

[Open in app ↗](#)[Sign up](#)[Sign in](#)**Medium**

Search

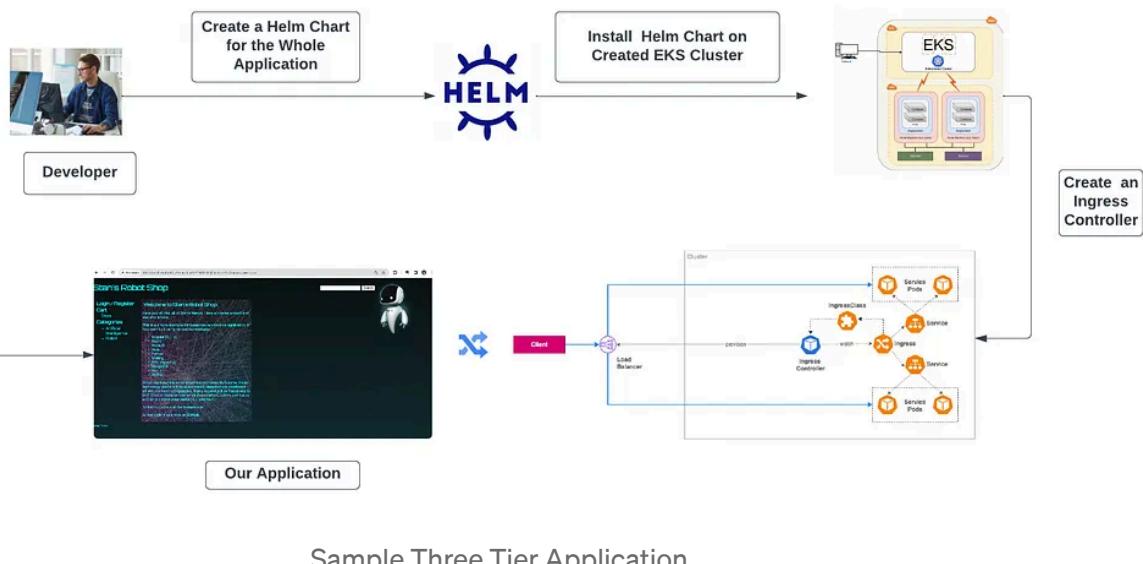


Deploy an E -Commerce Three Tier application on AWS EKS with Helm

6 min read · Feb 5, 2024



Sreedhar Reddy Madithati

[Follow](#)[Listen](#)[Share](#)

Introduction:

In the dynamic landscape of software development, architects and developers constantly seek robust design patterns that ensure scalability, maintainability, and efficient resource utilization. One such time-tested approach is the 3-tier architecture, a well-structured model that divides an application into three interconnected layers. This architectural style has been a cornerstone for building scalable and resilient applications for decades.

Understanding the Basics:

The 3-tier architecture is composed of three primary layers, each with distinct responsibilities:

1. Presentation Layer:

- Also known as the user interface layer, this tier is responsible for interacting with end-users.
- It encompasses the user interface components, such as web pages, mobile apps, or any other interface through which users interact with the application.
- The goal is to provide a seamless and intuitive user experience while keeping the presentation logic separate from the business logic.

2. Application (or Business Logic) Layer:

- Positioned between the presentation and data layers, the application layer contains the business logic that processes and manages user requests.
- It acts as the brain of the application, handling tasks such as data validation, business rules implementation, and decision-making.
- Separating the business logic from the presentation layer promotes code reusability, maintainability, and adaptability to changes.

3. Data Layer:

- The data layer is responsible for managing and storing the application's data.
- It includes databases, data warehouses, or any other data storage solutions.
- This layer ensures data integrity, security, and efficient data retrieval for the application.
- By isolating data-related operations, developers can optimize data access and storage mechanisms independently of the rest of the application.

Benefits of 3-Tier Architecture:

1. Scalability:

- The modular nature of 3-tier architecture allows for independent scaling of each layer.
- This enables efficient resource allocation, ensuring that specific components can be scaled based on demand without affecting the entire application.

2. Maintainability:

- With clear separation of concerns, developers can make changes to one layer without impacting others.
- This facilitates easier debugging, updates, and maintenance, as modifications can be confined to the relevant layer.

3. Flexibility and Adaptability:

- The architecture accommodates technology changes and updates without disrupting the entire system.
- New technologies can be integrated into specific layers, allowing the application to evolve over time.

Prerequisites:

1. kubectl — A command line tool for working with Kubernetes clusters. For more information, see Installing or updating kubectl.
<https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html>
2. eksctl — A command line tool for working with EKS clusters that automates many individual tasks. For more information, see Installing or updating.
<https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html>
3. AWS CLI — A command line tool for working with AWS services, including Amazon EKS. For more information, see Installing, updating, and uninstalling the AWS CLI <https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html> in the AWS Command Line Interface User Guide.
4. After installing the AWS CLI, I recommend that you also configure it. For more information, see Quick configuration with aws configure in the AWS Command Line Interface User Guide. <https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config>

Steps:

Step-1: Create an EKS Cluster

```
eksctl create cluster \
--name your-cluster-name \
```

```
--region your-region \
--nodegroup-name your-nodegroup-name \
--node-type t3.small \
--nodes 2 \
--nodes-min 1 \
--nodes-max 3
```

This will create an EKS Cluster along with the node-group

The screenshot shows a terminal window and a browser window. The terminal window displays the command used to create the EKS cluster and node group. The browser window is on the AWS CloudWatch console, specifically the EKS service, showing the 'Robotshop-Cluster' configuration. It includes sections for 'Cluster info', 'Nodes (0)', and 'Node groups (1)'. The 'Node groups (1)' section shows a single node group named 'Nodegroup-1' with a size of 2, using AMI release version 1.27.9-20240202, and a launch template named 'eksctl-Robotshop-Cluster-nodegroup-Nodegroup-1 (1)'. The status is listed as 'Active'.

Step-2: Configure IAM OIDC provider

1. Export Cluster Name and assign oidc_id

```
export cluster_name=<CLUSTER-NAME>
oidc_id=$(aws eks describe-cluster --name $cluster_name --query "cluster.identity.oidc.providerARN" --output text)
```

2. Check if there is an IAM OIDC provider configured already

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

3. If not, run the below command.

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name --approve
```

Step-3: Setup ALB Add-On

1. Download IAM policy.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-cor
```

2. Create IAM Policy.

```
aws iam create-policy \
--policy-name AWSLoadBalancerControllerIAMPolicy \
--policy-document file://iam_policy.json
```

```
adduser@SREE:~$ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.5.4/docs/install/iam_policy.json
% Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload   Total Spent  Left  Speed
100  8386  100  8386    0     0 24454      0 --:--:-- --:--:-- --:--:-- 24520
adduser@SREE:~$ aws iam create-policy \
--policy-name AWSLoadBalancerControllerIAMPolicy \
--policy-document file://iam_policy.json
An error occurred (EntityAlreadyExists) when calling the CreatePolicy operation: A policy called AWSLoadBalancerControllerIAMPolicy already exists. Duplicate names are not allowed.
```

3. Create IAM Role.

```
eksctl create iamserviceaccount \
--cluster=<your-cluster-name> \
--namespace=kube-system \
--name=aws-load-balancer-controller \
--role-name AmazonEKSLoadBalancerControllerRole \
--attach-policy-arn=arn:aws:iam::<your-aws-account-id>:policy/AWSLoadBalancer \
--approve
```

```
adduser@SREB:~$ eksctl create iamserviceaccount \
--cluster=Robotshop-Cluster \
--namespace=kube-system \
--name=aws-load-balancer-controller \
--role-name AmazonEKSLoadBalancerControllerRole \
--attach-policy-arn=arn:aws:iam::862547479026:policy/AWSLoadBalancerControllerIAMPolicy \
--approve
2024-02-05 10:57:32 [+] 1 iamserviceaccount (kube-system/aws-load-balancer-controller) was included (based on the include/exclude rules)
2024-02-05 10:57:32 [+] serviceaccounts that exist in Kubernetes will be excluded, use --override-existing-serviceaccounts to override
2024-02-05 10:57:32 [+] 1 task: {
  2 sequential sub-tasks: {
    create IAM role for serviceaccount "kube-system/aws-load-balancer-controller",
    create serviceaccount "kube-system/aws-load-balancer-controller",
  } }2024-02-05 10:57:32 [+] building iamserviceaccount stack "eksctl-Robotshop-Cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2024-02-05 10:57:32 [+] deploying stack "eksctl-Robotshop-Cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2024-02-05 10:57:32 [+] waiting for CloudFormation stack "eksctl-Robotshop-Cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2024-02-05 10:58:03 [+] waiting for CloudFormation stack "eksctl-Robotshop-Cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2024-02-05 10:58:04 [+] created serviceaccount "kube-system/aws-load-balancer-controller"
```

Step-4: Deploy ALB controller

1. Add helm repo.

```
helm repo add eks https://aws.github.io/eks-charts
```

2. Update the repo.

```
helm repo update eks
```

```
adduser@SREE:~$ helm repo add eks https://aws.github.io/eks-charts
"eks" already exists with the same configuration, skipping
adduser@SREE:~$ helm repo update eks
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "eks" chart repository
Update Complete. *Happy Helming!*
```

3. Install the chart.

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n k
```

4. Verify that the deployments are running.

```
kubectl get deployment -n kube-system aws-load-balancer-controller
```

```
adduser@SREE:~$ helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=Robotshop-Cluster --set serviceAccount.create=false --set serviceAccount.name=aws-load-balancer-controller --set region=us-east-1 --set vpcId=vpc-0bee82e8dbe06b03c
NAME: aws-load-balancer-controller
LAST DEPLOYED: Mon Feb 5 11:04:41 2024
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
adduser@SREE:~$ kubectl get deployment -n kube-system aws-load-balancer-controller
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           46s

```

Step-5: EBS CSI Plugin configuration.

1. The Amazon EBS CSI plugin requires IAM permissions to make calls to AWS APIs on your behalf.
2. Create an IAM role and attach a policy. AWS maintains an AWS managed policy or you can create your own custom policy. You can create an IAM role and attach the AWS managed policy with the following command. Replace my-cluster with the name of your cluster. The command deploys an AWS CloudFormation stack that creates an IAM role and attaches the IAM policy to it.

```
eksctl create iamserviceaccount \
--name ebs-csi-controller-sa \
--namespace kube-system \
--cluster <YOUR-CLUSTER-NAME> \
--role-name AmazonEKS_EBS_CSI_DriverRole \
--role-only \
--attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
--approve
```

```
adduser@SREE:~$ eksctl create iamserviceaccount \
--name ebs-csi-controller-sa \
--namespace kube-system \
--cluster Robotshop-Cluster \
--role-name AmazonEKS_EBS_CSI_DriverRole \
--role-only \
--attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
--approve
2024-02-05 11:08:57 [■] 1 existing iamserviceaccount(s) (kube-system/aws-load-balancer-controller) will be excluded
2024-02-05 11:08:57 [■] 1 iamserviceaccount (kube-system/ebs-csi-controller-sa) was included (based on the include/exclude rules)
2024-02-05 11:08:57 [!] serviceaccounts in Kubernetes will not be created or modified, since the option --role-only is used
2024-02-05 11:08:57 [■] 1 task: { create IAM role for serviceaccount "kube-system/ebs-csi-controller-sa" }
2024-02-05 11:08:57 [■] building iamserviceaccount stack "eksctl-Robotshop-Cluster-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2024-02-05 11:08:57 [■] deploying stack "eksctl-Robotshop-Cluster-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2024-02-05 11:08:58 [■] waiting for CloudFormation stack "eksctl-Robotshop-Cluster-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2024-02-05 11:09:29 [■] waiting for CloudFormation stack "eksctl-Robotshop-Cluster-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
```

3. Run the following command. Replace with the name of your cluster, with your account ID.

```
eksctl create addon --name aws-ebs-csi-driver --cluster <YOUR-CLUSTER-NAME> --s
```

```
adduser@SREE:~$ eksctl create addon --name aws-ebs-csi-driver --cluster Robotshop-Cluster --service-account-role-arn arn:aws:iam:862547479026:role/AmazonEK
S_EBS_CSI_DriverRole --force
2024-02-05 11:10:56 [+] Kubernetes version "1.27" in use by cluster "Robotshop-Cluster"
2024-02-05 11:10:57 [+] using provided ServiceAccountRoleARN "arn:aws:iam:862547479026:role/AmazonEKS_EBS_CSI_DriverRole"
2024-02-05 11:10:57 [+] creating addon
[...]
```

Step-6: Install the Helm Chart:

- Initially Clone the GitHub Repo:

Github URL :<https://github.com/uniquesreedhar/RobotShop-Project.git>

```
git clone https://github.com/uniquesreedhar/RobotShop-Project.git
```

- Then Navigate to the path where chart.yaml exists

```
cd RobotShop-Project/EKS/helm
```

- Create a namespace and then install the helm chart.

```
kubectl create ns robot-shop
helm install robot-shop --namespace robot-shop .
```

```
adduser@SREE:~/projects/RobotShop-Project/EKS/helm$ kubectl create ns robot-shop
namespace/robot-shop created
adduser@SREE:~/projects/RobotShop-Project/EKS/helm$ $ helm install robot-shop --namespace robot-shop .
$: command not found
adduser@SREE:~/projects/RobotShop-Project/EKS/helm$ helm install robot-shop --namespace robot-shop .
NAME: robot-shop
LAST DEPLOYED: Mon Feb 5 11:16:31 2024
NAMESPACE: robot-shop
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

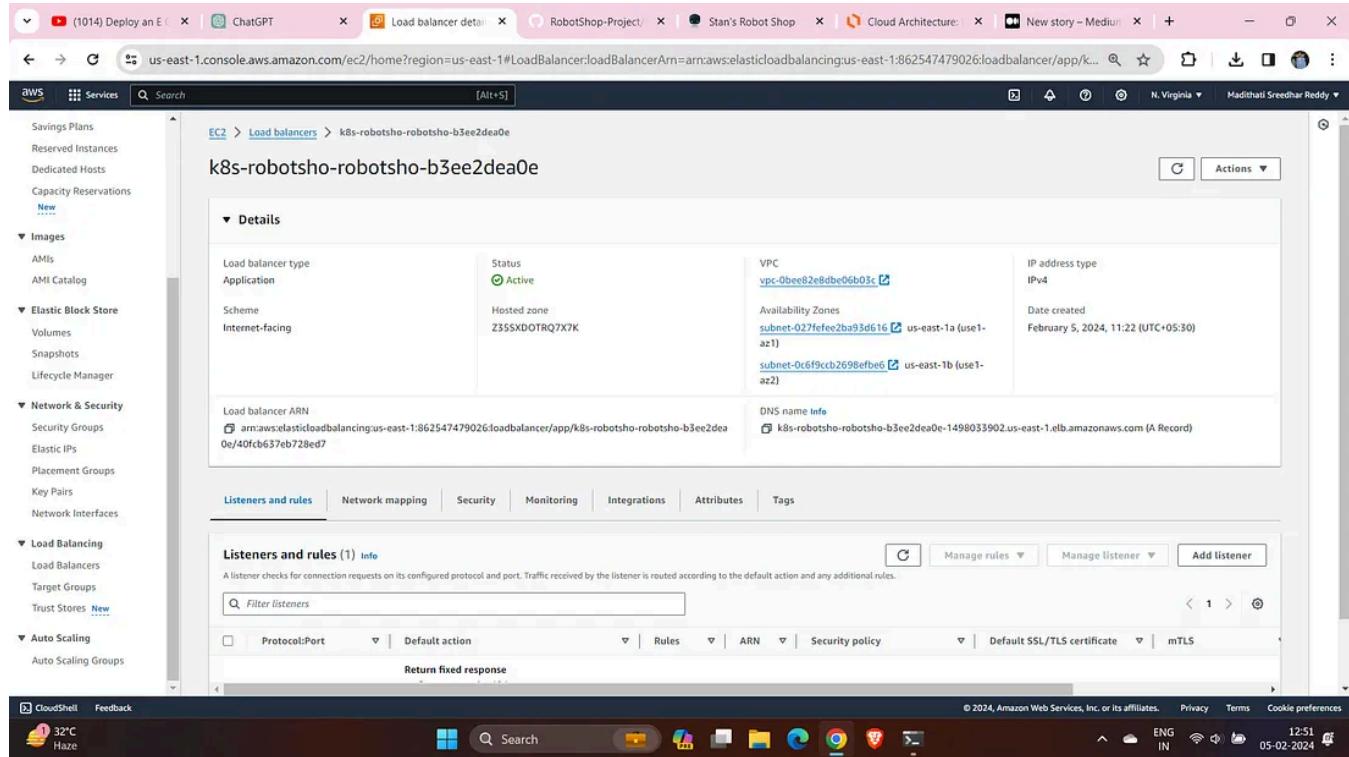
- Ensure all the pods are running if not troubleshoot the issues.

```
adduser@SREE:~/projects/RobotShop-Project/EKS/helm$ kubectl get pods -n robot-shop
NAME                               READY   STATUS    RESTARTS   AGE
cart-dd947f945-xsnks              1/1     Running   0          101m
catalogue-7fd84d6c48-trvdm        1/1     Running   0          101m
dispatch-66cb67494d-6j5dp         1/1     Running   0          101m
mongodb-db95c57c5-q99qf           1/1     Running   0          101m
mysql-8cbc4749d-ptfbv            1/1     Running   0          101m
payment-85f9dcf964-7jdq8          1/1     Running   0          101m
rabbitmq-79488b58b5-hmbfr         1/1     Running   0          101m
ratings-5c79749c7b-rmlvz          1/1     Running   0          101m
redis-0                            1/1     Running   0          101m
shipping-75f7cf6b6d-v4phr         1/1     Running   0          101m
user-59745f7ccb-cvtrt             1/1     Running   0          101m
web-656685795d-wn6jx              1/1     Running   0          101m
adduser@SREE:~/projects/RobotShop-Project/EKS/helm$ |
```

Step 7: Create Ingress

```
cd /RobotShop-Project/EKS/helm
kubectl apply -f ingress.yaml
```

This will create a Load Balancer on the AWS console.



Paste the DNS-name on your favorite browser and access the application.

Welcome to Stan's Robot Shop

Here you will find all of Stan's friends. Have a browse around and see who is here.

This is a simple example microservices ecommerce application. It has been built using various technologies:

- AngularJS (1.x)
- Nginx
- NodeJS
- Java
- Python
- Golang
- PHP (Apache)
- MongoDB
- Redis
- MySQL

When deployed into an environment monitored by Instana, these technology stacks will be automatically detected and monitored, all with minimum configuration. Every request will be traced end to end. Stan will keep an eye on all those metrics, events and traces and let you know what needs your attention.

To find out more visit the [Instana site](#).

All the code is available on [Github](#).

anonymous-4

Get Register and login into the application.

Stan's Robot Shop

Login / Register

Cart Greetings sree
€2048.00

Categories Email - sree@gmail.com

- Artificial Intelligence
 - Evoid
 - Stan
 - Watson
- Robot
 - Cyberneted Neutralization Android
 - Exceptional Medical Machine
 - Extreme Probe Emulator
 - High-Powered Travel Droid
 - Responsive Enforcer Droid
 - Robotic Mining Cyborg
 - Stan
 - Strategic Human Control

Order History
Order ID Items Total

32°C Haze

Rate the Artificial Intelligence.

The screenshot shows a product page for a robot named "Ewooid". The main image is a blue and white humanoid robot with three circular eyes on its head. Below the image, it says "Rating 4.0 from 1 votes" with five small icons below it. It is described as "Fully sentient assistant". The price is listed as "Price €200.00 Out of stock". The page includes a sidebar with "Login / Register" and "Cart" (Empty). A categories menu lists "Artificial Intelligence" (Ewooid, Stan, Watson) and "Robot". A search bar at the top right contains the word "Search". The bottom of the screen shows a Windows taskbar with various icons and a weather widget showing 32°C Haze.

Add to cart the robots.

The screenshot shows a product page for a robot named "Exceptional Medical Machine". The main image is a blue and white robot with a cylindrical body and two arms ending in medical tools. Below the image, it says "Rating No votes yet. Vote now." with five small icons below it. It is described as "Fully automatic surgery droid with exceptional bedside manner". The price is listed as "Price €1024.00 Quantity 2" with an "Add to cart" button. The page includes a sidebar with "Login / Register" and "Cart" (Empty). A categories menu lists "Artificial Intelligence" (Ewooid, Stan, Watson) and "Robot" (Cybernetic Neutralization Android, Exceptional Medical Machine, Extreme Probe Emulator, High-Powered Travel Droid, Responsive Enforcer Droid, Robotic Mining Cyborg, Stan, Strategic Human Control). A search bar at the top right contains the word "Search". The bottom of the screen shows a Windows taskbar with various icons and a weather widget showing 32°C Haze.

Checkout them.

The screenshot shows the Stan's Robot Shop cart page. The cart contains one item: "Exceptional Medical Machine" with a quantity of 2, sub total of €2048.00, inc tax of €341.33, and a total of €2048.00. There is a "Checkout" button. The left sidebar lists categories like Artificial Intelligence and Robot. A search bar at the top right has the word "Search". A small robot icon is in the top right corner.

Select your Location and then calculate the price and confirm the order.

The screenshot shows the Stan's Robot Shop shipping information page. It displays a dropdown for "Select country" set to "India" and an input field for "Enter location" containing "Hyderabad". Below these fields are buttons for "Calculate" and "Confirm". To the right, it shows a distance of "8925km" and a cost of "€446.25". The left sidebar and robot icon are visible.

Then place the Order.

Review your order

| QTY | Name | Sub Total |
|-----|-----------------------------|-----------|
| 2 | Exceptional Medical Machine | €2048.00 |
| 1 | Stan | €446.25 |
| | Inc Tax | €415.71 |
| | Total | €2494.25 |

[Pay Now](#)

Categories

- Artificial Intelligence
 - Ewoold
 - Stan
 - Watson
- Robot
 - Cybernated Neutralization Android
 - Exceptional Medical Machine
 - Extreme Probe Emulator
 - High-Powered Travel Droid
 - Responsive Enforcer Droid
 - Robotic Mining Cyborg
 - Stan
 - Strategic Human Control

And pay the amount .

Review your order

Order placed **f81cf1a8-4124-4c34-87d0-3251788f9f3d**

Thank you for your order [Continue shopping](#)

Categories

- Artificial Intelligence
 - Ewoold
 - Stan
 - Watson
- Robot
 - Cybernated Neutralization Android
 - Exceptional Medical Machine
 - Extreme Probe Emulator
 - High-Powered Travel Droid
 - Responsive Enforcer Droid
 - Robotic Mining Cyborg
 - Stan
 - Strategic Human Control

Congratulations... Your Order has been Successfully placed.. :)

DevOps

Devops Practice

Devops Project

AWS

Aws Eks

[Follow](#)

Written by Sreedhar Reddy Madithati

339 followers · 3 following

SRE at Autorabit , Hyderabad

Responses (1)



Write a response

What are your thoughts?



Mkx

Dec 30, 2024

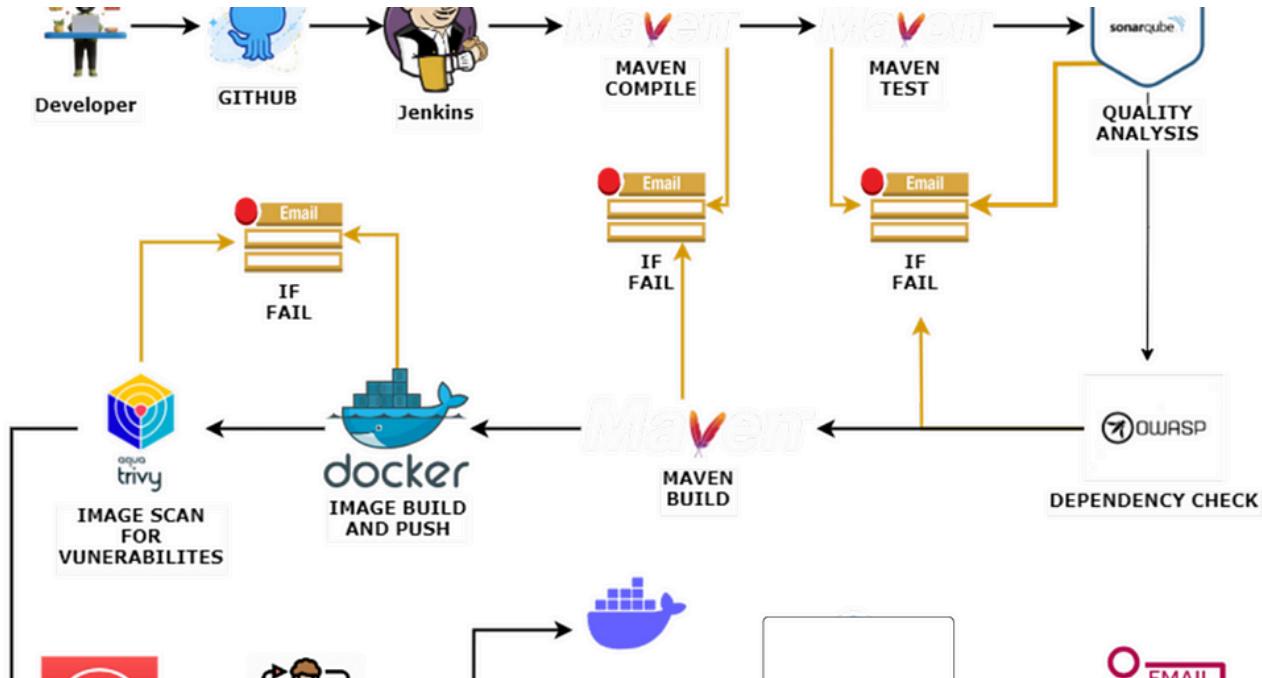


bro can you add Jenkins jenkins pipeline syntax or Jenkins file, it will help me a lot.



[Reply](#)

More from Sreedhar Reddy Madithati



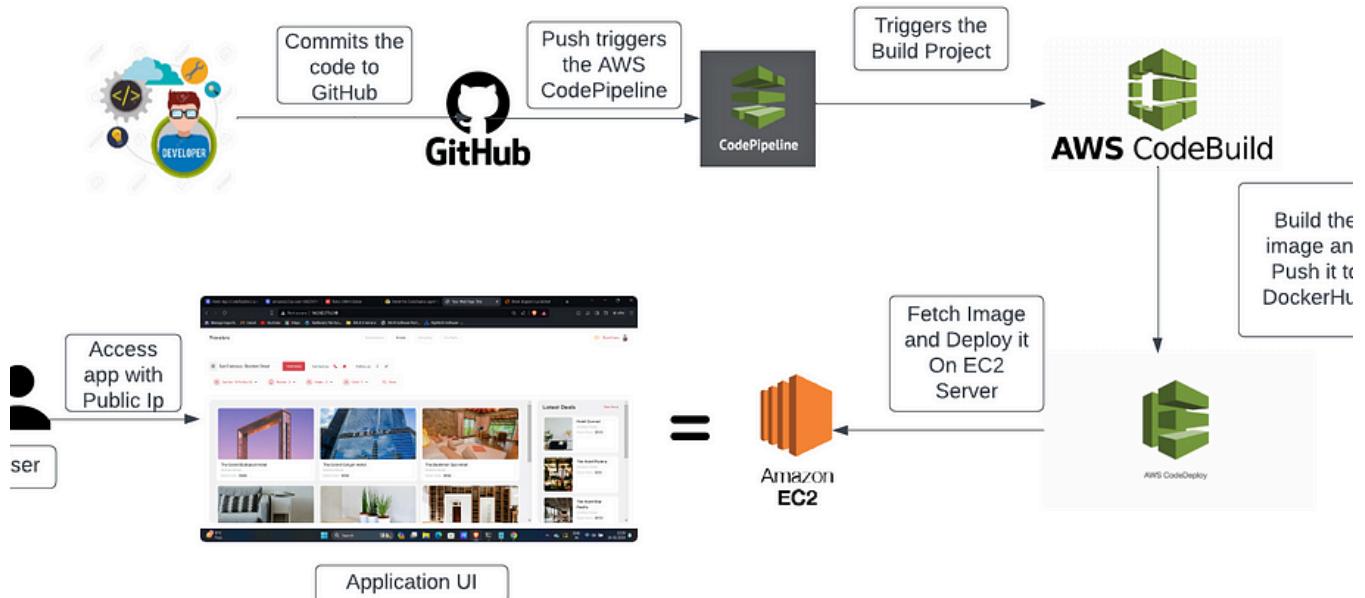
Sreedhar Reddy Madithati

Real Time CI/CD Pipeline for Java Application to Deploy on Apache Server.

Introduction:

Feb 18, 2024 138 2





Sreedhar Reddy Madithati

AWS CICD Pipeline: Deploy Application on EC2 with AWS Code Pipeline

Feb 14, 2024

84

3





Sreedhar Reddy Madithati

DevSecOps: Blue-Green Deployment of Swiggy-Clone on AWS ECS with AWS Code Pipeline

Introduction:

Apr 10, 2024 22 1

Sreedhar Reddy Madithati

DevSecOps: Zomato Clone Deployment in EKS with GitHub actions and Terraform

INTRODUCTION:

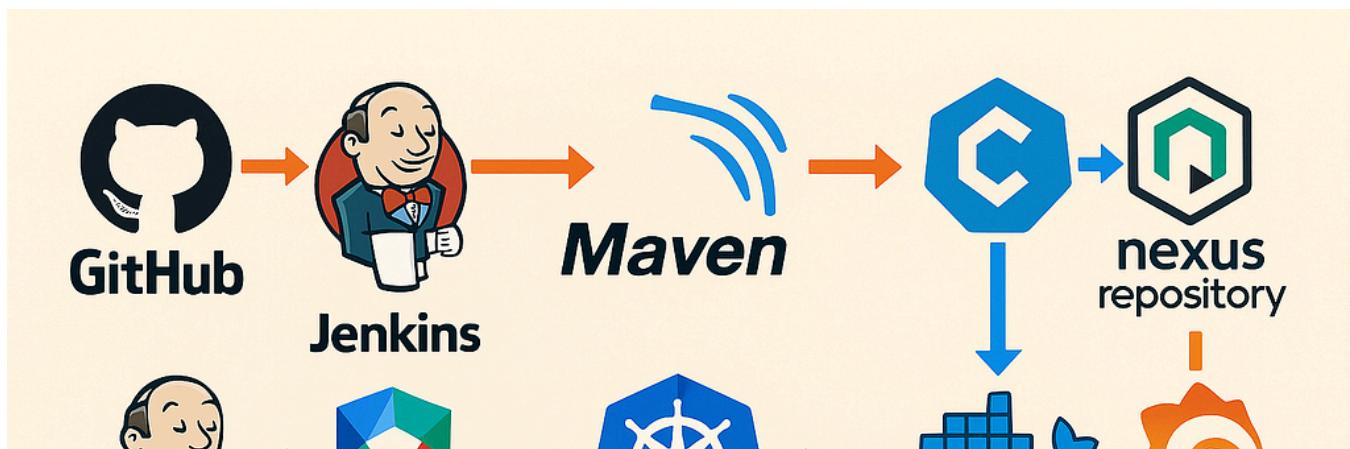
Mar 12, 2024

9



See all from Sreedhar Reddy Madithati

Recommended from Medium



AWS In AWS in Plain English by Master raj

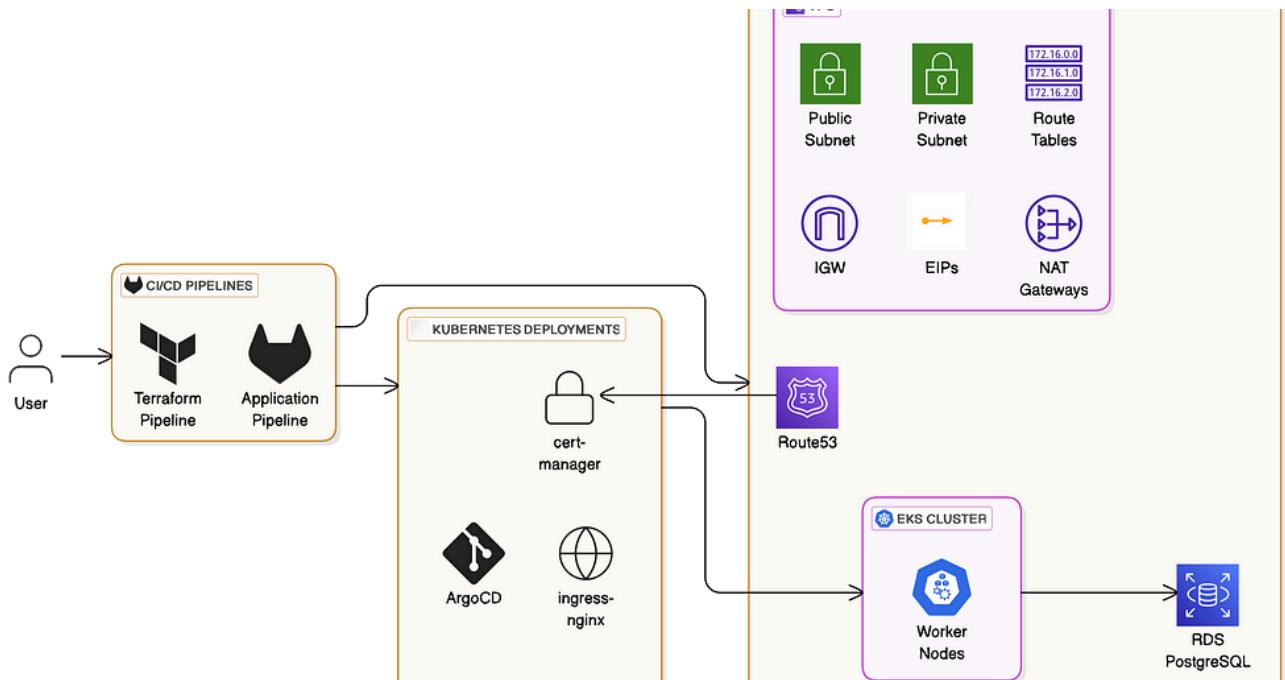
How I Built an End-to-End CI/CD Pipeline on AWS Using Jenkins, Docker, Kubernetes, SonarQube & More

This isn't just another tutorial. It's the exact pipeline I built—fully automated, security-checked, monitored, and deployed in a real...

Jul 8

6



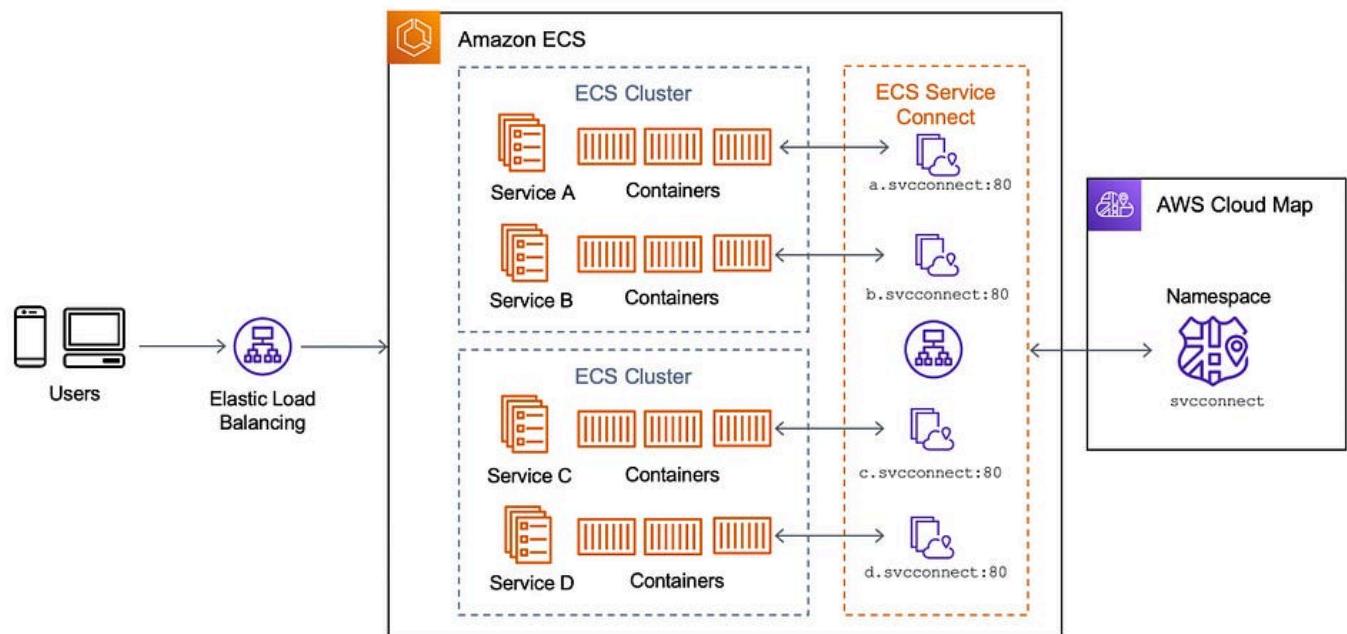


In FAUN.dev — Developer Community 🐾 by Mohamed ElEmam

Hands-On DevOps Project: Practice Full-Stack Deployment on AWS EKS with Terraform, Helm and more

Hands-On DevOps Project: Practice Full-Stack Deployment on AWS EKS with Terraform, Helm, SonarQube & GitLab CI/CD

Feb 18 103 1



Bhanu Reddy

ECS Best Practices— Cluster Design

1) Introduction :

⭐ Apr 2 ⌚ 10



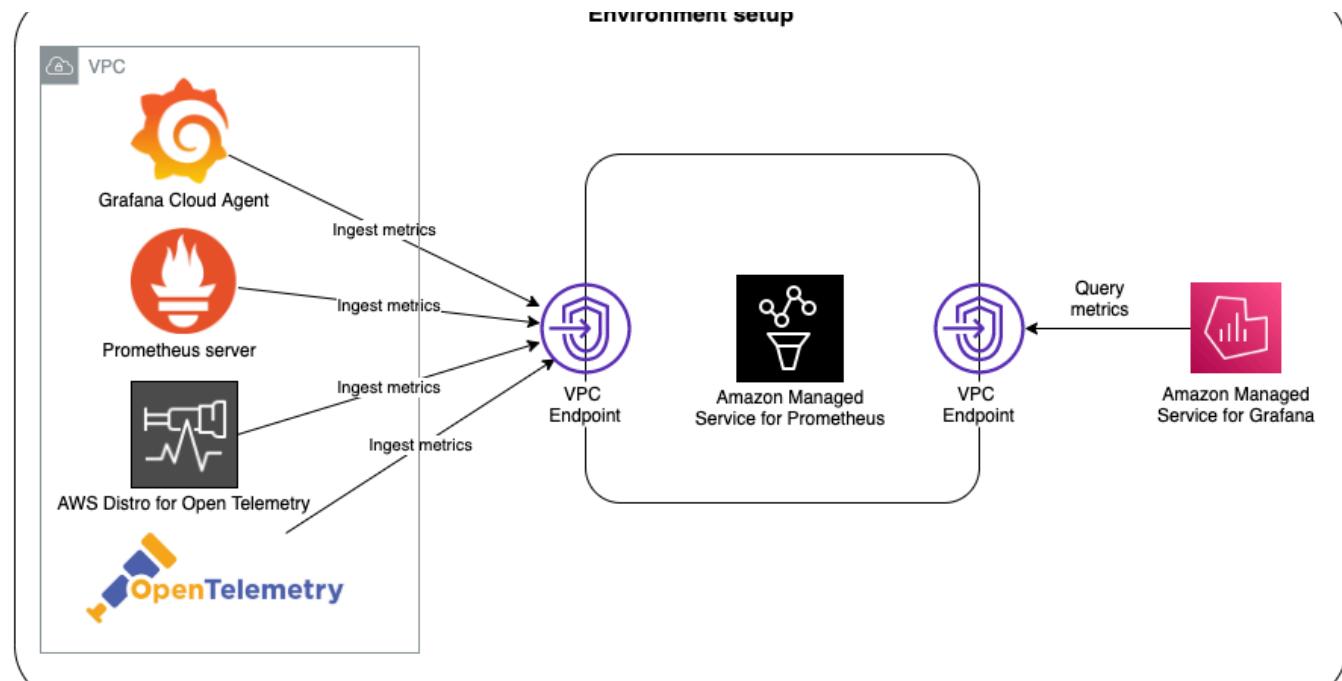
Enhance EKS Load Balancing. Enable IPVS Mode in AWS EKS

In Towards AWS by Ravi Kyada

Enhance EKS Load Balancing: Enable IPVS Mode in AWS EKS

If you're running Kubernetes in production, you already know that performance bottlenecks are the stuff of nightmares.

⭐ Apr 7 ⌚ 18



Nagarjun (Arjun) Nagesh

Monitoring and Logging in AWS: CloudWatch vs. Prometheus vs. ELK

Monitoring and logging are critical for ensuring application performance, security, and reliability in cloud environments. AWS offers...

Mar 11



**VPC
ENDPOINTS**

VS



**NAT
GATEWAY**

 Kinjal Thakkar

NAT Gateway or VPC Endpoints? AWS Cost Battle in Real ECS Deployments

Before we dive in, here's some context. In my previous blog, I discussed how VPC Endpoints could be a great alternative to NAT Gateways for...

Jul 8  5



See more recommendations