Amazon EKS

User Guide

Amazon EKS: User Guide

Table of Contents

What Is Amazon EKS?

Amazon Elastic Container Service for Kubernetes (Amazon EKS) is a managed service that makes it easy for you to run Kubernetes on AWS without needing to install and operate your own Kubernetes clusters. Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications. Operating Kubernetes for production applications presents a number of challenges. You must manage the scaling and availability of your Kubernetes masters and persistence layer by ensuring that you have chosen appropriate instance types, running them across multiple Availability Zones, monitoring their health, and replacing unhealthy nodes. You must patch and upgrade your masters and worker nodes to ensure that you are running the latest version of Kubernetes. This all requires expertise and a lot of manual work. With Amazon EKS, upgrades and high availability are managed for you by AWS. Amazon EKS runs three Kubernetes masters across three Availability Zones to ensure high availability. Amazon EKS automatically detects and replaces unhealthy masters, and it provides automated version upgrades and patching for the masters.

Amazon EKS is also integrated with many AWS services to provide scalability and security for your applications, including the following:

- Elastic Load Balancing for load distribution
- IAM for authentication
- Amazon VPC for isolation
- AWS CloudTrail for logging

Amazon EKS runs up to date versions of the open-source Kubernetes software, so you can use all the existing plugins and tooling from the Kubernetes community. Applications running on Amazon EKS are fully compatible with applications running on any standard Kubernetes environment, whether running in on-premises data centers or public clouds. This means that you can easily migrate any standard Kubernetes application to Amazon EKS without any code modification required.

For more information about creating your required resources and your first Amazon EKS cluster, see Getting Started with Amazon EKS (p. 3).

Preview Considerations

Amazon EKS is currently in preview and should not be considered to be production-ready at this time. Please do not run production workloads on Amazon EKS while the service is in preview. As a preview service, you should expect continuous changes and rapid iteration.

The Amazon EKS preview resources, such as the provided AWS CloudFormation templates,custom **kubectl** client, and plugins are available in Amazon S3. You can view the resources with the following AWS CLI command:

```
aws s3 ls --recursive s3://amazon-eks/2018-04-04/
```

Amazon EKS clusters require custom-built **kubectl** and **kubelet** binaries that include the Heptio Authenticator. This is the project that provides IAM authentication for your Kubernetes cluster. This requirement will be obsolete in Kubernetes version 1.10. For more information, see Download and Install the Custom **kubectl** Binary (p. 4) in *Getting Started* (p. 3).

Amazon EKS worker nodes are standard Amazon EC2 instances, and you are billed for them based on normal EC2 On-Demand prices. For more information, see Amazon EC2 Pricing.

The Amazon EKS worker node AMI is based on Amazon Linux 2, with a custom `kubelet` and `kube-proxy` built in. You can track security or privacy events for Amazon Linux 2 at the Amazon Linux Security Center or subscribe to the associated RSS feed. Security and privacy events include an overview of the issue, what packages are affected, and how to update your instances to correct the issue.

For support, work with your AWS technical account managers to file feedback and open issues. Try to work across accounts in your company to collate feedback before submission. Support for ongoing issues is provided on a best-effort basis.

Amazon EKS is not included in the public AWS CLI client. You must download and install a custom JSON model (p. 5) for the AWS CLI to make calls to the Amazon EKS APIs. Your system's Python version must be Python 3, or Python 2.7.9 or greater; otherwise, you will receive `hostname doesn't match` errors with AWS CLI calls to Amazon EKS. For more information, see What are "hostname doesn't match" errors? in the Python Requests FAQ.

The security group for the worker nodes and the security group for the control plane communication to the worker nodes have been setup to prevent communication to privileged ports in the worker nodes. If your applications require added inbound or outbound access from the control plane or worker nodes, you must add these rules to the security groups associated with your cluster. For more information, see Security Groups for Your VPC in the *Amazon VPC User Guide*.

Note
To allow proxy functionality on privileged ports or to run the CNCF conformance tests yourself, you must edit the security groups for your control plane and the worker nodes. The security group on the worker nodes side need to allow inbound access for ports 0 - 66535 from the control plane, and the control plane side needs to allow outbound access to the worker nodes on ports 0 - 65535.

You must activate AWS STS in the `us-west-2` region as a prerequisite. During the preview, Amazon EKS authentication operations communicate to the `us-west-2` AWS STS endpoint to allow access into the cluster. For more information, see Activating and Deactivating AWS STS in an AWS Region in the *IAM User Guide*.

Getting Started with Amazon EKS

This getting started guide helps you to create all of the required resources to use Amazon EKS during the preview.

Amazon EKS Preview Prerequisites

Before you can create an Amazon EKS cluster, you must create an IAM role that Kubernetes can assume to create AWS resources. For example, when a load balancer is created, Kubernetes assumes the role to create an Elastic Load Balancing load balancer in your account. This only needs to be done one time and can be used for multiple EKS clusters.

You must also create a VPC with certain tagging and security group requirements. Although the VPC and security groups can be used for multiple EKS clusters, we recommend that you use a separate VPC for each EKS cluster to provide better network isolation.

This section also helps you to install a custom **kubectl** binary that is configured to work with Amazon EKS.

Optionally, you can download a custom version of the AWS CLI to use with Amazon EKS.

Create your Amazon EKS Service Role

1. Open the AWS CloudFormation console at https://us-west-2.console.aws.amazon.com/cloudformation.
2. From the navigation bar, select the **US West (Oregon)** region.

   Note
   At this time, Amazon EKS is only available in the **US West (Oregon)** region.
3. Choose **Create stack**.
4. For **Choose a template**, select **Specify an Amazon S3 template URL**.
5. Paste the following URL into the text area and choose **Next**:

```
https://amazon-eks.s3-us-west-2.amazonaws.com/2018-04-04/amazon-eks-service-role.yaml
```

6. On the **Specify Details** page, fill out the parameters accordingly, and then choose **Next**.
   - **Stack name**: Choose a stack name for your AWS CloudFormation stack. For example, you can call it **eks-service-role**.
7. (Optional) On the **Options** page, tag your stack resources. Choose **Next**.
8. On the **Review** page, review your information and acknowledge that the stack might create IAM resources. Choose **Create**.
9. When your stack is created, select it in the console and choose **Outputs**.
10. Record the **RoleArn** for the role that was created. You need this when you create your EKS cluster.

Create your Amazon EKS Cluster VPC

1. Open the AWS CloudFormation console at https://us-west-2.console.aws.amazon.com/cloudformation.

2. From the navigation bar, select the **US West (Oregon)** region.

   Note
   At this time, Amazon EKS is only available in the **US West (Oregon)** region.
3. Choose **Create stack**.
4. For **Choose a template**, select **Specify an Amazon S3 template URL**.
5. Paste the following URL into the text area and choose **Next**:

```
https://amazon-eks.s3-us-west-2.amazonaws.com/2018-04-04/amazon-eks-vpc-sample.yaml
```

6. On the **Specify Details** page, fill out the parameters accordingly, and then choose **Next**.
   - **Stack name**: Choose a stack name for your AWS CloudFormation stack. For example, you can call it **eks-vpc**.
   - **Cluster name**: Choose a name to use for your Amazon EKS cluster. **You must use the same name when you create the cluster later.**
   - **VPC CIDR block**: Choose a CIDR range for your VPC. You may leave the default value.
   - **Subnet 1 block**: Choose a CIDR range for subnet 1. You may leave the default value.
   - **Subnet 2 block**: Choose a CIDR range for subnet 2. You may leave the default value.
7. (Optional) On the **Options** page, tag your stack resources. Choose **Next**.
8. On the **Review** page, choose **Create**.
9. When your stack is created, select it in the console and choose **Outputs**.
10. Record the **SecurityGroups** value for the security group that was created. You need this when you create your EKS cluster; this security group is applied to the cross-account elastic network interfaces that are created in your subnets that allow the Amazon EKS control plane to communicate with your worker nodes.
11. Record the **SubnetIds** for the subnets that were created. You need this when you create your EKS cluster; these are the subnets that your worker nodes are launched into.

Download and Install the Custom **kubectl** Binary

Amazon EKS clusters require custom-built **kubectl** and **kubelet** binaries that include the Heptio Authenticator to allow IAM authentication for your Kubernetes cluster. This custom version can replace your existing **kubectl** binary, and you do not need to maintain multiple versions.

Note
This requirement will be obsolete in Kubernetes version 1.10.

To download and install the custom kubectl binary

1. Download the **kubectl** binary from Amazon S3:
   - **Linux**: https://amazon-eks.s3-us-west-2.amazonaws.com/2018-04-04/bin/linux/amd64/kubectl
   - **MacOS**: https://amazon-eks.s3-us-west-2.amazonaws.com/2018-04-04/bin/darwin/amd64/kubectl
   - **Windows**: https://amazon-eks.s3-us-west-2.amazonaws.com/2018-04-04/bin/windows/amd64/kubectl.exe

   Use the command below to download the binary, substituting the correct URL for your platform. The example below is for macOS clients.

```
curl -o kubectl https://amazon-eks.s3-us-west-2.amazonaws.com/2018-04-04/bin/darwin/
amd64/kubectl
```

2. Apply execute permissions to the binary.

```
chmod +x ./kubectl
```

3. Copy the binary to a folder in your `$PATH`. If you have already installed **kubectl** (from Homebrew or Apt), then we recommend creating a `$HOME/bin/kubectl` and ensuring that `$HOME/bin` comes first in your `$PATH`.

```
cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

4. (Optional) Add the `$HOME/bin` path to your shell initialization file so that it is configured when you open a shell.

   - For Bash shells on macOS:

   ```
   echo 'export PATH=$HOME/bin:$PATH' >> ~/.bash_profile
   ```

   - For Bash shells on Linux:

   ```
   echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
   ```

Download and Install the Custom AWS CLI

Because Amazon EKS is not yet generally available, it is not included in the AWS CLI. However, you can download and install a custom version of the AWS CLI model to use Amazon EKS.

You must first download and install the standard AWS CLI and then add the Amazon EKS extension model. If you do not already have the standard AWS CLI installed, see Installing the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

Note
Your system's Python version must be Python 3, or Python 2.7.9 or greater; otherwise, you will receive `hostname doesn't match` errors with AWS CLI calls to Amazon EKS. For more information, see What are "hostname doesn't match" errors? in the Python Requests FAQ.

To download and install the custom AWS CLI model for Amazon EKS

1. Download the custom model. from Amazon S3: https://amazon-eks.s3-us-west-2.amazonaws.com/2018-04-04/eks-2017-11-01.normal.json

```
curl -O https://amazon-eks.s3-us-west-2.amazonaws.com/2018-04-04/
eks-2017-11-01.normal.json
```

2. To install the custom model for Amazon EKS, run the following command:

```
aws configure add-model --service-model file://eks-2017-11-01.normal.json --service-
name eks
```

Step 1: Create Your Amazon EKS Cluster

Now you can create your Amazon EKS cluster.

Important
When the Amazon EKS cluster is created, the IAM user who creates the cluster is added to the Kubernetes RBAC authorization table as the administrator. Initially, only that IAM user can make calls to the master API using **kubectl**. Also, the custom version of **kubectl** uses the AWS SDK for Go to authenticate against your Amazon EKS cluster. If you use the console to create the cluster, you must ensure that the same IAM user credentials are in the AWS SDK credential chain when you are running **kubectl** commands on your cluster.
If you install and configure the AWS CLI, you can configure the IAM credentials for your user. These also work for **kubectl**. If the AWS CLI is configured properly for your user, then **kubectl** can find those credentials as well. For more information, see Configuring the AWS CLI in the *AWS Command Line Interface User Guide*.

To create your cluster with the console

1. Open the Amazon EKS console at https://console.aws.amazon.com/eks/home?region=us-west-2.

   Important
   You must use IAM user credentials for this step, **not root credentials**. If you create your Amazon EKS cluster using root credentials, you cannot authenticate to the cluster. For more information, see How Users Sign In to Your Account in the *IAM User Guide*.
2. Choose **Create cluster**.

   Note
   If your IAM user does not have administrative privileges, you must explicitly add permissions for that user to call the Amazon EKS API operations. For more information, see Creating Amazon EKS IAM Policies (p. 28)
3. On the **Create master cluster** page, fill in the following fields and then choose **Create**:

   - **Master cluster name**: A unique name for your cluster. This must be the cluster name you used in Create your Amazon EKS Cluster VPC (p. 3).
   - **Kubernetes version**: The version of Kubernetes to use for your cluster. By default, the latest available version is selected.
   - **Role ARN**: The **RoleARN** value from the AWS CloudFormation output that you generated with Create your Amazon EKS Service Role (p. 3).
   - **Cluster Subnets**: The **SubnetIds** values (comma-separated) from the AWS CloudFormation output that you generated with Create your Amazon EKS Cluster VPC (p. 3).
   - **Security Groups**: The **SecurityGroups** value from the AWS CloudFormation output that you generated with Create your Amazon EKS Cluster VPC (p. 3).
4. On the **Clusters** page, choose the name of your newly created cluster to view the cluster information.
5. The **Status** field shows **CREATING** until the cluster provisioning process completes. When your cluster provisioning is complete (usually less than 10 minutes), and note the **Master endpoint** value. This is the endpoint for your Kubernetes master that you use in your **kubectl** configuration.
6. Retrieve the `certificateAuthority.data` for your cluster. Currently, this value is not displayed in the console, and you must use the following AWS CLI command to retrieve the value.

```
aws eks describe-cluster --region us-west-2 --cluster-name preview  --query
 cluster.certificateAuthority.data
```

To create your cluster with the AWS CLI

1. Create your cluster with the following command. Substitute your cluster name, the Amazon Resource Name (ARN) of your Amazon EKS service role that you created in Create your Amazon EKS Service Role (p. 3), and the subnet and security group IDs for the VPC you created in Create your Amazon EKS Cluster VPC (p. 3).

   Important
   You must use IAM user credentials for this step, **not root credentials**. If you create your Amazon EKS cluster using root credentials, you cannot authenticate to the cluster. For more information, see How Users Sign In to Your Account in the *IAM User Guide*.

   ```
   aws eks create-cluster --region us-west-2 --cluster-name preview --role-
   arn arn:aws:iam::111122223333:role/eks-service-role-AWSServiceRoleForAmazonEKS-
   EXAMPLEBKZRQR --subnets subnet-d474a49f subnet-e794259e --security-groups sg-e829f296
   ```

   Note
   If your IAM user does not have administrative privileges, you must explicitly add permissions for that user to call the Amazon EKS API operations. For more information, see Creating Amazon EKS IAM Policies (p. 28)

   Output:

   ```
   {
       "cluster": {
           "status": "NEW",
           "subnets": [
               "subnet-d474a49f",
               "subnet-e794259e"
           ],
           "clusterName": "preview",
           "roleArn": "arn:aws:iam::111122223333:role/eks-service-role-
   AWSServiceRoleForAmazonEKS-EXAMPLEBKZRQR",
           "desiredMasterVersion": "1.9",
           "certificateAuthority": {},
           "securityGroups": [
               "sg-e829f296"
           ],
           "createdAt": 1522779824913000
       }
   }
   ```

2. Cluster provisioning usually takes less than 10 minutes. You can query the status of your cluster with the following command; when your cluster status is `ACTIVE`, you can proceed.

   ```
   aws eks describe-cluster --region us-west-2 --cluster-name preview --query
    cluster.status
   ```

3. When your cluster provisioning is complete, retrieve the `masterEndpoint` and `certificateAuthority.data` values with the following commands. These must be added to your **kubectl** configuration so that you can communicate with your cluster.

   a. Retrieve the `masterEndpoint`.

   ```
   aws eks describe-cluster --region us-west-2 --cluster-name preview  --query
    cluster.masterEndpoint
   ```

   b. Retrieve the `certificateAuthority.data`.

```
aws eks describe-cluster --region us-west-2 --cluster-name preview  --query
 cluster.certificateAuthority.data
```

Step 2: Configure `kubectl` for Amazon EKS

In this section, you create a `kubeconfig` file for your cluster. The code block below shows the `kubeconfig` elements to add to your configuration. If you have an existing configuration and you are comfortable working with `kubeconfig` files, you can merge these elements into your existing setup. Be sure to replace the *`<endpoint-url>`* value with the full endpoint URL (for example, *`https://EXAMPLE_MASTER_ENDPOINT.yl4.us-west-2.eks.amazonaws.com`*) that was created for your cluster, replace the *`<base64-encoded-ca-cert>`* with the `certificateAuthority.data` value you retrieved earlier, and replace the *`<cluster-name>`* with your cluster name.

```
apiVersion: v1
clusters:
- cluster:
    server: <endpoint-url>
    certificate-authority-data: <base64-encoded-ca-cert>
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: aws
  name: aws
current-context: aws
kind: Config
preferences: {}
users:
- name: aws
  user:
    auth-provider:
      config:
        cluster-id: <cluster-name>
      name: aws
```

If you do not have an existing configuration, or to add the Amazon EKS cluster without modifying your existing configuration files, you can use the following procedure to add the Amazon EKS cluster to your configuration.

To create your `kubeconfig` file

1. Create the default **kubectl** folder if it does not already exist.

```
mkdir -p ~/.kube
```

2. Open your favorite text editor and copy the above `kubeconfig` code block into it.
3. Replace the *`<endpoint-url>`* with the endpoint URL that was created for your cluster.
4. Replace the *`<base64-encoded-ca-cert>`* with the `certificateAuthority.data` that was created for your cluster.
5. Replace the *`<cluster-name>`* with your cluster name.
6. Save the file to the default **kubectl** folder, with your cluster name in the file name. For example, if your cluster name is *`preview`*, save the file to `~/.kube/config-`*`preview`*.
7. Add that file path to your `KUBECONFIG` environment variable so that **kubectl** knows where to look for your cluster configuration.

```
export KUBECONFIG=$KUBECONFIG:~/.kube/config-preview
```

8. (Optional) Add the configuration to your shell initialization file so that it is configured when you open a shell.

   - For Bash shells on macOS:

   ```
   echo 'export KUBECONFIG=$KUBECONFIG:~/.kube/config-preview' >> ~/.bash_profile
   ```

   - For Bash shells on Linux:

   ```
   echo 'export KUBECONFIG=$KUBECONFIG:~/.kube/config-preview' >> ~/.bashrc
   ```

9. Test your configuration.

```
kubectl get all
```

   Note
   If you receive the error `No Auth Provider found for name "aws"`, you are not using the custom **kubectl** required for Amazon EKS during the preview. For more information, see Download and Install the Custom **kubectl** Binary (p. 4).

   Output:

```
NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
```

Step 3: Launch and Configure Amazon EKS Worker Nodes

Now that your VPC and Kubernetes master are created, you can launch and configure your worker nodes.

Important
Amazon EKS worker nodes are standard Amazon EC2 instances, and you are billed for them based on normal EC2 On-Demand prices. For more information, see Amazon EC2 Pricing.

To launch your worker nodes

1. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. From the navigation bar, select the **US West (Oregon)** region.

   Note
   Amazon EKS is only available in the **US West (Oregon)** region at this time.
3. Choose **Create stack**.
4. For **Choose a template**, select **Specify an Amazon S3 template URL**.
5. Paste the following URL into the text area and choose **Next**.

```
https://amazon-eks.s3-us-west-2.amazonaws.com/2018-04-04/amazon-eks-nodegroup.yaml
```

6. On the **Specify Details** page, fill out the parameters accordingly, and choose **Next**.

   - **Stack name**: Choose a stack name for your AWS CloudFormation stack. For example, you can call it ***<cluster-name>*-worker-nodes**.

- **ClusterName**: Enter the name that you used when you created your Amazon EKS cluster.

  Important
  This name must exactly match the name you used in Step 1: Create Your Amazon EKS Cluster (p. 6); otherwise, your worker nodes cannot join the cluster.
- **ClusterControlPlaneSecurityGroup**: Choose the **SecurityGroups** value from the AWS CloudFormation output that you generated with Create your Amazon EKS Cluster VPC (p. 3).
- **NodeGroupName**: Enter a name for your node group that is included in your Auto Scaling node group name.
- **NodeAutoScalingGroupMinSize**: Enter the minimum number of nodes that your worker node Auto Scaling group can scale in to.
- **NodeAutoScalingGroupMaxSize**: Enter the maximum number of nodes that your worker node Auto Scaling group can scale out to.

  Note
  We ask that you limit the number of worker nodes in your cluster no more than 100 during the preview.
- **NodeInstanceType**: Choose an instance type for your worker nodes.
- **NodeImageId**: Enter the current Amazon EKS worker node AMI ID (**ami-228dee5a**).

  Note
  The Amazon EKS worker node AMI is based on Amazon Linux 2, with a custom `kubelet` and `kube-proxy` built in. You can track security or privacy events for Amazon Linux 2 at the Amazon Linux Security Center or subscribe to the associated RSS feed. Security and privacy events include an overview of the issue, what packages are affected, and how to update your instances to correct the issue.
- **KeyName**: Enter the name of an Amazon EC2 SSH key pair that you can use to connect using SSH into your worker nodes with after they launch.
- **VpcId**: Enter the ID for the VPC that you created in Create your Amazon EKS Cluster VPC (p. 3).
- **Subnets**: Choose the subnets that you created in Create your Amazon EKS Cluster VPC (p. 3).

7. On the **Options** page, you can choose to tag your stack resources. Choose **Next**.
8. On the **Review** page, review your information, acknowledge that the stack might create IAM resources, and then choose **Create**.
9. When your stack has finished creating, select it in the console and choose the **Outputs** tab.
10. Record the **NodeInstanceRole** for the node group that was created. You need this when you configure your Amazon EKS worker nodes.

To enable worker nodes to join your cluster

1. Download, edit, and apply the AWS authenticator configuration map.

   a. Download the configuration map.

   ```
   curl -O https://amazon-eks.s3-us-west-2.amazonaws.com/2018-04-04/aws-auth-cm.yaml
   ```

   b. Open the file with your favorite text editor, replace the *<ARN of instance role (not instance profile)>* snippet with the **NodeInstanceRole** value that you recorded in the previous procedure, and save the file.

   Important
   Do not modify any other lines in this file.

   ```
   apiVersion: v1
   kind: ConfigMap
   ```

```
metadata:
  name: aws-auth
  namespace: default
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
        - system:node-proxier
```

c. Apply the configuration. (This command may take a few minutes to finish)

```
kubectl apply -f aws-auth-cm.yaml
```

Note
If you receive the error `No Auth Provider found for name "aws"`, you are not using the custom **kubectl** required for Amazon EKS during the preview. For more information, see Download and Install the Custom **kubectl** Binary (p. 4).

2. Watch the status of your nodes and wait for them to reach the `Ready` status.

```
kubectl get nodes --watch
```

Step 4: Launch a Guest Book Application

In this section, you create a sample guest book application to test your new cluster.

Note
For more information about setting up the guest book example, see https://github.com/kubernetes/examples/blob/master/guestbook-go/README.md in the Kubernetes documentation.

To create your guest book application

1. Create the Redis master replication controller.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/kubernetes/v1.9.2/
examples/guestbook-go/redis-master-controller.json
```

Note
If you receive the error `No Auth Provider found for name "aws"`, you are not using the custom **kubectl** required for Amazon EKS during the preview. For more information, see Download and Install the Custom **kubectl** Binary (p. 4).

Output:

```
replicationcontroller "redis-master" created
```

2. Create the Redis master service.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/kubernetes/v1.9.2/
examples/guestbook-go/redis-master-service.json
```

Output:

```
service "redis-master" created
```

3. Create the Redis slave replication controller.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/kubernetes/v1.9.2/
examples/guestbook-go/redis-slave-controller.json
```

   Output:

```
replicationcontroller "redis-slave" created
```

4. Create the Redis slave service.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/kubernetes/v1.9.2/
examples/guestbook-go/redis-slave-service.json
```

   Output:

```
service "redis-slave" created
```

5. Create the guestbook replication controller.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/kubernetes/v1.9.2/
examples/guestbook-go/guestbook-controller.json
```

   Output:

```
replicationcontroller "guestbook" created
```

6. Create the guestbook service.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/kubernetes/v1.9.2/
examples/guestbook-go/guestbook-service.json
```

   Output:

```
service "guestbook" created
```

7. Query the services in your cluster and wait until the **External IP** column for the `guestbook` service is populated.

   Note
   It may take several minutes before the IP address is available.

```
kubectl get services -o wide
```

8. After your external IP address is available, point a web browser to that address at port 3000 to view your guest book. For example, *`http://a7a95c2b9e69711e7b1a3022fdcfdf2e-1985673473.us-west-2.elb.amazonaws.com:3000`*

   Note
   It may take several minutes for DNS to propagate and for your guest book to show up.

Important
If you are unable to connect to the external IP address with your browser, be sure that your corporate firewall is not blocking non-standards ports, like 3000. You can try switching to a guest network to verify.

Step 5: Cleaning Up Guest Book Objects

When you are finished experimenting with your guest book application, you should clean up the resources that you created for it. The following command deletes all of the services and replication controllers for the guest book application:

```
kubectl delete rc/redis-master rc/redis-slave rc/guestbook svc/redis-master svc/redis-slave
 svc/guestbook
```

Note
If you receive the error `No Auth Provider found for name "aws"`, you are not using the custom **kubectl** required for Amazon EKS during the preview. Be sure that the custom **kubectl** for Amazon EKS is first in your `PATH`. For more information, see Download and Install the Custom **kubectl** Binary (p. 4).

Storage Classes

Amazon EKS clusters are not created with any storage classes. You must define storage classes for your cluster to use and you should define a default storage class for your persistent volume claims. For more information, see Storage Classes in the Kubernetes documentation.

To create an AWS storage class for your Amazon EKS cluster

1. Create an AWS storage class manifest file for your storage class. The below example defines a storage class called `gp2` that uses the Amazon EBS `gp2` volume type. For more information on the options available for AWS storage classes, see AWS in the Kubernetes documentation. For this example, the file is called `gp2-storage-class.yaml`.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: gp2
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
reclaimPolicy: Retain
mountOptions:
  - debug
```

2. Use **kubectl** to create the storage class from the manifest file.

```
kubectl create -f gp2-storage-class.yaml
```

Output:

```
storageclass "gp2" created
```

To define a default storage class

1. List the existing storage classes for your cluster. A storage class must be defined before you can set it as a default.

```
kubectl get storageclass
```

Output:

```
NAME      PROVISIONER              AGE
gp2       kubernetes.io/aws-ebs    8m
sc1       kubernetes.io/aws-ebs    6s
```

2. Choose a storage class and set it as your default by setting the `storageclass.kubernetes.io/is-default-class=true` annotation.

```
kubectl patch storageclass gp2 -p '{"metadata": {"annotations":
{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

Output:

```
storageclass "gp2" patched
```

3. Verify that the storage class is now set as default.

```
kubectl get storageclass
```

   Output:

```
gp2 (default)   kubernetes.io/aws-ebs   12m
sc1             kubernetes.io/aws-ebs   4m
```

Amazon EKS Networking

This chapter covers networking considerations for running Kubernetes on Amazon EKS.

VPC Considerations

When you create an Amazon EKS cluster, you specify the Amazon VPC subnets that your cluster will use. Amazon EKS requires subnets in at least two availability zones, but we recommend a network architecture that uses private subnets for your worker nodes and public subnets for Kubernetes to create public-facing load balancers within.

The subnets that you pass when you create the cluster influence where Amazon EKS places elastic network interfaces that are used for control plane to worker node communication.

It is possible to specify only public or private subnets when you create your cluster, but there are some limitations associated with these configurations:

- **Private-only**: Everything runs in a private subnet and Kubernetes cannot create internet-facing load balancers for your pods.
- **Public-only**: Everything runs in a public subnet, including your worker nodes.

Amazon EKS creates an elastic network interface in your private subnets to facilitate communication to your worker nodes. This communication channel supports Kubernetes functionality such as **kubectl exec** and **kubectl logs**. The security group that you specify when you create your cluster is applied to the elastic network interfaces that are created for your cluster control plane.

Subnet Tagging

All of your cluster's public and private subnets must be tagged appropriately for Kubernetes to discover them. Each public and private subnet that you want to use for your cluster should be tagged in the following way:

Key	Value
`kubernetes.io/cluster/<cluster-name>`	`shared` or `owned`

- **Key**: You must know the name of the cluster that you intend to use in this VPC, and replace the with that value.
- **Value**: Use `shared` to allow more than one cluster to use this VPC; you can apply multiple cluster tags to the same subnet if you use `shared`. Use `owned` if this is the only cluster that will use the VPC.

Security Group Considerations

If you create your VPC and worker node groups with the AWS CloudFormation templates provided in the Getting Started with Amazon EKS (p. 3) walkthrough, then your control plane and worker node security groups are configured with our recommended settings.

The security group for the worker nodes and the security group for the control plane communication to the worker nodes have been setup to prevent communication to privileged ports in the worker nodes.

If your applications require added inbound or outbound access from the control plane or worker nodes, you must add these rules to the security groups associated with your cluster. For more information, see Security Groups for Your VPC in the *Amazon VPC User Guide*.

Note
To allow proxy functionality on privileged ports or to run the CNCF conformance tests yourself, you must edit the security groups for your control plane and the worker nodes. The security group on the worker nodes side need to allow inbound access for ports 0 - 66535 from the control plane, and the control plane side needs to allow outbound access to the worker nodes on ports 0 - 65535.

The following tables show the minimum required and recommended security group settings for the control plane and worker node security groups for your cluster:

Control Plane Security Group

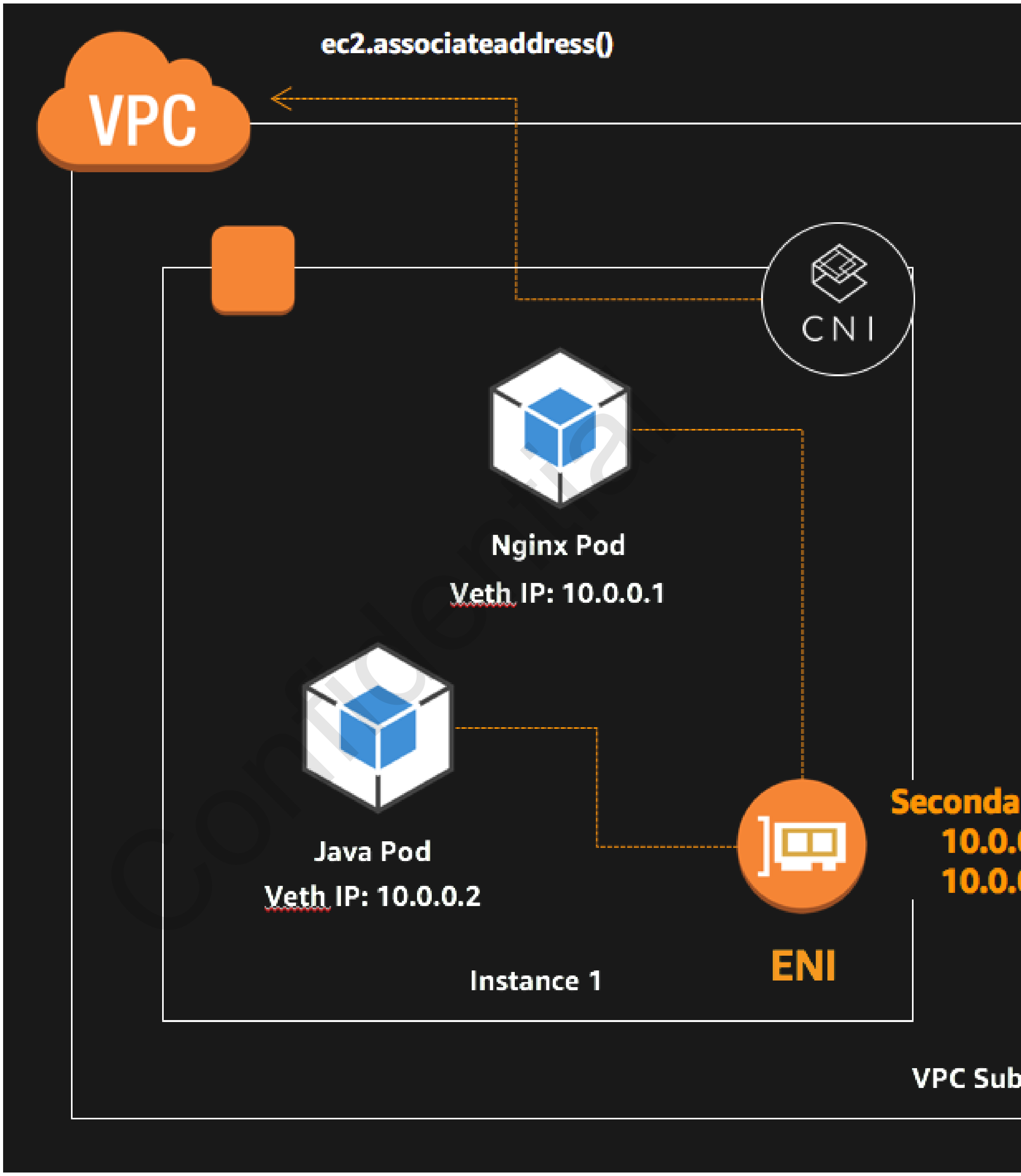
	Protocol	Port Range	Source	Destination
Minimum inbound traffic	TCP	443	Worker node security group	
Recommended inbound traffic	TCP	443	Worker node security group	
Minimum outbound traffic	TCP	10250		Worker node security group
Recommended outbound traffic	TCP	1025 - 65535		Worker node security group

Worker Node Security Group

	Protocol	Port Range	Source	Destination
Minimum inbound traffic	All TCP	All 10250	Worker node security group Control plane security group	
Recommended inbound traffic	All TCP	All 1025 - 65535	Worker node security group Control plane security group	
Minimum outbound traffic	TCP	443		Control plane security group
Recommended outbound traffic	All	All		0.0.0.0/0

Pod Networking

Amazon EKS supports native VPC networking via the Amazon VPC CNI plugin for Kubernetes. Using this CNI plugin allows Kubernetes pods to have the same IP address inside the pod as they do on the VPC network. This CNI plugin is an open source project that is maintained on GitHub.



The CNI plugin is responsible for allocating VPC IP addresses to Kubernetes nodes and configuring the necessary networking for pods on each node. The plugin consists of two primary components:

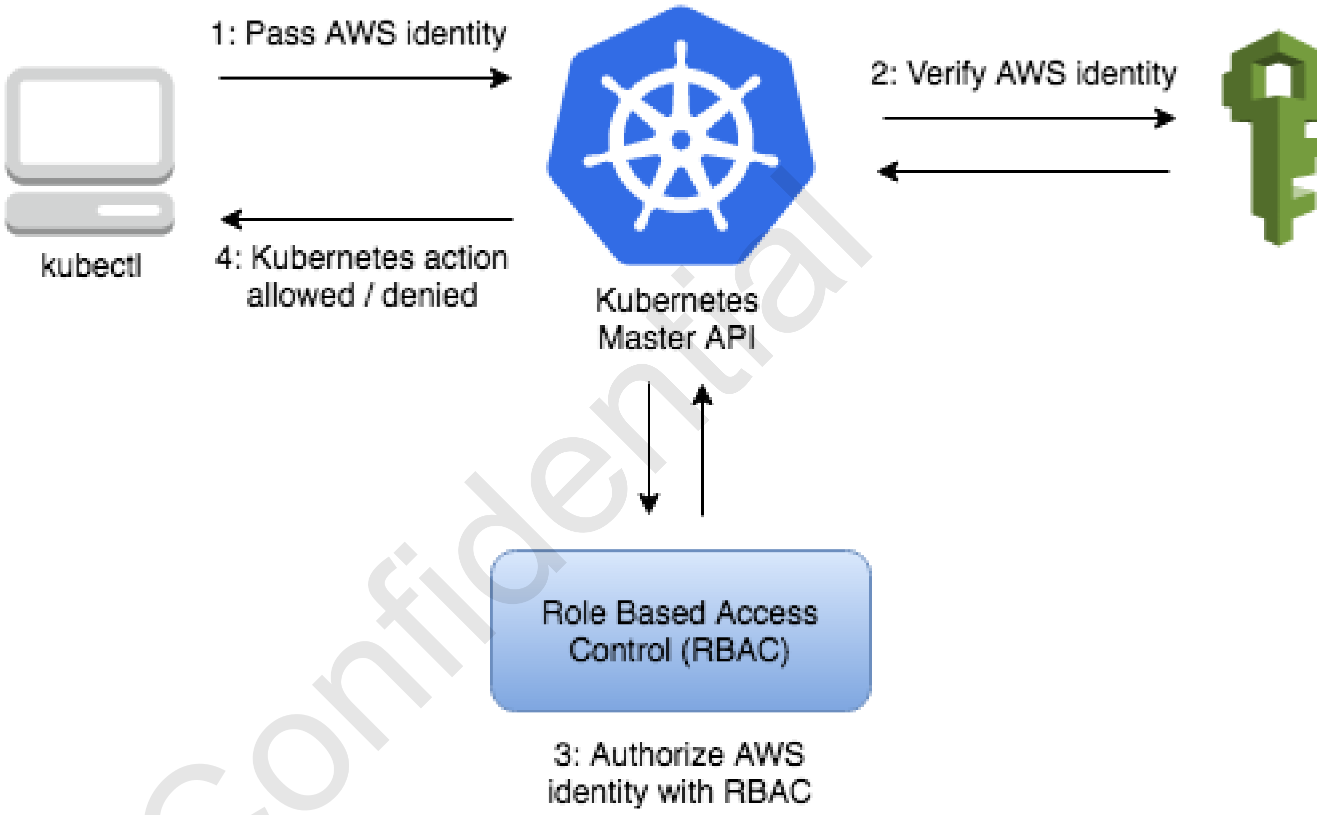
- The L-IPAM daemon is responsible for attaching elastic network interfaces to instances, assigning secondary IP addresses to elastic network interfaces, and maintaining a "warm pool" of IP addresses on each node for assignment to Kubernetes pods when they are scheduled.
- The CNI plugin itself is responsible for wiring the host network (for example, configuring the interfaces and virtual ethernet pairs) and adding the correct interface to the pod namespace.

For more information about the design and networking configuration, see Proposal: CNI plugin for Kubernetes networking over AWS VPC.

Elastic network interface and secondary IP address limitations by Amazon EC2 instance types are applicable. In general, larger instances can support more IP addresses. For more information, see IP Addresses Per Network Interface Per Instance Type in the *Amazon EC2 User Guide for Linux Instances*.

Managing Cluster Authentication

Amazon EKS uses IAM to provide authentication to your Kubernetes cluster (through the Heptio Authenticator), but it still relies on native Kubernetes Role Based Access Control (RBAC) for authorization. This means that IAM is only used for authentication of valid IAM entities. All permissions for interacting with your Amazon EKS cluster's Kubernetes API is managed through the native Kubernetes RBAC system.


1: Pass AWS identity
2: Verify AWS identity
kubectl
4: Kubernetes action allowed / denied
Kubernetes Master API
Role Based Access Control (RBAC)
3: Authorize AWS identity with RBAC

When you create an Amazon EKS cluster, your IAM user is automatically granted `system:master` permissions in the cluster's RBAC configuration. To grant additional AWS users the ability to interact with your cluster, you must edit the `aws-auth` ConfigMap within Kubernetes.

To add an IAM user or role to an Amazon EKS cluster

1. Ensure that the AWS credentials that **kubectl** is using are already authorized for your cluster. The IAM user that created the cluster has these permissions by default.
2. Open the `aws-auth` ConfigMap.

```
kubectl edit configmap/aws-auth
```

Note
If your default text editor does not wait for you to finish editing before returning to the command prompt, you see the following error:

```
Edit cancelled, no changes made.
```

You can switch your default text editor to a different editor, such as **vim**, or you can configure your default editor to wait until you are finished editing the file.

Example ConfigMap:

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will
 be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::111122223333:role/doc-test-worker-nodes-NodeInstanceRole-
WDO5P42N3ETB
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
        - system:node-proxier
kind: ConfigMap
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"mapRoles":"- rolearn: arn:aws:iam::111122223333:role/
doc-test-worker-nodes-NodeInstanceRole-WDO5P42N3ETB\n  username: system:node:
{{EC2PrivateDNSName}}\n  groups:\n    - system:bootstrappers\n    - system:nodes\n
 - system:node-proxier\n"},"kind":"ConfigMap","metadata":{"annotations":{},"name":"aws-
auth","namespace":"default"}}
  creationTimestamp: 2018-04-04T18:49:10Z
  name: aws-auth
  namespace: default
  resourceVersion: "780"
  selfLink: /api/v1/namespaces/default/configmaps/aws-auth
  uid: dcc31de5-3838-11e8-af26-02e00430057c
```

3. Add your IAM users, roles, or AWS accounts to the configMap.

   - **To add an IAM user:** add the user details to the `mapUsers` section of the ConfigMap, under `data`. Add this section if it does not already exist in the file. Each entry supports the following parameters:
     - **userarn**: The ARN of the IAM user to add.
     - **username**: The user name within Kubernetes to map to the IAM user. By default, the user name is the ARN of the IAM user.
     - **groups**: A list of groups within Kubernetes to which the user is mapped to.
   - **To add an IAM role:** add the role details to the `mapRoles` section of the ConfigMap, under `data`. Add this section if it does not already exist in the file. Each entry supports the following parameters:
     - **rolearn**: The ARN of the IAM role to add.
     - **username**: The user name within Kubernetes to map to the IAM role. By default, the user name is the ARN of the IAM role.
     - **groups**: A list of groups within Kubernetes to which the role is mapped to.
   - **To add an AWS account to be automapped:** add the AWS account ID (enclosed in quotation marks) to the `mapAccounts` section of the ConfigMap, under `data`. Add this section if it does not already exist in the file. Every AWS user and AWS role in that account is automatically mapped to a user in the Kubernetes cluster with the Amazon Resource Name (ARN) of that user or role as the `username`. However, no permissions are provided in RBAC by this action alone; you must still create role bindings in your cluster to provide these entities permissions.

For example, the block below contains:

- A `mapRoles` section that adds the worker node instance role so that worker nodes can register themselves with the cluster.
- A `mapUsers` section with the AWS users `admin` from the default AWS account, and `ops-user` from another AWS account. Both users are added to the `system:masters` group.
- A `mapAccounts` section with the AWS account, *111122223333*.

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will
 be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::555555555555:role/preview-worker-nodes-
NodeInstanceRole-74RF4UBDUKL6
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
        - system:node-proxier
  mapUsers: |
    - userarn: arn:aws:iam::555555555555:user/admin
      username: admin
      groups:
        - system:masters
    - userarn: arn:aws:iam::111122223333:user/ops-user
      username: ops-user
      groups:
        - system:masters
  mapAccounts: |
    - "111122223333"
```

4. Save the file and exit your text editor.

Amazon EKS Service Limits

The following table provides limitations for Amazon EKS.

Resource	Default Limit
Maximum number of Amazon EKS clusters	2
Maximum number of worker nodes per cluster *	100

* This is not an enforced limit, but we ask that you limit the number of worker nodes in your cluster to no more than 100 during the preview. To use more than 100 worker nodes in a cluster, contact the Amazon EKS service team in advance.

Subscribing to Amazon EKS Documentation Update Notifications

During the preview, the Amazon EKS documentation will receive updates. You can subscribe to the documentation update Amazon SNS topic to receive notifications when a documentation update is available. Notifications are available in all formats that Amazon SNS supports.

Note
Your user account must have `sns::subscribe` IAM permissions to subscribe to an SNS topic.

When you receive a documentation update notification, you can download the latest version from Amazon S3 and check the Document History (p. 41) topic to view a description of the update.

You can subscribe an Amazon SQS queue to this notification topic, but you must use a topic ARN that is in the same region. For more information, see Tutorial: Subscribing an Amazon SQS Queue to an Amazon SNS Topic in the *Amazon Simple Queue Service Developer Guide*.

You can also use an AWS Lambda function to trigger events when notifications are received. For more information, see Invoking Lambda functions using Amazon SNS notifications in the *Amazon Simple Notification Service Developer Guide*.

The Amazon SNS topic ARN is shown below.

AWS Region	Amazon SNS Topic ARN
`us-west-2`	`arn:aws:sns:us-west-2:602401143452:eks-docs-update`

To subscribe to documentation update notification emails in the AWS Management Console

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/v2/home.
2. In the region list, choose the same region as the topic ARN to which to subscribe. This example uses the `us-west-2` region.
3. Choose **Subscriptions** in the left navigation, then **Create subscription**.
4. In the **Create Subscription** dialog box, for **Topic ARN**, paste the Amazon EKS documentation update topic ARN: `arn:aws:sns:us-west-2:602401143452:eks-docs-update`.
5. For **Protocol**, choose **Email**. For **Endpoint**, type an email address you can use to receive the notification.
6. Choose **Create subscription**.
7. In your email application, open the message from AWS Notifications and open the link to confirm your subscription.

   Your web browser displays a confirmation response from Amazon SNS.

To subscribe to documentation update notification emails with the AWS CLI

1. Run the following command with the AWS CLI:

```
aws sns --region us-west-2 subscribe --topic-arn arn:aws:sns:us-
west-2:602401143452:eks-docs-update --protocol email --notification-
endpoint your_email@your_domain.com
```

2. In your email application, open the message from AWS Notifications and open the link to confirm your subscription.

   Your web browser displays a confirmation response from Amazon SNS.

Amazon EKS IAM Policies, Roles, and Permissions

By default, IAM users don't have permission to create or modify Amazon EKS resources, or perform tasks using the Amazon EKS API. (This means that they also can't do so using the Amazon EKS console or the AWS CLI.) To allow IAM users to create or modify clusters, you must create IAM policies that grant IAM users permission to use the specific resources and API actions they'll need, and then attach those policies to the IAM users or groups that require those permissions.

When you attach a policy to a user or group of users, it allows or denies the users permission to perform the specified tasks on the specified resources. For more information, see Permissions and Policies in the *IAM User Guide*. For more information about managing and creating custom IAM policies, see Managing IAM Policies.

Likewise, Amazon EKS makes calls to other AWS services on your behalf, so the service must authenticate with your credentials. This authentication is accomplished by creating an IAM role and policy that can provide these permissions and then associating that role with your compute environments when you create them. For more information, see Amazon EKS Service IAM Role (p. 29) and also IAM Roles in the *IAM User Guide*.

Getting Started

An IAM policy must grant or deny permissions to use one or more Amazon EKS actions.

Topics

Policy Structure

The following topics explain the structure of an IAM policy.

Topics

Policy Syntax

An IAM policy is a JSON document that consists of one or more statements. Each statement is structured as follows:

```
{
  "Statement":[{
    "Effect":"effect",
    "Action":"action",
    "Resource":"arn",
    "Condition":{
      "condition":{
        "key":"value"
        }
      }
    }
  ]
}
```

There are various elements that make up a statement:

- **Effect:** The *effect* can be `Allow` or `Deny`. By default, IAM users don't have permission to use resources and API actions, so all requests are denied. An explicit allow overrides the default. An explicit deny overrides any allows.
- **Action**: The *action* is the specific API action for which you are granting or denying permission.
- **Resource**: The resource that's affected by the action. Amazon EKS API operations currently do not support resource level permissions, so you must use the * wildcard to specify that all resources can be affected by the action.
- **Condition**: Conditions are optional. They can be used to control when your policy is in effect.

For more information about example IAM policy statements for Amazon EKS, see Creating Amazon EKS IAM Policies (p. 28).

Actions for Amazon EKS

In an IAM policy statement, you can specify any API action from any service that supports IAM. For Amazon EKS, use the following prefix with the name of the API action: `eks:`. For example: `eks:CreateCluster` and `eks:DeleteCluster`.

To specify multiple actions in a single statement, separate them with commas as follows:

```
"Action": ["eks:action1", "eks:action2"]
```

You can also specify multiple actions using wildcards. For example, you can specify all actions whose name begins with the word "Describe" as follows:

```
"Action": "eks:Describe*"
```

To specify all Amazon EKS API actions, use the * wildcard as follows:

```
"Action": "eks:*"
```

Checking That Users Have the Required Permissions

After you've created an IAM policy, we recommend that you check whether it grants users the permissions to use the particular API actions and resources they need before you put the policy into production.

First, create an IAM user for testing purposes, and then attach the IAM policy that you created to the test user. Then, make a request as the test user. You can make test requests in the console or with the AWS CLI.

Note
You can also test your policies with the IAM Policy Simulator. For more information on the policy simulator, see Working with the IAM Policy Simulator in the *IAM User Guide*.

If the policy doesn't grant the user the permissions that you expected, or is overly permissive, you can adjust the policy as needed and retest until you get the desired results.

Important
It can take several minutes for policy changes to propagate before they take effect. Therefore, we recommend that you allow five minutes to pass before you test your policy updates.

If an authorization check fails, the request returns an encoded message with diagnostic information. You can decode the message using the `DecodeAuthorizationMessage` action. For more information, see DecodeAuthorizationMessage in the *AWS Security Token Service API Reference*, and decode-authorization-message in the *AWS CLI Command Reference*.

Creating Amazon EKS IAM Policies

You can create specific IAM policies to restrict the calls and resources that users in your account have access to, and then attach those policies to IAM users.

When you attach a policy to a user or group of users, it allows or denies the users permission to perform the specified tasks on the specified resources. For more information, see Permissions and Policies in the *IAM User Guide*. For more information about managing and creating custom IAM policies, see Managing IAM Policies.

If your IAM user does not have administrative privileges, you must explicitly add permissions for that user to call the Amazon EKS API operations.

To create an IAM policy for an Amazon EKS admin user

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**, **Create policy**.
3. On the **JSON** tab, paste the following policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        }
    ]
}
```

4. Choose **Review policy**.
5. In the **Name** field, type your own unique name, such as `AmazonEKSAdminPolicy`.
6. Choose **Create Policy** to finish.

To attach an IAM policy to a user

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Users** and then choose the user you would like to attach the policy to.
3. Choose **Permissions**, **Add permissions**.
4. In the **Grant permissions** section, choose **Attach existing policies directly**.

5. Select the custom policy that you created in the previous procedure and choose **Next: Review**.
6. Review your details and choose **Add permissions** to finish.

Amazon EKS Service IAM Role

Amazon EKS makes calls to other AWS services on your behalf to manage the resources that you use with the service. Before you can use the service, you must have an IAM policy and role that provides the necessary permissions to Amazon EKS.

During the preview, the Amazon EKS service role is created with the AWS CloudFormation template in Create your Amazon EKS Service Role (p. 3).

The `AmazonEKSServiceRolePolicy` policy for the `AWSServiceRoleForAmazonEKS` IAM role is shown below.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:UpdateAutoScalingGroup",
                "ec2:AttachVolume",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:CreateRoute",
                "ec2:CreateSecurityGroup",
                "ec2:CreateTags",
                "ec2:CreateVolume",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteRoute",
                "ec2:DeleteSecurityGroup",
                "ec2:DeleteVolume",
                "ec2:DescribeInstances",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVolumes",
                "ec2:DescribeVpcs",
                "ec2:DetachVolume",
                "ec2:ModifyInstanceAttribute",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:RevokeSecurityGroupIngress",
                "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
                "elasticloadbalancing:AttachLoadBalancerToSubnets",
                "elasticloadbalancing:ConfigureHealthCheck",
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateLoadBalancerListeners",
                "elasticloadbalancing:CreateLoadBalancerPolicy",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:DeleteLoadBalancerListeners",
                "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
                "elasticloadbalancing:DescribeLoadBalancerAttributes",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DetachLoadBalancerFromSubnets",
                "elasticloadbalancing:ModifyLoadBalancerAttributes",
                "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
                "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

You can use the following procedure to check and see if your account already has the Amazon EKS service role and to attach the managed IAM policy if needed.

To check for the `AWSServiceRoleForAmazonEKS` in the IAM console

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Roles**.
3. Search the list of roles for `AWSServiceRoleForAmazonEKS`. If the role does not exist, see Create your Amazon EKS Service Role (p. 3) to create the role. If the role does exist, select the role to view the attached policies.
4. Choose **Permissions**.
5. Ensure that the **AmazonEKSServiceRolePolicy** inline policy is attached to the role. If the policy is attached, your Amazon EKS service role is properly configured.
6. Choose **Trust Relationships**, **Edit Trust Relationship**.
7. Verify that the trust relationship contains the following policy. If the trust relationship matches the policy below, choose **Cancel**. If the trust relationship does not match, copy the policy into the **Policy Document** window and choose **Update Trust Policy**.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

To create the `AWSServiceRoleForAmazonEKS` with the AWS CLI

1. Create a file named `AmazonEKSServiceRolePolicy` and add the following content to it.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:UpdateAutoScalingGroup",
                "ec2:AttachVolume",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:CreateRoute",
                "ec2:CreateSecurityGroup",
                "ec2:CreateTags",
                "ec2:CreateVolume",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteRoute",
                "ec2:DeleteSecurityGroup",
                "ec2:DeleteVolume",
                "ec2:DescribeInstances",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVolumes",
                "ec2:DescribeVpcs",
                "ec2:DetachVolume",
                "ec2:ModifyInstanceAttribute",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:RevokeSecurityGroupIngress",
                "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
                "elasticloadbalancing:AttachLoadBalancerToSubnets",
                "elasticloadbalancing:ConfigureHealthCheck",
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateLoadBalancerListeners",
                "elasticloadbalancing:CreateLoadBalancerPolicy",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:DeleteLoadBalancerListeners",
                "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
                "elasticloadbalancing:DescribeLoadBalancerAttributes",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DetachLoadBalancerFromSubnets",
                "elasticloadbalancing:ModifyLoadBalancerAttributes",
                "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
                "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

2. Create the `AmazonEKSServiceRolePolicy` in IAM. Note the policy ARN in the API response.

```
aws iam create-policy --policy-name AmazonEKSServiceRolePolicy --policy-document
 file://AmazonEKSServiceRolePolicy
```

3. Create the `AWSServiceRoleForAmazonEKS`.

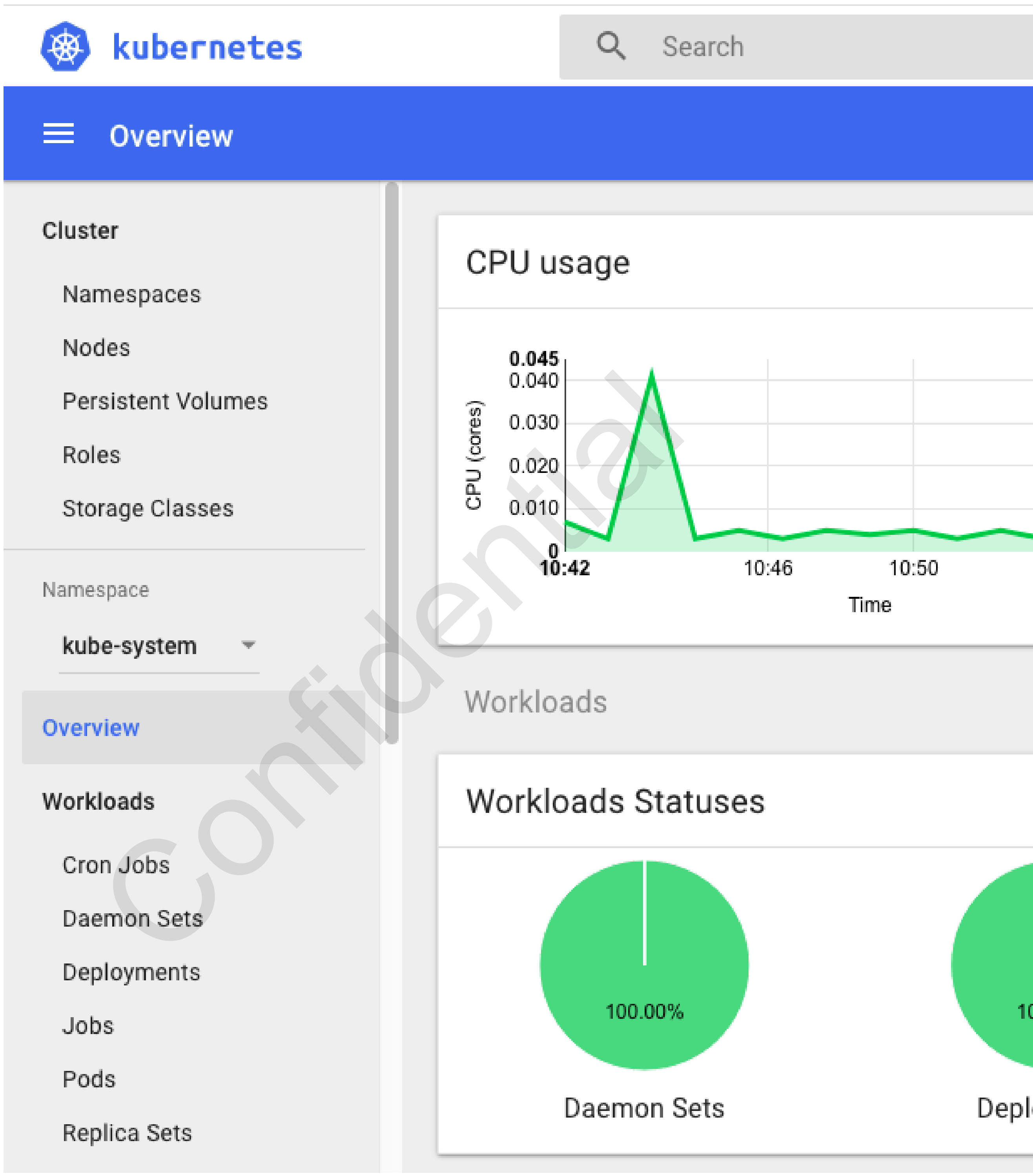
```
aws iam create-role --role-name AWSServiceRoleForAmazonEKS --assume-role-policy-
document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'
```

4. Attach the `AmazonEKSServiceRolePolicy` to the `AWSServiceRoleForAmazonEKS`. For `--policy-arn`, use the returned policy ARN from Step 2 (p. 31).

```
aws iam attach-role-policy --role-name AmazonEKSServiceRolePolicy --policy-arn
 arn:aws:iam::111122223333:policy/AmazonEKSServiceRolePolicy
```

Tutorial: Deploy the Kubernetes Web UI (Dashboard)

This tutorial guides you through deploying the Kubernetes dashboard to your Amazon EKS cluster, complete with CPU and memory metrics. It also helps you to create an Amazon EKS administrator service account that you can use to securely connect to the dashboard to view and control your cluster.



Prerequisites

This tutorial assumes the following:

- You have created an Amazon EKS cluster by following the steps in Getting Started with Amazon EKS (p. 3).
- The security groups for your control plane elastic network interfaces and worker nodes follow the recommended settings in Security Group Considerations (p. 16).
- You are using the custom **kubectl** client (p. 4) for Amazon EKS, and that it is configured to communicate with your Amazon EKS cluster (p. 8).

Step 1: Deploy the Dashboard

Use the following steps to deploy the Kubernetes dashboard, `heapster`, and the `influxdb` backend for CPU and memory metrics to your cluster.

To deploy the Kubernetes dashboard

1. Deploy the Kubernetes dashboard to your cluster.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/master/src/
deploy/recommended/kubernetes-dashboard.yaml
```

   Output:

```
secret "kubernetes-dashboard-certs" created
serviceaccount "kubernetes-dashboard" created
role "kubernetes-dashboard-minimal" created
rolebinding "kubernetes-dashboard-minimal" created
deployment "kubernetes-dashboard" created
service "kubernetes-dashboard" created
```

2. Deploy `heapster` to enable container cluster monitoring and performance analysis on your cluster.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/heapster/master/deploy/
kube-config/influxdb/heapster.yaml
```

   Output:

```
serviceaccount "heapster" created
deployment "heapster" created
service "heapster" created
```

3. Deploy the `influxdb` backend for `heapster` to your cluster.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/heapster/master/deploy/
kube-config/influxdb/influxdb.yaml
```

   Output:

```
deployment "monitoring-influxdb" created
service "monitoring-influxdb" created
```

4. Create the `heapster` cluster role binding for the dashboard.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/heapster/master/deploy/
kube-config/rbac/heapster-rbac.yaml
```

Output:

```
clusterrolebinding "heapster" created
```

Step 2: Create an `eks-admin` Service Account and Cluster Role Binding

By default, the Kubernetes dashboard user has limited permissions. In this section, you create an `eks-admin` service account and cluster role binding that you can use to securely connect to the dashboard with admin-level permissions. For more information, see Managing Service Accounts in the Kubernetes documentation.

To create the `eks-admin` service account and cluster role binding

Important
The example service account created with this procedure has full `cluster-admin` (superuser) privileges on the cluster. For more information, see Using RBAC Authorization in the Kubernetes documentation.

1. Create a file called `eks-admin-service-account.yaml` with the text below.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eks-admin
  namespace: kube-system
```

2. Apply the service account to your cluster.

```
kubectl apply -f eks-admin-service-account.yaml
```

Output:

```
serviceaccount "eks-admin" created
```

3. Create a file called `eks-admin-cluster-role-binding.yaml` with the text below.

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: eks-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: eks-admin
  namespace: kube-system
```

4. Apply the cluster role binding to your cluster.

```
kubectl apply -f eks-admin-cluster-role-binding.yaml
```

Output:

```
clusterrolebinding "eks-admin" created
```

Step 3: Connect to the Dashboard

Now that the Kubernetes dashboard is deployed to your cluster, and you have an administrator service account that you can use to view and control your cluster, you can connect to the dashboard with that service account.

To connect to the Kubernetes dashboard

1. Start the **kubectl proxy**.

   ```
   kubectl proxy
   ```

2. Retrieve an authentication token for the `eks-admin` service account. Copy the *<authentication_token>* value from the output; you will use this token to connect to the dashboard.

   ```
   kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep eks-
   admin | awk '{print $1}')
   ```

   Output:

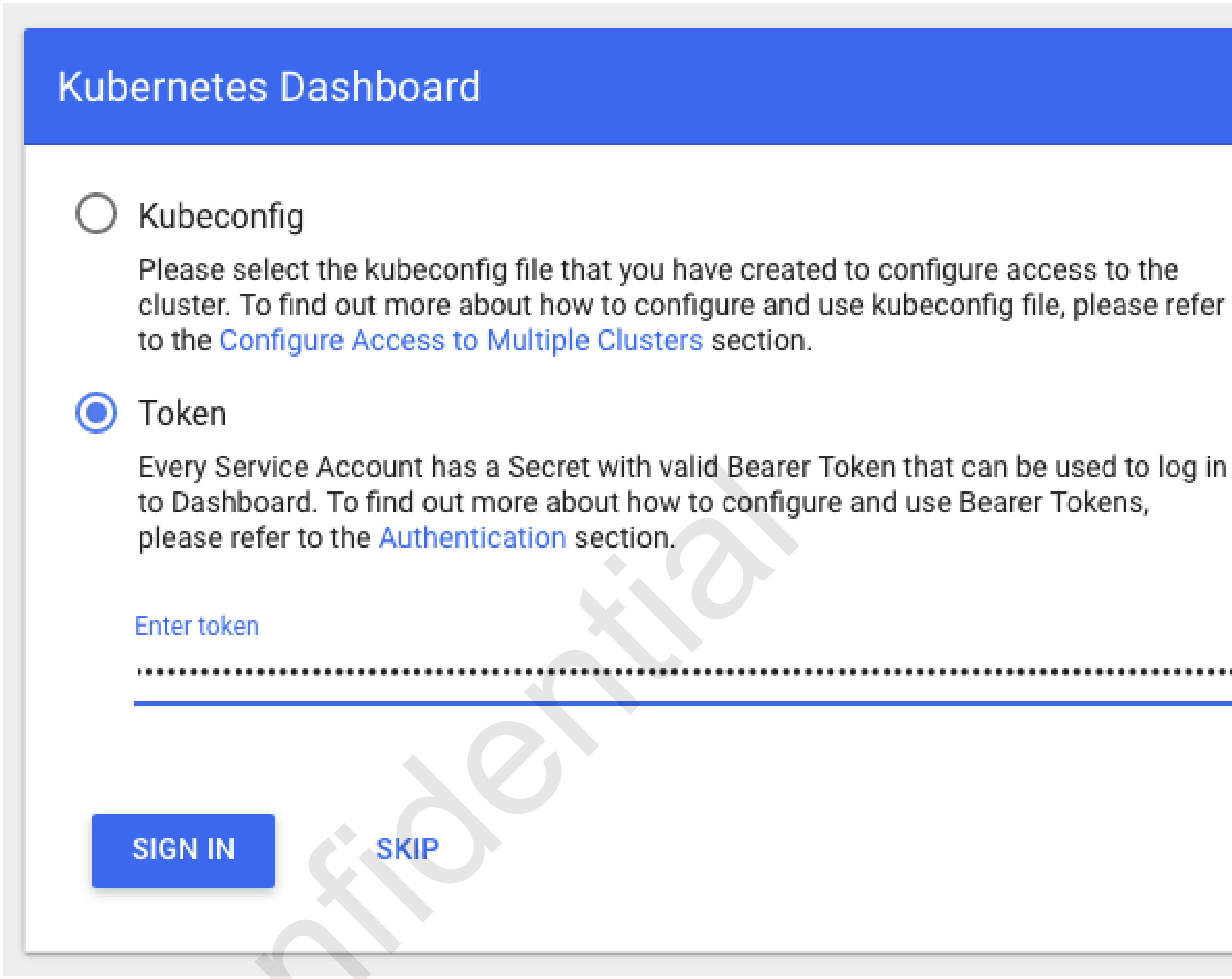
   ```
   Name:         eks-admin-token-b5zv4
   Namespace:    kube-system
   Labels:       <none>
   Annotations:  kubernetes.io/service-account.name=eks-admin
                 kubernetes.io/service-account.uid=bcfe66ac-39be-11e8-97e8-026dce96b6e8

   Type:  kubernetes.io/service-account-token

   Data
   ====
   ca.crt:      1025 bytes
   namespace:   11 bytes
   token:       <authentication_token>
   ```

3. Open the following link with a web browser to access the dashboard endpoint: http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/

4. Choose **Token**, paste the *<authentication_token>* output from the previous command into the token field, and choose **SIGN IN**.


Kubernetes Dashboard

Kubeconfig

Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the Configure Access to Multiple Clusters section.

Token

Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the Authentication section.

Enter token

Note
It may take a few minutes before CPU and memory metrics appear in the dashboard.

Step 4: Next Steps

After you have connected to your Kubernetes cluster dashboard, you can view and control your cluster using your `eks-admin` service account. For more information on using the dashboard, see the project documentation on GitHub.

Tutorial: Creating a VPC with Public and Private Subnets for Your Amazon EKS Cluster

This tutorial guides you through creating a VPC with two public subnets and two private subnets, which are provided with internet access through a NAT gateway. You can use this VPC for your Amazon EKS cluster. We recommend a network architecture that uses private subnets for your worker nodes, and public subnets for Kubernetes to create public-facing load balancers within.

Topics

Step 1: Create an Elastic IP Address for Your NAT Gateway

Worker nodes in private subnets require a NAT gateway for outbound internet access. A NAT gateway requires an Elastic IP address in your public subnet, but the VPC wizard does not create one for you. Create the Elastic IP address before running the VPC wizard.

To create an Elastic IP address

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the left navigation pane, choose **Elastic IPs**.
3. Choose **Allocate new address**, **Allocate**, **Close**.
4. Note the **Allocation ID** for your newly created Elastic IP address; you enter this later in the VPC wizard.

Step 2: Run the VPC Wizard

The VPC wizard automatically creates and configures most of your VPC resources for you.

To run the VPC wizard

1. In the left navigation pane, choose **VPC Dashboard**.
2. Choose **Start VPC Wizard**, **VPC with Public and Private Subnets**, **Select**.
3. For **VPC name**, give your VPC a unique name.

4. For **Elastic IP Allocation ID**, choose the ID of the Elastic IP address that you created earlier.
5. Choose **Create VPC**.
6. When the wizard is finished, choose **OK**. Note the Availability Zone in which your VPC subnets were created. Your additional subnets should be created in a different Availability Zone.

Step 3: Create Additional Subnets

The wizard creates a VPC with a single public and a single private subnet in a single Availability Zone. For greater availability, you should create at least one more of each subnet type in a different Availability Zone so that your VPC has both public and private subnets across two Availability Zones.

To create an additional private subnet

1. In the left navigation pane, choose **Subnets**.
2. Choose **Create Subnet**.
3. For **Name tag**, enter a name for your subnet, such as **Private subnet**.
4. For **VPC**, choose the VPC that you created earlier.
5. For **Availability Zone**, choose a different Availability Zone than your original subnets in the VPC.
6. For **IPv4 CIDR block**, enter a valid CIDR block. For example, the wizard creates CIDR blocks in 10.0.0.0/24 and 10.0.1.0/24 by default. You could use **10.0.3.0/24** for your second private subnet.
7. Choose **Yes, Create**.

To create an additional public subnet

1. In the left navigation pane, choose **Subnets** and then **Create Subnet**.
2. For **Name tag**, enter a name for your subnet, such as **Public subnet**.
3. For **VPC**, choose the VPC that you created earlier.
4. For **Availability Zone**, choose the same Availability Zone as the additional private subnet that you created in the previous procedure.
5. For **IPv4 CIDR block**, enter a valid CIDR block. For example, the wizard creates CIDR blocks in 10.0.0.0/24 and 10.0.1.0/24 by default. You could use **10.0.2.0/24** for your second public subnet.
6. Choose **Yes, Create**.
7. Select the public subnet that you just created and choose **Route Table**, **Edit**.
8. By default, the private route table is selected. Choose the other available route table so that the **0.0.0.0/0** destination is routed to the internet gateway (**igw-*xxxxxxxx***) and choose **Save**.
9. With your second public subnet still selected, choose **Subnet Actions**, **Modify auto-assign IP settings**.
10. Select **Enable auto-assign public IPv4 address** and choose **Save**, **Close**.

Step 4: Tag Your Subnets

All of your cluster's public and private subnets must be tagged appropriately for Kubernetes to discover them. Each public and private subnet that you want to use for your cluster should be tagged in the following way:

Key	Value
`kubernetes.io/cluster/<cluster-name>`	`shared` or `owned`

- **Key**: You must know the name of the cluster that you intend to create in this VPC, and replace the with that value.
- **Value**: Use `shared` to allow more than one cluster to use this VPC; you can apply multiple cluster tags to the same subnet if you use `shared`. Use `owned` if this is the only cluster that will use the VPC.

To tag your subnets

1. In the left navigation pane, filter your VPC's resources by choosing your VPC in the **Filter by VPC** field, then choose **Subnets**.
2. For each subnet in your VPC, complete the following steps.

   a. Choose the **Tags** tab, and then choose **Edit**.
   b. For **Key**, enter `kubernetes.io/cluster/`, replacing with the name of your cluster.
   c. For **Value**, enter `shared` to allow more than one cluster in this VPC, or `owned` to allow only this cluster to use your VPC.
   d. Choose **Save** to finish.

Step 5: Create a Control Plane Security Group

When you create an Amazon EKS cluster, your cluster control plane creates elastic network interfaces in your subnets to enable communication with the worker nodes. You should create a security group that is dedicated to your Amazon EKS cluster control plane, so that you can apply inbound and outbound rules to govern what traffic is allowed across that connection. When you create the cluster, you specify this security group, and that is applied to the elastic network interfaces that are created in your subnets.

The worker node AWS CloudFormation template used in Step 3: Launch and Configure Amazon EKS Worker Nodes (p. 9) creates a worker node security group, and it applies the necessary rules to allow communication with the master automatically, but you must specify the control plane security group when you create a stack from that template.

To create a control plane security group

1. In the left navigation pane, filter your VPC's resources by choosing your VPC in the **Filter by VPC** field, then choose **Security Groups**.
2. Choose **Create Security Group**.
3. Fill in the following fields and choose **Yes, Create**.

   - For **Name tag**, provide a name for your security group. For example, `*<cluster-name>*-control-plane`.
   - For **Description**, provide a description of your security group to help you identify it later.
   - For **VPC**, choose the VPC that you are using for your Amazon EKS cluster.

Next Steps

After you have created your VPC, you can try the Getting Started with Amazon EKS (p. 3) walkthrough, but you can skip the Create your Amazon EKS Cluster VPC (p. 3) section and use these subnets and security groups for your cluster.

Document History for Amazon EKS

The following table describes the documentation for this release of Amazon EKS. We also update the documentation frequently to address the feedback that you send us.

- **API version: 2017-11-01**
- **Latest documentation update:** April 10, 2018

Change	Description	Date
Added Kubernetes Dashboard installation tutorial	This tutorial guides you through installing the Kubernetes dashboard on your Amazon EKS cluster, complete with CPU and memory metrics. For more information, see Tutorial: Deploy the Kubernetes Web UI (Dashboard) (p. 32).	April 10, 2018
Added VPC and security group considerations	This topic describes the network and security group requirements and recommendations for Amazon EKS clusters. For more information, see Amazon EKS Networking (p. 16).	April 9, 2018
Added VPC creation tutorial	This tutorial creates a VPC with public and private subnets for you to use as a starting point for your cluster configuration. For more information, see Tutorial: Creating a VPC with Public and Private Subnets for Your Amazon EKS Cluster (p. 38).	April 6, 2018
Added support for **exec** and **logs** and CNCF Certified Kubernetes Conformance Program.	Amazon EKS clusters now support **exec** and **logs**. Amazon EKS is also now listed as a Certified Kubernetes platform by CNCF.	April 5, 2018
Worker node AMI security update	Updated the worker node AMI to `ami-74128f0c` to fix CVE-2018-1068.	March 20, 2018
Added support for Kubernetes 1.9.2	Amazon S3 resource and worker AMI update for Kubernetes 1.9.2 support in Amazon EKS clusters.	March 12, 2018
Added doc update Amazon SNS subscription topic	During the preview, the Amazon EKS documentation will receive regular updates. You can subscribe to an Amazon SNS	February 2, 2018

Change	Description	Date
	topic that will notify you when an update is available. For more information, see Subscribing to Amazon EKS Documentation Update Notifications (p. 24).	
Added Python minimum version for AWS CLI	The AWS CLI requires Python 3 or Python 2.7.9 or greater to make calls to Amazon EKS.	February 1, 2018
Updated worker AMI ID	Worker AMI update for Spectre and Meltdown security patches	January 26, 2018
Preview documentation	Initial getting started docs for preview	December 18, 2017

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the *AWS General Reference*.