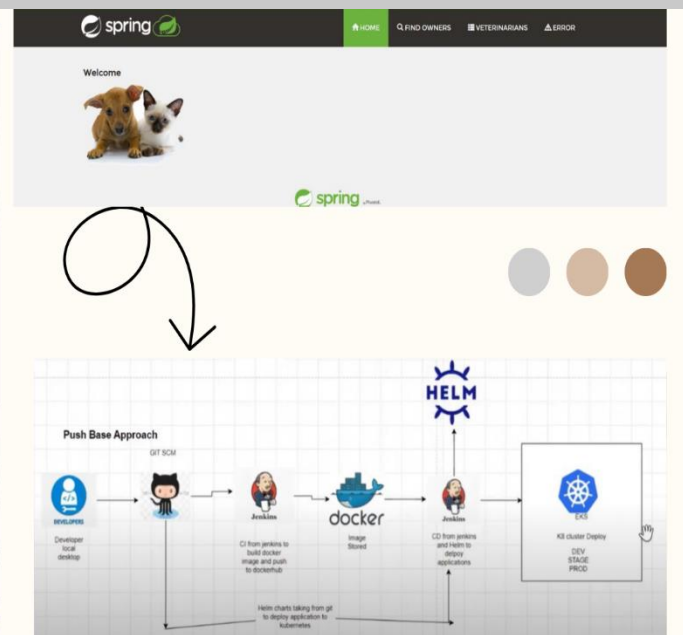


PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



Project

Deployment of Pet-clinic APP



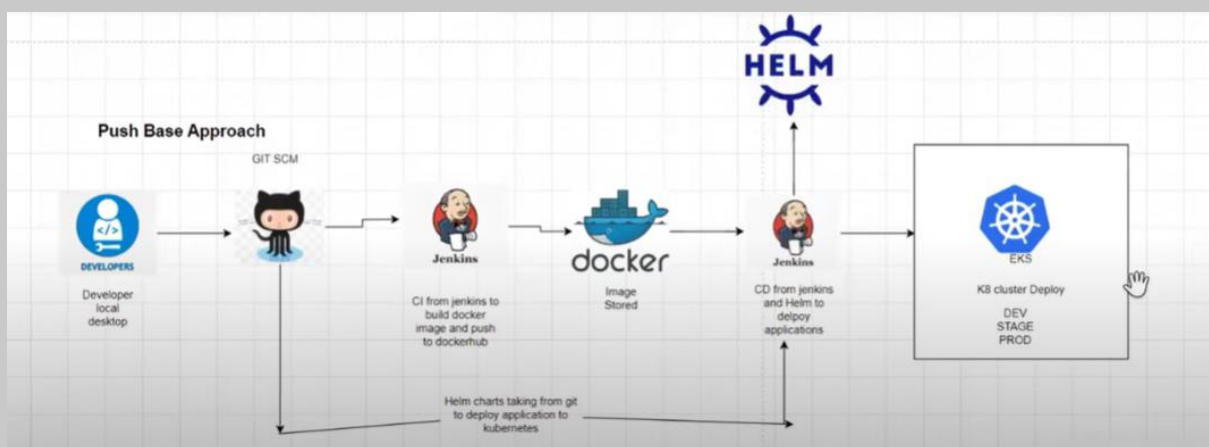
**SPRING-BOOT APPLICATION
DEPLOYMENT
PROJECT "PET-CLINIC"**

Divya Satpute

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



Jenkins on Server Push Based Approach CI/CD



Project content













1. Create Cluster using kubeadm script (step1)
2. How to deploy Jenkins with Our custom domain and SSL (step 2)
3. Step-by-step guide to configure SSL on Jenkins using Let's Encrypt and NGINX reverse proxy (step 3)
4. create webhook on Git-hub (step 4)

Referral GIT-REPO = git clone <https://github.com/divyasatpute/Jenkins.git>

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



Introduction

- **What is CI/CD?** 
 - Continuous Integration (CI) + Continuous Deployment (CD).
 - Automates software delivery process.
- **Server Push Based Approach** 
 - Code changes are automatically pushed to the server.
 - Reduces manual intervention and accelerates deployment.
- **Jenkins Overview** 
 - Popular open-source automation server.
 - Supports building, deploying, and automating projects.
- **How It Works** 
 - **Code Push:** Developers push code to a version control system (e.g., Git) .
 - **Webhook Trigger:** A webhook notifies Jenkins of changes .
 - **Build Process:** Jenkins pulls the latest code and runs builds/tests .
 - **Deployment:** Successful builds are automatically deployed to the server .
- **Benefits** 
 - **Automated Testing:** Ensures code quality through automated tests .
 - **Consistent Releases:** Reduces human error in deployments.
 - **Scalability:** Easily integrates with other tools and systems .
- **Common Tools Used** 
 - **Version Control:** Git
 - **Notification Services:** Slack, email alerts.

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



Prerequisites

you must have your purchase domain and create hosted zone in Route 53 and replace this namespaces with your domain

The screenshot shows the AWS Route 53 console for the hosted zone 'learnwithdivya.online'. The 'Records (6)' tab is selected, displaying a table of DNS records. The first record is an NS record for 'learnwith...' with a value of 'ns-1036.awsdns-01.org'. The other NS records are also listed, and the entire set of NS records is highlighted with a red box. The other records include SOA, A, and CNAME records for various subdomains.

Record	Type	Routing	Differ	Alias	Value/Route traffic to	TTL (s)	Health	Evaluation	Record ID
learnwith...	NS	Simple	-	No	ns-1036.awsdns-01.org ns-1903.awsdns-45.com ns-713.awsdns-25.net ns-199.awsdns-24.com	172800	-	-	-
learnwith...	SOA	Simple	-	No	ns-1036.awsdns-01.org. aws...	900	-	-	-
@.learnwi...	A	Simple	-	No	76.76.21.21	300	-	-	-
_1a3cb51...	CNAME	Simple	-	No	_926f28deb523da50b7e4e9...	300	-	-	-
jenkins.le...	A	Simple	-	No	13.127.199.105	300	-	-	-
www.lear...	CNAME	Simple	-	No	hashnode.network	300	-	-	-

Git clone

`$git clone https://github.com/divyasatpute/Jenkins.git`

push this repo on your github

`$git push <your github repo URL >`

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



Step by Step guide

For that you need to create 4 machine cluster

(step1)

Create Cluster using kubeadm script

In this POC we need 1 master and 2 worker machine and Jenkins machine separately

On AWS Console

- 4 EC2 machine
- ubuntu 20.04 AMI
- volume 40GB (storage)
- c5.xlarge Machine Type

after you have to connect all on gitbash

And update all cluster using following command:

```
$sudo apt update -y
```

And set hostname for each node

```
$sudo hostname Master
```

```
$sudo hostname Worker1
```

```
$sudo hostname Worker2
```

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



Come on root user

\$sudo -i

now on master node paste the following command (EKS
Script) Controlplane script

```
$bash <(curl -s  
https://raw.githubusercontent.com/isakibshaikh1/Kubeadm/main/kubeadm/master.sh)
```

And on another on both Worker node paste following
command (worker script)

```
$bash <(curl -s  
https://raw.githubusercontent.com/isakibshaikh1/Kubeadm/main/kubeadm/worker.sh)
```

```
--2024-07-18 14:07:56-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/11225014/c2055a83-495f-4439-87ad-16e8764054ec?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20240718%2Fus-east-1%2Faws4_request&X-Amz-Date=20240718T140756Z&X-Amz-Expires=300&X-Amz-Signature=bffe274ed8b3c50b65147d867007ff88e0a1f8f53b2da3598c7be3ae42a6f817&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=11225014&response-content-disposition=attachment%3B%20filename%3Detcd-v3.5.1-linux-amd64.tar.gz&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.111.133, 185.199.10.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19399491 (18M) [application/octet-stream]
Saving to: 'etcd-v3.5.1-linux-amd64.tar.gz'

etcd-v3.5.1-linux-amd64.tar.g 100%[=====] 18.50M --.-KB/s in 0.05s

2024-07-18 14:07:56 (386 MB/s) - 'etcd-v3.5.1-linux-amd64.tar.gz' saved [19399491/19399491]

### COMMAND TO ADD A WORKER NODE ###
kubeadm join 172.31.46.89:6443 --token qanxgm.tbmerexbyo3n0dpr --discovery-token-ca-cert-hash sha256:c2b6ac5e46556ed69c940fa8725eb37e6ccbeaa3f39117fc71a481a1798eac3c
root@master:~#
```

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



Note: now your master node is ready take this token from master node and paste on worker node for join worker node to master

as you can see node will be ready but worker node yet not ready state

Note that both nodes are NotReady. This is OK because we have not yet installed networking.

```
root@master:~# kubectl get nodes
NAME      STATUS    ROLES    AGE   VERSION
master    Ready     control-plane  21m   v1.29.2
worker1    Ready     <none>      28s   v1.29.2
worker2    NotReady  <none>      14s   v1.29.2
```

for that we have use some plugins Install a Network Plugin

Doc Ref : [Installing Network Plugin Addons](#)

Install Weave Net : [Weave Net](#)

\$kubectl apply -f

<https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml>

```
root@master:~# kubectl get nodes
NAME      STATUS    ROLES    AGE   VERSION
master    Ready     control-plane  22m   v1.29.2
worker1    Ready     <none>      96s   v1.29.2
worker2    Ready     <none>      82s   v1.29.2
```

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



Installation of Jenkins (step 2)

How to deploy Jenkins with Our custom domain and SSL

For install Jenkins we need to follow below shell script, i am using ubuntu 20.04 this script also work for ubuntu 22.04

\$vi jenkins.sh

Paste this script in Jenkins.sh

```
#!/bin/bash

sudo apt update

sudo apt install openjdk-11-jre -y

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

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 install -y maven

sudo apt-get install jenkins -y

sudo systemctl enable jenkins

sudo systemctl start jenkins

sudo systemctl status Jenkins
```


PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



Run script using this command

\$ssh jenkins.sh

After that we need to check using Jenkins using ec2 instance id for example:- 34.203.34.186:8080 for checking purpose but our end goal its should not be run on this after configuring nginx reverse proxy will be remove port 8080 from ec2 instance security group and check , Here you can see our Jenkins up and running

```
Active: active (running) since Wed 2024-08-07 16:39:17 UTC; 4s ago
Main PID: 4354 (java)
Tasks: 44 (limit: 4676)
Memory: 479.0M (peak: 485.5M)
CPU: 20.444s
CGroup: /system.slice/jenkins.service
└─4354 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache>

Aug 07 16:39:10 jenkins jenkins[4354]: a9071988ca2741eeab6b57e5acdb177f
Aug 07 16:39:10 jenkins jenkins[4354]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Aug 07 16:39:10 jenkins jenkins[4354]: *****
Aug 07 16:39:10 jenkins jenkins[4354]: *****
Aug 07 16:39:10 jenkins jenkins[4354]: *****
Aug 07 16:39:17 jenkins jenkins[4354]: 2024-08-07 16:39:17.760+0000 [id=32] INFO jenkins.InitReac>
Aug 07 16:39:17 jenkins jenkins[4354]: 2024-08-07 16:39:17.859+0000 [id=22] INFO hudson.lifecycle>
Aug 07 16:39:17 jenkins systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Aug 07 16:39:18 jenkins jenkins[4354]: 2024-08-07 16:39:18.600+0000 [id=47] INFO h.m.DownloadServ>
root@jenkins:~# cat /var/lib/jenkins/secrets/initialAdminPassword
a9071988ca2741eeab6b57e5acdb177f
root@jenkins:~#
```

Access your Jenkins Dashboard on browser now

Note: Make sure you enable 8080 port in Security Group Inbound Rules.

Get the initial administration password

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



\$sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Getting Started

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	** JQuery3 API
✓ Timestampers	✓ Workspace Cleanup	✓ Ant	⌚ Gradle	** ECharts API
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline Graph View	** Display URL API
⌚ Git	⌚ SSH Build Agents	⌚ Matrix Authorization Strategy	⌚ PAM Authentication	** Checks API
⌚ LDAP	⌚ Email Extension	✓ Mailer	⌚ Dark Theme	** JUnit

Workspace Cleanup

Ant

** Jakarta Activation API

** Jakarta Mail API

** Apache HttpComponents Client 4.x API

Mailer

** Pipeline: Basic Steps

Gradle

** - required dependency

Jenkins 2.452.2

Now your Jenkins started: **Create Admin user**

Getting Started

Create First Admin User

Username

divya

Password

Confirm password

Full name

Divya Vasant Satpute

E-mail address

Jenkins 2.452.2

Skip and continue as admin

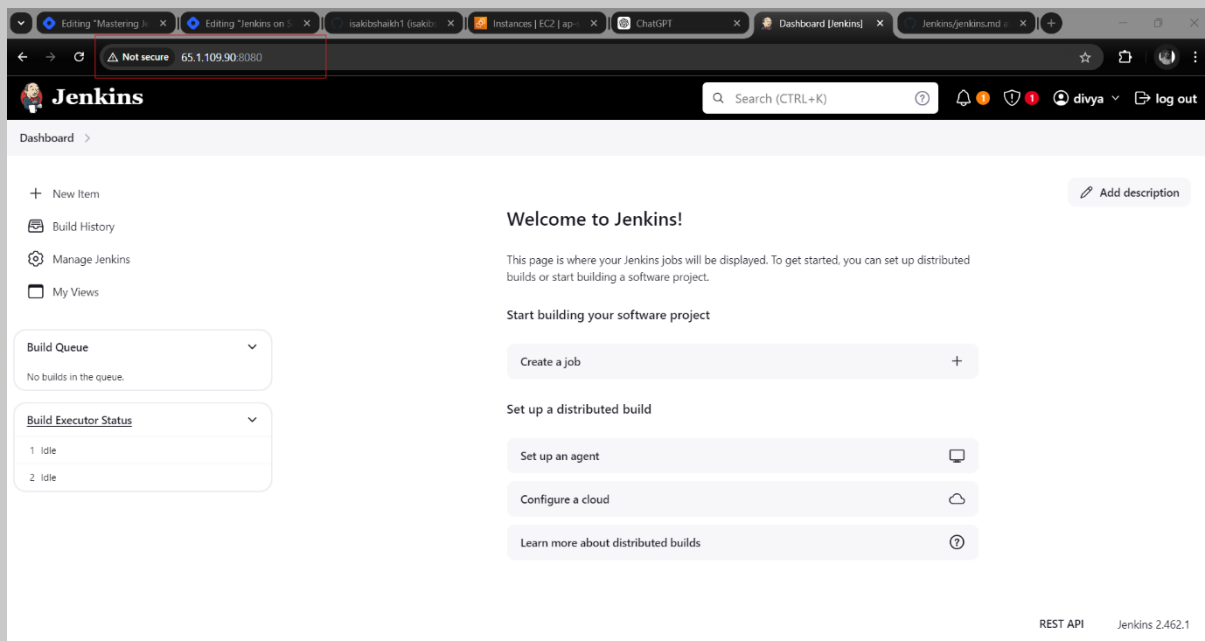
Save and Continue

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



🎉 Now, appreciate yourself you successfully deploy Jenkins on your machine 🎉

It is Look like this: Jenkins up and Running but this is not yet secured we have to add SSL for secure connection



Step-by-step guide to configure SSL on Jenkins using Let's Encrypt and NGINX reverse proxy (step 3)

Install NGINX on your server if it's not already installed. You can do this by running the following command:

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



#update repository first

\$sudo apt-get update -y

#install nginx by following commad

\$sudo apt-get install nginx -y

#check nginx status

\$sudo systemctl status nginx

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@jenkins:~# sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Wed 2024-08-07 17:03:44 UTC; 55s ago
     Docs: man:nginx(8)
   Process: 5869 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Process: 5871 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Main PID: 5872 (nginx)
    Tasks: 3 (limit: 4676)
   Memory: 2.4M (peak: 2.5M)
      CPU: 15ms
   CGroup: /system.slice/nginx.service
           └─5872 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─5873 "nginx: worker process"
               └─5875 "nginx: worker process"

Aug 07 17:03:44 jenkins systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server:
Aug 07 17:03:44 jenkins systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server:
11:11 1 17/17 /5MB
```

Create a new server block configuration file for Jenkins. You can do this by creating a new file in the `/etc/nginx/sites-available/` directory. For example:

\$vi /etc/nginx/sites-available/jenkins

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



Add the following content to the file: Note: Replace jenkins.example.com with your Jenkins domain name.

```
server {  
    listen 80;  
    server_name jenkins.learnwithdivya.online;  
  
    location / {  
        proxy_pass http://localhost:8080;  
        proxy_set_header Host $host;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
        proxy_set_header X-Forwarded-Proto $scheme;  
    }  
}
```

Create a symbolic link to the server block configuration file in the `/etc/nginx/sites-enabled/` directory:

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

Note: you can see `/etc/nginx/sites-available/jenkins` on this path the file - represented simple file (`$sudo ls -l /etc/nginx/sites-available/jenkins`)

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



```
root@jenkins:~# ls -lrt /etc/nginx/sites-available/jenkins
-rw-r--r-- 1 root root 354 Aug  7 17:25 /etc/nginx/sites-available/jenkins
```

after running above command its create shortcut because by default Jenkins look this file on /etc/nginx/sites-enabled location (ls -lrt /etc/nginx/sites-enabled)

```
root@jenkins:~# ls -lrt /etc/nginx/sites-enabled/
total 0
lrwxrwxrwx 1 root root 34 Aug  7 17:03 default -> /etc/nginx/sites-available/default
lrwxrwxrwx 1 root root 34 Aug  7 17:28 jenkins -> /etc/nginx/sites-available/jenkins
```

Test the NGINX configuration and restart the NGINX service:

\$sudo nginx -t

```
ubuntu@jenkins:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Restart the nginx server

\$sudo systemctl restart nginx

Install Certbot, the Let's Encrypt client, by running the following commands:

\$sudo apt-get update -y

\$sudo apt-get install certbot python3-certbot-nginx -y

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



Obtain an SSL certificate for your Jenkins domain name using Certbot:

```
$sudo certbot --nginx -d jenkins.learnwithdivya.online
```

#Note: Replace jenkins.example.com with your Jenkins domain name.

After running above you can found error like this:

```
Certbot failed to authenticate some domains (authenticator: nginx). The Certificate Authority reported these problems:
  Domain: jenkins.learnwithdivya.online
  Type:   dns
  Detail: DNS problem: NXDOMAIN looking up A for jenkins.learnwithdivya.online - check that a DNS record exists for this domain; DNS problem: NXDOMAIN looking up AAAA for jenkins.learnwithdivya.online - check that a DNS record exists for this domain

Hint: The Certificate Authority failed to verify the temporary nginx configuration changes made by Certbot. Ensure the listed domains point to this nginx server and that it is accessible from the internet.

Some challenges have failed.
Ask for help or search for solutions at https://community.letsencrypt.org. See the logfile /var/log/letsencrypt/letsencrypt.log or re-run Certbot with -v for more details.
ubuntu@jenkins:~$
```

For resolve this issue you need to create A name record on your Route53 service

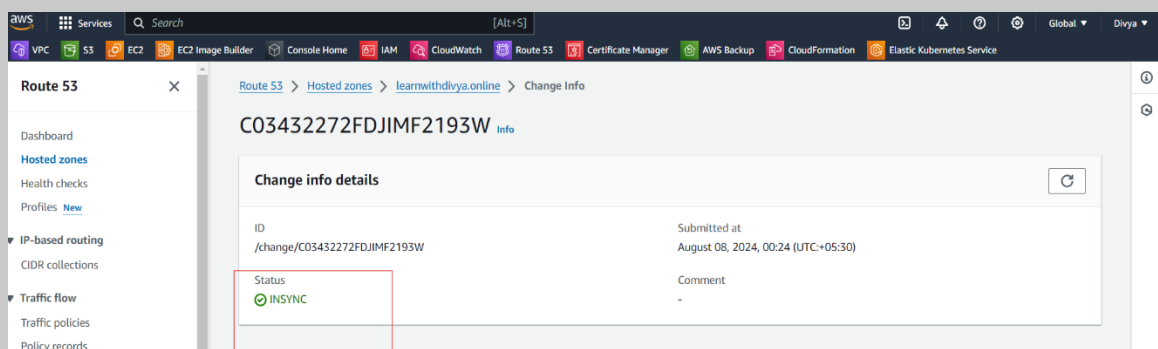
PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



Go to route53 --> click on hosted zone --> click on your domain name ---> create on record, once it will be insync then you can try again above command

Then run the command

\$sudo certbot --nginx -d jenkins.learnwithdivya.online



```
ubuntu@jenkins:~$ sudo certbot --nginx -d jenkins.learnwithdivya.online
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Requesting a certificate for jenkins.learnwithdivya.online

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/jenkins.learnwithdivya.online/fullchain.pem
Key is saved at: /etc/letsencrypt/live/jenkins.learnwithdivya.online/privkey.pem
This certificate expires on 2024-11-05.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in the background.

Deploying certificate
Successfully deployed certificate for jenkins.learnwithdivya.online to /etc/nginx/sites-enabled/jenkins
Congratulations! You have successfully enabled HTTPS on https://jenkins.learnwithdivya.online

-----
If you like Certbot, please consider supporting our work by:
* Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
* Donating to EFF: https://eff.org/donate-le
```

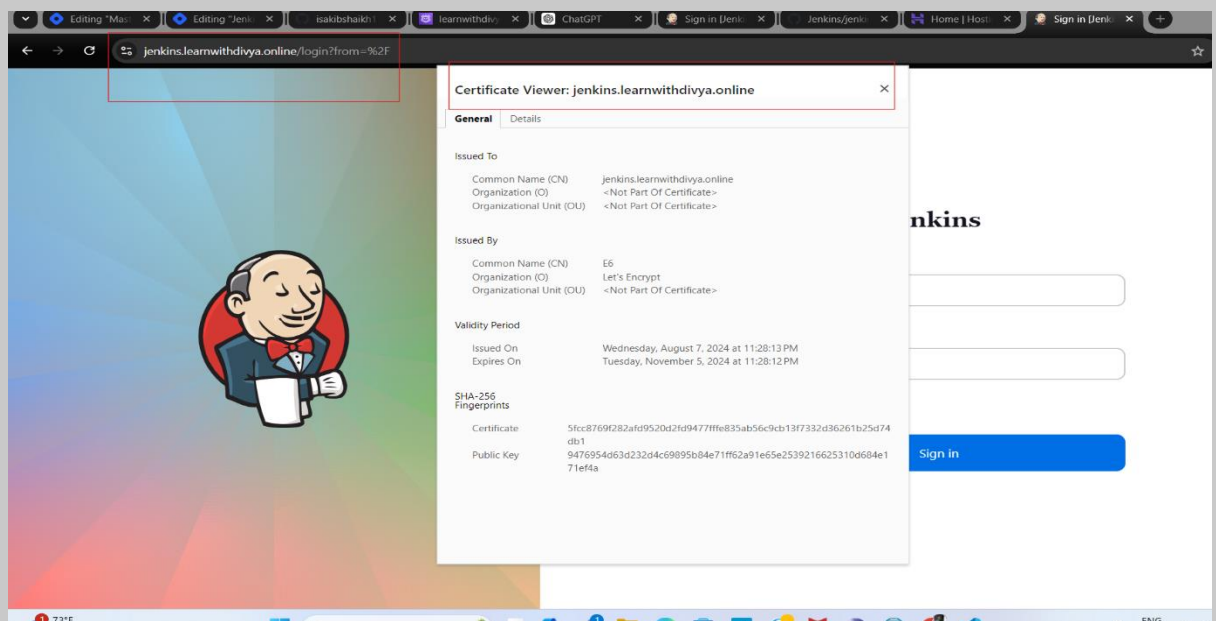
🎉 Congratulations! 🎉

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



You've put in hard work, dedication, and perseverance to reach this milestone!

Now your Jenkins runs in secure with SSL certificate in secure site



It means without using LLB we are use nginx as a reverse proxy : if you are trying to click <http://> still it will be redirect to <https://>

Sounds interesting, right?

Now move to next

create webhook on Github(step 4)

- Click on repo settings ---->

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



- click on webhook ---->
- Add new Webhook --->
- in Payload URL configure your Jenkins URL followed by github-webhook and content type json
- click on ADD webhook

github.com/divyasatpute/Jenkins/settings/hooks/new

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

Security

Code security and analysis

Deploy keys

Secrets and variables

Integrations

GitHub Apps

Email notifications

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

https://jenkins.learnwithdivya.online/github-webhook

Content type *

application/json

Secret

SSL verification

By default, we verify SSL certificates when delivering payloads.

☒ Enable SSL verification ☐ Disable (not recommended)

Which events would you like to trigger this webhook?

☒ Just the push event.

☐ Send me everything.

☐ Let me select individual events.

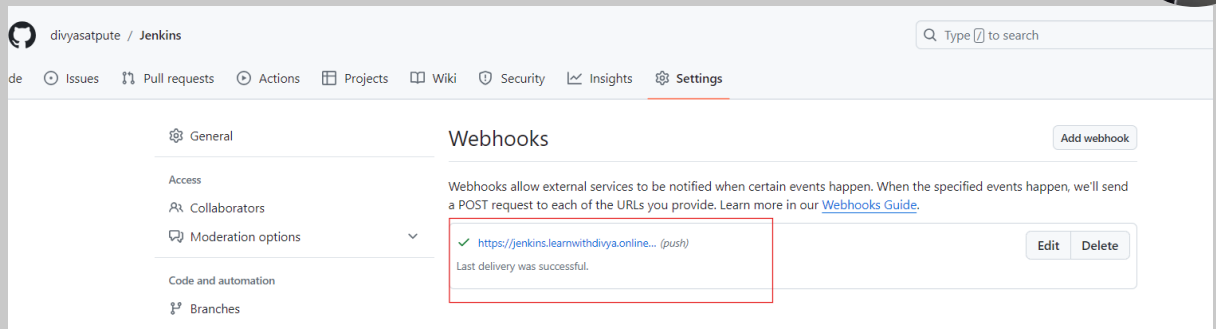
☒ Active

We will deliver event details when this hook is triggered.

Add webhook

Our Webhooks are ready

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



Docker Install on Jenkins machine

\$sudo apt install docker.io -y

Give permission to socket

\$sudo usermod -aG docker \$USER

\$sudo chown \$USER:docker /var/run/docker.sock

change ownership

\$sudo chown jenkins:docker /var/run/docker.sock

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



Go to docker hub account generate token

now on your jenkins machine click on manage jenkins -->
click on [Credentials](#) --->

system ---> global [Credentials](#)

ADD your docker token as a password

ADD (ID) which is you given in your pipeline

```
stage('Build Docker Image'){
    steps{
        script {
            def customImage = docker.build("divyasatpute/petclinic:${env.BUILD_NUMBER}", "./docker")
            docker.withRegistry('https://registry.hub.docker.com', 'dockerhub') {
                customImage.push()
            }
        }
    }
}
```

Plugins Installation

Go TO Jenkins Dashboard → Manage jenkins -> plugins ---> available plugins ---->

1. docker
2. docker pipeline
3. docker commons
4. docker -build-steps
5. docker slave

install it

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



Installation of Helm

Helm Install on **Jenkins Machine** and **Master Node**

Run this command on both machine

```
$curl -fsSL -o get_helm.sh
```

```
https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
```

```
chmod 700 get_helm.sh
```

```
./get_helm.sh
```

configuration on master node

change directory

```
$ cd .kube/
```

copy config file in /home/ubuntu

```
$cp config /home/ubuntu/
```

change directory

```
$cd /home/ubuntu/
```

change ownership

```
$sudo chown ubuntu:ubuntu config
```

break connection come into your local machine download folder

```
$exit
```

PROJECT ON PUSHBASED APPROACH ON JENKINS

DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



```
total 8
-rw----- 1 ubuntu ubuntu 5652 Aug 10 14:12 config
root@master:/home/ubuntu# exit
logout
ubuntu@master:~$ exit
logout
Connection to ec2-43-204-109-240.ap-south-1.compute.amazonaws.com closed.

Rinku@Rinku MINGW64 ~/downloads (master)
$
```

Now fire this command on local (copy file from remote to local)

\$scp -i divya.pem ubuntu@43.204.109.240:/home/ubuntu/config .

```
config 100% 5652 204.7KB/s 00:00
Rinku@Rinku MINGW64 ~/downloads (master)
-rw-r--r-- 1 Rinku 197121 5652 Aug 10 19:53 jenkins_key.pem
-rw-r--r-- 1 Rinku 197121 5652 Aug 10 19:53 config
```

Now to go to Jenkins dashboard

1. click on manage Jenkins
2. click on Credencials
3. select as a secrete file
4. and upload config file which is we downloaded in our local machine

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP

A screenshot of the Jenkins web interface showing the 'New credentials' form. The form has a header 'New credentials' and a breadcrumb trail: 'Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >'. The form fields include: 'Kind' (a dropdown menu with 'Secret file' selected), 'Scope' (a dropdown menu with 'Global (Jenkins, nodes, items, all child items, etc)' selected), 'File' (a section with a 'Choose File' button and a 'config' text input), 'ID' (a text input field), and 'Description' (a text input field). The 'File' section is highlighted with a red rectangle.

- Go to Jenkins dashboard
- click on manage Jenkins
- click on available plugins
- 1. [Kubernetes Client API](#)
- 2. [Kubernetes Credentials](#)
- 3. [Kubernetes](#)
- 4. [Kubernetes CLI](#)
- 5. [Kubernetes Credentials Provider](#)
- install it

How to Set Up the Jenkins + GitHub
Integration ####

NOW LAST STEP TO ACHIEVE THE THINGS

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



create Jenkins pipeline

```
pipeline {  
  agent any  
  stages {  
    stage('Build Maven') {  
      steps {  
        sh 'pwd'  
        sh 'mvn clean install package'  
      }  
    }  
  
    stage('Copy Artifacts') {  
      steps {  
        sh 'pwd'  
        sh 'cp -r target/*.jar docker'  
      }  
    }  
  
    stage('Unit Tests') {  
      steps {  
        sh 'mvn test'  
      }  
    }  
  
    stage('Build Docker Image'){
```


PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



```
    steps{  
        script {  
            def customImage = docker.build("iamsakib/petclinic:${env.BUILD_NUMBER}",  
            "./docker")  
            docker.withRegistry('https://registry.hub.docker.com', 'dockerhub') {  
                customImage.push()  
            }  
        }  
    }  
}  
  
stage('Build on kubernetes'){  
    steps {  
        withKubeConfig([credentialsId: 'kubeconfig']) {  
            sh 'pwd'  
            sh 'cp -R helm/* .'  
            sh 'ls -ltrh'  
            sh 'pwd'  
            sh '/usr/local/bin/helm upgrade --install petclinic-app petclinic --set  
image.repository=iamsakib/petclinic --set image.tag=${BUILD_NUMBER}'  
        }  
    }  
}  
  
}
```

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



}

#replace yours id

Deployment of Application

- Go to Jenkins dashboard
- click on new item

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



- give name to your project
- select pipeline
- select GitHub hook trigger for GITScm polling
- select pipeline script with SCM
- SCM =git and configure GIT URL and branch
- script path should be jenkinsfile (**jenkinsfile should be available on your git repo**)
- save and apply
- Access your application
<worker_node_Pliblic_IP:32740>

And boommmmmmmmmmmmmmmmm

🌟 **Way to Go!** 🌟

You've done it—what an incredible achievement! This moment is a testament to your effort, resilience, and talent. You faced the challenge head-on and came out on top, and now it's time to celebrate that success.

Remember, this is proof of what you're capable of. Keep pushing forward, and don't forget to enjoy every victory along the way. You've earned it—**congratulations!**

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



✓ Test Results ✓

The screenshot shows the Jenkins Pipeline Console for Build #4. The build is successful, having completed in 7.7 seconds. The pipeline consists of several stages, with the 'Build on Kubernetes' stage being the current focus. This stage includes four steps: 'pwd', 'cp -R helm/*', 'ls -ltrh', and 'pwd', all of which are marked as successful. The console output for each step is visible, showing the execution of shell scripts.

The screenshot shows the Jenkins Job Configuration page for the 'spring-boot-PetClinic-app' job. The job is in a successful state. The left sidebar contains various configuration options such as Status, Changes, Build Now, Configure, Delete Pipeline, Stages, Rename, Pipeline Syntax, and GitHub Hook Log. The main content area displays the job name and a list of recent builds. The 'Build History' section shows a list of builds, with the first build (#1) being the most recent and successful. The build details for the first build are visible, showing it was completed on 7 Aug 2024 at 20:16.

PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP



```
root@master:~# k get all
```

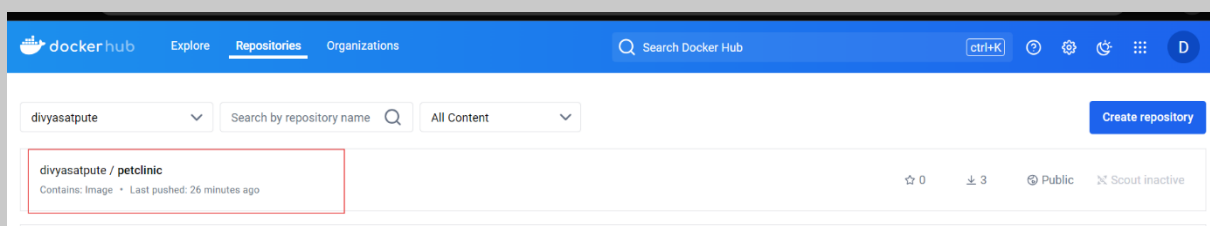
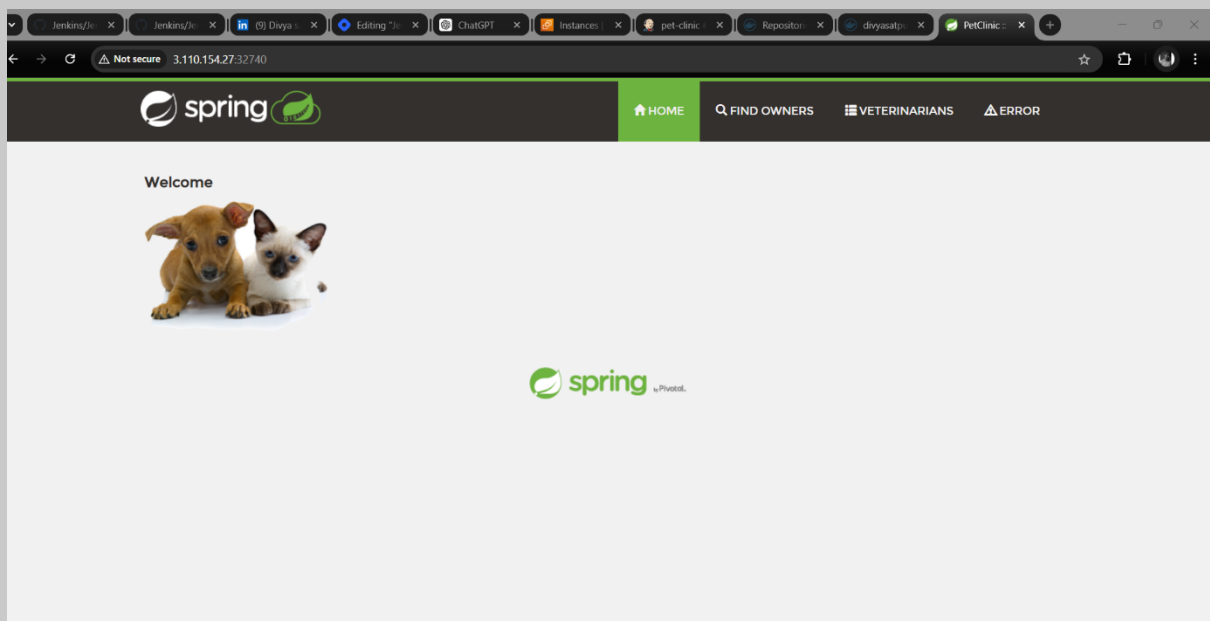
NAME	READY	STATUS	RESTARTS	AGE
pod/petclinic-deployment-6b5d5d5bbc-hb662	0/1	ErrImagePull	0	3m18s
pod/petclinic-deployment-6b5d5d5bbc-n7n7h	0/1	ErrImagePull	0	3m18s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	162m
service/petclinic-service	NodePort	10.103.157.81	<none>	8080:32740/TCP	3m18s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/petclinic-deployment	0/2	2	0	3m18s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/petclinic-deployment-6b5d5d5bbc	2	2	0	3m18s

NAME	MAXPODS	REPLICAS	AGE	REFERENCE	TARGETS	MINPODS
horizontalpodautoscaler.autoscaling/petclinic-deployment	5	2	3m18s	Deployment/petclinic-deployment	<unknown>/50%	1



PROJECT ON PUSHBASED APPROACH ON JENKINS DEPLOYMENT OF SPRING BOOT PET-CLINIC APP

Thank you so much for referring my Notes I hope it will be helpful for you!

🌟 really appreciate your support! If you found them helpful, feel free to like, share, or leave a comment! Your feedback means a lot to me and helps me create even better content.

😊 Thanks again! 🙌

