1.Introduction

EKS Setup

Create EKS Cluster and node group with terraform

https://github.com/deesirouss/k8s-may-12

Install kubectl

Installing or updating kubectl - Amazon EKS

Kubernetes 1.25

```
curl -O
https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.7/2023-03-17/bin/l
inux/amd64/kubectl
```

Kubernetes 1.25

```
curl -O
https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.7/2023-03-17/bin/l
inux/amd64/kubectl.sha256
sha256sum -c kubectl.sha256
```

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

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export
PATH=$PATH:$HOME/bin
echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc

kubectl version --short --client
```

sudo apt update

sudo apt install awscli

aws eks --region <region> update-kubeconfig --name <eks-cluster-name>

Installing or updating eksctl

https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html

```
# for ARM systems, set ARCH to: `arm64`, `armv6` or `armv7`
ARCH=amd64
PLATFORM=$(uname -s)_$ARCH

curl -sLO
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$PLATFORM.tar.gz"

# (Optional) Verify checksum
curl -sL
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_checksums.txt" | grep
$PLATFORM | sha256sum --check

tar -xzf eksctl_$PLATFORM.tar.gz -C /tmp && rm eksctl_$PLATFORM.tar.gz

sudo mv /tmp/eksctl /usr/local/bin
```

2.Security Considerations

Internal TLS and encryption

Configuring Cilium CNI

Install Helm - https://helm.sh/docs/intro/install/

```
$ curl -fsSL -o get_helm.sh
https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
$ chmod 700 get_helm.sh
$ ./get_helm.sh
```

Configuring Cilium - https://docs.cilium.io/en/stable/,
https://docs.cilium.io/en/stable/installation/k8s-install-helm/#install-cilium

```
$ helm repo add cilium https://helm.cilium.io/
$ helm install cilium cilium/cilium --version 1.13.2 \
 --namespace kube-system \
 --set eni.enabled=true \
 --set ipam.mode=eni \
 --set egressMasqueradeInterfaces=eth0 \
 --set tunnel=disabled \
 --set nodeinit.enabled=true \
 --set encryption.enabled=true \
 --set encryption.type=wireguard \
 --set hubble.relay.enabled=true \
 --set hubble.listenAddress=":4244" \
 --set hubble.ui.enabled=true \
 --set l7Proxy=false \
 --set kubeProxyReplacement=strict \
 --set
k8sServiceHost=F016F90BF76075ACD9503A60D41F5749.gr7.us-east-1.eks.amazonaws.com \
 --set k8sServicePort=443 \
```

1. `--set eni.enabled=true`: This argument enables the use of Amazon Elastic Network Interfaces (ENIs) for pod networking.
2. `--set ipam.mode=eni`: This argument sets the IP address management (IPAM) mode to ENI, which enables Cilium to use ENIs for pod networking.

3. `--set egressMasqueradeInterfaces=eth0`: This argument specifies the network interface to be used for egress traffic masquerading. In this case, the interface is `eth0`.
4. `--set tunnel=disabled`: This argument disables the use of tunnels for network traffic between nodes.
5. `--set nodeinit.enabled=true`: This argument enables the Cilium Nodeinit daemonset, which sets up various network settings and creates a secure communication channel between nodes.
6. `--set encryption.enabled=true`: This argument enables the encryption of pod-to-pod traffic using WireGuard.
7. `--set encryption.type=wireguard`: This argument specifies the encryption protocol to be used for pod-to-pod traffic. In this case, the protocol is `WireGuard`.
8. `--set hubble.relay.enabled=true`: This argument enables the Hubble Relay, a component that collects network traffic data from all nodes in the cluster.
9. `--set hubble.listenAddress=":4244"`: This argument specifies the network address where the Hubble Relay listens for incoming connections. In this case, the address is `:4244`.
10. `--set hubble.ui.enabled=true`: This argument enables the Hubble UI, a web-based user interface for visualizing network traffic data.
11. `--set l7Proxy=false`: This argument disables the use of an L7 proxy for network traffic.
12. `--set kubeProxyReplacement=strict`: This argument enables strict enforcement of network policies at the application layer.
13. `--set k8sServiceHost=6075Agr7.us-east-2.eks.amazonaws.com`: This argument specifies the Kubernetes API server hostname.
14. `--set k8sServicePort=443`: This argument specifies the port number where the Kubernetes API server is listening. In this case, the port is `443`.

With successful installation, you will

```
 --set k8sServicePort=443 \
NAME: cilium
LAST DEPLOYED: Tue May 30 06:52:07 2023
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
You have successfully installed Cilium with Hubble Relay and Hubble

Your release version is 1.13.2.

For any further help, visit https://docs.cilium.io/en/v1.13/getting
ubuntu@ip-172-23-2-63:~$
```

Delete aws-node daemon set from namespace kube-system

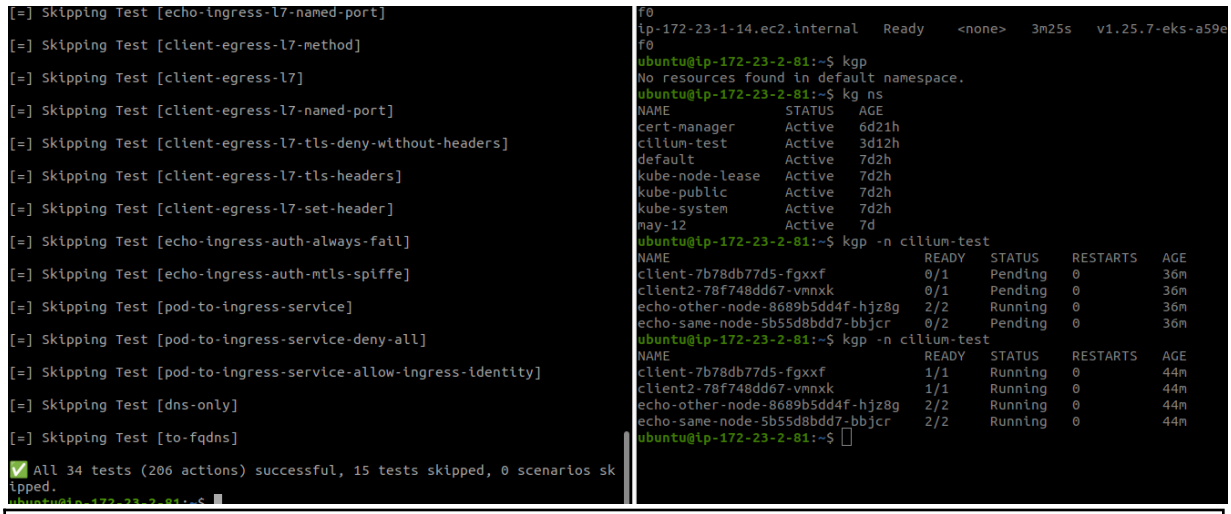
```
kubectl -n kube-system delete daemonset aws-node
```

Create node-groups now with terraform

Install Cilium CLI Tool
https://docs.cilium.io/en/stable/installation/k8s-install-helm/#install-cilium

```
CILIUM_CLI_VERSION=$(curl -s
https://raw.githubusercontent.com/cilium/cilium-cli/master/stable.txt)
CLI_ARCH=amd64
if [ "$(uname -m)" = "aarch64" ]; then CLI_ARCH=arm64; fi
curl -L --fail --remote-name-all
https://github.com/cilium/cilium-cli/releases/download/${CILIUM_CLI_VERSION}/cilium-linu
x-${CLI_ARCH}.tar.gz{,.sha256sum}
sha256sum --check cilium-linux-${CLI_ARCH}.tar.gz.sha256sum
sudo tar xzvfC cilium-linux-${CLI_ARCH}.tar.gz /usr/local/bin
rm cilium-linux-${CLI_ARCH}.tar.gz{,.sha256sum}
```

```
cilium status –wait
cilium connectivity test
```



communication flows between Cilium, eBPF, nodes, clusters, and pods:

1. Cilium Deployment: Cilium is deployed as a daemonset on each worker node in the EKS cluster. It runs as a Kubernetes agent responsible for managing networking and security policies.
2. eBPF Integration: Cilium utilizes eBPF programs to interact with the Linux kernel and perform efficient packet processing and filtering operations at the network layer.
3. Node-level Communication: Each node in the EKS cluster runs Cilium with eBPF, allowing it to intercept and handle network traffic at the kernel level. This includes processing packets, enforcing policies, and applying network-related operations.
4. Cluster-level Communication: Cilium enables secure communication between pods across nodes within the EKS cluster by leveraging eBPF. It implements a distributed firewall that enforces network policies, ensuring that only allowed traffic is allowed to flow between pods.
5. Pod Communication: When a pod within the EKS cluster sends or receives network traffic, Cilium, through eBPF, intercepts and processes the packets at the kernel level. It enforces network policies, performs network address translation (NAT), and applies other relevant operations before forwarding the packets to their

intended destinations.

Pod-level network and security rules

- Use network policy implementations like Calico or Cilium
- Define network policies at the pod level, specifying ingress and egress rules based on various criteria such as pod labels, namespaces, or IP addresses

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-app-traffic
spec:
  podSelector:
    matchLabels:
      app: my-app
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: my-app
    ports:
    - protocol: TCP
      port: 80
  egress:
  - to:
    - podSelector:
        matchLabels:
          app: my-app
```

```
    ports:
    - protocol: TCP
      port: 80
```

In this example, the Network Policy is named "allow-app-traffic" and applies to Pods with the label "app: my-app". The policy allows incoming traffic on port 80 from other Pods with the same label, and also allows outgoing traffic to other Pods with the same label.

Note that in order for this Network Policy to take effect, your Kubernetes cluster must have a networking plugin installed that supports Network Policies, such as Calico or Weave Net.

Documentation: https://kubernetes.io/docs/concepts/services-networking/network-policies/

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

  name: default-deny-ingress

spec:

  podSelector: {}

  policyTypes:

  - Ingress

RBAC to control access to resources

Admin access

kubectl edit -n kube-system configmap/aws-auth
mapUsers: |
  # map user IAM user Alice in 000000000000 to user "alice" in group "system:masters"
  - userarn: arn:aws:iam::000000000000:user/Alice
    username: alice
    groups:
    - system:masters

aws sts get-caller-identity

```
apiVersion: rbac.authorization.k8s.io/v1
```

```
kind: Role
metadata:
  name: deployment-role
  namespace: default
rules:
  - apiGroups:
          - ""
          - extensions
          - apps
          resources:
          - deployments
          - replicasets
          - pods
          verbs:
          - create
          - get
          - list
          - update
          - delete
          - watch
          - patch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: deployment-rolebinding
  namespace: default
roleRef:
  apiGroup: ""
  kind: Role
  name: deployment-role
subjects:
  - kind: User
          name: bibek
          apiGroup: ""
```

```
kubectl edit -n kube-system configmap/aws-auth
mapUsers: |
  # map user IAM user Alice in 000000000000 to user "alice" in group "system:masters"
  - userarn: arn:aws:iam::000000000000:user/Alice
    username: bibek
    groups:
    - deployment-role
```

Deployment restrictions to specific node groups

nodeName

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 containers:
 - name: nginx
   image: nginx
 nodeName: kube-01
```

Limitations

If the named node does not

- exist,the Pod will not run (automatically deleted).
- have the resources to accommodate the Pod, the Pod will fail and its reason will indicate why, for example OutOfmemory or OutOfcpu.
- Node names in cloud environments are not always predictable or stable.

Affinity and anti-affinity
https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/

`podAffinity` or `podAntiAffinity` with `preferredDuringSchedulingIgnoredDuringExecution` expresses a preferred preference,
allowing some flexibility in scheduling decisions.

On the other hand, combining them with `requiredDuringSchedulingIgnoredDuringExecution` indicates **a strict**

requirement, **enforcing the scheduling decision** based on affinity or anti-affinity rules.

```
Unset
apiVersion: v1
kind: Pod
metadata:
 name: with-pod-affinity
spec:
 affinity:
   podAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
     - labelSelector:
         matchExpressions:
         - key: security
           operator: In
           values:
           - S1
       topologyKey: topology.kubernetes.io/zone
   podAntiAffinity:
     preferredDuringSchedulingIgnoredDuringExecution:
     - weight: 100
       podAffinityTerm:
         labelSelector:
           matchExpressions:
           - key: security
             operator: In
             values:
             - S2
      topologyKey: topology.kubernetes.io/zone
 containers:
 - name: with-pod-affinity
   image: registry.k8s.io/pause:2.0
```

Node affinity

```
Unset

apiVersion: v1
```

```yaml
kind: Pod
metadata:
 name: with-node-affinity
spec:
 affinity:
   nodeAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
       nodeSelectorTerms:
       - matchExpressions:
         - key: topology.kubernetes.io/zone
           operator: In
           values:
           - antarctica-east1
           - antarctica-west1
     preferredDuringSchedulingIgnoredDuringExecution:
     - weight: 1
       preference:
         matchExpressions:
         - key: another-node-label-key
           operator: In
           values:
           - another-node-label-value
 containers:
 - name: with-node-affinity
   image: registry.k8s.io/pause:2.0
```

AWS EKS Managed node group https://docs.aws.amazon.com/eks/latest/userguide/managed-node-groups.html

Self Managed Node Group https://docs.aws.amazon.com/eks/latest/userguide/worker.html

```
Unset
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - rolearn: <YOUR_NODE_INSTANCE_ROLE_ARN>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

kubectl apply -f self-managed.yaml

3. Scalability

Cluster AutoScaler
https://aws.github.io/aws-eks-best-practices/cluster-autoscaling/
https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/cloudprovider/aws/README.md
https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler/cloudprovider/aws/examples

```
eksctl create iamserviceaccount --cluster=k8s-may-12-eks-staging
--namespace=kube-system \
--name=cluster-autoscaler
--attach-policy-arn=arn:aws:iam::aws:policy/AutoScalingFullAccess \
--override-existing-serviceaccounts --approve --region=us-east-1
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-autoscaler
```

```
  labels:
    k8s-addon: cluster-autoscaler.addons.k8s.io
    k8s-app: cluster-autoscaler
rules:
  - apiGroups: [""]
    resources: ["events", "endpoints"]
    verbs: ["create", "patch"]
  - apiGroups: [""]
    resources: ["pods/eviction"]
    verbs: ["create"]
  - apiGroups: [""]
    resources: ["pods/status"]
    verbs: ["update"]
  - apiGroups: [""]
    resources: ["endpoints"]
    resourceNames: ["cluster-autoscaler"]
    verbs: ["get", "update"]
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["watch", "list", "get", "update"]
  - apiGroups: [""]
    resources:
      - "namespaces"
      - "pods"
      - "services"
      - "replicationcontrollers"
      - "persistentvolumeclaims"
      - "persistentvolumes"
    verbs: ["watch", "list", "get"]
  - apiGroups: ["extensions"]
    resources: ["replicasets", "daemonsets"]
    verbs: ["watch", "list", "get"]
  - apiGroups: ["policy"]
    resources: ["poddisruptionbudgets"]
    verbs: ["watch", "list"]
  - apiGroups: ["apps"]
    resources: ["statefulsets", "replicasets", "daemonsets"]
    verbs: ["watch", "list", "get"]
```

```
  - apiGroups: ["storage.k8s.io"]
    resources: ["storageclasses", "csinodes", "csidrivers",
"csistoragecapacities"]
    verbs: ["watch", "list", "get"]
  - apiGroups: ["batch", "extensions"]
    resources: ["jobs"]
    verbs: ["get", "list", "watch", "patch"]
  - apiGroups: ["coordination.k8s.io"]
    resources: ["leases"]
    verbs: ["create"]
  - apiGroups: ["coordination.k8s.io"]
    resourceNames: ["cluster-autoscaler"]
    resources: ["leases"]
    verbs: ["get", "update"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: cluster-autoscaler
  namespace: kube-system
  labels:
    k8s-addon: cluster-autoscaler.addons.k8s.io
    k8s-app: cluster-autoscaler
rules:
  - apiGroups: [""]
    resources: ["configmaps"]
    verbs: ["create","list","watch"]
  - apiGroups: [""]
    resources: ["configmaps"]
    resourceNames: ["cluster-autoscaler-status",
"cluster-autoscaler-priority-expander"]
    verbs: ["delete", "get", "update", "watch"]

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cluster-autoscaler
```

```
  labels:
    k8s-addon: cluster-autoscaler.addons.k8s.io
    k8s-app: cluster-autoscaler
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-autoscaler
subjects:
  - kind: ServiceAccount
    name: cluster-autoscaler
    namespace: kube-system

---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: cluster-autoscaler
  namespace: kube-system
  labels:
    k8s-addon: cluster-autoscaler.addons.k8s.io
    k8s-app: cluster-autoscaler
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: cluster-autoscaler
subjects:
  - kind: ServiceAccount
    name: cluster-autoscaler
    namespace: kube-system

---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: cluster-autoscaler
  namespace: kube-system
  labels:
    app: cluster-autoscaler
```

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: cluster-autoscaler
  template:
    metadata:
      labels:
        app: cluster-autoscaler
      annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/port: '8085'
    spec:
      priorityClassName: system-cluster-critical
      securityContext:
        runAsNonRoot: true
        runAsUser: 65534
        fsGroup: 65534
      serviceAccountName: cluster-autoscaler
      containers:
        - image:
registry.k8s.io/autoscaling/cluster-autoscaler:v1.22.2
          name: cluster-autoscaler
          resources:
            limits:
              cpu: 100m
              memory: 600Mi
            requests:
              cpu: 100m
              memory: 600Mi
          command:
            - ./cluster-autoscaler
            - --v=4
            - --stderrthreshold=info
            - --cloud-provider=aws
            - --skip-nodes-with-local-storage=false
            - --expander=least-waste
```

```
            -
--node-group-auto-discovery=asg:tag=k8s.io/cluster-autoscaler/enabled,
k8s.io/cluster-autoscaler/evoke-eks-evoke
          volumeMounts:
            - name: ssl-certs
              mountPath: /etc/ssl/certs/ca-certificates.crt
#/etc/ssl/certs/ca-bundle.crt for Amazon Linux Worker Nodes
              readOnly: true
          imagePullPolicy: "Always"
      volumes:
        - name: ssl-certs
          hostPath:
            path: "/etc/ssl/certs/ca-bundle.crt"
```

Parameter	Description	Default
scan-interval	How often cluster is reevaluated for scale up or down	10 seconds
max-empty-bulk-delete	Maximum number of empty nodes that can be deleted at the same time.	10
scale-down-delay-after-add	How long after scale up that scale down evaluation resumes	10 minutes
scale-down-delay-after-delete	How long after node deletion that scale down evaluation resumes, defaults to scan-interval	scan-interval
scale-down-delay-after-failure	How long after scale down failure that scale down evaluation resumes	3 minutes

Pod Autoscaler

HPA
https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/
https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

Metric Server
https://github.com/kubernetes-sigs/metrics-server#deployment

`kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/high-availability-1.21+.yaml`
wget https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/high-availability-1.21+.yaml

Metrics Server offers:

- A single deployment that works on most clusters (see Requirements)
- Fast autoscaling, collecting metrics every 15 seconds.
- Resource efficiency, using 1 mili core of CPU and 2 MB of memory for each node in a cluster.
- Scalable support up to 5,000 node clusters.

Unset

```
kubectl apply -f https://k8s.io/examples/application/php-apache.yaml

kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
```

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: php-apache
  minReplicas: 1
```

```
  maxReplicas: 10

  targetCPUUtilizationPercentage: 50
```

```
# Run this in a separate terminal

# so that the load generation continues and you can carry on with the
rest of the steps

kubectl run -i --tty load-generator --rm --image=busybox:1.28
--restart=Never -- /bin/sh -c "while sleep 0.01; do wget -q -O-
http://php-apache; done"
```

1. Metric Collection Interval: The HPA collects metrics at regular intervals to assess the resource utilization of the pods. By default, **the metrics collection interval is 30 seconds**. However, you can configure a different interval by setting the `--horizontal-pod-autoscaler-sync-period` flag on the Kubernetes controller manager.
2. Scaling Stabilization Window: After each scaling action, the HPA waits for a stabilization window before making further scaling decisions. **The default stabilization window is 5 minutes (300 seconds).** During this window, the HPA observes the impact of the previous scaling action on the resource utilization and allows time for the new replicas to stabilize. This prevents rapid, unnecessary scaling actions in response to short-lived spikes in resource utilization.

kg hpa --watch

kg rs –watch

kgp –watch

4. ELK

https://www.shebanglabs.io/logging-with-efk-on-aws-eks/
https://www.elastic.co/guide/en/cloud-on-k8s/master/k8s-deploy-eck.html

If you have not done it before:

```
eksctl utils associate-iam-oidc-provider --region us-east-1 --cluster
```

k8s-may-12-eks-staging `--approve`

```
Create policy if you have not created:

aws iam create-policy --policy-name AWSLoadBalancerControllerIAMPolicy

--policy-document file://iam_policy.json
```

The policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAddresses",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeInstances",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeTags",
                "ec2:GetCoipPoolUsage",
                "ec2:DescribeCoipPools",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeLoadBalancerAttributes",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeListenerCertificates",
                "elasticloadbalancing:DescribeSSLPolicies",
                "elasticloadbalancing:DescribeRules",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetGroupAttributes",
                "elasticloadbalancing:DescribeTargetHealth",
                "elasticloadbalancing:DescribeTags"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cognito-idp:DescribeUserPoolClient",
                "acm:ListCertificates",
                "acm:DescribeCertificate",
                "iam:ListServerCertificates",
                "iam:GetServerCertificate",
```

```
                "waf-regional:GetWebACL",
                "waf-regional:GetWebACLForResource",
                "waf-regional:AssociateWebACL",
                "waf-regional:DisassociateWebACL",
                "wafv2:GetWebACL",
                "wafv2:GetWebACLForResource",
                "wafv2:AssociateWebACL",
                "wafv2:DisassociateWebACL",
                "shield:GetSubscriptionState",
                "shield:DescribeProtection",
                "shield:CreateProtection",
                "shield:DeleteProtection"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupIngress"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateSecurityGroup"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateSecurityGroup"
                },
                "Null": {
                    "aws:RequestTag/elbv2.k8s.aws/cluster": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags",
                "ec2:DeleteTags"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/elbv2.k8s.aws/cluster": "true",
                    "aws:ResourceTag/elbv2.k8s.aws/cluster": "false"
                }
            }
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:DeleteSecurityGroup"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/elbv2.k8s.aws/cluster": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateTargetGroup"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/elbv2.k8s.aws/cluster": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:DeleteListener",
                "elasticloadbalancing:CreateRule",
                "elasticloadbalancing:DeleteRule"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:RemoveTags"
            ],
            "Resource": [
                "arn:aws:elasticloadbalancing:*:*:targetgroup/*/*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/net/*/*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/app/*/*"
            ],
            "Condition": {
                "Null": {
                    "aws:RequestTag/elbv2.k8s.aws/cluster": "true",
                    "aws:ResourceTag/elbv2.k8s.aws/cluster": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:RemoveTags"
            ],
            "Resource": [
                "arn:aws:elasticloadbalancing:*:*:listener/net/*/*/*",
```

```
                "arn:aws:elasticloadbalancing:*:*:listener/app/*/*/*",
                "arn:aws:elasticloadbalancing:*:*:listener-rule/net/*/*/*",
                "arn:aws:elasticloadbalancing:*:*:listener-rule/app/*/*/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:ModifyLoadBalancerAttributes",
                "elasticloadbalancing:SetIpAddressType",
                "elasticloadbalancing:SetSecurityGroups",
                "elasticloadbalancing:SetSubnets",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:ModifyTargetGroup",
                "elasticloadbalancing:ModifyTargetGroupAttributes",
                "elasticloadbalancing:DeleteTargetGroup"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/elbv2.k8s.aws/cluster": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets"
            ],
            "Resource": "arn:aws:elasticloadbalancing:*:*:targetgroup/*/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:SetWebAcl",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:AddListenerCertificates",
                "elasticloadbalancing:RemoveListenerCertificates",
                "elasticloadbalancing:ModifyRule"
            ],
            "Resource": "*"
        }
    ]
}
```

Create SA

eksctl create iamserviceaccount --cluster k8s-may-12-eks-staging --namespace kube-system --name ebs-csi-controller-sa --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy --override-existing-serviceaccounts --approve --region us-east-1

Create a load balancer controller SA

helm repo add eks https://aws.github.io/eks-charts

helm repo update

kubectl apply -k "github.com/aws/eks-charts/stable/aws-load-balancer-controller/crds?ref=master"

helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n aws-alb --set clusterName=k8s-may-12-eks-staging --set serviceAccount.create=false --set serviceAccount.name=aws-load-balancer-controller --set region=us-east-1 --set vpcId=vpc-0a26e7b899f09b784

Get the latest version:

```
aws eks describe-addon-versions --addon-name aws-ebs-csi-driver --region us-east-1 | grep v1
```

Get the arn of the role:

kg sa ebs-csi-controller-sa -o yaml -n kube-system

Use the above arn in below command:

eksctl create addon --name aws-ebs-csi-driver --cluster k8s-may-12-eks-staging --service-account-role-arn arn:aws:iam::949263681218:role/eksctl-k8s-may-12-eks-staging-addon-iamservi-Role1-78BGKDW9RMP5 --region us-east-1 --force

If you want to update or get the information of addons
eksctl get addon --name aws-ebs-csi-driver --cluster k8s-may-12-eks-staging
eksctl update addon --name aws-ebs-csi-driver --version v1.11.4-eksbuild.1 --cluster k8s-may-12-eks-staging --force

Creating CIDR and necessary operators following the official documentation:
`kubectl create -f`

`https://download.elastic.co/downloads/eck/2.8.0/crds.yaml`

Install the operator with its RBAC rules:

```
kubectl apply -f

https://download.elastic.co/downloads/eck/2.8.0/operator.yaml
```

Optional:

1. **Monitor the operator logs:**
   `kubectl -n elastic-system logs -f`

```
statefulset.apps/elastic-operator
```

```
If there are other storage classes which are created by default, edit it and make it
false as default.

k edit sc

----------------------------------------

kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
  name: ebs-sc
provisioner: ebs.csi.aws.com
parameters:
  type: gp2
  encrypted: 'true'
volumeBindingMode: WaitForFirstConsumer
reclaimPolicy: Delete

kubectl apply -f storage_class.yaml
```

Using an Init Container to set virtual memory

To add an init container that changes the host kernel setting before your Elasticsearch container starts, you can use the following example Elasticsearch spec:

```
cat <<EOF | kubectl apply -f -
apiVersion: elasticsearch.k8s.elastic.co/v1
kind: Elasticsearch
metadata:
  name: quickstart
spec:
  version: 8.8.0
  nodeSets:
  - name: default
    count: 3
    podTemplate:
      spec:
        initContainers:
        - name: sysctl
          securityContext:
            privileged: true
            runAsUser: 0
          command: ['sh', '-c', 'sysctl -w
vm.max_map_count=262144']
```

```
EOF
```

Monitor cluster health and creation progress

kubectl get elasticsearch

One Pod is in the process of being started:

```
kubectl get pods
--selector='elasticsearch.k8s.elastic.co/cluster-name=quickstar
t'
```

Request Elasticsearch access

A ClusterIP Service is automatically created for your cluster:

```
kubectl get service quickstart-es-http
```

Username is elastic

And password is :

```
kubectl get secret quickstart-es-elastic-user
-o=jsonpath='{.data.elastic}' | base64 --decode; echo
```

Access the elasticsearch:

kubectl port-forward service/quickstart-es-http --address 0.0.0.0 9200:9200

Logging in to the elastic search

kg secret

PASSWORD=$(kubectl get secret elasticsearch-es-elastic-user -o go-template='{{.data.elastic | base64decode}}')

echo $PASSWORD

Browse the network load balancer DNS on port 9200
username is : elastic
Password : the one you extracted above.

Kibana

```
cat <<EOF | kubectl apply -f -
apiVersion: kibana.k8s.elastic.co/v1
kind: Kibana
metadata:
  name: kibana
spec:
  version: 8.8.0
  count: 1
  elasticsearchRef:
    name: elasticsearch
EOF
```

```
kubectl get kibana
```

```
kubectl get pod
--selector='kibana.k8s.elastic.co/name=quickstart'
```

kubectl port-forward service/quickstart-kb-http --address 0.0.0.0 5601:5601

Login with same username and password as elasticsearch

FluentD

Config file

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

```
metadata:
  name: fluentd-config
  labels:
    app: fluentd
data:
  fluent.conf: |
    <match fluent.**>
        # this tells fluentd to not output its log on stdout
        @type null
    </match>
    # here we read the logs from Docker's containers and parse them
    <source>
      @type tail
      path /var/log/containers/*.log
      pos_file /var/log/containers.log.pos
      tag kubernetes.*
      read_from_head true
      <parse>
        @type "#{ENV['FLUENT_CONTAINER_TAIL_PARSER_TYPE'] || 'json'}"
      </parse>
    </source>
    # we use kubernetes metadata plugin to add metadatas to the log
    <filter kubernetes.**>
        @type kubernetes_metadata
    </filter>
    <match kubernetes.var.log.containers.**kube-logging**.log>
    @type null
    </match>
    # <match kubernetes.var.log.containers.**kube-system**.log>
    # @type null
    # </match>
    <match kubernetes.var.log.containers.**monitoring**.log>
    @type null
    </match>
     # we send the logs to Elasticsearch
    <match kubernetes.**>
      @type elasticsearch_dynamic
      @log_level info
      include_tag_key true
      host "#{ENV['FLUENT_ELASTICSEARCH_HOST']}"
      port "#{ENV['FLUENT_ELASTICSEARCH_PORT']}"
      user "#{ENV['FLUENT_ELASTICSEARCH_USER']}"
      password "#{ENV['FLUENT_ELASTICSEARCH_PASSWORD']}"
      scheme "#{ENV['FLUENT_ELASTICSEARCH_SCHEME'] || 'http'}"
      ssl_verify "#{ENV['FLUENT_ELASTICSEARCH_SSL_VERIFY'] || 'true'}"
      reload_connections true
      logstash_format true
      logstash_prefix ${record['kubernetes']['pod_name']}
      <buffer>
```

```
      @type file
      path /var/log/fluentd-buffers/kubernetes.system.buffer
      flush_mode interval
      retry_type exponential_backoff
      flush_thread_count 2
      flush_interval 5s
      retry_forever true
      retry_max_interval 30
      chunk_limit_size 2M
      queue_limit_length 32
      overflow_action block
    </buffer>
  </match>
```

Fluentd.yaml

```
kind: ServiceAccount
metadata:
  name: fluentd
  labels:
    app: fluentd
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: fluentd
  labels:
    app: fluentd
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - namespaces
  verbs:
  - get
  - list
  - watch
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: fluentd
roleRef:
  kind: ClusterRole
  name: fluentd
  apiGroup: rbac.authorization.k8s.io
```

```
subjects:
- kind: ServiceAccount
  name: fluentd
  namespace: default
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd
  labels:
    app: fluentd
spec:
  selector:
    matchLabels:
      app: fluentd
  template:
    metadata:
      labels:
        app: fluentd
    spec:
      serviceAccount: fluentd
      serviceAccountName: fluentd
      tolerations:
      - key: node-role.kubernetes.io/master
        effect: NoSchedule
      containers:
      - name: fluentd
        image: fluent/fluentd-kubernetes-daemonset:v1.15.1-debian-elasticsearch7-1.1
        env:
          - name:  FLUENT_ELASTICSEARCH_HOST
            value: "54.81.170.149"
          - name:  FLUENT_ELASTICSEARCH_PORT
            value: "9200"
          - name: FLUENT_ELASTICSEARCH_SCHEME
            value: "https"
          - name: FLUENTD_SYSTEMD_CONF
            value: disable
          - name: FLUENT_ELASTICSEARCH_SSL_VERIFY
            value: "false"
          - name: FLUENT_ELASTICSEARCH_SSL_VERSION
            value: "TLSv1_2"
          - name: FLUENT_ELASTICSEARCH_USER
            value: "elastic"
          - name: FLUENT_ELASTICSEARCH_PASSWORD
            value: "Eph4t656WR1C72T4Fm4aJr2f"
          - name: FLUENT_CONTAINER_TAIL_EXCLUDE_PATH
            value: /var/log/containers/fluent*
          - name: FLUENT_CONTAINER_TAIL_PARSER_TYPE
            value: /^(?<time>.+) (?<stream>stdout|stderr)( (?<logtag>.))? (?<log>.*)$/
```

```
      resources:
        limits:
          memory: 512Mi
        requests:
          cpu: 100m
          memory: 200Mi
      volumeMounts:
      - name: fluentd-config
        mountPath: /fluentd/etc
      - name: varlog
        mountPath: /var/log
      - name: varlibdockercontainers
        mountPath: /var/lib/docker/containers
        readOnly: true
    terminationGracePeriodSeconds: 30
    volumes:
    - name: fluentd-config
      configMap:
        name: fluentd-config
    - name: varlog
      hostPath:
        path: /var/log
    - name: varlibdockercontainers
      hostPath:
        path: /var/lib/docker/containers
```

Login to kibana and manage index pattern, logs everything 😄

Prometeous-grafama

```
## Define public Kubernetes chart repository in the Helm configuration
```
helm repo add prometheus-community
https://prometheus-community.github.io/helm-charts
```
## Update local repositories
```
helm repo update
```
## Search for newly installed repositories
```
helm repo list
```
## Create a namespace for Prometheus and Grafana resources
```
kubectl create ns prometheus
```
```

Install Prometheus using HELM
```
helm install prometheus prometheus-community/kube-prometheus-stack -n prometheus
```
Check all resources in Prometheus Namespace
```
kubectl get all -n prometheus
```
check helm manifest
```
helm get manifest prometheus -n prometheus
```
Port forward the Prometheus service
```
kubectl port-forward -n prometheus svc/prometheus-operated --address 0.0.0.0
9090:9090
```
Get the Username
```
kubectl get secret -n prometheus prometheus-grafana -o=jsonpath='{.data.admin-user}'
|base64 -d
```
Get the Password
```
kubectl get secret -n prometheus prometheus-grafana
-o=jsonpath='{.data.admin-password}' |base64 -d
```
Port forward the Grafana service
```
kubectl port-forward -n prometheus svc/prometheus-grafana 3000:80
```

To add smtp details:

k edit deploy prometheus-grafana -n prometheus

Add environment variables:

- name: GF_SMTP_ENABLED

  value: "true"

- name: GF_SMTP_HOST

  value: email-smtp.us-east-1.amazonaws.com:465

- name: GF_SMTP_PORT

  value: "465"

- name: GF_SMTP_USER

  value: AKIA52BEGI3BHWJS5O5K

- name: GF_SMTP_PASSWORD

  value: BKPRipcZY2V1/oQwzZZzCK9zKqkzyDqeHUgPm3hQ8+45

- name: GF_SMTP_FROM_ADDRESS

  value: bibekmishra@lftechnology.com

- name: GF_SMTP_FROM_NAME

```
          value: admin
```

cilium status

```
export HUBBLE_VERSION=$(curl -s
https://raw.githubusercontent.com/cilium/hubble/master/stable.txt)
HUBBLE_ARCH=amd64
if [ "$(uname -m)" = "aarch64" ]; then HUBBLE_ARCH=arm64; fi
curl -L --fail --remote-name-all
https://github.com/cilium/hubble/releases/download/$HUBBLE_VERSION/hubble-linux-${HUBB
LE_ARCH}.tar.gz{,.sha256sum}
sha256sum --check hubble-linux-${HUBBLE_ARCH}.tar.gz.sha256sum
sudo tar xzvfC hubble-linux-${HUBBLE_ARCH}.tar.gz /usr/local/bin
rm hubble-linux-${HUBBLE_ARCH}.tar.gz{,.sha256sum}
```

hubble status

hubble observe

cilium hubble ui

kg svc -n kube-system

k port-forward --address 0.0.0.0 svc/hubble-ui -n kube-system 12000:80

Roughs

To give access to non-AWS User to view pods

To give a non-AWS IAM user access to view pods in an AWS EKS cluster, you can use the Kubernetes Role-Based Access Control (RBAC) mechanism to create a user and grant appropriate permissions. Here are the steps to achieve this:
 	
1. Create a Kubernetes ServiceAccount: Create a YAML manifest file, let's name it `user-view-pods.yaml`, with the following content:
 	
yaml
`apiVersion: v1`
`kind: ServiceAccount`
`metadata:`
`  name: non-aws-user`
 	
Apply the manifest using the `kubectl apply` command:
 	
sql
`kubectl apply -f user-view-pods.yaml`
 	
2. Create a ClusterRole: Create another YAML manifest file, let's name it `user-view-pods-role.yaml`, with the following content:
 	
yaml
`apiVersion: rbac.authorization.k8s.io/v1`
`kind: ClusterRole`
`metadata:`
`  name: view-pods-role`

```
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "list", "watch"]
```

Apply the manifest using the `kubectl apply` command:

sql
```
kubectl apply -f user-view-pods-role.yaml
```

3. Bind the ServiceAccount and ClusterRole: Create a YAML manifest file, let's name it `user-view-pods-binding.yaml`, with the following content:

yaml
```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: view-pods-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: view-pods-role
subjects:
- kind: ServiceAccount
  name: non-aws-user
  namespace: default
```

Apply the manifest using the `kubectl apply` command:

sql
```
kubectl apply -f user-view-pods-binding.yaml
```

4. Obtain the ServiceAccount Token: Get the token associated with the ServiceAccount using the following command:

sql
```
kubectl get secret $(kubectl get sa non-aws-user -o
jsonpath='{.secrets[0].name}') -o jsonpath='{.data.token}' | base64
--decode
```

Make a note of the token value. This will be used to authenticate the non-AWS IAM user.

5. Provide the token to the non-AWS IAM user: Share the token with the non-AWS IAM user, along with the cluster endpoint URL. The user can use these credentials to access the EKS cluster and view pods using the `kubectl` command-line tool or other Kubernetes clients.

Note: Providing a non-AWS IAM user access to an EKS cluster introduces potential security risks. Ensure you carefully manage the user's permissions and revoke access when it's no longer needed.

By following these steps, you can grant a non-AWS IAM user the ability to view pods in an AWS EKS cluster using RBAC.

Certainly! Here's an example of a config file that you can provide to the non-AWS IAM user, including the token:

yaml

```
apiVersion: v1
clusters:
- cluster:
    server: <CLUSTER_ENDPOINT_URL>
    certificate-authority-data: <CLUSTER_CA_CERTIFICATE>
  name: my-eks-cluster
contexts:
- context:
    cluster: my-eks-cluster
    user: non-aws-user
  name: my-eks-context
current-context: my-eks-context
kind: Config
preferences: {}
users:
- name: non-aws-user
  user:
    token: <TOKEN_VALUE>
```

Instructions:

1. Replace `<CLUSTER_ENDPOINT_URL>` with the actual endpoint URL of your EKS cluster. This should be the value you share with the user.
2. Replace `<CLUSTER_CA_CERTIFICATE>` with the CA certificate data for your EKS cluster. You can obtain this by running the following command:
   css

```
kubectl config view --raw --minify --flatten -o
jsonpath='{.clusters[].cluster.certificate-authority-data}' | base64
--decode
```

1.
2. Replace `<TOKEN_VALUE>` with the actual token value obtained in Step 4.

Provide this YAML config file to the non-AWS IAM user. They can use it with the `kubectl` command-line tool or other Kubernetes clients by setting the `KUBECONFIG` environment variable to point to this file.

Example command to set the `KUBECONFIG` environment variable:

javascript
```
export KUBECONFIG=/path/to/eks-config.yaml
```

Please ensure that you have securely shared the config file and token with the user, as this grants them access to your EKS cluster.

To create NLB Load Balancer of ElasticSearch Service.

kg svc elasticsearch-es-http -o yaml > elastic_svc.yaml

Edit the elastic_svc.yaml and format it according to this example for converting this service into Network Loadbalacer

```
# 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
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags:
name=eks-access,creator=bibekmishra,project=may-12
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: instance
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
    service.beta.kubernetes.io/aws-load-balancer-subnets:
subnet-0723147f090d3f9fa,subnet-05b82cb6119dc7373
    service.beta.kubernetes.io/aws-load-balancer-type: external
  labels:
    app: wazuh-manager
  name: wazuh
  namespace: wazuh
spec:
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: registration
    port: 1515
```

```
    protocol: TCP
    targetPort: 1515
  - name: api
    port: 55000
    protocol: TCP
    targetPort: 55000
  selector:
    app: wazuh-manager
    node-type: master
  sessionAffinity: None
  type: LoadBalancer
  loadBalancerClass: service.k8s.aws/nlb
status:
  loadBalancer: {}
```

Add the annotation and update it according to your requirements, tags and name.