

Summary:

- Infrastructure creation.
- Pipeline creation.
- Deployment.
- Nginx and DNS Configuration.

The source code and the required files are available in the repo:

https://github.com/Murali90102/sample_node_app.git.

First of all, clone the files in the repository to the system.

Open the Terraform folder in a terminal and export the user details.

Infrastructure Setup:

Create an IAM user, give permission policies for creating EC2, VPC, Subnets, Route tables, Security groups and export all the access keys.

```
export AWS_ACCESS_KEY_ID=key
```

```
export AWS_SECRET_ACCESS_KEY=key
```

Now run the Terraform commands

1.terraform init

```
kittu@MuraliKrishna MINGW64 ~/Downloads/sample_node_app/Infra_Terraform (main)
$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/local from the dependency lock file
- Reusing previous version of hashicorp/http from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/tls from the dependency lock file
- Using previously-installed hashicorp/local v2.5.1
- Using previously-installed hashicorp/http v3.4.2
- Using previously-installed hashicorp/aws v4.67.0
- Using previously-installed hashicorp/tls v4.0.5

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.
```

2.terraform plan

```
MINGW64/C:/Users/kittu/Downloads/sample_node_app/Infra_Terraform/its
kittu@Muralikrishna MINGW64 ~/Downloads/sample_node_app/Infra_Terraform (main)
$ terraform plan
data.http.workstation-external-ip: Reading...
data.http.workstation-external-ip: Read complete after 1s [id=http://ipv4.icanhazip.com]
data.aws_availability_zones.az: Reading...
data.aws_availability_zones.az: Read complete after 1s [id=us-east-1]

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_eip.natgw_elastic_ip will be created
+ resource "aws_eip" "natgw_elastic_ip" {
+   allocation_id      = (known after apply)
+   association_id     = (known after apply)
+   carrier_ip         = (known after apply)
+   customer_owned_ip  = (known after apply)
+   domain             = (known after apply)
+   id                 = (known after apply)
+   instance           = (known after apply)
+   network_border_group = (known after apply)
+   network_interface  = (known after apply)
+   private_dns        = (known after apply)
+   private_ip         = (known after apply)
+   public_dns         = (known after apply)
+   public_ip          = (known after apply)
+   public_ipv4_pool   = (known after apply)
+   tags_all           = (known after apply)
+   vpc                = true
}

# aws_instance.jenkins-ec2[0] will be created
+ resource "aws_instance" "jenkins-ec2" {
+   ami              = "ami-0c1704bac156af62c"
+   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     = false
+   get_password_data = (known after apply)
+   host_id           = (known after apply)
+   host_resource_group_arn = (known after apply)
+   iam_instance_profile = (known after apply)
+   id               = (known after apply)
+   instance_initiated_shutdown_behavior = (known after apply)
+   instance_state    = (known after apply)
+   instance_type     = "t2.medium"
+   ipv6_address_count = (known after apply)
+   ipv6_addresses    = (known after apply)
+   key_name          = "dev-ssh-key"
+   monitoring        = (known after apply)
+   outpost_arn       = (known after apply)
+   password_data     = (known after apply)
+   placement_group    = (known after apply)
+   placement_partition_number = (known after apply)
}
```

3.terraform apply

```
MINGW64/C:/Users/kittu/Downloads/sample_node_app/Infra_Terraform/its
kittu@Muralikrishna MINGW64 ~/Downloads/sample_node_app/Infra_Terraform (main)
$ terraform apply
data.http.workstation-external-ip: Reading...
data.http.workstation-external-ip: Read complete after 0s [id=http://ipv4.icanhazip.com]
data.aws_availability_zones.az: Reading...
data.aws_availability_zones.az: Read complete after 2s [id=us-east-1]

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_eip.natgw_elastic_ip will be created
+ resource "aws_eip" "natgw_elastic_ip" {
+   allocation_id      = (known after apply)
+   association_id     = (known after apply)
+   carrier_ip         = (known after apply)
+   customer_owned_ip  = (known after apply)
+   domain             = (known after apply)
+   id                 = (known after apply)
+   instance           = (known after apply)
+   network_border_group = (known after apply)
+   network_interface  = (known after apply)
+   private_dns        = (known after apply)
+   private_ip         = (known after apply)
+   public_dns         = (known after apply)
+   public_ip          = (known after apply)
+   public_ipv4_pool   = (known after apply)
+   tags_all           = (known after apply)
+   vpc                = true
}

# aws_instance.jenkins-ec2[0] will be created
+ resource "aws_instance" "jenkins-ec2" {
+   ami              = "ami-0c1704bac156af62c"
+   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     = false
+   get_password_data = (known after apply)
+   host_id           = (known after apply)
+   host_resource_group_arn = (known after apply)
+   iam_instance_profile = (known after apply)
+   id               = (known after apply)
+   instance_initiated_shutdown_behavior = (known after apply)
+   instance_state    = (known after apply)
+   instance_type     = "t2.medium"
+   ipv6_address_count = (known after apply)
+   ipv6_addresses    = (known after apply)
+   key_name          = "dev-ssh-key"
+   monitoring        = (known after apply)
+   outpost_arn       = (known after apply)
+   password_data     = (known after apply)
+   placement_group    = (known after apply)
+   placement_partition_number = (known after apply)
}
```

After the execution of terraform apply command, the infrastructure is created and you can find the I.P's of the EC2 that were created.

```
MINGW64~/c:/Users/kittu/Downloads/sample_node_app/Infra_Terraform/tls
aws_instance.public-ec2[0]: Creation complete after 1s [id=rtbassoc-092f05efcc02567b0]
aws_route.table-association.rta-sub[1]: Creation complete after 1s [id=rtbassoc-00f36befaabd2a38a]
aws_nat_gateway.nat_gw: Still creating... [10s elapsed]
aws_instance.jenkins-ec2[0]: Still creating... [10s elapsed]
aws_instance.public-ec2[0]: Still creating... [10s elapsed]
aws_instance.public-ec2[0]: Still creating... [20s elapsed]
aws_nat_gateway.nat_gw: Still creating... [20s elapsed]
aws_instance.jenkins-ec2[0]: Still creating... [20s elapsed]
aws_instance.public-ec2[0]: Creation complete after 26s [id=i-0f48d4d797f173b8f]
aws_instance.jenkins-ec2[0]: Still creating... [30s elapsed]
aws_nat_gateway.nat_gw: Still creating... [30s elapsed]
aws_nat_gateway.nat_gw: Still creating... [40s elapsed]
aws_instance.jenkins-ec2[0]: Still creating... [40s elapsed]
aws_instance.jenkins-ec2[0]: Creation complete after 47s [id=i-056ff15d2a3b51c52]
aws_nat_gateway.nat_gw: Still creating... [50s elapsed]
aws_nat_gateway.nat_gw: Still creating... [1m0s elapsed]
aws_nat_gateway.nat_gw: Still creating... [1m10s elapsed]
aws_nat_gateway.nat_gw: Still creating... [1m20s elapsed]
aws_nat_gateway.nat_gw: Still creating... [1m30s elapsed]
aws_nat_gateway.nat_gw: Creation complete after 1m39s [id=nat-0eeef8f1efcd4a30]
aws_route.private_nat_gateway[0]: Creating...
aws_route.private_nat_gateway[1]: Creating...
aws_route.private_nat_gateway[0]: Creation complete after 2s [id=r-rtb-04dee9e91d8af96bc1080289494]
aws_route.private_nat_gateway[1]: Creation complete after 2s [id=r-rtb-018c6efed6faf003b1080289494]

Warning: Deprecated attribute
  on get_external_ip.tf line 6, in locals:
   6: workstation_external_ip = "${chomp(data.http.workstation-external-ip.body)}/32"
The attribute "body" is deprecated. Refer to the provider documentation for details.

Apply complete! Resources: 28 added, 0 changed, 0 destroyed.

Outputs:
jenkins-ip = "54.224.185.16"
ec2_public_ip = [
  "50.17.117.9",
]
local_workstation_ip = "43.227.131.160/32"
private-ec2-ip = []

kittu@MuraliKrishna MINGW64 ~/Downloads/sample_node_app/Infra_Terraform (main)
$ ls
ec2.tf      outputs.tf  provider.tf  terraform.tfstate  tls/        vpc.tf
get_external_ip.tf  private_subnets.tf  public_subnets.tf  terraform.tfstate.backup  variables.tf

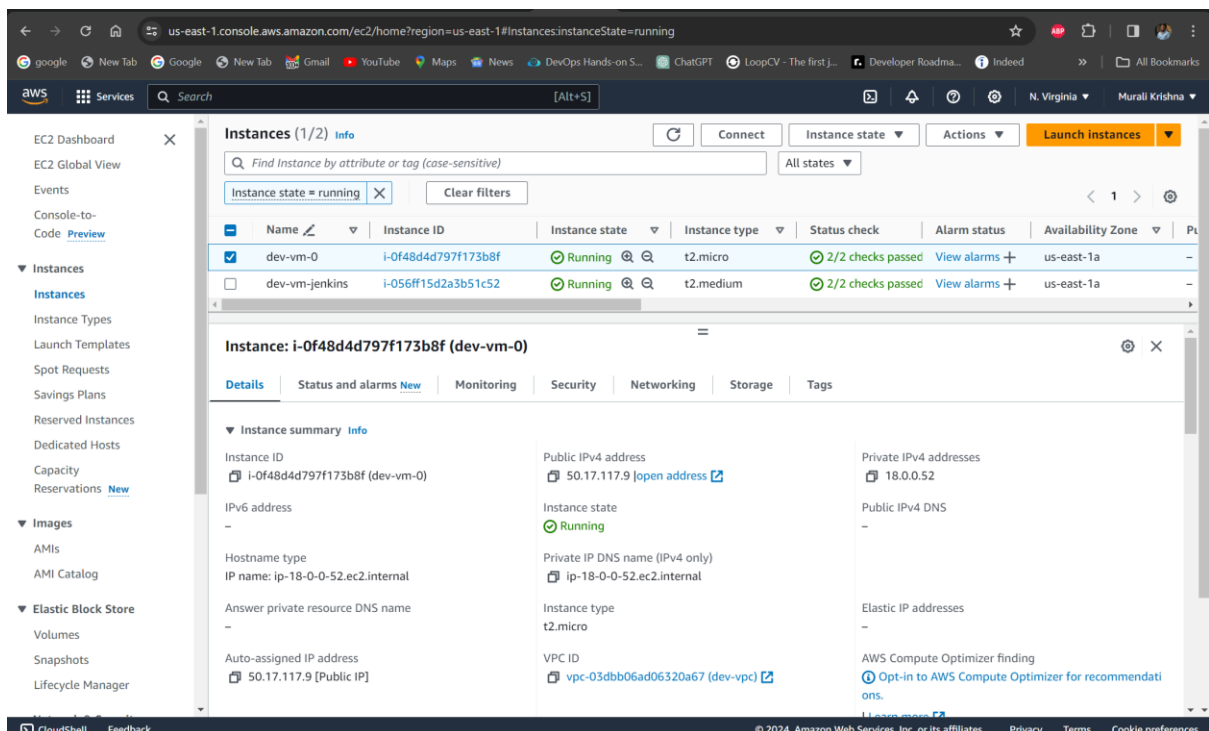
kittu@MuraliKrishna MINGW64 ~/Downloads/sample_node_app/Infra_Terraform (main)
$ cd tls

kittu@MuraliKrishna MINGW64 ~/Downloads/sample_node_app/Infra_Terraform/tls (main)
$ ls
private.pem

kittu@MuraliKrishna MINGW64 ~/Downloads/sample_node_app/Infra_Terraform/tls (main)
$ cat private.pem
-----BEGIN RSA PRIVATE KEY-----
```

In the tls directory, we can find a pem key which can be used for our purpose.

In AWS, we can find all the infrastructure that was created by Terraform. Verify all the services were created.



The image displays two screenshots of the AWS Management Console, specifically the VPC details page for the VPC with ID `vpc-03dbb06ad06320a67` in the `us-east-1` region.

Top Screenshot: Shows the VPC details page. The VPC is named `dev-vpc`. The details section shows the VPC ID, State (Available), DNS hostnames (Disabled), DNS resolution (Enabled), Tenancy (Default), DHCP option set (`dopt-080bd272f09f8decd`), Main route table (`rtb-01391955851652436`), Main network ACL (`acl-0be1d68c76bf4cc83`), Default VPC (No), IPv4 CIDR (`18.0.0.0/16`), IPv6 pool (None), IPv6 CIDR (Network border group), Network Address Usage metrics (Disabled), Route 53 Resolver DNS Firewall rule groups (None), and Owner ID (`339712882094`).

Bottom Screenshot: Shows the Resource map for the VPC. The map displays the VPC, Subnets (4), Route tables (5), and Network connections (2). The subnets are `us-east-1a` (containing `dev-public-subnet-0` and `dev-private-subnet-0`) and `us-east-1b` (containing `dev-public-subnet-1` and `dev-private-subnet-1`). The route tables are `dev-private-route-table-1`, `dev-public-route-table-0`, `rtb-01391955851652436`, `dev-private-route-table-0`, and `dev-public-route-table-1`. The network connections are `dev-igw` and `dev-nat-gw`.

Now, to create the pipeline, log into the EC2 that was created for Jenkins using the pem.key in `tls` directory.

Pipeline Creation:

```
root@ip-18-0-0-167: ~
$ ssh -i ~/Downloads/sample_node_app/Infra_Terraform/tls/private.pem ubuntu@54.224.185.16
The authenticity of host '54.224.185.16 (54.224.185.16)' can't be established.
ED25519 key fingerprint is SHA256:27XwSrQq8gR0WjSUVVWc1X12bWv5d02tchKw48DMWqy.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.224.185.16' (ED25519) to the list of known hosts.
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-1020-aws x86_64)

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

System information as of Mon Apr  8 03:40:40 UTC 2024

System load:  0.19          Processes:           113
Usage of /:   38.6% of 7.57GB Users logged in:          0
Memory usage: 37%          IPv4 address for docker0: 172.17.0.1
Swap usage:   0%           IPv4 address for eth0:   18.0.0.167

195 updates can be applied immediately.
148 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

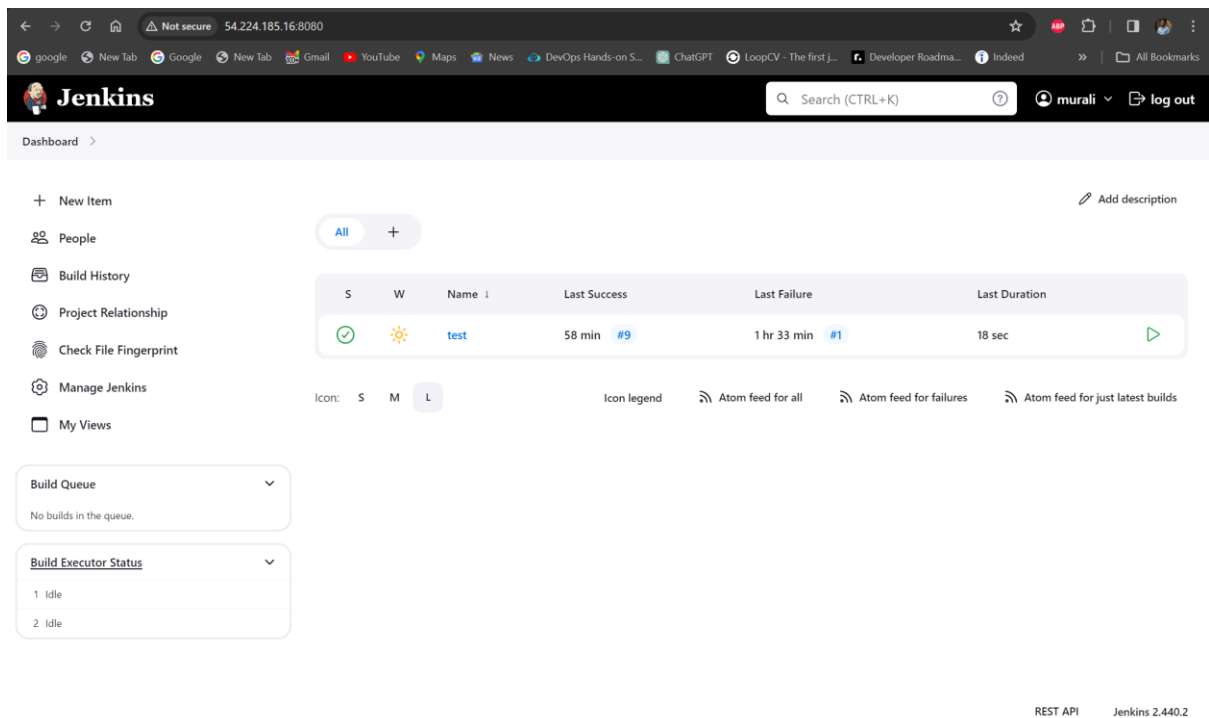
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-18-0-0-167:~$ sudo -i
root@ip-18-0-0-167:~# cat /var/lib/jenkins/secrets/initialAdminPassword
d11d0e365a6418f8f80bb3ef0b016
root@ip-18-0-0-167:~# systemctl jenkins restart
Unknown operation jenkins.
root@ip-18-0-0-167:~# systemctl restart jenkins
root@ip-18-0-0-167:~# |
```

Install the plugins and get into the Jenkins Dashboard.



Create a project and use the Jenkinsfile given in the repository for pipeline script.

Dashboard > test > Configuration

Configure

- General
- Advanced Project Options
- Pipeline**

Definition
Pipeline script from SCM

SCM ?
Git

Repositories ?

Repository URL ?
https://github.com/Murali90102/sample_node_app.git

Credentials ?
Murali90102/* (githubCreds)

+ Add

Advanced

Add Repository

Branches to build ?

Save Apply

Dashboard > test > Configuration

Configure

- General
- Advanced Project Options
- Pipeline**

Repository browser ?
(Auto)

Additional Behaviours

≡ Polling ignores commits from certain users

Excluded Users ?
jenkins_build_user

Add

Script Path ?
Jenkinsfile

☒ Lightweight checkout ?

Pipeline Syntax

Save Apply

Exclude the User jenkins_build_user as the pipeline will create a commit when it passes the docker build stage, it will make changes in the docker-compose file. We don't want the pipeline to get triggered when that commit happens. So, we ignore the commits done by the pipeline which are named under jenkins_build_user.

We want to trigger the pipeline build process only when a certain push is happened in the github repository by a developer or user. So, I checked the GitHub hook trigger for GIT SCM polling and configured it with GitHub webhook.

Steps to create webhook:

1. Open the GitHub repository.

2.Settings→Webhooks→Add webhook

3.Add the Jenkins URL in the payload URL and add /github-webhook/ at the end.

Example: <http://54.224.185.16:8080/github-webhook/>

This will help to trigger the pipelines in Jenkins when a commit is happened in the GitHub repository.

Configuration of Credentials:

Go back to the Jenkins Dashboard and add the credentials of GitHub and Docker Hub in the Configure Credentials tab with the ID's that are provided in the Jenkins pipeline.

Jenkins Search (CTRL+K) murali log out

Dashboard > Manage Jenkins > Credentials

Credentials

T	P	Store	Domain	ID	Name
		System	(global)	githubCreds	Murali90102/***** (githubCreds)
		System	(global)	dockerHubCreds	murali90102/***** (dockerHubCreds)

Stores scoped to Jenkins

P	Store	Domains
	System	(global)

Icon: S M L

REST API Jenkins 2.440.2

Now, configure the VM to host the application. Install the Publish over SSH plugin in Jenkins.

Go to the Jenkins Dashboard→Manage Jenkins→System configuration→System.

Enter the name, private IP, username and password.

Dashboard > Manage Jenkins > System >

SSH Servers

SSH Server

Name ?

vm

Hostname ?

18.0.0.52

Username ?

ubuntu

Remote Directory ?

☐ Avoid sending files that have not changed ?

Advanced ▾ Edited

Success

Test Configuration

Save

Apply

The setup that is required to do the task is configured and ready to go.

I will just commit in the github repo and it will trigger the build pipeline in Jenkins by using GitHub webhook.

github.com/Murali90102/sample_node_app

Murali90102 / sample_node_app

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

sample_node_app Public

main 1 Branch 0 Tags

Go to file Add file <> Code

jenkins_build_user	Updated docker-compose.yaml from CI	dc1c5be · now	77 Commits
Infra_Terraform	updated instance conf, added Jenkinsfile	2 days ago	
Dockerfile	Updated Dockerfile	2 days ago	
Jenkinsfile	webhook ignore working	yesterday	
README.md	Update README.md	now	
app.js	Update app.js	yesterday	
docker-compose.yaml	Updated docker-compose.yaml from CI	now	
package-lock.json	Node.js Sample App added	2 days ago	
package.json	Node.js Sample App added	2 days ago	

README

sample_node_app

About

No description, website, or topics provided.

Readme

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Languages

HCL 97.4% JavaScript 1.7% Dockerfile 0.9%

I have updated the README.md file and it triggered the pipeline in Jenkins.

The screenshot shows the Jenkins 'test' pipeline in 'Stage View'. The pipeline is successful, indicated by a green checkmark. The left sidebar contains navigation options: Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Rename, Pipeline Syntax, and GitHub Hook Log. The main area displays a table of stage execution times for the last four builds.

	Declarative: Checkout SCM	SCM Checkout	Docker build	DockerHub_login	Push2DockerHub	Deploy to VM
Average stage times: (Average full run time: ~18s)	336ms	459ms	2s	454ms	3s	10s
#10 Apr 08 11:46 2 commits	426ms	501ms	1s	444ms	3s	13s
#9 Apr 08 10:02 2 commits	240ms	358ms	1s	431ms	3s	11s
#8 Apr 08 10:00 1 commit	224ms	316ms	1s	419ms	3s	11s
#7 Apr 08 10:00 2 commits	214ms	412ms	1s	426ms	3s	11s

The screenshot shows the Jenkins 'Console Output' for build #10. The left sidebar includes options: Status, Changes, Console Output, View as plain text, Edit Build Information, Delete build '#10', Polling Log, Git Build Data, Restart from Stage, Replay, Pipeline Steps, Workspaces, and Previous Build. The console output text is as follows:

```
Started by GitHub push by Murali90102
Obtained Jenkinsfile from git https://github.com/Murali90102/sample_node_app.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/test
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential githubCreds
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/test/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Murali90102/sample_node_app.git # timeout=10
Fetching upstream changes from https://github.com/Murali90102/sample_node_app.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
using GIT_ASKPASS to set credentials githubCreds
> git fetch --tags --force --progress -- https://github.com/Murali90102/sample_node_app.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision bba11f4be6ec17c929ad2aa9245baa75ff74f483 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f bba11f4be6ec17c929ad2aa9245baa75ff74f483 # timeout=10
Commit message: "Update README.md"
+ git rev 11:46:00.000000000Z # timeout=10
+ git rev 11:46:00.000000000Z # timeout=10
```

The console output states that pipeline was started by GitHub push. The pipeline passed all the stages and it was SUCCESS.

Deployment verification:

To verify, we'll log into the VM to host the application.

```
root@ip-18-0-0-52: ~  
kittu@MuraliKrishna MINGW64 ~ (master)  
$ ssh -i ~/Downloads/sample_node_app/infra/Terraform/tls/private.pem ubuntu@50.17.117.9  
The authenticity of host '50.17.117.9 (50.17.117.9)' can't be established.  
ED25519 key fingerprint is SHA256:hsq1Tdjl5Jf9N67Y/yplsmYRd6lFkfk/nq0lDtx6c8U.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '50.17.117.9' (ED25519) to the list of known hosts.  
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-1020-aws x86_64)  
  
 * Documentation:  https://help.ubuntu.com  
 * Management:    https://landscape.canonical.com  
 * Support:       https://ubuntu.com/advantage  
  
System information disabled due to load higher than 1.0  
  
 * Ubuntu Pro delivers the most comprehensive open source security and  
   compliance features.  
   https://ubuntu.com/aws/pro  
  
72 updates can be applied immediately.  
6 of these updates are standard security updates.  
To see these additional updates run: apt list --upgradable  
  
New release '22.04.3 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
*** System restart required ***  
ubuntu@ip-18-0-0-52:~$ sudo -i  
root@ip-18-0-0-52:~# docker ps  
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES  
60507cb75261   murali90102/sample_node_app:10     "docker-entrypoint.s..." 6 minutes ago  Up 6 minutes  0.0.0.0:3000->3000/tcp, :::3000->3000/tcp  sample_node_app  
root@ip-18-0-0-52:~# curl localhost:3000  
root@ip-18-0-0-52:~# curl localhost:3000  
root@ip-18-0-0-52:~# curl localhost:3000  
Hello world!root@ip-18-0-0-52:~#
```

The docker-compose.yaml file has been executed and the docker image has been pulled from the dockerhub. Application is running successfully.

Now, edit the nginx sites-enabled default file with the server details and enable it to access on port 80.

Commands:

1. `vi /etc/nginx/sites-enabled/default`

Paste the following in the default file

```
server {  
  
    listen 80;  
  
    listen [::]:80 ;  
  
    server_name test.domain_name.in;  
  
    location / {  
  
        # First attempt to serve request as file, then
```

as directory, then fall back to displaying a 404.

```
proxy_pass http://localhost:3000;
```

```
}
```

```
}
```

#Replace the domain with the required name.

2. certbot --nginx -d test.domain.in #replace domain name
3. systemctl restart nginx

The application should be accessible in the browser with the domain name.