Member-only story

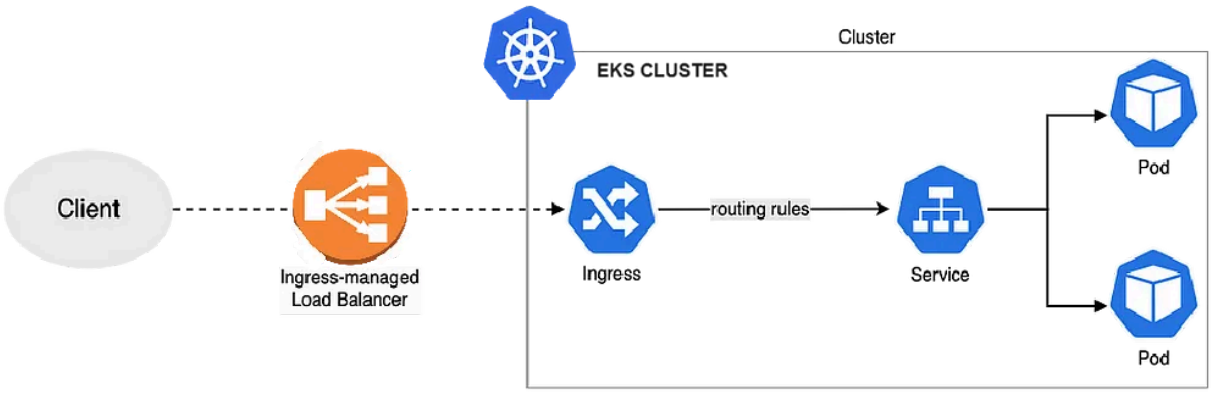
Path-Based Routing with AWS Load Balancer Controller: An Ingress Journey on Amazon EKS

@Harsh · Follow

7 min read · May 11, 2024

Listen Share More

In today's cloud-native landscape, orchestrating applications efficiently is paramount. Amazon Elastic Kubernetes Service (EKS) provides a robust platform for managing containerized workloads, while Ingress simplifies routing traffic to services within a cluster. In this comprehensive guide, we'll explore the intricacies of setting up the AWS Load Balancer Controller on Amazon EKS using Ingress, empowering you to streamline application delivery and enhance scalability.



Understanding AWS Load Balancer Controller:

In Kubernetes, an Ingress resource serves as a way to manage external access to services within the cluster. It acts as a layer of abstraction that defines how external traffic should be routed to different services based on rules specified by the user.

The AWS Load Balancer Controller enhances this functionality by integrating seamlessly with AWS Elastic Load Balancers (ELBs) and Application Load Balancers (ALBs), effectively extending the capabilities of Ingress resources to AWS cloud infrastructure.

By leveraging Ingress resources, the AWS Load Balancer Controller defines the rules and configurations for routing incoming traffic to backend services based on HTTP and HTTPS requests. Ingress resources serve as an abstraction layer, allowing users to define routing rules, TLS termination, and load balancing policies in a declarative manner.

In summary, the AWS Load Balancer Controller simplifies load balancing operations in Kubernetes clusters by automating the provisioning and management of AWS load balancers, while Ingress resources provide a declarative approach to defining routing rules and configurations for incoming traffic. Together, they form a powerful solution for efficient traffic management in cloud-native environments.

Before starting the steps, lets understand the flow of the process:

1. Create IAM Policy: The process begins with the creation of an IAM policy in AWS. This policy defines the permissions required for the AWS Load Balancer Controller to interact with AWS services such as ELB and ALB.
2. Attach Policy to Role: Next, the IAM policy is attached to an IAM role. This role will be assumed by the service account running in the Kubernetes cluster, allowing it to assume the necessary permissions to manage AWS resources
3. Create Kubernetes Service Account: In the Kubernetes cluster, a service account is created. This service account is associated with the IAM role created in the previous step using annotations.
4. Install AWS Load Balancer Controller: With the service account configured, the AWS Load Balancer Controller is installed in the Kubernetes cluster. This controller is responsible for synchronizing Kubernetes Ingress resources with AWS load balancers.

5. Define IngressClass: Define an IngressClass in Kubernetes, specifying parameters for the AWS load balancer such as type (e.g., application load balancer) and any additional annotations or settings.

6. Define Ingress Resources: In the Kubernetes cluster, Ingress resources are defined to specify how external traffic should be routed to different services within the cluster. These resources contain rules for path-based routing, TLS termination, and other configurations.

7. Controller Watches Ingress Resources: The AWS Load Balancer Controller continuously monitors the Kubernetes API server for changes to Ingress resources. When changes are detected, such as the creation or modification of an Ingress resource, the controller reacts accordingly.

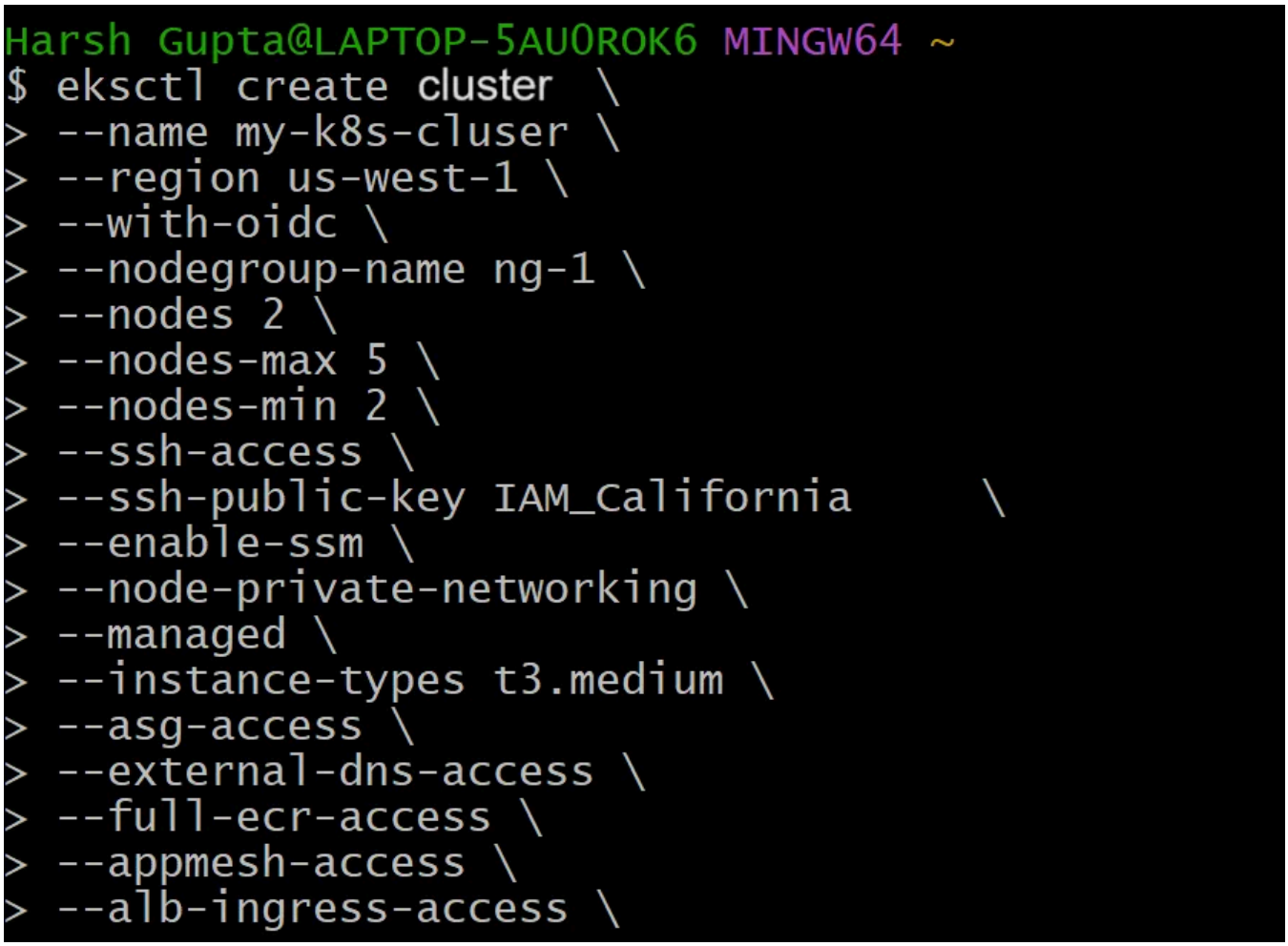
8. Provision AWS Load Balancer: Based on the definitions specified in the Ingress resources, the AWS Load Balancer Controller provisions AWS load balancers (ELB or ALB) in the AWS account. The controller configures the load balancers to match the desired settings specified in the Ingress.

9. Route Traffic to Kubernetes Services: Once the AWS load balancers are provisioned and configured, external traffic is routed to the appropriate Kubernetes services within the cluster based on the rules defined in the Ingress resources.

10. Monitor and Manage Load Balancers: The AWS Load Balancer Controller continues to monitor the status of AWS load balancers and ensures that they remain in sync with the corresponding Ingress resources in Kubernetes. Any changes or updates to the Ingress resources are automatically reflected in the configuration of the AWS load balancers.

Launch EKS Cluster

- Launch your cluster with following necessary permissions.




```
eksctl create cluster \
 --name my-k8s-cluser \
 --region us-west-1 \
 --with-oidc \
 --nodegroup-name ng-1 \
 --nodes 2 \
 --nodes-max 5 \
 --nodes-min 2 \
 --ssh-access \
 --ssh-public-key <Your-Key-Name> \
 --enable-ssm \
 --node-private-networking \
 --managed \
 --instance-types t3.medium \
 --asg-access \
 --external-dns-access \
 --full-ecr-access \
 --appmesh-access \
 --alb-ingress-access \
```

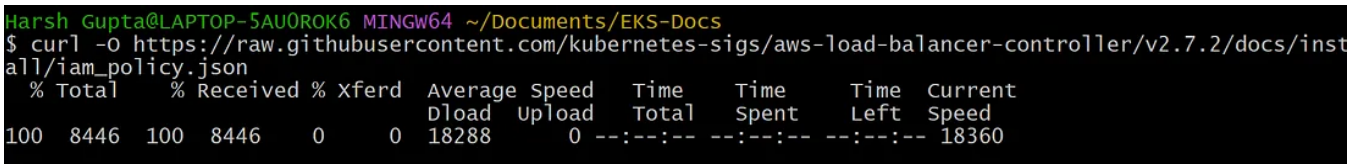
Preparing the Environment:

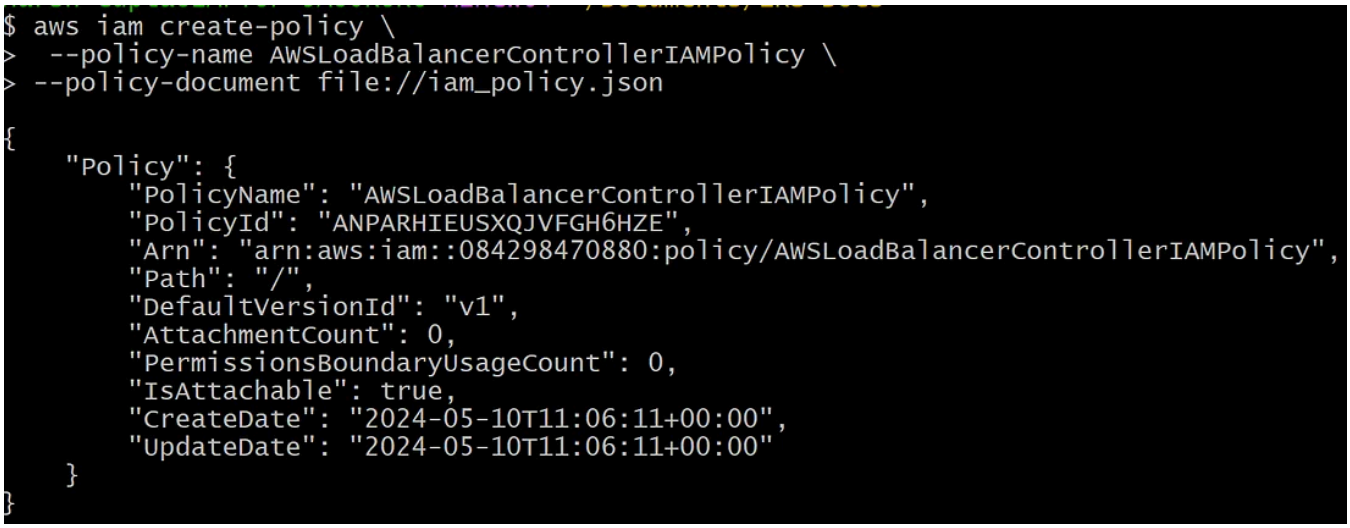
Creating IAM Policy:

Begin by crafting a custom IAM policy granting the necessary permissions for the AWS Load Balancer Controller to manage resources in your AWS environment.

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

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





Creating Service Account:

Utilize `eksctl` to create a dedicated Kubernetes service account associated with the IAM policy, ensuring secure access to AWS resources.

```
 eksctl create iamserviceaccount \
 --cluster=<cluster-name> \
 --region <region-name> \
 --namespace=kube-system \
 --name=aws-load-balancer-controller \
 --role-name AmazonEKSLoadBalancerControllerRole \
 --attach-policy-arn=<Your-Policy-ARN> \
 --approve
```

```
Harsh Gupta@LAPTOP-5AUOROK6 MINGW64 ~/Documents/EKS-Docs
$ eksctl create iamserviceaccount \
> --cluster=my-k8s-cluser \
> --region us-west-1 \
> --namespace=kube-system \
> --name=aws-load-balancer-controller \
> --role-name AmazonEKSLoadBalancerControllerRole \
> --attach-policy-arn=arn:aws:iam::084298470880:policy/AWSLoadBalancerControllerIAMPolicy \
> --approve

2024-05-10 16:42:28 [ℹ]  1 existing iamserviceaccount(s) (kube-system/aws-node) will be excluded
2024-05-10 16:42:28 [ℹ]  1 iamserviceaccount (kube-system/aws-load-balancer-controller) was included (bas
ed on the include/exclude rules)
2024-05-10 16:42:28 [!]  serviceaccounts that exist in Kubernetes will be excluded, use --override-existi
ng-serviceaccounts to override
2024-05-10 16:42:28 [ℹ]  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",
    } }2024-05-10 16:42:28 [ℹ]  building iamserviceaccount stack "eksctl-my-k8s-cluser-addon-iamserviceac
count-kube-system-aws-load-balancer-controller"
2024-05-10 16:42:29 [ℹ]  deploying stack "eksctl-my-k8s-cluser-addon-iamserviceaccount-kube-system-aws-lo
ad-balancer-controller"
```

- Replace the <cluster-name>, <region-name>, <Your-Policy-ARN> with the original one.

```
$ kubectl describe sa/aws-load-balancer-controller -n kube-system
Name:                aws-load-balancer-controller
Namespace:           kube-system
Labels:              app.kubernetes.io/managed-by=eksctl
Annotations:         eks.amazonaws.com/role-arn: arn:aws:iam::084298470880:role/AmazonEKSLoadBalancerCont
rollerRole
Image pull secrets:  <none>
Mountable secrets:   <none>
Tokens:              <none>
Events:              <none>
```

Installing AWS Load Balancer Controller:

Harnessing the power of Helm, deploy the AWS Load Balancer Controller to your Amazon EKS cluster effortlessly. Helm charts simplify the installation process, providing configurable options for fine-tuning controller behavior.

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

```
helm repo update eks
```

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
 -n kube-system \
 --set clusterName=my-k8s-cluser \
 --set serviceAccount.create=false \
 --set serviceAccount.name=aws-load-balancer-controller
```

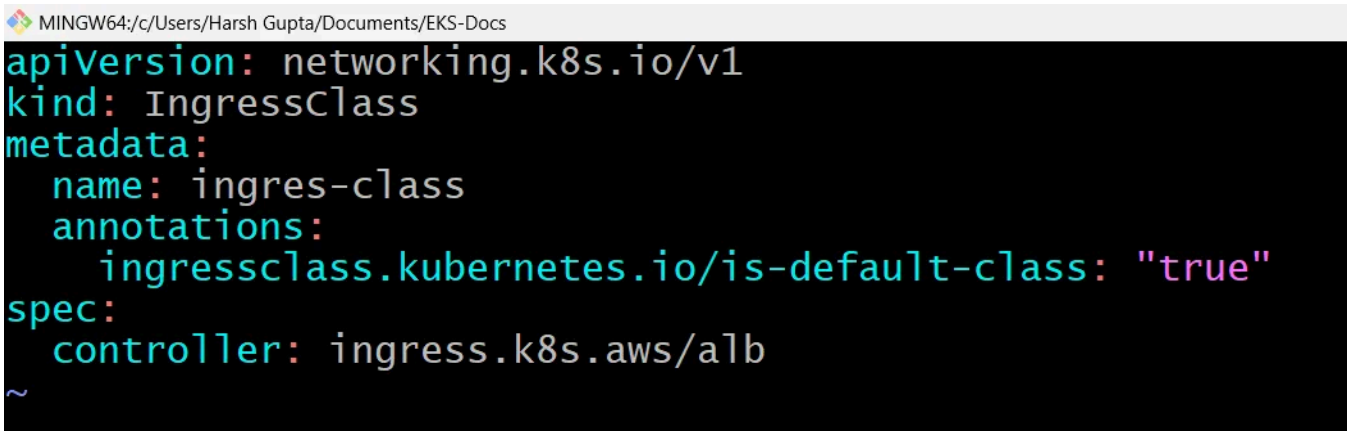
```
Harsh Gupta@LAPTOP-5AU0ROK6 MINGW64 ~/Documents/EKS-Docs
$ kubectl get pods -n kube-system
NAME                                            READY   STATUS    RESTARTS   AGE
aws-load-balancer-controller-6c9ff6755f-29shf   1/1     Running   0          95s
aws-load-balancer-controller-6c9ff6755f-tc624   1/1     Running   0          95s
aws-node-pjsw5                                  2/2     Running   0          71m
aws-node-s6qmb                                  2/2     Running   0          71m
coredns-7fc597d7cd-fqvz4                        1/1     Running   0          78m
coredns-7fc597d7cd-jk56s                        1/1     Running   0          78m
kube-proxy-gnct6                                1/1     Running   0          71m
kube-proxy-vjcrl                                1/1     Running   0          71m
```

Configuring Ingress Class:

Define an Ingress class tailored to the AWS Load Balancer Controller, specifying it as the ingress controller for path-based routing. This crucial step establishes the link between Ingress resources and AWS load balancers.

- Create a file `ingress-class.yaml`

```yaml
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: ingres-class
  annotations:
    ingressclass.kubernetes.io/is-default-class: "true"
spec:
  controller: ingress.k8s.aws/alb
```



- Here we specify our controller that will connect us to the AWS Load Balancer.

```
kubectl apply -f ingress-class.yaml
```

```
Harsh Gupta@LAPTOP-5AU0ROK6 MINGW64 ~/Documents/EKS-Docs
$ kubectl apply -f ingress-class.yaml
ingressclass.networking.k8s.io/ingres-class created

Harsh Gupta@LAPTOP-5AU0ROK6 MINGW64 ~/Documents/EKS-Docs
$ kubectl get ingressclass/ingres-class
NAME           CONTROLLER            PARAMETERS   AGE
ingres-class   ingress.k8s.aws/alb   <none>       24s

Harsh Gupta@LAPTOP-5AU0ROK6 MINGW64 ~/Documents/EKS-Docs
$ kubectl describe ingressclass/ingres-class
Name:         ingres-class
Labels:       <none>
Annotations:  ingressclass.kubernetes.io/is-default-class: true
Controller:   ingress.k8s.aws/alb
Events:       <none>
```

Deploying and Exposing Applications:

With the infrastructure in place, deploy your applications onto the Amazon EKS cluster. Expose the services to external traffic by creating Kubernetes Services, facilitating seamless communication with the outside world.

```
kubectl create deployment web --image=gcr.io/google-samples/hello-app:1.0
kubectl create deployment web2 --image=gcr.io/google-samples/hello-app:2.0
```

```
kubectl expose deployment web --type=NodePort --port=8080
kubectl expose deployment web2 --port=8080 --type=NodePort
```

```
Harsh Gupta@LAPTOP-5AU0ROK6 MINGW64 ~/Documents/EKS-Docs
$ kubectl create deployment web --image=gcr.io/google-samples/hello-app:1.0
deployment.apps/web created

Harsh Gupta@LAPTOP-5AU0ROK6 MINGW64 ~/Documents/EKS-Docs
$ kubectl create deployment web2 --image=gcr.io/google-samples/hello-app:2.0
deployment.apps/web2 created

Harsh Gupta@LAPTOP-5AU0ROK6 MINGW64 ~/Documents/EKS-Docs
$ kubectl expose deployment web --type=NodePort --port=8080
service/web exposed

Harsh Gupta@LAPTOP-5AU0ROK6 MINGW64 ~/Documents/EKS-Docs
$ kubectl expose deployment web2 --port=8080 --type=NodePort
service/web2 exposed
```

```
Harsh Gupta@LAPTOP-5AU0ROK6 MINGW64 ~/Documents/EKS-Docs
$ kubectl get deployment
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
web    1/1     1            1           5m11s
web2   1/1     1            1           4m55s

Harsh Gupta@LAPTOP-5AU0ROK6 MINGW64 ~/Documents/EKS-Docs
$ kubectl get  service
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
kubernetes   ClusterIP   10.100.0.1       <none>        443/TCP          5m45s
web          NodePort    10.100.112.235   <none>        8080:31837/TCP   4m35s
web2         NodePort    10.100.246.32    <none>        8080:32436/TCP   4m16s
```

Creating Ingress Resources:

Craft Ingress resources to define routing rules for incoming traffic. Leverage path-based routing to direct requests to the appropriate backend services based on URL paths.

Here we also define many annotations that will define our Application Load balancer such as load-balancer-name, healthcheck, scheme and many more.

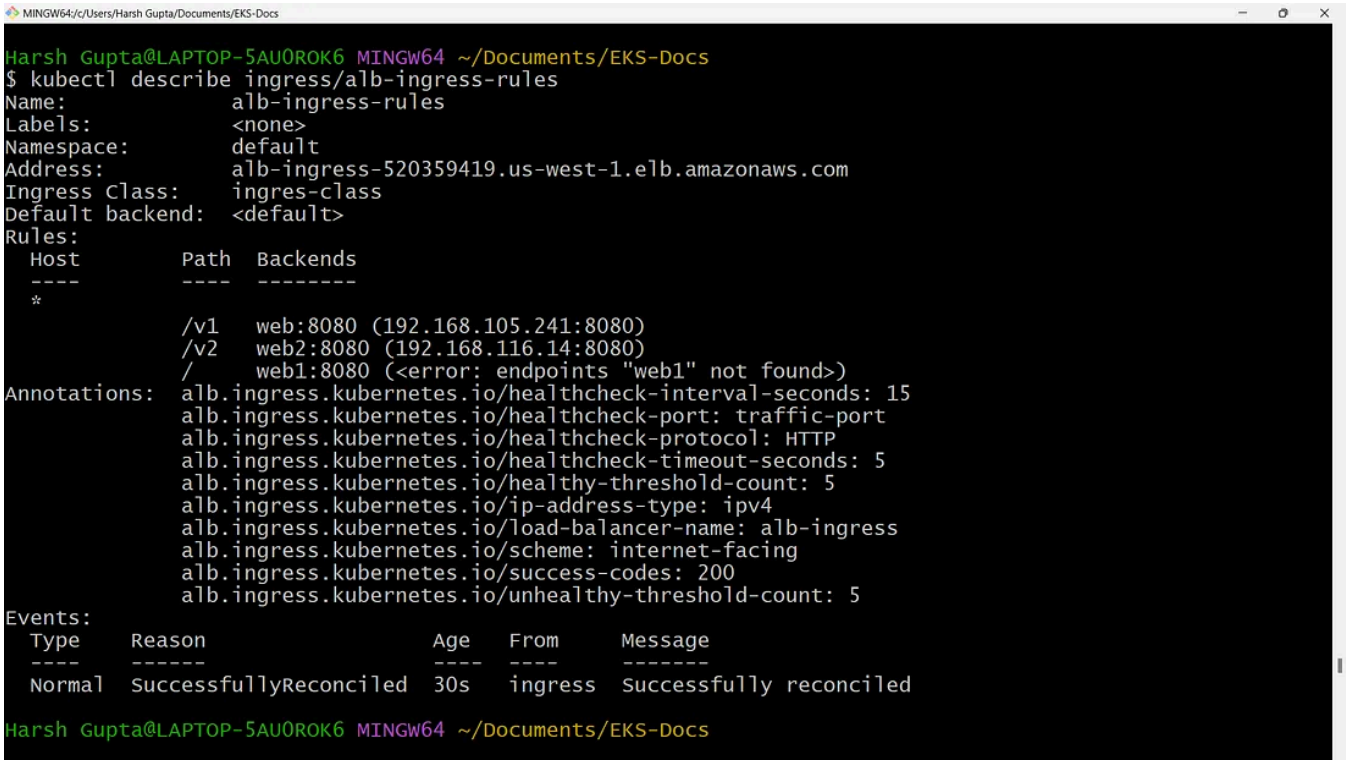
```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: alb-ingress-rules
  annotations:
    alb.ingress.kubernetes.io/load-balancer-name: "alb-ingress"
    alb.ingress.kubernetes.io/ip-address-type: ipv4
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/healthcheck-protocol: HTTP
    alb.ingress.kubernetes.io/healthcheck-port: traffic-port
    alb.ingress.kubernetes.io/healthcheck-interval-seconds: '15'
    alb.ingress.kubernetes.io/healthcheck-timeout-seconds: '5'
    alb.ingress.kubernetes.io/healthy-threshold-count: '5'
    alb.ingress.kubernetes.io/unhealthy-threshold-count: '5'
    alb.ingress.kubernetes.io/success-codes: '200'
spec:
  ingressClassName: ingres-class
  rules:
    - http:
        paths:
          - path: /v1
            pathType: Prefix
            backend:
              service:
                name: web
                port:
                  number: 8080
          - path: /v2
            pathType: Prefix
            backend:
```

```
            service:
              name: web2
              port:
                number: 8080
          - path: /
            pathType: Prefix
            backend:
              service:
                name: web
                port:
                  number: 8080
```

- Backend means where our app is running and that is our service that we created.

Apply this file:

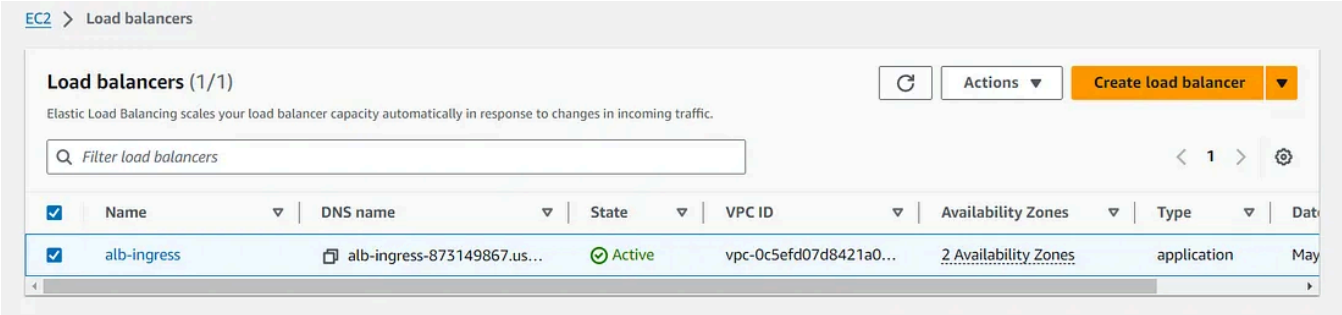
```
kubectl apply -f ingress.yaml
```

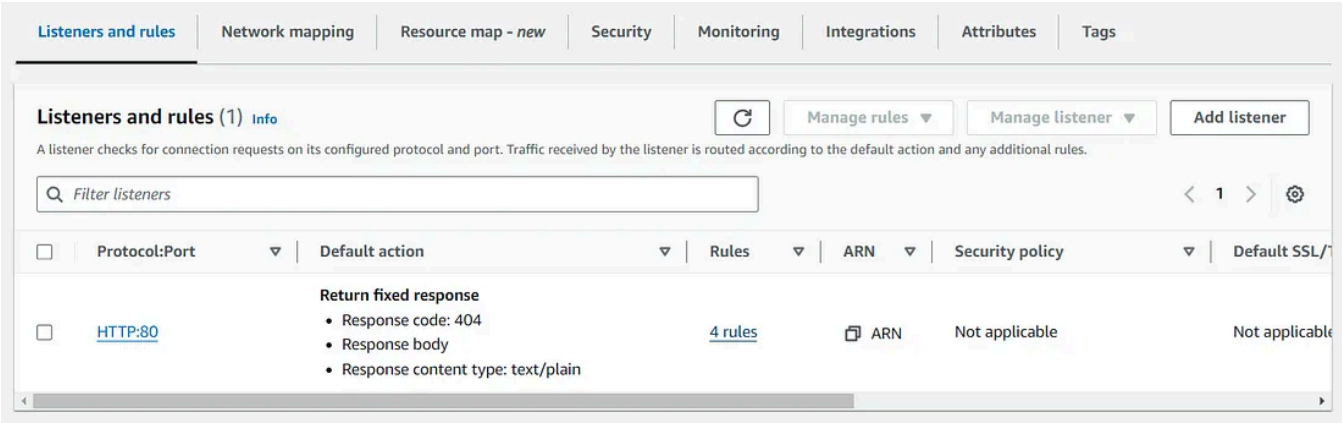


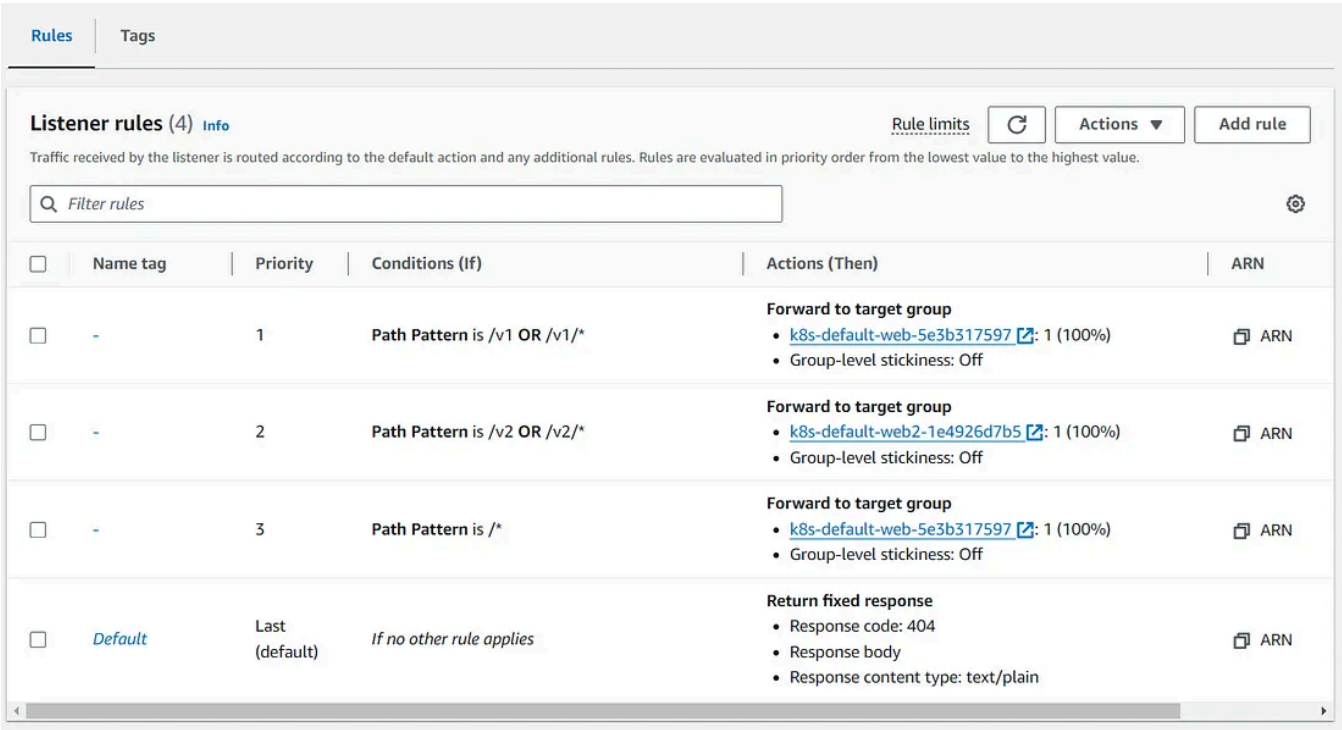
Observing Automatic ALB Creation:

Upon creating the Ingress resources, observe the magic unfold as AWS Load Balancer Controller automatically provisions ALBs to serve as entry points for

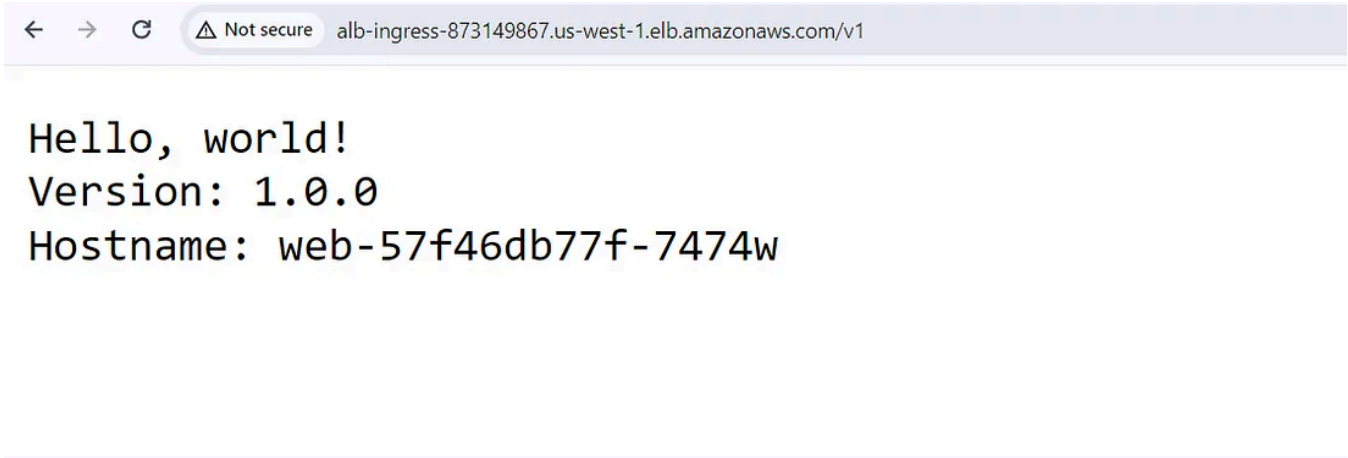
incoming traffic. This automated process simplifies load balancing and enhances scalability without manual intervention.

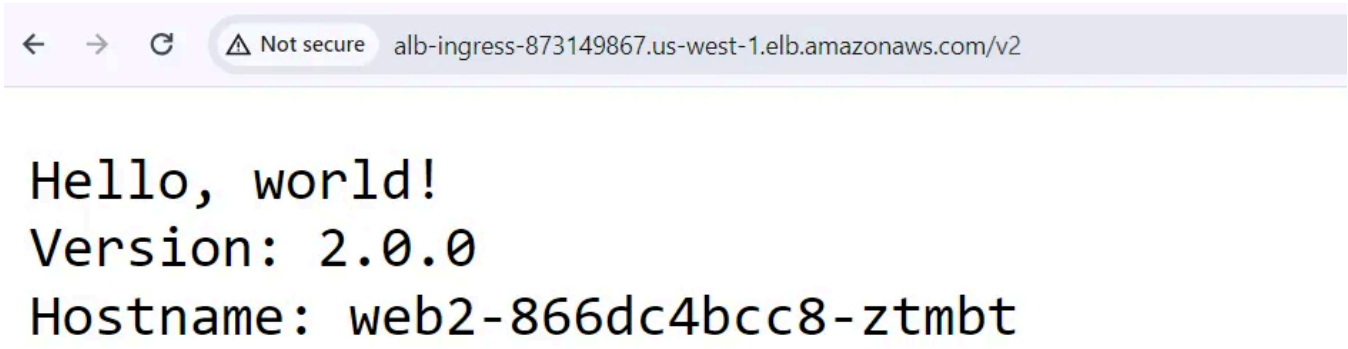






Test the EndPoint and ingress-routing.





By following these comprehensive steps, you'll master the deployment of AWS Load Balancer Controller on Amazon EKS, unlocking the full potential of Ingress for seamless application delivery. Elevate your Kubernetes infrastructure with robust load balancing capabilities, empowering your applications to scale effortlessly in the cloud-native ecosystem. 🚀

Amazon Eks | DevOps | Kubernetes | Containerization | AWS

Follow

Written by @Harsh

725 Followers · 0 Following

A devOps engineer from India

Responses (2)

t Raajatech

What are your thoughts?

Abdelfattah Sekak
May 16, 2024

Adding to my knowledge, cheers!

1 Reply

Walid LARABI
Nov 2, 2024

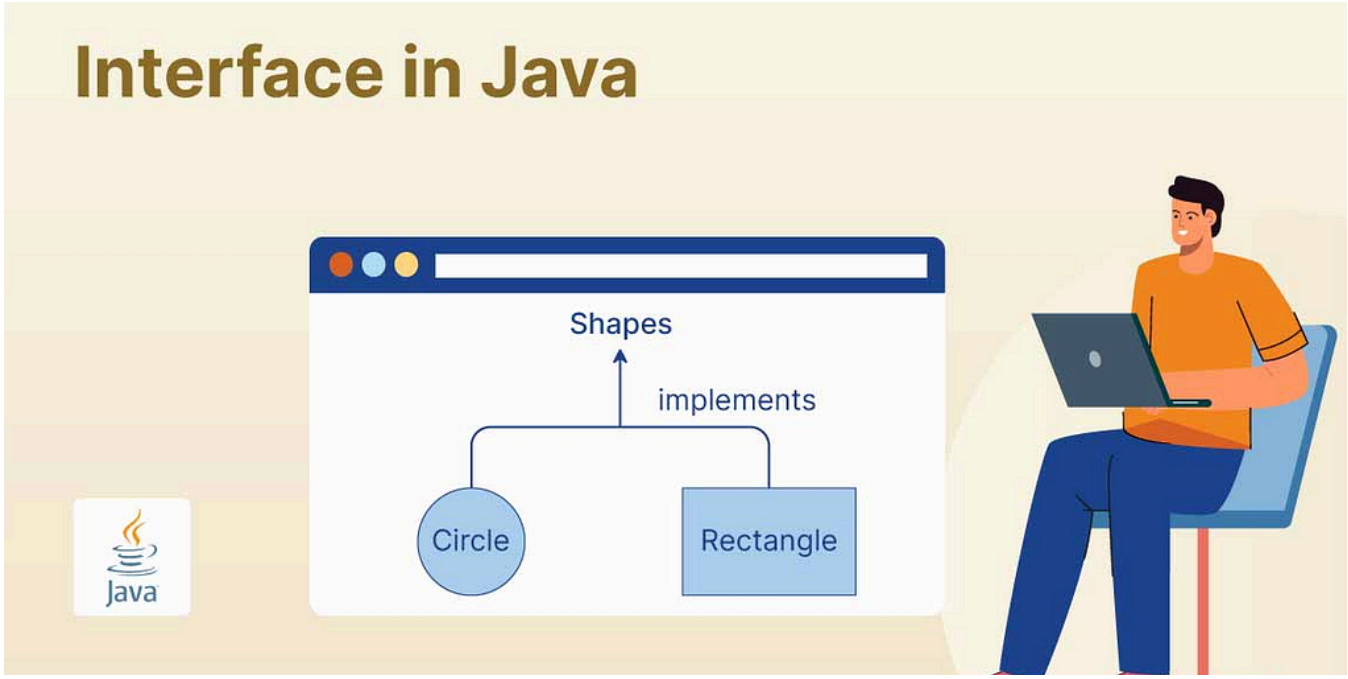
Great article,

I've added you as a writer to https://medium.com/weeklycloud publication, if you want to publish your article there ! :)

Have a nice day

Reply

More from @Harsh

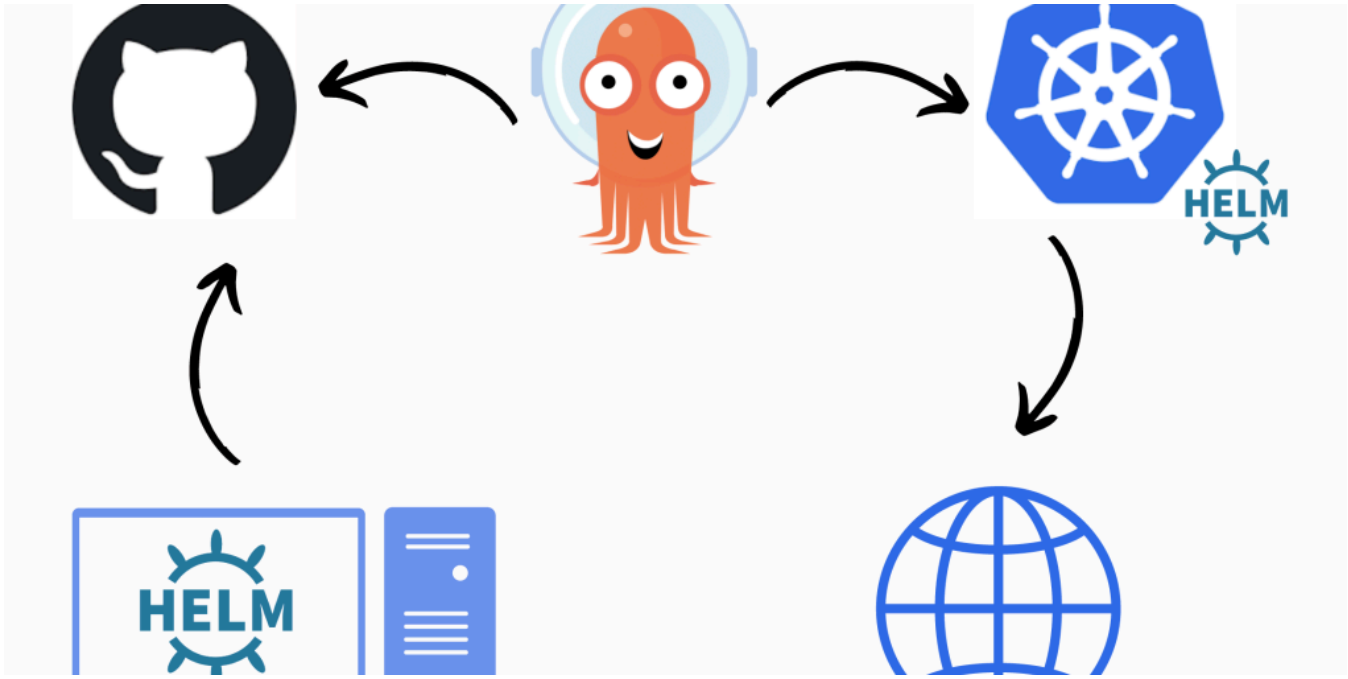


@Harsh

Abstract Classes vs Interfaces in Java: When and How to Use Them

As I continue deepening my understanding of Java, I recently revisited two foundational concepts: abstract classes and interfaces. These...

Oct 10, 2024 167 5

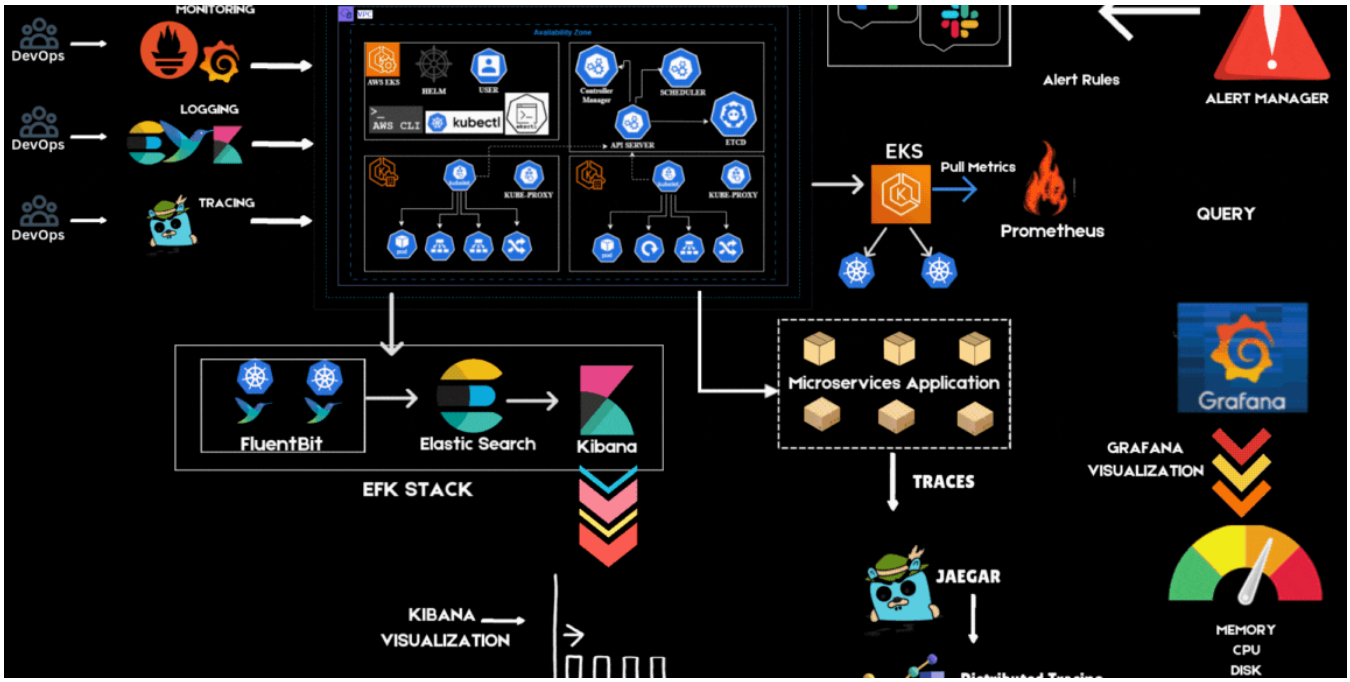


@Harsh

Using Helm Charts with ArgoCD: Streamlining Kubernetes Deployments

In this blog, we will explore the integration of Helm and ArgoCD, a powerful combination for managing Kubernetes applications. Helm...

Dec 3, 2024 88 1

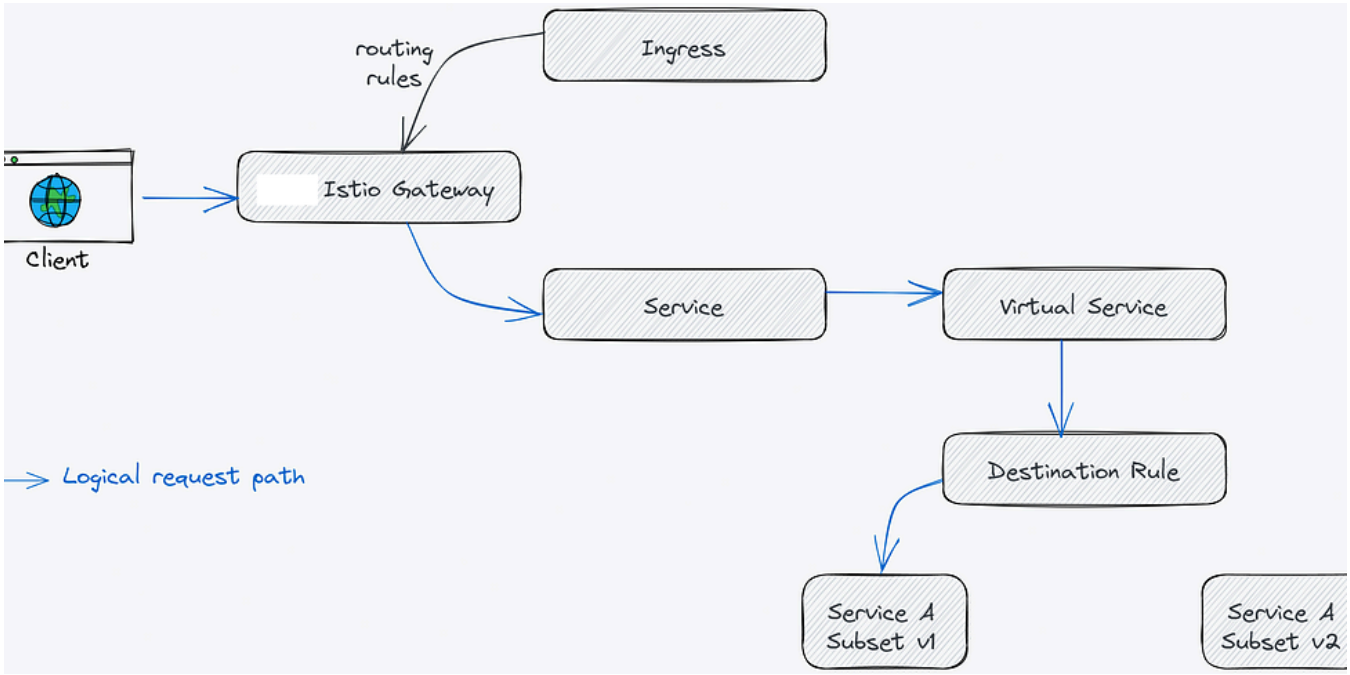


In DevOps.dev by @Harsh

Building a Full-Scale Observability Stack on EKS with Prometheus, Grafana, EFK, and Jaeger

What is Observability?

Apr 8 13



@Harsh

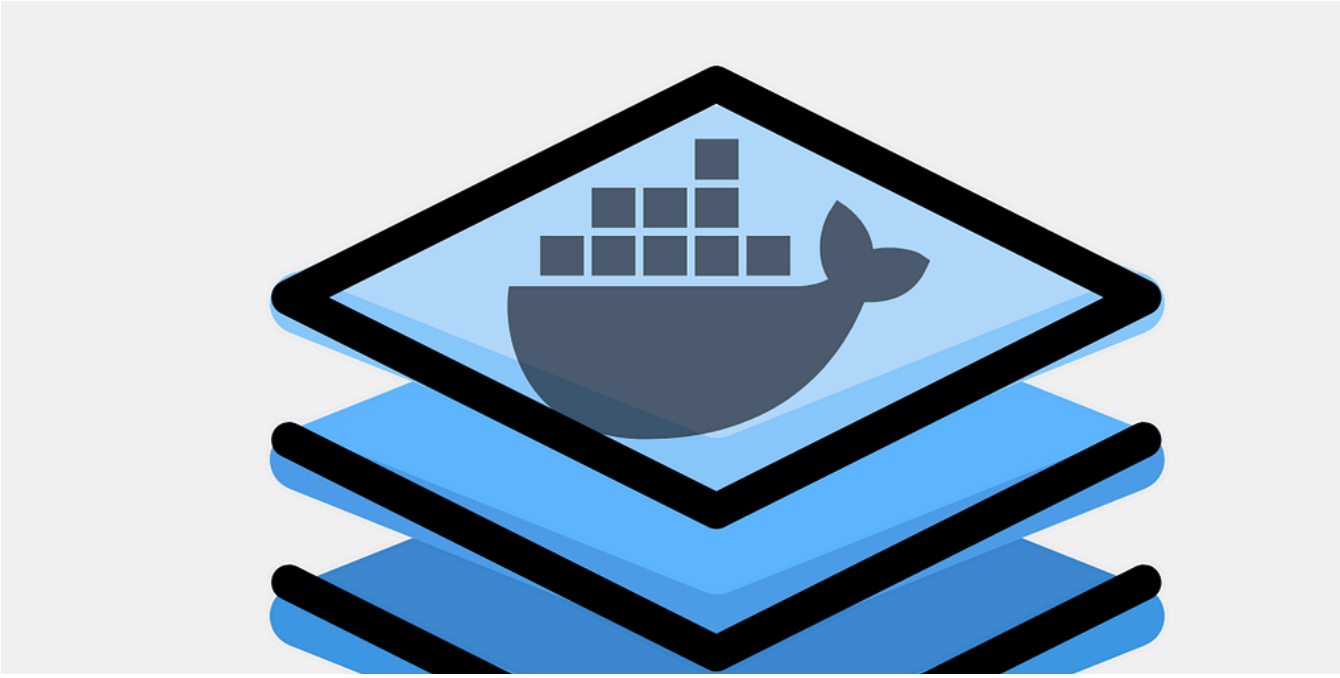
Understanding Ingress Gateway in Istio: A Detailed Guide

Istio, as a service mesh, streamlines service-to-service communication, security, and observability within a microservices environment...

Nov 21, 2024 50

See all from @Harsh

Recommended from Medium

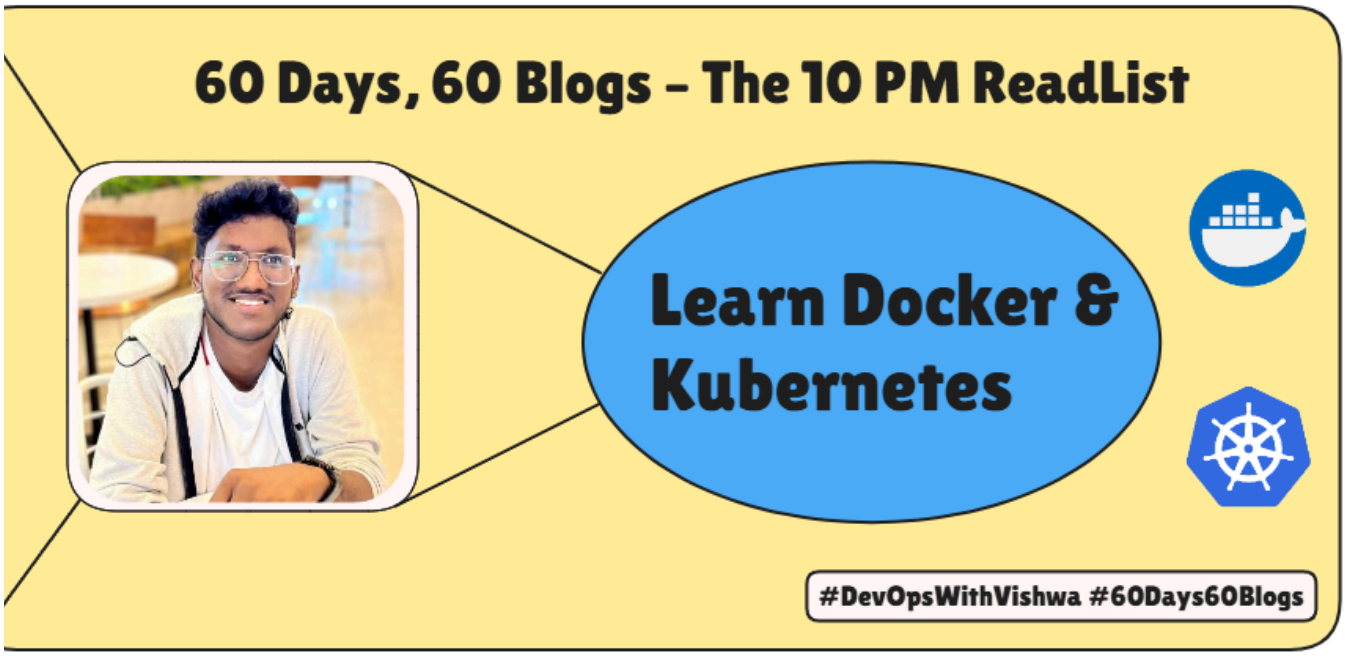


In AWS in Plain English by The Devops Girl

Mastering Networking Troubleshooting in Containers: Common Interview Questions and How to Tackle...

In the world of containerized applications, networking is a critical component that ensures smooth communication between services. Whether...

Nov 22, 2024 57 1

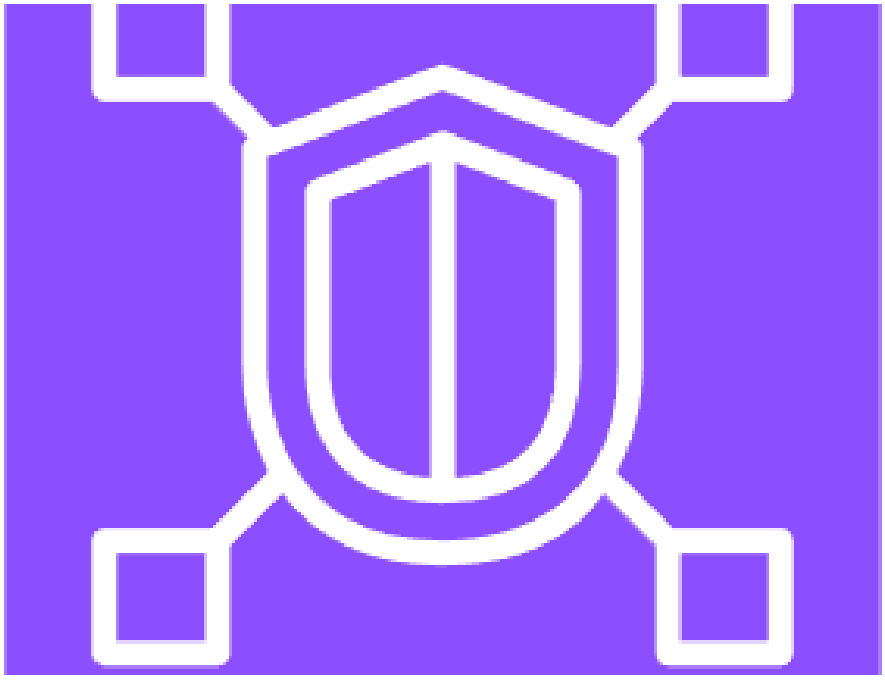


VISHWA S

Kubernetes Services: Why and How it Works?

Services in Kubernetes provide a way to expose applications running on Pods to the outside world or to other components within the cluster...

Apr 5

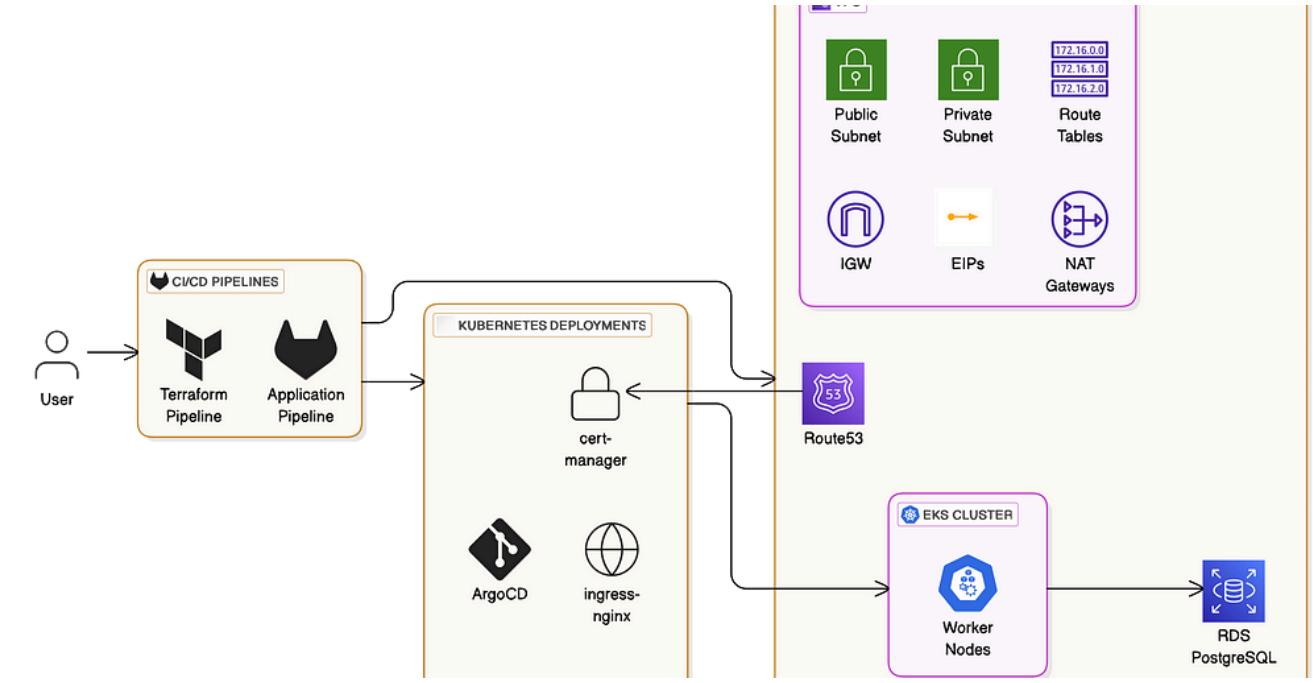


Kiran Chhablani

Amazon VPC Lattice: Simplifying Service-to-Service Communication

Secure, Scalable, and Hassle-Free Networking for Modern Applications

Dec 19, 2024 11

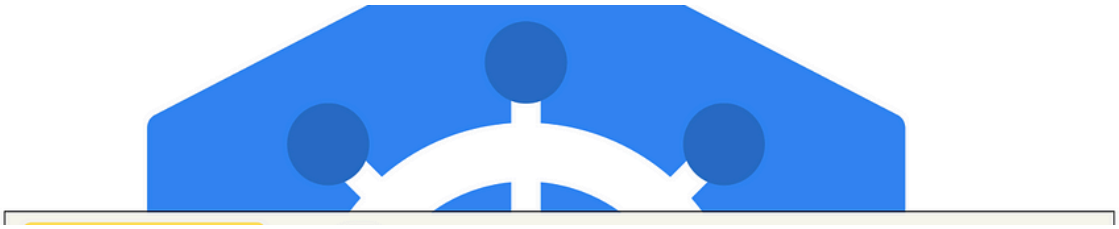


In FAUN—Developer Community by Mohamed ElEmam

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 57 1

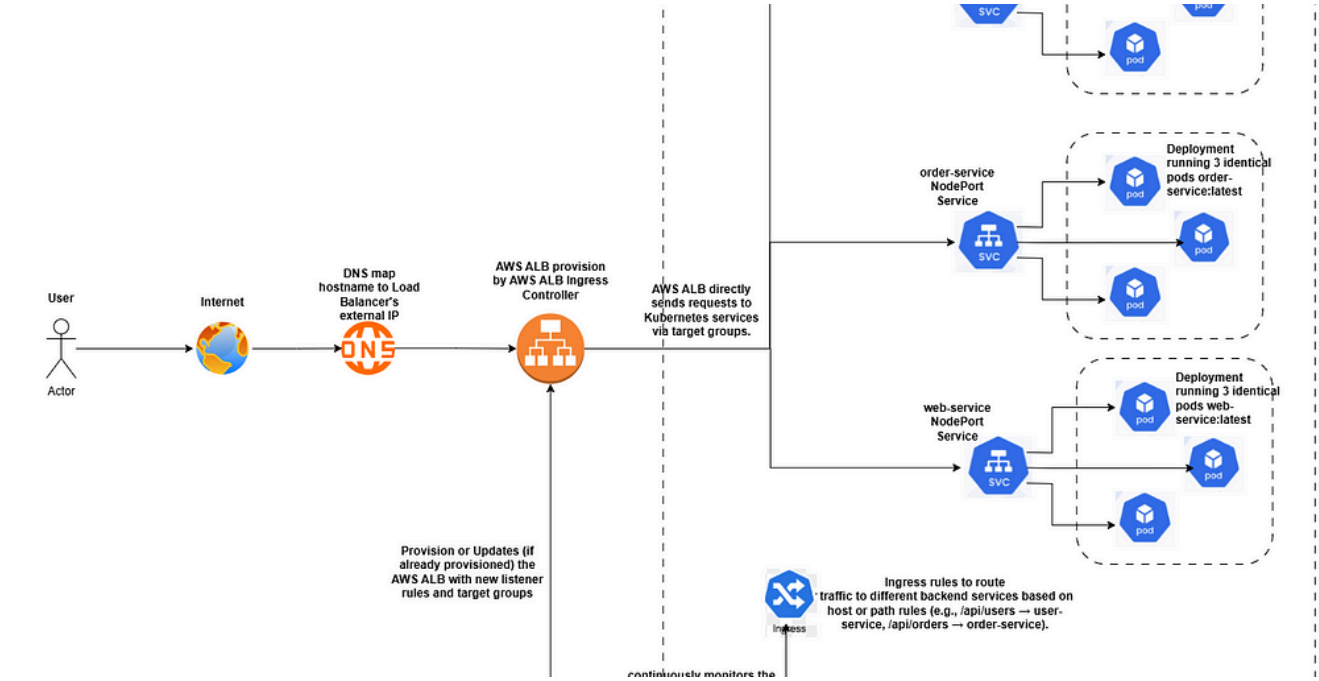


Devops Diaries

Top 5 Kubernetes Errors and How to Fix Them Like a Pro

Ever deployed a Kubernetes app.

3d ago



Anil Goyal

Understanding AWS ALB Ingress Controller in Kubernetes: End-to-End Request Flow

Lets discuss in detail how we can manage external traffic in kubernetes in a microservices environment.

Mar 28 22

See more recommendations