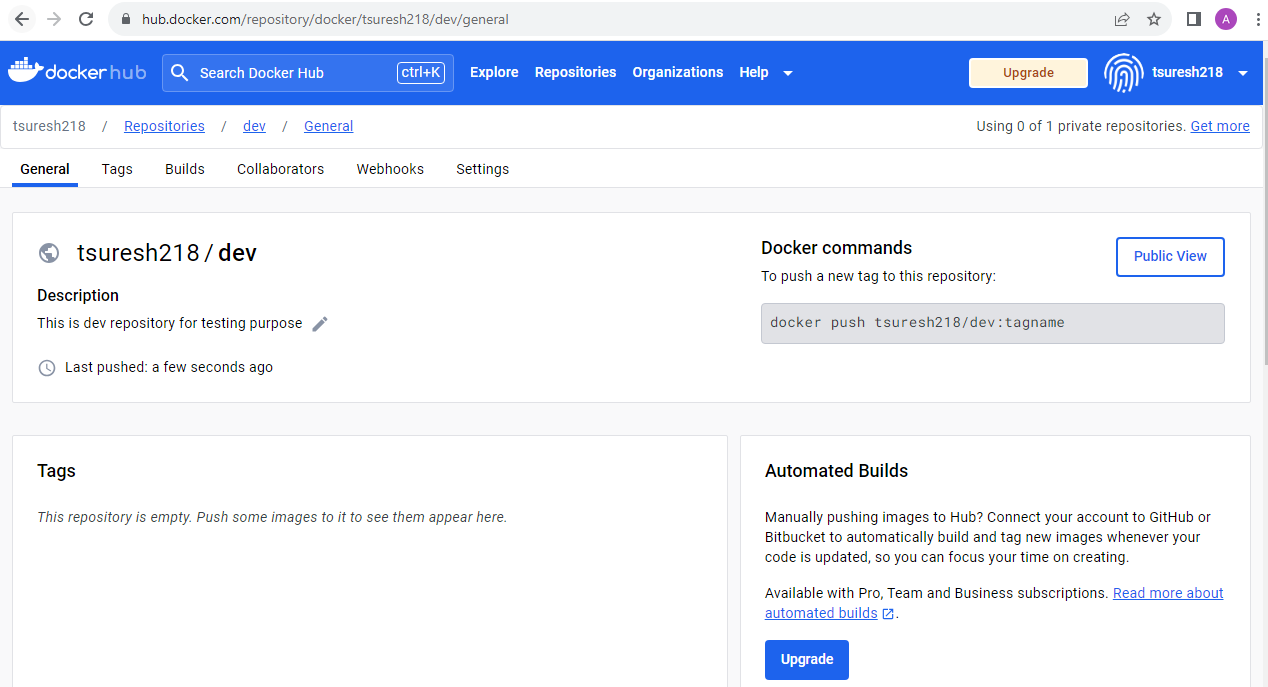


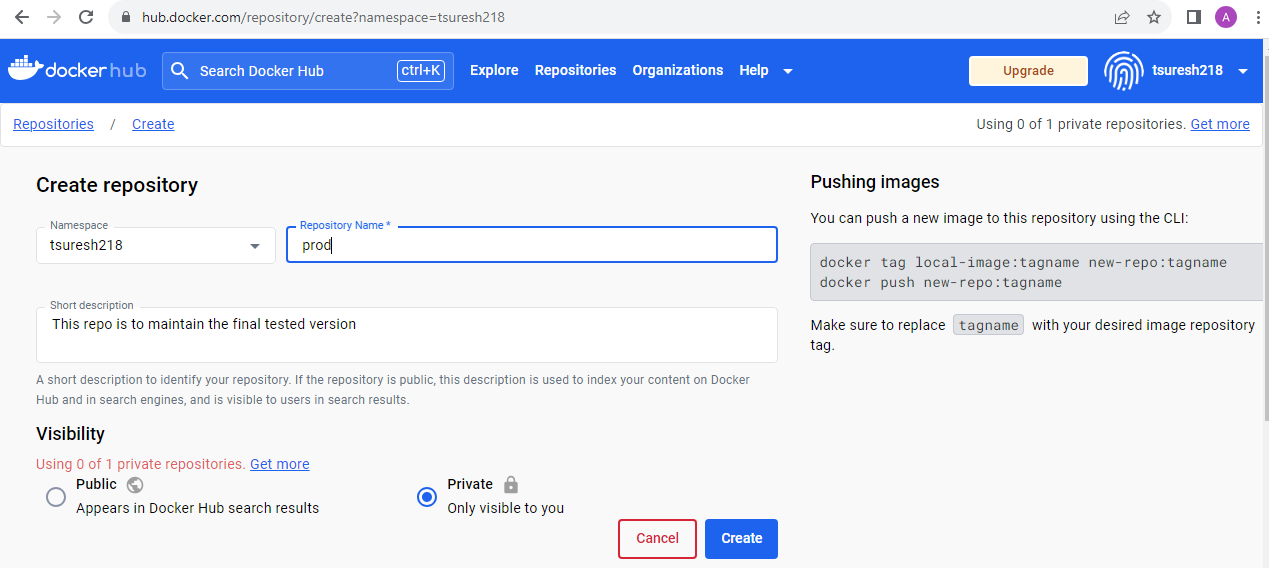


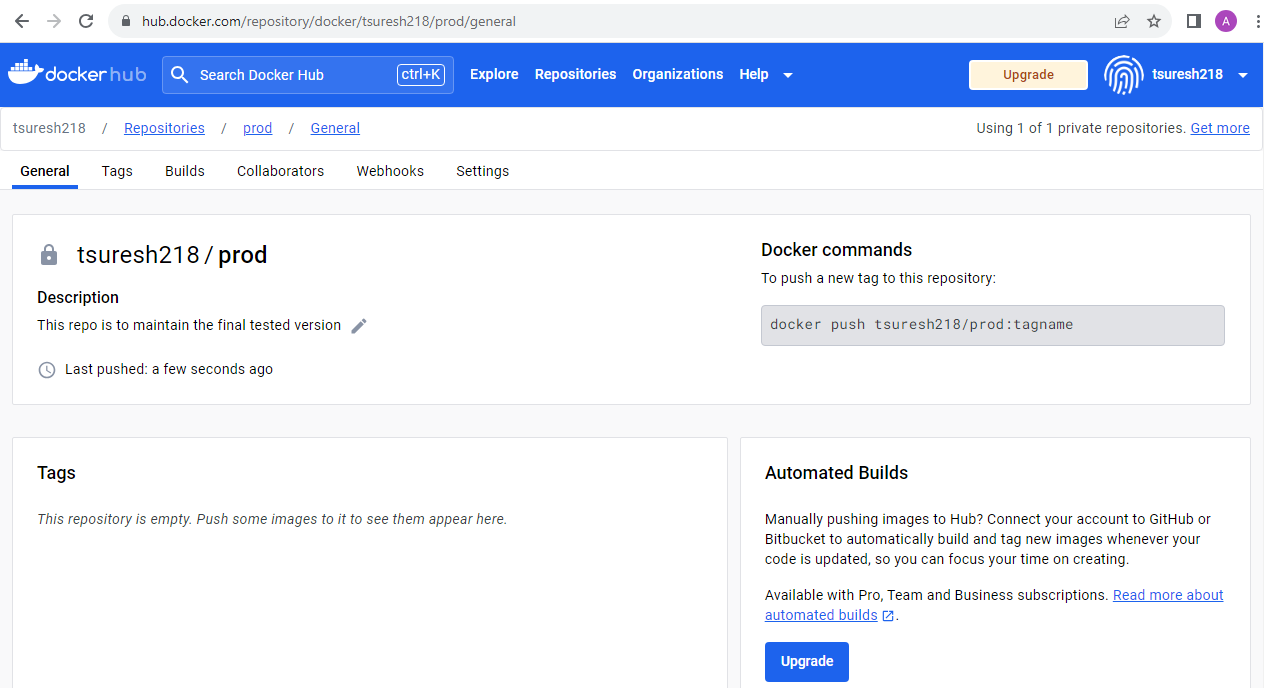
**Docker hub**

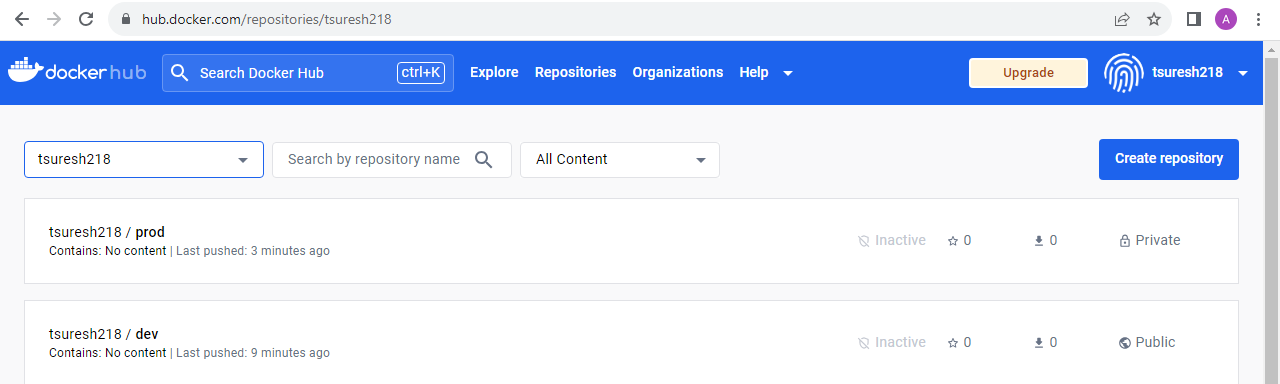
* Created two repos “dev” and “prod” and dev repo made it has public and prod made it as private as shown below











**Jenkins**

**Installation of Jenkins**

* install\_jenkins.sh script helps to install the Jenkins as shown below

#!/bin/bash

# Define the variable to store the process name

jenkins\_process="jenkins"

#Download jdk:

# Check for JDK version

java\_version=$(java -version 2>&1 >/dev/null | grep 'openjdk version' | awk '{print $3}' | tr -d \" | cut -d'.' -f1-2)

if [[ $java\_version == "11.0" ]]; then

    echo " JDK 11 is already installed"

else

    echo " JDK 11 is not installed. Installing now..."

fi

# update the package list

sudo apt update -y

# Install the JDK

sudo apt install openjdk-11-jdk -y

if java -version 2>&1 >/dev/null | grep -q "openjdk version \"11."; then

    echo "JDK 11 is installed successfully"

else

    echo " JDK 11 installation failed"

fi

# Download the repo and import the required key:

wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -

sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt-get update

sudo apt-get install jenkins -y

#Enable Jenkins:

sudo service jenkins enable

#Start Jenkins:

sudo service jenkins start

if sudo service jenkins status

   then

   echo "Jenkins is running"

else

   echo "Jenkins is not running, starting the service"

   sudo service jenkins start

fi

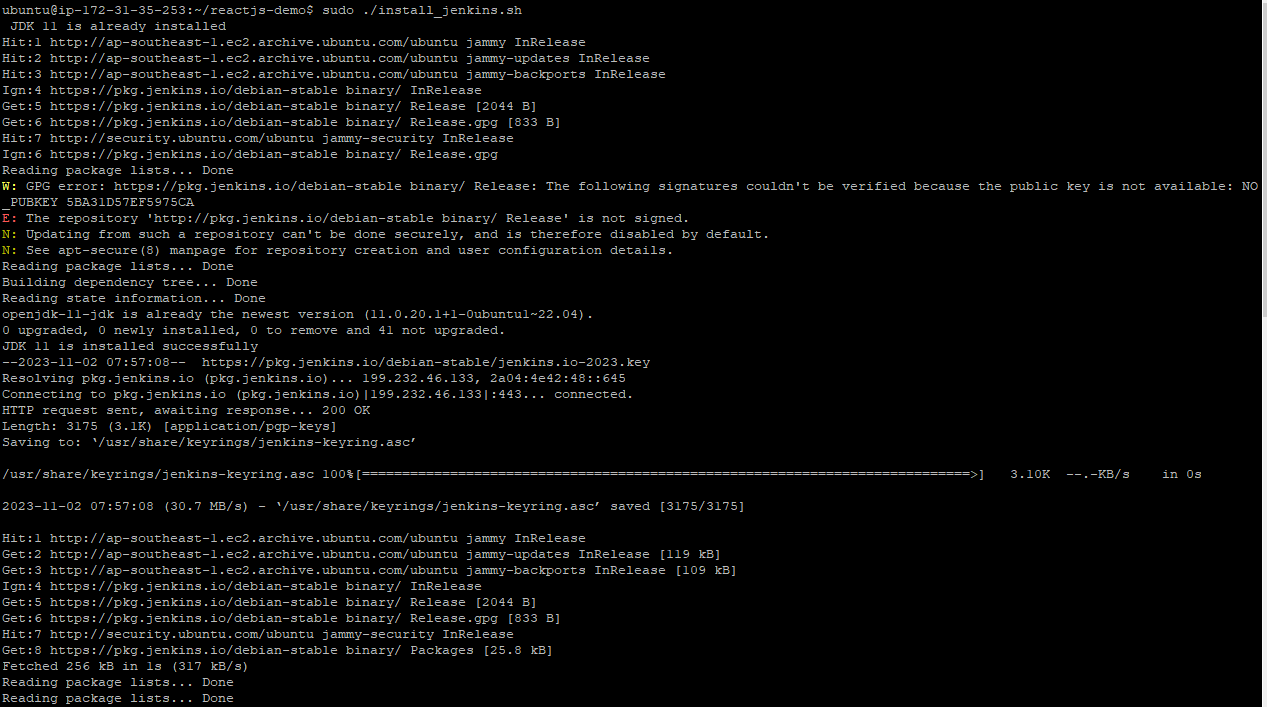
   if [ $? -eq 0 ]; then

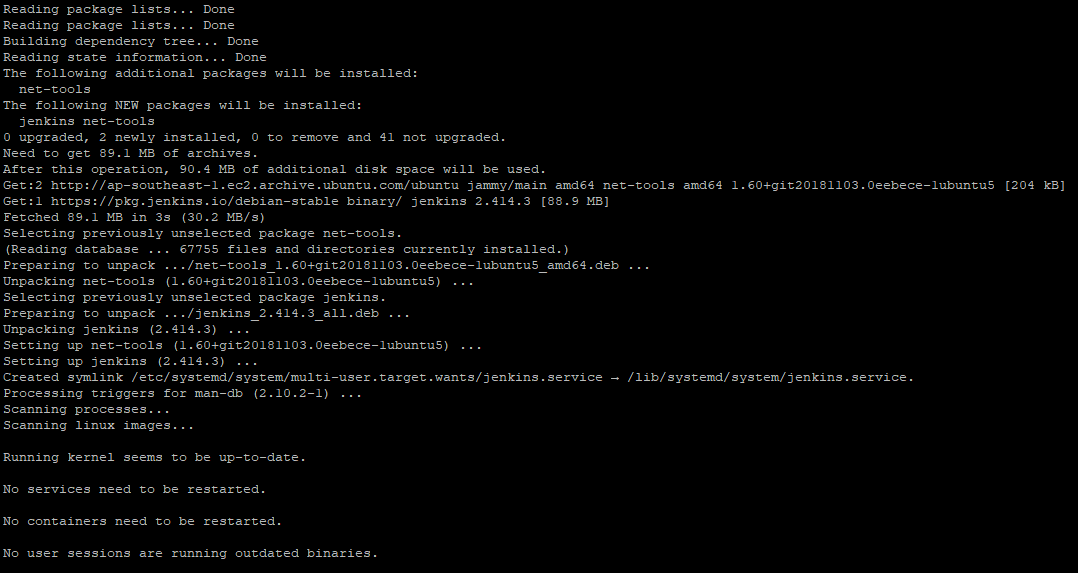
        echo "Jenkins started successfully"

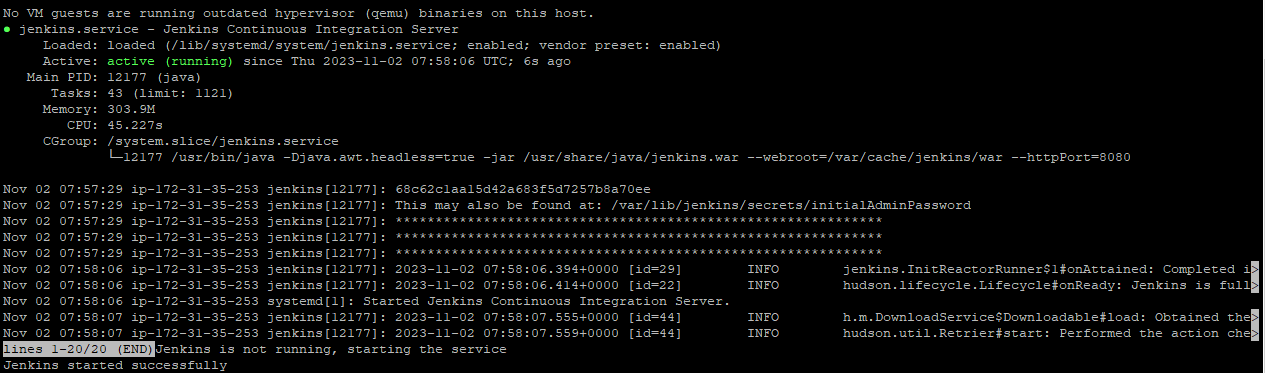
    else

        echo "Failed to start Jenkins"

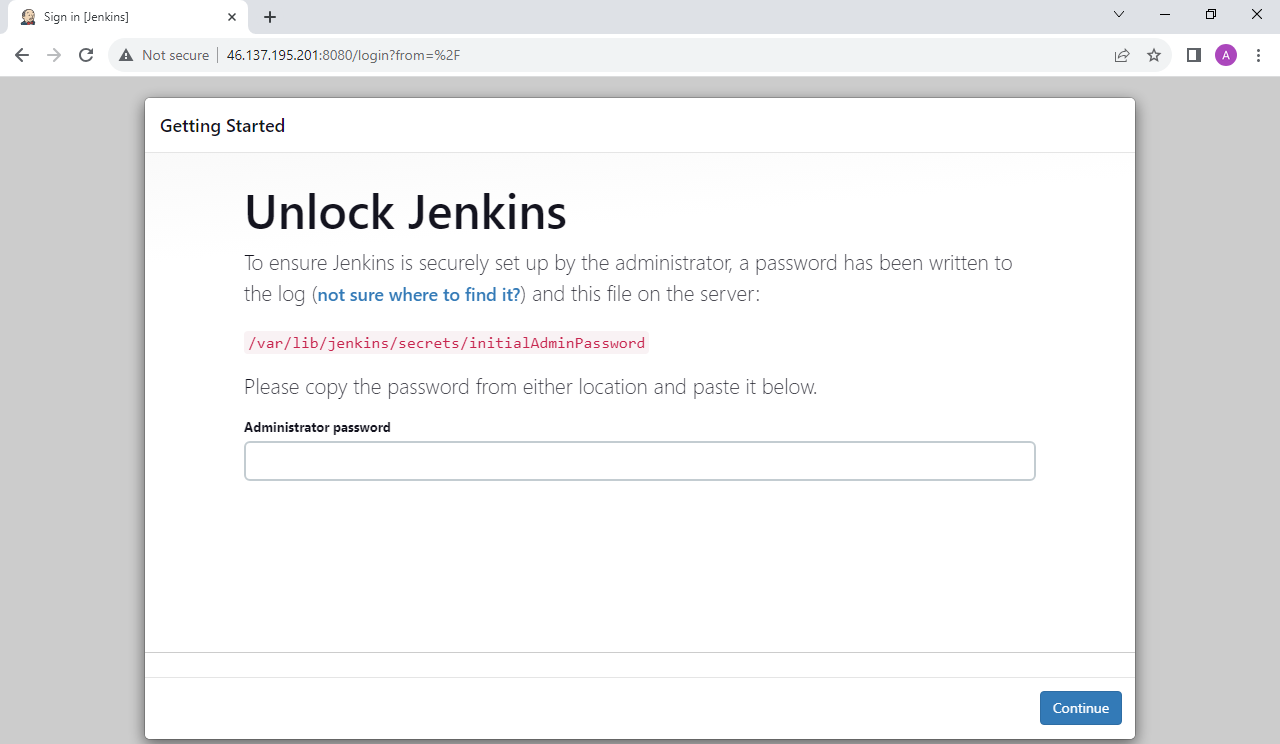
    fi

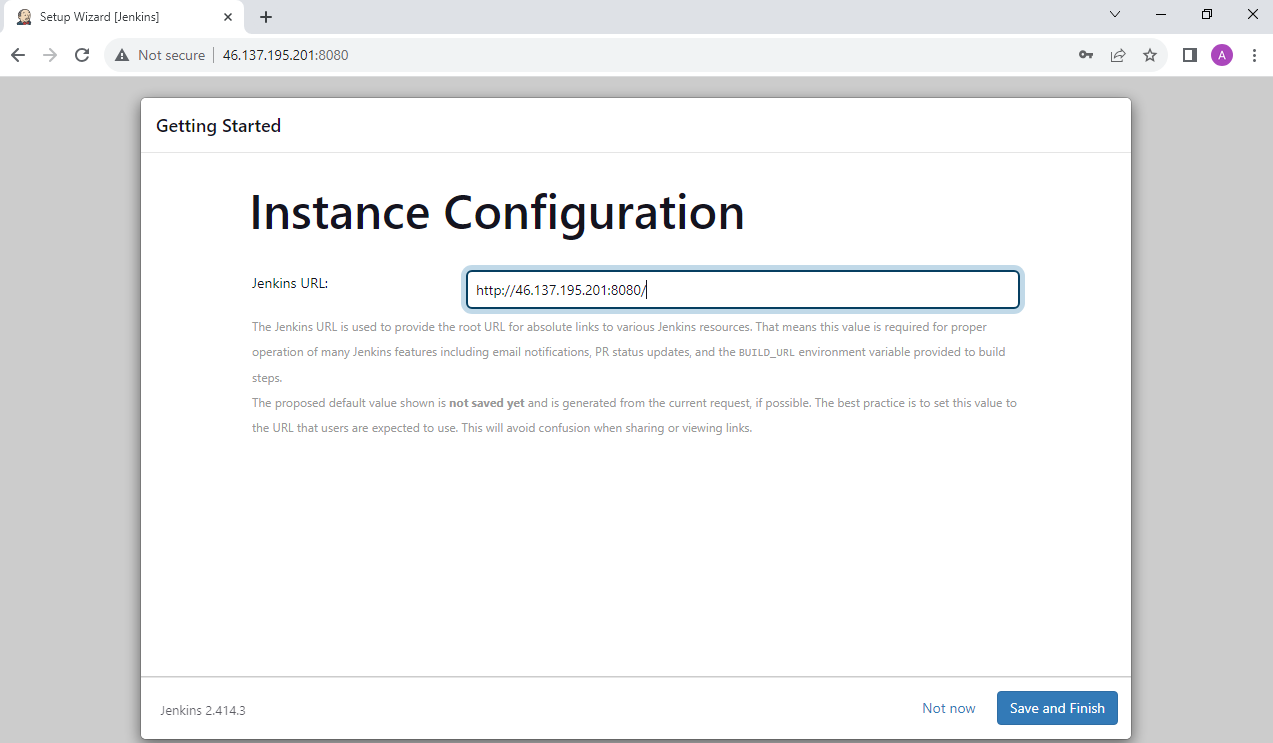


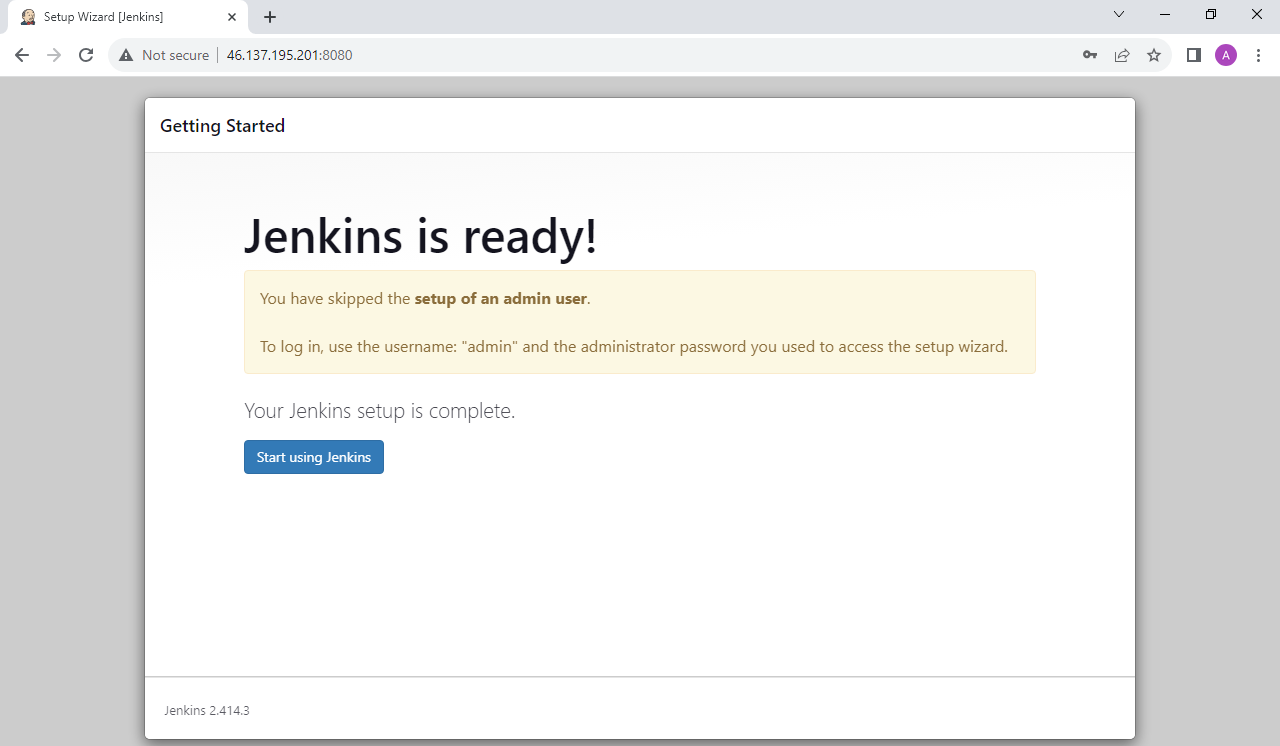


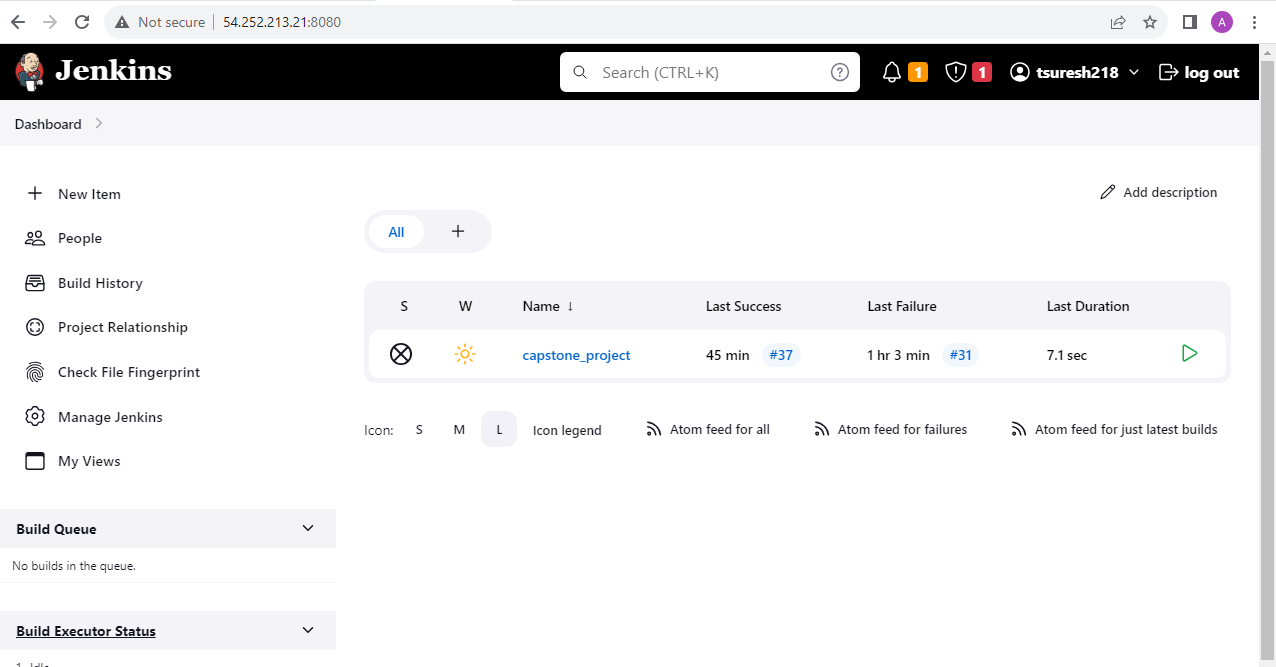


* Below screenshot shows the initial setup of Jenkins application



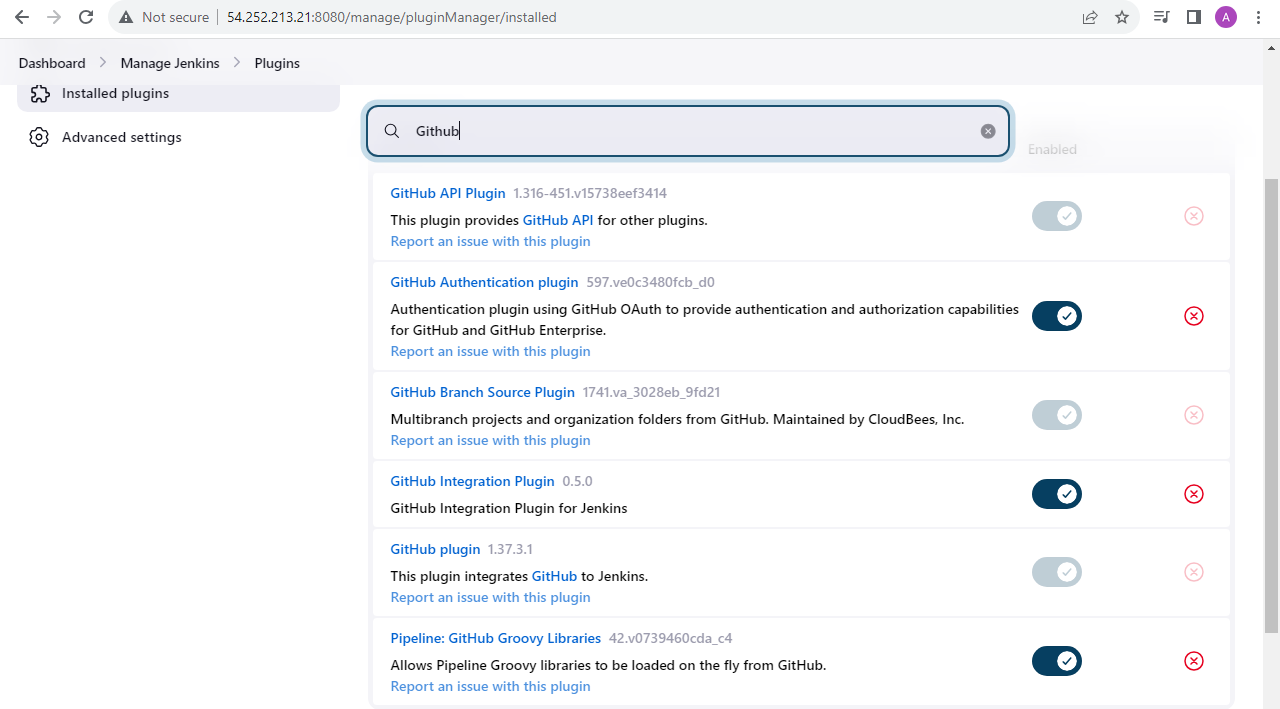






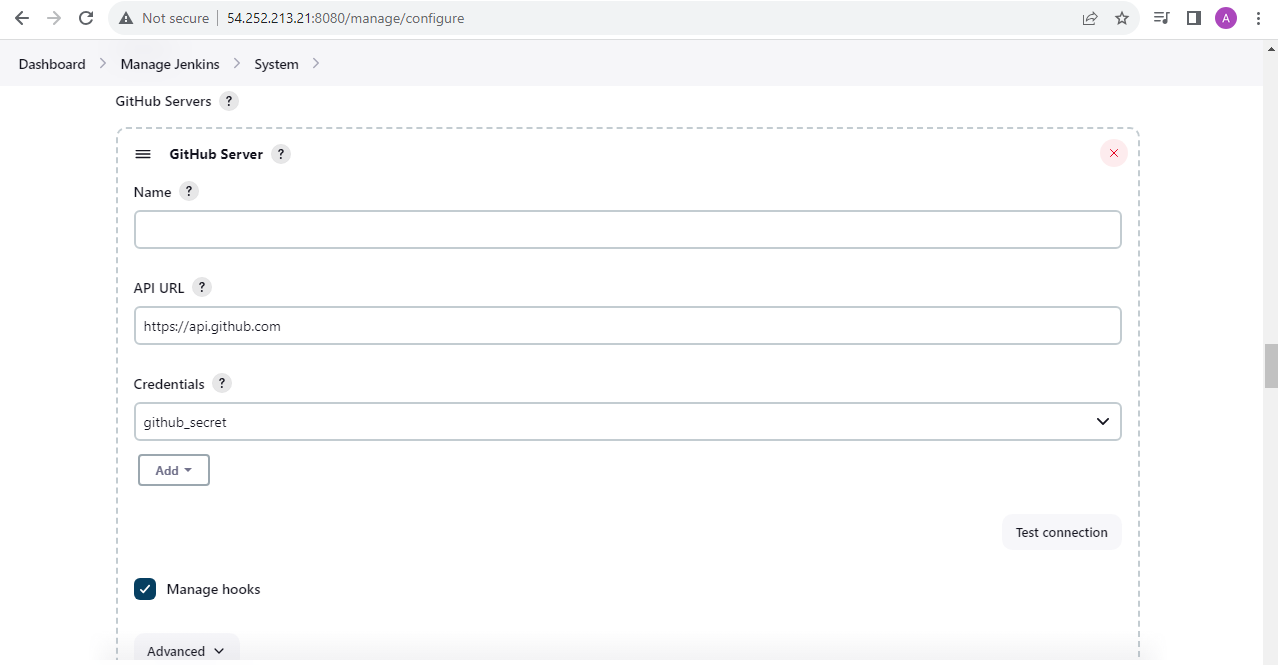
**Configure Jenkins in order to build, push & deploy the application:**

* Install the necessary plugins to connect github & dockerhub (i.e. Github plugin, Docker & SSH plugins)

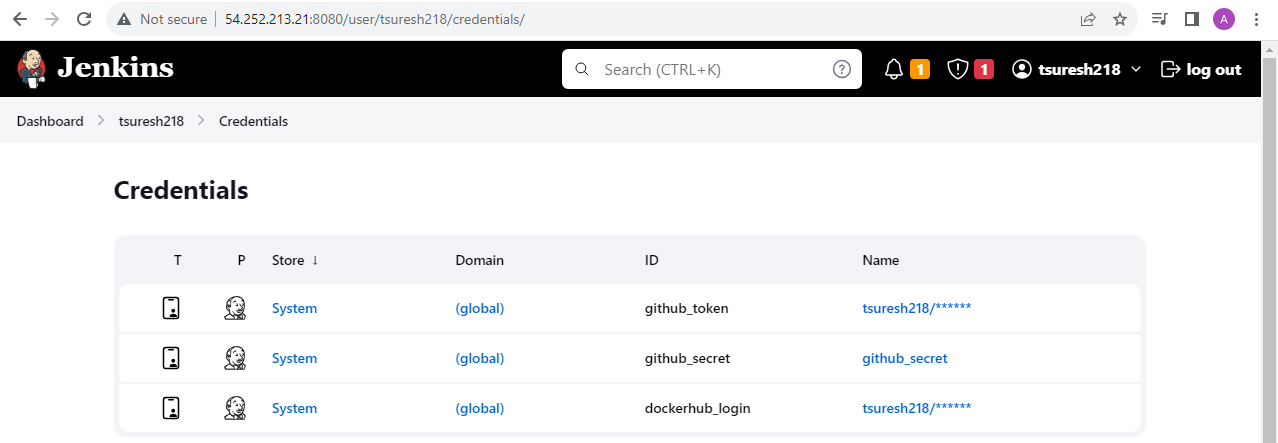




* Setup a github server under Dashboard 🡪 Manage Jenkins 🡪 System



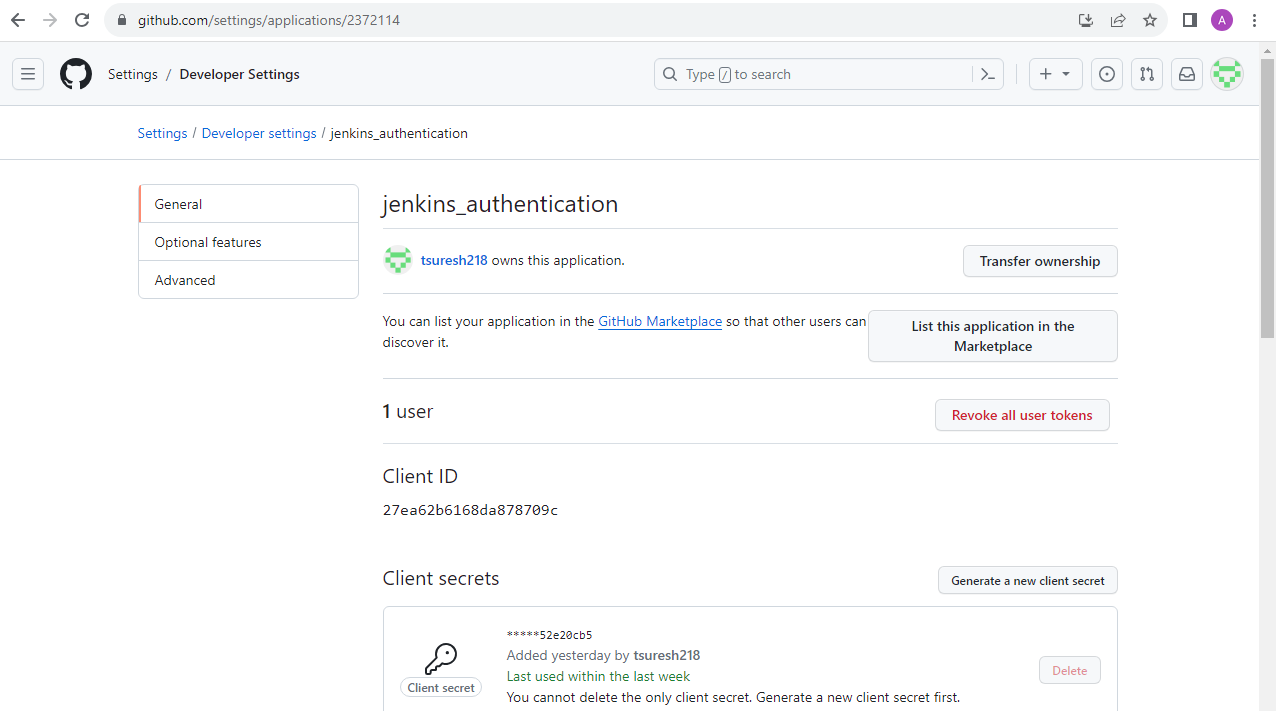
* Setup the credentials for github & Docker hub under credentials section

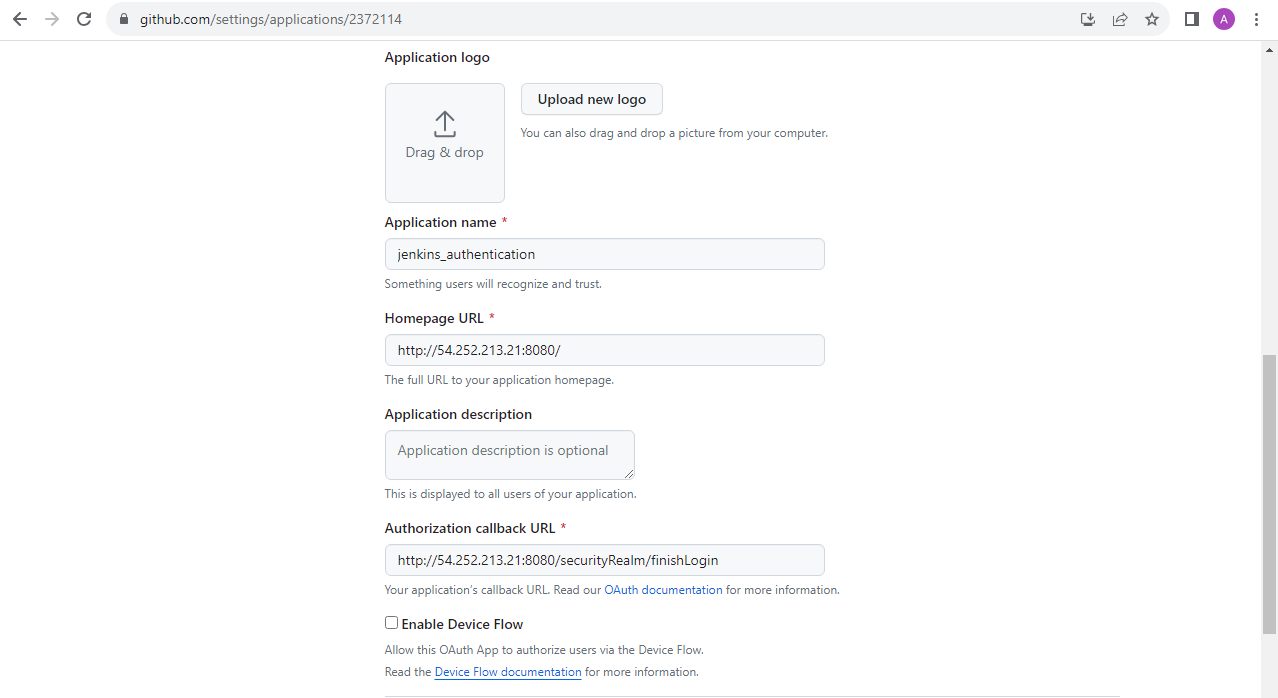


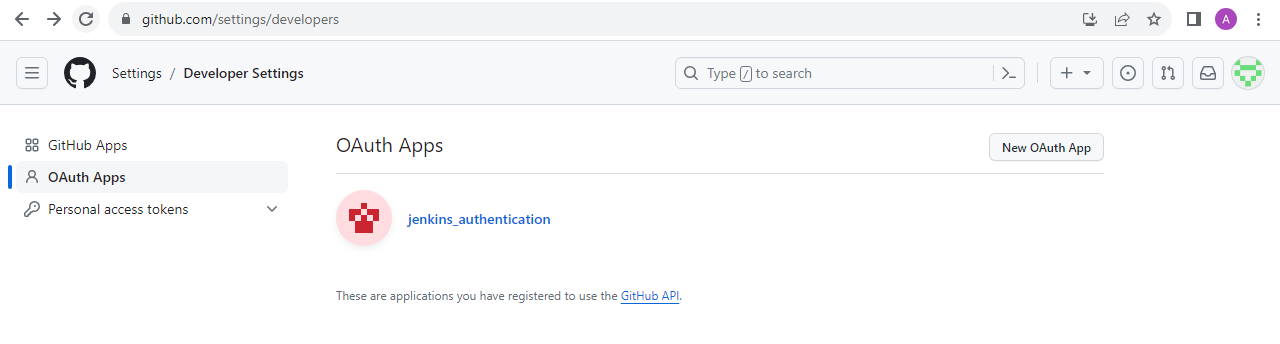
* Setup OAuth from github which is an open-source protocol for authorization that allows applications to gain access to HTTP services. In the context of Jenkins and github, OAuth is typically used as a way to authenticate Jenkins so that it can access a Github project every time a new commit is pushed to the repository

**Step 1: Create a new OAuth application in Github**

1. Under Settings 🡪 Developer Settings click on OAuth Apps
2. Click New OAuth App
3. Fill the form and click Register application
4. After registering OAuth application, we will be able to get the ‘Client ID’ and ‘Client Secret’

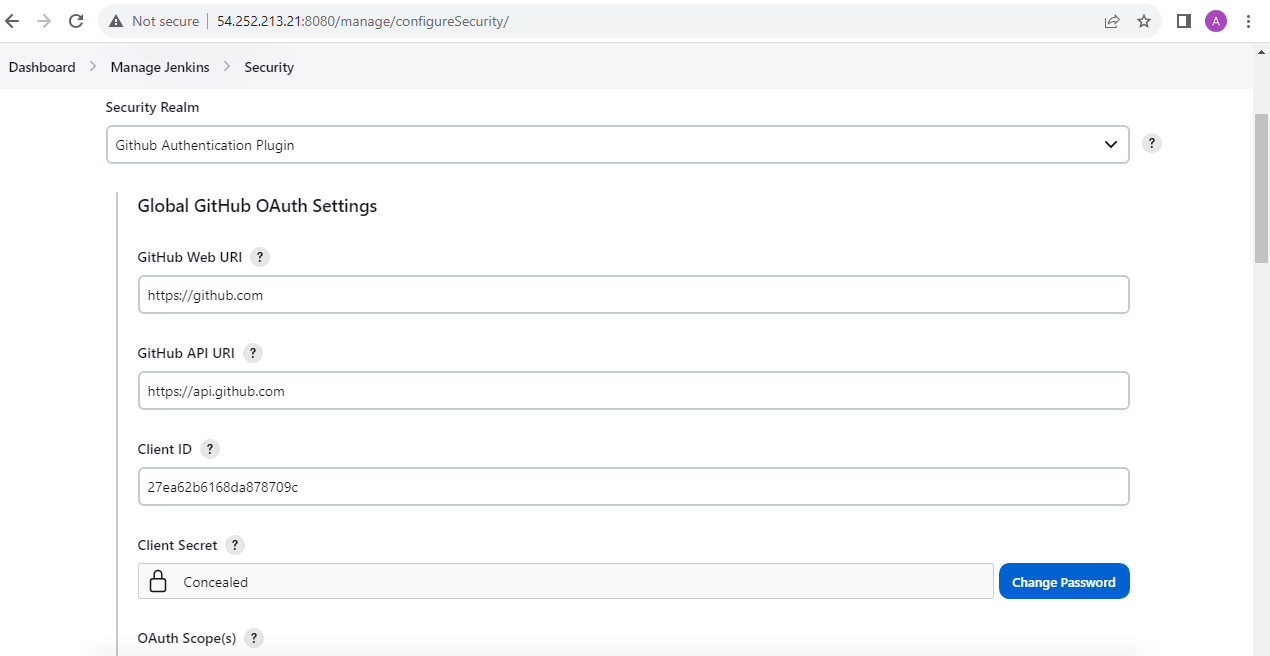


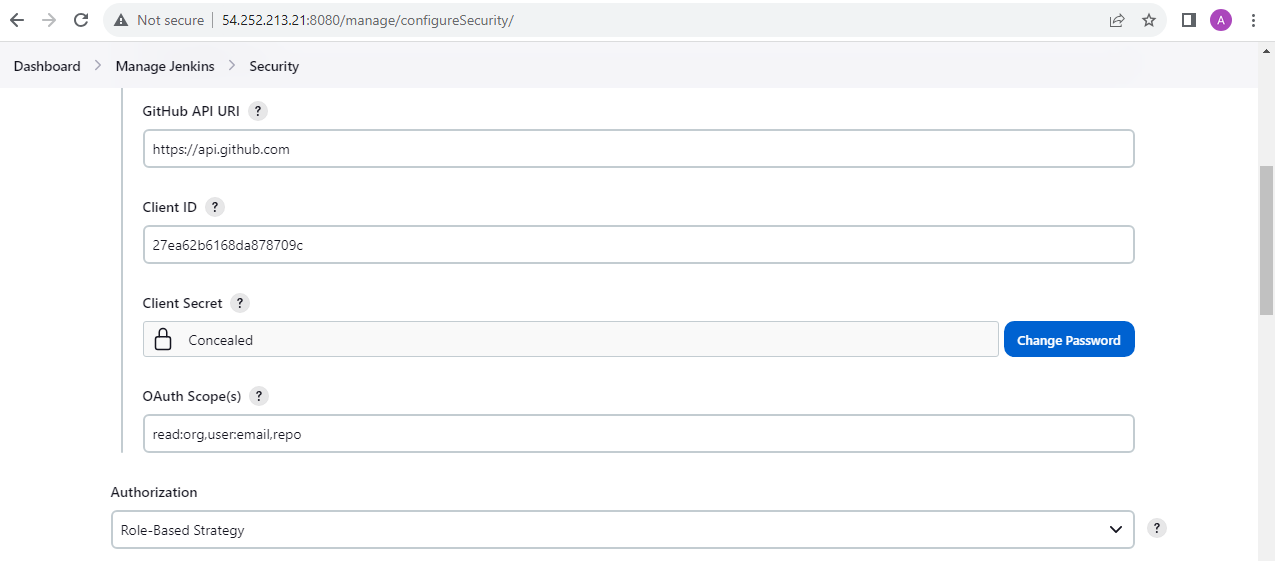




**Step 2: Configure Github OAuth in Jenkins**

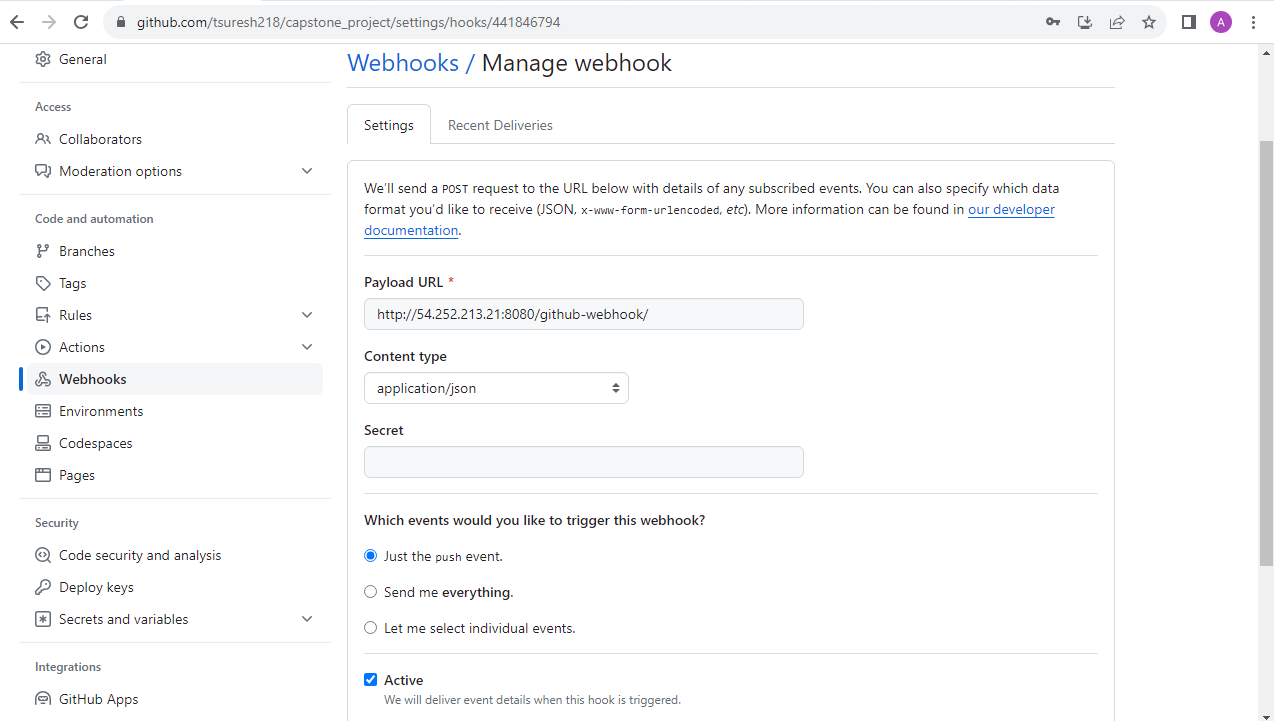
1. In Jenkins, navigate to Manage Jenkins 🡪 Configure Global Security
2. Under Access Control 🡪 Authorization, select Github Authentication Plugin and fill in the “Github Web URI”, “Client ID” and “Client Secret” fields





This will allow Jenkins to authenticate with Github using OAuth, when a build triggered, it will be authenticated and able to clone the repo and carry out various operations

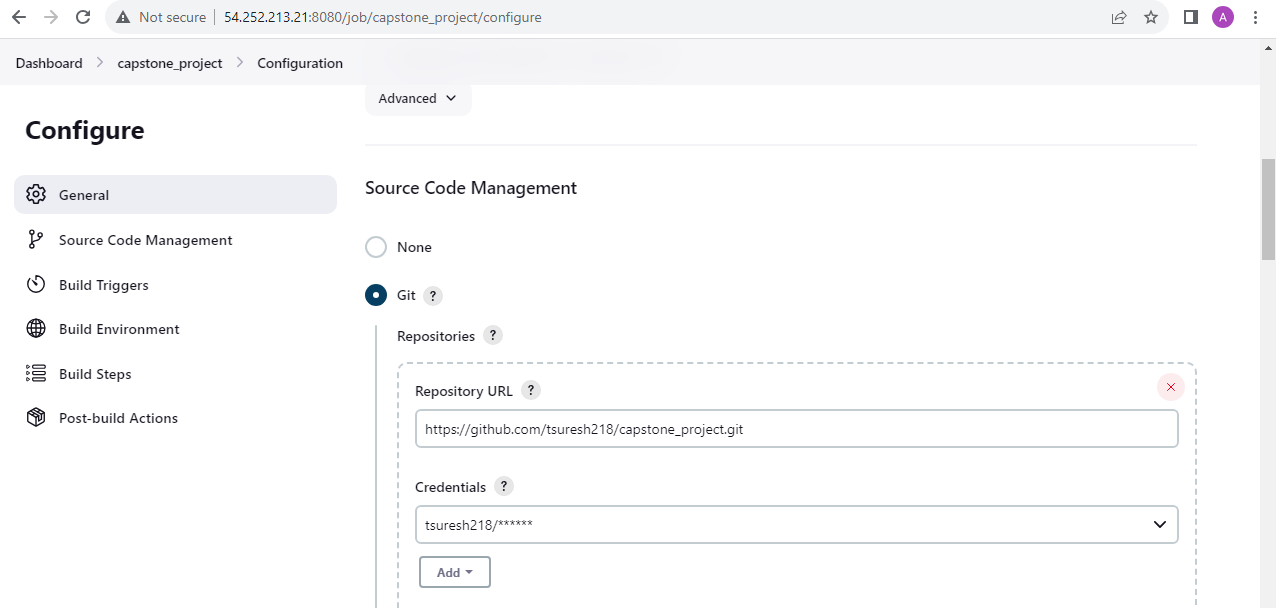
**Step 3: Add necessary webhooks in Github repository to trigger the Jenkins job**

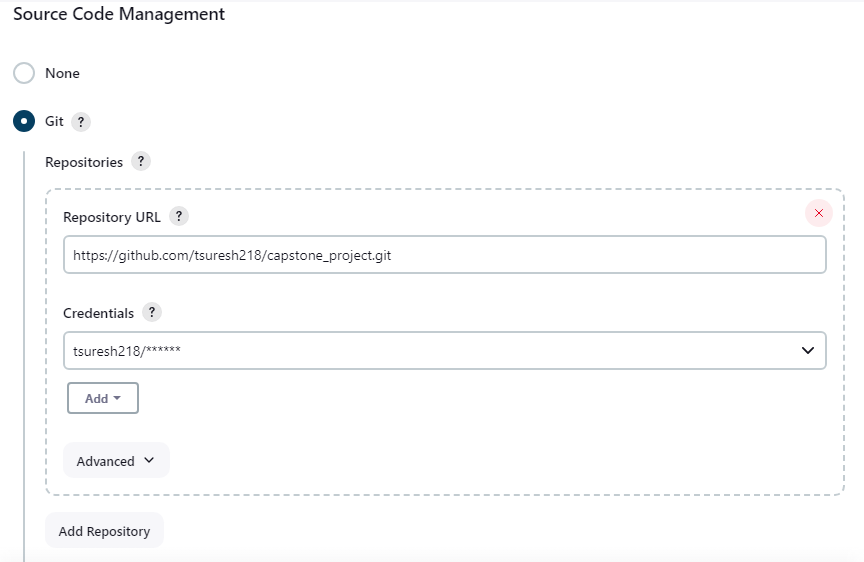




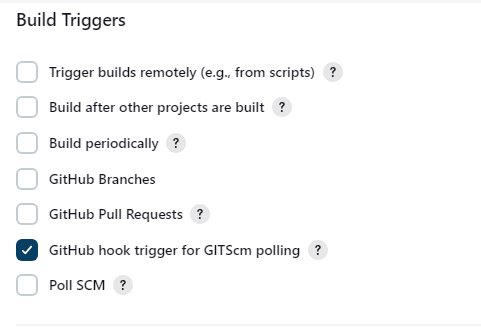
**Step 4: Create a Jenkins job to build and push the docker image based on the github branch commit**

**Jenkins job configuration:**

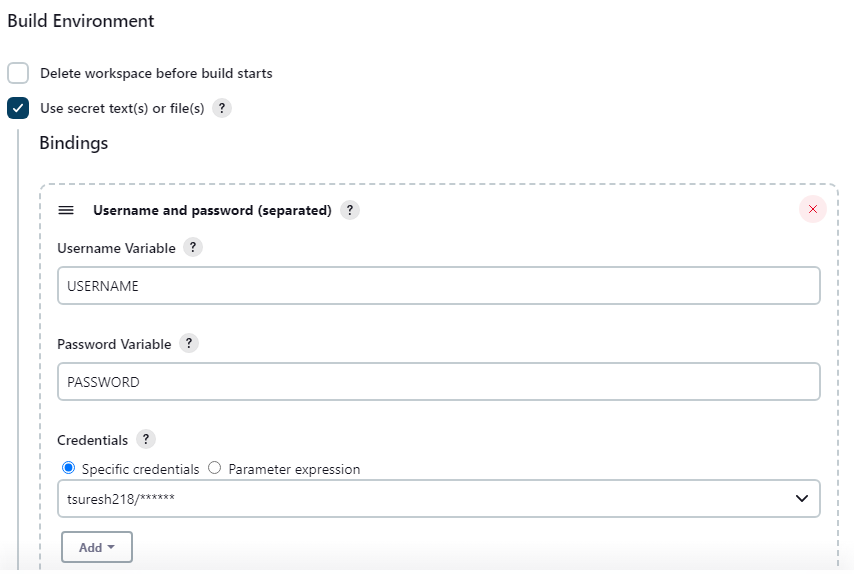


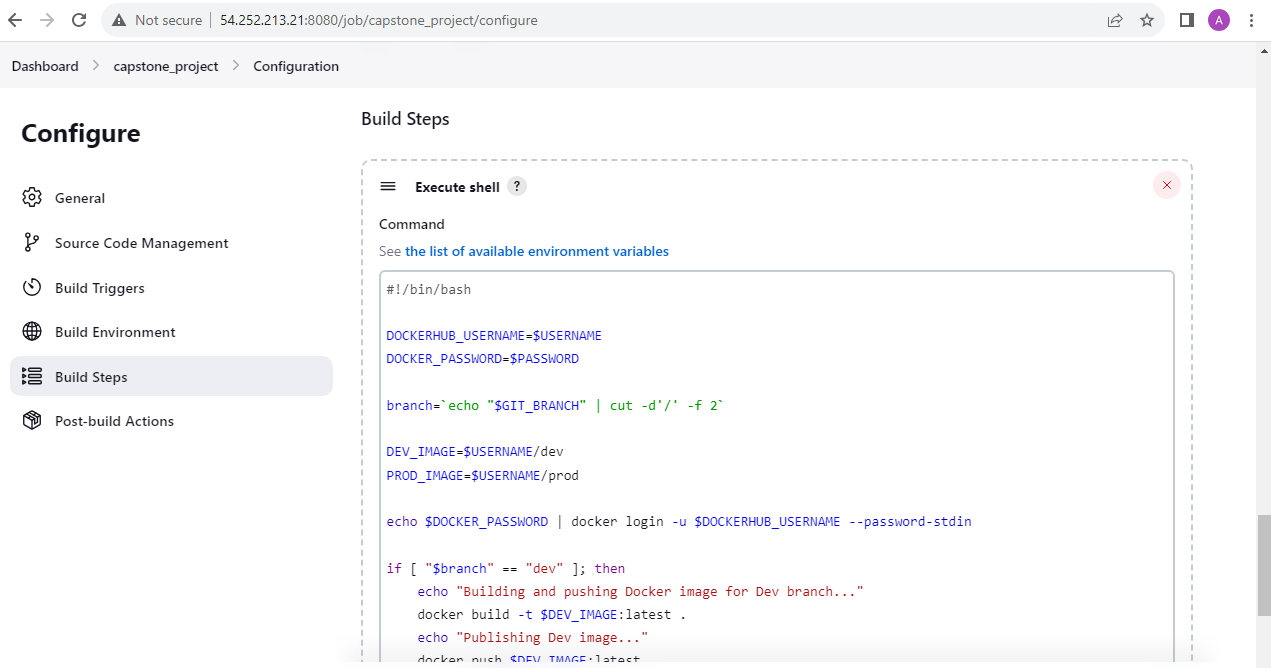






* Binding option helps to store the username & password variables:





**Execute Shell script (Jenkins\_script.sh) for the Jenkins job to update the git branches:**

#!/bin/bash

# Docker credentials

DOCKERHUB\_USERNAME=$USERNAME

DOCKER\_PASSWORD=$PASSWORD

# Jenkins sets the job's Git branch to the variable $GIT\_BRANCH

branch=`echo "$GIT\_BRANCH" | cut -d'/' -f 2`

# Docker image names

DEV\_IMAGE=$USERNAME/dev

PROD\_IMAGE=$USERNAME/prod

# Docker login

echo $DOCKER\_PASSWORD | docker login -u $DOCKERHUB\_USERNAME --password-stdin

if [ "$branch" == "dev" ]; then

echo "Building and pushing Docker image for Dev branch..."

docker build -t $DEV\_IMAGE:latest .

echo "Publishing Dev image..."

docker push $DEV\_IMAGE:latest

fi

if [ "$branch" == "master" ]; then

echo "Building and pushing Docker image for Prod branch..."

docker build -t $PROD\_IMAGE:latest .

docker push $PROD\_IMAGE:latest

fi

**Jenkins job Console Output for dev branch commit in github:**

* Below Jenkins job output shows when any commit made in dev github branch, jobs will be kicked automatically and docker image will be build and pushed to dev repo in Docker hub

Started by GitHub push by tsuresh218

Running as SYSTEM

Building in workspace /var/lib/jenkins/workspace/capstone\_project

The recommended git tool is: NONE

using credential github\_token

> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/capstone\_project/.git # timeout=10

Fetching changes from the remote Git repository

> git config remote.origin.url <https://github.com/tsuresh218/capstone_project.git> # timeout=10

Fetching upstream changes from <https://github.com/tsuresh218/capstone_project.git>

> git --version # timeout=10

> git --version # 'git version 2.34.1'

using GIT\_ASKPASS to set credentials

> git fetch --tags --force --progress -- <https://github.com/tsuresh218/capstone_project.git> +refs/heads/\*:refs/remotes/origin/\* # timeout=10

Seen branch in repository origin/dev

Seen branch in repository origin/master

Seen 2 remote branches

> git show-ref --tags -d # timeout=10

Checking out Revision 49cf1a3a6a6b598ae2c35ba48a05b34b16087939 (origin/dev)

> git config core.sparsecheckout # timeout=10

> git checkout -f 49cf1a3a6a6b598ae2c35ba48a05b34b16087939 # timeout=10

Commit message: "Delete Jenkinsfile"

> git rev-list --no-walk 682a0c4c614b50e7d96e2a460c5f9e49668ef3b0 # timeout=10

[capstone\_project] $ /bin/bash /tmp/jenkins11689417470409933926.sh

WARNING! Your password will be stored unencrypted in /var/lib/jenkins/.docker/config.json.

Configure a credential helper to remove this warning. See

<https://docs.docker.com/engine/reference/commandline/login/#credentials-store>

Login Succeeded

Building and pushing Docker image for Dev branch...

DEPRECATED: The legacy builder is deprecated and will be removed in a future release.

Install the buildx component to build images with BuildKit:

<https://docs.docker.com/go/buildx/>

Sending build context to Docker daemon 645.6kB

Step 1/7 : FROM node:14

---> 1d12470fa662

Step 2/7 : WORKDIR /app

---> Using cache

---> 908782b8c549

Step 3/7 : COPY package\*.json ./

---> Using cache

---> f23f05c072d2

Step 4/7 : COPY . .

---> 51ee02c8635a

Step 5/7 : RUN npm install && npm install -g serve && npm run build && rm -fr node\_modules

---> Running in 956c9393b932

> core-js@2.6.11 postinstall /app/node\_modules/babel-runtime/node\_modules/core-js

> node -e "try{require('./postinstall')}catch(e){}"

[96mThank you for using core-js ([94m <https://github.com/zloirock/core-js> [96m) for polyfilling JavaScript standard library![0m

[96mThe project needs your help! Please consider supporting of core-js on Open Collective or Patreon: [0m

[96m>[94m <https://opencollective.com/core-js> [0m

[96m>[94m <https://www.patreon.com/zloirock> [0m

[96mAlso, the author of core-js ([94m <https://github.com/zloirock> [96m) is looking for a good job -)[0m

> core-js@3.6.5 postinstall /app/node\_modules/core-js

> node -e "try{require('./postinstall')}catch(e){}"

[96mThank you for using core-js ([94m <https://github.com/zloirock/core-js> [96m) for polyfilling JavaScript standard library![0m

[96mThe project needs your help! Please consider supporting of core-js on Open Collective or Patreon: [0m

[96m>[94m <https://opencollective.com/core-js> [0m

[96m>[94m <https://www.patreon.com/zloirock> [0m

[96mAlso, the author of core-js ([94m <https://github.com/zloirock> [96m) is looking for a good job -)[0m

> core-js-pure@3.6.5 postinstall /app/node\_modules/core-js-pure

> node -e "try{require('./postinstall')}catch(e){}"

[96mThank you for using core-js ([94m <https://github.com/zloirock/core-js> [96m) for polyfilling JavaScript standard library![0m

[96mThe project needs your help! Please consider supporting of core-js on Open Collective or Patreon: [0m

[96m>[94m <https://opencollective.com/core-js> [0m

[96m>[94m <https://www.patreon.com/zloirock> [0m

[96mAlso, the author of core-js ([94m <https://github.com/zloirock> [96m) is looking for a good job -)[0m

[91mnpm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node\_modules/webpack-dev-server/node\_modules/fsevents):

npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"linux","arch":"x64"})

[0m[91mnpm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node\_modules/watchpack-chokidar2/node\_modules/fsevents):

npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"linux","arch":"x64"})

[0m[91mnpm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node\_modules/jest-haste-map/node\_modules/fsevents):

npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"linux","arch":"x64"})

[0m[91mnpm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.2 (node\_modules/fsevents):

npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"linux","arch":"x64"})

[0m[91m

[0madded 1655 packages from 812 contributors and audited 1659 packages in 34.017s

64 packages are looking for funding

run `npm fund` for details

found 407 vulnerabilities (3 low, 200 moderate, 163 high, 41 critical)

run `npm audit fix` to fix them, or `npm audit` for details

/usr/local/bin/serve -> /usr/local/lib/node\_modules/serve/build/main.js

+ serve@14.2.1

added 91 packages from 43 contributors in 5.525s

> react-front-end@0.1.0 build /app

> react-scripts build

Creating an optimized production build...

[91mBrowserslist: caniuse-lite is outdated. Please run:

npx browserslist@latest --update-db

[0mCompiled successfully.

File sizes after gzip:

40.46 KB build/static/js/2.c71cdbdb.chunk.js

22.47 KB build/static/css/2.af3c1da9.chunk.css

793 B build/static/js/main.d9af885d.chunk.js

779 B build/static/js/runtime-main.c2a4c5c8.js

547 B build/static/css/main.5f361e03.chunk.css

The project was built assuming it is hosted at /.

You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.

You may serve it with a static server:

serve -s build

Find out more about deployment here:

bit.ly/CRA-deploy

Removing intermediate container 956c9393b932

---> d4f4cb693383

Step 6/7 : EXPOSE 3000

---> Running in fccdd097c394

Removing intermediate container fccdd097c394

---> 3ed05e8d0aa2

Step 7/7 : CMD [ "serve", "-s", "build" ]

---> Running in a200e325bffc

Removing intermediate container a200e325bffc

---> 03d59ef70d21

Successfully built 03d59ef70d21

Successfully tagged tsuresh218/dev:latest

Publishing Dev image...

The push refers to repository [docker.io/tsuresh218/dev]

df6b3044b6fd: Preparing

7f00c391760a: Preparing

48664ac10d74: Preparing

d456ad64e1a7: Preparing

0d5f5a015e5d: Preparing

3c777d951de2: Preparing

f8a91dd5fc84: Preparing

cb81227abde5: Preparing

e01a454893a9: Preparing

c45660adde37: Preparing

fe0fb3ab4a0f: Preparing

f1186e5061f2: Preparing

b2dba7477754: Preparing

cb81227abde5: Waiting

e01a454893a9: Waiting

c45660adde37: Waiting

fe0fb3ab4a0f: Waiting

3c777d951de2: Waiting

f1186e5061f2: Waiting

b2dba7477754: Waiting

f8a91dd5fc84: Waiting

0d5f5a015e5d: Layer already exists

48664ac10d74: Layer already exists

d456ad64e1a7: Layer already exists

3c777d951de2: Layer already exists

f8a91dd5fc84: Layer already exists

cb81227abde5: Layer already exists

e01a454893a9: Layer already exists

c45660adde37: Layer already exists

fe0fb3ab4a0f: Layer already exists

f1186e5061f2: Layer already exists

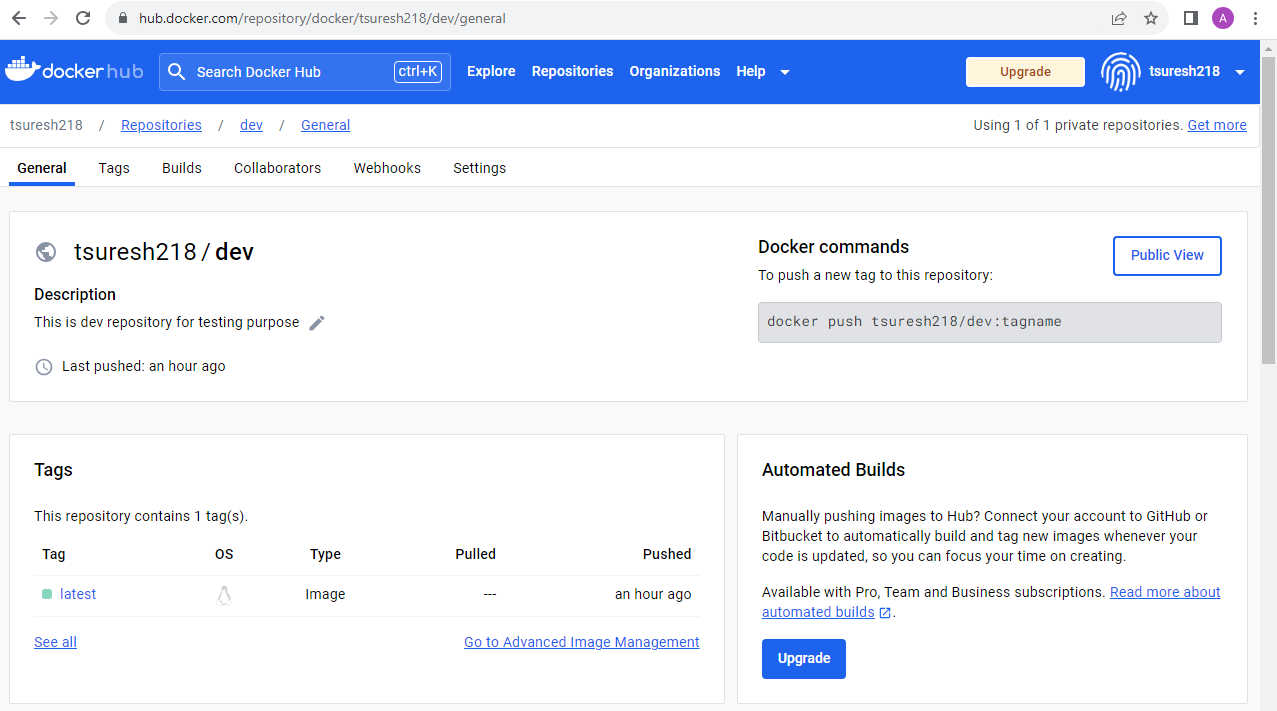
b2dba7477754: Layer already exists

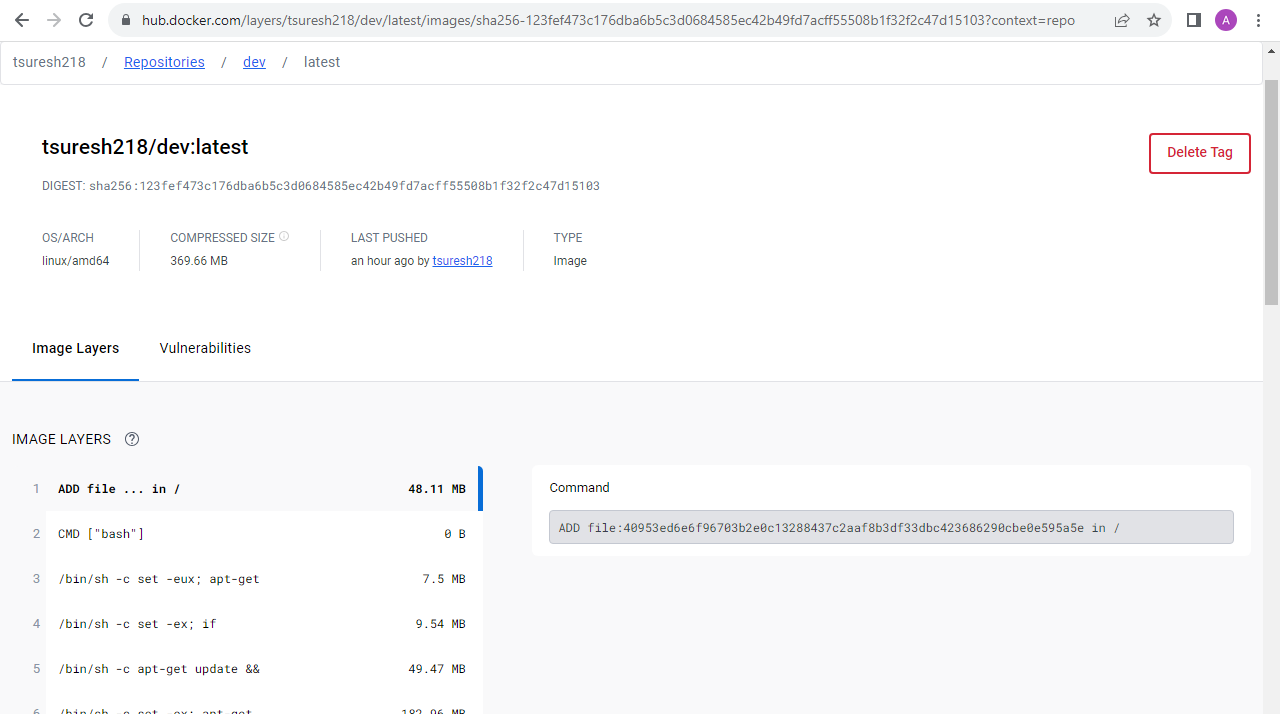
7f00c391760a: Pushed

df6b3044b6fd: Pushed

latest: digest: sha256:123fef473c176dba6b5c3d0684585ec42b49fd7acff55508b1f32f2c47d15103 size: 3053

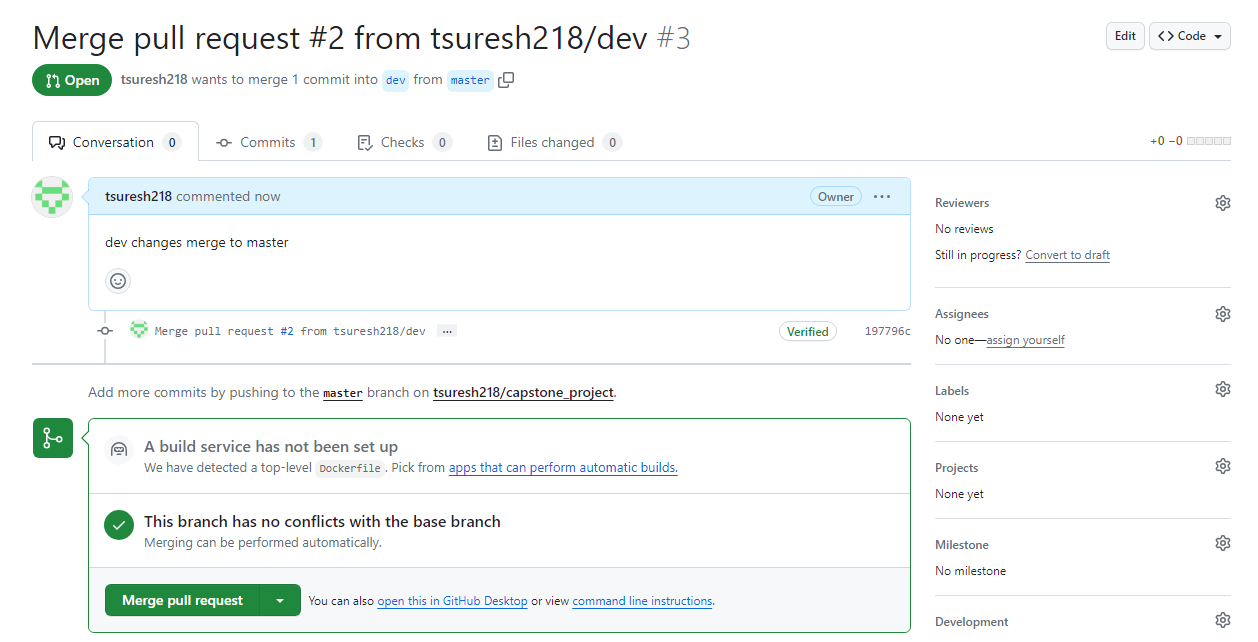
Finished: SUCCESS



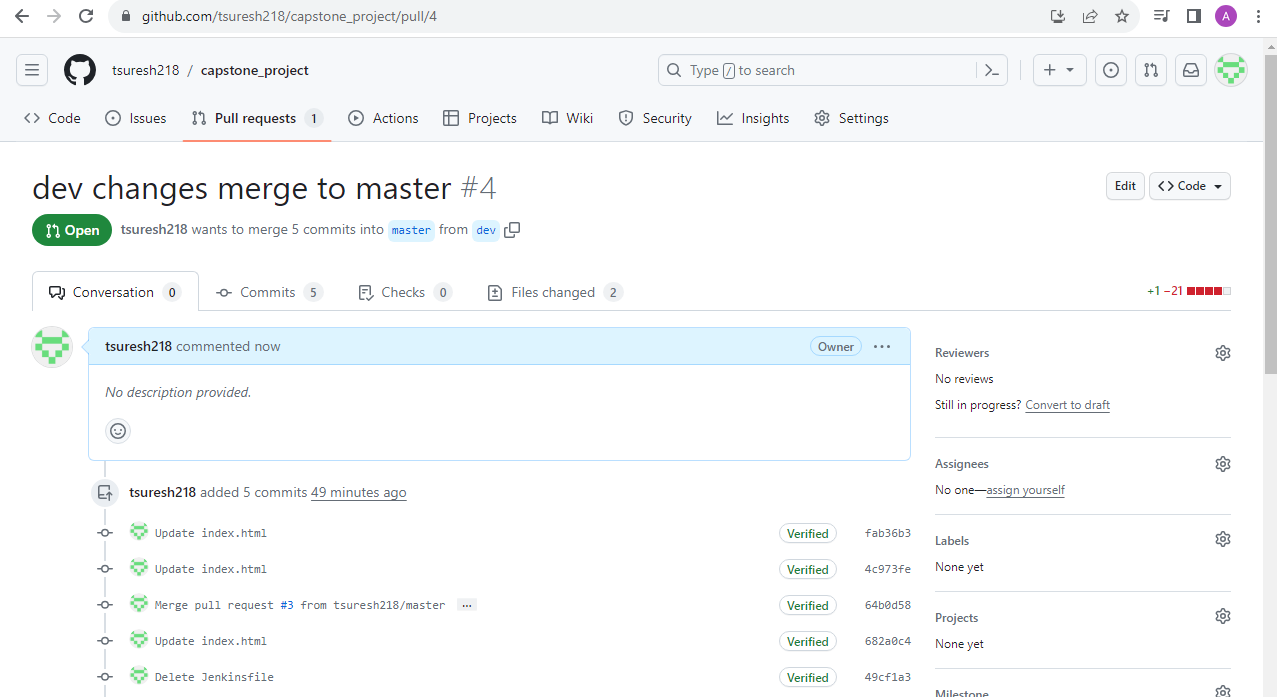


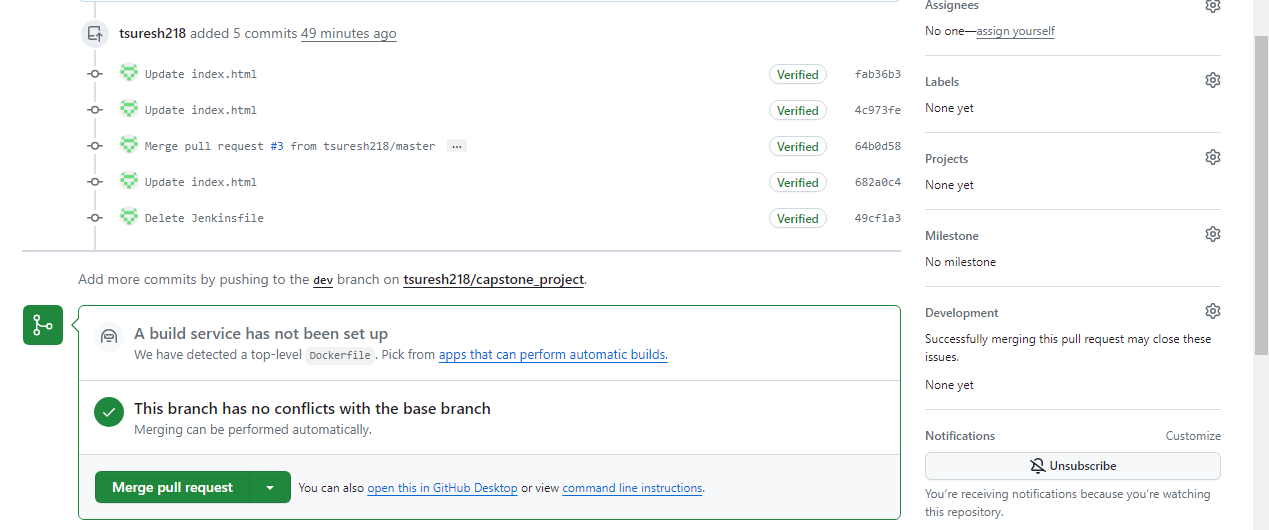
**Jenkins job Console Output for dev branch changes to merge to master in github:**

* Once dev is merged to master branch in github automatically build is started in Jenkins and docker image will be build and pushed to prod branch in Docker hub









Started by GitHub push by tsuresh218

Running as SYSTEM

Building in workspace /var/lib/jenkins/workspace/capstone\_project

The recommended git tool is: NONE

using credential github\_token

> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/capstone\_project/.git # timeout=10

Fetching changes from the remote Git repository

> git config remote.origin.url <https://github.com/tsuresh218/capstone_project.git> # timeout=10

Fetching upstream changes from <https://github.com/tsuresh218/capstone_project.git>

> git --version # timeout=10

> git --version # 'git version 2.34.1'

using GIT\_ASKPASS to set credentials

> git fetch --tags --force --progress -- <https://github.com/tsuresh218/capstone_project.git> +refs/heads/\*:refs/remotes/origin/\* # timeout=10

Seen branch in repository origin/dev

Seen branch in repository origin/master

Seen 2 remote branches

> git show-ref --tags -d # timeout=10

Checking out Revision a3165262f181f1f715d69a67b440dd9cd15c0e9b (origin/master)

> git config core.sparsecheckout # timeout=10

> git checkout -f a3165262f181f1f715d69a67b440dd9cd15c0e9b # timeout=10

Commit message: "Merge pull request #4 from tsuresh218/dev"

> git rev-list --no-walk 197796c053a60b7ce76dcd3fd90cccb1938462e4 # timeout=10

[capstone\_project] $ /bin/bash /tmp/jenkins9593541569654746633.sh

WARNING! Your password will be stored unencrypted in /var/lib/jenkins/.docker/config.json.

Configure a credential helper to remove this warning. See

<https://docs.docker.com/engine/reference/commandline/login/#credentials-store>

Login Succeeded

Building and pushing Docker image for Prod branch...

DEPRECATED: The legacy builder is deprecated and will be removed in a future release.

Install the buildx component to build images with BuildKit:

<https://docs.docker.com/go/buildx/>

Sending build context to Docker daemon 645.6kB

Step 1/7 : FROM node:14

---> 1d12470fa662

Step 2/7 : WORKDIR /app

---> Using cache

---> 908782b8c549

Step 3/7 : COPY package\*.json ./

---> Using cache

---> f23f05c072d2

Step 4/7 : COPY . .

---> Using cache

---> 51ee02c8635a

Step 5/7 : RUN npm install && npm install -g serve && npm run build && rm -fr node\_modules

---> Using cache

---> d4f4cb693383

Step 6/7 : EXPOSE 3000

---> Using cache

---> 3ed05e8d0aa2

Step 7/7 : CMD [ "serve", "-s", "build" ]

---> Using cache

---> 03d59ef70d21

Successfully built 03d59ef70d21

Successfully tagged tsuresh218/prod:latest

The push refers to repository [docker.io/tsuresh218/prod]

df6b3044b6fd: Preparing

7f00c391760a: Preparing

48664ac10d74: Preparing

d456ad64e1a7: Preparing

0d5f5a015e5d: Preparing

3c777d951de2: Preparing

f8a91dd5fc84: Preparing

cb81227abde5: Preparing

e01a454893a9: Preparing

c45660adde37: Preparing

fe0fb3ab4a0f: Preparing

f1186e5061f2: Preparing

b2dba7477754: Preparing

f8a91dd5fc84: Waiting

cb81227abde5: Waiting

e01a454893a9: Waiting

c45660adde37: Waiting

fe0fb3ab4a0f: Waiting

f1186e5061f2: Waiting

b2dba7477754: Waiting

3c777d951de2: Waiting

0d5f5a015e5d: Layer already exists

d456ad64e1a7: Layer already exists

48664ac10d74: Layer already exists

f8a91dd5fc84: Layer already exists

3c777d951de2: Layer already exists

cb81227abde5: Layer already exists

df6b3044b6fd: Mounted from tsuresh218/dev

e01a454893a9: Layer already exists

c45660adde37: Layer already exists

7f00c391760a: Mounted from tsuresh218/dev

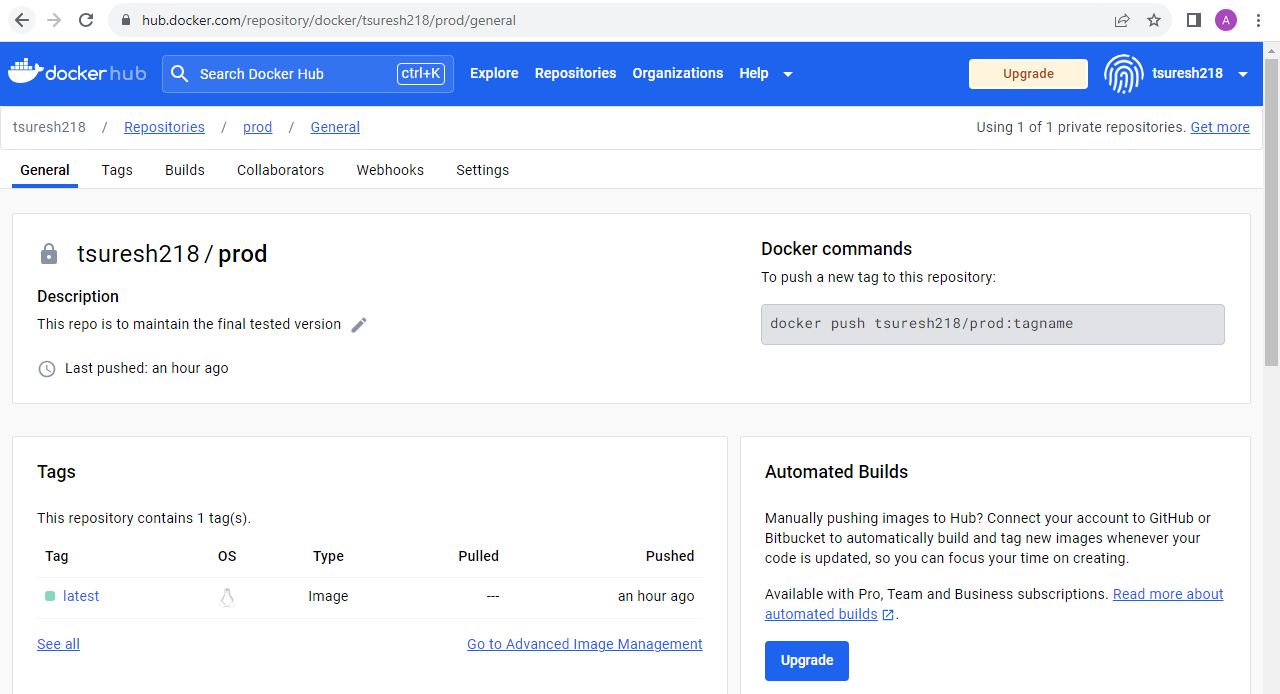
fe0fb3ab4a0f: Layer already exists

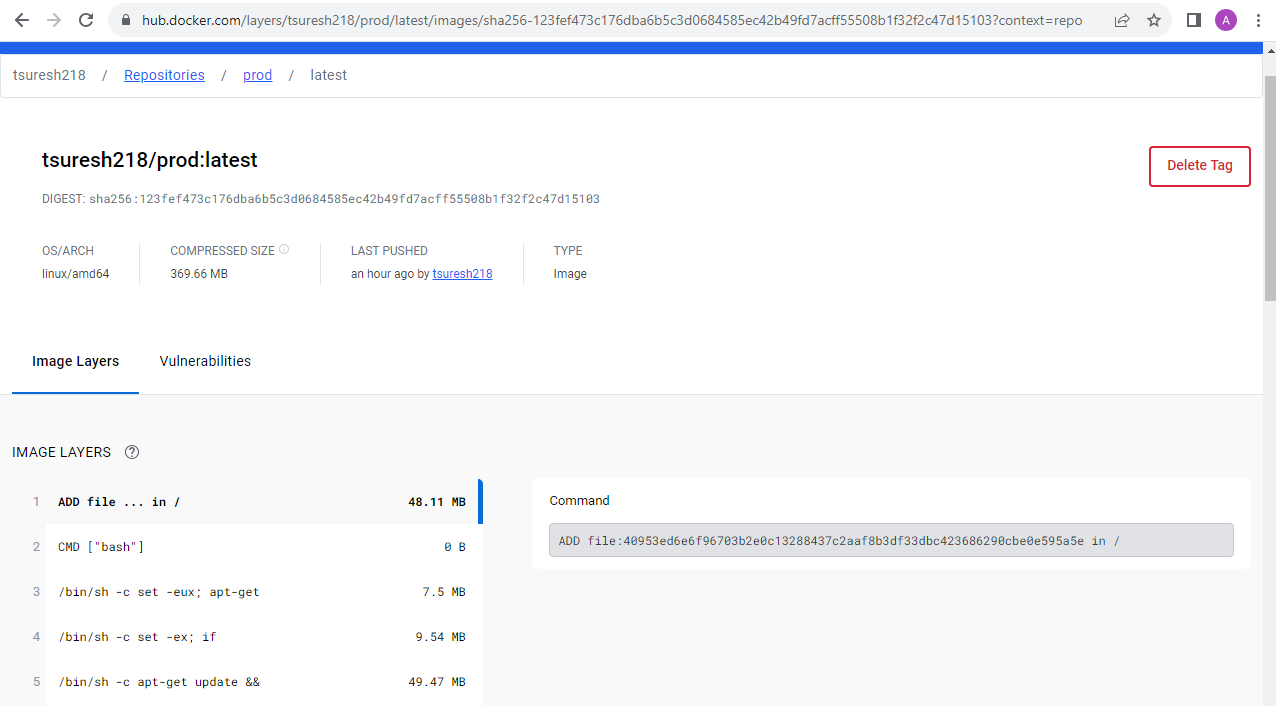
f1186e5061f2: Layer already exists

b2dba7477754: Layer already exists

latest: digest: sha256:123fef473c176dba6b5c3d0684585ec42b49fd7acff55508b1f32f2c47d15103 size: 3053

Finished: SUCCESS

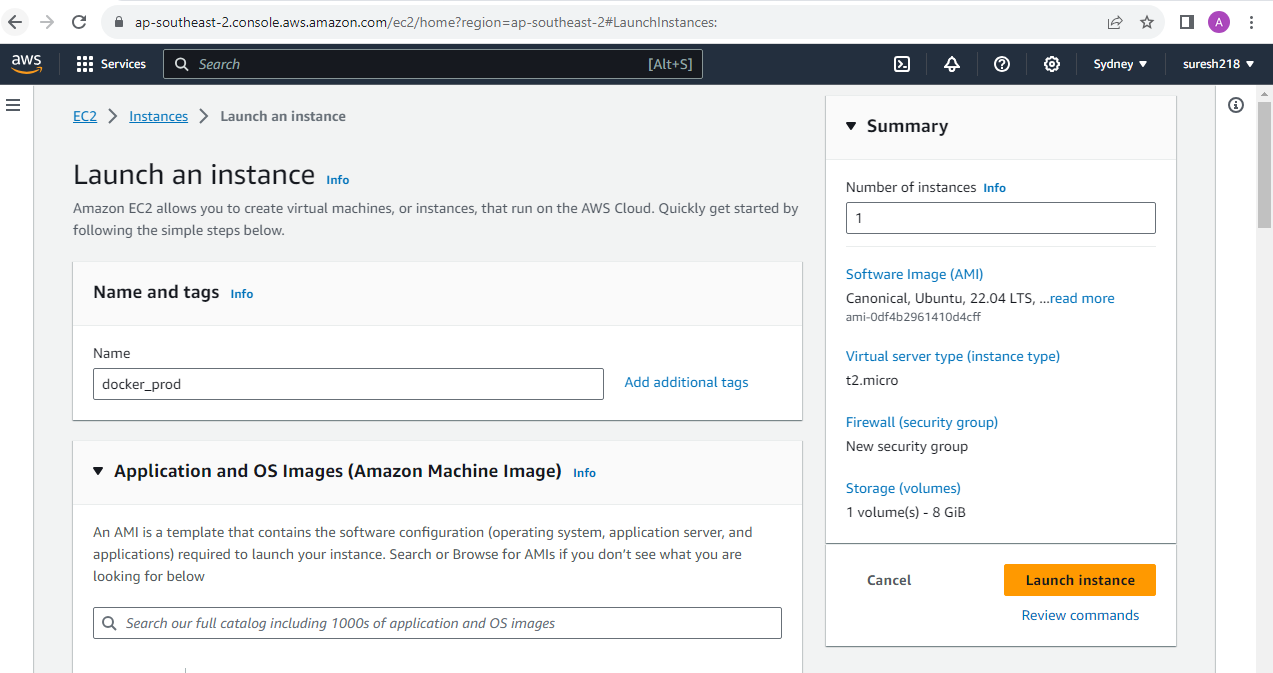


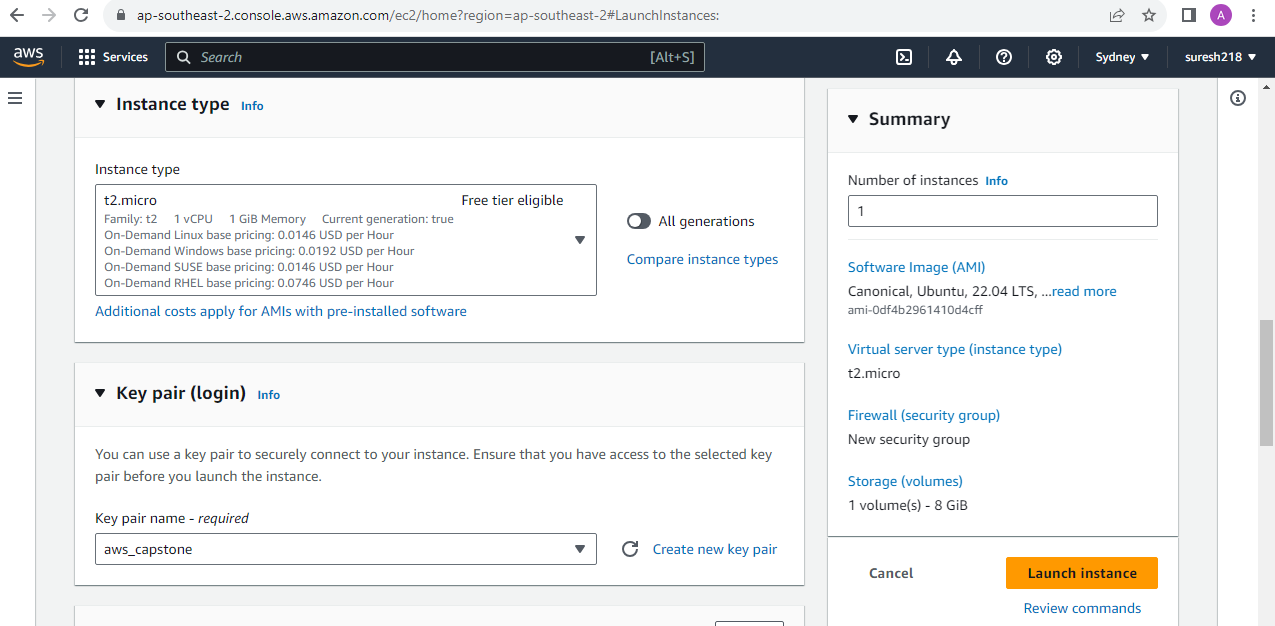


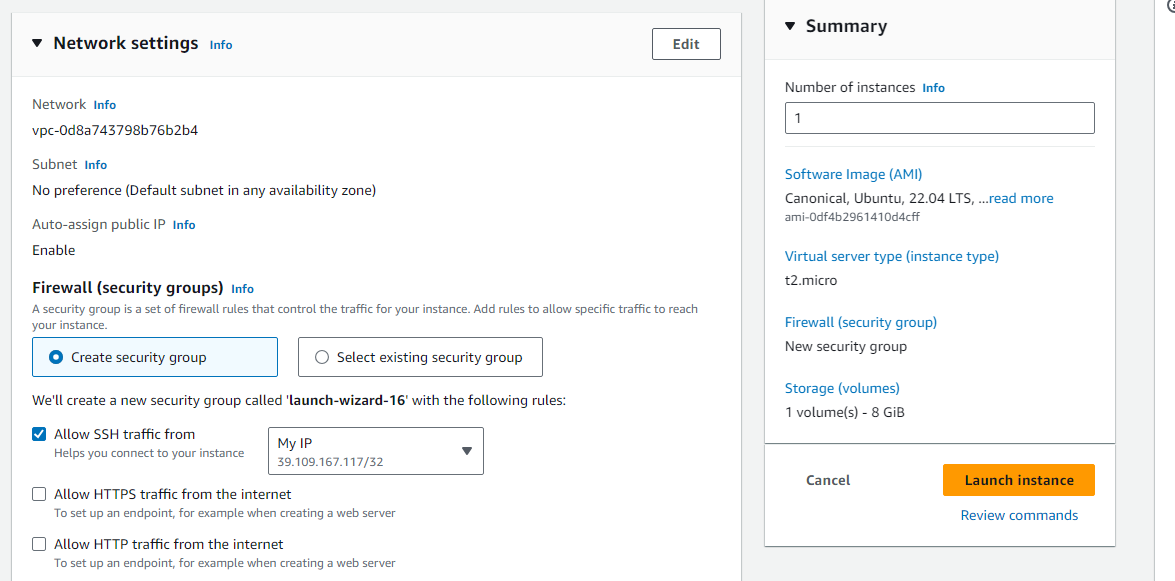
**AWS**

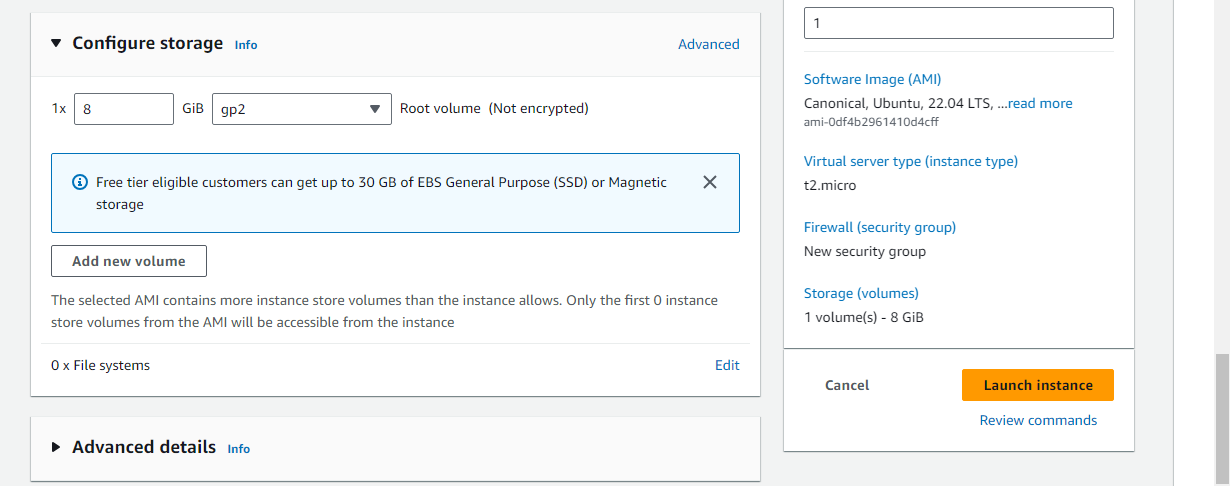
Launched a t2. micro instance and deployed the prod ready created docker application

* Instance has been created with free tier options and security groups have been updated as per requirements





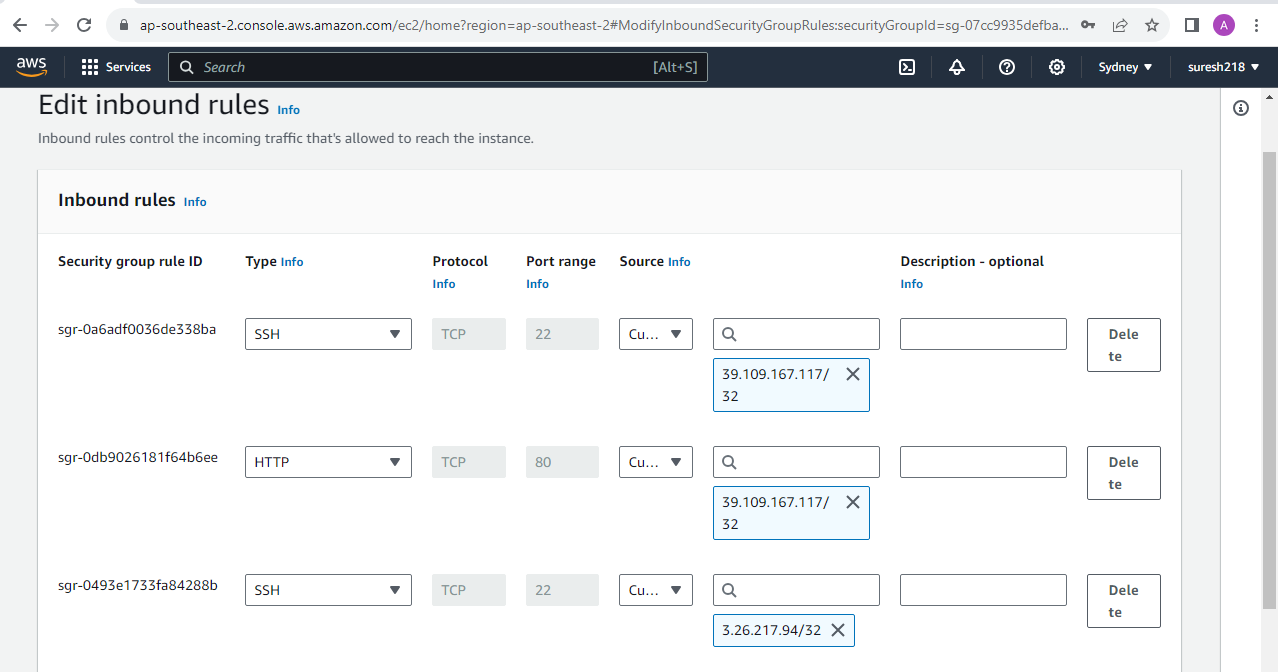






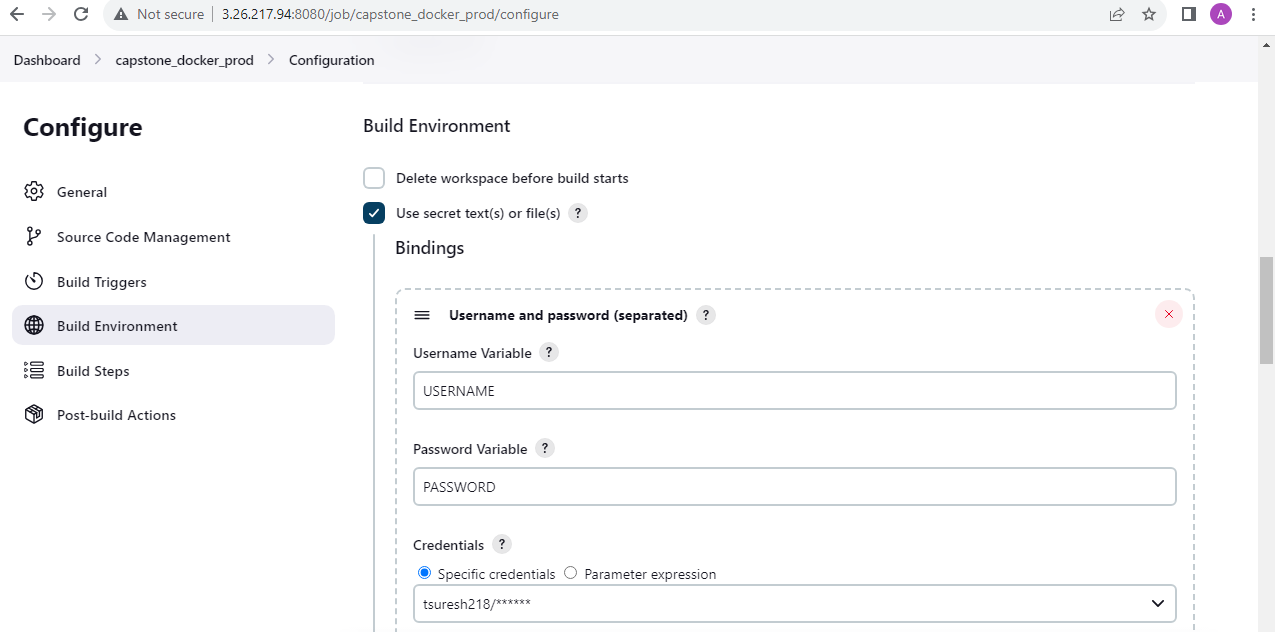
**Security Group access restrictions:**

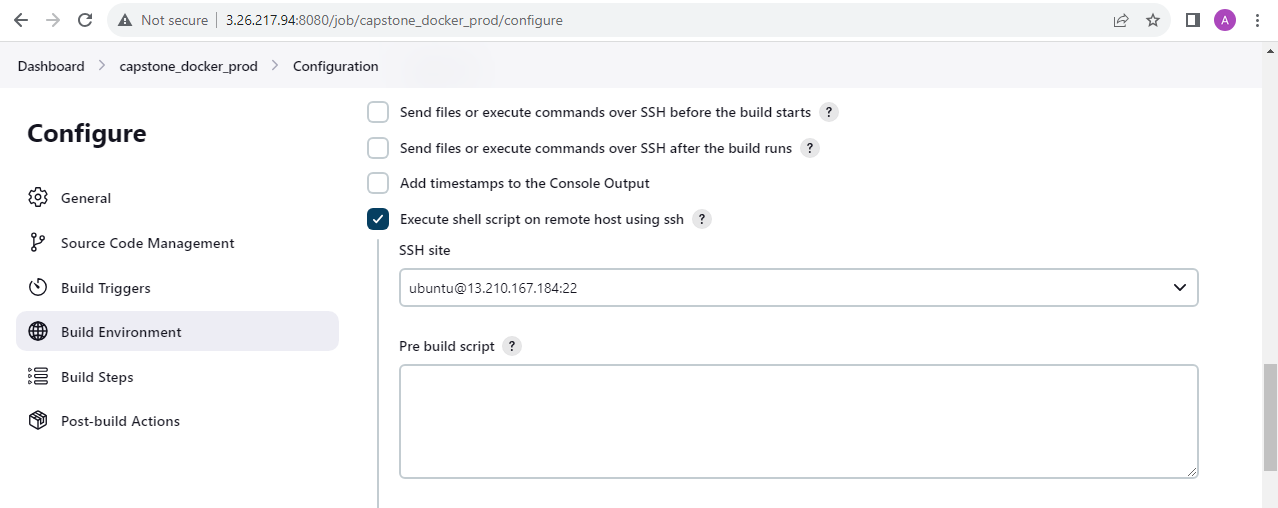
* Given access to jenkins server to ssh and install docker & deploy the application
* Login (SSH) to server can should be made only from this ip address
* Below SG shows that whoever has the ip address can access the application



**Jenkins job configuration to install docker & deploy the docker application**

* Jenkins job has been created with shell script to connect to the aws instance via ssh and install docker and deploy the prod ready docker application





**jenkins\_ssh\_script** has been used to execute to connect the ec2 instance and install the docker and deploy the docker application as below

#bin/bash

DOCKER\_IMAGE=tsuresh218/prod:latest

CONTAINER\_NAME=my-application

DOCKERHUB\_USERNAME=$USERNAME

DOCKER\_PASSWORD=$PASSWORD

# Update repositories

sudo apt-get update

# Install docker

sudo apt-get install -y docker.io

# Verify Docker is installed

if [ "$(which docker)" != "" ]; then

    echo "Docker is installed at $(which docker)"

else

    echo "Docker is not installed, please check the installation step"

    exit 1

fi

#Give permission

sudo usermod -aG docker ubuntu

# Login to Docker

echo $DOCKER\_PASSWORD | docker login -u $DOCKERHUB\_USERNAME --password-stdin

# Pull the Docker image

docker pull $DOCKER\_IMAGE

if [ "$(sudo docker images -q $DOCKER\_IMAGE 2> /dev/null)" != "" ]; then

    echo "Successfully pulled $DOCKER\_IMAGE image"

else

    echo "Image could not be pulled, please verify the image name"

    exit 1

fi

# Run the image

sudo docker run -d -p 80:3000 --name $CONTAINER\_NAME $DOCKER\_IMAGE

#Verify the image is running

# Check if Docker image is running

if [ $(docker inspect -f '{{.State.Running}}' $CONTAINER\_NAME) = "true" ]; then

    echo "Docker container '$CONTAINER\_NAME is deployed and running"

else

    echo "Docker container '$CONTAINER\_NAME is not deployed and running"

fi

# Check the docker container is running

docker ps -a

## Exit Successfully

exit 0

**Console Output:**

Started by user [tsuresh218](http://3.26.217.94:8080/user/tsuresh218)

Running as SYSTEM

Building in workspace /var/lib/jenkins/workspace/capstone\_docker\_prod

[SSH] executing pre build script:

[SSH] executing post build script:

USERNAME="tsuresh218"

USER="jenkins"

PASSWORD=\*\*\*\*\*\*\*\*\*\*

#bin/bash

DOCKER\_IMAGE=tsuresh218/prod:latest

CONTAINER\_NAME=my-application

DOCKERHUB\_USERNAME=$USERNAME

DOCKER\_PASSWORD=$PASSWORD

# Update repositories

sudo apt-get update

# Install docker

sudo apt-get install -y docker.io

# Verify Docker is installed

if [ "$(which docker)" != "" ]; then

echo "Docker is installed at $(which docker)"

else

echo "Docker is not installed, please check the installation step"

exit 1

fi

#Give permission

sudo usermod -aG docker ubuntu

# Login to Docker

echo $DOCKER\_PASSWORD | docker login -u $DOCKERHUB\_USERNAME --password-stdin

# Pull the Docker image

docker pull $DOCKER\_IMAGE

if [ "$(sudo docker images -q $DOCKER\_IMAGE 2> /dev/null)" != "" ]; then

echo "Successfully pulled $DOCKER\_IMAGE image"

else

echo "Image could not be pulled, please verify the image name"

exit 1

fi

# Run the image

sudo docker run -d -p 80:3000 --name $CONTAINER\_NAME $DOCKER\_IMAGE

#Verify the image is running

# Check if Docker image is running

if [ $(docker inspect -f '{{.State.Running}}' $CONTAINER\_NAME) = "true" ]; then

echo "Docker container '$CONTAINER\_NAME is deployed and running"

else

echo "Docker container '$CONTAINER\_NAME is not deployed and running"

fi

# Check the docker container is running

docker ps -a

## Exit Successfully

exit 0

Hit:1 <http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu> jammy InRelease

Hit:2 <http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu> jammy-updates InRelease

Hit:3 <http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu> jammy-backports InRelease

Hit:4 <http://security.ubuntu.com/ubuntu> jammy-security InRelease

Reading package lists...

Reading package lists...

Building dependency tree...

Reading state information...

docker.io is already the newest version (24.0.5-0ubuntu1~22.04.1).

0 upgraded, 0 newly installed, 0 to remove and 49 not upgraded.

Docker is installed at /usr/bin/docker

WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.

Configure a credential helper to remove this warning. See

<https://docs.docker.com/engine/reference/commandline/login/#credentials-store>

Login Succeeded

latest: Pulling from tsuresh218/prod

Digest: sha256:123fef473c176dba6b5c3d0684585ec42b49fd7acff55508b1f32f2c47d15103

Status: Image is up to date for tsuresh218/prod:latest

docker.io/tsuresh218/prod:latest

Successfully pulled tsuresh218/prod:latest image

9adcd5280000a56183b083582afc23f39db2c792ed8552affb08b8143869c509

Docker container 'my-application is deployed and running

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

9adcd5280000 tsuresh218/prod:latest "docker-entrypoint.s…" 1 second ago Up Less than a second 0.0.0.0:80->3000/tcp, :::80->3000/tcp my-application

[SSH] completed

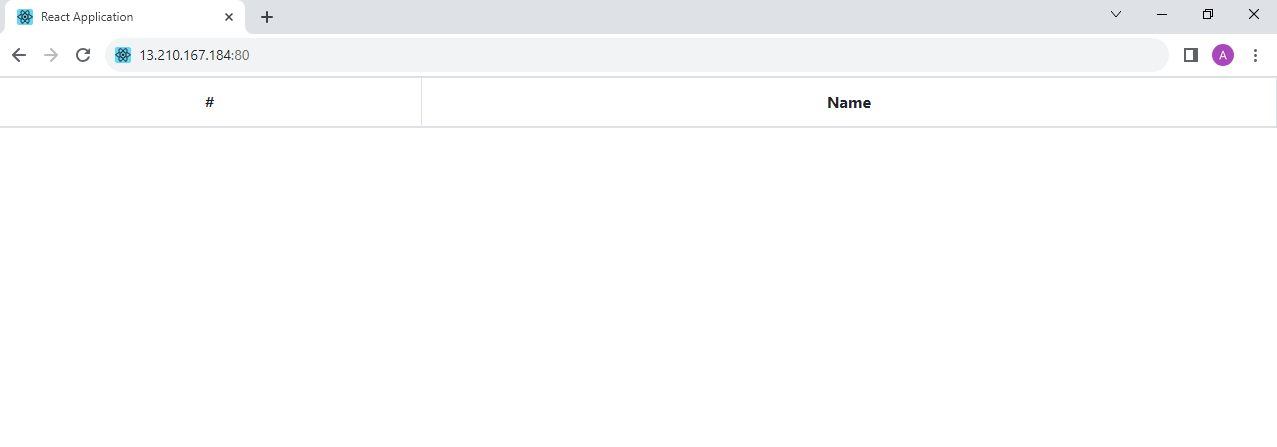
[SSH] exit-status: 0

Finished: SUCCESS

**Deployed Site page:**

* Docker application has been deployed and brought up as shown below.

<http://13.210.167.184:80/>



**Monitoring**

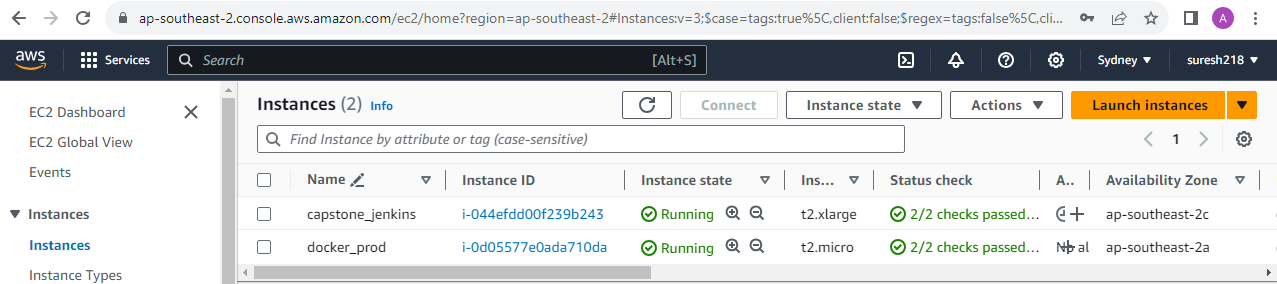
**Setup Monitoring system to check the health status of the application**

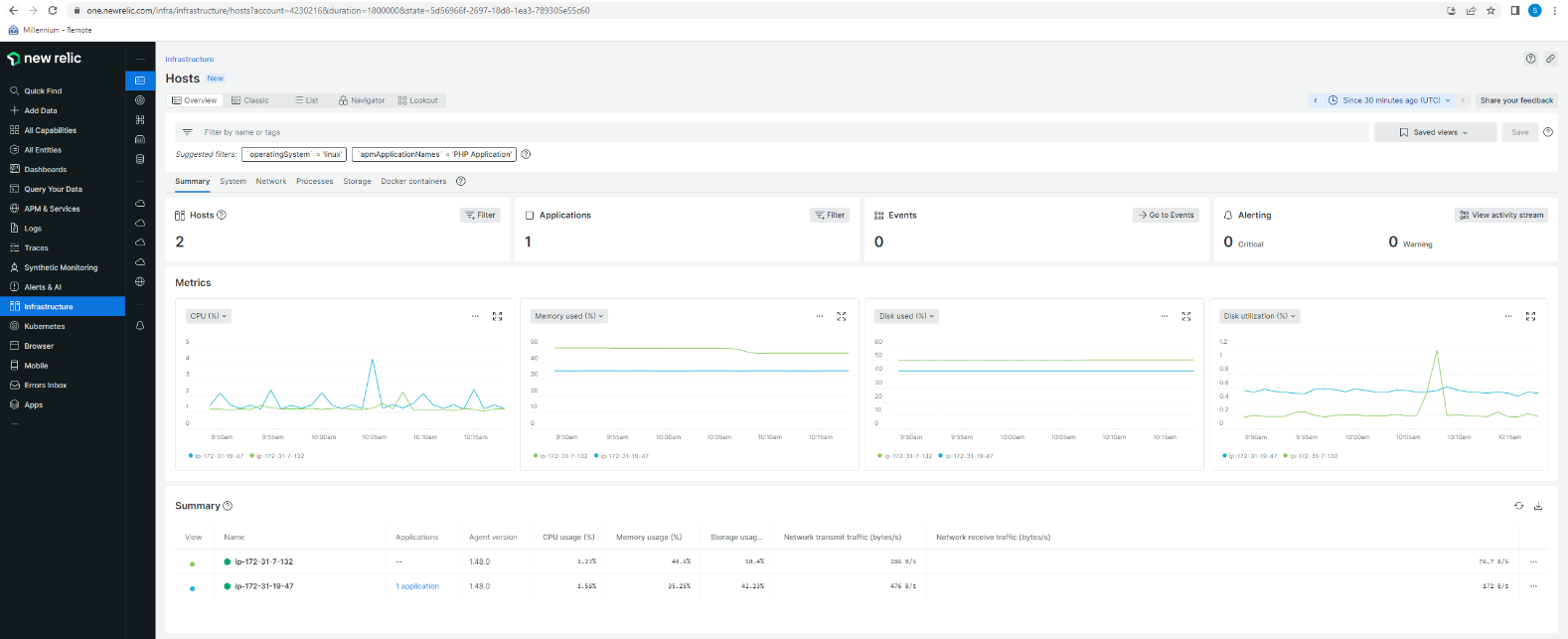
Health Monitoring of the servers have been setup and configured in New Relic which is a web-based software used for full-stack monitoring. It allows us to monitor applications, infrastructure and other components on a single platform.

Here’s the monitoring tool URL:

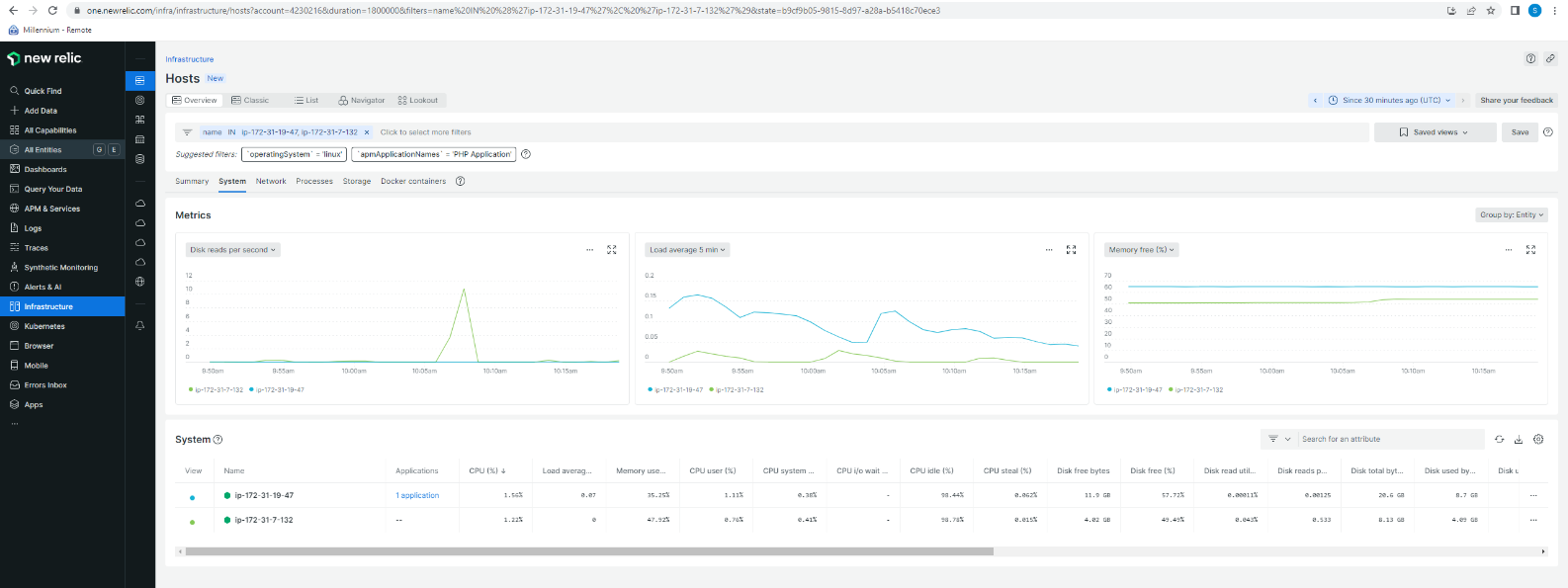
<https://one.newrelic.com/infra/infrastructure/hosts?account=4230216&duration=1800000&state=67b12b61-dbe7-273e-d307-fa87c27eee74>

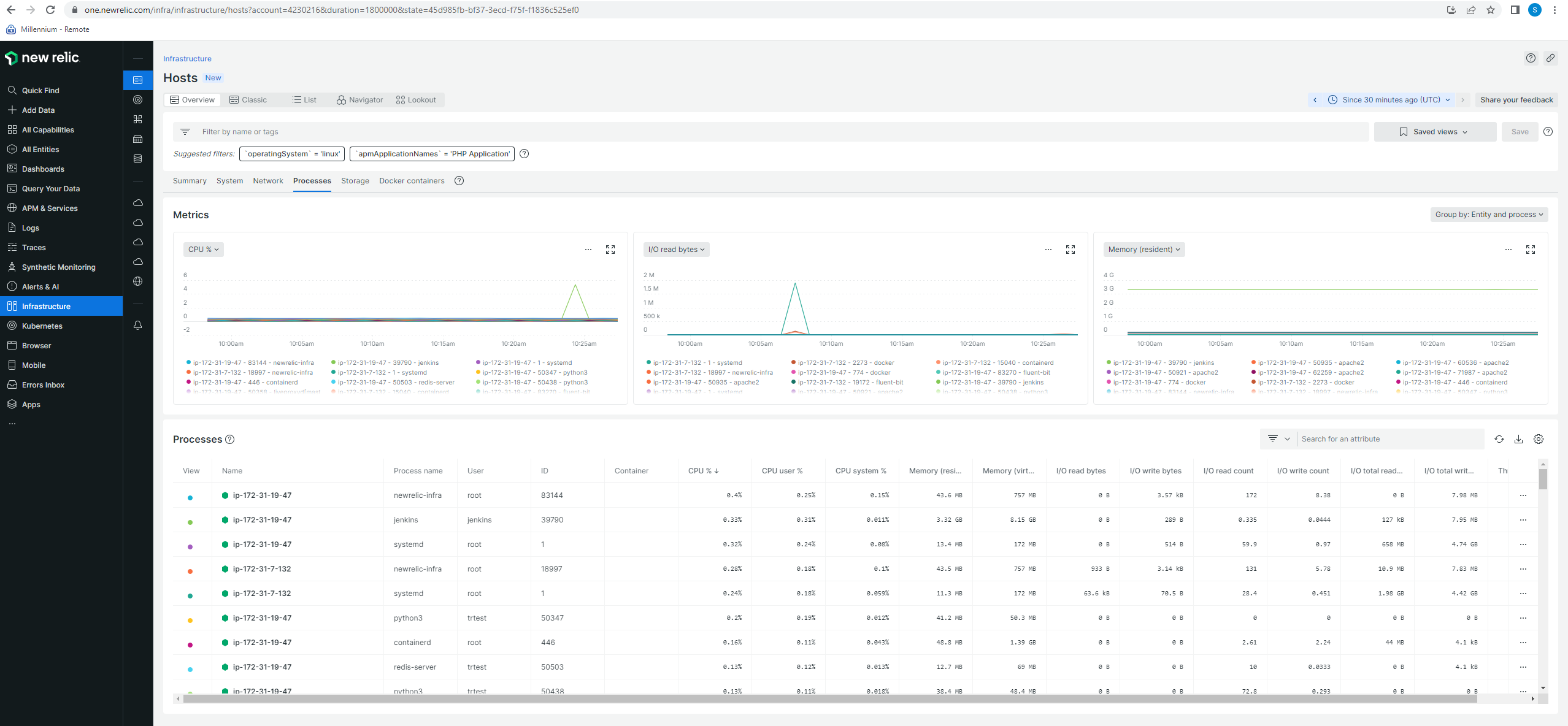
* Below dashboard screenshot shows the various components of the aws instances which was used for this project.
* Two instances have been used for this testing and deployment
* Dashboard shows the different instance component information (i.e. System, Network, Process & Storage)

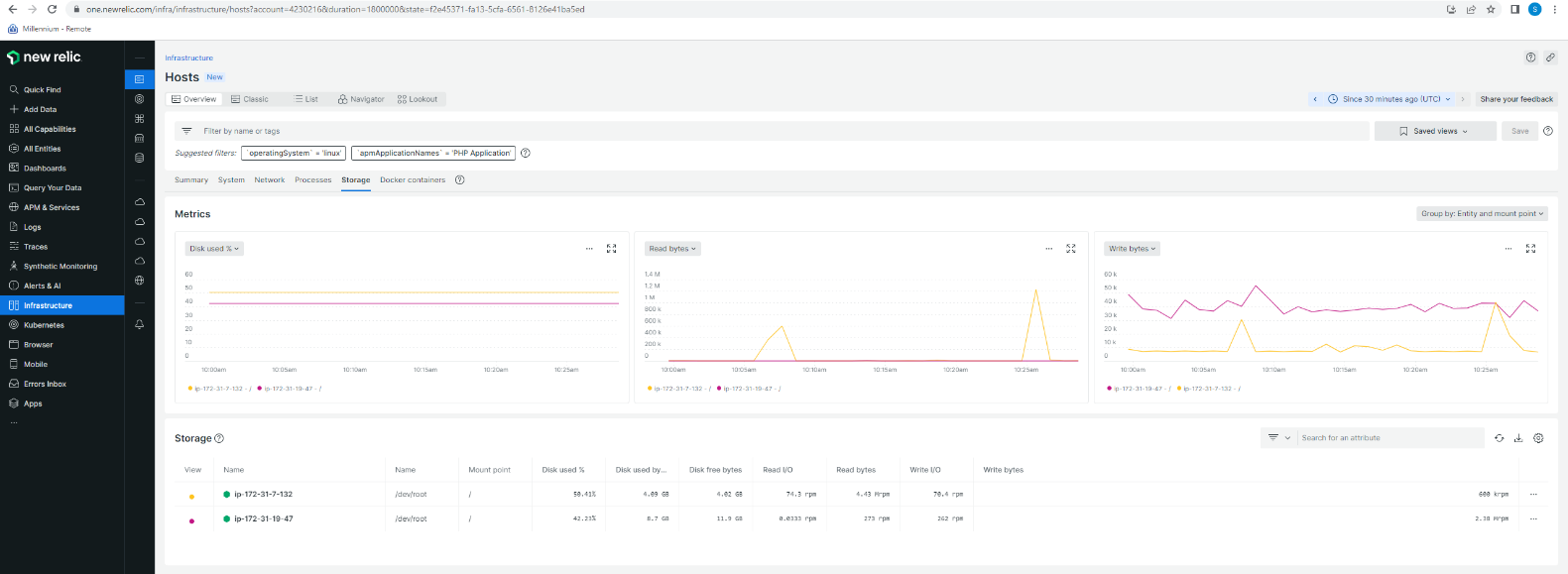


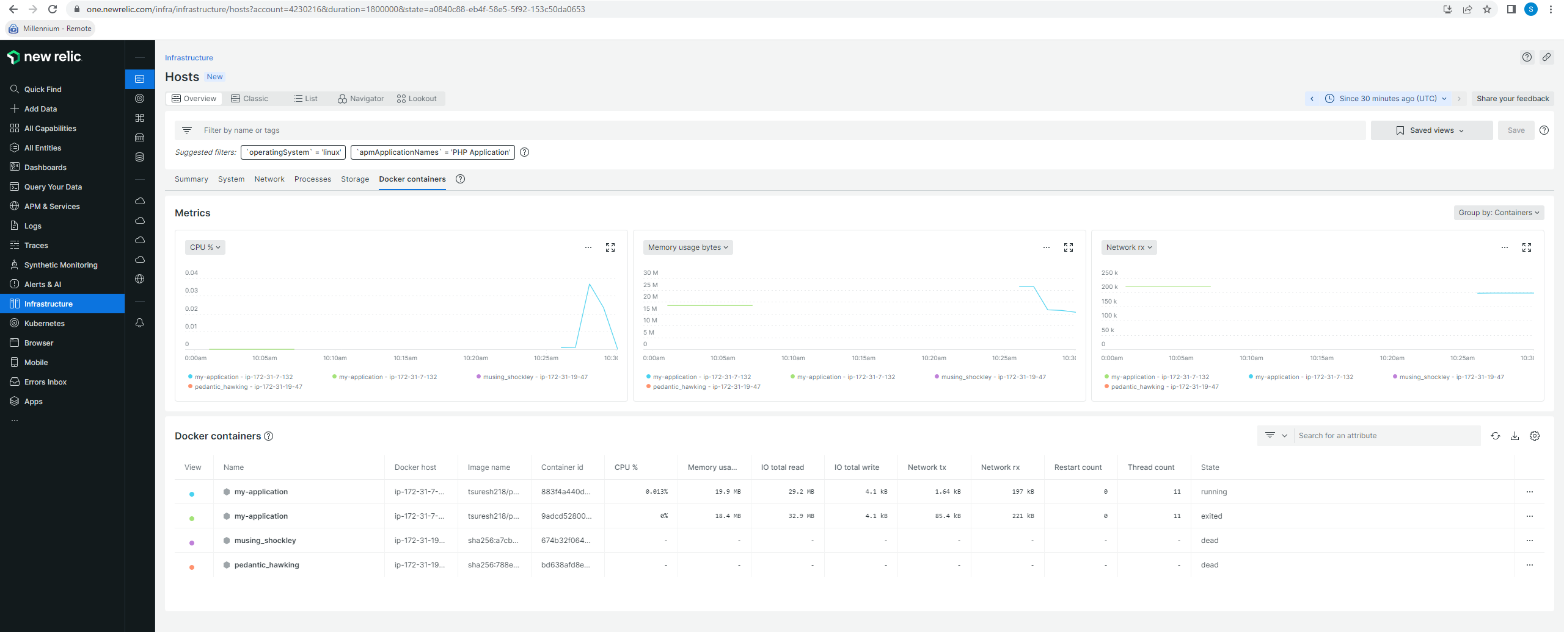
****

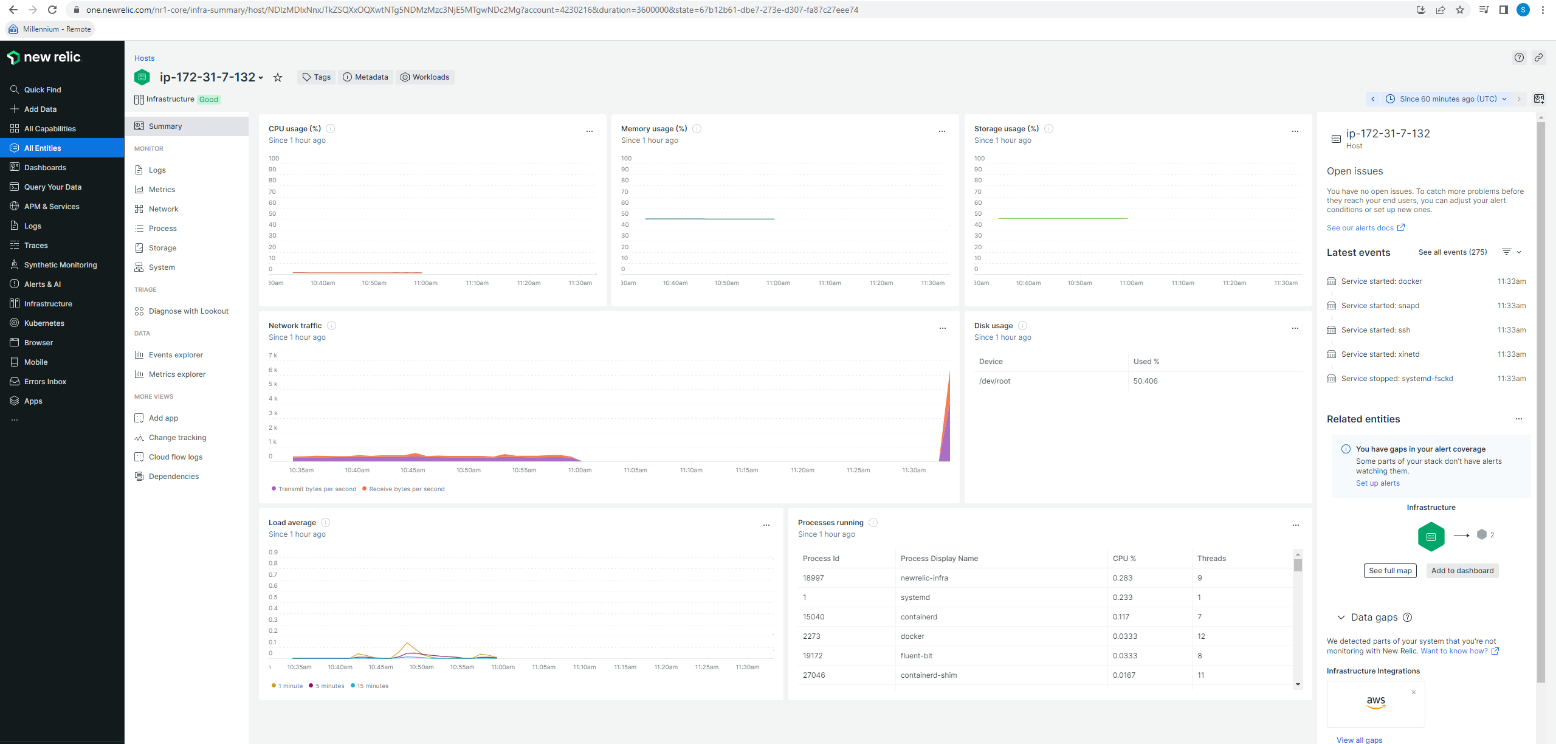
****

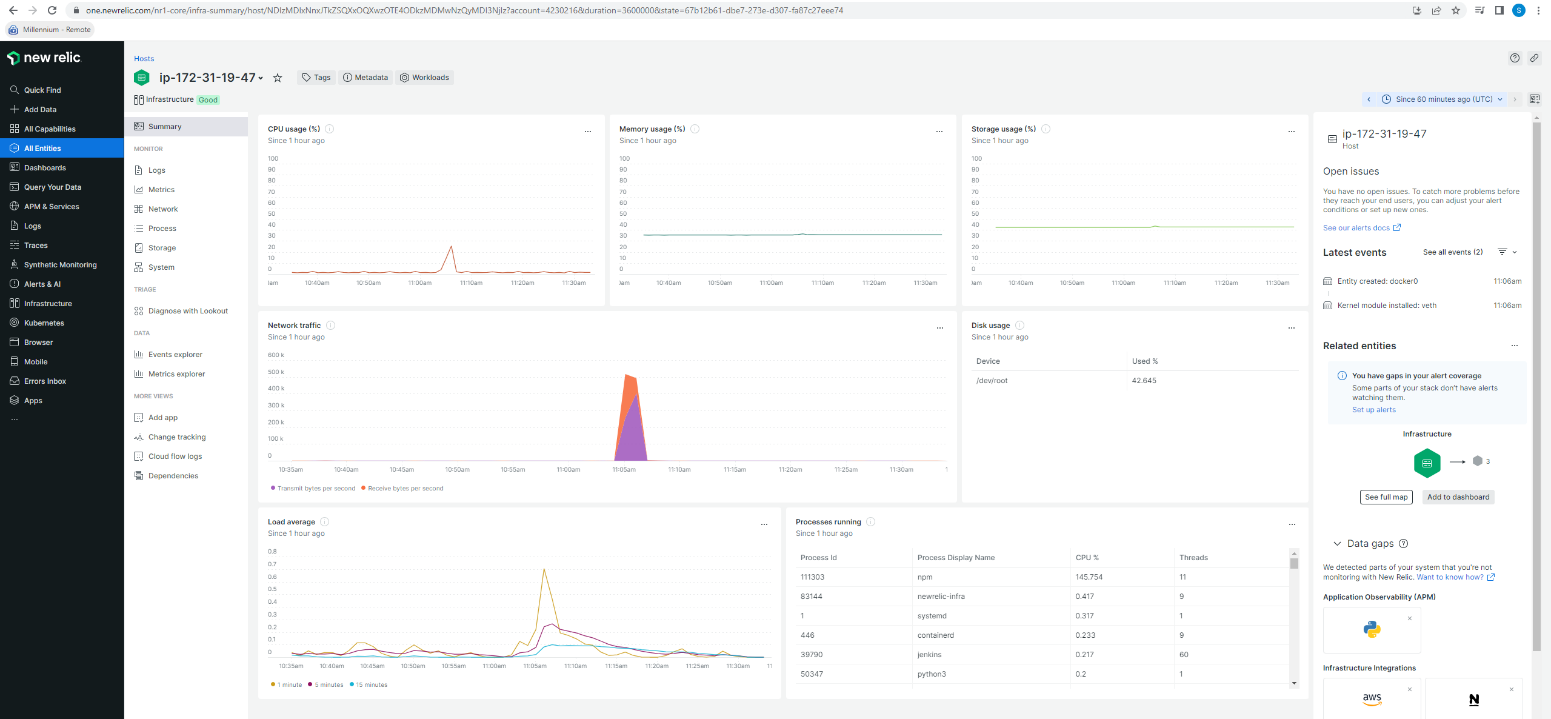






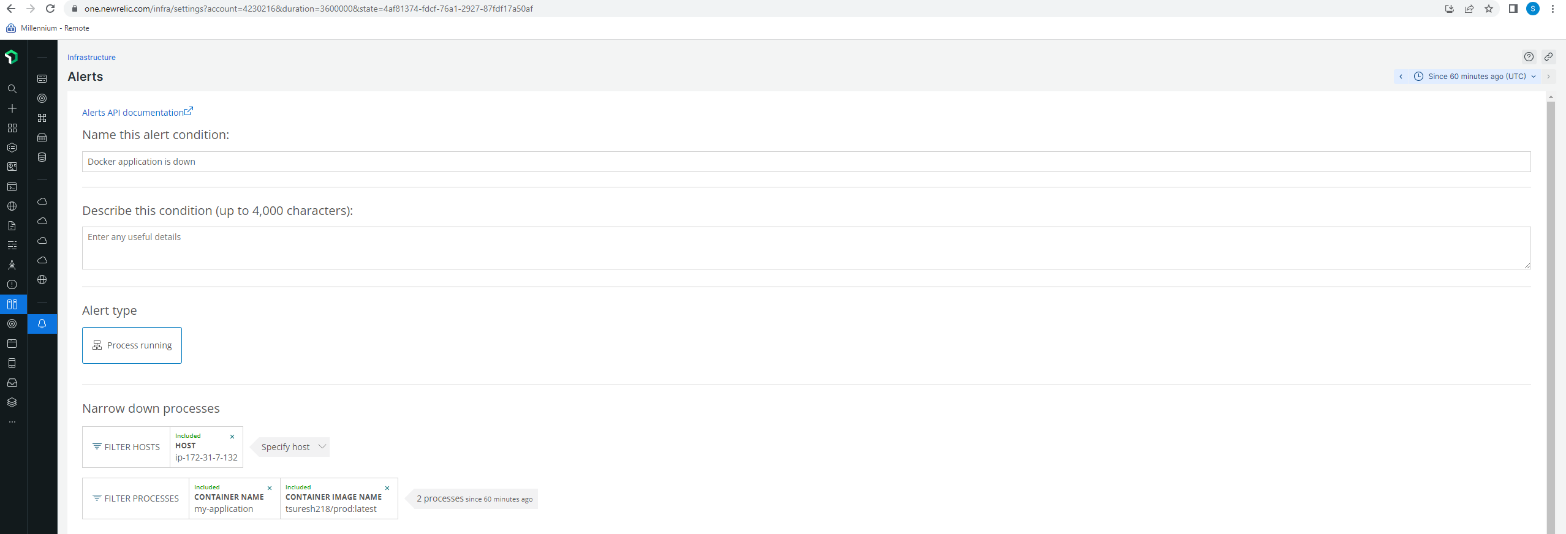


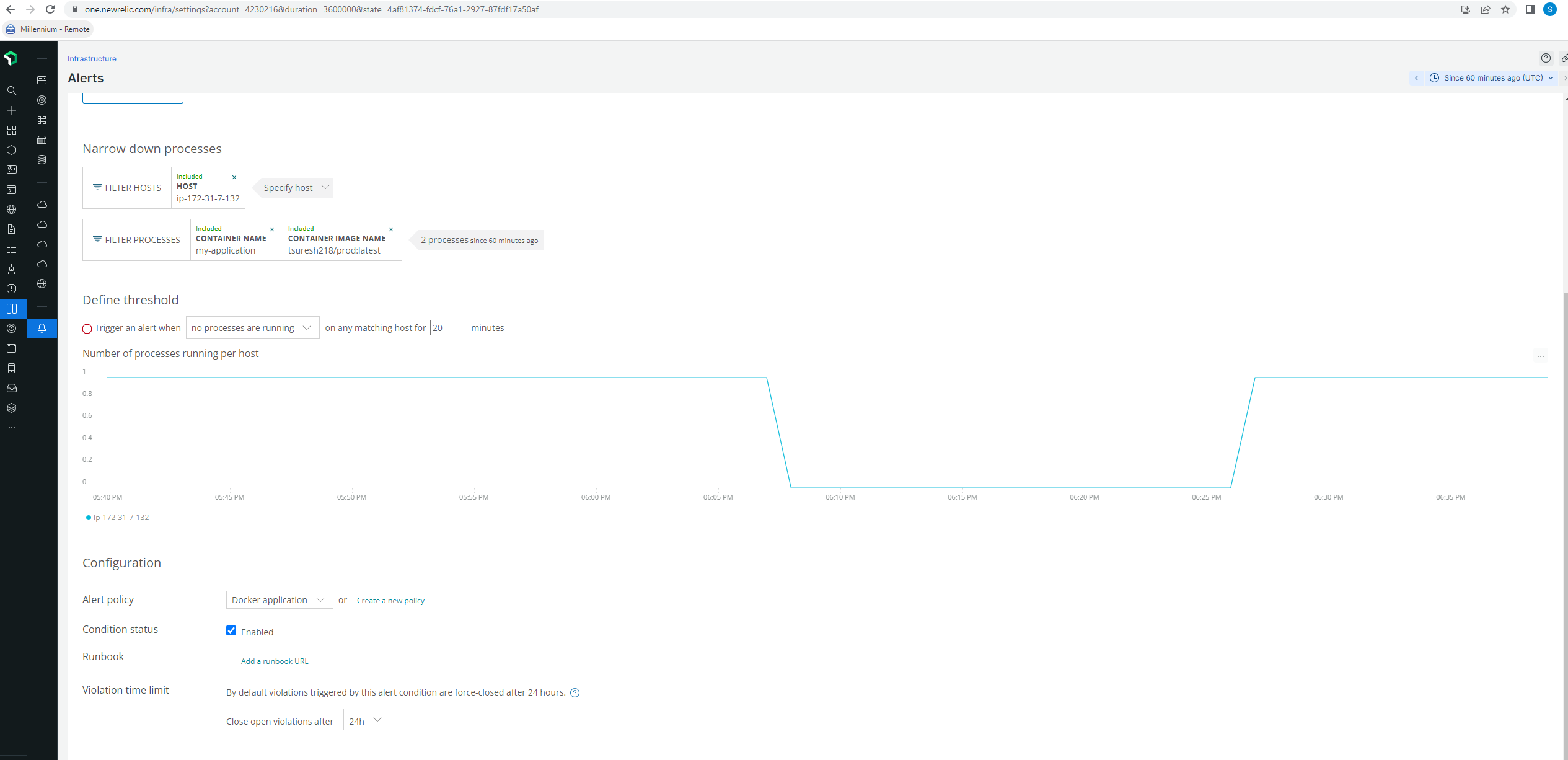




**Sending notification when application goes down**

* Below screenshot shows how to setup and configure an alert notification when a docker application goes down





* Below screenshot shows the alert notification of a docker application, it says docker application is down

