

NODEJS APPLICATION DEPLOYMENT TO EC2 WITH DOCKER

PROBLEM STATEMENT:

XYZ Corp is developing a Node.js-based web application and needs a robust deployment solution to efficiently build and deploy the application to EC2 instances. The challenge is to create a streamlined process using Jenkins, Docker, and AWS EC2, ensuring scalability and ease of management.

USE CASE SCENARIO:

- ➔ **Business Requirement:** XYZ Corp aims to deploy its Node.js application on AWS EC2 instances, leveraging Docker containers for consistency and scalability. The solution should integrate seamlessly with Jenkins, allowing for automated builds, tests, and deployments.
- ➔ **Technical Challenge:** The objective is to create a Dockerfile and Docker Compose file for the Node.js application, enabling easy containerization and deployment. Jenkins will be configured to trigger the build process, and the resulting Docker image will be deployed to EC2 instances, ensuring a reliable and efficient deployment pipeline for the Node.js application.

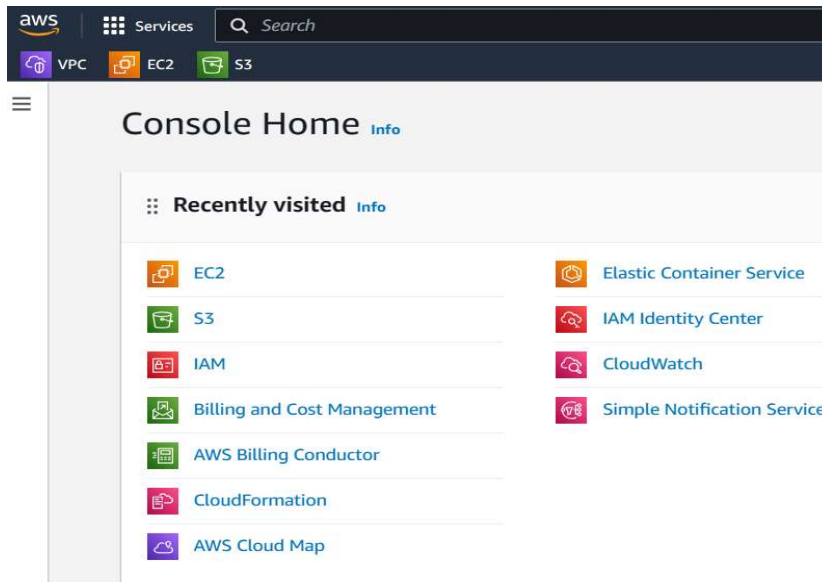
SOLUTION:

REQUIREMENTS:

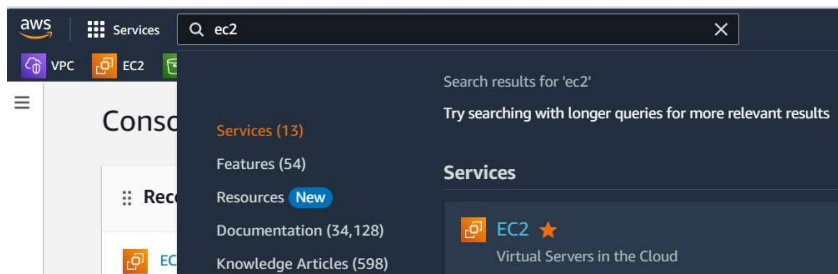
- ➔ AWS Cloud
- ➔ EC2 instance
- ➔ Docker
- ➔ Docker-compose
- ➔ GitHub
- ➔ Java
- ➔ Jenkins

Step:1 – Launching an EC2 Instance:

→ First login into your AWS instance:



→ Then on service search panel search EC2, click that one:



→ Then click launch instances, for creating an EC2 instance:

EC2 Dashboard X

EC2 Global View

Events

▼ **Instances**

- Instances
- Instance Types
- Launch Templates
- Spot Requests
- Savings Plans
- Reserved Instances
- Dedicated Hosts
- Capacity Reservations
- New**

▼ **Images**

- AMIs
- AMI Catalog

Resources EC2 Global view ⚙️ 🔄

You are using the following Amazon EC2 resources in the Asia Pacific (Mumbai) Region:

Instances (running)	0	Auto Scaling Groups	0	Dedicated Hosts	0
Elastic IPs	0	Instances	3	Key pairs	3
Load balancers	0	Placement groups	0	Security groups	9
Snapshots	0	Volumes	3		

Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Launch instance ▼

[Migrate a server](#) 🔗

Service health

[AWS Health Dashboard](#) 🔗 🔄

Region

Asia Pacific (Mumbai)

➔ Then name the instance according to your preferences:

[EC2](#) > [Instances](#) > Launch an instance

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines following the simple steps below.

Name and tags [Info](#)

Name

nodejs-deployment

➔ Then select the operating system according to your preferences:


▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, applications) required to launch your instance. Search or Browse for AMIs if you do below


Recents

Quick Start


Amazon Linux




macOS




Ubuntu




Windows



Red Hat



SUSE



Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type
ami-0287a05f0ef0e9d9a (64-bit (x86)) / ami-0b6581fde9e6e7779 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

➔ Then select the instance type: according to your preferences, but here I am selecting **t2.micro**

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro

Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true

On-Demand Linux base pricing: 0.0124 USD per Hour

On-Demand Windows base pricing: 0.017 USD per Hour

On-Demand RHEL base pricing: 0.0724 USD per Hour

On-Demand SUSE base pricing: 0.0124 USD per Hour

[Additional costs apply for AMIs with pre-installed software](#)

➔ Then select the key pair, according to your preferences, but here I am proceeding with **key pair option**, you can go with proceed with **without key pair option**:

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access before you launch the instance.

Key pair name - *required*

➔ Then keeping the default options under network settings:

▼ Network settings [Info](#)

Network [Info](#)
vpc-04dc687e3ffd22a68

Subnet [Info](#)
No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)
Enable

Firewall (security groups) [Info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to and from your instance.

☒ Create security group ☐ Select existing security group

We'll create a new security group called 'launch-wizard-4' with the following rules:

☒ Allow SSH traffic from

Anywhere
0.0.0.0/0

Helps you connect to your instance

☐ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

☐ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

➔ Then keeping default options for the rest of the settings, click launch instance:

▼ Configure storage [Info](#) Advanced

1x GiB Root volume (Not encrypted)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

Add new volume

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

Click refresh to view backup information

The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

0 x File systems Edit

► Advanced details [Info](#)

Software Image (AMI)
Canonical, Ubuntu, 22.04 LTS, ...[read more](#)
ami-0287a05f0ef0e9d9a

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which Amazon Free Tier is available)

Cancel **Launch instance**

[Review commands](#)

➔ The instance has been launched successfully:

Instances (1) Info			
<input type="text"/> Find Instance by attribute or tag (case-sensitive)			
<input type="checkbox"/>	Name	Instance ID	Instance state
<input type="checkbox"/>	nodejs-deployment	i-07b8a342536eed26a	Pending

➔ Connecting the instance:

```

2. nodejs
• X11-forwarding : ✓ (remote display is forwarded through SSH)
► For more info, ctrl+click on help or visit our website.

Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-1012-aws x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

System information as of Fri Dec  8 16:37:48 UTC 2023

System load:  0.201171875      Processes:           115
Usage of /:   20.5% of 7.57GB   Users logged in:     0
Memory usage: 5%               IPv4 address for eth0: 172.31.20.19
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

```

Step:2 – Installing necessary packages required for this task:

➔ Creating a shell script, in that writing a necessary script to install these packages:

- ❖ Java
- ❖ Docker
- ❖ Docker-compose
- ❖ Jenkins

Script contains:

```
#!/bin/bash
```

```

#installing java:
apt-get update
apt-get install -y openjdk-11-jre

#installing docker:
apt-get update
apt-get install -y docker.io

#installing docker-compose:
apt-get update
apt-get install -y docker-compose

#installing jenkins:
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
    https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
    https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
    /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install -y jenkins

#checking the installed services:
echo "-----"
echo "-----"
echo "-----"
echo "This is the Java package - "
java --version
echo "-----"
echo "This is Jenkins package - "
jenkins --version
echo "-----"
echo "This is Docker package - "
docker --version
echo "-----"
echo "This is Docker package - "
docker-compose --version

```

➔ Changing the file permission and executing it:


```

root@ip-172-31-20-19:/home/ubuntu# vi package.sh
root@ip-172-31-20-19:/home/ubuntu# chmod +x package.sh
root@ip-172-31-20-19:/home/ubuntu# ./package.sh
Hit:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:4 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]

```

→ The packages have been installed successfully:

```

-----
-----
-----
This is the Java package -
openjdk 11.0.21 2023-10-17
OpenJDK Runtime Environment (build 11.0.21+9-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.21+9-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
-----
This is Jenkins package -
2.426.1
-----
This is Docker package -
Docker version 24.0.5, build 24.0.5-0ubuntu1~22.04.1
-----
This is Docker package -
docker-compose version 1.29.2, build unknown
root@ip-172-31-20-19:/home/ubuntu# █

```

Step:3 – Containerizing nodejs application by using docker and deploying it through Docker-compose:

→ Assuming that you having Nodejs application like this:

```

root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# ls -l
total 1196
-rw-r--r-- 1 root root 1295 Dec 8 16:48 README.md
-rw-r--r-- 1 root root 1207543 Dec 8 16:48 package-lock.json
-rw-r--r-- 1 root root 1123 Dec 8 16:48 package.json
drwxr-xr-x 2 root root 4096 Dec 8 16:48 public
drwxr-xr-x 8 root root 4096 Dec 8 16:48 src
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# █

```

→ Creating a dockerfile according to the application:

Dockerfile contains:

```

#choosing the base image as the build stage:
FROM node:16-alpine as build

```



```
#choosing working directory for the application:
WORKDIR /app

#copying the package.json file to app directory and
installing packages:
COPY package.json .
RUN npm install

#copying the rest of application code to the working
directory:
COPY . .

#building the application:
RUN npm run build

#second stage base image:
FROM nginx:alpine

#setting the working directory for this base image:
WORKDIR /usr/share/nginx/html/

#copying the first stage code to this stage
COPY --from=build /app/build .

#exposing the application:
EXPOSE 80

#Executing the application after creating image:
CMD ["nginx", "-g", "daemon off;"]
```

```
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# ls -l
total 1196
-rw-r--r-- 1 root root    1295 Dec  8 16:48 README.md
-rw-r--r-- 1 root root 1207543 Dec  8 16:48 package-lock.json
-rw-r--r-- 1 root root    1123 Dec  8 16:48 package.json
drwxr-xr-x 2 root root    4096 Dec  8 16:48 public
drwxr-xr-x 8 root root    4096 Dec  8 16:48 src
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# vi dockerfile
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app#
```

➔ Creating a docker image from the dockerfile by using **docker build** command:

```
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app#
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# docker build -t nodejs:web .
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 30.12MB
Step 1/11 : FROM node:16-alpine as build
--> 2573171e0124
Step 2/11 : WORKDIR /app
--> Using cache
--> e7ec186a32a2
Step 3/11 : COPY package.json .
--> Using cache
--> 3843e06bea2e
Step 4/11 : RUN npm install
--> Using cache
--> 4b099804d6bc
Step 5/11 : COPY . .
--> Using cache
--> 7043a550bdd8
```

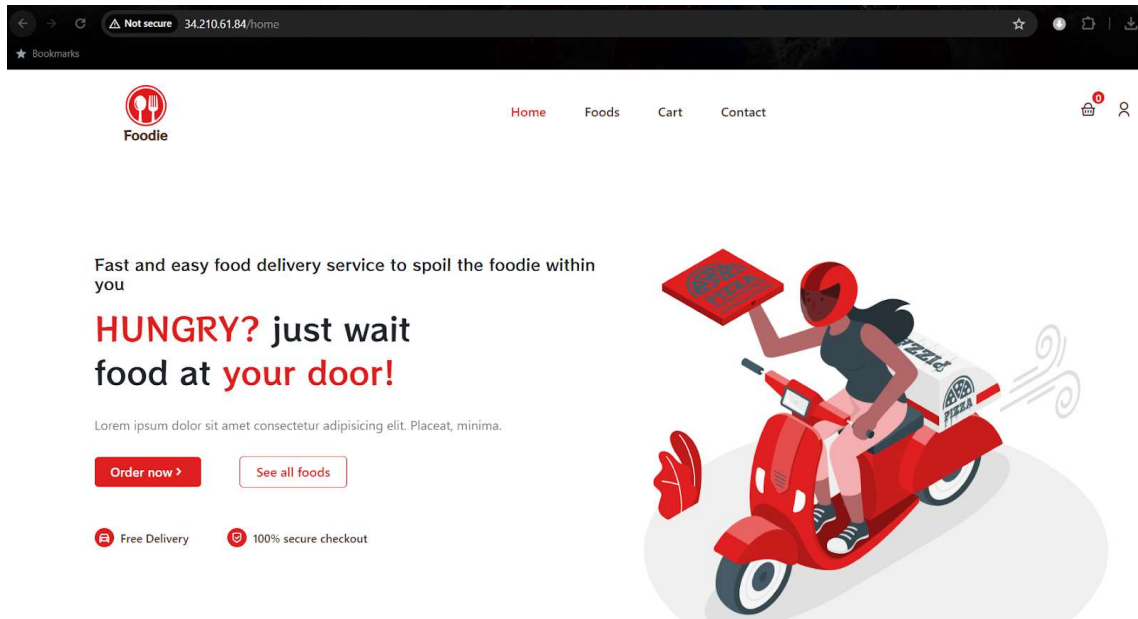
➔ Checking whether image is created or not by using **docker images** command:

```
Step 11/11 : CMD ["nginx", "-g", "daemon off;"]
--> Running in bd93527d571f
Removing intermediate container bd93527d571f
--> 5d27baeac2a8
Successfully built 5d27baeac2a8
Successfully tagged nodejs:web
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
nodejs        web       5d27baeac2a8   About a minute ago   60MB
<none>        <none>    52ada153227e   2 minutes ago       624MB
nginx         alpine    01e5c69afaf6   7 days ago         42.6MB
node          16-alpine 2573171e0124   3 months ago        118MB
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app#
```

➔ checking the image by running container from the created image by using **docker run** command:

```
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# docker run -d -it -p 80:80 nodejs:web
f4489dbb089ac10ade091523636bbccf0bb31f159a632beacdc5f53016484cf
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
f4489dbb089a   nodejs:web "/docker-entrypoint..." 4 seconds ago  Up 3 seconds  0.0.0.0:80->80/tcp, :::80->80/tcp
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app#
```

➔ checking the browser output:



The container is working fine:

→ creating a docker-compose file for deployment purpose:

docker-compose.yml file contains:

```
version: '3'
services:
  nodejs-app:
    image: nodejs:web
    container_name: nodejs
    ports:
      - 80:80
    volumes:
      - nodejs-vol:/usr/share/nginx/html/
volumes:
  nodejs-vol:
    external: true
```

→ creating a docker volume: by using **docker volume create** command:

```

root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# docker volume create nodejs-vol
nodejs-vol
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# docker volume ls
DRIVER      VOLUME NAME
local       nodejs-vol
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# vi docker-compose.yml
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app#

```

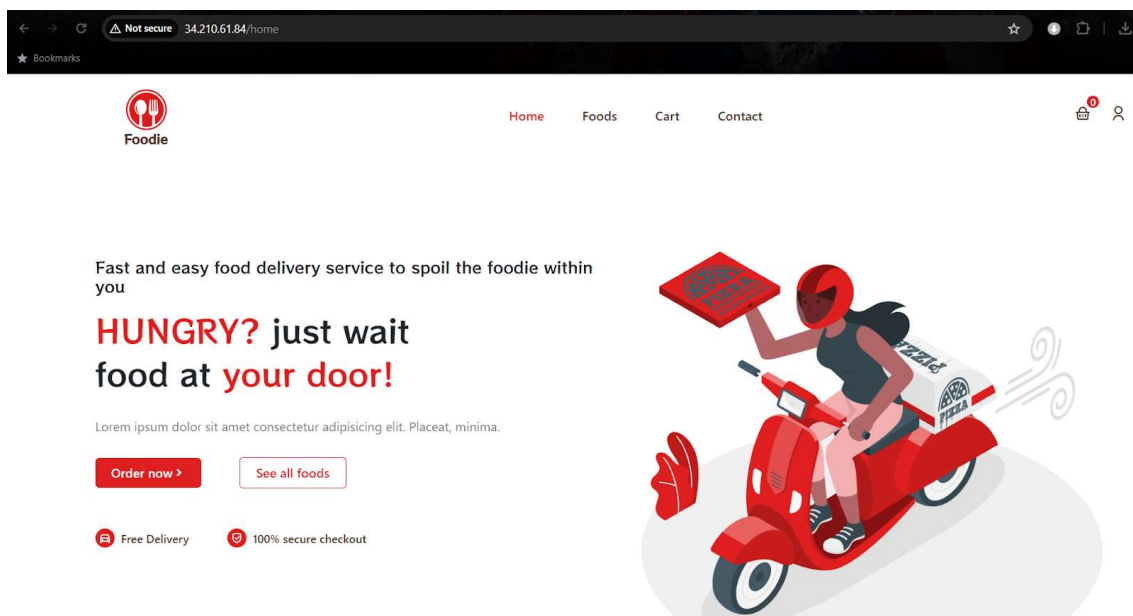
➔ Deploying the application through docker-compose:

```

root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# docker-compose up -d
Creating network "food-delivery-react-redux-app_default" with the default driver
Creating nodejs ... done
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# docker-compose ps
Name                Command                  State      Ports
-----
nodejs              /docker-entrypoint.sh ngin ... Up         0.0.0.0:80->80/tcp,:::80->80/tcp
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app#

```

Browser output:



Step:4 - Creating Build and Deployment scripts:

- ➔ Setting up Docker hub credentials as environmental variables:
- ➔ Creating build.sh for building docker image:

Build.sh contains:

```
#!/bin/bash
#login into dockerhub:
docker login -u $DOCKER_USER -p $DOCKER_PASS

#building the image:
docker build -t nodejs:web .
docker images

#stopping the already running container:
docker stop nodejs
docker rm nodejs
```

➔ Changing the file permission and executing it:

```
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# vi build.sh
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# chmod +x build.sh
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# ./build.sh
```

```
Step 10/11 : EXPOSE 80
---> Using cache
---> fc7e4dd06771
Step 11/11 : CMD ["nginx", "-g", "daemon off;"]
---> Using cache
---> 5d27baeac2a8
Successfully built 5d27baeac2a8
Successfully tagged nodejs:web
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<none>	<none>	98b8e5e447c0	Less than a second ago	624MB
nodejs	web	5d27baeac2a8	14 minutes ago	60MB
<none>	<none>	52ada153227e	16 minutes ago	624MB
nginx	alpine	01e5c69afaf6	7 days ago	42.6MB
node	16-alpine	2573171e0124	4 months ago	118MB

```
nodejs
nodejs
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app#
```

The script is working fine:

➔ Creating a Deploy.sh script for deployment of the container through docker-compose:

Deploy.sh file contains:

```
#!/bin/bash
```



```
#deploying the container:
docker-compose up -d

#tagging the image:
docker tag nodejs:web ravivarman46/nodejs:cicd

#pushing an image to dockerhub:
docker push ravivarman46/nodejs:cicd
echo "the image has been pushed to docker hub"
```

➔ Changing the file permission and executing it:

```
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# vi deploy.sh
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# chmod +x d
deploy.sh      docker-compose.yml  dockerfile
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# chmod +x deploy.sh
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# ./deploy.sh
Creating nodejs ... done
The push refers to repository [docker.io/ravivarman46/nodejs]
f7013e16d4b2: Pushed
a34b395c0ca3: Layer already exists
5e728486380e: Layer already exists
b968c967e155: Layer already exists
92ef9174e989: Layer already exists
28c7b3c0b176: Layer already exists
fbed1f6990ee: Layer already exists
dd731ddf52be: Layer already exists
9fe9a137fd00: Layer already exists
cicd: digest: sha256:bad153ed58e0563e78a323a437f22404a437cf06a48712c12feda33004d418d9 size: 2201
the image has been pushed to docker hub
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# █
```

Deploy.sh is working is fine:

Step:5 – Creating a Jenkinsfile for Jenkins job: pipeline (optional)

Jenkinsfile contains:

```
pipeline {
    agent any

    stages {
        stage('Login into DockerHub') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'docker',
                    passwordVariable: 'DOCKER_PASSWORD', usernameVariable:
                    'DOCKER_USERNAME')]) {
```

```

        sh "echo \${DOCKER_PASSWORD} | docker
login -u \${DOCKER_USERNAME} --password-stdin docker.io"
    }
}

stage('Changing the File Permission') {
    steps {
        sh 'chmod +x build.sh'
        sh 'chmod +x deploy.sh'
    }
}

stage('Executing the File') {
    steps {
        sh './build.sh'
        sh './deploy.sh'
    }
}
}
}
}

```

Step:6 – Creating a GitHub repository and pushing all the files:

- ➔ Creating a new repository on GitHub:

- ➔ Copying the https code and cloning it on command line:

- ➔ Copying all the files to the cloned directory and pushing it to the GitHub repository:


```

root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# mv * nodejs-food-deployment-with-docker/
mv: cannot move 'nodejs-food-deployment-with-docker' to a subdirectory of itself, 'nodejs-food-deployment-wit
oyment-with-docker'
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# ls
nodejs-food-deployment-with-docker
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# ls nodejs-food-deployment-with-docker/
README.md build.sh deploy.sh docker-compose.yml dockerfile package-lock.json package.json public src
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app#


```

```

root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app/nodejs-food-deployment-with-docker# git push origin main
Username for 'https://github.com': Ravivarman16
Password for 'https://Ravivarman16@github.com':
Enumerating objects: 118, done.
Counting objects: 100% (118/118), done.
Delta compression using up to 2 threads
Compressing objects: 100% (108/108), done.
Writing objects: 100% (118/118), 13.63 MiB | 15.13 MiB/s, done.
Total 118 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/Ravivarman16/nodejs-food-deployment-with-docker.git
 * [new branch]    main -> main
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app/nodejs-food-deployment-with-docker#

```

➔ Checking the GitHub repository:

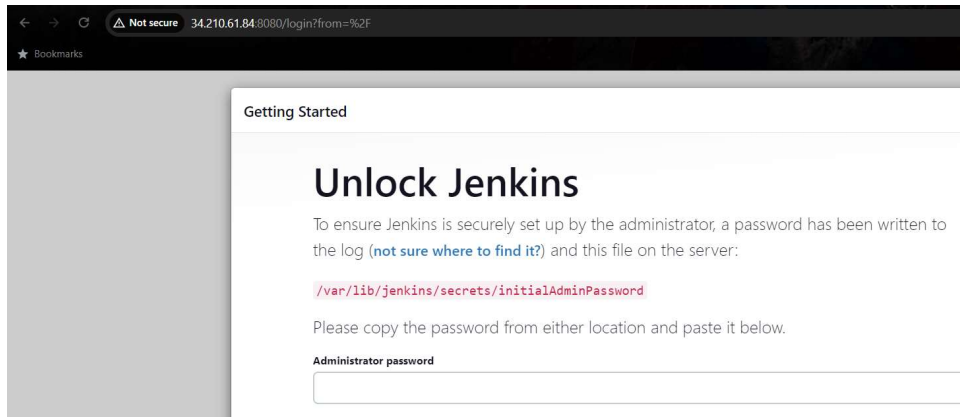

nodejs-food-deployment-with-docker
Public
Pin
Unwatch 1

main
1 branch
0 tags
Go to file
Add file
Code

root nodejs files		684e5c2 1 minute ago	1 commit
public	nodejs files	1 minute ago	
src	nodejs files	1 minute ago	
README.md	nodejs files	1 minute ago	
build.sh	nodejs files	1 minute ago	
deploy.sh	nodejs files	1 minute ago	
docker-compose.yml	nodejs files	1 minute ago	
dockerfile	nodejs files	1 minute ago	
package-lock.json	nodejs files	1 minute ago	
package.json	nodejs files	1 minute ago	

Step:7 – Setting up Jenkins dashboard:

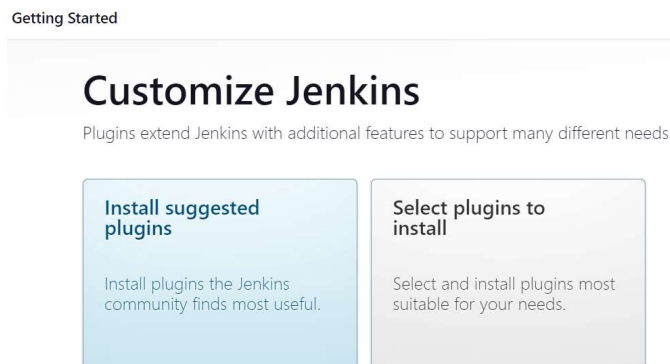
- ➔ Copy the public ip address of the instance along with the port number 8080 on the browser:



➔ Just copy the path paste it on the command line to get the password and paste it there and click continue:

```
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app# cat /var/lib/jenkins/secrets/initialAdminPassword
ccf291decc85474eb472b396deb39180
root@ip-172-31-20-19:/home/ubuntu/food-delivery-react-redux-app#
```

➔ Then select install suggested plugins option, it will start installing plugins:



Getting Started

<input checked="" type="checkbox"/> Folders	<input checked="" type="checkbox"/> OWASP Markup Formatter	<input type="checkbox"/> Build Timeout	<input type="checkbox"/> Credentials Binding	** Ionicons API
<input checked="" type="checkbox"/> Timestampers	<input checked="" type="checkbox"/> Workspace Cleanup	<input type="checkbox"/> Ant	<input checked="" type="checkbox"/> Gradle	Folders
<input checked="" type="checkbox"/> Pipeline	<input type="checkbox"/> GitHub Branch Source	<input checked="" type="checkbox"/> Pipeline: GitHub Groovy Libraries	<input checked="" type="checkbox"/> Pipeline: Stage View	OWASP Markup Formatter
<input type="checkbox"/> Git	<input type="checkbox"/> SSH Build Agents	<input checked="" type="checkbox"/> Matrix Authorization Strategy	<input type="checkbox"/> PAM Authentication	
<input type="checkbox"/> LDAP	<input checked="" type="checkbox"/> Email Extension	<input type="checkbox"/> Mailer		

➔ Then we can able to see credentials dashboard just setup the credentials for Jenkins, click save and continue:

Create First Admin User

Username

Password

Confirm password

Full name

➔ Then click save and finish:

Instance Configuration

Jenkins URL:

http://34.210.61.84:8080/

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.426.1

Not now

Save and Finish

➔ Then click start using Jenkins option to see the Jenkins dashboard:

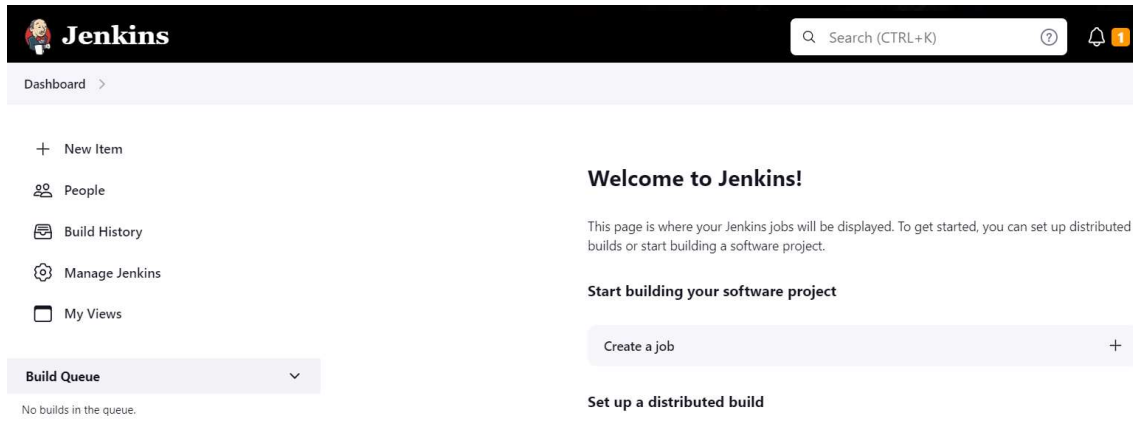
Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

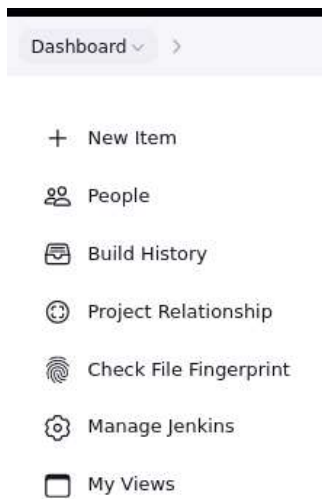
Start using Jenkins

➔ We can able to the Jenkins dashboard:



Step:8 – Setting Docker Hub credentials in Jenkins: (optional)

➔ On Jenkins dashboard, we can able to see manage Jenkins, click that one:



➔ We can able to see credentials, click that one:

Security



Security

Secure Jenkins; define who is allowed to access/use the system.



Credentials

Configure credentials



Credential Providers

Configure the credential providers and types

➔ Then click global one to add credentials:

Dashboard > Manage Jenkins > Credentials

Credentials

T	P	Store ↓	Domain	ID	Name
---	---	---------	--------	----	------

Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

Icon: S M L

➔ Then click add credentials:

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

Global credentials (unrestricted)

[+ Add Credentials](#)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
This credential domain is empty. How about adding some credentials?			

➔ Then give the docker hub username:

Dashboard > Manage Jenkins > Credentials

New credentials

Kind

Username with password

Scope ?

Global (Jenkins, nodes, ite

Username ?

ravivarman46

➔ Then give the Docker hub token, give the id as docker: click create:

The screenshot shows the 'Global credentials (unrestricted)' form in Jenkins. It has three input fields: 'Password' (empty), 'ID' (containing 'docker'), and 'Description' (containing 'docker'). Each field has a question mark icon to its right. Below the fields is a blue 'Create' button.

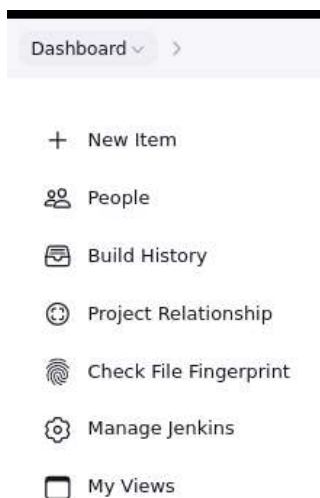
➔ The credentials have been created successfully:

The screenshot shows the 'Global credentials (unrestricted)' page in Jenkins. At the top, there is a breadcrumb trail: Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >. Below this is a blue button labeled '+ Add Credentials'. The main content area shows a table of credentials. The table has four columns: ID, Name, Kind, and Description. There is one credential listed with ID 'docker', Name 'ravivarman46/***** (docker)', Kind 'Username with password', and Description 'docker'. A small icon of a key is visible in the bottom right corner of the table.

ID	Name	Kind	Description
docker	ravivarman46/***** (docker)	Username with password	docker

Step:9 – Setting up Environmental variables:

➔ On Jenkins dashboard we able to see manage Jenkins, click that one:



➔ Then click system: under global properties, select environment variables:
set the variables: click save and apply

The screenshot shows the 'Global properties' configuration page in Jenkins. On the left, under 'Global properties', the 'Environment variables' checkbox is checked. Below it, a 'List of variables' section shows a table with one entry: 'Name' is 'DOCKER_PASS' and 'Value' is 'DOCKER_USER'. On the right, a detailed view of a variable shows 'Name' as 'DOCKER_USER' and 'Value' as 'ravivarman46'. At the bottom right, there are 'Save' and 'Apply' buttons.

Name	Value
DOCKER_PASS	DOCKER_USER

Step:10 – Creating a Jenkins job:

➔ On the Jenkins dashboard, we can able to see new item click that one:


The screenshot shows the Jenkins dashboard. At the top, there is a 'Dashboard >' breadcrumb. Below it, there is a list of links: '+ New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. The '+ New Item' link is highlighted with a light blue background.


➔ Name the job and select **freestyle project** as job type:


Enter an item name

nodejs-deployment

» Required field

**Freestyle project**
This is the central feature of Jenkins and can be even used for something other than building code.

**Pipeline**
Orchestrates long-running activities (as workflows) and/or organizing complex operations.


**Multi-configuration project**
Suitable for projects that need a large number of platform-specific builds, etc.


OK


➔ Give the description according to your preferences:

Dashboard > nodejs-deployment > Configuration

Configure

**General**

 Source Code Management

 Build Triggers

General

Description

nodejs-deployment

➔ Then under source-code select Git, enter your GitHub repository link:

Dashboard > nodejs-deployment > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?

`https://github.com/Ravivarman16/nodejs-food-deployment-with-docker.git`

Credentials ?

→ Then enter the branch name correctly:

Dashboard > nodejs-deployment > Configuration

Configure

- General
- Source Code Management
- Build Triggers

Source Code Management

Branches to build ?

Branch Specifier (blank for 'any') ?

`*/main`

Add Branch

→ Then under build triggers, select GitHub hook trigger for GITScm polling option:

Dashboard > nodejs-deployment > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment

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 ?

➔ Then under build steps, select execute shell option: enter the steps to be done:

```
chmod +x build.sh
```

```
chmod +x deploy.sh
```

```
./build.sh
```

```
./deploy.sh
```

```
echo "-----"
```

```
echo "Pipeline is build successfully"
```

The screenshot shows the Jenkins Configuration page for a job named 'nodejs-deployment'. The breadcrumb trail is 'Dashboard > nodejs-deployment > Configuration'. On the left, the 'Configure' section has a sidebar with options: General, Source Code Management, Build Triggers, Build Environment, **Build Steps** (selected), and Post-build Actions. The main area is titled 'Build Steps' and contains a single step named 'Execute shell'. The 'Command' field for this step contains the following shell script:

```
chmod +x build.sh
chmod +x deploy.sh
./build.sh
./deploy.sh
echo "-----"
```

Below the command field, there is a link that says 'See the list of available environment variables'.

➔ Then click apply & save:

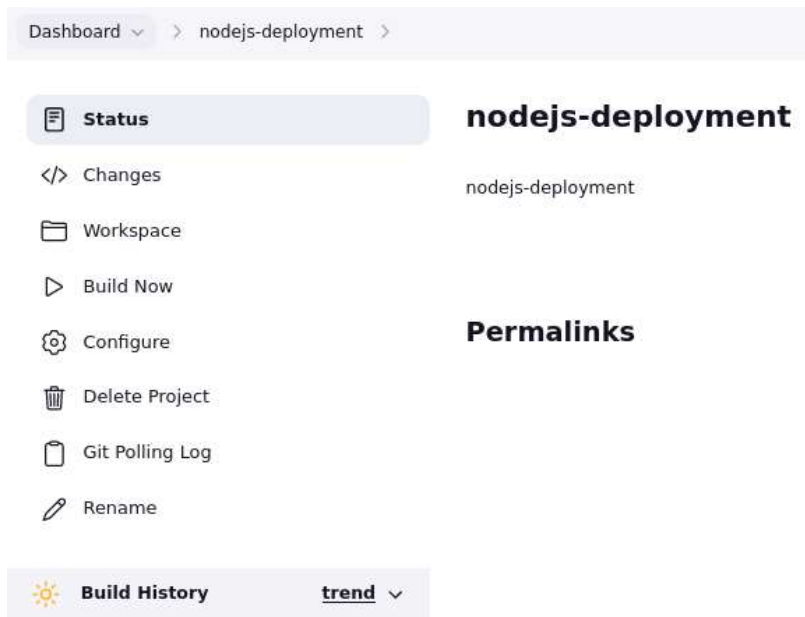
Post-build Actions

Add post-build action ▾

Save

Apply

➔ Jenkins freestyle job is ready:



➔ Then click build now option to start the freestyle job: we can able to see job got initiated successfully:



We can able to see job started to run:

➔ Then click that one we can see output of the job:

Dashboard > nodejs-deployment > #3

Status

</> Changes

Console Output

Edit Build Information

Delete build '#3'

Git Build Data

Previous Build

✓ #3 (Dec 9, 2023, 11:18:20 PM)

No changes.

Started by user [jenkins](#)

Revision: a0eb28644789d6e854a3733691dbad57d3f2ad7e
Repository: <https://github.com/Ravivarman16/nodejs-food-deployment-with-docker.git>
• refs/remotes/origin/main

➔ Then click console output to know more details of this job:

Dashboard > nodejs-deployment > #3 > Console Output

Status

</> Changes

Console Output

View as plain text

Edit Build Information

Delete build '#3'

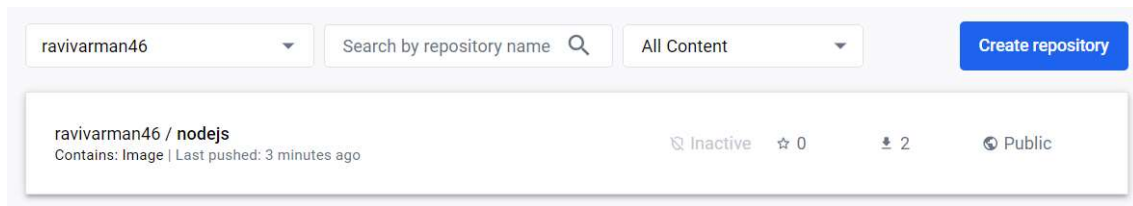
Git Build Data

✓ Console Output

Started by user [jenkins](#)
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/nodejs-deployment
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/nodejs-deployment/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url <https://github.com/Ravivarman16/nodejs-food-deployment-with-docker.git>
Fetching upstream changes from <https://github.com/Ravivarman16/nodejs-food-deployment-with-docker.git>
> git --version # timeout=10

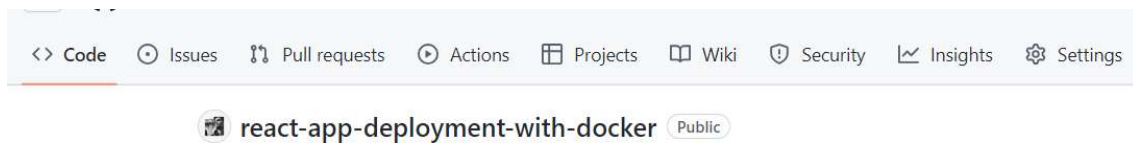
```
fbed1f6990ee: Layer already exists
28c7b3c0b176: Layer already exists
dd731ddf52be: Layer already exists
9fe9a137fd00: Layer already exists
cicd: digest: sha256:2c146fefbc3bb04b1554422fd86bd374a6ca7ecf5d202086ce25dfdb8bd06d5c size: 2201
the image has been pushed to docker hub
+ echo -----
+ echo Pipeline is build successfully
Pipeline is build successfully
Finished: SUCCESS
```

Docker Hub output:

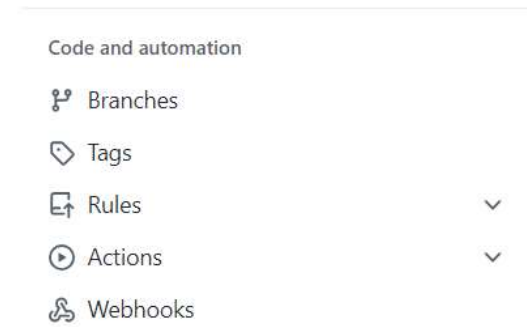


Step:11 – Making the pipeline automated by GitHub hook Trigger:

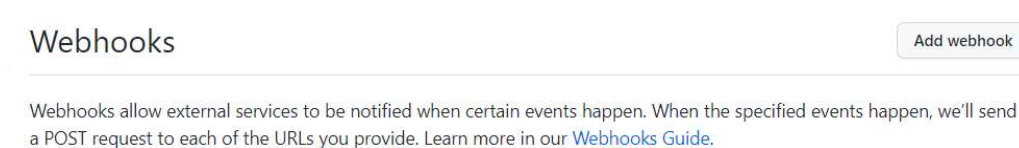
➔ On the GitHub repository, we can able to see settings, click that one:



➔ Then on the left side, we can able to see webhooks, click that one:



➔ Then we can able to see add webhook option, click that one:



➔ Then under payload URL, enter the Jenkins URL along with github-webhook:

Webhooks / Add webhook

We'll send a POST request to the URL below with detail: format you'd like to receive (JSON, x-www-form-urlencoded [documentation](#)).

Payload URL *

`http://34.210.61.84:8080/github-webhook/`

➔ Then select the event and click add webhooks:

Which events would you like to trigger this webhook?

- ☒ Just the push event.
- ☐ Send me **everything**.
- ☐ Let me select individual events.

☒ **Active**

We will deliver event details when this hook is triggered.

Add webhook

➔ We can able to see webhook trigger is set perfectly:

Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

✓ `http://34.210.61.84:8080/github-we... (push)`

Edit

Delete

➔ Checking it by making changes in GitHub:

Commit changes

Commit message

Update Jenkinsfile

Extended description

Add an optional extended description..

☒ Commit directly to the main branch

☐ Create a new branch for this commit and start a pull request

[Learn more about pull requests](#)

Cancel

Commit changes

➔ We can able to see the Jenkins job got triggered automatically:

Filter builds...

#4

(pending—In the quiet period. Expires in 1.5 sec)

#3

Dec 9, 2023, 11:18 PM

#2

Dec 9, 2023, 11:16 PM

Atom feed for all

Atom feed for failures

Console output:

Dashboard > nodejs-deployment > #4

Status

Changes

Console Output

Edit Build Information

Polling Log

Git Build Data

Previous Build

#4 (Dec 9, 2023, 11:24:08 PM)

Progress:

Keep this build forever

Add description

Started 39 sec ago
Build has been executing for 39 sec

Changes

1. Update Jenkinsfile (details / githubweb)

Started by an SCM change

Revision: d4c61ae98d317e2faf204da7e1941922b463cf79
Repository: https://github.com/Ravivarmaan16/nodejs-food-deployment-with-docker.git
• refs/remotes/origin/main

Benefits of above task:

- ➔ **Efficiency and Automation:** Automation through Jenkins enables the continuous integration and deployment of the Node.js application, reducing manual intervention and minimizing the risk of human errors.

Docker containers provide a consistent and isolated environment for the application, ensuring that it runs reliably across different stages of development and deployment.

- ➔ **Scalability and Resource Optimization:** Leveraging Docker allows for efficient scaling of the application by deploying multiple instances of the Docker container. This ensures optimal resource utilization and the ability to handle varying workloads effectively.

EC2 instances in AWS provide scalability by allowing the organization to adjust the compute capacity based on demand, ensuring the application can scale horizontally to meet changing performance requirements.

- ➔ **Cost-Effective Hosting and Resource Management:** Docker's lightweight containers reduce infrastructure costs by optimizing resource consumption, enabling more efficient utilization of server resources.

AWS EC2 instances can be configured and scaled based on actual usage, helping to manage costs effectively. Additionally, the use of Jenkins for automation contributes to cost savings by streamlining the development and deployment processes.

All the files for the above task have been uploaded under this GitHub repository:

<https://github.com/yasminjeelani/simple-fooddeliveryapp-deployment.git>