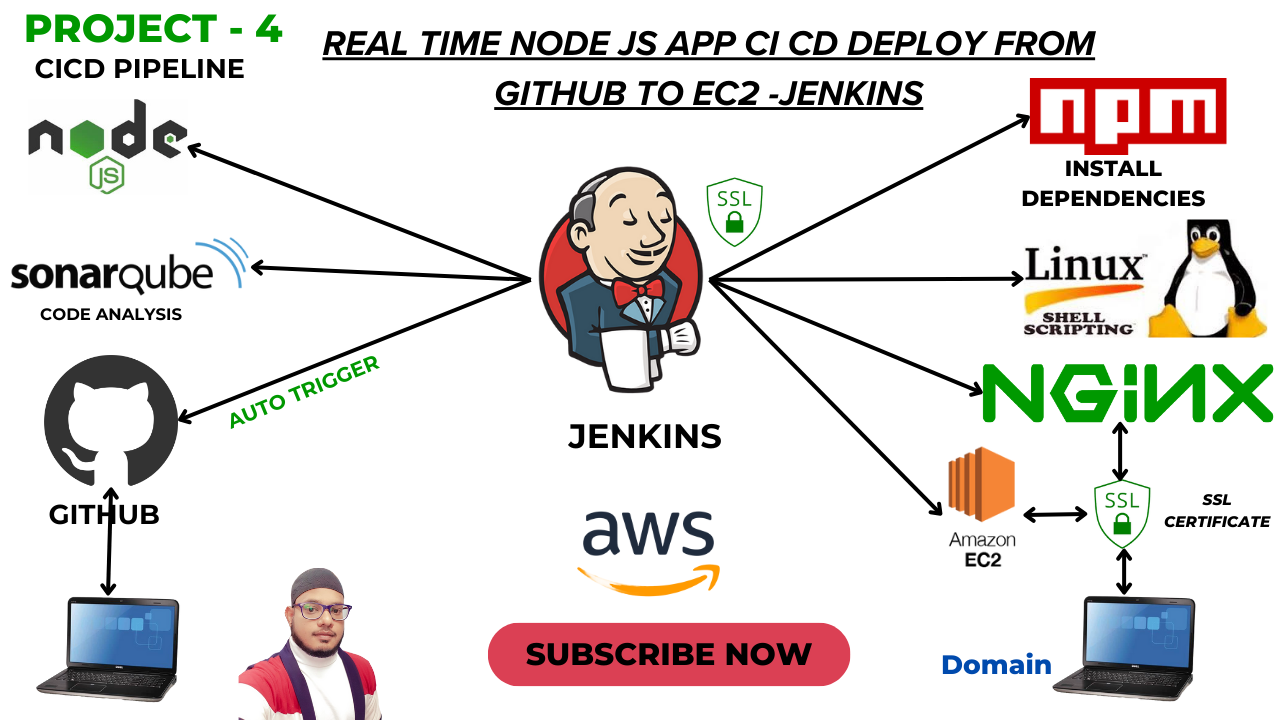
Real-time NODE.js App deployment withPM2 , Shell script, Jenkins, SonarQube ,Github ,Domain SSL cert

Project - 5



**TABLE OF CONTENT**

1. [**Install NodeJS and NPM using nvm**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#install-nodejs-and-npm-using-nvm)
   * [**1.1 Update and Install Dependencies**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#11-update-and-install-dependencies)
   * [**1.2 Configure NodeSource**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#12-configure-nodesource)
   * [**1.3 Verify NodeJS Installation**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#13-verify-nodejs-installation)
2. [**Install PM2 globally**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#install-pm2-globally)
3. [**Start your app with PM2**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#start-your-app-with-pm2)
   * [**3.1 Update ecosystem.config.js**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#31-update-ecosystemconfigjs)
4. [**Jenkins Installation on Ubuntu**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#jenkins-installation-on-ubuntu)
   * [**4.1 Jenkins Script Setup**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#41-jenkins-script-setup)
   * [**4.2 Access Jenkins**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#42-access-jenkins)
   * [**4.3 Configure Jenkins Pipeline**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#43-configure-jenkins-pipeline)
5. [**Nginx Installation on Ubuntu**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#nginx-installation-on-ubuntu)
   * [**5.1 Nginx Configuration for Jenkins**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#51-nginx-configuration-for-jenkins)
   * [**5.2 SSL Certificate Setup (Optional)**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#52-ssl-certificate-setup-optional)
6. [**Deploy Script**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#deploy-script)
7. [**Jenkins User Permissions**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#jenkins-user-permissions)
   * [**7.1 Grant sudo Permissions**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#71-grant-sudo-permissions)
   * [**7.2 Restart Jenkins**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#72-restart-jenkins)
8. [**Jenkins User Deployment**](https://chat.openai.com/c/92acc6fb-f857-4927-9194-e4d9a9ef93fb#jenkins-user-deployment)

**Installation Steps:**

* 1. Install NodeJS and NPM using nvm

Follow this Medium blog to install Node.js & npm on EC2 server. Link below

<https://medium.com/@mohdaseemakram19/node-js-introduction-installation-of-node-js-on-aws-ec2-2350b7b95581>

sudo apt-get update

sudo apt-get install -y ca-certificates curl gnupg sudo mkdir -p /etc/apt/keyrings curl -fsSL https://deb.nodesource.com/gpgkey/nodesource-repo.gpg.key | sudo gpg --dearmor -o /etc/apt/keyrings/nodesource.gpg

NODE\_MAJOR=20 echo "deb [signed-by=/etc/apt/keyrings/nodesource.gpg] https://deb.nodesource.com/node\_$NODE\_MAJOR.x nodistro main" | sudo tee /etc/apt/sources.list.d/nodesource.list

sudo apt-get update

sudo apt-get install nodejs -y

Check Installation of nodejs packages

node -v

v18.18.2

npm -v

9.8.1

* 2. Install PM2 globally:

npm install -g pm2

* 3. Navigate to your Node.js app directory: cd /var/project/node-js-docker-cicd

Install app dependencies: npm install

nano ecosystem.config.js

module.exports = {

apps: [

{

name: 'nodeapp', // Updated app name to nodeapp

script: 'index.js', // Updated script name to index.js

// instances: 1,

// autorestart: false,

// watch: false,

env: {

NODE\_ENV: 'production',

PORT: 3000 // Replace with your desired port

}

}

]

};

* Start your app with PM2:

pm2 start ecosystem.config.js

pm2 restart ecosystem.config.js

pm2 stop ecosystem.config.js

If your package.json file contains the specified dependencies, and you want to combine the installation of dependencies and the deployment script, you can update the "deploy" script in your package.json as follows

"scripts": {

"start": "node index.js",

"deploy": "npm install && pm2 start ecosystem.config.js"

},

"dependencies": {

"express": "^4.18.2"

}

* 4. Jenkins installation on ubuntu

nano jenkins.sh

link for Jenkins script [Linux (jenkins.io)](https://www.jenkins.io/doc/book/installing/linux/#debianubuntu)

sh jenkins.sh

chmod u+x

chmod 777 jenkins.sh

access with ip:8080

assign jenkins.domain

create a pipeline freestyle project.

jenkins password to login

admin

admin123

* 5.Nginx installation on ubuntu

Follow my YouTube video to install NginX & reverse proxy to Jenkins

[(1) JENKINS Reverse Proxy to Domain | NginX Webserver | Domain with SSL Certificate Lets-encrypt | AWS - YouTube](https://www.youtube.com/watch?v=lAWLXA2zL6s&list=UULF8EVuF-XAXwsMr3USVcMrPg&index=7)

To install Nginx on Ubuntu, you can use the following steps:

1. Update the package list to ensure you have the latest information about available packages:

sudo apt update

1. Install Nginx:

sudo apt install nginx

1. Start the Nginx service:

sudo systemctl start nginx

1. Enable Nginx to start on boot:

sudo systemctl enable nginx

Now, Nginx should be installed and running on your Ubuntu system. You can check the status to ensure it's active:

sudo systemctl status nginx

**6. Create a Jenkins Nginx Configuration File:**

Create a new Nginx configuration file for Jenkins. For example, create a file named **/etc/nginx/sites-available/jenkins**:

sudo nano /etc/nginx/sites-available/jenkins

Add the following configuration. Make sure to replace **your\_domain** with your actual domain or subdomain:

server {

server\_name jenkins.xyz;

location / {

proxy\_pass http://127.0.0.1:8080;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

# Error handling

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

}

Save the file and exit the editor.

1. **Create a Symbolic Link:**

Create a symbolic link to enable the configuration:

sudo ln -s /etc/nginx/sites-available/jenkins /etc/nginx/sites-enabled/

1. **Test Nginx Configuration:**

Check the Nginx configuration for syntax errors:

sudo nginx -t

If there are no errors, restart Nginx:

sudo systemctl restart nginx

* 7. **SSL Certificate for Nginx (Optional)**

To obtain an SSL certificate for a subdomain using Certbot, you can follow these general steps. These instructions assume you have Nginx already installed and configured.

1. **Install Certbot:**

sudo apt update

sudo apt install certbot python3-certbot-nginx

1. **Obtain the SSL Certificate:**

Replace **your\_subdomain** and **your\_domain** with your actual subdomain and domain.

sudo certbot --nginx -d jenkins.aseemcloudtech.com

sudo certbot --nginx -d app.aseemcloudtech.com

Follow the prompts to configure Certbot. Certbot will automatically modify your Nginx configuration to enable SSL and reload Nginx.

1. **Automate Certificate Renewal:**

Certificates obtained through Certbot are typically valid for 90 days. To automate the renewal process, add a cron job:

sudo crontab -e

Add the following line to run the renewal check twice daily:

0 0,12 \* \* \* certbot renew --quiet

Save and exit the editor.

8.1 Create jenkins freestyle pipeline

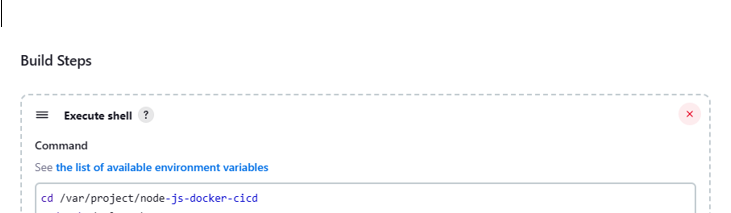
Name as nodeapp

Git – url of public repo

https://github.com/Aseemakram19/node-js-docker-cicd.git

Branch – dev

Test the pipeline.



8.2 Manual Code deploy

follow readme document to deploy the application

8.3 Access through url : ip:3000

Successfully

* **8.4 Deploy Script**

nano deploy.sh

#!/bin/bash

# Navigate to your project jenkins/workspace/ directory

cd /var/lib/jenkins/workspace/nodeapp

# Copy files using rsync

cp -r \* /var/nodeapp/node-js-docker-cicd

# Navigate to your project directory

cd /var/nodeapp/node-js-docker-cicd

# Find the process ID (PID) using port 3000

PID=$(lsof -t -i:3000)

# If the PID is not empty, kill the process

if [ -n "$PID" ]; then

su -c "kill -9 $PID" -s /bin/bash root

fi

# Navigate to your project directory

cd /var/nodeapp/node-js-docker-cicd

# stop pm2 process

pm2 stop ecosystem.config.js

# install node dependency

npm install

# start pm2 process

pm2 start ecosystem.config.js

# Find the process ID (PID) using port 3000

PID=$(lsof -t -i:3000)

# You may want to check if the process is still running after the deploy

if [ -n "$PID" ]; then

echo "The process is still running with PID $PID."

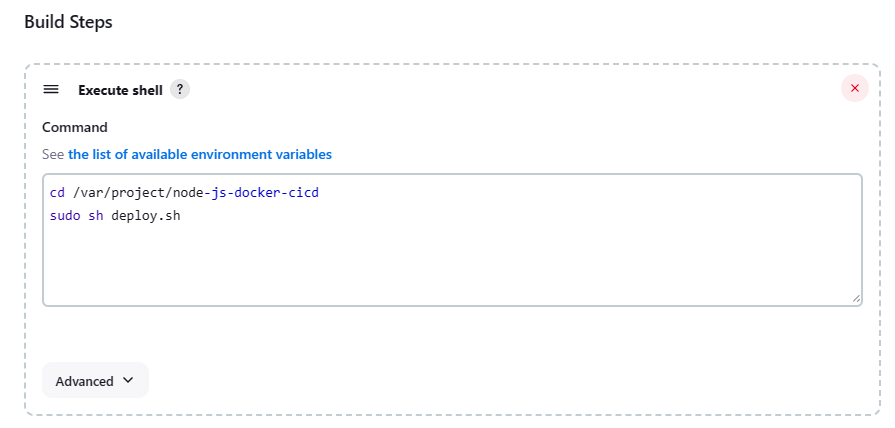
else

echo "Deployment successful. process running on port 3000."

fi

* **9. JENKINS PIPELINES :**

create a Pipeline & try to execute this shell command from Jenkins .



* **10. Jenkins User Permissions**
* sudo -i -u jenkins

sudo -i -u jenkins

sudo ./deploy.sh

Give Jenkins User Appropriate Permissions: Ensure that the Jenkins user on your EC2 instance has the necessary permissions to perform the deployment actions.

• For security reasons, create a specific user on your EC2 instance for Jenkins, and grant it the necessary permissions to execute the deployment script.

• Use sudo visudo to edit the sudoers file and allow the Jenkins user to run specific commands without a password prompt.

# Allow members of group sudo to execute any command

%sudo ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "@include" directives:

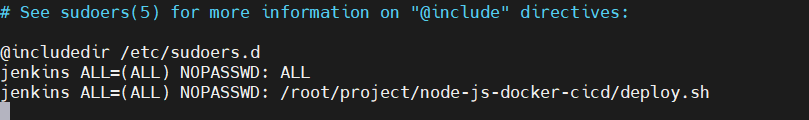
Example sudoers entry:

open as rootuser

sudo visudo

jenkins ALL=(ALL) NOPASSWD: ALL

jenkins ALL=(ALL) NOPASSWD: /var/nodeapp/node-js-docker-cicd/deploy.sh



visudo

@includedir /etc/sudoers.d

jenkins ALL=(ALL) NOPASSWD: ALL

restart your jenkins service to apply the changes

systemctl restart jenkins

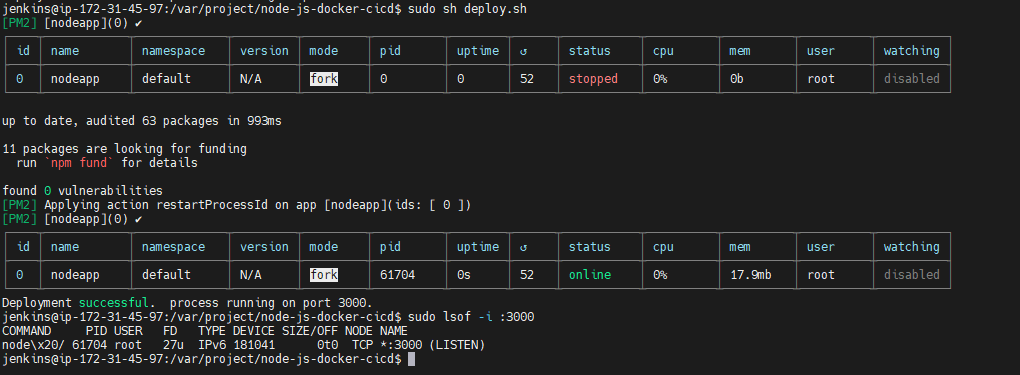
systemctl status jenkins

open new terminal & switch to jenkins user with following command as below :

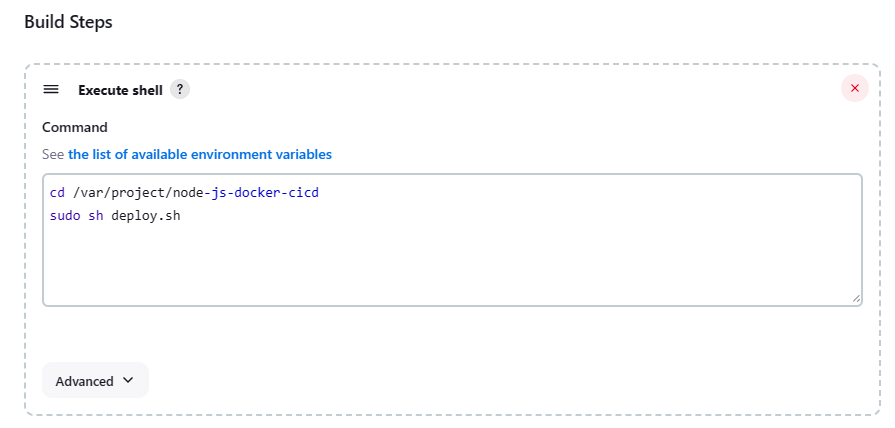
sudo -i -u jenkins

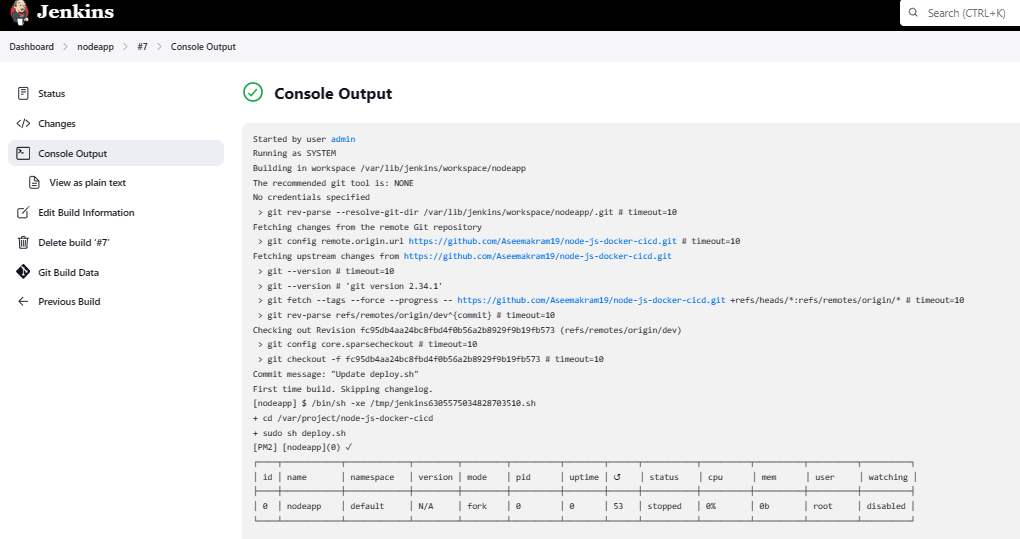
execute the deploy script being the Jenkins user. successfully deployment.

sudo ./deploy.sh







- 

* Nginx configuration for setting up a reverse proxy for a Node.js app running on port 3000:

nginx

server {

listen 80;

server\_name your\_domain;

location / {

proxy\_pass http://127.0.0.1:3000;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

# Additional configuration (if needed)

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

}

Installation steps in aws machine: Download Sonar from http://www.sonarqube.org/downloads/

Ø Docker installation on Ubuntu EC2

1. install docker in ubuntu

sudo apt install docker.io -y

2. docker version add permission to docker to ubuntu

sudo usermod -aG docker ubuntu

sudo usermod -aG docker $USER

refresh the group

newgrp docker

to validate the all the permission is successfully done

docker ps

go to the directory 7 file docker.sock provide permissions to all

sudo chmod 777 /var/run/docker.sock

sudo systemctl restart docker

docker install successfully

Ø After the docker installation, we will create a Sonarqube container (Remember to add 9000 ports in the security group). https://www.sonarsource.com/products/sonarqube/downloads/success-download-community-edition/

Run this command on your EC2 instance to create a SonarQube container:

SonarQube installation on same server with docker images SonarQube - Official Image | Docker Hub

docker run -d --name sonar -p 9000:9000 sonarqube:lts-community

Once Sonar server is up and Running use url to access sonar server dash board. http://IP:9000/ or http://<ip\_addr>:9000/

1. Login the sonar dash board using default username and password. The default username and password is admin/admin.

login with admin & admin default username & password

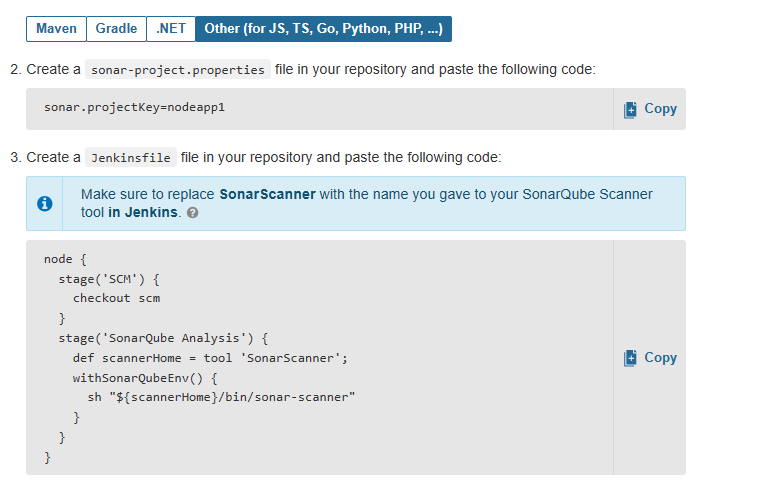
login : admin

sonarqube – aseemadmin

Go to profile- then administrator

real

nodeapp1



sonar.projectKey=nodeapp1

node {

stage('SCM') {

checkout scm

}

stage('SonarQube Analysis') {

def scannerHome = tool 'SonarScanner';

withSonarQubeEnv() {

sh "${scannerHome}/bin/sonar-scanner"

}

}

}

Token for sonar integration : sqa\_b651d4f0a18d066be63401016013d93784a993f4

jenkins file

go to jenkins & install sonar-scanner plugin

1. Goto Jenkins dashboard Jenkins Home à Manage Jenkins à Manage Plugins à Available tab à search with

SonarQube Scanner

sonar-scanner plugin Plugin select install without restart.

2. Jdk Eclipse termurin installer plugin

A. dashboard Jenkins Home à Manage Jenkins à system

JDK17 .

Successfully able to view code analysis in the SonarQube dashboard.