

CON 361

Deep Dive into Amazon EKS

Brandon Chavis - Sr. Product Manager
Amazon EKS

Eswar Bala – Engineering Manager
Amazon EKS

aws
re:Invent

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.



In this talk, we cover the configuration and networking details needed to run a production-ready Kubernetes cluster with **Amazon Elastic Container Service for Kubernetes (Amazon EKS)**. We walk through the new features and updates for Amazon EKS in 2018.

Agenda

Customer use cases

What's new in Amazon Elastic Container Service for Kubernetes (Amazon EKS) ?

Amazon EKS Control Plane

Amazon EKS Data Plane

Amazon EKS networking and load balancing

aws
re:Invent

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.



The EKS Control Plane deals with everything that AWS runs for you while the EKS Data Plane deals with everything that you run in your account.

How are customers using Amazon EKS?

Microservices

PaaS

Enterprise app migrations

Machine learning

ML with TF based workloads using EKS

Which customers are using Amazon EKS?



"We built the next generation of our PaaS using EKS for large enterprise workloads. We manage thousands of applications and have hundreds of DevOps teams."

Which customers are using Amazon EKS?



"Snapchat serves millions of people around the world every day, and we're thrilled to now leverage Amazon EKS as a core compute service that can meet our needs now, as well as upcoming plans to host several critical workloads in the coming months."

Which customers are using Amazon EKS?



Buy.
Earn.
Redeem.

"Kubernetes is fast becoming the preferred solution for container orchestration. Its biggest downside is that it is not simple to set up and operate. EKS gives us all the benefits of Kubernetes, but takes care of managing the hard stuff. We can dedicate less resources to deployment and operations as result."

Which customers are using Amazon EKS?



"The performance from Amazon EKS makes it feasible to effectively manage large-scale databases delivering over a million queries per second. EKS also helps with our cluster management and scalability challenges."

What's new?

April: Amazon EKS achieved K8s conformance

June: Amazon EKS is HIPAA-eligible

July: Amazon EKS AMI build scripts available in GitHub

August: New EKS-optimized AMI and updated AWS CloudFormation template for provisioning worker nodes

August: Amazon EKS supports GPU-enabled Amazon Elastic Compute Cloud (Amazon EC2) instances

August: Amazon EKS platform version 2 launched

August: Amazon EKS supports HPA with custom metrics

September: Amazon EKS launches in Dublin, Ireland

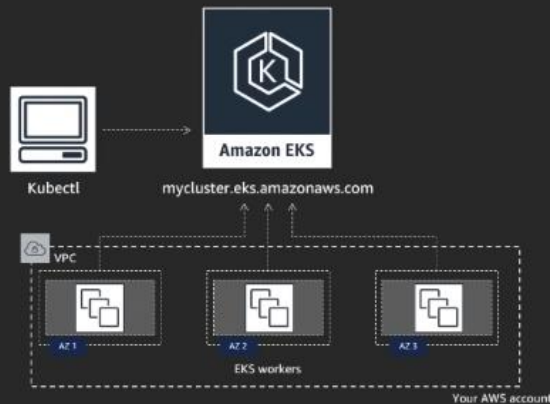
September: Amazon EKS simplifies cluster setup with `update-kubeconfig` CLI command

October: Amazon EKS adds support for Dynamic Admission Controllers (Istio)

November: Amazon EKS launches in Ohio

Amazon EKS Control Plane

Amazon EKS Architecture



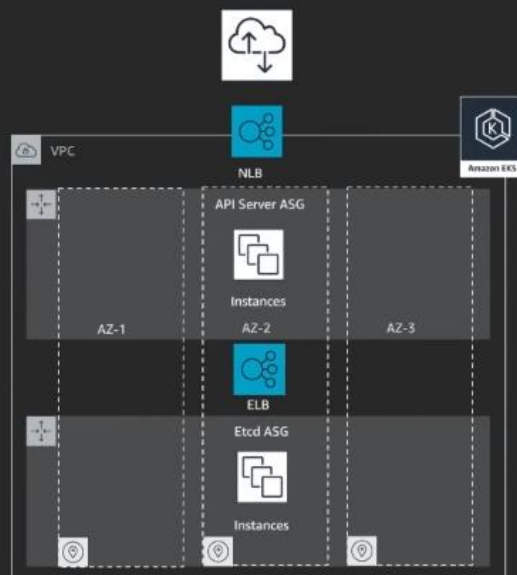
We handle all the control plane functions like etcd functions, losing nodes, connectivity issues between AZs, etc and provide you with a cluster endpoint dedicated to your cluster via Route53. You need to focus on your worker nodes that run in your VPCs. If you use Kubectl, you will need to point it to the cluster endpoint provided to you.

Kubernetes control plane

Highly available and single tenant infrastructure

All "native AWS" components

Fronted by an NLB



This is the infrastructure of the K8s control plane that AWS runs for you.

Creating a cluster: Planning

Create cluster

Cluster configuration

Cluster name
Enter a unique name for your Amazon EKS cluster.

Kubernetes version
Select the Kubernetes version to install.

1.10

Role name
Select the IAM Role to allow Amazon EKS and the Kubernetes control plane to manage AWS resources on your behalf.

Networking

VPC
Select a VPC to use for your EKS Cluster resources.

Subnets
Choose the subnets in your VPC where your worker nodes will run.

Find subnet

Subnet	Name	Availability Zone	Subnet IPv4 CIDR
--------	------	-------------------	------------------

Security groups
Choose the security groups to apply to the EKS-managed Elastic Network Interfaces that are created in your worker node subnets.

Find selection

Group	Name	Description
-------	------	-------------

Creating a cluster: Amazon EKS role

Provide Amazon EKS a role with the correct policies attached

Create cluster

Cluster configuration

Cluster name
Enter a unique name for your Amazon EKS cluster.

Kubernetes version
Select the Kubernetes version to install.

1.10

Role name
Select the IAM Role to allow Amazon EKS and the Kubernetes control plane to manage AWS resources on your behalf.

Filter policies: Q EKS

Policy name	Type	Used as
AmazonEKS_CNI_Policy	AWS managed	Permissions policy (2)
AmazonEKSClusterPolicy	AWS managed	Permissions policy (1)
AmazonEKSServicePolicy	AWS managed	Permissions policy (1)
AmazonEKSWorkerNodePolicy	AWS managed	Permissions policy (2)

Permissions | Trust relationships | Access Advisor | Session sessions

Permissions policies (2 policies applied)

Attach policies

Policy name	Policy type
AmazonEKSClusterPolicy	AWS managed policy
AmazonEKSWorkerNodePolicy	AWS managed policy

Permissions boundary (not set)

You need to provide a role so that AWS can manage resources on your behalf, the role will have the needed policies to allow AWS manage the resources for you. In IAM, there are 4 different policies available and 2 of the policies are applicable to the EKS control plane. You need to take these 2 policies **AmazonEKSClusterPolicy** and **AmazonEKSServicePolicy** and associate them with a role, then give EKS the role so that it can create the control plane for your cluster. The **AmazonEKSServicePolicy** is going to be used for managing network infrastructure in your account for things like the network interfaces that will be created. The **AmazonEKSClusterPolicy** contains all the permissions that K8s needs to do things in your account, like creating LBs, EBS volumes, modifying tags, etc.

Creating a cluster: Amazon Virtual Private Cloud (Amazon VPC)

Networking

VPC [?](#)
Select a VPC to use for your EKS Cluster resources.

Subnets [?](#)
Choose the subnets in your VPC where your worker nodes will run.

Find subnet

Subnet	Name	Availability Zone	Subnet IPv4 CIDR
--------	------	-------------------	------------------

Security groups [?](#)
Choose the security groups to apply to the EKS-managed Elastic Network Interfaces that are created in your worker node subnets.

Find selection

Group	Name	Description
-------	------	-------------

Provide all subnets that will host Kubernetes resources: Load balancers and worker nodes

Subnets can be public, private or both

Amazon EKS will tag the subnets with `kubernetes.io/cluster/<cluster-name> = shared`

Subnets that will host internal load balancers need the tag `kubernetes.io/role/internal-elb = 1`

re:Invent

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.



There are also VPC requirements, this is for the VPC that your data plane is going to be running inside and it includes the LBs and worker nodes. You need to give EKS all the private and/or public subnets that you are going to use within the VPC and then you might then decide to run your worker nodes in the private subnets and run the ELBs in the public subnets. This allows k8s to tag those subnets for you as above.

Creating a cluster: Amazon VPC

Networking

VPC [?](#)
Select a VPC to use for your EKS Cluster resources.

Subnets [?](#)
Choose the subnets in your VPC where your worker nodes will run.

Find subnet

Subnet	Name	Availability Zone	Subnet IPv4 CIDR
--------	------	-------------------	------------------

Security groups [?](#)
Choose the security groups to apply to the EKS-managed Elastic Network Interfaces that are created in your worker node subnets.

Find selection

Group	Name	Description
-------	------	-------------

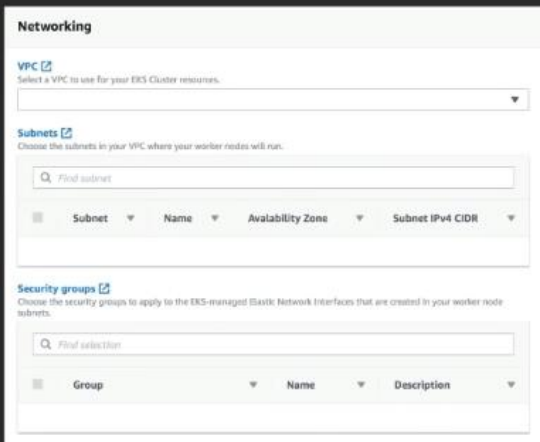
Plan ahead with subnet sizes! Pods each consume an Amazon VPC IP address.

/24 subnet = 254 IPs total. Subtract one for each node in your cluster, and the remainder is what you have for pods (*probably not enough!*)

Amazon EKS provided sample VPC template uses a /18

You have to plan your subnet sizes ahead of time and use a slightly bigger subnet size, this is because EKS allocates a lot of pods for you and it normally allocates an IP per pod created. The /24 subnet is a little too small with 254 possible pods, so we use a default /18 subnet that gives you more possible IPs with a larger CIDR block size.

Creating a cluster: Control plane Security Group



The screenshot shows the 'Networking' section of the AWS Management Console. It includes three main sections: 'VPC' with a dropdown to select a VPC, 'Subnets' with a search bar and a table with columns 'Subnet', 'Name', 'Availability Zone', and 'Subnet IPv4 CIDR', and 'Security groups' with a search bar and a table with columns 'Group', 'Name', and 'Description'.

This Security Group defines connectivity between the Kubernetes control plane and worker nodes

At minimum, Kubernetes needs 443 inbound and 10250 outbound

This security group needs permissions that align with the worker node security group

K8s needs the port 443 inbound to the control plane and the control plane needs the port 10250 outbound, the 10250 is the port that the kubelet listens on and it is what the control plane uses to send instructions to the kubelet running in your worker nodes. The 443 outbound on your worker nodes so that you can reach the HTTP endpoint that provides the k8s API. There are 2 security groups that you need to note, the control plane SG and the worker node SG, both of these SGs have to have permissions that are aligned with no exclusive denies.

Kubernetes version

1.11.3 coming soon

Amazon EKS will support up to three versions of Kubernetes at once

“Deprecation” will prevent new cluster creation on old versions

You also need to provide the k8s version you want to run in your cluster, EKS will run the last 3 K8s versions and deprecate older ones.

Amazon EKS platform version

Platform version revisions represent API server configuration changes or Kubernetes patches

Platform versions increment within a Kubernetes version only

K8s 1.10

eks.1

eks.2

eks.3

K8s 1.11

eks.1

eks.2

K8s 1.12

eks.1

API server configuration

Kubernetes version	Kubernetes patch version	Amazon EKS platform version	Enabled admission controllers	Release notes
1.10	1.10.3	eks.2	Initializers, NamespaceLifecycle, LimitRanger, ServiceAccount, DefaultStorageClass, ResourceQuota, DefaultTolerationSeconds, NodeRestriction, MutatingAdmissionWebhook, ValidatingAdmissionWebhook	<ul style="list-style-type: none">*Added support for Kubernetes aggregation layer.*Added support for Kubernetes Horizontal Pod Autoscaler (HPA).*Kubernetes Metrics Server 0.3.0 or greater is compatible with Amazon EKS platform version eks.2.

You can look at the documentation to see what changes are available in the new EKS K8s version changes

EKS Kubernetes version updates – coming soon

New UpdateClusterVersion API – supports in-place updates of Kubernetes version

Introduces an *update* EKS API object

ListUpdates and DescribeUpdate APIs to provide visibility into the status of a given update

There are some APIs to help you listed above.

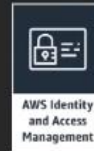
AWS Identity and Access Management (IAM) Authentication



Kubectl



K8s API



AWS Identity and Access Management

We have integrated IAM with the K8s RBAC, we split the responsibilities of authentication and authorization.

AWS Identity and Access Management (IAM) Authentication

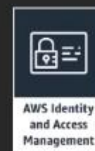


Kubectl

1) Passes AWS identity



K8s API



AWS Identity and Access Management

In Kubectl, we have an external authentication binary called the **AWS Authenticator** that allows kubectl to pass your AWS identity to the K8s API Server. This means that kubectl can look in the exact credential chain as the AWS CLI like looking at your session token, your environment variables, your secret access keys or active roles. It can package this information up and send it to the k8s API server.

AWS Identity and Access Management (IAM) Authentication



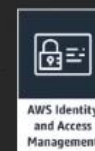
Kubectl

1) Passes AWS identity



K8s API

2) Verifies AWS identity



AWS Identity and Access Management

3) Authorizes AWS identity with RBAC

We then have a webhook on the K8s API server that verifies/authenticates the identity as a valid user and has the authorized permissions to make the specific kubectl call, it will then allow or deny the action.

PKI configuration

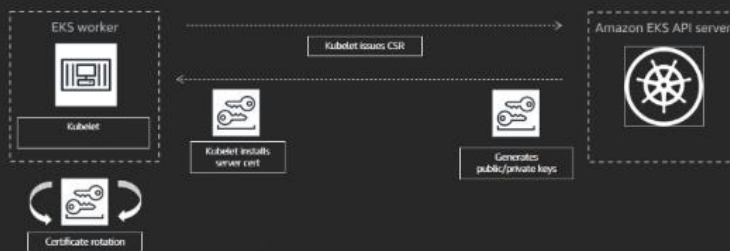
Each Amazon EKS cluster is a unique CA



Each EKS cluster is also a unique CA, and AWS manages this for you since the PKIs in K8s is complicated to set up by you. We use the built-in K8s CA system and generate a set of certificates for the control plane.

PKI configuration

Each Amazon EKS cluster is a unique CA



The kubelet can also get a certificate by issuing a CRS to the EKS API Server that will generate both public and private keys. Then the kubelet knows how to install the generated certificates. The kubelet's are also set up to do certificate rotation automatically.

Amazon EKS is ready for sensitive and regulated workloads

HIPAA-eligible

ISO 9001, 27001, 27017, 27018 – coming soon

PCI DSS – coming soon

Wrapping it all up

Easily export control plane config

```
$aws eks update-kubeconfig --name my-cluster
```

CLI Helper command to create kubeconfig file

Creates a new context for each cluster in the config

This helper command will dump all the config information you care about from the control plane like the certificate data, cluster name into a kubeconfig file that you can start to use.

Amazon EKS Data plane

This is the stuff that runs on your account.

Bring your own instances

Instance Flexibility

P2 and P3 instances for GPUs? ✓



i3.metal instances? ✓

Spot instances? ✓

A mix of all of the above? ✓

You have some things to stand up on your end but we give you an AMI and a CloudFormation (CF) template that you can use. You are not limited on the instance types that you can use and launch for your worker nodes group.

Bring your own OS

EKS AMI build scripts

<https://github.com/awslabs/amazon-eks-ami>



Source of truth for Amazon EKS Optimized AMI

Easily build your own Amazon EKS AMI

Build assets for Amazon EKS AMI for each supported Kubernetes version

AI/ML with Amazon EKS

Amazon EKS-optimized AMI with GPU support



Easily run Tensorflow on Amazon EKS

Includes NVIDIA packages to support Amazon P2 and P3 instances

Available on AWS Marketplace

Worker node setup – Bootstrapping

```
/etc/eks/bootstrap.sh [options] <cluster-name>
```

Supports arbitrary kubelet bootstrap args and defining `--max-pods` for kubelet

You have a couple of options for bootstrapping your worker nodes, AWS provides a CF template to use. You can also use the **bootstrap.sh** script in the GitHub repo. This allows you to tell the kubelet what is the cluster name that it should be joining, that instance will then join that cluster as a worker node. It also allows you to pass flags to kubelet to modify its behavior at boot time, this could be for defining max-pods, etc.

Worker node setup – Bootstrapping

Amazon-eks-nodegroup.yaml AWS Cloudformation template supports
`BootstrapArguments` parameter

Passes parameter directly to bootstrap.sh: `/etc/eks/bootstrap.sh
${ClusterName} ${BootstrapArguments}`

Result? Modify Kubelet config directly through CFN parameter: `--kubelet-
extra-args --node-labels=mykey=myvalue,nodegroup=NodeGroup1`

This is also extensible to our CF template using the parameter called **BootstrapArguments** that allow you to feed these arguments directly to the **bootstrap.sh** script. This allows you to modify the cluster configuration at scale across all your instances using the flags via CF.

Worker node setup - Authentication

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - roleARN: <ARN of instance role>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

aws
re:Invent

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.



An important part of this is the Authentication component, all of the bootstrapping of the instance that you do requires that you tell the cluster to let the particular IAM policy join the cluster as a node. This works by using a ConfigMap on the AWS cluster and we have a group for the node, you basically tell it the ARN for the node that should be allowed to join that cluster.

Updating worker nodes

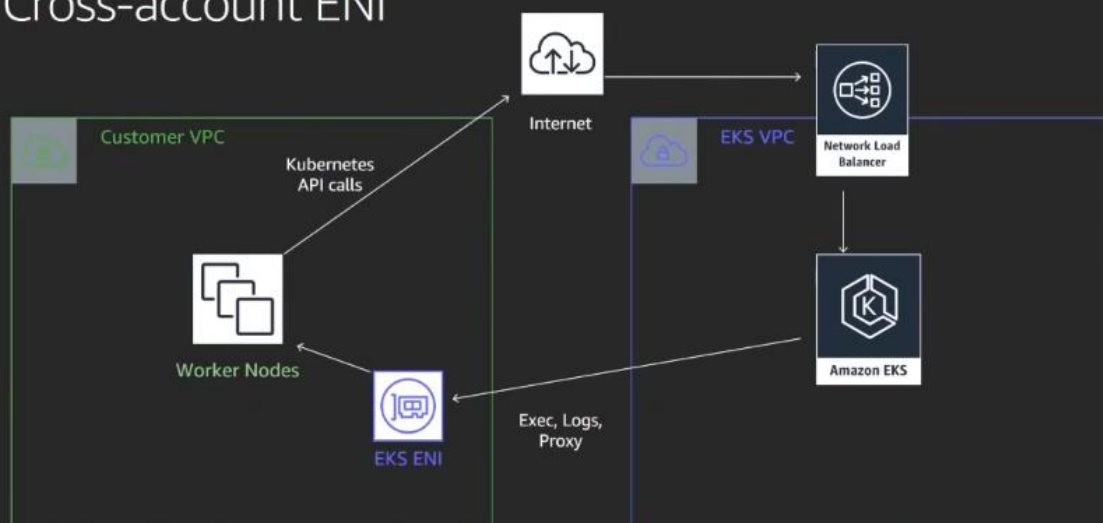
Two options:

- 1) Create new node group with latest Amazon EKS AMI; drain old nodes; terminate old CFN template
- 2) Simply update AMI in CFN template; "rolling" replacement policy terminates nodes

(Downsides: un-graceful termination of applications)

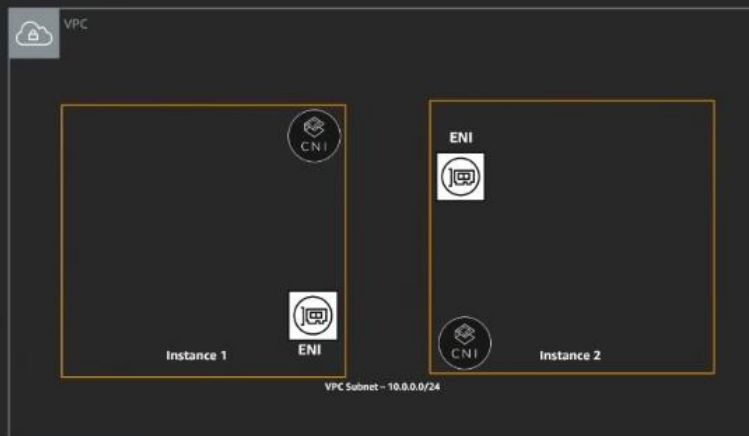
Amazon EKS networking

Cross-account ENI



Amazon VPC CNI plugin

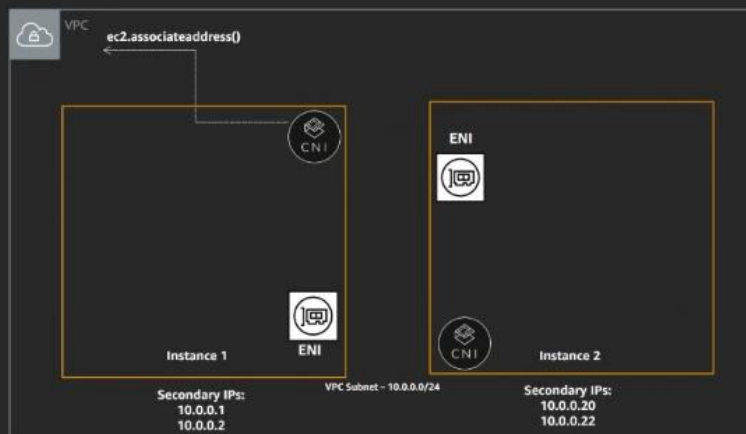
<https://github.com/aws/amazon-vpc-cni-k8s>



This is what runs in your account and VPC K8s pods. We also have and use a CNI plugin which is an OSS project for allowing your applications communicate with each other and to external resources. This means that every pod in your K8s cluster gets a native VPC IP address. Using the CNI allows you to go directly to your pods for debugging purposes, LBs can also go directly to the pod's IP address instead of going through a NodePort using an additional hop, etc.

Amazon VPC CNI plugin

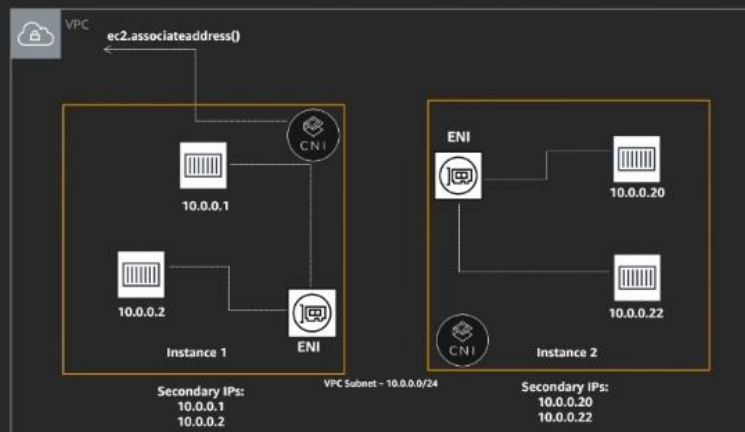
<https://github.com/aws/amazon-vpc-cni-k8s>



The CNI plugin attaches an ENI on-demand to the worker nodes where the pods run, it then uses secondary IP addresses by calling **ec2.associateaddress()** to the instances. This approach allows us to get more pod density per worker node instance. Different EC2 instance types can support varying number of ENIs per instance, bigger ENIs can support more secondary addresses. The T2 instance has 2 ENIs and 6-8 secondary IP addresses.

Amazon VPC CNI plugin

<https://github.com/aws/amazon-vpc-cni-k8s>



As pods are scheduled to your worker nodes, the CNI wires up the network interfaces for you. The secondary IP addresses are going to attach to an ENI and CNI plugin knows that when a pod is scheduled, it should take one of those IP addresses and wire it into the pod. You need to size your VPC appropriately.

CNI custom network support

1) Edit CNI `aws-node` daemonset:

```
env:  
  name: AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG  
  value: "true"
```

2) Deploy ENIconfig.yaml

Custom resource that applies custom networking configs

You can also expand the IP address ranges to include secondary ranges to be used by pods in your worker nodes.

CNI custom network support

3) Create custom resource definition:

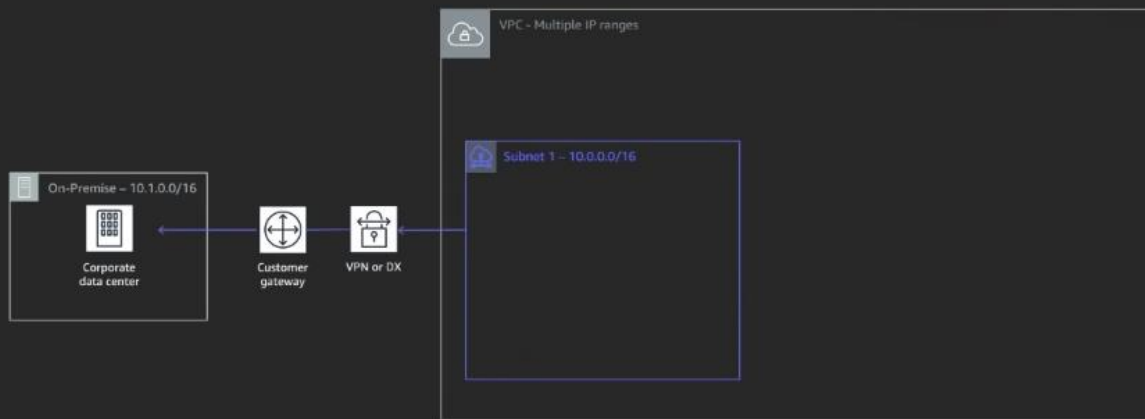
```
apiVersion: crd.k8s.amazonaws.com/v1alpha1  
kind: ENIConfig  
metadata:  
  name: group1-pod-netconfig  
spec:  
  subnet: subnet-0123456
```

CNI custom network support

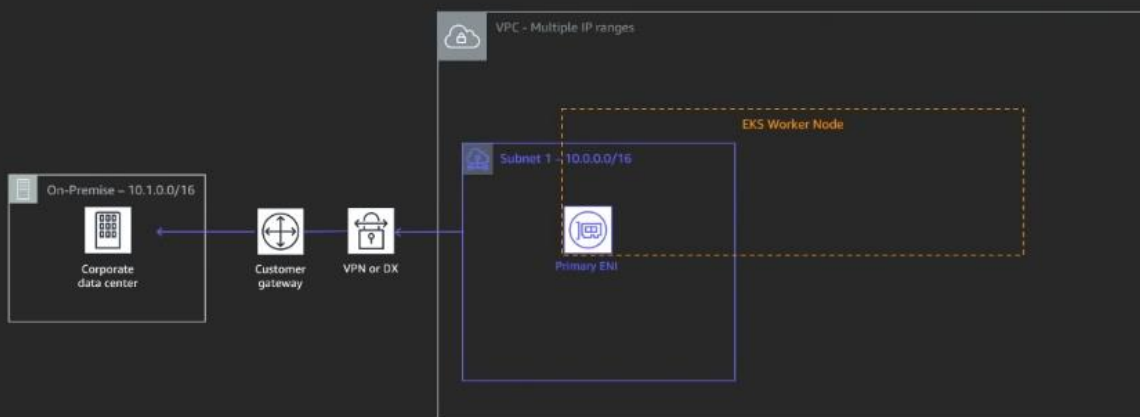
4) Annotate nodes to use custom config:

```
`kubectl annotate node <nodename>.<region>.compute.internal  
k8s.amazonaws.com/eniConfig=group1-pod-netconfig`
```

EKS Supports Advanced Networking Architectures

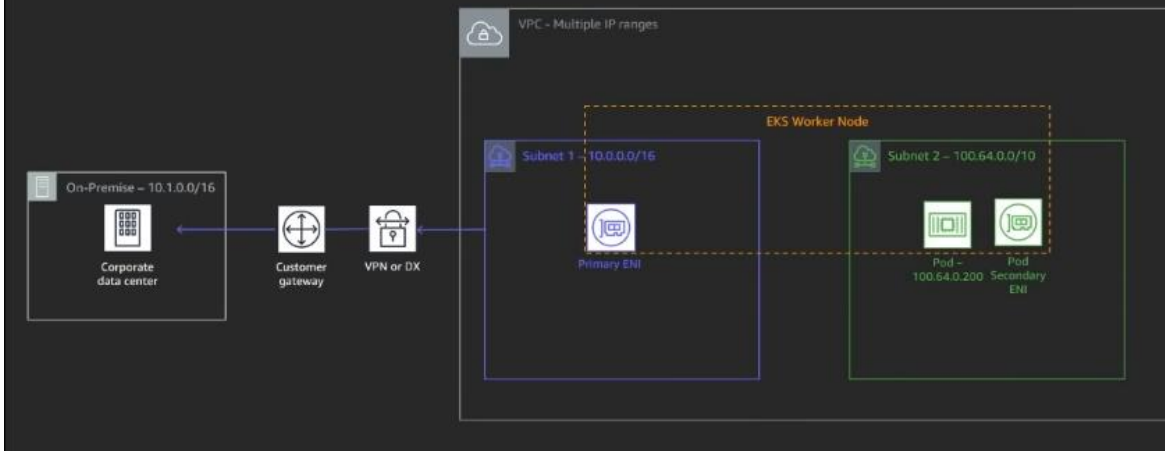


EKS Supports Advanced Networking Architectures



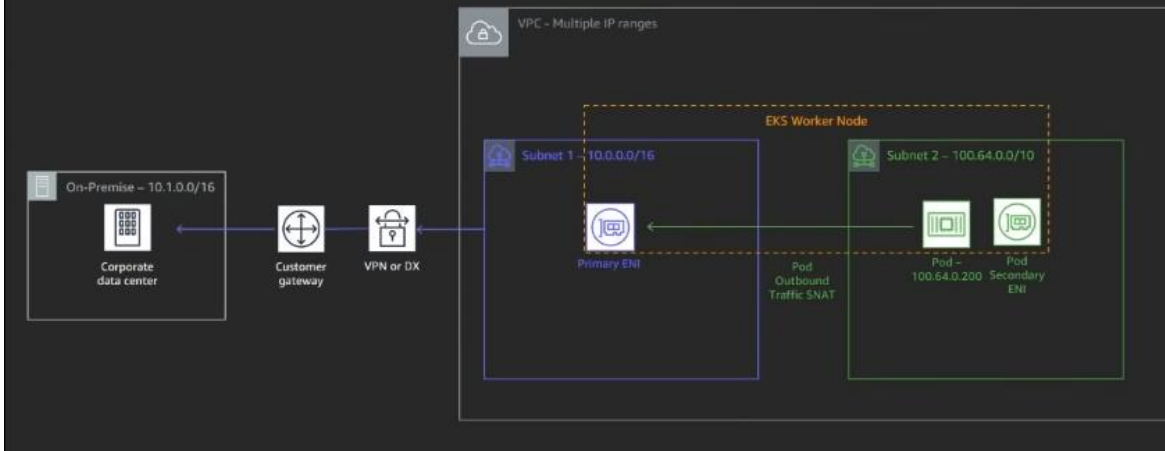
The primary ENI always remains on the primary subnet in your VPC, this ensures that pods running in your worker nodes have communication access back to your on-premises workloads.

EKS Supports Advanced Networking Architectures



When you have a secondary subnet in your VPC, the worker node will see this and add a secondary ENI to that subnet and also add a pod to it.

EKS Supports Advanced Networking Architectures



We then use source-NATting to allow pods on the secondary subnet to communicate with the on-premises workloads via the primary ENI.

Load balancing

All three AWS Elastic Load Balancing products are supported

NLB and CLB supported by Kubernetes Service
type=LoadBalancer

Internal and External Load Balancer support

Load balancing

Want to use an Internal Load Balancer? Use annotation:

```
service.beta.kubernetes.io/aws-load-balancer-internal: 0.0.0.0/0
```

Want to use an NLB? Use annotation:

```
service.beta.kubernetes.io/aws-load-balancer-type: nlb
```

You need to have a private subnet for the internal LB when you tell K8s to create the LB using the command above. Do the same for creating the NLB as above.

ALB Ingress controller

Production-ready 1.0 release

Supported by Amazon EKS team

Open source development: <https://github.com/kubernetes-sigs/aws-alb-ingress-controller>

Customers are using it in production today!



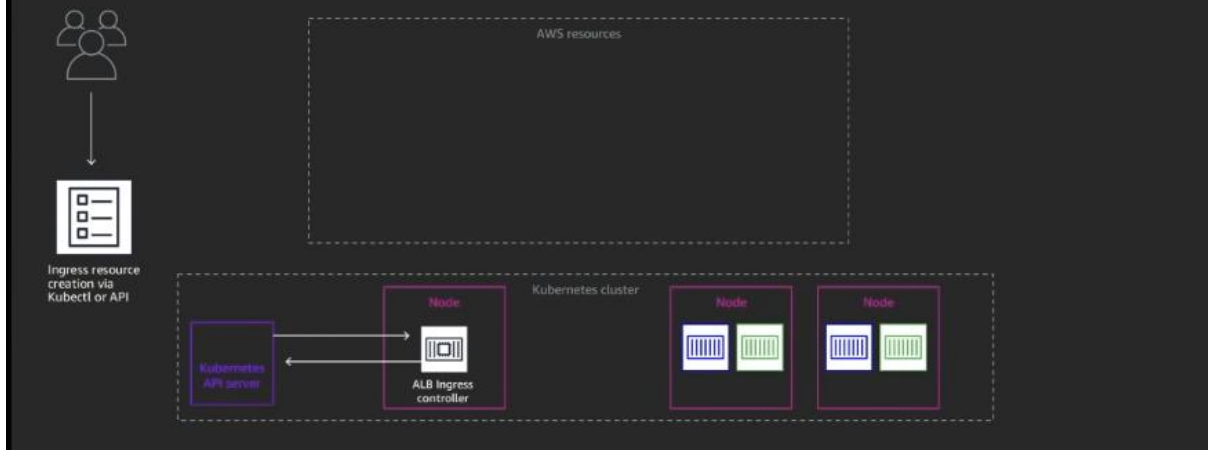
This gracefully maps the ALB to the ingress resource or API in K8s. Ingress is the layer-7 abstraction for K8s.

ALB Ingress controller

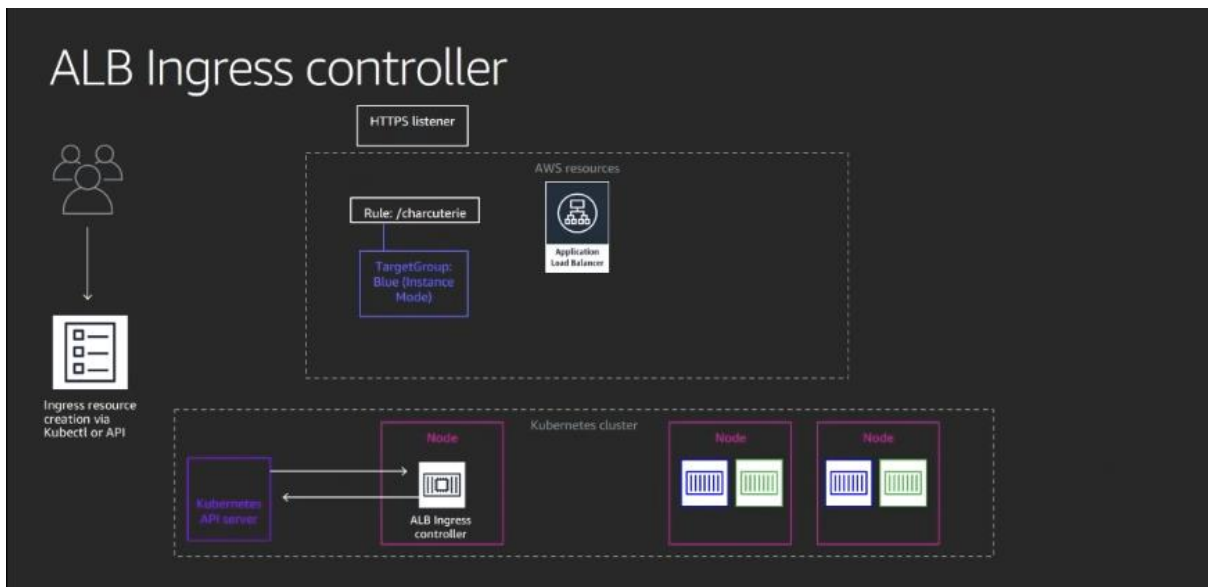


The ALB Ingress controller enables you to expose applications running inside your K8s cluster as an external HTTP or HTTPS service, it relies on the ALB to provide all the context-based routing.

ALB Ingress controller

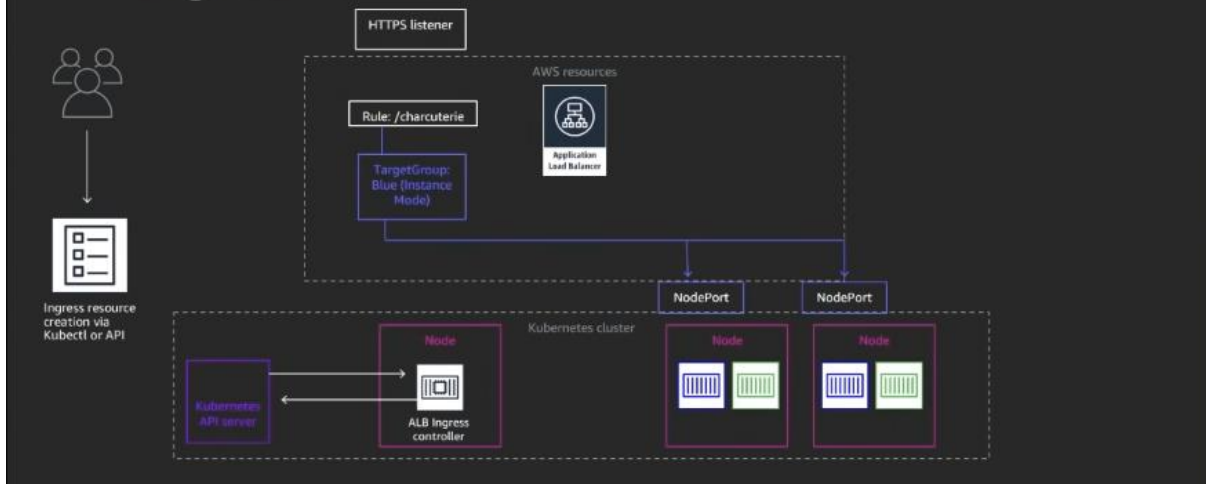


To use the ALB Ingress controller, you would first deploy it as a deployment on one of your worker nodes. This is just a simple app that watches the API Server and looks for notifications from the API Server that says 'someone has created an ingress resource'. This will include things like wanting an ALB, what routes should be configured in the ALB, what SGs



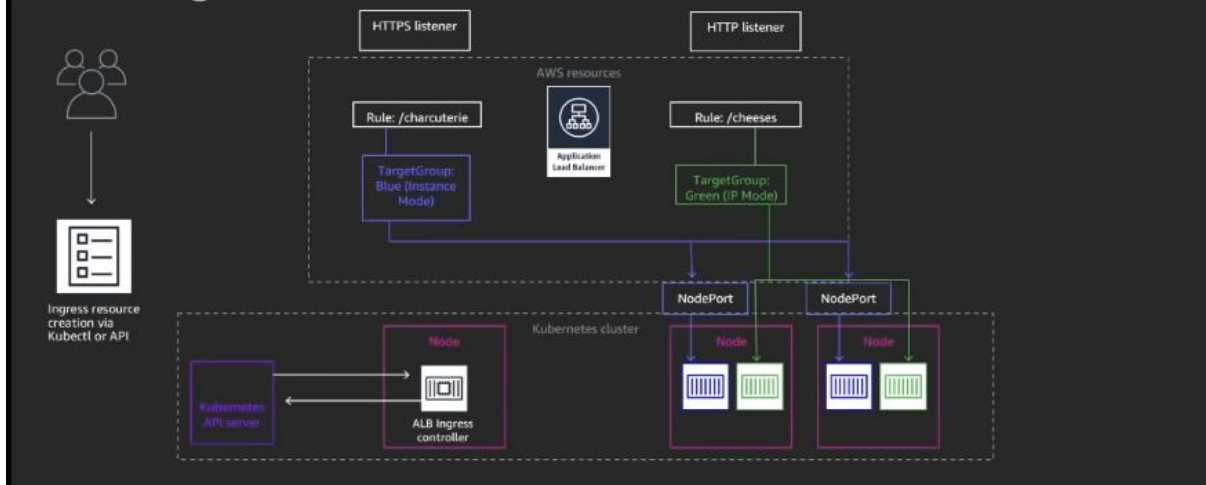
The ingress controller then starts to create the ALB for you and all other things like the HTTPS Listener and a Rule to what TargetGroup.

ALB Ingress controller



Traffic is actually routed to the NodePorts on your worker nodes that allows the traffic to get to your pods

ALB Ingress controller



You can also have another HTTP Listener, it uses IP Mode that points directly to the running pods IP addresses

Recap!

Amazon EKS GA for production workloads in June 2018 🎉

ALB Ingress controller 1.0: open source and ready for production workloads now

New features coming: Kubernetes version updates, Kubernetes 1.11.3

Compliance: HIPAA-eligible today, ISO and PCI coming soon

Global availability

Launch Regions: Virginia, Oregon

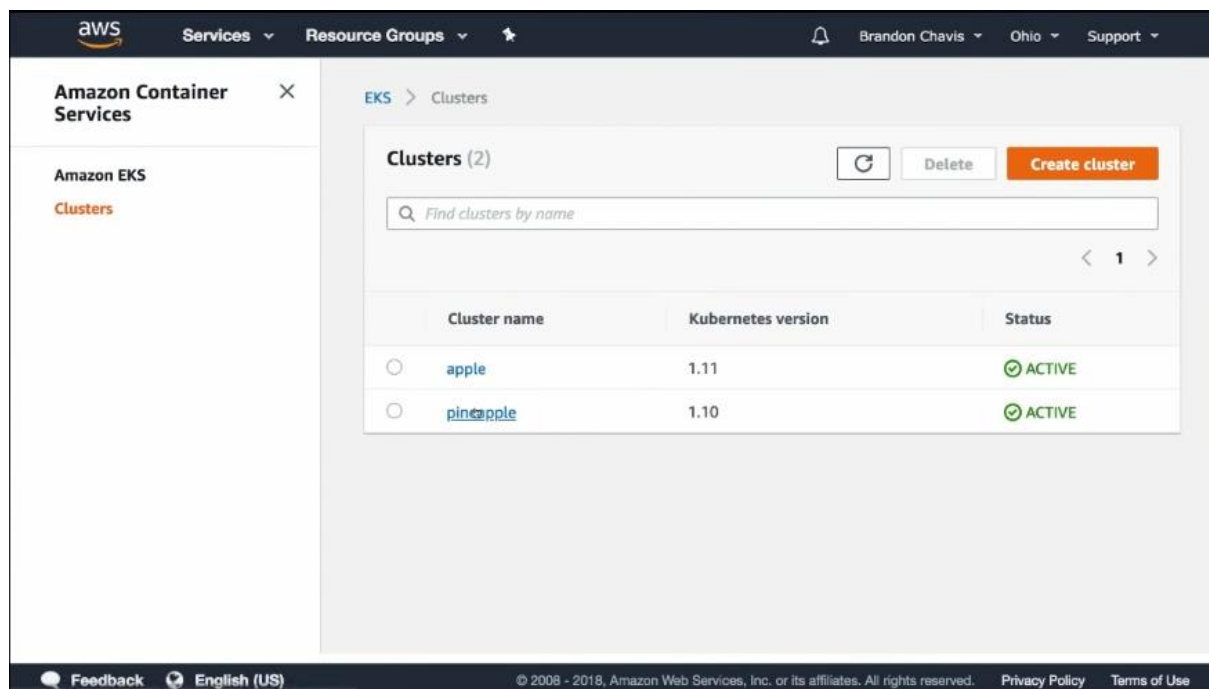
Recent Regions: Ireland, Ohio

Coming soon: Singapore, Sydney, Seoul, Tokyo, Frankfurt

Demo: EKS updates

```
[ec2-user@ip-172-31-32-49 ~]$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
bb                                  1/1     Running   1           14h
my-nginx-77f56b88c8-2wl5g          1/1     Running   0           14h
my-nginx-77f56b88c8-bf8bp          1/1     Running   0           14h
[ec2-user@ip-172-31-32-49 ~]$
```

We have a couple of pods running on a **pineapple** cluster



The screenshot shows the AWS Management Console interface for Amazon EKS. The left sidebar contains the navigation menu with 'Amazon Container Services' and 'Amazon EKS' expanded, showing 'Clusters'. The main content area is titled 'EKS > Clusters' and shows a list of clusters. At the top, there are buttons for 'Refresh', 'Delete', and 'Create cluster'. Below is a search bar 'Find clusters by name'. The cluster list has columns for 'Cluster name', 'Kubernetes version', and 'Status'. Two clusters are listed: 'apple' (Kubernetes version 1.11, Status ACTIVE) and 'pineapple' (Kubernetes version 1.10, Status ACTIVE). The footer contains 'Feedback', 'English (US)', and copyright information.

Cluster name	Kubernetes version	Status
apple	1.11	ACTIVE
pineapple	1.10	ACTIVE

aws

Services

Resource Groups

Brandon Chavis

Ohio

Support

Amazon Container Services

Amazon EKS

Clusters

EKS > Clusters > pineapple

pineapple

Update cluster version

Delete

General configuration

Kubernetes version

Platform version

Status

1.10

eks.2

ACTIVE

API server endpoint

Certificate authority

https://f58c0f9284e42a11c4f84706f9710d98.yl4.us-east-2.eks.amazonaws.com

LS0tLS1CRUdJTiBDRVJUSUZJQ0FUR50tLS0tCk1JSUN5RENDQWJDZ0F3SUJBZ0lCQURBTklna3Foa2lHOXcwQkFRc0ZBREFTVjNdOVRWURWU

Cluster ARN

Role ARN

arn:aws:eks:us-east-2:572384241454:cluster/pineapple

arn:aws:iam::572384241454:role/pineapple-SERVICE-ROLE-AWSServiceRoleForAmazonEKS

Feedback

English (US)

© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

aws

Services

Resource Groups

Brandon Chavis

Ohio

Support

Amazon Container Services

Amazon EKS

Clusters

EKS > Clusters > pineapple

API server endpoint

Certificate authority

https://f58c0f9284e42a11c4f84706f9710d98.yl4.us-east-2.eks.amazonaws.com

LS0tLS1CRUdJTiBDRVJUSUZJQ0FUR50tLS0tCk1JSUN5RENDQWJDZ0F3SUJBZ0lCQURBTklna3Foa2lHOXcwQkFRc0ZBREFTVjNdOVRWURWU

Cluster ARN

Role ARN

arn:aws:eks:us-east-2:572384241454:cluster/pineapple

arn:aws:iam::572384241454:role/pineapple-SERVICE-ROLE-AWSServiceRoleForAmazonEKS-4JZLPR033W89

Networking

VPC

Subnets

Security groups

vpc-047316fefaf92c131

subnet-0493e576bfec605c4

sg-00f05242306d7e57b

subnet-0393ed31d3d8c6e7d

subnet-021975bf9c6876262

Feedback

English (US)

© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

aws

Services

Resource Groups

Brandon Chavis

Ohio

Support

EKS > Clusters > pineapple > Update cluster version

Update cluster version

Cluster configuration

Cluster name

Enter a unique name for your Amazon EKS cluster.

pineapple

Kubernetes version

Select the Kubernetes version to install.

1.11

Cancel

Update

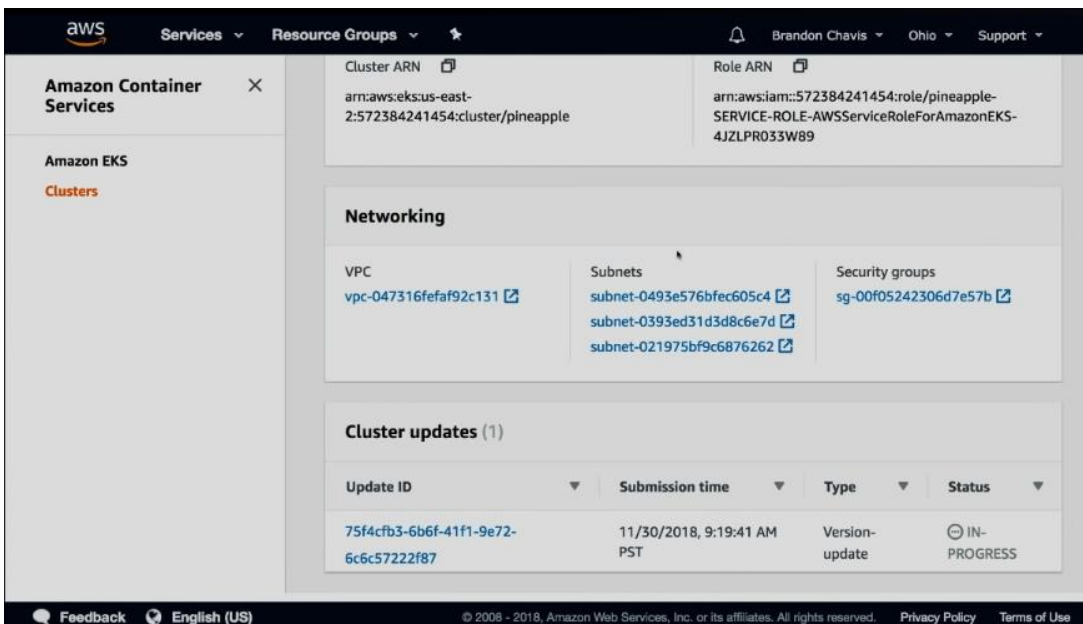
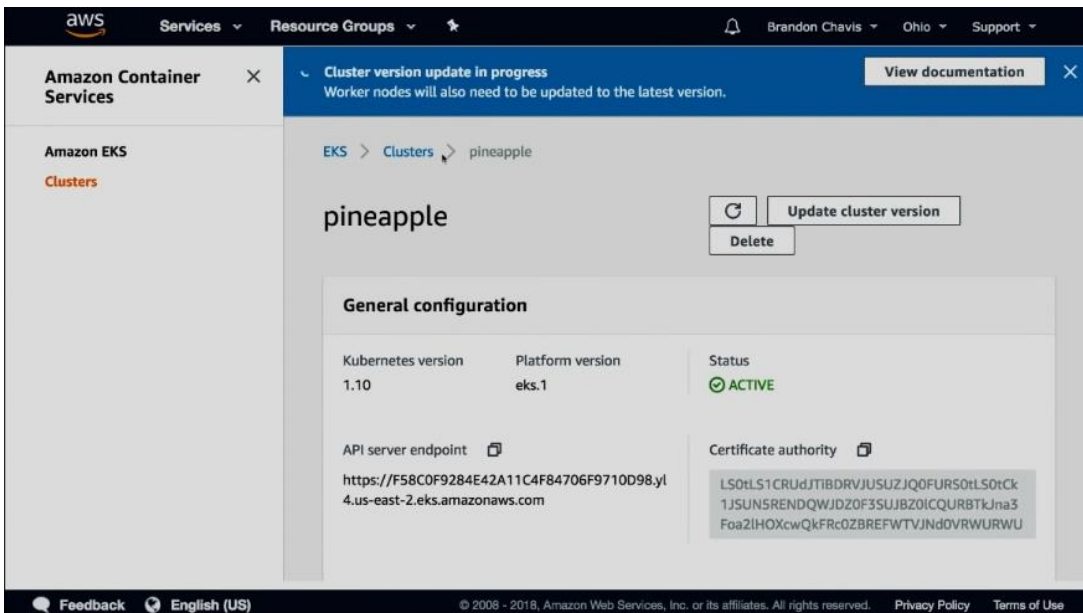
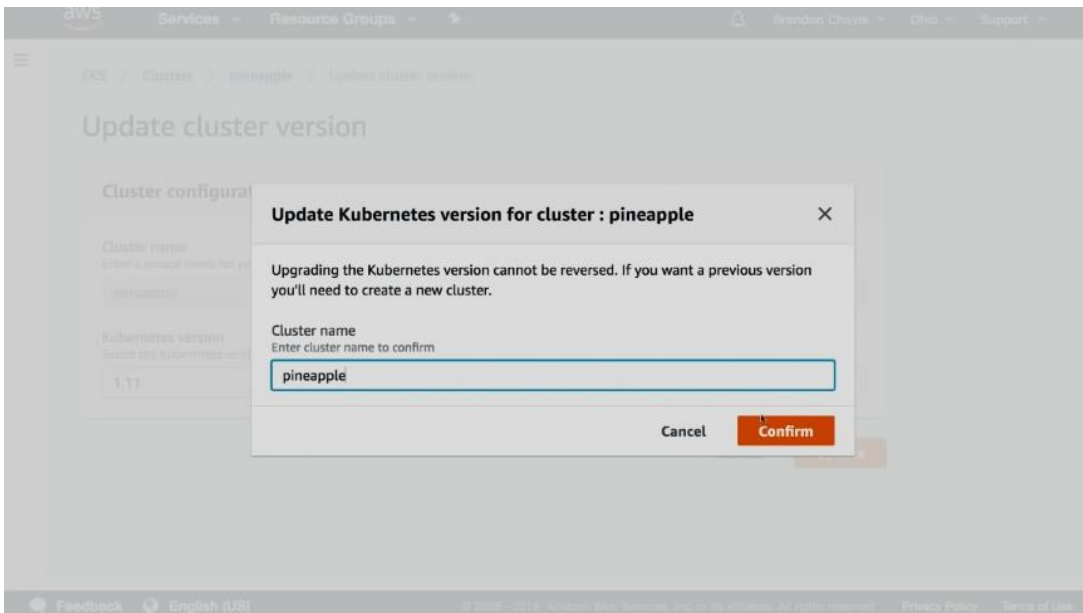
Feedback

English (US)

© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use



aws Services Resource Groups

Amazon Container Services

Amazon EKS Clusters

Update ID : 75f4cfb3-6b6f-41f1-9e72-6c6c57222f87

General configuration

Update ID 75f4cfb3-6b6f-41f1-9e72-6c6c57222f87	Kubernetes Version 1.11
Status IN-PROGRESS	Platform Version eks.1
Type Version-update	

Errors (0)

Feedback English (US) © 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

```
[ec2-user@ip-172-31-32-49 ~]$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
bb            1/1     Running   1           14h
my-nginx-77f56b88c8-2w15g  1/1     Running   0           14h
my-nginx-77f56b88c8-bf8bp  1/1     Running   0           14h
[ec2-user@ip-172-31-32-49 ~]$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
bb            1/1     Running   1           14h
my-nginx-77f56b88c8-2w15g  1/1     Running   0           14h
my-nginx-77f56b88c8-bf8bp  1/1     Running   0           14h
[ec2-user@ip-172-31-32-49 ~]$ kubectl exec bb -- /bin/wget my-nginx -O -
```

```
my-nginx-77f56b88c8-bf8bp  1/1     Running   0           14h
[ec2-user@ip-172-31-32-49 ~]$ kubectl exec bb -- /bin/wget my-nginx -O -
Connecting to my-nginx (10.100.17.194:80)
- 100% |*****| 612 0:00:00 ETA

<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[ec2-user@ip-172-31-32-49 ~]$
```