

# Creation of an infrastructure and deployment of a web application on the Elastic Kubernetes Service by AWS

By

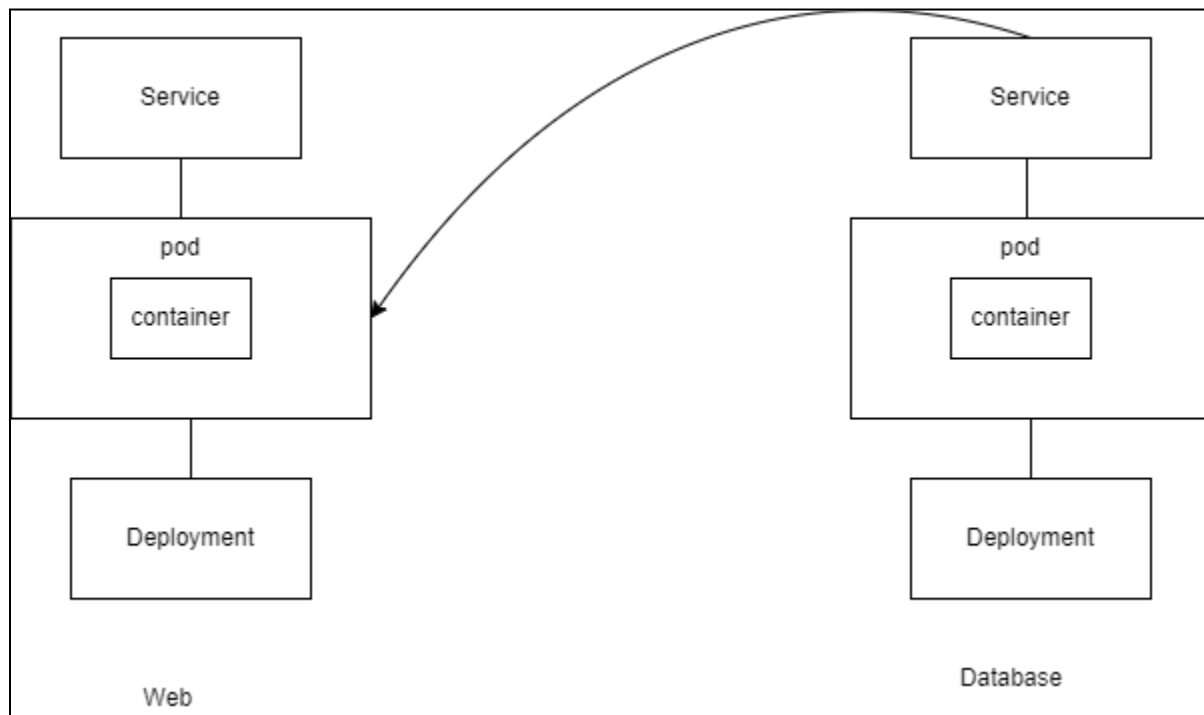
Akshat Parikh

## Introduction

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed Kubernetes service that makes it easy to run Kubernetes on AWS and on-premises. Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications.

## Application Infrastructure

- We have used a sample web application consisting of two services. One is a Web app, which will be our UI, and the second one is a database service.
- The database service is connected to a web service for data communication purposes.
- We have set up a public-facing load balancer to serve external requests.



- The database part uses PostgreSQL and the web part uses Angular and .NET Core Web API.
- For each part, we have Kubernetes services and deployment.

Next, I created an EKS cluster for the application.

The screenshot shows the AWS EKS console for a cluster named 'DemoByAkshat'. At the top, there's a navigation bar with 'EKS > Clusters > DemoByAkshat'. The cluster name is displayed prominently. A notification bar at the top indicates 'New versions are available for 2 add-ons'. Below this, the 'Cluster info' section shows the Kubernetes version as 1.22, the status as 'Active' with a green checkmark, and the provider as EKS. A horizontal menu below the cluster info includes 'Overview' (selected), 'Resources', 'Compute', 'Networking', 'Add-ons', 'Authentication', 'Logging', 'Update history', and 'Tags'. The 'Details' section is expanded, showing the API server endpoint, OpenID Connect provider URL, Created time (an hour ago), Certificate authority, Cluster IAM role ARN, Cluster ARN, and Platform version (eks.2).

Afterwards, I created a node group having on-demand nodes with a minimum size of 1 node and a maximum size of 3 nodes. As shown below, the nodes are established and healthy.

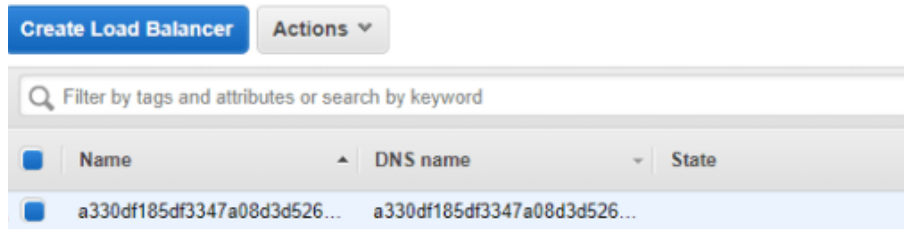
The screenshot shows the AWS EKS console for a node group named 'NodeDemo' within the 'DemoByAkshat' cluster. The navigation bar shows 'EKS > Clusters > DemoByAkshat > Node group: NodeDemo'. The node group name is displayed. Buttons for 'Refresh', 'Edit', and 'Delete' are visible. The 'Node group configuration' section shows the Kubernetes version as 1.22, the AMI type as 'AL2\_x86\_64', and the status as 'Active' with a green checkmark. Below this, the AMI release version (1.22.6-20220526), instance types ('t3.micro'), and disk size ('5 GiB') are listed. A horizontal menu below the configuration includes 'Details' (selected), 'Nodes', 'Health issues' (with a green circle icon), 'Kubernetes labels', 'Update config', 'Kubernetes taints', 'Update history', and 'Tags'. The 'Details' section is expanded, showing the Node group ARN, Autoscaling group name, Capacity type (On-Demand), Minimum size (1 node), Maximum size (3 nodes), Desired size (2 nodes), Subnets, and the configuration for SSH access to nodes (Disabled).

The deployment to EKS is handled by YAML files, which are the configuration files for K8s.

I have applied those files to the cluster created in the previous step.

We can see the information about deployments and other resources by simply running the command 'kubectl get all'.

As we are leveraging the AWS services, we can see the load balancer being created based on our service manifest file.



There are other AWS resources created, such as VPC, Security Groups, etc., to make all this happen.

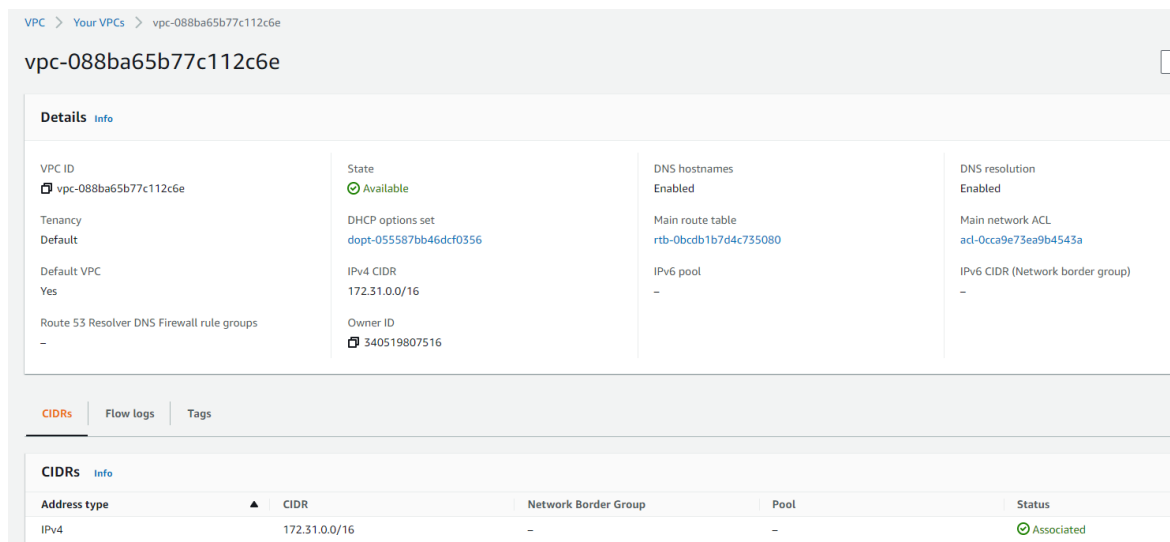


Image of VPC configuration

EC2 > Security Groups > sg-020deeff12c88d3 - default

### sg-020deeff12c88d3 - default

Actions ▾

**Details**

Security group name default	Security group ID sg-020deeff12c88d3	Description default VPC security group	VPC ID vpc-088ba65b77c112c6e
Owner 340519807516	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

**Inbound rules** | Outbound rules | Tags

You can now check network connectivity with Reachability Analyzer [Run Reachability Analyzer](#)

**Inbound rules (1/1)** [Manage tags](#) [Edit inbound rules](#)

Filter security group rules

<input checked="" type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
<input checked="" type="checkbox"/>	-	sg-03f8ad05a07ac4a6	-	All traffic	All	All	sg-020deeff12c88d...	-

## Security group configuration

Let's go to the public address of the load balancer, and our web application dashboard will show up.

