

Deploy a web application in a cloud-based Kubernetes solution, ensuring proper logging and monitoring are in place.

Application Functionality

The web application shall have a simple static page of content.

Through Terraform we will deploy - VPC, EKS Cluster, Node Group, IAM Role associated with eks and nodegroup.

Created a folder and inside the folder created .tf file for

- provider.tf - We need to specify terraform the resources it need to interact and manage, here i have provided aws provider as we are deploying resources in AWS Environment.
- vpc.tf - Need to create a vpc, where we can deploy our eks cluster and worker nodes. I have created 2 public and 2 private subnets.
- eks-cluster.tf - Inside the file we need to specify terraform about the creation of resources - EKS cluster, nodegroups, IAM policy, IAM role and attachment with resources created for permissions.
- variables.tf - The variables used within the above templates, here we can provide values to them.

We can also create terraform workspace for multi-environment configuration, helping in maintaining particular state file. We can use .tfvars file to provide the value to the variables used.

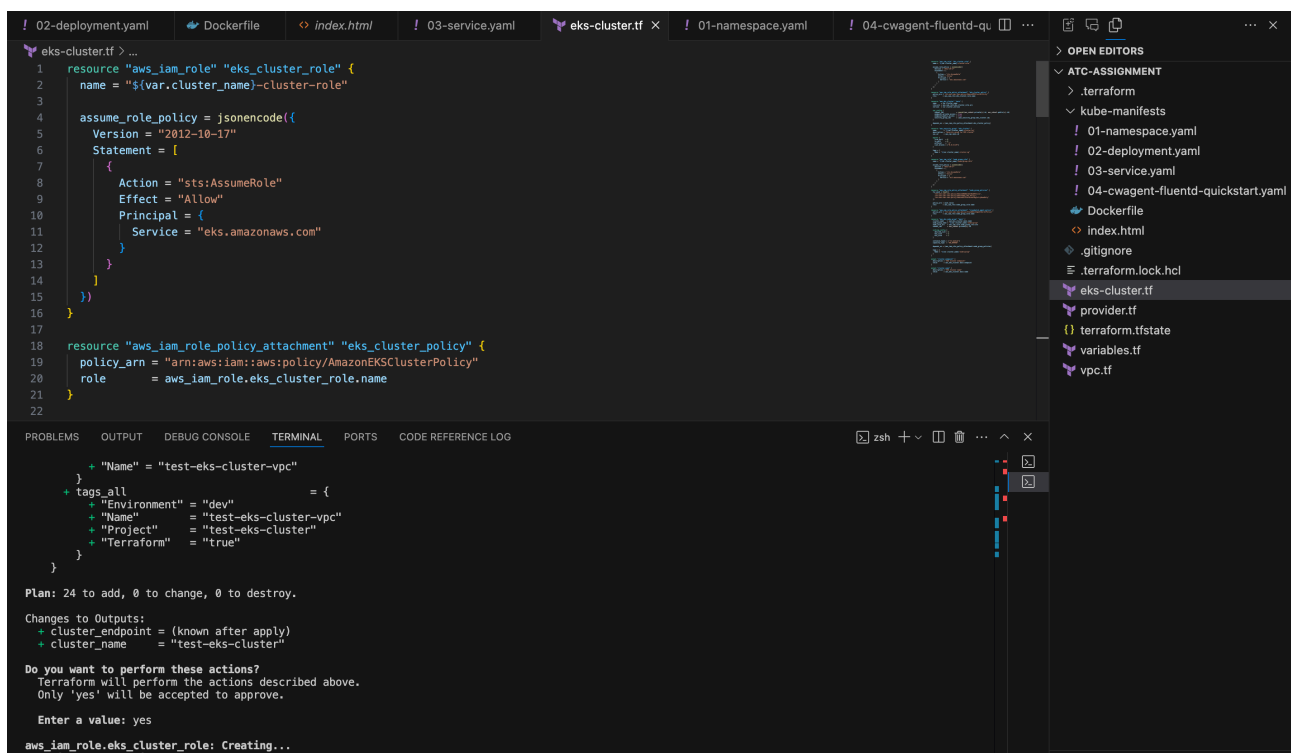
To create webserver, we are using nginx which we will specify in Dockerfile. Make sure to create the image as per the architecture of worker nodes otherwise it will create an error at the time of deployment of pods. For content to show in webpage, create an index.html file and provide valid content. Now create an image through docker build image command and push the image to docker hub. I have provided the image publicly available. Use the image inside the spec of container in deployment. It will build the pods with that container image.

The screenshot displays the Docker Hub interface for the repository 'ashish13997/web-app'. The interface is divided into a sidebar on the left and a main content area. The sidebar contains navigation links for 'Repositories', 'Settings', 'Default privacy', 'Notifications', 'Billing', 'Usage', 'Pulls', and 'Storage'. The main content area shows the repository details, including the tag 'v1' and a table of image digests for different architectures (linux/amd64). The table shows the digest 'a175d7cfba64' for the linux/amd64 architecture, pushed less than 1 day ago, with a compressed size of 16.07 MB.

Digest	OS/ARCH	Last pull	Compressed size
a175d7cfba64	linux/amd64	less than 1 day	16.07 MB

- `namespace.yaml` - To isolate and organise the resources with others. Basically it act as environment where we can create multi namespaces to differentiate prod, dev and provide limits with resource quota.
- `deployment.yaml` - This file help us with managing and deploying our application. Here i have specified replicas = 2, so we will have two pods created with specified image within spec of container's template. In labels we have to specify namespace, as per the value it will deploy the template under that namespace.
- `service.yaml` - This file help us in exposing our application to network traffic. Here i have taken Load Balancer, it is best practise to deploy worker nodes in private subnets. Therefore, to access our webserver, load balancer is used to provide internet access. By default, in service if we opt for load balancer it will take classic load balancer.
- `cloudwatch-fluentd.yaml` - To monitor our eks cluster and nodes cpu utilization, memory etc. we are using AWS CloudWatch Container-insights, through the template we are creating namespace for cloudwatch, security account, cluster role, cluster role binding, daemon set for cloudwatch agent and fluentd to collect metrics and send them to AWS container insights, config-map for both the daemon sets.

- After the creation of above templates, give command - terraform init
- To initialise the mandate plugins as specified under provider.tf file.
- To check the output of our terraform file, give command: terraform plan, it will give us an output regarding resource creation.
- After checking give command: terraform plan and yes or —auto-approve
- After the creation of VPC, EKS Cluster, nodegroups and IAM Role.
- Verify Namespace - kubectl get ns
- Verify EKS Cluster - aws eks list-clusters
- To describe the cluster - aws eks describe-cluster --name test-eks-cluster
- Verify worker nodes are up and running - kubectl get nodes



```

aws_eks_node_group.main: Still creating... [1m10s elapsed]
aws_eks_node_group.main: Still creating... [1m20s elapsed]
aws_eks_node_group.main: Still creating... [1m30s elapsed]
aws_eks_node_group.main: Still creating... [1m40s elapsed]
aws_eks_node_group.main: Creation complete after 1m49s [id=test-eks-cluster:test-eks-cluster-node-group]

Apply complete! Resources: 24 added, 0 changed, 0 destroyed.

● Outputs:

cluster_endpoint = "https://FE5E40534081E2727F92B0C8B9FF96D7.gr7.ap-south-1.eks.amazonaws.com"
cluster_name = "test-eks-cluster"
tanish@Tanishs-Laptop ATC-Assignment % aws eks update-kubeconfig --name test-eks-cluster --region ap-south-1
Updated context arn:aws:eks:ap-south-1:767398024481:cluster/test-eks-cluster in /Users/tanish/.kube/config
● tanish@Tanishs-Laptop ATC-Assignment % kubectl get nodes
NAME                                STATUS    ROLES    AGE     VERSION
ip-10-0-1-70.ap-south-1.compute.internal Ready    <none>    5m44s   v1.28.15-eks-aeac579
ip-10-0-2-67.ap-south-1.compute.internal Ready    <none>    5m46s   v1.28.15-eks-aeac579
tanish@Tanishs-Laptop ATC-Assignment %

```

```

● tanish@Tanishs-Laptop ATC-Assignment % aws eks list-clusters
{
  "clusters": [
    "test-eks-cluster"
  ]
}
● tanish@Tanishs-Laptop ATC-Assignment % kubectl get nodes
NAME                                STATUS    ROLES    AGE     VERSION
ip-10-0-1-70.ap-south-1.compute.internal Ready    <none>    7m42s   v1.28.15-eks-aeac579
ip-10-0-2-67.ap-south-1.compute.internal Ready    <none>    7m44s   v1.28.15-eks-aeac579
○ tanish@Tanishs-Laptop ATC-Assignment %

```

- Go the path where our kubernetes manifest folder is present, inside manifest folder add the namespace, deployment, service and cloud-watch yaml files.
- Give command - `cd ..`
- Now give command - `kubectl apply -f <kube-manifest-folder-name>/`
It will automatically create all the yaml files and configuration inside given.

```

tanish@Tanishs-Laptop ATC-Assignment % kubectl apply -f kube-manifests
namespace/test created
deployment.apps/web-app created
service/web-app-service created
namespace/amazon-cloudwatch created
serviceaccount/cloudwatch-agent created
clusterrole.rbac.authorization.k8s.io/cloudwatch-agent-role created
clusterrolebinding.rbac.authorization.k8s.io/cloudwatch-agent-role-binding created
configmap/cwagentconfig created
daemonset.apps/cloudwatch-agent created
configmap/cluster-info created
serviceaccount/fluentd created
clusterrole.rbac.authorization.k8s.io/fluentd-role created
clusterrolebinding.rbac.authorization.k8s.io/fluentd-role-binding created
configmap/fluentd-config created
daemonset.apps/fluentd-cloudwatch created

```

```

tanish@Tanishs-Laptop ATC-Assignment % kubectl get pods -n test
NAME                                READY   STATUS    RESTARTS   AGE
web-app-75b6c67489-2x7h5           1/1     Running   0           72s
web-app-75b6c67489-4hrhv           1/1     Running   0           72s
tanish@Tanishs-Laptop ATC-Assignment % kubectl get pods -n amazon-cloudwatch
NAME                                READY   STATUS    RESTARTS   AGE
cloudwatch-agent-ctd8w             1/1     Running   0           87s
cloudwatch-agent-gkv5l             1/1     Running   0           87s
fluentd-cloudwatch-7bx8j           1/1     Running   0           87s
fluentd-cloudwatch-mhsr6           1/1     Running   0           87s
tanish@Tanishs-Laptop ATC-Assignment % kubectl get svc -n test
NAME                                TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
web-app-service                    LoadBalancer 172.20.206.139  ad87341296a82413eb7b5b035cc70047-1731120878.ap-south-1.elb.amazonaws.com 80:31531/TCP    2m

```

- To check pods are running for app server and cloudwatch -
- `kubectl get pods -n amazon-cloudwatch`
- `kubectl get pods -n test`
- To access our webpage - `kubectl get svc -n test`
- Copy the DNS name or from AWS console > EC2 > LoadBalancer > Click on LoadBalancer created and copy the DNS Name.

The screenshot shows the AWS Management Console interface. The breadcrumb navigation at the top indicates the path: **EC2** > **Load balancers** > **ad87341296a82413eb7b5b035cc70047**. The left-hand navigation pane is expanded to show **Load Balancing**. The main content area displays the details for the selected Load Balancer:

- Load balancer type:** Classic
- Status:** 2 of 2 instances in service
- Scheme:** Internet-facing
- Hosted zone:** ZP97RAFLXTNZK
- VPC:** [vpc-0410a25573169c3f2](#)
- Availability Zones:**
 - [subnet-08417ccb75da98afe](#) ap-south-1b (aps1-az3)
 - [subnet-05a25abf0eb59eb6c](#) ap-south-1a (aps1-az1)
- DNS name:** [ad87341296a82413eb7b5b035cc70047-1731120878.ap-south-1.elb.amazonaws.com](#) (A Record)
- Distribution of targets by Availability Zone (AZ):** For each enabled Availability Zone, you can view the number of registered instances and their current health states. Selecting any values here will apply the table.

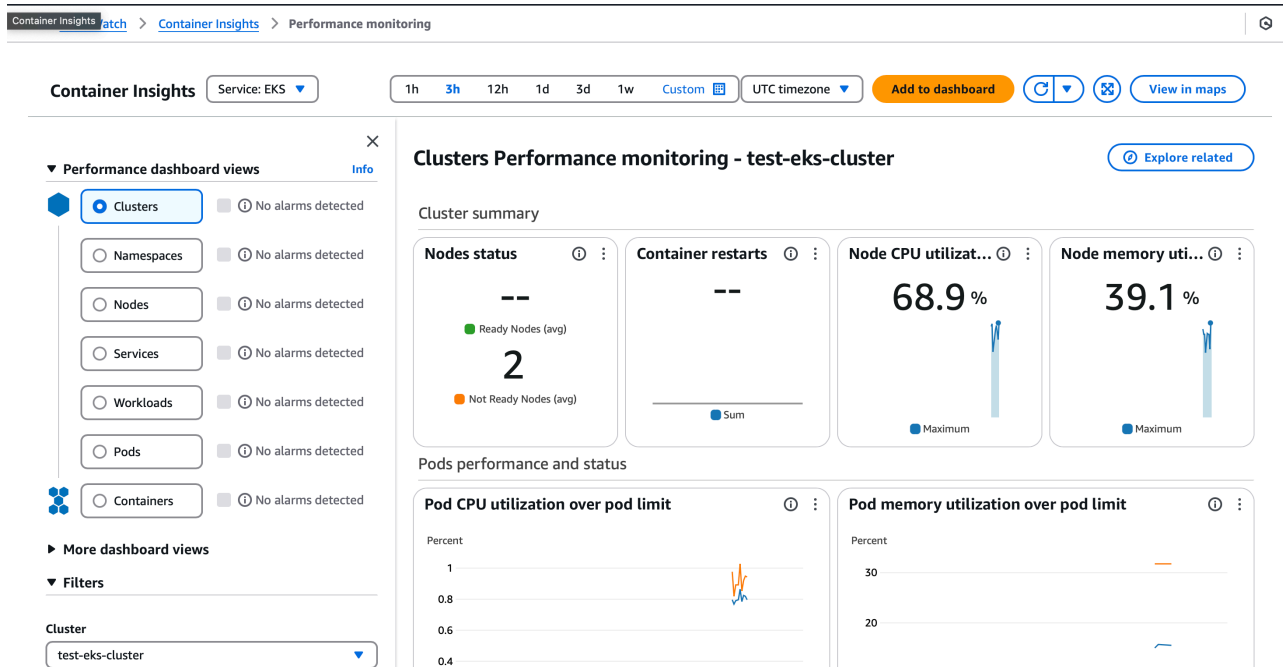
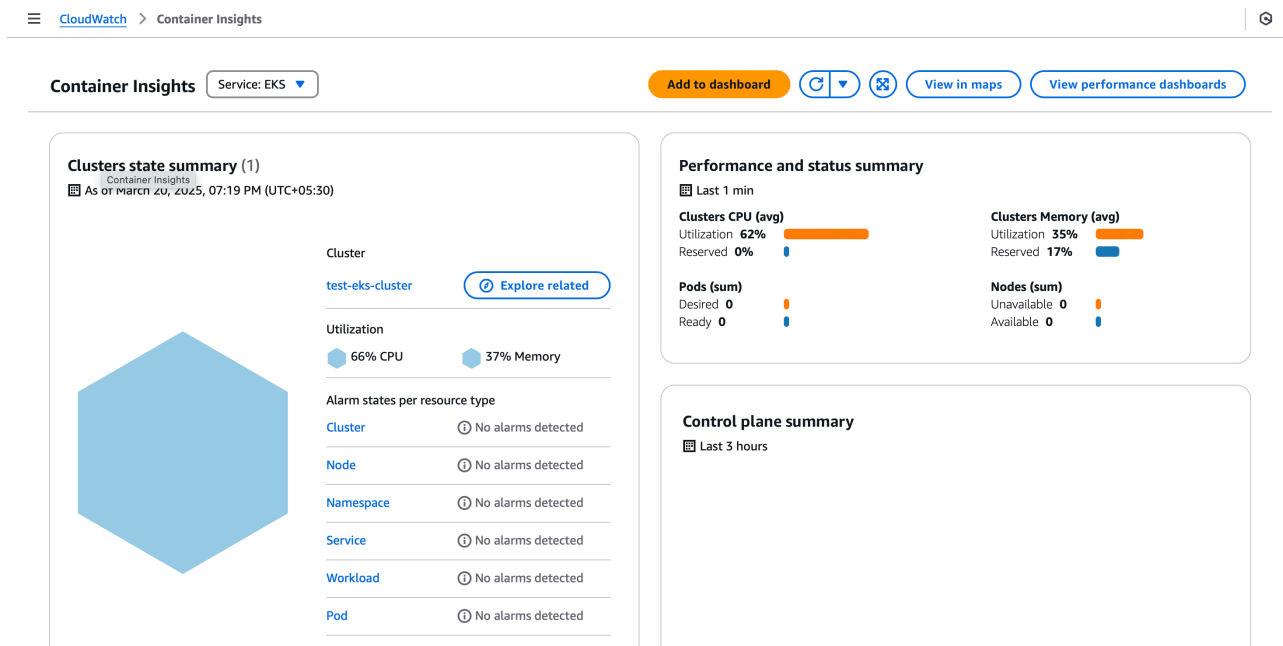
At the bottom, there is a horizontal tab bar with the following tabs: **Listeners** (selected), **Network mapping**, **Security**, **Health checks**, **Target instances**, **Monitoring**, and **Attribu**.

- Paste the URL over the browser, here we can access the webpage.

The screenshot shows a web browser window. The address bar displays the URL: `ad87341296a82413eb7b5b035cc70047-1731120878.ap-south-1.elb.amazonaws.com`. The browser status bar at the bottom indicates "Not Secure".

Hello, welcome to ATC assignment

- To check the monitoring and utilisation of our cluster and nodes, go to AWS Console > CloudWatch > Container Insights
- Here we can see metrics related to our EKS Cluster.



Based upon our load we can also use Vertical Pod Autoscaler (By default present within EKS) and Horizontal Pod Autoscaler.