



EKS

Kubernetes on AWS





“Make this easier for me”



“Native AWS Integrations.”



"An Open Source Kubernetes Experience."



Amazon EKS

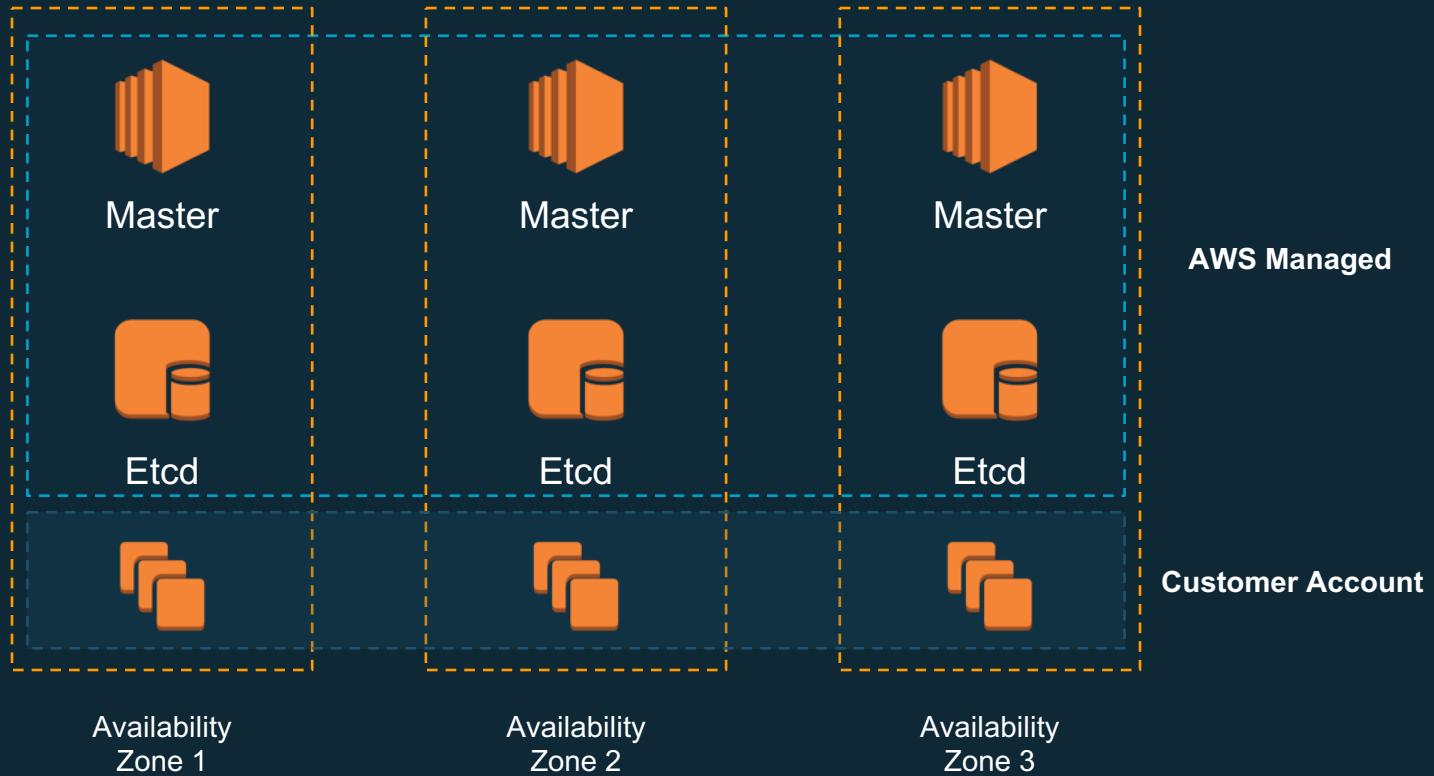
ELASTIC CONTAINER SERVICE FOR KUBERNETES
(EKS)

EKS is Kubernetes Certified

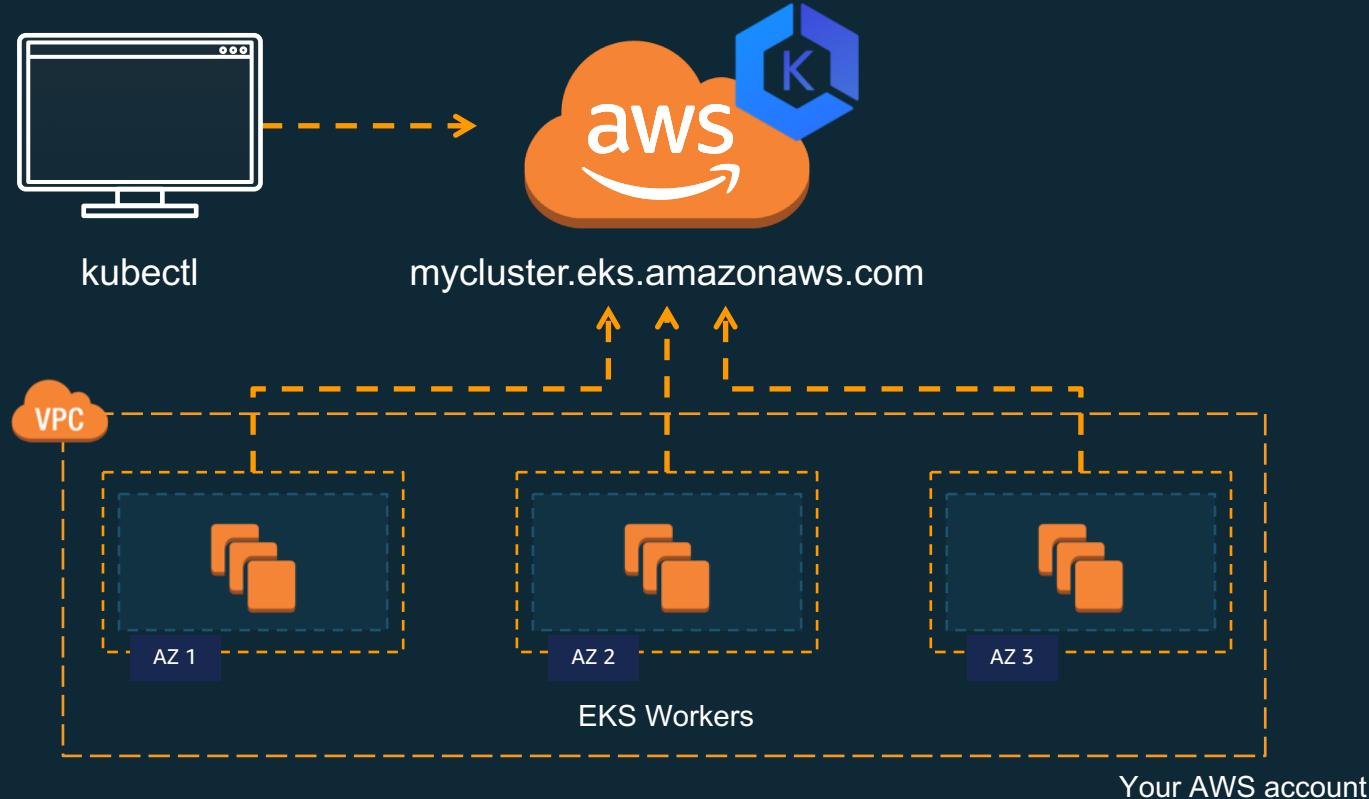


Architecture

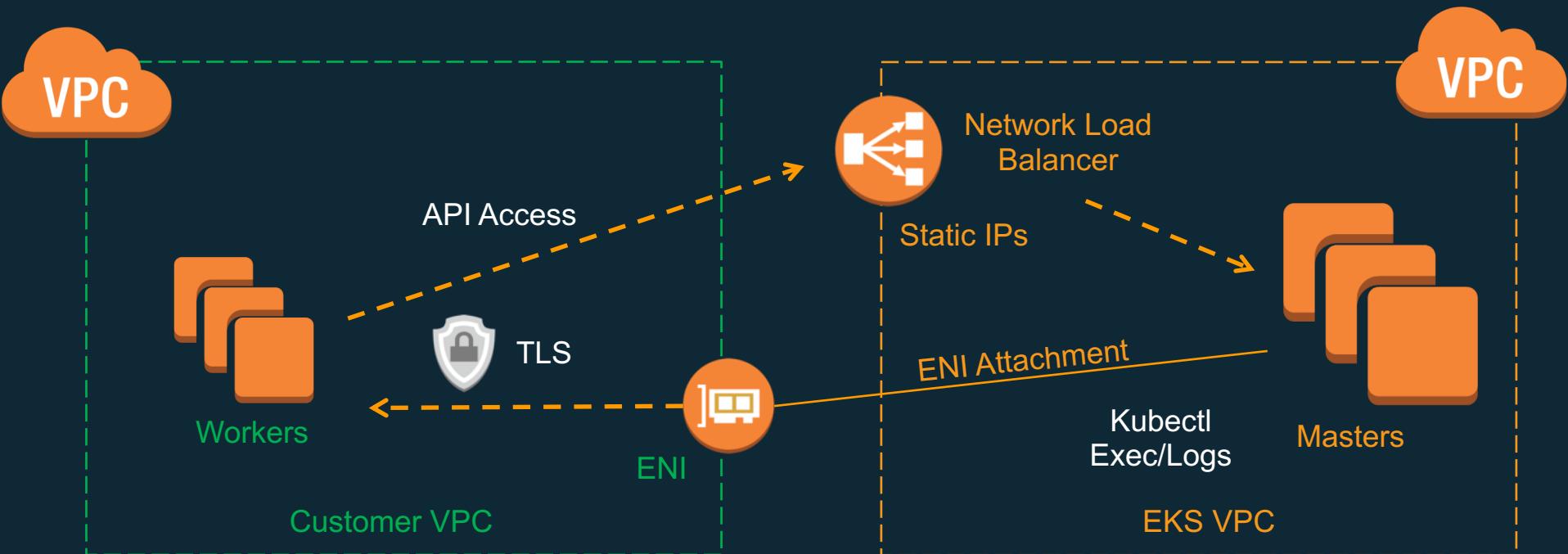




Amazon EKS



Cross-account Kubernetes



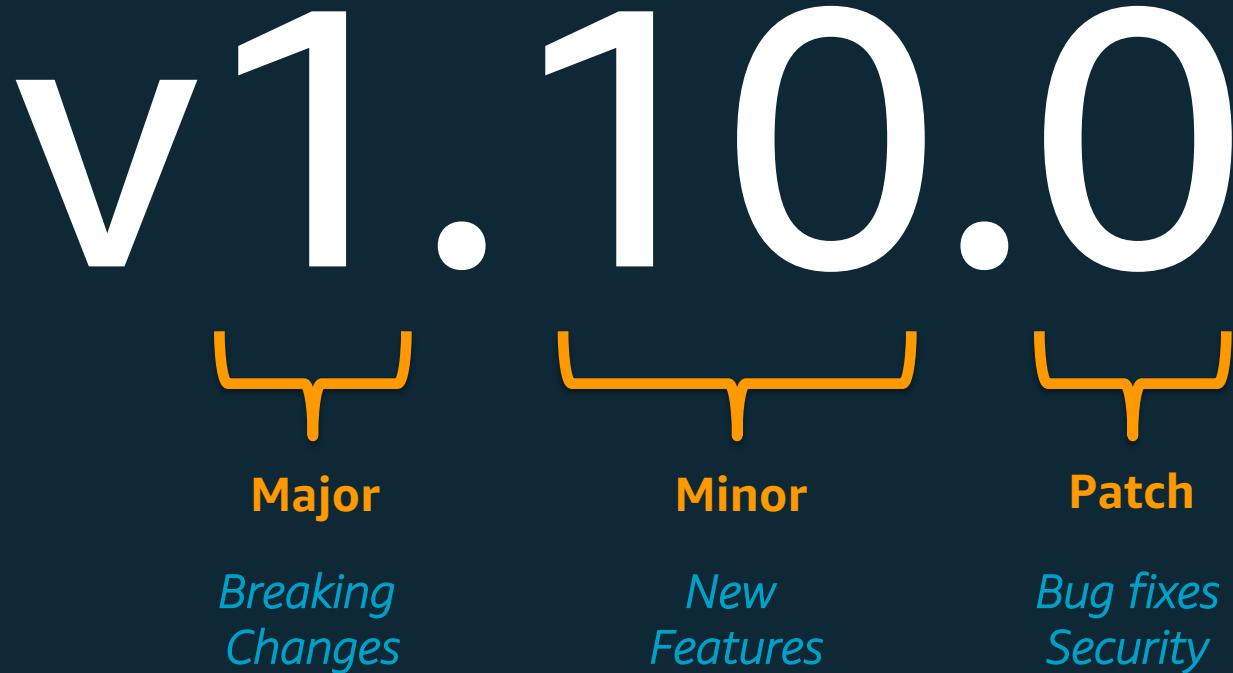
EKS Cross-Account Networking: Availability Zones



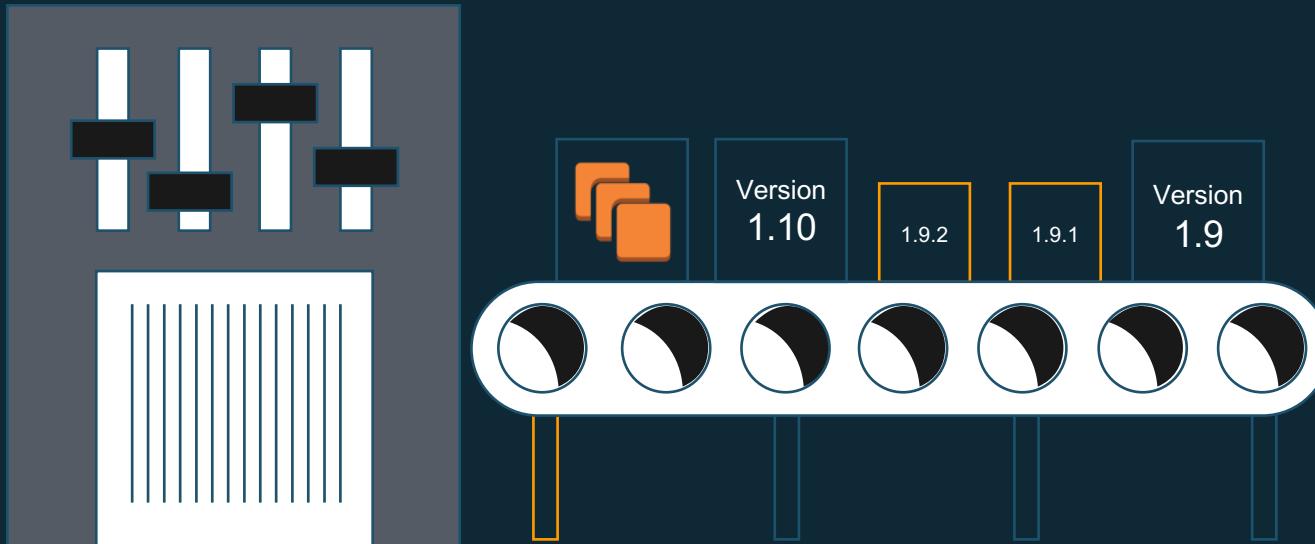
Versions and Upgrades



Semantic Versioning (semver)



Kubernetes Upgrades



EKS Networking





CNI



Native VPC networking
with CNI plugin



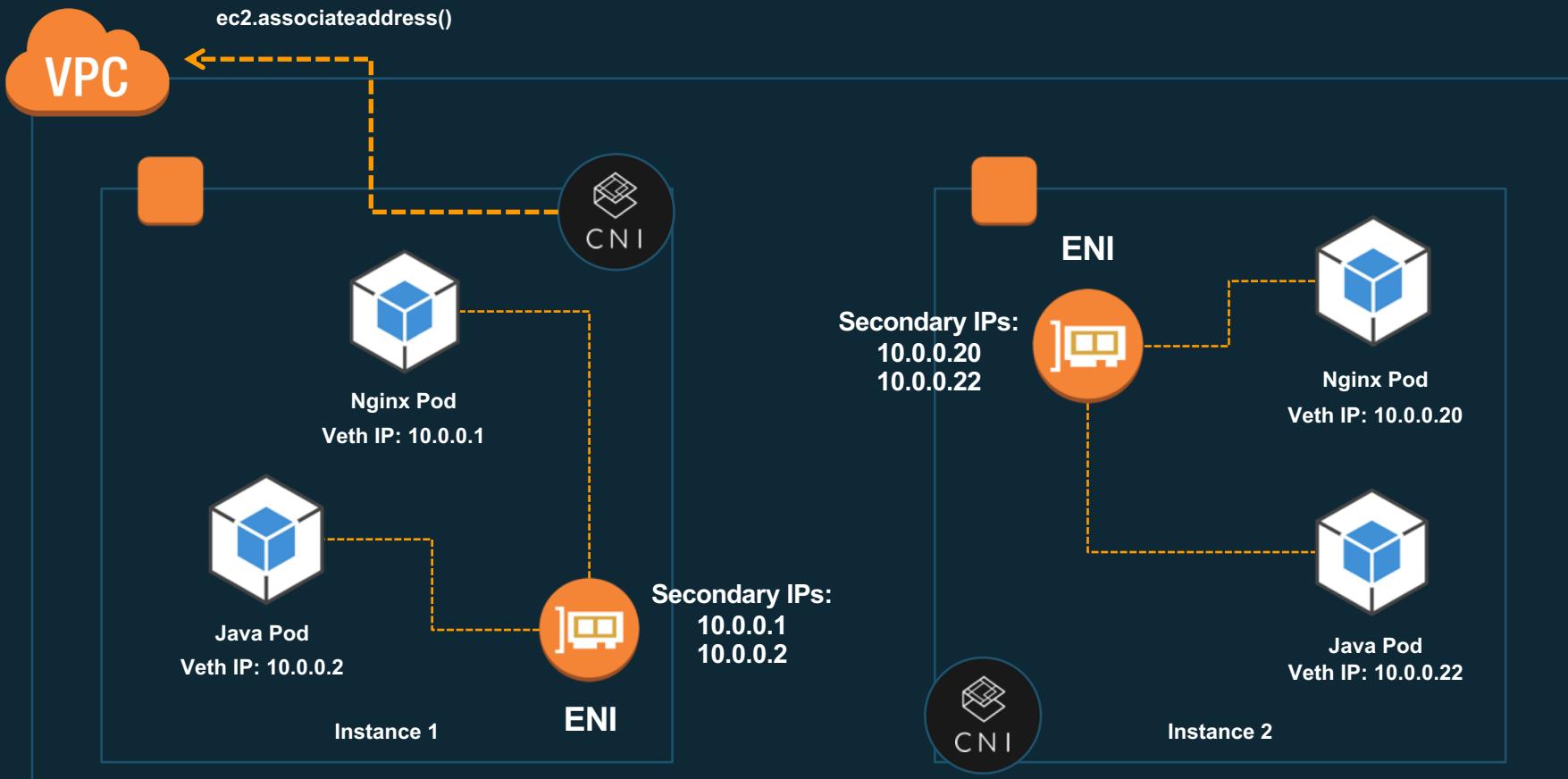
Pods have the same VPC
address inside the pod
as on the VPC



Simple, secure networking



Open source and
on Github



VPC Subnet – 10.0.0.0/24

How do I provision EKS nodes?



Integrations



Identity and Access Management (IAM)

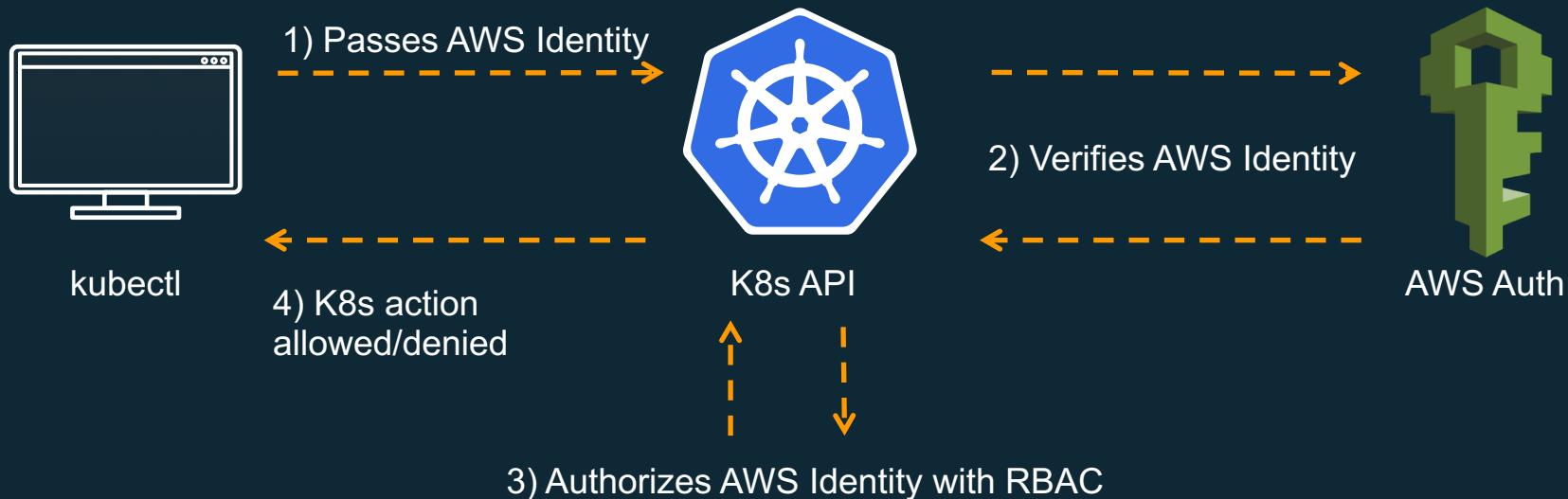


I want to use AWS accounts to operate Kubernetes

An open source approach to integrating
AWS IAM authentication with Kubernetes



IAM Authentication with kubectl



I want to give a pod permissions to an AWS service

The screenshot shows the GitHub repository page for `kube2iam`. The repository is owned by `jtblin`. Key statistics displayed include 34 issues, 5 pull requests, 0 projects, and 103 forks. The repository provides different AWS IAM roles for pods running on Kubernetes. It includes tags for `kubernetes` and `aws`. The repository has 108 commits, 1 branch, 31 releases, and 29 contributors.

jtblin / **kube2iam**

Watch 34 Star 570 Fork 103

Code Issues 24 Pull requests 5 Projects 0 Wiki Insights

kube2iam provides different AWS IAM roles for pods running on Kubernetes

kubernetes aws

108 commits 1 branch 31 releases 29 contributors

- Runs as a DaemonSet on your workers
- Creates iptables rules to redirect metadata service to kube2iam
- Add annotations to your pods to grant them AWS IAM Roles

kube2iam example

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      annotations:
        iam.amazonaws.com/role: arn:aws:iam:123567989012:role/nginx-role
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.9.1
        ports:
        - containerPort: 80
```

Container Registry: Amazon ECR



Amazon ECR

< All repositories : nlb-test

Repository ARN arn:aws:ecr:us-west-2:860498507463:repository/nlb-test

Repository URI 860498507463.dkr.ecr.us-west-2.amazonaws.com/nlb-test

[View Push Commands](#)

[Images](#) [Permissions](#) [Dry run of lifecycle rules](#) [Lifecycle policy](#)

Amazon ECR limits the number of images to 1,000 per repository. [Request a limit increase.](#)

Image sizes may appear compressed. [Learn more](#)

Last updated on April 25, 2018 12:03:50 PM (0m ago) [↻](#)

<input type="checkbox"/> Image tags	Digest	Size (MiB)	Pushed at
d7216e0 ea55d36-dirty	view all sha256:256668e603b886b352da57b71e862aafb...	3.93	2018-03-26 20:12:56 +0100
9e32413-dirty-f86650b19bad0eb5	view all sha256:5e2b36097a67fd6a26870d25ef752a0a84f...	3.93	2018-03-30 06:38:13 +0100
42287b3-dirty-a3e923300cd0c712	view all sha256:22068bcd5b43761985de879bc2dcab810...	3.93	2018-03-30 09:20:25 +0100
latest ed75a99-dirty 9e32413-dirty-0285c447...	view all sha256:5441a0c36e304986efca28c548b989e62b...	3.93	2018-03-26 21:46:14 +0100
0de924c-dirty	view all sha256:3e87f35bae9b10cdebbad3b894e19d056...	3.93	2018-03-26 20:00:44 +0100
	sha256:b151c34922869d6bdfec6900c10954ec3...	3.93	2018-03-26 19:37:37 +0100
434c927	view all sha256:081eb1eaa459061d5c41d19ed8ba212b5...	3.93	2018-03-26 20:25:06 +0100

- Simple to create
- High Availability by default
- IAM permissions
- Lifecycle rules
- Encrypted at rest
- Billed on storage

Load Balancers



Services: LoadBalancer

```
$ kubectl run nginx --image=nginx --replicas 3 --port=80  
$ kubectl expose deployment nginx --type=LoadBalancer  
$ kubectl get services -o=wide
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
nginx	LoadBalancer	100.70.217.164	a5cefe533ac1d11e7a38f0a67818e472-1987464052.eu-west-1.elb.amazonaws.com	80:31108/TCP

DNS name	Availability Zones	Type	Port Configuration	
a5cefe533ac1d11e7a38f0a67818e472-1987464052.eu-west-1.elb.amazonaws.com	eu-west-1b, eu-west-1c, eu-west-1a	classic	80 (TCP) forwarding to 31108 (TCP)	
Instance ID	Name	Availability Zone	Status	Actions
i-0478980a1a86faa09	micro.k8s.demote.cloud	eu-west-1b	InService ⓘ	Remove from Load Balancer
i-0885393f80f3db7de	micro.k8s.demote.cloud	eu-west-1a	InService ⓘ	Remove from Load Balancer
i-0d701a00358fb084f	micro.k8s.demote.cloud	eu-west-1c	InService ⓘ	Remove from Load Balancer
i-0a3b00eeabdf3b0ce	micro.k8s.demote.cloud	eu-west-1c	InService ⓘ	Remove from Load Balancer
i-08617f4b745d3bb74	micro.k8s.demote.cloud	eu-west-1b	InService ⓘ	Remove from Load Balancer
i-077d170e688971c98	micro.k8s.demote.cloud	eu-west-1a	InService ⓘ	Remove from Load Balancer

Configure your load balancers via annotations

```
service.beta.kubernetes.io/ aws-load-balancer-type  
service.beta.kubernetes.io/ aws-load-balancer-internal  
service.beta.kubernetes.io/ aws-load-balancer-proxy-protocol  
service.beta.kubernetes.io/ aws-load-balancer-access-log-emit-interval  
service.beta.kubernetes.io/ aws-load-balancer-access-log-enabled  
service.beta.kubernetes.io/ aws-load-balancer-access-log-s3-bucket-name  
service.beta.kubernetes.io/ aws-load-balancer-access-log-s3-bucket-prefix  
service.beta.kubernetes.io/ aws-load-balancer-connection-draining-enabled  
service.beta.kubernetes.io/ aws-load-balancer-connection-draining-timeout  
service.beta.kubernetes.io/ aws-load-balancer-connection-idle-timeout  
service.beta.kubernetes.io/ aws-load-balancer-cross-zone-load-balancing-enabled  
service.beta.kubernetes.io/ aws-load-balancer-extra-security-groups  
service.beta.kubernetes.io/ aws-load-balancer-ssl-cert  
service.beta.kubernetes.io/ aws-load-balancer-ssl-ports  
service.beta.kubernetes.io/ aws-load-balancer-ssl-negotiation-policy  
service.beta.kubernetes.io/ aws-load-balancer-backend-protocol  
service.beta.kubernetes.io/ aws-load-balancer-additional-resource-tags  
service.beta.kubernetes.io/ aws-load-balancer-healthcheck-healthy-threshold  
service.beta.kubernetes.io/ aws-load-balancer-healthcheck-unhealthy-threshold  
service.beta.kubernetes.io/ aws-load-balancer-healthcheck-timeout  
service.beta.kubernetes.io/ aws-load-balancer-healthcheck-interval
```

- Draining
- Logging
- SSL Certs
- Tagging
- Security groups
- Health checks

Network Load Balancer (layer 4)

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: "nlb"
    service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags: 'Name=nginx'
spec:
  type: LoadBalancer
  externalTrafficPolicy: Local
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
```

Application Load Balancer (layer 7)

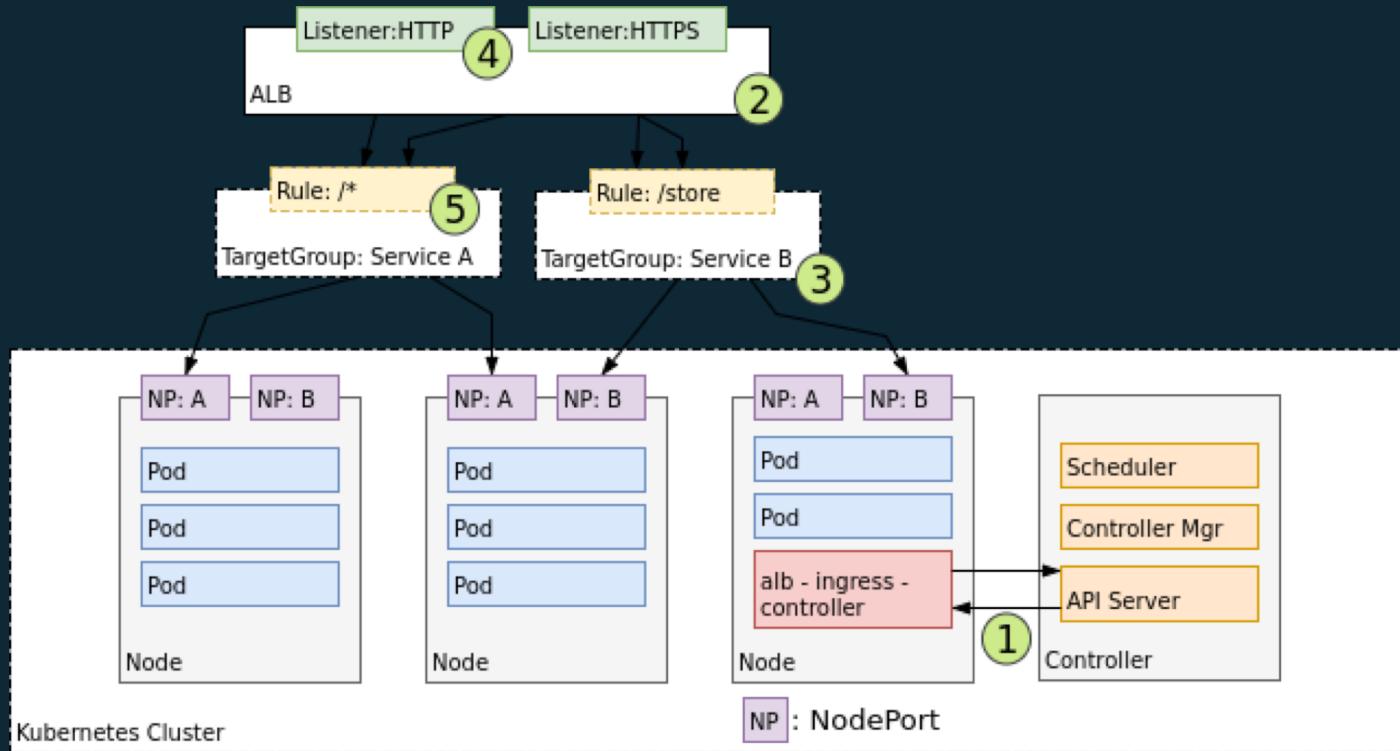


CoreOS ALB Ingress Controller: Supported by AWS

Exposes ALB functionality to Kubernetes via Ingress Resources

Layer 7 load balancing, supports content-based routing by host or path

Load Balancing



DNS



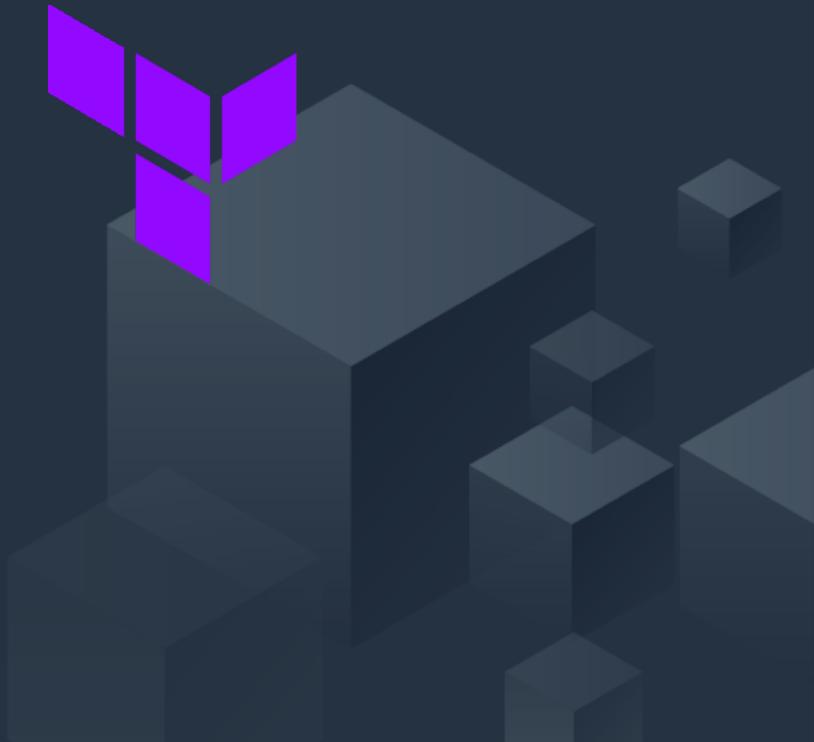
Automatic Route53 DNS creation for services

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    external-dns.alpha.kubernetes.io/hostname: nginx.demotehe.cloud.
spec:
  type: LoadBalancer
  ports:
  - port: 80
    name: http
    targetPort: 80
  selector:
    app: nginx
```

...works with ingress too

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: nginx
  annotations:
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
  - host: nginx.demotehe.cloud
    http:
      paths:
      - backend:
          serviceName: nginx
          servicePort: 80
```

Orchestration



Deploying AWS resources with K8s (operator)

<https://github.com/linki/cloudformation-operator>

```
apiVersion: cloudformation.linki.space/v1alpha1
kind: Stack
metadata:
  name: my-bucket
spec:
  template: |
    ...
  AWSTemplateFormatVersion: '2010-09-09'

Resources:
  S3Bucket:
    Type::AWS::S3::Bucket
    Properties:
      BucketName: my-bucket
```

Deploy AWS resources right from your K8s YAML files.

User's don't need AWS permissions, the IAM Role for the host(s) running the operator do.

Demo

Cluster auto scaler

Recap



Recap

- EKS runs the control plane for you (just bring nodes)
- EKS is upstream open source Kubernetes
- All integrations are open source
- The master nodes are HA (across 3 AZ's)

One more thing





Roadmap:
Can I use Fargate with EKS?



Thank you!

