Create an EKS Cluster:

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install

aws configure
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
curl --silent --location "https://github.com/weaveworks/eksctl/releases/download/0.75.0/eksctl_Linux_amd64.tar.gz" | tar xz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin
eksctl create cluster --name my-cluster --region us-west-2 --nodes 3

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
aws configure
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
curl --silent --location "https://github.com/weaveworks/eksctl/releases/download/0.75.0/eksctl_Linux_amd64.tar.gz" | tar xz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin
```

```
eksctl create cluster --name my-cluster --region us-west-2 --nodes 3
k get svc
eksctl get pods
eksctl get nodegroup
eksctl get podidentityassociatio
eksctl get podidentityassociation
root@developer2:/tmp# eksctl get cluster
NAME            REGION
my-cluster      us-west-2
```

Deploy Jenkins:

kubectl config set-context --current --namespace="jenkins"

k create -f jenkins-deployment.yaml

sudo mkdir -p /mnt/data

sudo chmod 777 /mnt/data
kubectl get services -n jenkins

k logs -f jenkins-5b7bc8c5d4-wt4bp

```
aws eks --region us-west-2 update-kubeconfig --name my-cluster
k get po
k get ns
kubectl get nodes
k get po -A
vi ns.yml
k create -f ns.yml
k get ns
vi jenkins-pv.yaml
k create -f jenkins-pv.yaml
vi jenkins-pvc.yaml
k create -f jenkins-pvc.yaml
k get pv,pvc
vi jenkins-deployment.yaml
k create -f jenkins-deployment.yaml
k get deploy
vi jenkins-deployment.yaml
kubectl config set-context --current --namespace="jenkins"
k get deploy
k get po
k logs -f jenkins-6547c655d4-qjpxh
sudo mkdir -p /mnt/data
sudo chmod 777 /mnt/data
k get deploy
k get po
k delete deploy jenkins
k get po
>jenkins-deployment.yaml
vi jenkins-deployment.yaml
k create -f jenkins-deployment.yaml
k get deploy
k get po
vi jenkins-service.yaml
k create -f jenkins-service.yaml
```

aws ec2 run-instances --image-id ami-0c272455b0778ebeb --count 1 --instance-type t2.micro --key-name ec2instance --security-groups ec2sg
aws ec2 authorize-security-group-ingress --group-id sg-081e395d2ec65bd12 --protocol tcp --port 8080 --cidr 0.0.0.0/0
aws ec2 authorize-security-group-egress --group-id sg-081e395d2ec65bd12 --protocol -1 --port all --cidr 0.0.0.0/0

Install eBPF Tools:

Install eBPF tools on EKS nodes. Used tools like bcc or bpftrace
sudo yum update -y
sudo yum install -y bcc bcc-tools python3-bcc

Develop eBPF Programs:

bpftrace is a high-level tracing language for Linux eBPF

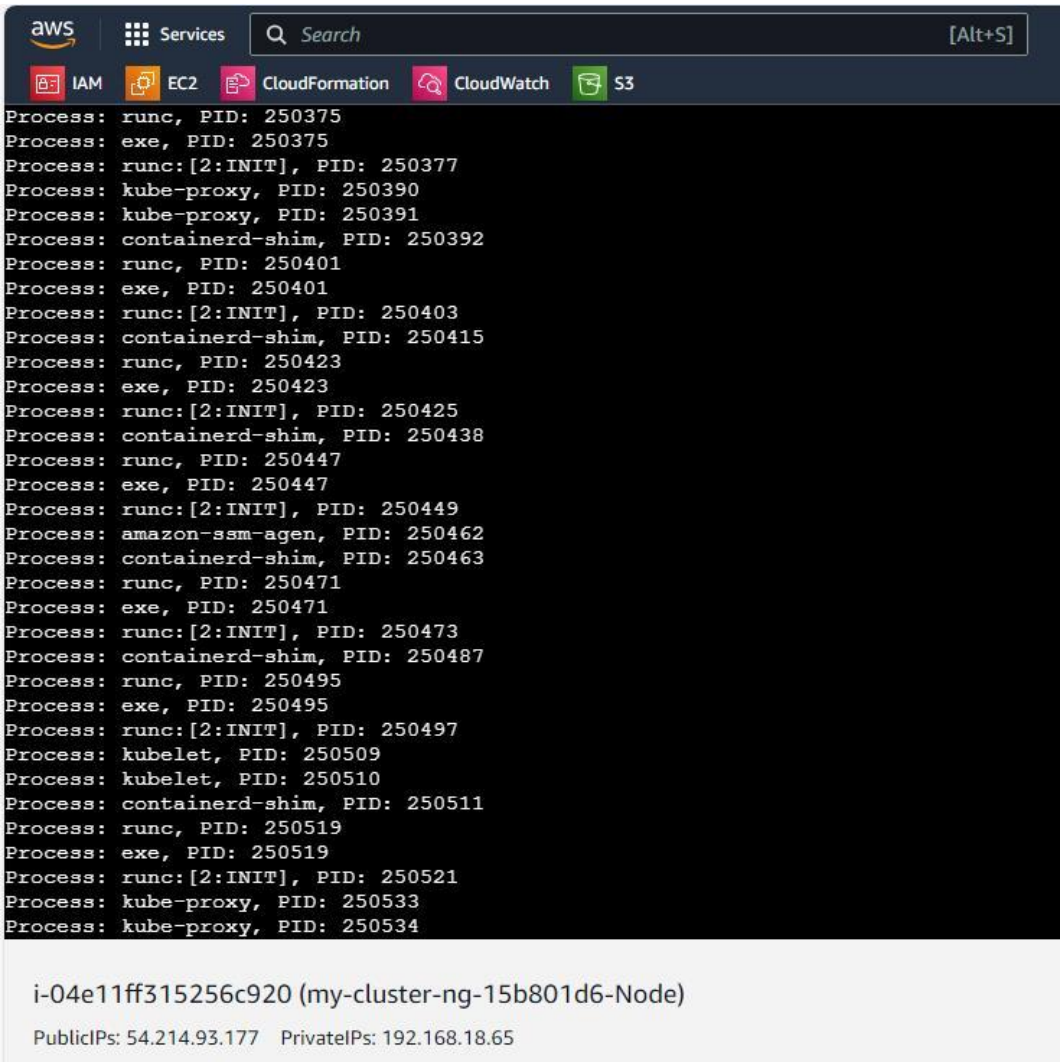
```
sudo yum update -y
sudo yum install -y bpftrace
```

eBPF programs to monitor container events and verification of image hashes in real-time below:

```
#!/usr/bin/env bpftrace

tracepoint:syscalls:sys_enter_execve
{
   printf("Process: %s, PID: %d\n", comm, pid);
}
```

```
sudo bpftrace process_monitor.bt
```



Implement Zero Trust Network Access Control:

Install and Configure Cilium:
Used tools like cilium which provides eBPF-based networking, security, and observability for Kubernetes.

helm repo add cilium https://helm.cilium.io/
helm install cilium cilium/cilium --version 1.10.4 --namespace kube-system
kubectl get pods -n kube-system -l k8s-app=cilium

```
root@developer2:/tmp# helm install cilium cilium/cilium --version 1.10.4 --namespace kube-system

W0830 06:08:42.014109    6753 warnings.go:70] spec.template.spec.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms[1].matchExpressions[0].key: beta.k
ubernetes.io/os is deprecated since v1.14; use "kubernetes.io/os" instead
W0830 06:08:42.014153    6753 warnings.go:70] spec.template.metadata.annotations[scheduler.alpha.kubernetes.io/critical-pod]: non-functional in v1.16+; use the "priorityClassName" field
instead
NAME: cilium
LAST DEPLOYED: Fri Aug 30 06:08:33 2024
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
You have successfully installed Cilium with Hubble.

Your release version is 1.10.4.

For any further help, visit https://docs.cilium.io/en/v1.10/gettinghelp
root@developer2:/tmp#
root@developer2:/tmp# kubectl get pods -n kube-system -l k8s-app=cilium
NAME           READY   STATUS            RESTARTS   AGE
cilium-pvwxj   0/1     PodInitializing   0          14s
cilium-vc2dm   0/1     Init:1/2          0          14s
cilium-vzjs7   0/1     Running           0          14s
root@developer2:/tmp#
root@developer2:/tmp#
root@developer2:/tmp# kubectl get pods -n kube-system -l k8s-app=cilium


NAME           READY   STATUS    RESTARTS   AGE
cilium-pvwxj   1/1     Running   0          18s
cilium-vc2dm   0/1     Running   0          18s
cilium-vzjs7   1/1     Running   0          18s
root@developer2:/tmp#
```

Deploy eBPF Programs:

Deployed eBPF programs as DaemonSets in EKS:

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: ebpf-verifier
spec:
  selector:
    matchLabels:
      name: ebpf-verifier
  template:
    metadata:
      labels:
        name: ebpf-verifier
    spec:
      containers:
      - name: ebpf-verifier
        image: nginx
        securityContext:
          privileged: true
```

```
root@developer2:/tmp# vi daemonset.yaml
root@developer2:/tmp#
root@developer2:/tmp#
root@developer2:/tmp#
root@developer2:/tmp# k create -f daemonset.yaml
daemonset.apps/ebpf-verifier created
root@developer2:/tmp#
root@developer2:/tmp#
root@developer2:/tmp# k get ds
NAME            DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
ebpf-verifier   3         3         0       3            0           <none>          5s
root@developer2:/tmp#
```

Create Network Policies:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-all
  namespace: default
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  - Egress
```

Integrate with Jenkins Pipeline:

```
NODE_NAME="ip-192-168-18-65.us-west-2.compute.internal"
INSTANCE_ID=$(kubectl get node $NODE_NAME -o jsonpath='{.spec.providerID}' | cut -d/ -f5)
aws ec2 describe-instances --instance-ids $INSTANCE_ID --query 'Reservations[*].Instances[*].PublicDnsName' --output text

pipeline {
   agent any

   environment {
      EKS_CLUSTER_NAME = 'my-cluster'
      EKS_NAMESPACE = 'jenkins'
      IMAGE_NAME = 'nginx'
      IMAGE_TAG = 'latest'
   }

    stages {

       stage('Build and Push Image') {
          steps {
             script {
                docker.build("${env.IMAGE_NAME}:${env.IMAGE_TAG}").push()
             }
          }
       }
```

```
        stage('Verify Image Hash') {
            steps {
                script {
                    def imageHash = sh(script: "docker inspect --format='{{index .RepoDigests 0}}'
${env.IMAGE_NAME}:${env.IMAGE_TAG}", returnStdout: true).trim()
                    // Call eBPF program to verify the image hash
                    sh "ebpf-verify-image-hash.sh ${imageHash}"
                }
            }
        }



        stage('Deploy to EKS') {
            steps {
                script {
                    withCredentials([string(credentialsId: 'aws-eks-kubeconfig', variable:
'KUBECONFIG')]) {
                        sh "kubectl apply -f k8s/deployment.yaml -n ${env.EKS_NAMESPACE}"
                        sh "kubectl apply -f k8s/service.yaml -n ${env.EKS_NAMESPACE}"
                    }
                }
            }
        }

         stage('Apply Network Policies') {
            steps {
                script {
                    withCredentials([string(credentialsId: 'aws-eks-kubeconfig', variable:
'KUBECONFIG')]) {
                        sh "kubectl apply -f k8s/network-policy.yaml -n ${env.EKS_NAMESPACE}"
                    }
                }
            }
        }
    }

     post {
        always {
            cleanWs()
        }
    }
}
```