

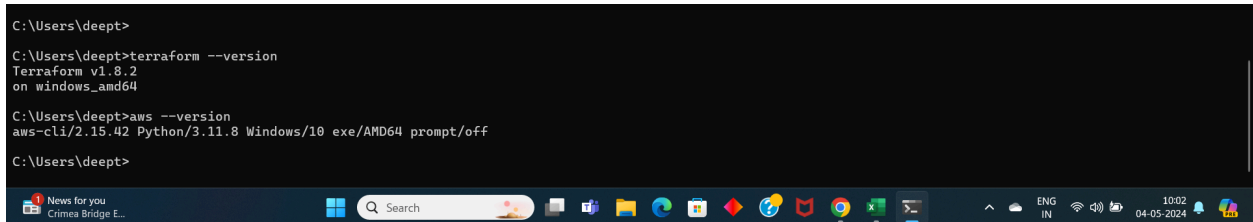
Container orchestration using kubernetes and route application using ingress controller

Deploy two different web applications in kubernetes and create a single route using K8s Ingress controller.

Infrastructure setup

Setup Infrastructure in AWS using **terraform** to create EC2 using Windows command prompt.

Downloaded terraform for windows from official website and installed awscli and configured aws access key in windows command prompt



```
C:\Users\deept>
C:\Users\deept>terraform --version
Terraform v1.8.2
on windows_amd64
C:\Users\deept>aws --version
aws-cli/2.15.42 Python/3.11.8 Windows/10 exe/AMD64 prompt/off
C:\Users\deept>
```

Created infrastructure using main.tf file

```
provider "aws" {
  region = "ap-south-1"
}

resource "aws_instance" "myInstance" {
  ami          = "ami-0f58b397bc5c1f2e8"
  instance_type = "t2.micro"

  tags = {
    Name = "MyInstance"
  }
}
```

```
Command Prompt
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.47.0

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.

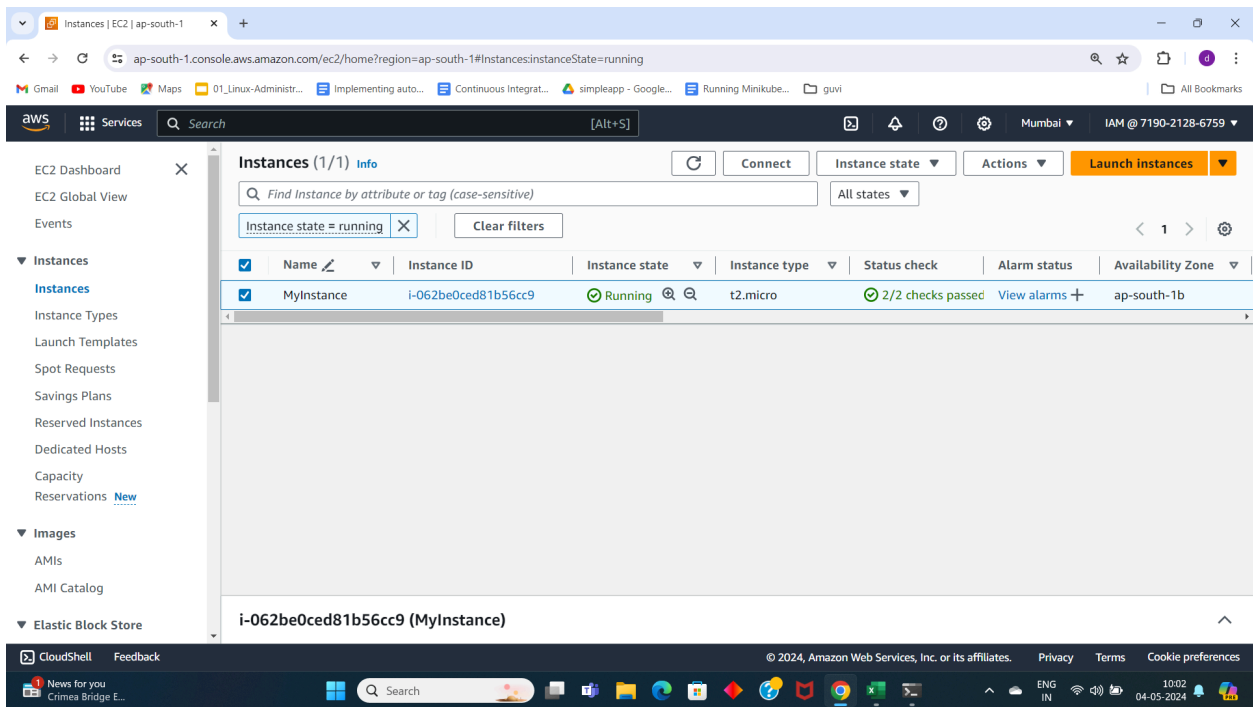
If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\Users\deept\Desktop>terraform plan
aws_instance.myInstance: Refreshing state... [id=i-062be0ced81b56cc9]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

C:\Users\deept\Desktop>terraform show
# aws_instance.myInstance:
resource "aws_instance" "myInstance" {
  ami              = "ami-0f58b397bc5c1f2e8"
  arn              = "arn:aws:ec2:ap-south-1:719021286759:instance/i-062be0ced81b56cc9"
  associate_public_ip_address = true
  availability_zone = "ap-south-1b"
  cpu_core_count   = 1
  cpu_threads_per_core = 1
  disable_api_stop  = false
  disable_api_termination = false
  ebs_optimized     = false
  get_password_data = false
  hibernation       = false
  host_id           = null
  iam_instance_profile = null
  id               = "i-062be0ced81b56cc9"
  instance_initiated_shutdown_behavior = "stop"
  instance_lifecycle = null
  instance_state     = "running"
  instance_type      = "t2.micro"
```



Kubernetes setup

Create K8s cluster using EKS command line interface (1 Master and 3 Worker nodes).

Installed kubectl using command

```
curl -o kubectl
```

```
https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl
```

```
chmod +x ./kubectl
```

```
mv ./kubectl /usr/local/bin
```

```
kubectl version --short --client
```

And setup eksctl using command

```
curl --silent --location
```

```
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
```

```
1. sudo mv /tmp/eksctl /usr/local/bin
```

```
2. eksctl version
```

Installed awscli and configured aws access key

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

```
unzip awscliv2.zip
```

```
sudo ./aws/install
```

Created cluster using commands

```
eksctl create cluster --name ingress1 --region ap-south-1 --node-type t2.micro --nodes-min 3 --nodes-max 10
```

The screenshot shows the AWS Management Console interface. The main content area displays a table of EC2 instances. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. There are four instances listed, all with a state of 'Running' and '2/2 checks passed'. The instances are named 'MyInstance', 'ingress1-ng-a...', 'ingress1-ng-a...', and 'ingress1-ng-a...'. The table also shows the Instance ID, Instance type (t2.micro), Status check (2/2 checks passed), Alarm status (View alarms), and Availability Zone (ap-south-1a). The sidebar on the left contains navigation links for EC2 Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity, Reservations, Images, AMIs, and AMI Catalog. The bottom of the screen shows a Windows taskbar with various application icons and a system tray with the date 04-05-2024.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
MyInstance	i-062be0ced81b56cc9	Running	t2.micro	2/2 checks passed	View alarms	ap-south-1b
ingress1-ng-a...	i-07fe253fe9cc380a1	Running	t2.micro	2/2 checks passed	View alarms	ap-south-1b
ingress1-ng-a...	i-0891c931bbfb5791d	Running	t2.micro	2/2 checks passed	View alarms	ap-south-1a
ingress1-ng-a...	i-098cb9e94c02a003c	Running	t2.micro	2/2 checks passed	View alarms	ap-south-1a

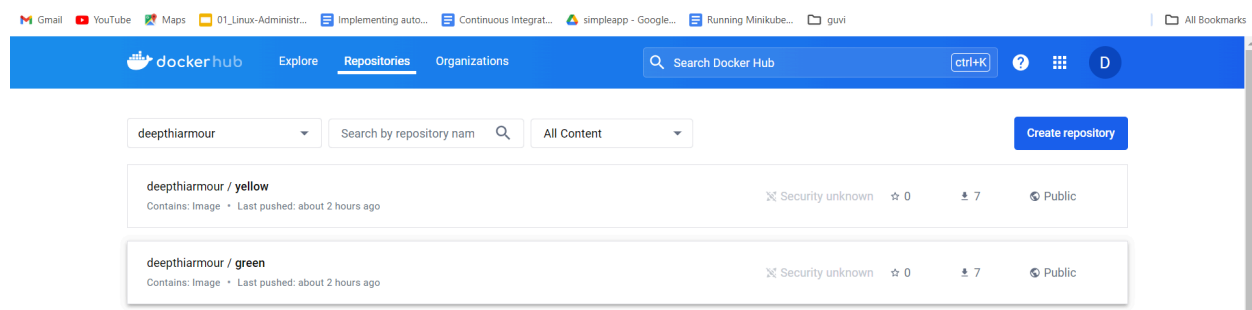
Commands used for docker

```
docker build -t yellow .
```

```
docker tag yellow deepthiarmour/yellow
```

```
docker push deepthiarmour/yellow
```

Created 2 deploymnets



Used **namespace** for deployments

```
kubectl get deployments -n color-app
```

Setup Ingress Route

Setup Ingress controller in K8s cluster and map local ip into some domain name and use that domain name to route 2 application.

Deploy AWS ALB Ingress controller

—1 `curl -O`

<https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.1.4/docs/examples/iam-policy.json>

—2 `aws iam create-policy \`

`--policy-name ALBIngressControllerIAMPolicy \`

`--policy-document file://iam-policy.json`

—3 `eksctl utils associate-iam-oidc-provider --region ap-south-1 --cluster ingress1 --approve`

—4 `eksctl create iamserviceaccount \`

`--cluster=ingress1 \`

`--namespace=kube-system\`

`--name=alb-ingress-controller \`

`--attach-policy-arn=arn:aws:iam::719021286759:policy/ALBIngressControllerIAMPolicy \`

`--override-existing-serviceaccounts \`

`--approve`

— 5 `curl -sS`

```
"https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.1.4/docs/examples/alb-ingress-controller.yaml" \
| sed "s/# --cluster-name=devCluster/- --cluster-name=ingress1/g" \
| kubectl apply -f -
```

The screenshot shows a terminal window within the AWS CloudShell interface. The terminal output displays the execution of a `curl` command to fetch a YAML manifest from GitHub, followed by a `sed` command to modify the cluster name and a `kubectl apply` command to deploy the ingress controller. The output shows the successful creation of the `deployment.apps/alb-ingress-controller`.

```
--name=alb-ingress-controller \
--attach-policy-arn=arn:aws:iam::719021286759:policy/ALBIngressControllerIAMPolicy \
--override-existing-serviceaccounts \
--approve
2024-05-04 08:24:00 [i] 1 existing iamserviceaccount(s) (color-app/alb-ingress-controller) will be excluded
2024-05-04 08:24:00 [i] 1 iamserviceaccount (kube-system/alb-ingress-controller) was included (based on the include/exclude rules)
2024-05-04 08:24:00 [!] metadata of serviceaccounts that exist in Kubernetes will be updated, as --override-existing-serviceaccounts was set
2024-05-04 08:24:00 [i] 1 task: {
  2 sequential sub-tasks: {
    create IAM role for serviceaccount "kube-system/alb-ingress-controller",
    create serviceaccount "kube-system/alb-ingress-controller",
  } )
2024-05-04 08:24:00 [i] building iamserviceaccount stack "eksctl-ingress1-addon-iamserviceaccount-kube-system-alb-ingress-controller"
2024-05-04 08:24:00 [i] deploying stack "eksctl-ingress1-addon-iamserviceaccount-kube-system-alb-ingress-controller"
2024-05-04 08:24:30 [i] waiting for CloudFormation stack "eksctl-ingress1-addon-iamserviceaccount-kube-system-alb-ingress-controller"
2024-05-04 08:24:30 [i] serviceaccount "kube-system/alb-ingress-controller" already exists
2024-05-04 08:24:30 [i] updated serviceaccount "kube-system/alb-ingress-controller"
root@ip-172-31-2-177:/home/ubuntu# curl -sS "https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.1.4/docs/examples/alb-ingress-controller.yaml" \
| sed "s/# --cluster-name=devCluster/- --cluster-name=attractive-gopher/g" \
| kubectl apply -f -
deployment.apps/alb-ingress-controller created
root@ip-172-31-2-177:/home/ubuntu#
```

i-062be0ced81b56cc9 (MyInstance)
PublicIPs: 3.111.36.34 PrivateIPs: 172.31.2.177

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

99°F Mostly sunny

Search

ENG IN 13:54 04-05-2024