

HOW TO DEPLOY ONE SAMPLE STATIC APPLICATION ON TOP OF K8S CLUSTER By USING EC2.

➔ First, we have to create Security group for static application.

SSH-----22-----MY-IP-----This for admin Purpose

ALL-TCP----- 0-65535-----anywhere (0.0.0.0) -----This for End-user purpose

➔ First, we have to login to the console -----Click to EC2 Dashboard -----Click on Create instance -----After that we can pass name of an instance, AMI and Storage.

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags [Info](#)

Name
 [Add additional tags](#)

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

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Recents | **Quick Start**

Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux

Summary

Number of instances [Info](#)

Software Image (AMI)
Amazon Linux 2023 AMI 2023.3.2...[read more](#)
ami-07b6f6261f14c3a45

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

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

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

[Cancel](#) [Launch instance](#)
[Review commands](#)

➔ After we Click on Existing Key-pair and -----Click on Existing Security_Group.

➔ After That Click on Create instance -----The instance is Created .

Instances (1) [Info](#)

[All states](#)

[Clear filters](#)

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Ela
<input type="checkbox"/>	cluster-name	i-0ddc0efa1bd4630b2	Running	t2.micro	2/2 checks passed	View alarms	us-west-2a	ec2-35-162-51-248.us-...	35.162.51.248	-

➔ After We Connect to the Application we Install pre-request Commands.

Ssh -I <pem.file> user_name@public_ip

```
duggi@pavansiva MINGW64 ~/Downloads
$ ssh -i "node.pem" ec2-user@ec2-35-162-51-248.us-west-2.compute.amazonaws.com
#_
~\##### Amazon Linux 2023
~~\#####\
~~\###|
~~\#/ https://aws.amazon.com/linux/amazon-linux-2023
~~V~'-'>
~~~
~~.-.
~/m/'-/
```

Last login: Tue Mar 19 15:51:46 2024 from 104.28.220.170
[ec2-user@ip-172-31-26-190 ~]\$ sudo su -
Last login: Sun Mar 17 17:29:38 UTC 2024 on pts/1
[root@ip-172-31-26-190 ~]# |

- ## 1. GIT Installation Commands

```
yum install git -y
```

- ## 2. Terraform Installation Commands

```
wget https://releases.hashicorp.com/terraform/1.3.7/terraform_1.3.7_linux_amd64.zip
```

Is

```
unzip terraform_1.3.7_linux_amd64.zip
```

```
mv terraform /usr/local/bin
```

```
terraform -v
```

- ### 3. Kubectl Installation Commands

```
curl -LO https://dl.k8s.io/release/\$\(curl -L -s https://dl.k8s.io/release/stable.txt\)/bin/linux/amd64/kubectl
```

```
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

```
chmod +x kubectl
mkdir -p ~/.local/bin
mv ./kubectl ~/.local/bin/kubectl
```

```
kubectl version
```

- #### 4. IAM ROLE

Policy you given Eks full access after we attached to the role this ec2 machine.

Select Ec2-Machine-----Click It Action -----Security-----Modify IAM Role---Attached That iam role.

- ## 5. Docker Installation Commands

```
yum install docker -y
```

```
systemctl start docker
```

```
systemctl enable docker.
```

➔ For Cluster Created By using terraform. For that I am Passing Github url in the below.

<https://github.com/sivaram2662/eks-cluster-terraform-code.git>

- ➔ After That we clone Github url on top of Ec2 machine BY using below command.

Git clone <https://github.com/sivaram2662/eks-cluster-terraform-code.git>

```
[root@ip-172-31-26-190 ~]# ls
eks-cluster-terraform-code  terraform_1.3.7_linux_amd64.zip
[root@ip-172-31-26-190 ~]# |
```

- ➔ After we enter to the that folder by using below command

```
Cd eks-cluster-terraform-code
ls
```

```
[root@ip-172-31-26-190 ~]# ls
eks-cluster-terraform-code  terraform_1.3.7_linux_amd64.zip
[root@ip-172-31-26-190 ~]# cd eks-cluster-terraform-code/
[root@ip-172-31-26-190 eks-cluster-terraform-code]# ls
LICENSE README.md main.tf outputs.tf terraform.tf variables.tf
[root@ip-172-31-26-190 eks-cluster-terraform-code]#
```

- ➔ After that we will Follow steps

1. Terraform init

Scan the *.tf files and finds provider and download provider plugin From hashicorp website . After Terraform init we will get File like [terraform.lock.hcl](#)

```
[root@ip-172-31-26-190 eks-cluster-terraform-code]# terraform init
Initializing modules...

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/cloudinit from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/kubernetes from the dependency lock file
- Reusing previous version of hashicorp/random from the dependency lock file
- Reusing previous version of hashicorp/tls from the dependency lock file
- Using previously-installed hashicorp/aws v4.47.0
- Using previously-installed hashicorp/kubernetes v2.17.0
- Using previously-installed hashicorp/random v3.4.3
- Using previously-installed hashicorp/tls v4.0.4
- Using previously-installed hashicorp/cloudinit v2.2.0

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
```

2. Terraform validate

It will be checking whether the syntax is correct or not

```
[root@ip-172-31-26-190 eks-cluster-terraform-code]# terraform validate
Success! The configuration is valid.
```

3. Terraform fmt

It will be checking whether alignment is correct or not i.e proper alignment of the code.

4. Terraform plan

Its dry-run before creating resources validate what resources are getting created terraform plan.

+Created
-deleted
~updated

```
+ multi_region          = false
+ policy                 = (known after apply)
+ tags_all               = (known after apply)
}
```

Plan: 58 to add, 0 to change, 0 to destroy.

Changes to Outputs:

```
+ cluster_endpoint      = (known after apply)
+ cluster_name          = (known after apply)
+ cluster_security_group_id = (known after apply)
+ region                = "us-west-2"
```

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

5. Terraform apply

It will be Create resource

(or)

Without asking "yes" during the terraform apply, instead of the way we are using below command.

terraform apply - auto -approve

```
[root@ip-172-31-26-190 eks-cluster-terraform-code]# terraform apply --auto-approve
module.irs-a-ebs-csi.data.aws_caller_identity.current: Reading...
module.eks.module.kms.data.aws_partition.current: Reading...
module.eks.data.aws_partition.current: Reading...
module.eks.module.eks_managed_node_group["two"].data.aws_caller_identity.current: Reading...
ing...
```

Apply complete! Resources: 58 added, 0 changed, 0 destroyed.

Outputs:

```
cluster_endpoint = "https://DC296E200955CF7FECCAE298EB87FACB.gr7.us-west-2.amazonaws.com"
cluster_name = "education-eks-L9NgjlFI"
cluster_security_group_id = "sg-02601daec5361d6cf"
region = "us-west-2"
[root@ip-172-31-26-190 eks-cluster-terraform-code]# Connection to ec2-35-162-51-248.us-west-2.compute.amazonaws.com closed by remote host.
Connection to ec2-35-162-51-248.us-west-2.compute.amazonaws.com closed.
```

➔ After Create The Cluster we run to below Command.

`aws eks --region <region name> update-kubeconfig --name <cluster name>`

```
[root@ip-172-31-26-190 ~]# aws eks --region us-west-2 update-kubeconfig --name education-eks-L9NgjlFI
Updated context arn:aws:eks:us-west-2:201299920544:cluster/education-eks-L9NgjlFI in /root/.kube/config
[root@ip-172-31-26-190 ~]#
```

➔ First we write One sample Docker file on top Ec2 machine by using below Commands.

`vi dockerfile`

```
ARG TAG=latest
FROM ubuntu: ${TAG}
RUN apt update \
    && apt install -y apache2 zip \
    && apt install -y apache2-utils \
    && apt clean
ADD https://www.free-css.com/assets/files/free-css-templates/download/page292/grandcoffee.zip /tmp/
RUN cd /tmp \
    && unzip /tmp/grandcoffee.zip \
    && cp -r /tmp/html/* /var/www/html/
# COPY grandcoffee.zip /tmp/
EXPOSE 80
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

➔ After we save the File, we run to the below Commands.

To build to the images below Command.

1. `Docker build -t name:1.`

```

[root@ip-172-31-26-190 ~]# docker build -t name:1 .
[+] Building 25.3s (9/9) FINISHED                                docker:default
=> [internal] load build definition from dockerfile                0.0s
=> => transferring dockerfile: 506B                               0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest   1.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                     0.0s
=> [1/4] FROM docker.io/library/ubuntu:latest@sha256:77906da86b60585ce12215807 2.5s
=> => resolve docker.io/library/ubuntu:latest@sha256:77906da86b60585ce12215807 0.0s
=> => sha256:77906da86b60585ce12215807090eb327e7386c8fafb54023 1.13kB / 1.13kB 0.0s
=> => sha256:aa772c98400ef833586d1d517d3e8de670f7e712bf581ce605316 424B / 424B 0.0s
=> => sha256:ca2b0f26964cf2e80ba3e084d5983dab293fdb87485dc6445 2.30kB / 2.30kB 0.0s
=> => sha256:b8cd105490ab0525ba61b103d1b80a601b17ea9bdee9768 20.54MB / 20.54MB 0.5s

```

2. docker images -a

```

[root@ip-172-31-26-190 ~]# docker images -a
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
name          1         f20b75a13e8d   4 minutes ago  264MB
[root@ip-172-31-26-190 ~]#

```

3. Docker run -d -p host port: container port <image_id>

```

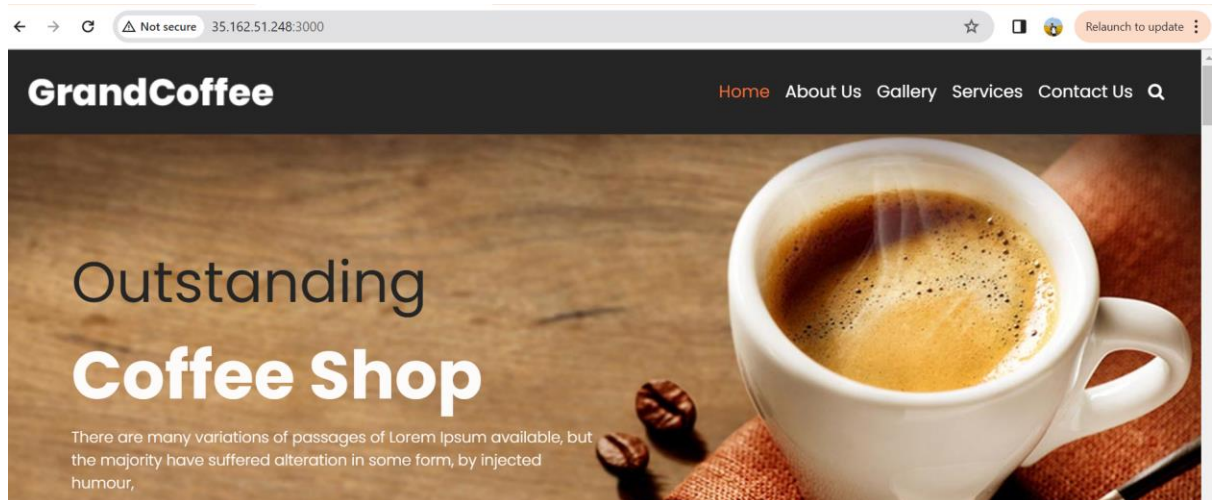
[root@ip-172-31-26-190 ~]# docker run -d -p 3000:80 f20b75a13e8d
ld872979c8d3e57b0ad1db963e74f860a34dd34f136e19941977f298e31fd6aa
[root@ip-172-31-26-190 ~]# |

```

4. To Expose to the application on container.

To copy to the publicip: hostport

<http://35.162.51.248:3000/>



➔ After we create image to send to the ECR Repository.

1. First we have Create one ECR Repository

[Amazon ECR](#) > [Private registry](#) > [Repositories](#) > [Create repository](#)

Create repository

General settings

Visibility settings | [Info](#)

Choose the visibility setting for the repository.

☒ **Private**
Access is managed by IAM and repository policy permissions.

☐ **Public**
Publicly visible and accessible for image pulls.

Repository name

Provide a concise name. A developer should be able to identify the repository contents by the name.

201299920544.dkr.ecr.us-west-2.amazonaws.com/

17 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

Tag immutability | [Info](#)

Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.

☐ **Disabled**

[i](#) Once a repository is created, the visibility setting of the repository can't be changed.

Image scan settings

➔ After Create ECR we select the-----View Push Commands ----we run to the below command

```
1. aws ecr get-login-password --region us-west-2 | docker login --  
   username AWS --password-stdin 201299920544.dkr.ecr.us-west-  
   2.amazonaws.com
```

```
[root@ip-172-31-26-190 ~]# aws ecr get-login-password --region us-west-2 | docker login  
--username AWS --password-stdin 201299920544.dkr.ecr.us-west-2.amazonaws.com  
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  
[root@ip-172-31-26-190 ~]#
```


2. `docker build -t 201299920544.dkr.ecr.us-west-2.amazonaws.com/dokcer-image-push:latest .`

```
root@ip-172-31-26-190 ~]# docker build -t 201299920544.dkr.ecr.us-west-2.amazonaws.com/dokcer-image-push:latest .
[+] Building 1.5s (9/9) FINISHED                                docker:default
=> [internal] load build definition from dockerfile              0.0s
=> => transferring dockerfile: 506B                             0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest 0.6s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [1/4] FROM docker.io/library/ubuntu:latest@sha256:77906da86b60585ce12215807 0.0s
=> https://www.free-css.com/assets/files/free-css-templates/download/page292/g 0.8s
=> CACHED [2/4] RUN apt update && apt install -y apache2 zip && apt inst 0.0s
=> CACHED [3/4] ADD https://www.free-css.com/assets/files/free-css-templates/d 0.0s
=> CACHED [4/4] RUN cd /tmp && unzip /tmp/grandcoffee.zip && cp -r /tm 0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:f20b75a13e8ded304504da6b3924583f709928c101ffd9b2c1a 0.0s
=> => naming to 201299920544.dkr.ecr.us-west-2.amazonaws.com/dokcer-image-push 0.0s
```

3. `docker push 201299920544.dkr.ecr.us-west-2.amazonaws.com/dokcer-image-push:latest`

```
root@ip-172-31-26-190 ~]# docker push 201299920544.dkr.ecr.us-west-2.amazonaws.com/dokcer-image-push:latest
The push refers to repository [201299920544.dkr.ecr.us-west-2.amazonaws.com/dokcer-image-push]
0035c922d232: Pushed
a94879b6789a: Pushed
5d9b3d489f62: Pushed
5498e8c22f69: Pushed
latest: digest: sha256:c79ef57eaa8a172ec24dae37548b8d401d9291a159c0fb8bd0d85dc26a226dbf size: 1164
```

- ➔ After we can see ECR Repo image or not .
dokcer-image-push

View push commands

Edit

Images (1)

Search artifacts

Refresh

Delete

Details

Scan

< 1 > ⚙

<input type="checkbox"/>	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
<input type="checkbox"/>	latest	Image	March 20, 2024, 22:21:32 (UTC+05:5)	119.32	Copy URI	sha256:c79ef57eaa8a172ec24dae37548b8d...

- ➔ After we run to the below commands .

1. `Kubect1 get nodes`

```
root@ip-172-31-26-190 ~]# kubect1 get nodes
NAME                                STATUS    ROLES    AGE    VERSION
ip-10-0-1-123.us-west-2.compute.internal Ready    <none>    171m   v1.24.17-eks-5e0fdde
ip-10-0-2-96.us-west-2.compute.internal Ready    <none>    171m   v1.24.17-eks-5e0fdde
ip-10-0-3-114.us-west-2.compute.internal Ready    <none>    171m   v1.24.17-eks-5e0fdde
root@ip-172-31-26-190 ~]#
```


➔ First, we can create namespace

```
Kubectl create ns docker-image
```

```
[root@ip-172-31-26-190 ~]# kubectl create ns docker-image
namespace/docker-image created
[root@ip-172-31-26-190 ~]# |
```

➔ After we can create the yaml.files

1. vi deployment.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deployment-app
  namespace: docker-image
  labels:
    app: docker-app
spec:
  selector:
    matchLabels:
      app: docker-app
  replicas: 1
  template:
    metadata:
      labels:
        app: docker-app
    spec:
      containers:
        - name: docker-app
          image: 201299920544.dkr.ecr.us-west-2.amazonaws.com/dokcer-image
push:latest
  ports:
    - containerPort: 80
      name: docker-app
```

2. vi service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: service-app
  namespace: docker-image
spec:
  selector:
    app: docker-app
  type: LoadBalancer
  ports:
    - name: docker-app
      protocol: TCP
      port: 80
      targetPort: 80
```

3. After we save the files we run the below commands

1. Kubectl apply -f .

```
[root@ip-172-31-26-190 ~]# kubectl apply -f .
deployment.apps/deployment-app created
service/service-app created
[root@ip-172-31-26-190 ~]#
```

→ Kubectl get deploy -n docker-image.

```
[root@ip-172-31-26-190 ~]# kubectl get deploy -n docker-image
NAME                READY    UP-TO-DATE    AVAILABLE    AGE
deployment-app      1/1      1              1            110s
[root@ip-172-31-26-190 ~]#
```

→ Kubectl get pod -n docker-image

```
[root@ip-172-31-26-190 ~]# kubectl get pod -n docker-image
NAME                                READY    STATUS    RESTARTS    AGE
deployment-app-5f9cd5cf8f-rqzqj    1/1      Running   0            2m52s
[root@ip-172-31-26-190 ~]#
```

→ Kubectl get svc -n docker-image.

```
root@ip-172-31-26-190 ~]# kubectl get svc -n docker-image
NAME                TYPE                CLUSTER-IP    EXTERNAL-IP
service-app         LoadBalancer        172.20.27.129  a4d2895d021af4655b5c7fbd7d613371-762011312.us-west-2.elb.amazonaws.com
onaws.com           80:31442/TCP        3m51s
root@ip-172-31-26-190 ~]#
```

→ After we can copy to the Load balancer External-ip Url and search to the Browser.

