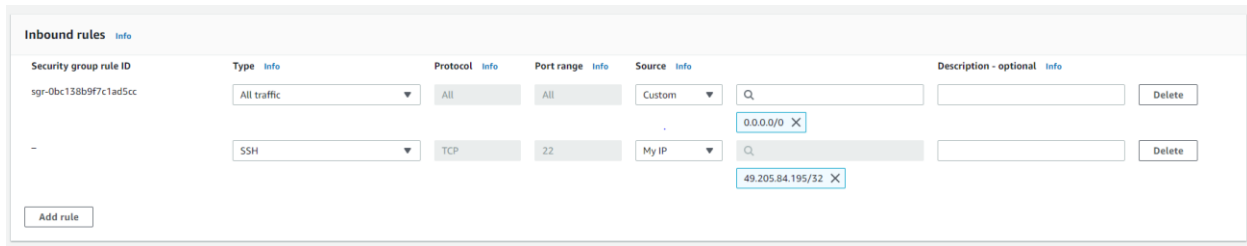


Step1.

EC2 instance with below mentioned **Security Groups**.



The screenshot shows the 'Inbound rules' configuration for a security group. It lists two rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	Actions
sgr-0bc138b9f7c1ad5cc	All traffic	All	All	Custom	0.0.0.0/0	Delete
-	SSH	TCP	22	My IP	49.205.84.195/32	Delete

An 'Add rule' button is located at the bottom left.

Step2.

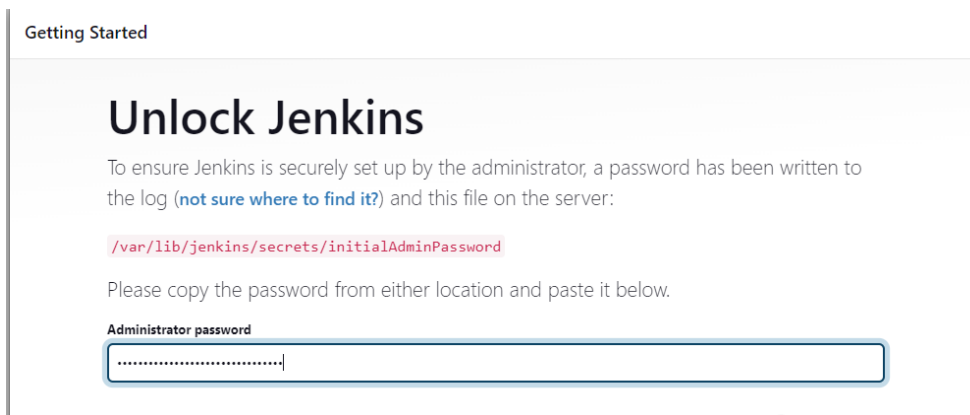
In EC2 instance installation packages

1. `sudo apt-get update`
2. `sudo apt install docker.io -y`
3. `sudo apt install docker-compose -y`
 adding docker user to the ubuntu group
 `sudo usermod -aG docker ubuntu` – and restart the terminal
 `sudo chmod 777 /var/run/docker.sock`
4. Jenkins installation steps
 {Install jdk first - `sudo apt install openjdk-11-jre -y`
 And following the steps mentioned in this Jenkins website
 <https://www.jenkins.io/doc/book/installing/linux/>

Step3.

Jenkins

`sudo cat /var/lib/jenkins/secrets/initialAdminPassword`



The screenshot shows the 'Getting Started' section of the Jenkins web interface, specifically the 'Unlock Jenkins' step. It explains that a password has been written to the log and the file `/var/lib/jenkins/secrets/initialAdminPassword`. It instructs the user to copy the password and paste it into the provided field.

Administrator password

.....|

Installing plug-in

Getting Started

Getting Started

✓ Folders	🔄 OWASP Markup Formatter	🔄 Build Timeout	🔄 Credentials Binding	** Ionicons API Folders ** JavaBeans Activation Framework (JAF) API ** JavaMail API ** bouncycastle API
🔄 Timestampers	🔄 Workspace Cleanup	🔄 Ant	🔄 Gradle	
🔄 Pipeline	🔄 GitHub Branch Source	🔄 Pipeline: GitHub Groovy Libraries	🔄 Pipeline: Stage View	
🔄 Git	🔄 SSH Build Agents	<input type="radio"/> Matrix Authorization Strategy	🔄 PAM Authentication	
🔄 LDAP	🔄 Email Extension	🔄 Mailer		

Jenkins Admin User

Create First Admin User

Username

Prakash

Password

.....

Confirm password

.....

Full name

Prakash

E-mail address

prakash@guvi.in

Description

This is Capstone project

[Plain text] [Preview](#)

☒ Discard old builds ?

Strategy

Log Rotation

Days to keep builds

if not empty, build records are only kept up to this number of days

Max # of builds to keep

if not empty, only up to this number of build records are kept

☒ This project is parameterized ?

≡ String Parameter ?

Name ?

IMAGE_NAME

Default Value ?

reactjs:version1

Description ?

This parameter for docker image name and version

Git ?

Repositories ?

Repository URL ?

https://github.com/Pakas142/Capstone.git

Credentials ?

Pakas142/***** (git user name and passwords)

Add ▾

Advanced ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/master

Branch Specifier (blank for 'any') ?

*/dev

Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ GitHub hook trigger for GITScm polling ?
- ☐ Poll SCM ?

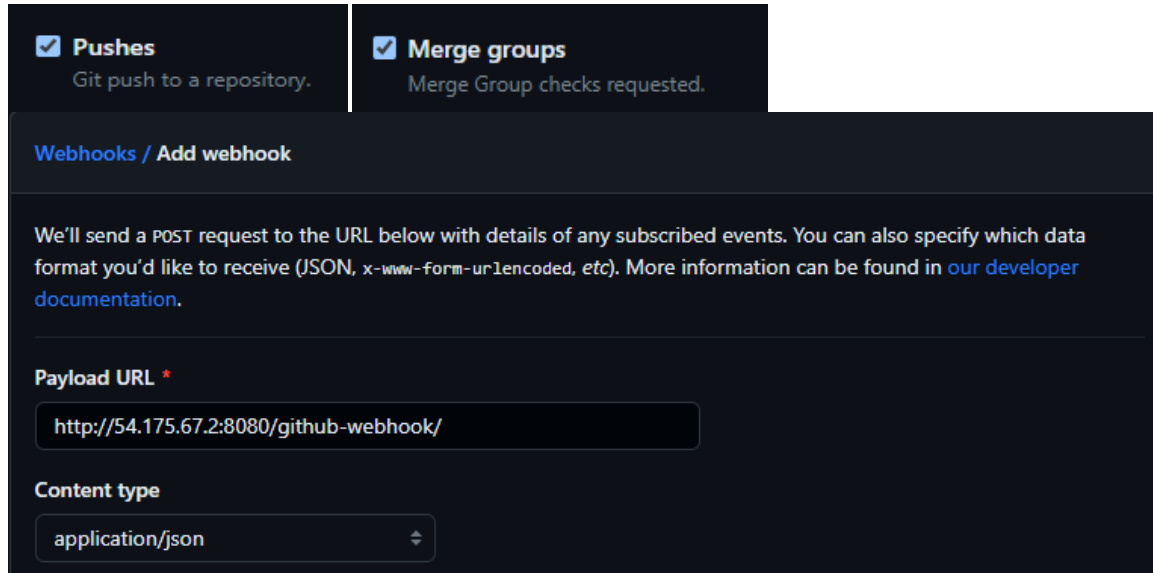
Build Environment

- ☒ Delete workspace before build starts

Advanced ▾

- ☐ Use secret text(s) or file(s) ?
- ☐ Add timestamps to the Console Output
- ☐ Inspect build log for published build scans
- ☐ Terminate a build if it's stuck
- ☐ With Ant ?

Github Webhook configuration



The screenshot shows the GitHub Webhook configuration page. At the top, there are two sections: 'Pushes' with a checked checkbox and the description 'Git push to a repository.', and 'Merge groups' with a checked checkbox and the description 'Merge Group checks requested.'. Below these is a header 'Webhooks / Add webhook'. The main content area explains that a POST request will be sent to the provided URL with details of subscribed events. It mentions that the data format can be JSON, x-www-form-urlencoded, etc., and refers to developer documentation. There are two input fields: 'Payload URL' with the value 'http://54.175.67.2:8080/github-webhook/' and 'Content type' with a dropdown menu showing 'application/json'.

Linux CLI Commands

- `git clone https://github.com/rvsp/reactjs-demo.git`
- `cd reactjs-demo/`

vi Dockerfile

```
FROM node:16-alpine as build
WORKDIR /usr/app
COPY . /usr/app
RUN npm install
COPY . .
RUN npm run build

FROM nginx:1.21-alpine
RUN rm
/usr/share/nginx/html/index.html
WORKDIR /usr/share/nginx/html/
COPY --from=build /usr/app/build/ ./
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

vi docker-compose.yml

```
version: "3"
services:
  reactjs:
    image: $IMAGE_NAME
    container_name: reactjs
    ports:
      - "80:80"
```

vi build.sh

```
if [ "$GIT_BRANCH" = "origin/dev" ]; then
  docker stop $(docker ps -aq)
  docker image prune -af
  docker build -t $IMAGE_NAME .
  docker login -u pakas142 -p Pakas@142
  docker tag $IMAGE_NAME pakas142/dev:latest
  docker push pakas142/dev:latest
elif [ "$GIT_BRANCH" = "origin/master" ]; then
  docker stop $(docker ps -aq)
  docker image prune -af
  docker build -t $IMAGE_NAME .
  docker login -u pakas142 -p Pakas@142
  docker tag $IMAGE_NAME pakas142/prod:latest
  docker push pakas142/prod:latest
else echo "either git pull not available in dev or
master "
fi
```

vi deploy.sh

```
docker stop $(docker ps -aq)
docker-compose up -d
docker ps
```

vi .dockerignore

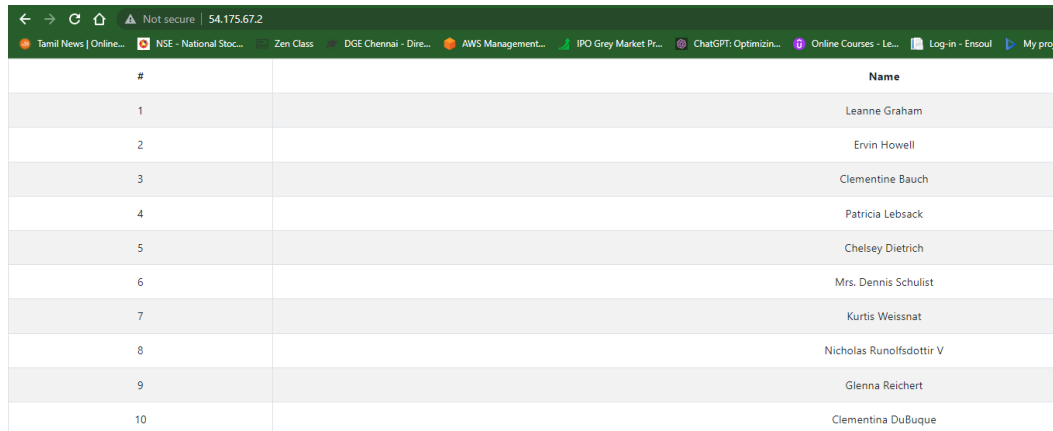
```
node_modules
.gitignore
build.sh
deploy.sh
```

GIT CLI commands

```
git clone https://github.com/rvsp/reactjs-demo.git
git remote remove origin
git add .
git commit -m "first commit"
git checkout -b master
git remote add origin https://github.com/Pakas142/Capstone.git
git push origin master
git checkout -b dev
git add .
git commit -m "last commit"
git push origin dev
git checkout master
git merge dev
```

Output of reactjs application

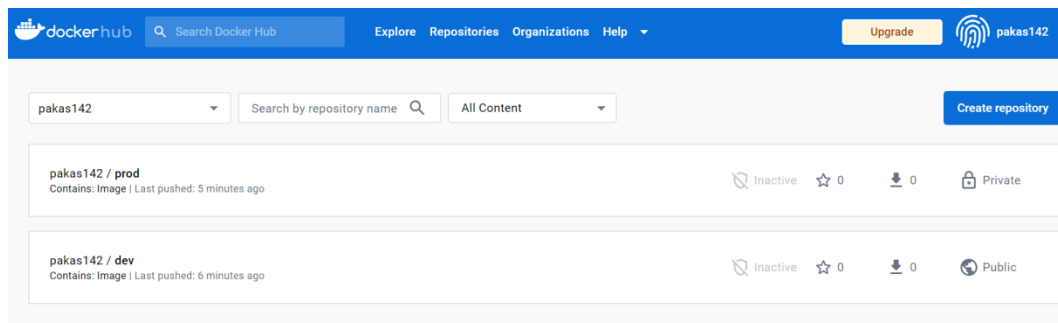
After Jenkins triggered the output be(<https://jsonplaceholder.typicode.com/users> used this fake address for the output names)



The screenshot shows a web browser window with a table of 10 users. The table has two columns: '#' and 'Name'. The data is as follows:

#	Name
1	Leanne Graham
2	Ervin Howell
3	Clementine Bauch
4	Patricia Lebsack
5	Chelsey Dietrich
6	Mrs. Dennis Schulist
7	Kurtis Weissnat
8	Nicholas Runolfsdottir V
9	Glenna Reichert
10	Clementina DuBuque

Pushed image to docker hub



Install monitoring tool (Prometheus & Grafana)

- Creating ec2 instance t2.micro for monitoring
 - `cd /opt`
 - `Sudo wget https://github.com/prometheus/prometheus/releases/download/v2.43.0/prometheus-2.43.0.linux-amd64.tar.gz`
 - `tar -xvzf prometheus-2.43.0.linux-amd64.tar.gz – untaring`
 - `sudo cp prometheus promtool /usr/local/bin/`
 - `./prometheus --config.file=prometheus.yml &`

- After starting Prometheus enter the ipaddress with host of 9090
- <http://34.228.157.58:9090/> - Prometheus dashboard
- In deployment server install node exporter
 - `cd /opt`
 - `wget https://github.com/prometheus/node_exporter/releases/download/v1.5.0/node_exporter-1.5.0.linux-amd64.tar.gz`
 - `tar -xvzf node_exporter-1.5.0.linux-amd64.tar.gz` – untaring
 - `./node_exporter` – starting a node exporter
- <http://54.175.67.2:9100/> - node exporter
- Go to the monitor server and adding the node ip-address and host number in yml file
- then kill the old monitoring Prometheus and start again with this command `“./prometheus --config.file=prometheus.yml &”`
- Installing Grafana
 - `wget https://dl.grafana.com/enterprise/release/grafana-enterprise-9.5.1.linux-amd64.tar.gz`
 - `cd /opt`
 - `tar -zxvf grafana-enterprise-9.5.1.linux-amd64.tar.gz` – untaring
 - `cd bin`
 - `./grafana-server &` - starting grafana
- 34.228.157.58/3000 – Grafana dashboard monitoring deployment server “Capstone”

