**1. Self-Managed Kubernetes Cluster (kubeadm)**

In a self-managed Kubernetes cluster (e.g., using kubeadm), you need to set up encryption at rest for secrets stored in **etcd**. This is done by configuring an encryption-config.yaml file.

**Steps for kubeadm:**

1. **Create the encryption-config.yaml File**: You need to create a file named encryption-config.yaml that contains the encryption configuration. Here’s an example configuration that uses AES encryption for secrets:

yaml

Copy code

kind: EncryptionConfig

apiVersion: admission.k8s.io/v1

resources:

- resources:

- secrets

providers:

- aescbc:

keys:

- name: key1

secret: <base64-encoded-key>

- identity: {}

1. **Generate a Base64 Encoded Key**: Use the following openssl command to generate a random 32-byte encryption key and then base64 encode it:

bash

Copy code

openssl rand -base64 32

Replace <base64-encoded-key> in the encryption-config.yaml file with the generated base64 key.

1. **Apply the Encryption Config**:
   * Edit the kube-apiserver manifest (located at /etc/kubernetes/manifests/kube-apiserver.yaml) and add the following line to specify the encryption config file:

yaml

Copy code

- --encryption-provider-config=/etc/kubernetes/encryption-config.yaml

* + Save the file. The kube-apiserver will restart automatically because it runs as a static pod in the /etc/kubernetes/manifests/ directory.

1. **Verify Encryption**: To verify that the encryption is applied, create a secret and check if it is encrypted in **etcd**.

Example:

bash

Copy code

kubectl create secret generic my-secret --from-literal=username=myuser --from-literal=password=mypassword

kubectl get secret my-secret -o jsonpath='{.data.username}' | base64 --decode

When viewing the secret, the secret data should be encrypted.

**2. Amazon EKS (Elastic Kubernetes Service)**

In **Amazon EKS**, AWS manages the control plane, and encryption at rest for secrets is handled through **AWS Key Management Service (KMS)**. You can configure encryption at rest by using either **AWS-managed keys** or **customer-managed keys**.

**Steps for EKS:**

1. **Create a KMS Key**: To enable encryption, you first need to create a **KMS key** using AWS CLI or the AWS Console.

Example to create a key using AWS CLI:

bash

Copy code

aws kms create-key --description "K8s Encryption Key" --key-usage ENCRYPT\_DECRYPT --origin AWS\_KMS

1. **Grant Permissions to EKS IAM Role**: Ensure that the IAM role associated with your EKS cluster has permissions to use the KMS key. You will need to attach the following permissions to the IAM role:

json

Copy code

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"kms:Encrypt",

"kms:Decrypt",

"kms:GenerateDataKey",

"kms:DescribeKey"

],

"Resource": "<KMS-key-ARN>"

}

]

}

1. **Create or Update an EKS Cluster with Encryption**: When creating a new EKS cluster, or updating an existing one, specify the KMS key ARN for encryption.

Example for creating a new EKS cluster:

bash

Copy code

aws eks create-cluster \

--name my-cluster \

--role-arn arn:aws:iam::123456789012:role/eks-service-role \

--resources-vpc-config subnetIds=subnet-abcdefg,subnet-hijklmn,securityGroupIds=sg-123456 \

--encryption-config resources=secrets,keyArn=arn:aws:kms:us-west-2:123456789012:key/<KMS-key-ID>

Or, if you’re updating an existing cluster:

bash

Copy code

aws eks update-cluster-config \

--name my-cluster \

--region us-west-2 \

--encryption-config resources=secrets,keyArn=arn:aws:kms:us-west-2:123456789012:key/<KMS-key-ID>

1. **Verify Encryption**: After creating or updating the EKS cluster, use the following command to check the encryption configuration:

bash

Copy code

aws eks describe-cluster --name my-cluster --region us-west-2

Look for the encryptionConfig field in the output, which should show the KMS key being used for secrets encryption.

**3. Azure AKS (Azure Kubernetes Service)**

In **Azure AKS**, encryption at rest is managed through **Azure Key Vault** and **customer-managed keys (CMK)**. By default, AKS uses **Azure-managed keys** to encrypt secrets, but you can configure **customer-managed keys** from **Azure Key Vault** for more control.

**Steps for AKS:**

1. **Create an Azure Key Vault**: If you don’t already have one, create a new Key Vault to store your encryption keys.

Example to create a Key Vault using Azure CLI:

bash

Copy code

az keyvault create --name myKeyVault --resource-group myResourceGroup --location eastus

1. **Create a Key in Key Vault**: You will need to create a key in the Key Vault that will be used for encryption.

bash

Copy code

az keyvault key create --vault-name myKeyVault --name myEncryptionKey --protection software

1. **Create or Update an AKS Cluster with Encryption**: When creating a new AKS cluster, or updating an existing one, you can specify the **Azure Key Vault** and **customer-managed key** for encryption.

Example for creating a new AKS cluster with a customer-managed key:

bash

Copy code

az aks create \

--resource-group myResourceGroup \

--name myAKSCluster \

--enable-managed-identity \

--enable-secret-encryption \

--encryption-keyvault-id /subscriptions/{subscription-id}/resourceGroups/myResourceGroup/providers/Microsoft.KeyVault/vaults/myKeyVault \

--encryption-key-id /subscriptions/{subscription-id}/resourceGroups/myResourceGroup/providers/Microsoft.KeyVault/vaults/myKeyVault/keys/myEncryptionKey \

--node-count 3 \

--enable-addons monitoring

Replace the placeholders {subscription-id}, myResourceGroup, and myKeyVault with your actual Azure subscription ID, resource group, and Key Vault details.

1. **Verify Encryption**: After the cluster is created or updated, you can verify that encryption at rest is enabled by using the following command:

bash

Copy code

az aks show --resource-group myResourceGroup --name myAKSCluster --query "addonProfiles"

Look for the encryption configuration in the output to verify that the encryption key is being used.

**Summary:**

* **In kubeadm (self-managed)**: You manually configure encryption by creating an encryption-config.yaml file and applying it to the kube-apiserver.
* **In EKS**: Encryption is managed by AWS using **AWS Key Management Service (KMS)**. You enable encryption by specifying a KMS key during cluster creation or update.
* **In AKS**: Encryption is managed by **Azure Key Vault**. You can enable **customer-managed keys** from Azure Key Vault when creating or updating the AKS cluster