

Comprehensive CI/CD Pipeline with Monitoring: Git, Jenkins, Docker, Kubernetes, Prometheus, and Grafana

Project Overview

This project demonstrates the implementation of a comprehensive CI/CD pipeline using Git, Terraform, Ansible, Docker, and Kubernetes to automate software deployment, scaling, and operations across a Kubernetes cluster. Additionally, the pipeline includes monitoring of Jenkins jobs with Prometheus and Grafana for enhanced observability.

Challenges Addressed

- Increasing Demand: The company required a platform to handle increased demand for their product, which utilized Docker containers.
- Monolithic Architecture: Previously, they used a monolithic architecture with manual deployments.
- Version Control: Implementing a Git workflow for version control and controlled releases.

Solution Implemented

- CI/CD Pipeline: A Jenkins pipeline automates the entire deployment lifecycle.
- Git Integration: Version control is implemented with Git, triggering builds on commits to the master branch.
- Dockerization: Dockerfiles containerize the application code for deployment.
- Kubernetes Deployment: Containerized code gets deployed on a Kubernetes cluster with two replicas.
- NodePort Service: A NodePort service exposes the application on port 30008.
- Infrastructure Provisioning: Terraform manages infrastructure creation on AWS.
- Configuration Management: Ansible deploys necessary software and configurations on servers.
- Monitoring Integration: Prometheus and Grafana are used for monitoring Jenkins jobs. Prometheus collects metrics from Jenkins, and Grafana provides dashboards for visualizing these metrics.

Monitoring Integration

- Prometheus and Grafana: Monitoring of Jenkins jobs is integrated using Prometheus and Grafana.
 - Prometheus Setup: Prometheus is configured to scrape Jenkins metrics and store them.
 - Grafana Dashboard: Grafana is set up to visualize Jenkins metrics, including job performance, build duration, and failure rates.

Key Technologies

- CI/CD Tool: Jenkins

- Version Control: Git
- Infrastructure Provisioning: Terraform
- Configuration Management: Ansible
- Containerization: Docker
- Container Orchestration: Kubernetes
- Monitoring: Prometheus, Grafana

Skills Demonstrated

- CI/CD Pipeline Design and Implementation
- Git Integration for Version Control
- Dockerfile Creation and Containerization
- Kubernetes Deployment and Management
- Infrastructure Provisioning with Terraform
- Configuration Management with Ansible
- Monitoring with Prometheus and Grafana

Project Outcomes

- Automated Deployment: Streamlined the deployment process for faster releases.
- Scalability: Enabled easy scaling to handle increased demand.
- Improved Efficiency: Reduced manual intervention and improved operational efficiency.
- Enhanced Observability: Provided real-time monitoring and insights into Jenkins job performance and system health.

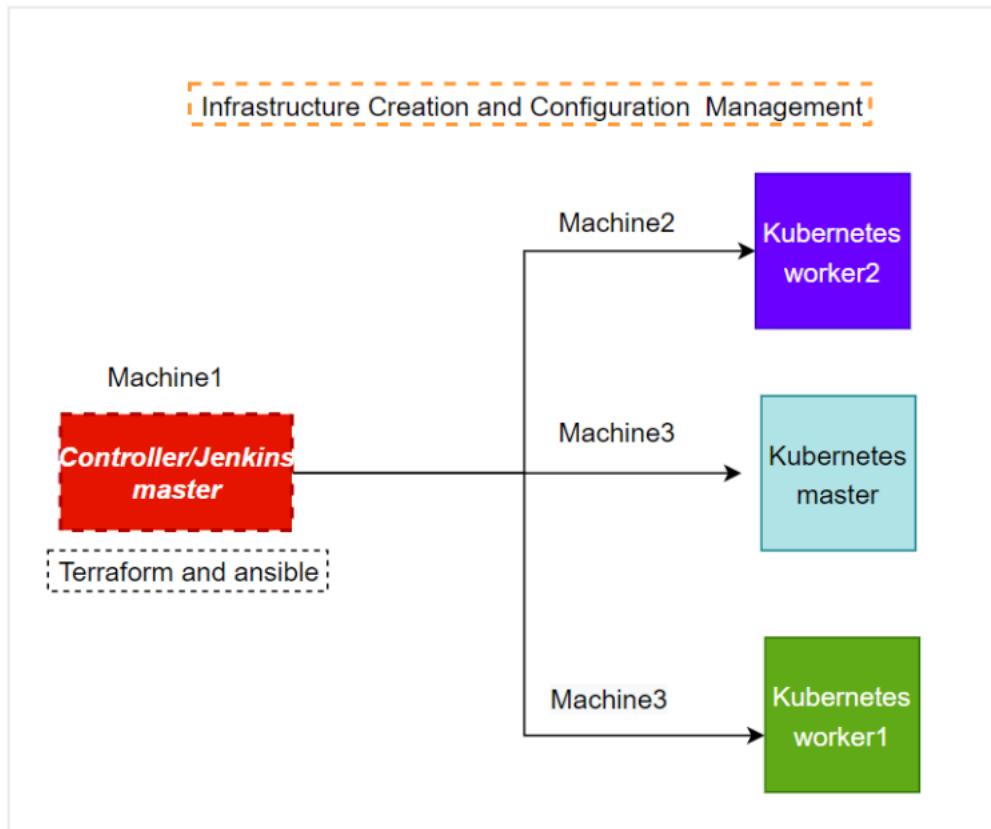
Documentation

This PDF file outlines the complete deployment process with screenshots.

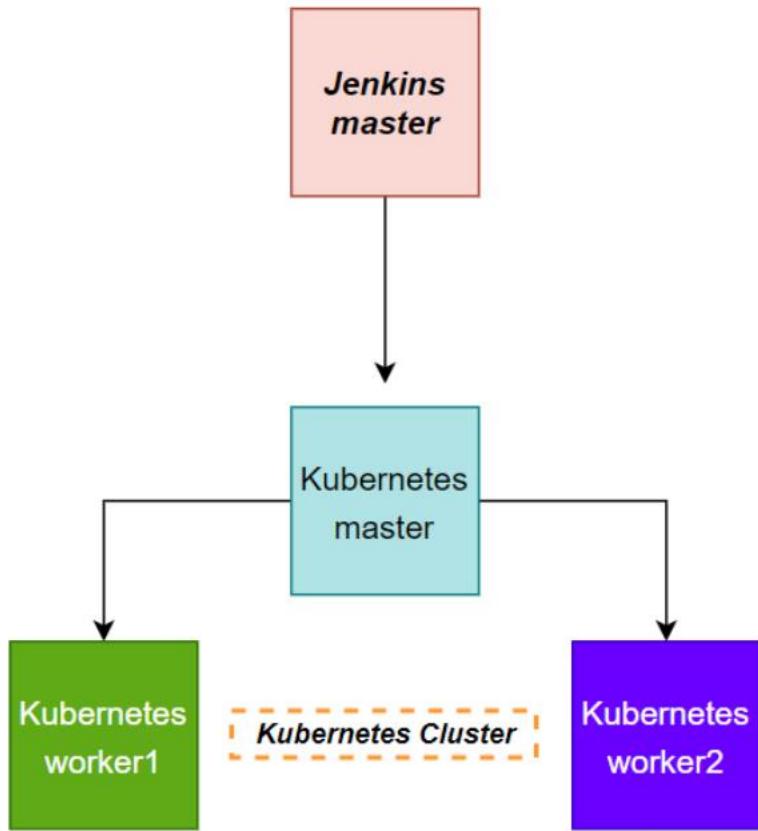
By showcasing this project, I aim to demonstrate my ability to design, implement, and manage complex DevOps CI/CD pipelines for efficient and scalable application deployments.

Contact Information

- Name: Kuruva Venkata Narayana
- Email: venkatnarayananakuruva@gmail.com
- Phone: +91-8309285798



Servers for jenkins and kubernetes configuration



Terraform installation

- Created a EC2 instance manually in aws console
- Navigated into the EC2 instance and installed terraform through a shellscript

```
ubuntu@worker1:~$ vi terraforminstall.sh
ubuntu@worker1:~$ bash terraforminstall.sh

i-00b8547948a9cfa59 (Worker1)

PublicIPs: 3.141.13.91  PrivateIPs: 172.31.26.221

sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
wget -O https://apt.releases.hashicorp.com/gpg | \
gpg --dearmor | \
sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg
gpg --no-default-keyring \
--keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \
--fingerprint
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \
https://apt.releases.hashicorp.com ${lsb_release -cs} main" | \
sudo tee /etc/apt/sources.list.d/hashicorp.list
sudo apt update
sudo apt-get install terraform -y
~
~
~
~
~
~
~
"terraforminstall.sh" 12L, 548B
```

Architecture creation

After installation of terraform created a directory named infracreation

- Navigated inside the directory and created a main.tf file with a hcl script to create three EC2 instances

```
ubuntu@worker1:~$ mkdir infracreation
ubuntu@worker1:~$ cd infracreation
ubuntu@worker1:~/infracreation$ vi main.tf
```

```

provider "aws"{
    secret_key= ""
    access_key= ""
    region= "eu-east-2"
}

resource "aws_instance" "k8master"{
    ami= "ami-05fb0b8c1424f266b"
    instance_type= "t2.medium"
    key_name= "venkatohio"
    provider= "aws.worker2"
    tags= {
        Name= "worker2"
    }
}
resource "aws_instance" "k8slave"{
    ami= "ami-05fb0b8c1424f266b"
    instance_type= "t2.medium"
    key_name= "venkatohio"
    provider= "aws.worker3"
    tags= {
        Name= "worker3"
    }
}
resource "aws_instance" "k8slave"{
    ami= "ami-05fb0b8c1424f266b"
    instance_type= "t2.medium"
    key_name= "venkatohio"
    provider= "aws.worker4"
    tags= {
        Name= "worker4"
    }
}

```

- Then initialized terraform

```

ubuntu@worker1:~/infracreation$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
ubuntu@worker1:~/infracreation$ 

```

- Performed terraform plan

```
ubuntu@worker1:~/infracreation$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.this2 will be created
+ resource "aws_instance" "this2" {
    + ami                               = "ami-05fb0b8c1424f266b"
    + arn                               = (known after apply)
    + associate_public_ip_address      = (known after apply)
    + availability_zone                = (known after apply)
    + cpu_core_count                   = (known after apply)
    + cpu_threads_per_core            = (known after apply)
    + disable_api_stop                = (known after apply)
    + disable_api_termination         = (known after apply)
    + ebs_optimized                    = (known after apply)
    + get_password_data               = false
    + host_id                          = (known after apply)
```

- And then applied the changes by terraform apply command

```
aws_instance.this3: Creating...
aws_instance.this4: Creating...
aws_instance.this2: Creating...
aws_instance.this3: Still creating... [10s elapsed]
aws_instance.this4: Still creating... [10s elapsed]
aws_instance.this2: Still creating... [10s elapsed]
aws_instance.this3: Still creating... [20s elapsed]
aws_instance.this4: Still creating... [20s elapsed]
aws_instance.this2: Still creating... [20s elapsed]
aws_instance.this3: Still creating... [30s elapsed]
aws_instance.this4: Still creating... [30s elapsed]
aws_instance.this2: Still creating... [30s elapsed]
aws_instance.this3: Creation complete after 31s [id=i-07edd73dde8245ebf]
aws_instance.this4: Creation complete after 31s [id=i-03b429699eedc391e]
aws_instance.this2: Still creating... [40s elapsed]
aws_instance.this2: Still creating... [50s elapsed]
aws_instance.this2: Creation complete after 52s [id=i-0b31db3012c176c15]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
ubuntu@worker1:~/infracreation$ []
```

Ansible configuration

- Then installed ansible in the server through a ansibleinstall.sh shell script file

```
GNU nano 6.2                                         ansibleinstall.sh
sudo apt update
sudo apt install software-properties-common
sudo apt-add-repository ppa: ansible/ansible
sudo apt install ansible

[ Read 4 lines ]
^G Help      ^C Write Out   ^W Where Is   ^K Cut       ^T Execute   ^C Location   M-U Undo
^X Exit      ^R Read File    ^\ Replace     ^U Paste     ^J Justify    ^/ Go To Line  M-E Redo
```

Ansible is successfully installed

```
ubuntu@worker1:~$ ansible --version
ansible 2.10.8
  config file = None
  configured module search path = ['/home/ubuntu/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]
ubuntu@worker1:~$ 
```

- Ssh key is generated

```
ubuntu@worker1:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:rSSzP1TSJq9shqpp9O+E2D2pfoOyS920KCG421+fwCE ubuntu@worker1
The key's randomart image is:
+---[RSA 3072]---+
|                               |
|                               |
|                               |
|                               . |
| .   O.+ |
| o . E o S*. |
| o.= O +... |
| ..+=. @+.. |
| +ooo+B+*o |
| ..*B==o0=. |
+---[SHA256]---+ 
```

- Ssh private key is copied

```
ubuntu@worker1:~$ sudo cat /home/ubuntu/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQBgQCxV10IcrTODX1Kaqg427G+1Zh5PnV/EICAQDk+aJKqEswZDR4NCQiFPYhPdTnjAsUCs7qR4berZwxAAAdS7tvZ2o3oiiki20uuud0imjNhULp0okVKWCATUm66IW/3IWinmb70vKOzc5e+H12Htw3HMbF7Cvc1+xx/QYFE8IXUJ9yIQ5C03BYARqi++aB5r63DP4VGj5qb67nHEz9IM9j1aFILIE/7wV1wbYbEX6vPCP7Ruitp+L/6IewxGP0HgCLT+ew09rTCCB4jaB9FpZvGewgPI200V3q7nOpwgHXv8nTwDnqZDTQ8r0uLTQdDnxkhNA+x100q5kx90Bps2dzeb0uQ0ho9Oh+GSmNeIzV6kWxID+eF7t+Kv5QyZHUMsmGJ0sXRpVsILsMqd00DN53Lcr27Lu0Y6J7cqQgQl/mXsek1NNUTpmfS3AO8lIKtRe0DwRhB1jb5q1L2L0qRe+RZkfkLoxsG27GalzvTs58Let6IDj1h4fsCdUHM= ubuntu@work
er1
ubuntu@worker1:~$ 
```

- Navigated to the server worker2 and opened authorized keys file and pasted ssh key generated in worker 1 to establish connection between the two

Similarly same steps performed in worker 3 and worker4

```
GNU nano 6.2                                         ./ssh/authorized_keys *
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCWFKI/ORqXVvGT4Pdb/6/VH4BmPA9uryj2vGHFYASYrhbo0V+2s1n0btYcRcAaD
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQGQxVi0IcrTODX1Kaqs427G+1Zh5PnV/EICAQDk+aJKqEswZDR4NQiFPYhPdTNjA
^G Help      ^O Write Out   ^W Where Is   ^K Cut        ^T Execute   ^C Location M-U Undo
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line M-E Redo
i-0b31db3012c176c15 (worker2)
PublicIPs: 18.116.61.112  PrivateIPs: 172.31.20.162
```

Navigated back to the main server ie., worker1 and created a file named inventory and added private IPs of the worker 2 3 & 4 servers

- Choosen worker3 as a kubernates master and jenkins slave node
- Remaining two servers as kubernates slaves

```
GNU nano 6.2                                         inventory
[k8Master]
worker3 ansible_host=172.31.20.131
[k8slaves]
worker2 ansible_host=172.31.20.162
worker4 ansible_host=172.31.19.234
^G Help      ^O Write Out   ^W Where Is   ^K Cut        [ Read 5 lines ] ^T Execute   ^C Location M-
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line M-
i-00b8547948a9cfa59 (Worker1)
PublicIPs: 3.141.13.91  PrivateIPs: 172.31.26.221
```

- With that ansible cluster setup is completed. Ensured connection between all servers by performing ansible ping

```
ubuntu@worker1:~$ sudo nano inventory
ubuntu@worker1:~$ sudo nano inventory
ubuntu@worker1:~$ ansible -i inventory all -m ping
The authenticity of host '172.31.20.131 (172.31.20.131)' can't be established.
ED25519 key fingerprint is SHA256:xLB5RiGAGABPN0R0DbVSz2KIO59if0V27dyio4nbkU0.
This key is not known by any other names
-----
The authenticity of host '172.31.19.234 (172.31.19.234)' can't be established.
ED25519 key fingerprint is SHA256:ZkeI/lv14HezAbYz0UxAc1Dr8T0CtPQfBwmuDVgypA.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? worker2 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
yes
worker3 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
yes
worker4 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
ubuntu@worker1:~$ 
```

Configuration deployments

- After that a shell script named localhost.sh is created to install java and jenkins in localhost(worker1)

```
GNU nano 6.2                                     localhost.sh
sudo apt update
sudo apt install openjdk-11-jdk -y
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 jenkins -y
```

- Similarly another shellscript worker3.sh is created to install java docker and kubernates in worker3

```
GNU nano 6.2                                         worker3.sh
sudo apt update
sudo apt install openjdk-11-jdk -y
sudo apt update
sudo apt install docker.io -y
sudo apt update -y

sudo apt install curl apt-transport-https -y
curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg|sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/k8s.gpg
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt update -y
sudo apt install kubelet kubeadm kubectl -y
sudo apt-mark hold kubelet kubeadm kubectl

sudo swapoff -a
sudo sed -i '/ swap / s/^.*$/#\1/g' /etc/fstab
sudo tee /etc/modules-load.d/k8s.conf <EOF
[ Read 46 lines ]
^G Help      ^O Write Out   ^W Where Is    ^K Cut        ^C Location   M-U Undo     M-A Set Mark   M-[ To Bracket
^X Exit      ^R Read File   ^T Replace     ^U Paste      ^J Justify     ^Y Go To Line  M-P Redo     M-B Copy      ^Q Where Was

```

- Finally last shellscript is created to install docker and kubernates on worker2 and worker4

```
GNU nano 6.2                                         slaves.sh *
sudo apt update
sudo apt install docker.io -y
sudo apt update -y

sudo apt install curl apt-transport-https -y
curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg|sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/k8s.gpg
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt update -y
sudo apt install kubelet kubeadm kubectl -y
sudo apt-mark hold kubelet kubeadm kubectl

sudo swapoff -a
sudo sed -i '/ swap / s/^.*$/#\1/g' /etc/fstab
sudo tee /etc/modules-load.d/k8s.conf <EOF
overlay
```

- An ansible playbook is created to eecute these shellscripts in their respective servers/hosts

```
---
- hosts: localhost
  name: installing jenkins and java
  become: yes
  tasks:
    - name: executing localhost.sh
      script: localhost.sh
- hosts: k8Master
  name: installing java docker and kubernates
  become: yes
  tasks:
    - name: executing worker3.sh
      script: worker3.sh
- hosts: k8slaves
  name: installing docker and kubernates
  become: yes
  tasks:
    - name: executing slaves.sh
      script: slaves.sh
~
"playbook.yml" 19L, 427B
```

- The playbook is executed successfully resulting the installations in the target host servers

```

"No VM guests are running outdated hypervisor (qemu) binaries on this host.",
" kubelet set on hold.",
" kubeadm set on hold.",
" kubectl set on hold.",
" net.bridge.bridge-nf-call-iptables=1",
"",
" net.bridge.bridge-nf-call-iptables = 1",
"Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.",
"Executing: /lib/systemd/systemd-sysv-install enable docker"
]
}
META: ran handlers
META: ran handlers

PLAY RECAP ****
localhost          : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
worker2            : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
worker3            : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
worker4            : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
ubuntu@worker1:~$ []

```

Kubernetes cluster configuration

Navigated to the kubernetes master and initialised kubeadm

```

ubuntu@ip-172-31-20-131:~$ sudo kubeadm init --apiserver-advertise-address=172.31.20.131 --pod-network-cidr=10.244.0.0/16
I0107 06:18:34.581405 50266 version.go:256] remote version is much newer: v1.29.0; falling back to: stable-1.28
[init] Using Kubernetes version: v1.28.5
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'

```

Generated kubeadm token is copied

```

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:
kubeadm join 172.31.20.131:6443 --token fe9ej1.gd40acezkaqa0st9 \
    --discovery-token-ca-cert-hash sha256:7428b54a21823b234b8af394bdb4863597695b345ae2e4655f5773731f1ff59d
ubuntu@ip-172-31-20-131:~$ []

```

kubeadm join 172.31.20.131:6443 --token fe9ej1.gd40acezkaqa0st9 \

--discovery-token-ca-cert-hash

sha256:7428b54a21823b234b8af394bdb4863597695b345ae2e4655f5773731f1ff59d

- Kubeadm token is ran in the kubernetes slaves joining them to the cluster
- Kubernetes cluster is successfully configured

```

ubuntu@ip-172-31-20-131:~$ kubectl get nodes
NAME           STATUS   ROLES      AGE     VERSION
ip-172-31-19-234 Ready    <none>    3m59s   v1.28.2
ip-172-31-20-131 Ready    control-plane   26m    v1.28.2
ip-172-31-20-162 Ready    <none>    3m56s   v1.28.2
ubuntu@ip-172-31-20-131:~$ []

```

i-07edd73dde8245ebf (worker3)

PublicIPs: 13.58.198.22 PrivateIPs: 172.31.20.131

Github configuration

- Then navigated to the git repo provided and forked to my github account

The screenshot shows the GitHub repository page for 'Capstone-Project-II'. At the top, there's a navigation bar with links for Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation bar, the repository name 'Capstone-Project-II' is displayed, along with a 'Public' badge and a note that it was forked from 'hshar/website'. There are buttons for Pin, Watch (0), Fork (2.2k), and Star (0). The main content area shows a file tree with 'master' branch details: 'index.html' was modified by 'Ubuntu' 6 years ago; 'images' folder was updated 'final' 6 years ago. On the right, there's an 'About' section with a description: 'No description, website, or topics provided.' It also shows activity metrics: 2 commits, 0 stars, 0 watching, and 2.2k forks.

- Created a new branch named develop

The screenshot shows the GitHub branches page for the same repository. The 'New branch' button is visible at the top right. Under the 'Default' section, the 'master' branch is listed with an 'Updated' status of 'now' and a 'Check status' of 'Default'. In the 'Your branches' section, a new branch named 'develop' is listed with an 'Updated' status of 'now' and a 'Check status' of '0 | 0'. The 'Active branches' section is also shown.

- Then created a new file named dockerfile to copy the entire repo and create a custom image out of it

The screenshot shows the GitHub code editor for the 'Dockerfile' in the 'master' branch of the repository. The Dockerfile contains the following code:

```
1 FROM ubuntu
2 RUN apt update
3 RUN apt-get install apache2 -y
4 ENTRYPOINT apache2 -D FOREGROUND
5 ADD . /var/www/html
```

At the top right, there are 'Cancel changes' and 'Commit changes...' buttons. The code editor interface includes tabs for Edit, Preview, and GitHub Copilot, and settings for Spaces (2), No wrap.

- Similarly created a new file named hshar-deployment.yaml to deploy hshar webpage from the image created by dockerfile with 3 replicas and create and attach nodeport service on port 30008

Venkat-Narayan-07 Update hshar-deployment.yaml

Code Blame 21 lines (21 loc) · 352 Bytes Code 55% faster with GitHub Copilot

```

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: hshar-deployment
5    labels:
6      web: hshar
7  spec:
8    replicas: 1
9    selector:
10      matchLabels:
11        web: hshar
12    template:
13      metadata:
14        labels:
15          web: hshar
16      spec:
17        containers:
18          - name: hshar
19            image: venkatnarayan16/hshar
20            ports:
21              - containerPort: 80

```

Capstone-Project-II / hshar-Service.yaml in master

Edit Preview Code 55% faster with GitHub Copilot

```

1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: hshar-service
5  spec:
6    type: NodePort
7    selector:
8      web: hshar
9    ports:
10      - port: 80
11        nodePort: 30008
12

```

- With that our git repo is ready

Venkat-Narayan-07 Rename Service.yaml to hshar-Service.yaml		
		aedd131 · now ⏲ 20 Commits
images	final	5 years ago
Capstone project II.pdf	Add files via upload	2 days ago
Dockerfile	Create Dockerfile	7 months ago
Jenkinsfile	Create Jenkinsfile	33 minutes ago
README.md	Update README.md	35 minutes ago
hshar-Service.yaml	Rename Service.yaml to hshar-Service.yaml	now
hshar-deployment.yml	Update hshar-deployment.yml	2 days ago
index.html	modified	5 years ago
main.tf	Update main.tf	7 minutes ago

- Finally in the repo settings, a webhook is created to get triggered whenever there is a push to the repo

The screenshot shows the GitHub repository settings page for 'Venkat-Narayan-07 / Capstone-Project-II'. The left sidebar is open, showing various repository settings like General, Access, Collaborators, and Webhooks. The 'Webhooks' section is currently selected and highlighted with a blue bar. A modal window titled 'Webhooks / Add webhook' is open on the right. Inside the modal, there is a text input field labeled 'Payload URL *' which contains the URL 'http://3.141.13.91:8080/'. Below this, there is a dropdown menu for 'Content type' set to 'application/x-www-form-urlencoded'. There is also a 'Secret' input field and a section for selecting events. A radio button for 'Just the push event.' is selected.

Jenkins configuration

- Afterthat navigated to the jenkins dashboard and signed in

The screenshot shows the Jenkins 'Unlock Jenkins' configuration page. At the top, it says 'Getting Started' and 'Unlock Jenkins'. It instructs the user to ensure Jenkins is securely set up by reading the log (not sure where to find it) and provides the path `/var/lib/jenkins/secrets/initialAdminPassword`. A text input field is provided for pasting the password, labeled 'Administrator password'. A 'Continue' button is at the bottom right.

```
ubuntu@worker1:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
1104584b404c4f5998b0fa1460455790
ubuntu@worker1:~$ []
```

i-00b8547948a9cfa59 (Worker1)
PublicIPs: 3.141.13.91 PrivateIPs: 172.31.26.221

The screenshot shows the Jenkins dashboard. The top navigation bar includes 'Jenkins', a search bar, and user information for 'venkata narayana'. Below the navigation, there are links for 'Dashboard', 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. A 'Create a job' button is available. The 'Build Queue' section shows 'No builds in the queue.' The 'Build Executor Status' section shows '1 Idle' and '2 Idle'. The main content area features a 'Welcome to Jenkins!' message, a 'Start building your software project' call-to-action, and a 'Set up a distributed build' section with links for 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'.

- In the jenkins dashboard navigated to manage jenkins and nodes and added a new node

The screenshot shows the Jenkins 'Nodes' page. On the left, there are navigation links for 'Nodes' and 'Clouds'. Below these are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main area is titled 'Nodes' and contains a table with one row. The table columns are: S, Name (sorted by Name), Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, and Response Time. The single node listed is 'Built-In Node' (Architecture: Linux (amd64), Clock Difference: In sync, Free Disk Space: 4.10 GB, Free Swap Space: 0 B, Free Temp Space: 4.10 GB, Response Time: 0ms). A status icon shows a red exclamation mark and 0 B.

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	4.10 GB	0 B	4.10 GB	0ms
		Data obtained	6 min 39 sec	6 min 39 sec	6 min 40 sec	6 min 39 sec	6 min 39 sec

- Named the node as kubernetes master and provided private IP of the worker3 server

The screenshot shows the 'New node' creation form. It has fields for 'Node name' (set to 'kubernetes-master') and 'Type' (radio button selected for 'Permanent Agent'). A descriptive text explains that this adds a plain, permanent agent to Jenkins. At the bottom is a blue 'Create' button.

New node

Node name
kubernetes-master

Type
 Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Create

Name ?

kubernetes-master

Description ?

Plain text [Preview](#)

Number of executors ?

1

Remote root directory ?

/home/ubuntu/jenkins

Labels ?

Usage ?

Use this node as much as possible



Launch method ?

Launch agents via SSH



Host ?

172.31.20.131

Credentials ?

ubuntu



+ Add ▾

Host Key Verification Strategy ?

Non verifying Verification Strategy



Advanced ▾

Availability ?

Keep this agent online as much as possible



Node Properties



Disable deferred wipeout on this node ?

Save

- Worker3 is successfully added as a jenkins node

The screenshot shows the Jenkins 'Nodes' page. On the left, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (two entries: 'Built-In Node' and 'kubernetes-master'). The 'kubernetes-master' entry is highlighted with a yellow box. On the right, a table lists the nodes with columns: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, and Response Time. The 'kubernetes-master' row shows values: Linux (amd64), In sync, 3.88 GB, 0 B, 3.88 GB, 0ms.

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-in Node	Linux (amd64)	In sync	3.88 GB	0 B	3.88 GB	0ms
	kubernetes-master	Linux (amd64)	In sync	3.16 GB	0 B	3.16 GB	34ms
	Data obtained			0.27 sec	0.26 sec	0.23 sec	0.24 sec

- Then created a job1

The screenshot shows the 'New Item' creation page. The 'item name' field contains 'job1'. The 'item type' section shows three options: 'Freestyle project', 'Pipeline', and 'Multi-configuration project'. The 'Pipeline' option is highlighted with a yellow box. A description for 'Pipeline' states: 'Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.' At the bottom is an 'OK' button.

- Job1 is configured to the worker3 and git repo and the master branch is specified in source code management.
- Enabled webhook to trigger build whenever there is a push to master branch

GeneralEnabled **Configure** General Advanced Project Options Pipeline

Description

End-to-end Jenkins CI/CD pipeline using Docker and Kubernetes

Plain text [Preview](#)

- Discard old builds [?](#)
- Do not allow concurrent builds
- Do not allow the pipeline to resume if the controller restarts
- GitHub project
- Pipeline speed/durability override [?](#)

Build Triggers

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

Advanced Project OptionsAdvanced [▼](#)**Pipeline****Configure** General Advanced Project Options Pipeline

Pipeline script from SCM

SCM [?](#)

Git

Repositories [?](#)Repository URL [?](#)

https://github.com/Venkat-Narayan-07/Capstone-Project-II.git

Credentials [?](#)

ubuntu

+ Add [▼](#)Advanced [▼](#)

Configure

General
Advanced Project Options
Pipeline

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?
*/master

Add Branch

Repository browser ?
(Auto)

Additional Behaviours

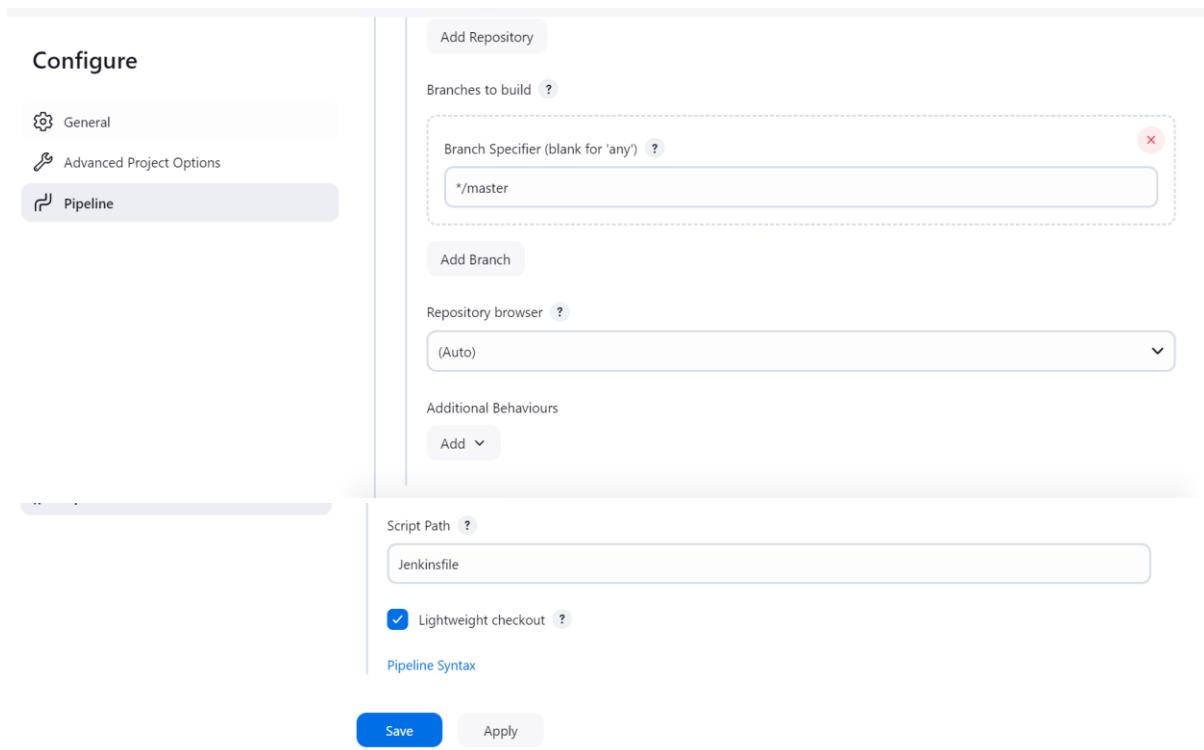
Add ▾

Script Path ?
Jenkinsfile

Lightweight checkout ?

Pipeline Syntax

Save **Apply**



- Saved the job1 to the Jenkins dashboard
- Then manually ran build now to test the job.

Dashboard > job1 >

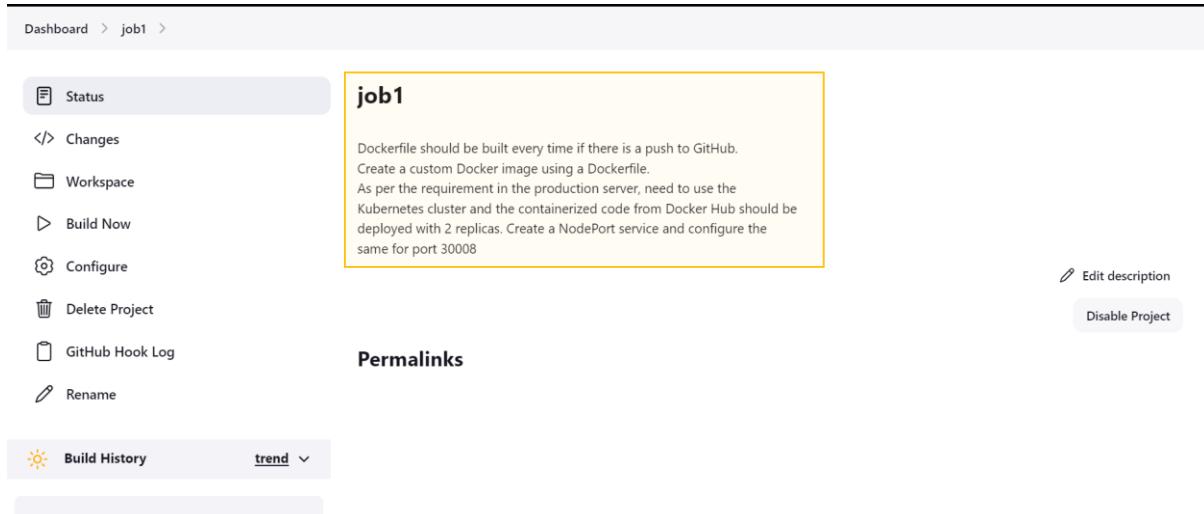
Status Changes Workspace Build Now Configure Delete Project GitHub Hook Log Rename

job1

Dockerfile should be built every time if there is a push to GitHub.
Create a custom Docker image using a Dockerfile.
As per the requirement in the production server, need to use the
Kubernetes cluster and the containerized code from Docker Hub should be
deployed with 2 replicas. Create a NodePort service and configure the
same for port 30008

Edit description Disable Project

Build History trend ▾



- Job1 is successfully completed built operation

</> Changes

[Console Output](#)

[Edit Build Information](#)

[Delete build '#2'](#)

[Timings](#)

[Git Build Data](#)

[Pipeline Overview](#)

[Pipeline Console](#)

[Restart from Stage](#)

[Replay](#)

[Pipeline Steps](#)

[Workspaces](#)

[Previous Build](#)

Started by user **venkat**
 Obtained Jenkinsfile from git <https://github.com/Venkat-Narayan-07/Capstone-Project-II.git>
 [Pipeline] Start of Pipeline
 [Pipeline] withCredentials
 Masking supported pattern matches of \$DOCKERHUB_CREDENTIALS or \$DOCKERHUB_CREDENTIALS_PSW
 [Pipeline] {
 [Pipeline] stage
 [Pipeline] { (git)
 [Pipeline] node
 Running on **K8master** in /home/ubuntu/jenkins/workspace/job1
 [Pipeline] {
 [Pipeline] checkout
 Selected Git installation does not exist. Using Default
 The recommended git tool is: NONE
 using credential d0fe9a57-068a-40a6-a956-da9c9eebd01b
 Fetching changes from the remote Git repository
 Checking out Revision f0366e8a941f6431578e03b754f060392417c288 (refs/remotes/origin/master)
 Commit message: "Update Jenkinsfile"
 > git rev-parse --resolve-git-dir /home/ubuntu/jenkins/workspace/job1/.git # timeout=10
 > git config remote.origin.url <https://github.com/Venkat-Narayan-07/Capstone-Project-II.git> # timeout=10
 Fetching upstream changes from <https://github.com/Venkat-Narayan-07/Capstone-Project-II.git>
 > git --version # timeout=10
 > git --version # 'git version 2.34.1'
 using GIT_SSH to set credentials
 Verifying host key using known hosts file
 You're using 'Known hosts file' strategy to verify ssh host keys, but your known_hosts file does not exist, please go to 'Manage Jenkins' -> 'Security' -> 'Git Host Key Verification Configuration' and configure host key verification.
 > git fetch --tags --force --progress -- <https://github.com/Venkat-Narayan-07/Capstone-Project-II.git>
 +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git config core.sparsecheckout # timeout=10
 > git checkout -f f0366e8a941f6431578e03b754f060392417c288 # timeout=10
 > git rev-list --no-walk 9b9872b0b7643e67c98cf71affda75fae018be30 # timeout=10
 [Pipeline] withEnv
 [Pipeline] {
 [Pipeline] script
 [Pipeline] {
 [Pipeline] git
 Selected Git installation does not exist. Using Default
 The recommended git tool is: NONE
 No credentials specified
 Fetching changes from the remote Git repository
 Checking out Revision f0366e8a941f6431578e03b754f060392417c288 (refs/remotes/origin/master)
 Commit message: "Update Jenkinsfile"
 [Pipeline] }
 [Pipeline] // script
 [Pipeline] }
 [Pipeline] // withEnv
 [Pipeline] }
 [Pipeline] // node
 [Pipeline] }
 [Pipeline] // stage
 [Pipeline] stage
 [Pipeline] { (docker)
 [Pipeline] node
 Running on **K8master** in /home/ubuntu/jenkins/workspace/job1
 [Pipeline] {
 [Pipeline] checkout
 Selected Git installation does not exist. Using Default
 The recommended git tool is: NONE
 using credential d0fe9a57-068a-40a6-a956-da9c9eebd01b
 Fetching changes from the remote Git repository
 > git rev-parse --resolve-git-dir /home/ubuntu/jenkins/workspace/job1/.git # timeout=10
 > git config remote.origin.url <https://github.com/Venkat-Narayan-07/Capstone-Project-II.git> # timeout=10
 Fetching upstream changes from <https://github.com/Venkat-Narayan-07/Capstone-Project-II.git>
 > git --version # timeout=10
 > git --version # 'git version 2.34.1'
 > git fetch --tags --force --progress -- <https://github.com/Venkat-Narayan-07/Capstone-Project-II.git>
 +refs/heads/*:refs/remotes/origin/* # timeout=10

```

> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git config core.sparsecheckout # timeout=10
> git checkout -f f0366e8a941f6431578e03b754f060392417c288 # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git branch -D master # timeout=10
> git checkout -b master f0366e8a941f6431578e03b754f060392417c288 # timeout=10
> git rev-parse --resolve-git-dir /home/ubuntu/jenkins/workspace/job1/.git # timeout=10
> git config remote.origin.url https://github.com/Venkat-Narayan-07/Capstone-Project-II.git # timeout=10
Fetching upstream changes from https://github.com/Venkat-Narayan-07/Capstone-Project-II.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
using GIT_SSH to set credentials
Verifying host key using known hosts file
You're using 'Known hosts file' strategy to verify ssh host keys, but your known_hosts file does not exist,
please go to 'Manage Jenkins' -> 'Security' -> 'Git Host Key Verification Configuration' and configure host
key verification.
> git fetch --tags --force --progress -- https://github.com/Venkat-Narayan-07/Capstone-Project-II.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
Checking out Revision f0366e8a941f6431578e03b754f060392417c288 (refs/remotes/origin/master)
Commit message: "Update Jenkinsfile"
[Pipeline] withEnv
[Pipeline] {
[Pipeline] script
[Pipeline] {

[Pipeline] sh
+ sudo docker build /home/ubuntu/jenkins/workspace/prtwebsite/ -t venkatnarayan16/hshar
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 146.9kB

Step 1/5 : From ubuntu
--> 35a88802559d
Step 2/5 : RUN apt update
--> Using cache
--> 402de2009e6d
Step 3/5 : RUN apt-get install apache2 -y
--> Using cache
--> bcd2c3035568
Step 4/5 : ENTRYPOINT apachectl -D FOREGROUND
--> Using cache
--> 7be6d0ba8b1c
Step 5/5 : ADD . /var/www/html
--> Using cache
--> 0acb334df88d
Successfully built 0acb334df88d
Successfully tagged venkatnarayan16/hshar:latest
Successfully tagged venkatnarayan16/hshar:latest
[Pipeline] sh
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git config core.sparsecheckout # timeout=10
> git checkout -f f0366e8a941f6431578e03b754f060392417c288 # timeout=10
+ sudo echo ****
+ sudo docker login -u venkatnarayan16 --password-stdin
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[Pipeline] sh
+ sudo docker push venkatnarayan16/hshar
Using default tag: latest
The push refers to repository [docker.io/venkatnarayan16/hshar]
2f46104f539a: Preparing
2f71b4f6ef9f: Preparing
325d33a349f4: Preparing
a30a5965a4f7: Preparing
325d33a349f4: Layer already exists
2f71b4f6ef9f: Layer already exists
a30a5965a4f7: Layer already exists
2f46104f539a: Layer already exists

```

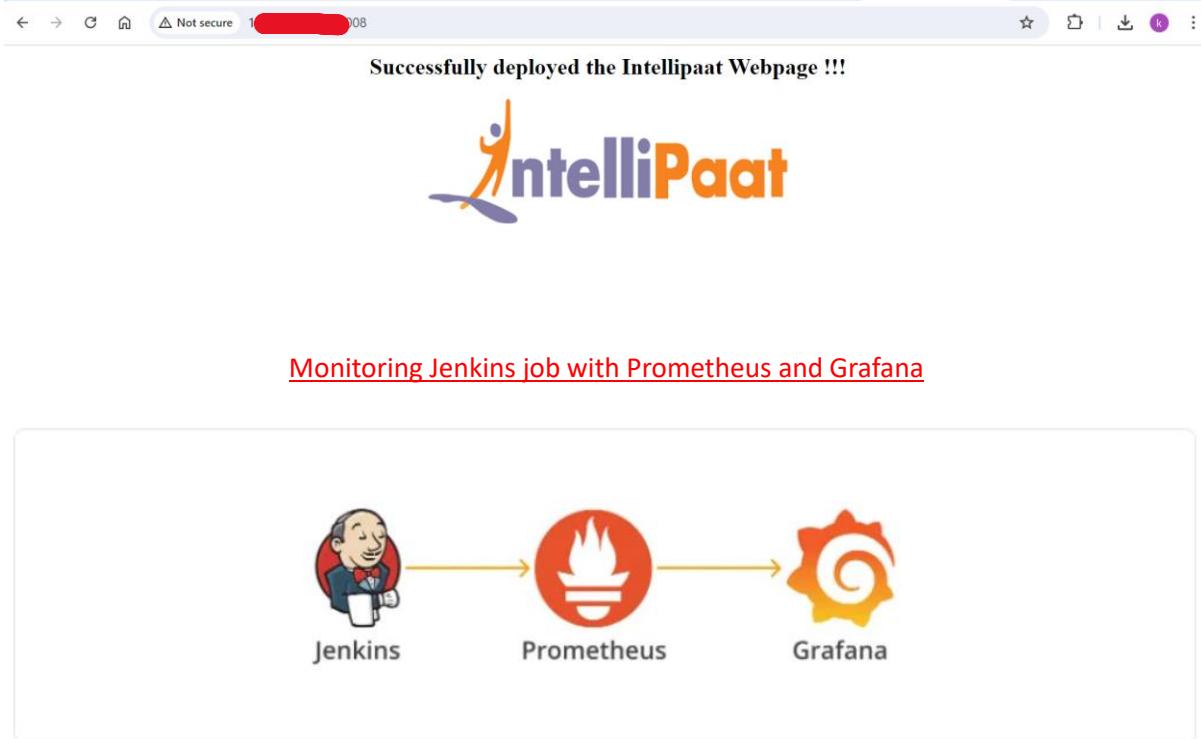
```

latest: digest: sha256:14bc098d76d4a3b69ec7b5e5877ef5468245f335103d354898a0003874034a05 size: 1162
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (k8s)
[Pipeline] node
Running on K8master in /home/ubuntu/jenkins/workspace/job1
[Pipeline] {
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential d0fe9a57-068a-48a6-a956-da9c9eebd01b
Fetching changes from the remote Git repository
Checking out Revision f0366e8a941f6431578e03b754f060392417c288 (refs/remotes/origin/master)
Commit message: "Update Jenkinsfile"
[Pipeline] withEnv
[Pipeline] {
[Pipeline] exec:sh
[Pipeline] {
[Pipeline] sh
+ kubectl apply -f hshare-deployment.yaml
deployment.apps/hshare-deployment unchanged
[Pipeline] sh
> git rev-parse --resolve-git-dir /home/ubuntu/jenkins/workspace/job1/.git # timeout=10
> git config remote.origin.url https://github.com/Venkat-Narayan-07/Capstone-Project-II.git # timeout=10
Fetching upstream changes from https://github.com/Venkat-Narayan-07/Capstone-Project-II.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
using GIT_SSH to set credentials
Verifying host key using known hosts file
You're using 'Known hosts file' strategy to verify ssh host keys, but your known_hosts file does not exist,
please go to 'Manage Jenkins' -> 'Security' -> 'Git Host Key Verification Configuration' and configure host
key verification.
> git fetch --tags --force --progress -- https://github.com/Venkat-Narayan-07/Capstone-Project-II.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git config core.sparsecheckout # timeout=10
> git checkout -f f0366e8a941f6431578e03b754f060392417c288 # timeout=10
+ kubectl apply -f hshare-service.yaml
service/hshare-service unchanged
[Pipeline] }
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] End of Pipeline
Finished: SUCCESS

```

- Whenever there is a push made to the master branch, Jenkins job gets triggered and automatically containerizes the code and deploys replicas of the webpage using docker and Kubernetes respectively.

- Now with the k8s slave machines IP and on port 30008 we can access the webpage



Now download the Prometheus metrics plugin in Jenkins

Install	Name	Released
<input checked="" type="checkbox"/>	Prometheus metrics 780.v7c50a_d288424	26 days ago
<input type="checkbox"/>	Cortex Metrics 1.0.1	3 yr 5 mo ago
<input type="checkbox"/>	Otel agent host metrics monitoring 1.2.2	

Then go to the **Manage Jenkins -> Configure System** page.

Scroll down to the *Prometheus* section. Prometheus works by retrieving data from the endpoint that we specify. In this case, by default Prometheus will retrieve Jenkins metrics from the **/prometheus** endpoint. restart Jenkins.

Dashboard > Manage Jenkins > System >

Path ?
prometheus

Default Namespace ?
default

Enable authentication for prometheus end-point ?

Collecting metrics period in seconds ?
5

Count duration of successful builds ?

 Count duration of unstable builds ?

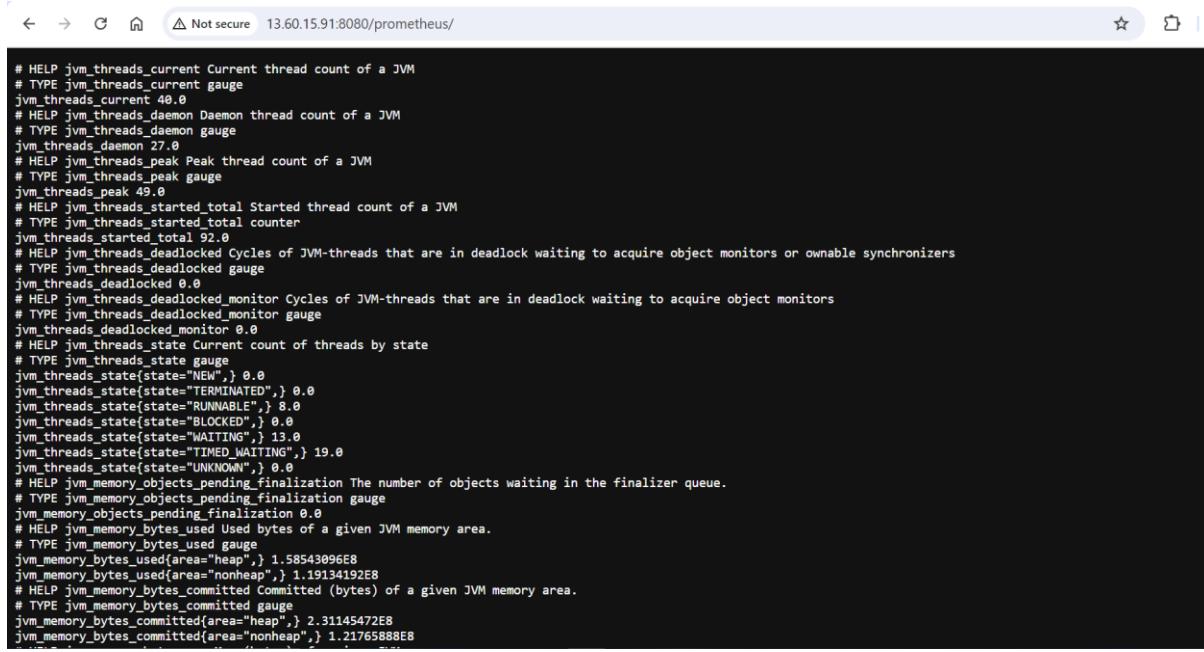
 Count duration of failed builds ?

 Count duration of not-built builds ?

 Count duration of aborted builds ?

Save **Apply**

Add **/Prometheus** suffix to the Jenkins URL. Here we will see a large number of metrics. Prometheus will retrieve Jenkins metrics data from this endpoint



```

# HELP jvm_threads_current Current thread count of a JVM
# TYPE jvm_threads_current gauge
jvm_threads_current 40.0
# HELP jvm_threads_daemon Daemon thread count of a JVM
# TYPE jvm_threads_daemon gauge
jvm_threads_daemon 27.0
# HELP jvm_threads_peak Peak thread count of a JVM
# TYPE jvm_threads_peak gauge
jvm_threads_peak 49.0
# HELP jvm_threads_started_total Started thread count of a JVM
# TYPE jvm_threads_started_total counter
jvm_threads_started_total 92.0
# HELP jvm_threads_deadlocked Cycles of JVM-threads that are in deadlock waiting to acquire object monitors or ownable synchronizers
# TYPE jvm_threads_deadlocked gauge
jvm_threads_deadlocked 0.0
# HELP jvm_threads_deadlocked_monitor Cycles of JVM-threads that are in deadlock waiting to acquire object monitors
# TYPE jvm_threads_deadlocked_monitor gauge
jvm_threads_deadlocked_monitor 0.0
# HELP jvm_threads_state Current count of threads by state
# TYPE jvm_threads_state gauge
jvm_threads_state{state="NEW"} 0.0
jvm_threads_state{state="TERMINATED"} 0.0
jvm_threads_state{state="RUNNABLE"} 8.0
jvm_threads_state{state="BLOCKED"} 0.0
jvm_threads_state{state="WAITING"} 13.0
jvm_threads_state{state="TIMED_WAITING"} 19.0
jvm_threads_state{state="UNKNOWN"} 0.0
# HELP jvm_memory_objects_pending_finalization The number of objects waiting in the finalizer queue.
# TYPE jvm_memory_objects_pending_finalization gauge
jvm_memory_objects_pending_finalization 0.0
# HELP jvm_memory_bytes_used Used bytes of a given JVM memory area.
# TYPE jvm_memory_bytes_used gauge
jvm_memory_bytes_used{area="heap"} 1.58543096E8
jvm_memory_bytes_used{area="nonheap"} 1.19134192E8
# HELP jvm_memory_bytes_committed Committed (bytes) of a given JVM memory area.
# TYPE jvm_memory_bytes_committed gauge
jvm_memory_bytes_committed{area="heap"} 2.31145472E8
jvm_memory_bytes_committed{area="nonheap"} 1.21765888E8

```

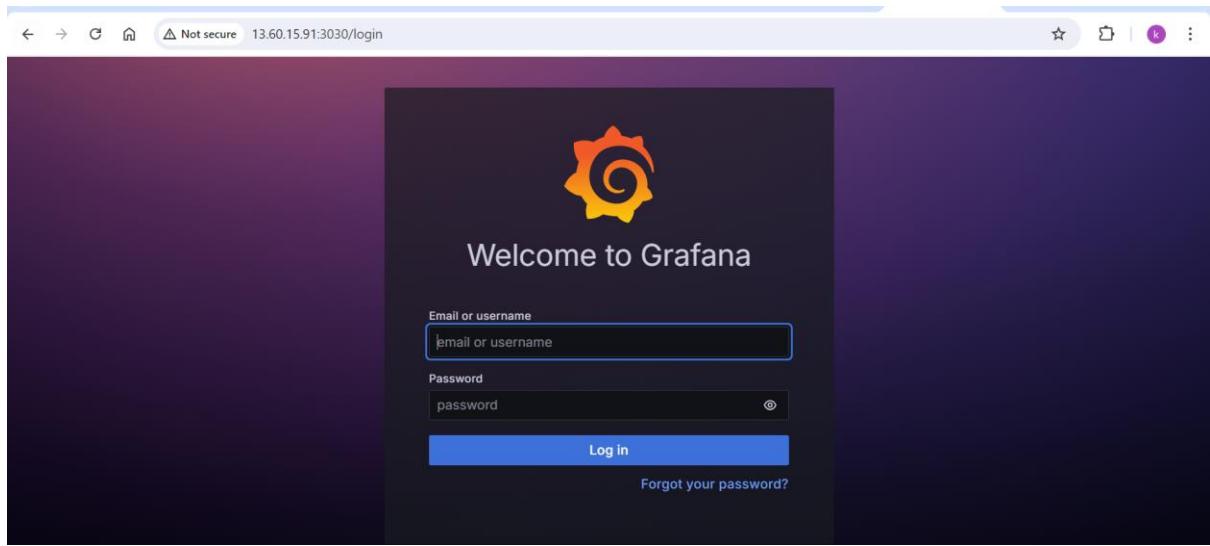
Then goto Jenkins instance and run deploy a prometheus container on port 9090 with prometheus image

```
ubuntu@ip-10-0-1-226:~$ sudo docker run -d --name prometheus -p 9090:9090 prom/prometheus
Unable to find image 'prom/prometheus:latest' locally
latest: Pulling from prom/prometheus
9fa9226be034: Pull complete
1617e25568b2: Pull complete
92ff7cbea015: Pull complete
3e818186829e: Pull complete
e5110b75bf71: Pull complete
154ef881db4f: Pull complete
eaafa8ad3e2d: Pull complete
fea56ff08967: Pull complete
6e62e059c561: Pull complete
443ffcabdce2: Pull complete
d59855f97034: Pull complete
b32c911ealld7: Pull complete
Digest: sha256:cafe963e591c872d38f3ea41ff8eb22cee97917b7c97b5c0ccd43a419f11f613
Status: Downloaded newer image for prom/prometheus:latest
2f8e63ab373a977ce6217b708f1e0d5036daca26433d7ebf1d9057f18cfclfe9
ubuntu@ip-10-0-1-226:~$
```

Similarly, run/deploy Grafana container on port 3030 with grafana image

```
ubuntu@ip-10-0-1-207:~$ sudo docker run -d --name grafana -p 3030:3030 -e "GF_SERVER_HTTP_PORT=3030" grafana/grafana
Unable to find image 'grafana/grafana:latest' locally
latest: Pulling from grafana/grafana
4abcf2066143: Pull complete
b02b4b1ae159: Pull complete
3bb77895c022: Pull complete
06ba99fa00c6: Pull complete
105bf4eba93e: Pull complete
94aeabba6e9f: Pull complete
d5ef5293514d: Pull complete
09419083b5bb: Pull complete
56983880012c: Pull complete
4alff49c903a: Pull complete
Digest: sha256:886b56d5534e54f69a8cfcb4b8928da8fc753178a7a3d20c3f9b04b660169805
Status: Downloaded newer image for grafana/grafana:latest
97fe0aed2a4f6b4445bbb2d40e039ab81f367a5e26d0a78f2f37b97c67b5c4dc
ubuntu@ip-10-0-1-207:~$
```

Access prometheus and grafana on respective ports



By default, Prometheus only exports its own metrics. To monitor the system outside Prometheus, we must use the Prometheus exporter. Prometheus metrics plugin in Jenkins works by exposing metrics to an endpoint (namely `/prometheus` by default) when the Prometheus server can later retrieve data from that endpoint.

we need to configure the Prometheus server so that it can recognize these endpoints.

We can see that in Metrics Explorer various metics are available but Jenkins metrics are not available.

A screenshot of the Prometheus Metrics Explorer interface. The URL in the address bar is 13.60.15.91:9090/graph?g0.expr=&g0.tab=1&g0.display_mode=lines&g0.show_exemplars=0&g0.range_input=1h. The main area shows a list of metrics under the 'Graph' tab. The list includes: go_gc_cycles_automatic_gc_cycles_total, go_gc_cycles_forced_gc_cycles_total, go_gc_cycles_total_gc_cycles_total, go_gc_duration_seconds, go_gc_duration_seconds_count, go_gc_duration_seconds_sum, go_gc_gogc_percent, go_gc_gomemlimit_bytes, go_gc_heap_allocs_by_size_bytes_bucket, go_gc_heap_allocs_by_size_bytes_count, go_gc_heap_allocs_by_size_bytes_sum, and go_gc_heap_allocs_bytes_total. A search bar is at the top, and a 'Remove Panel' button is visible on the right.

Exec into the Prometheus container, open the `prometheus.yml` file.

```
ubuntu@ip-10-0-1-226:~$ sudo docker exec -it prometheus sh  
/prometheus $ vi /etc/prometheus/prometheus.yml
```

Insert Jenkins Job with Jenkins Url as target. Save and exit then restart promethues container.

```

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]

  - job_name: "jenkins"
    metrics_path: /prometheus
    static_configs:
      - targets: ["13.60.15.91:8080"]

```

```

ubuntu@ip-10-0-1-226:~$ sudo docker container restart prometheus
prometheus
ubuntu@ip-10-0-1-226:~$ █

```

Now verify that the Jenkins endpoint is added in status > targets in Prometheus dashboard.

Targets					
jenkins (1/1 up)					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://13.60.15.91:8080/prometheus/	UP	instance="13.60.15.91:8080" job="jenkins"	7.797s ago	13.094ms	

prometheus (1/1 up)					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	13.700s ago	7.536ms	

We can see the Jenkins Metrics in Metrics Explorer

The screenshot shows the Prometheus Metrics Explorer interface. On the left, there are filter options: 'Use local time' (unchecked), 'Enable querier' (unchecked), and a search bar labeled 'Expression (press Shift+Enter)'. Below these are tabs for 'Table' (selected) and 'Graph'. Under 'Table', there are dropdown menus for 'Evaluation time' and 'No data queried yet'. A blue button 'Add Panel' is at the bottom. The main area is titled 'Metrics Explorer' and contains a list of Jenkins metrics:

- default_jenkins_builds_available_builds_count
- default_jenkins_builds_discard_active
- default_jenkins_builds_duration_milliseconds_summary_count
- default_jenkins_builds_duration_milliseconds_summary_created
- default_jenkins_builds_duration_milliseconds_summary_sum
- default_jenkins_builds_health_score
- default_jenkins_builds_last_build_duration_milliseconds
- default_jenkins_builds_last_build_logfile_size_bytes
- default_jenkins_builds_last_build_result
- default_jenkins_builds_last_build_result_ordinal
- default_jenkins_builds_last_build_start_time_milliseconds
- default_jenkins_builds_last_stage_duration_milliseconds_summary_count
- default_jenkins_builds_last_stage_duration_milliseconds_summary_created

A blue button 'Execute' is located in the top right corner, and a link 'Remove Panel' is at the bottom right.

Goto Jenkins portal and trigger Jenkins job to check metrics

The screenshot shows the Jenkins Job1 dashboard. At the top, there's a navigation bar with 'Jenkins', 'Search (CTRL+K)', notifications, and user 'venkat'. Below it, the breadcrumb path is 'Dashboard > job1 >'. The main content area has a tab 'Status' (selected) and a summary: 'job1' (green checkmark), 'End-to-end Jenkins CI/CD pipeline using Docker and Kubernetes'. To the right is an 'Edit description' button. On the left, there's a sidebar with links: 'Changes', 'Build Now', 'Configure', 'Delete Pipeline', 'Stages', 'Rename', 'Pipeline Syntax', and 'GitHub Hook Log'. In the center, under 'Permalinks', there's a list of recent builds:

- Last build (#3), 2 min 38 sec ago
- Last stable build (#3), 2 min 38 sec ago
- Last successful build (#3), 2 min 38 sec ago
- Last failed build (#1), 2 days 1 hr ago
- Last unsuccessful build (#1), 2 days 1 hr ago
- Last completed build (#3), 2 min 38 sec ago

We can see that the pipeline took 2 min21sec to complete the job

The screenshot shows the Jenkins Build #3 details page. The breadcrumb path is 'Dashboard > job1 > #3'. The main title is 'Build #3 (Aug 23, 2024, 9:29:12 AM)'. To the right are buttons for 'Keep this build forever', 'Add description', 'Started 7 min 4 sec ago', and 'Took 2 min 20 sec'. On the left is a sidebar with links: 'Status' (selected), 'Changes', 'Console Output', 'Edit Build Information', 'Delete build #3', 'Timings', 'Git Build Data', 'Pipeline Overview', 'Pipeline Console', 'Restart from Stage', 'Replay', 'Pipeline Steps', and 'Workspaces'. The 'Changes' section lists commits:

- Delete Capstone project II.pdf (details / githubweb)
- Add files via upload (details / githubweb)
- Update README.md (details / githubweb)

The 'Timings' section shows 'Started by user venkat' and 'This run spent:' with items: '1 min 58 sec waiting;', '2 min 20 sec build duration;', and '2 min 21 sec total from scheduled to completion.' At the bottom, there's a 'git' icon, revision '40a87f6a1e30ccb05809b8fce8dbbd314b982688', repository 'https://github.com/Venkat-Narayan-07/Capstone-Project-II.git', and branch 'refs/remotes/origin/master'.

Back to tab Prometheus, by choosing metrics

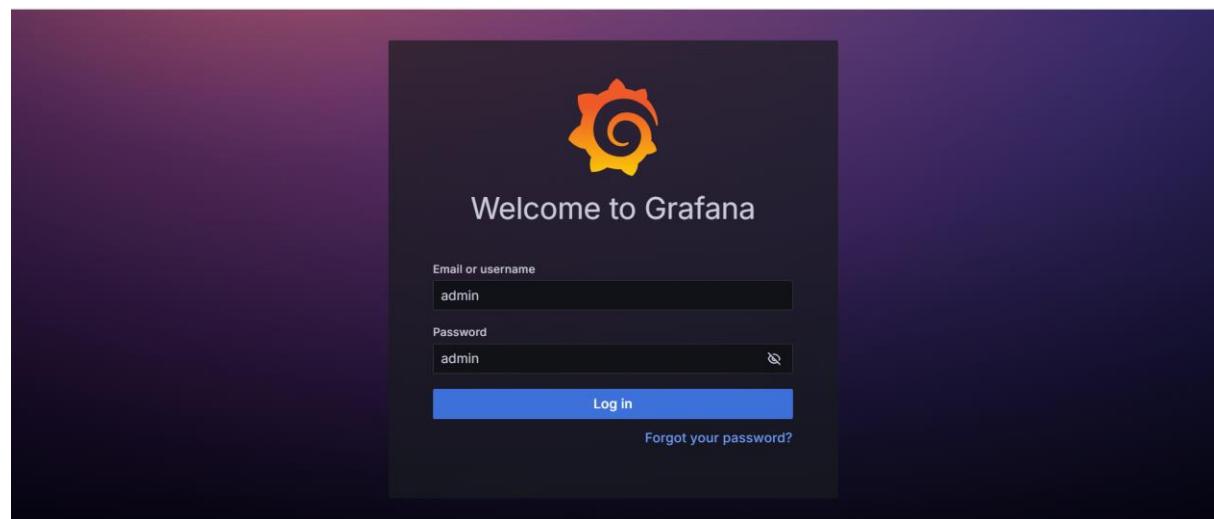
'default_jenkins_builds_last_build_duration_milliseconds' we can see how long the last build duration is in milliseconds.

The screenshot shows the Prometheus web interface at the URL 13.60.15.91:9090/graph?g0.expr=default_jenkins_builds_last_build_duration_milliseconds&g0.tab=1&g0.display_mode=lines&g0.show... The page title is "Prometheus". The search bar contains the query "default_jenkins_builds_last_build_duration_milliseconds". The results are displayed in a table format. The table has two columns: the metric name and its value. The values are: 12462, 22899, 1407, 23621, 471, and 140983. The last value, 140983, corresponds to the Jenkins job "job1". The table includes navigation arrows for evaluation time and a "Remove Panel" button.

default_jenkins_builds_last_build_duration_milliseconds{buildable="true", instance="13.60.15.91:8080", jenkins_job="prtwebsite", job="jenkins", repo="NA"}	12462
default_jenkins_builds_last_build_duration_milliseconds{buildable="true", instance="13.60.15.91:8080", jenkins_job="jenkinsfile", job="jenkins", repo="NA"}	22899
default_jenkins_builds_last_build_duration_milliseconds{buildable="true", instance="13.60.15.91:8080", jenkins_job="simple-node-js-react-npm-app-deirect script", job="jenkins", repo="NA"}	1407
default_jenkins_builds_last_build_duration_milliseconds{buildable="true", instance="13.60.15.91:8080", jenkins_job="web", job="jenkins", repo="NA"}	23621
default_jenkins_builds_last_build_duration_milliseconds{buildable="true", instance="13.60.15.91:8080", jenkins_job="simple-node-js-react-npm-app", job="jenkins", repo="NA"}	471
default_jenkins_builds_last_build_duration_milliseconds{buildable="true", instance="13.60.15.91:8080", jenkins_job="job1", job="jenkins", repo="NA"}	140983

last build duration is 140983 miliseconds = 140.9 seconds = 2.35 minutes. It's the same with the result from Jenkins.

Go to page Grafana in <localhost>:3030. In the login page, give username and password as **admin**.



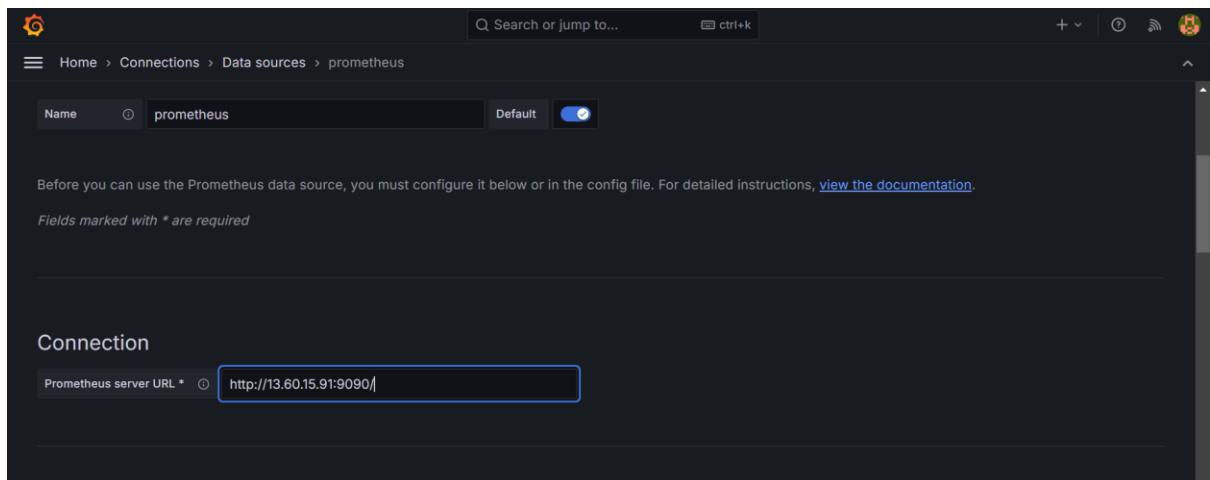
The screenshot shows the Grafana Home page. At the top, there's a search bar and navigation links for Documentation, Tutorials, Community, and Public Slack. Below the header, a "Welcome to Grafana" message is displayed. On the left, a "Basic" section provides steps for setting up Grafana. To the right, there are three main sections: "TUTORIAL" (with "DATA SOURCE AND DASHBOARDS" and "Grafana fundamentals" sub-sections), "DATA SOURCES" (with "Add your first data source" and a "Learn how in the docs" link), and "DASHBOARDS" (with "Create your first dashboard" and a "Learn how in the docs" link). A "Remove this panel" button is located in the top right corner of the main content area.

Create a Data Source. From the Home page, click add your first data source. Choose option Prometheus

The screenshot shows the "Add data source" page in Grafana. The URL in the browser is "Home > Connections > Data sources > Add data source". The page title is "Add data source" and it says "Choose a data source type". There is a search bar labeled "Q. Filter by name or type" and a "Cancel" button. Below this, there is a section titled "Time series databases" with three options: "Prometheus" (selected, shown with its logo and description "Open source time series database & alerting Core"), "Graphite" (shown with its logo and description "Open source time series database Core"), and "InfluxDB" (partially visible).

Fill

- Name: Jenkins-Prometheus
- URL : <IP_Adress>:9090



create a Grafana dashboard. Scroll up and click **build a dashboard -> Create your first dashboard -> Add visualization**

Type: Prometheus

Alerting Supported

Explore data Build a dashboard

Settings Dashboards

Start your new dashboard by adding a visualization

Select a data source and then query and visualize your data with charts, stats and tables or create lists, markdowns and other widgets.

+ Add visualization

configure

- Title: Last build duration
- Description: Last build duration
- Transparent background: Active
- Graph mode: None

- Metric: default_jenkins_builds_last_build_duration_milliseconds

Click button Run queries. Save the panel by click Save

The screenshot shows the Grafana interface for creating a new dashboard. A single panel is visible, titled "Last build duration". The panel configuration includes a "Metric" dropdown set to "default_jenkins_builds_last_build_duration_milliseconds", a "Label filters" section, and a "Run queries" button. On the right side, the "Panel options" sidebar is expanded, showing the panel's title, description, and a tooltip section.

We can see the Jenkins metrics

Time	{__name__="default_jenkins_builds_last_bu...}
2024-08-23 14:54:30	22899
2024-08-23 14:54:45	22899
2024-08-23 14:55:00	22899
2024-08-23 14:55:15	22899
2024-08-23 14:55:30	22899

Similarly We can get dashboard too from a template. Search on google Grafana Dashboard/Grafana labs then find Jenkins by Prometheus in Data Source.

Back to dashboard we created before. In the right menu click New -> Import

Paste ID from Grafana Dashboard the click Load

Import dashboard from file or Grafana.com

Upload dashboard JSON file

Drag and drop here or click to browse
Accepted file types: json, txt

Find and import dashboards for common applications at grafana.com/dashboards

9964 | Load

Import via dashboard JSON model

```
{  
  "title": "Example - Repeating Dictionary variables",  
  "uid": "_0HnEoN4z",  
  "panels": [...]  
}
```

In the next page, in data source, choose Prometheus — Jenkins. Click **Import**

Updated on 2023-08-24 15:04:53

Options

Name Jenkins: Performance and Health Overview

Folder Dashboards

Unique identifier (UID)
The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.

haryan-jenkins Change uid

Prometheus

Import Cancel

Now we will see various metrics in Grafana

The screenshot shows the Grafana interface for managing dashboards. At the top, there's a search bar labeled "Search or jump to..." and a "ctrl+k" hotkey indicator. Below it, a breadcrumb navigation shows "Home > Dashboards". The main title is "Dashboards" with the subtitle "Create and manage dashboards to visualize your data". A search bar "Search for dashboards and folders" is followed by a "Filter by tag" dropdown and a "Starred" checkbox. On the right, there are icons for creating a new dashboard and managing existing ones. The main area lists dashboards with columns for "Name" and "Tags". There are four entries:

Name	Tags
Jenkins: Performance and Health Overview	
Last Build Duration	
Last Build Duration	

Successfully configured monitoring Jenkins with Prometheus and Grafana!

