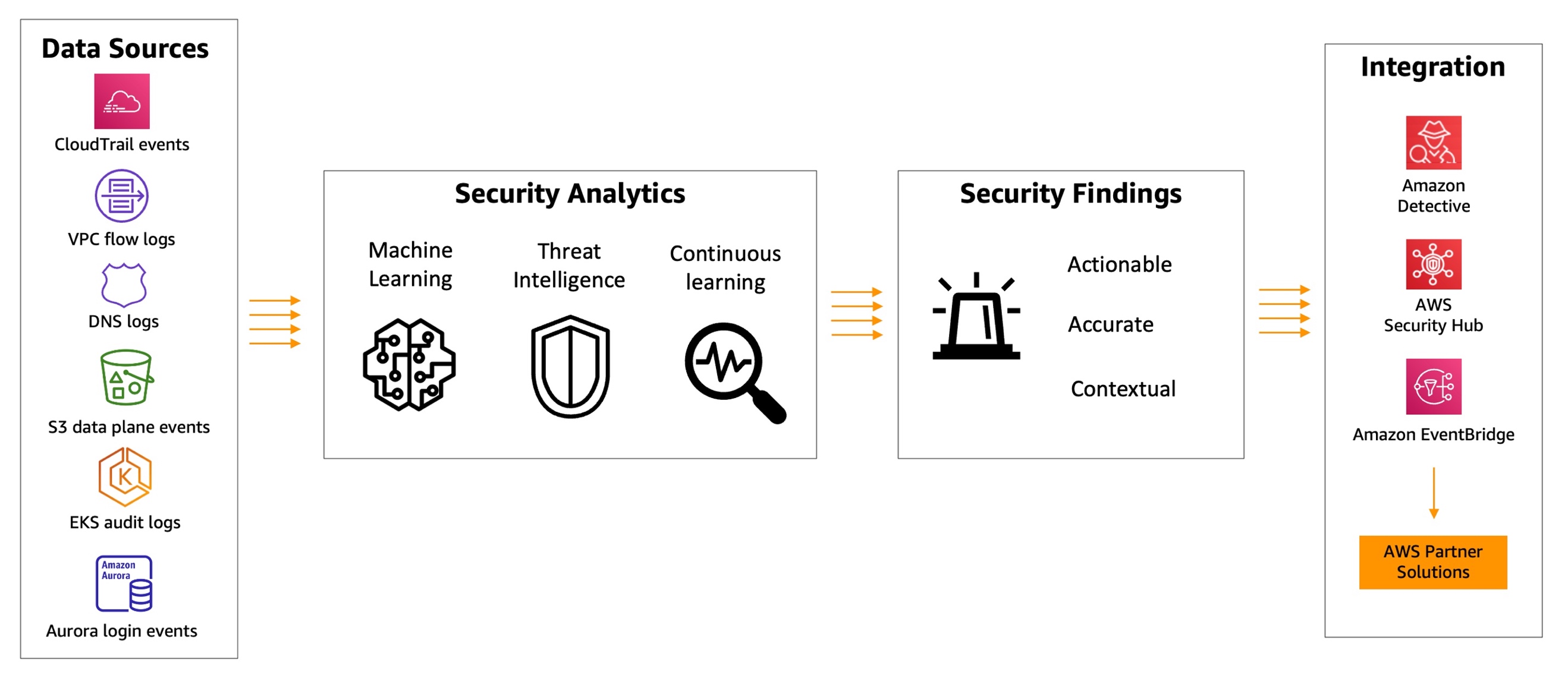
Amazon GuardDuty Protection for Amazon EKS

[Amazon GuardDuty](https://aws.amazon.com/guardduty/) has expanded coverage to continuously monitor and profile Amazon Elastic Kubernetes Service (Amazon EKS) cluster activity to identify malicious or suspicious behavior that represents potential threats to container workloads. EKS Protection in Amazon GuardDuty provides threat detection coverage to help you protect Amazon EKS clusters within your AWS environment. EKS Protection includes [EKS Audit Log Monitoring](https://aws.amazon.com/about-aws/whats-new/2022/01/amazon-guardduty-elastic-kubernetes-service-clusters/) and [EKS Runtime Monitoring](https://aws.amazon.com/blogs/aws/amazon-guardduty-now-supports-amazon-eks-runtime-monitoring/).

**EKS Audit Log Monitoring** : EKS Audit Log Monitoring helps you detect potentially suspicious activities in EKS clusters within Amazon Elastic Kubernetes Service (Amazon EKS). EKS Audit Log Monitoring uses Kubernetes audit logs to capture chronological activities from users, applications using the Kubernetes API, and the control plane. For more information, see [Kubernetes audit logs](https://docs.aws.amazon.com/guardduty/latest/ug/features-kubernetes-protection.html#guardduty_k8s-audit-logs).

**EKS Runtime Monitoring**: EKS Runtime Monitoring uses operating system-level events to help you detect potential threats in Amazon EKS nodes and containers within your Amazon EKS clusters. For more information, see [Runtime Monitoring](https://docs.aws.amazon.com/guardduty/latest/ug/features-kubernetes-protection.html#guardduty_runtime-monitoring).



Enable Amazon GuardDuty Protection for Amazon EKS

**Note**

Note that below steps are for configuring EKS Audit Log Monitoring for a standalone account. To configure EKS Audit Log Monitoring in multiple-account environments,you need to use delegated administrator account to manage their member accounts using AWS Organizations. Please refer to the [documentation](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty-eks-audit-log-monitoring.html)

* Using AWS CLI
* Using AWS Console

Run the following command to enable Amazon GuardDuty and then also enable EKS Protection for both EKS Audit Log Monitoring and EKS Runtime Monitoring.

Create a configuration file to enable the Amazon EKS protection features.

cd ~/environment

cat > guardduty-eks-protection-config.json <<EOF

[

{

"Name": "EKS\_AUDIT\_LOGS",

"Status": "ENABLED",

"Name": "EKS\_RUNTIME\_MONITORING",

"Status": "ENABLED",

"AdditionalConfiguration": [

{

"Name": "EKS\_ADDON\_MANAGEMENT",

"Status": "ENABLED"

}

]

}

]

EOF

Run the below command to enable EKS Protection for Amazon GuardDuty.

1

2

GUARDDUTY\_DETECTOR\_ID=$(aws guardduty create-detector --enable --features file://guardduty-eks-protection-config.json | jq -r '.DetectorId')

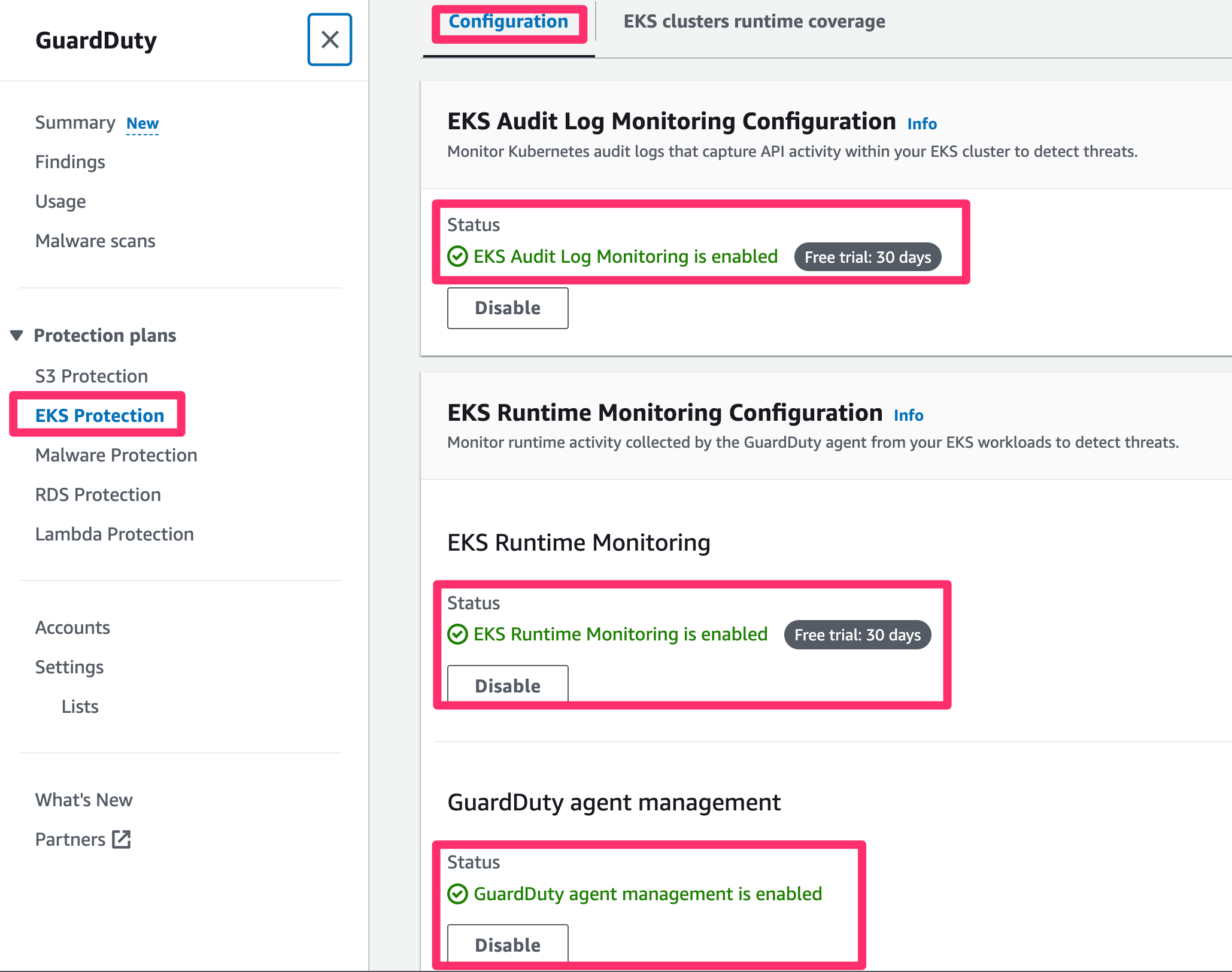
echo $GUARDDUTY\_DETECTOR\_ID

The output will look like below

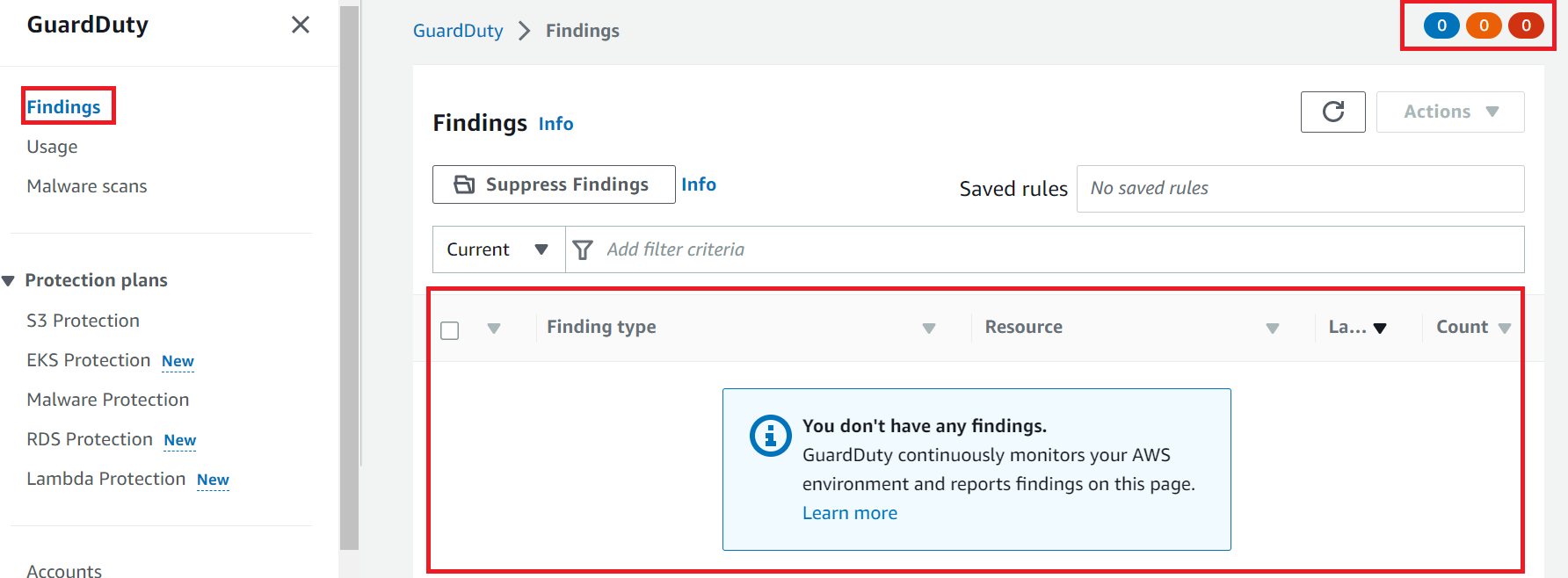
1

b6b992d6d2f48e64bc59180bfexample

After EKS Protection in Amazon GuardDuty is enabled, it looks like below in the [AWS GuardDuty Console](https://us-west-2.console.aws.amazon.com/guardduty/home?region=us-west-2#/k8s-protection).

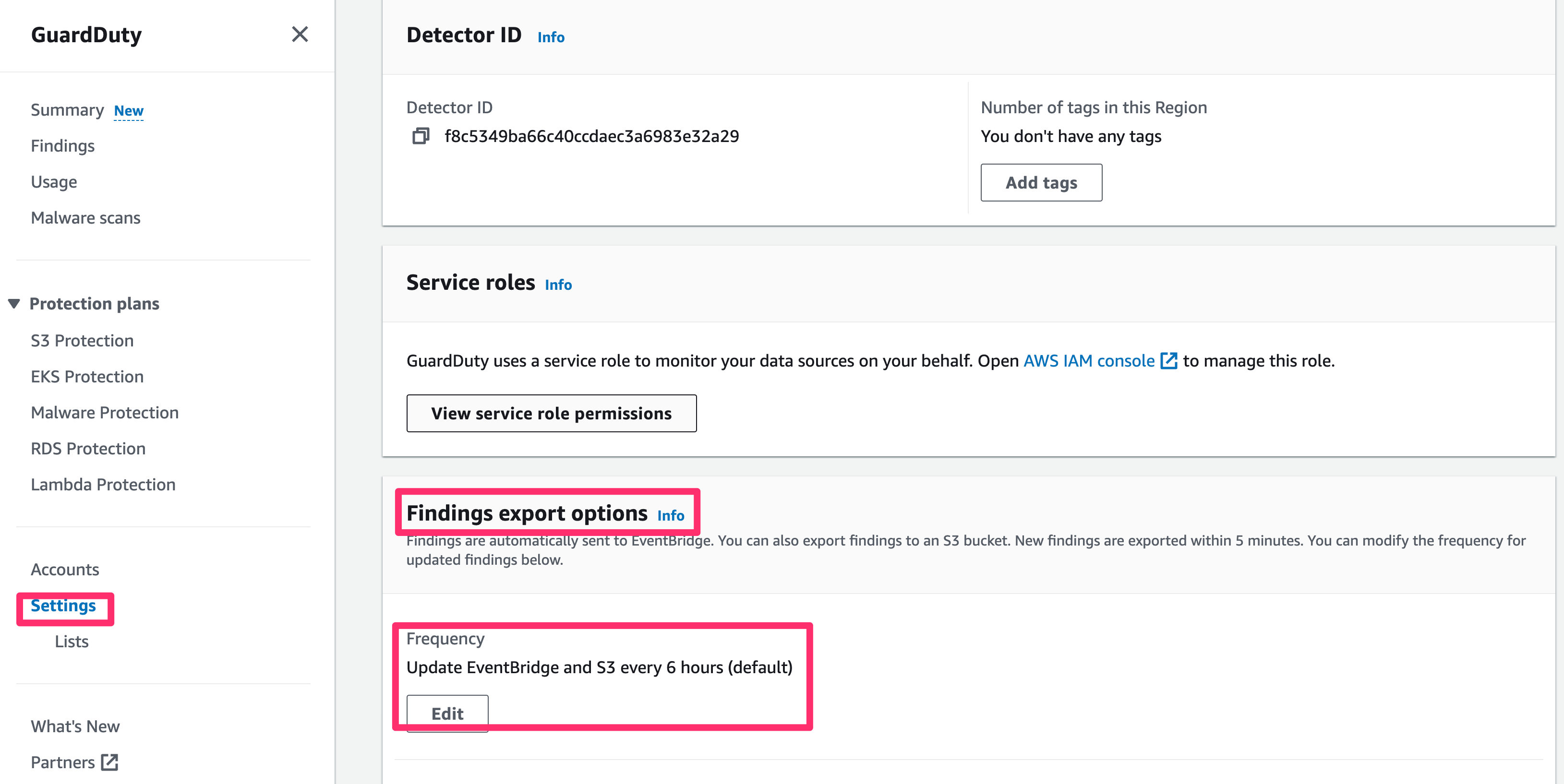


Go to Findings. You should see there are no findings available yet.

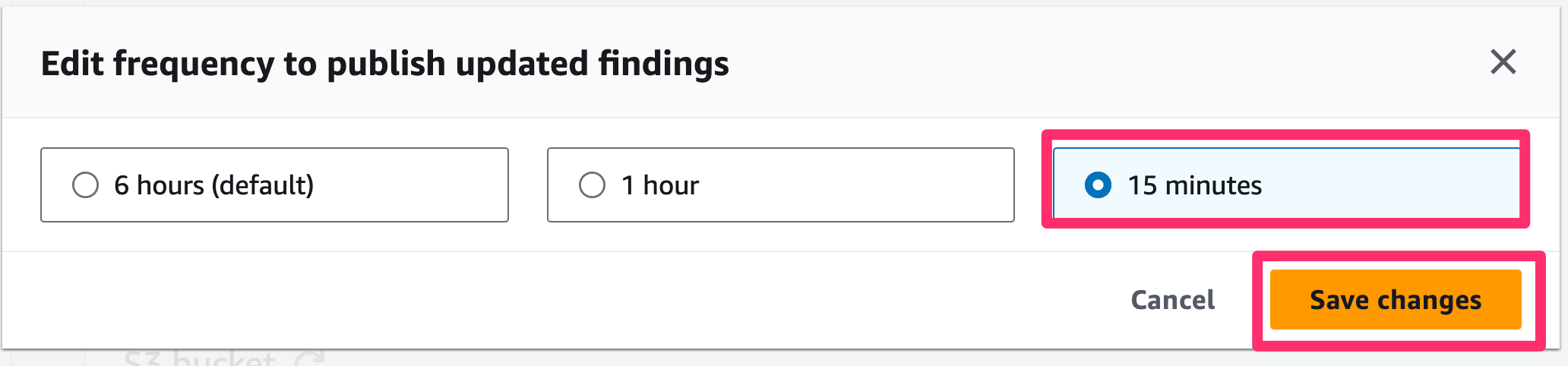


GuardDuty Findings are automatically sent to EventBridge. You can also export findings to an S3 bucket. New findings are exported within 5 minutes. You can modify the frequency for updated findings below. Update to EventBridge and S3 occurs every 6 hours by default. Let us change it to 15 mins.

Go to the **Settings** --> **Findings export options** and Click on the Edit.



Select **15 minutes** and Click on **Save Changes**.



Findings export options

With Amazon GuardDuty already turned on with protection for your EKS clusters, you are now ready to see it in action. GuardDuty for EKS does not require you to turn on or store EKS Control Plane logs. GuardDuty can look at the EKS cluster audit logs through direct integration. It will look at the audit log activity and report on the new GuardDuty finding types that are specific to your Kubernetes resources.

EKS Audit Log Monitoring

**WARNING**

Before starting this module, ensure you have completed the section [Enable Amazon GuardDuty Protection for Amazon EKS](https://catalog.us-east-1.prod.workshops.aws/workshops/165b0729-2791-4452-8920-53b734419050/en-US/5-detective-controls/1-guardduty-protection-for-eks/enable-guardduty/)

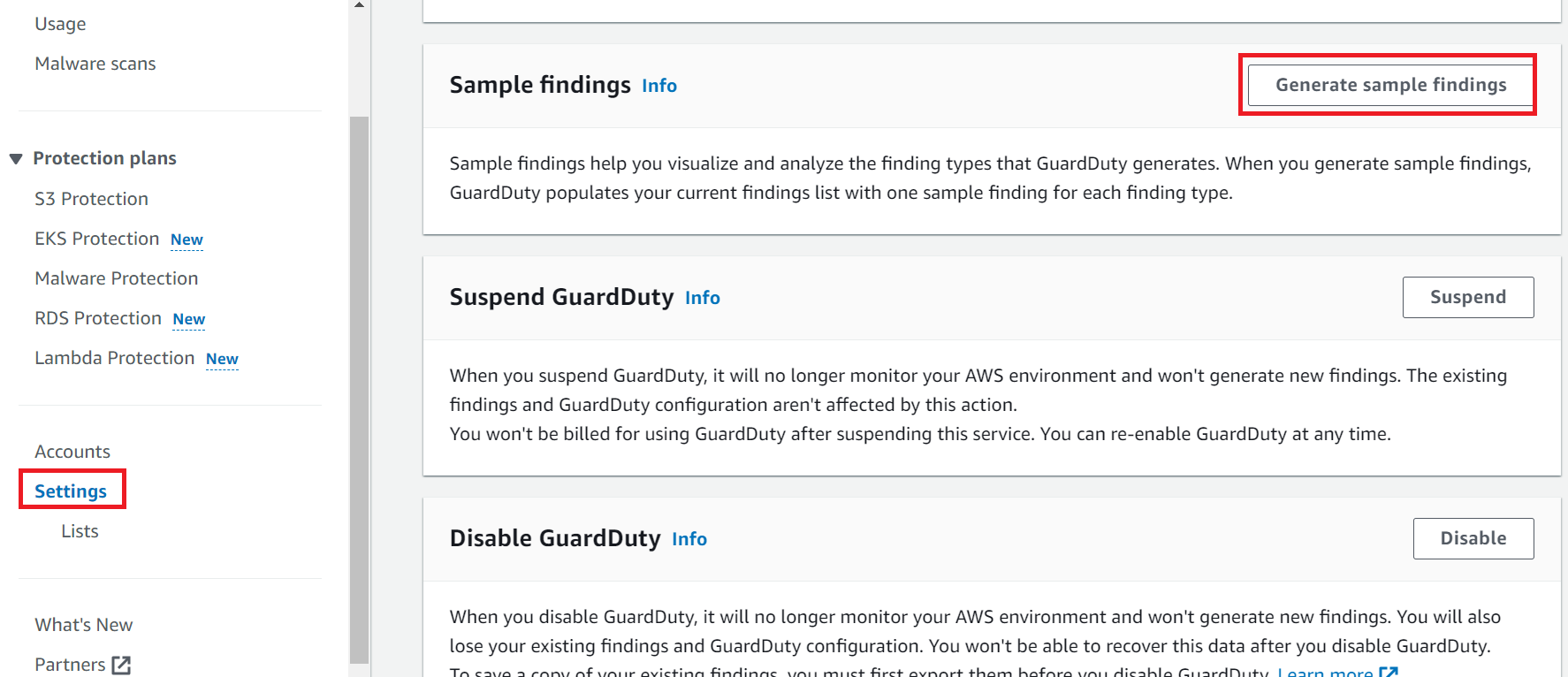
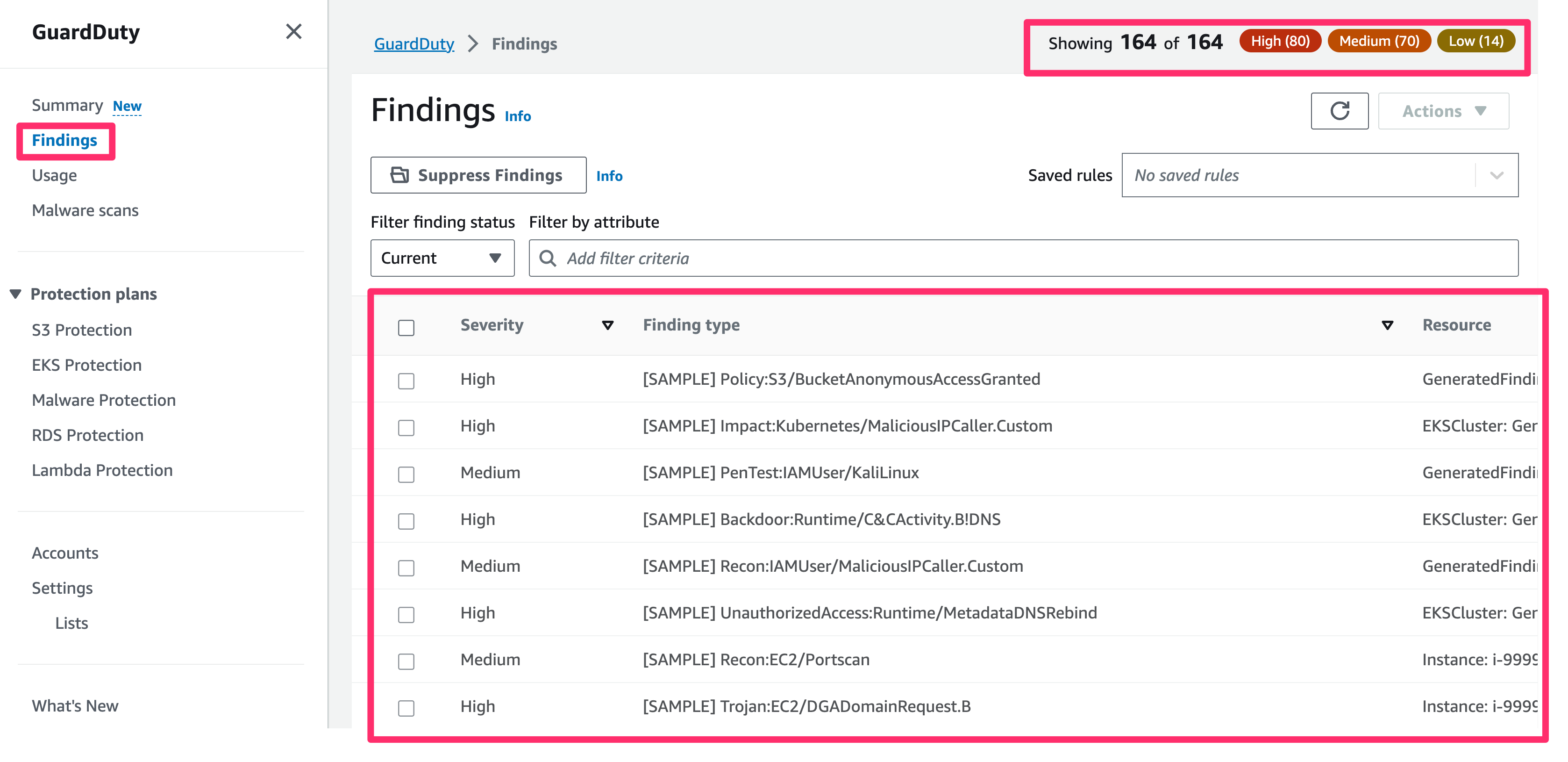
In this module, we will generate sample findings in GuardDuty for EKS Audit Log Monitoring and view them in Amazon GuardDuty Console. We will create custom responses to GuardDuty findings with Amazon CloudWatch Events.

Generating sample findings through the GuardDuty console or API

Choose an access method to learn how to generate sample findings through that method.

* Using AWS Console
* Using AWS CLI

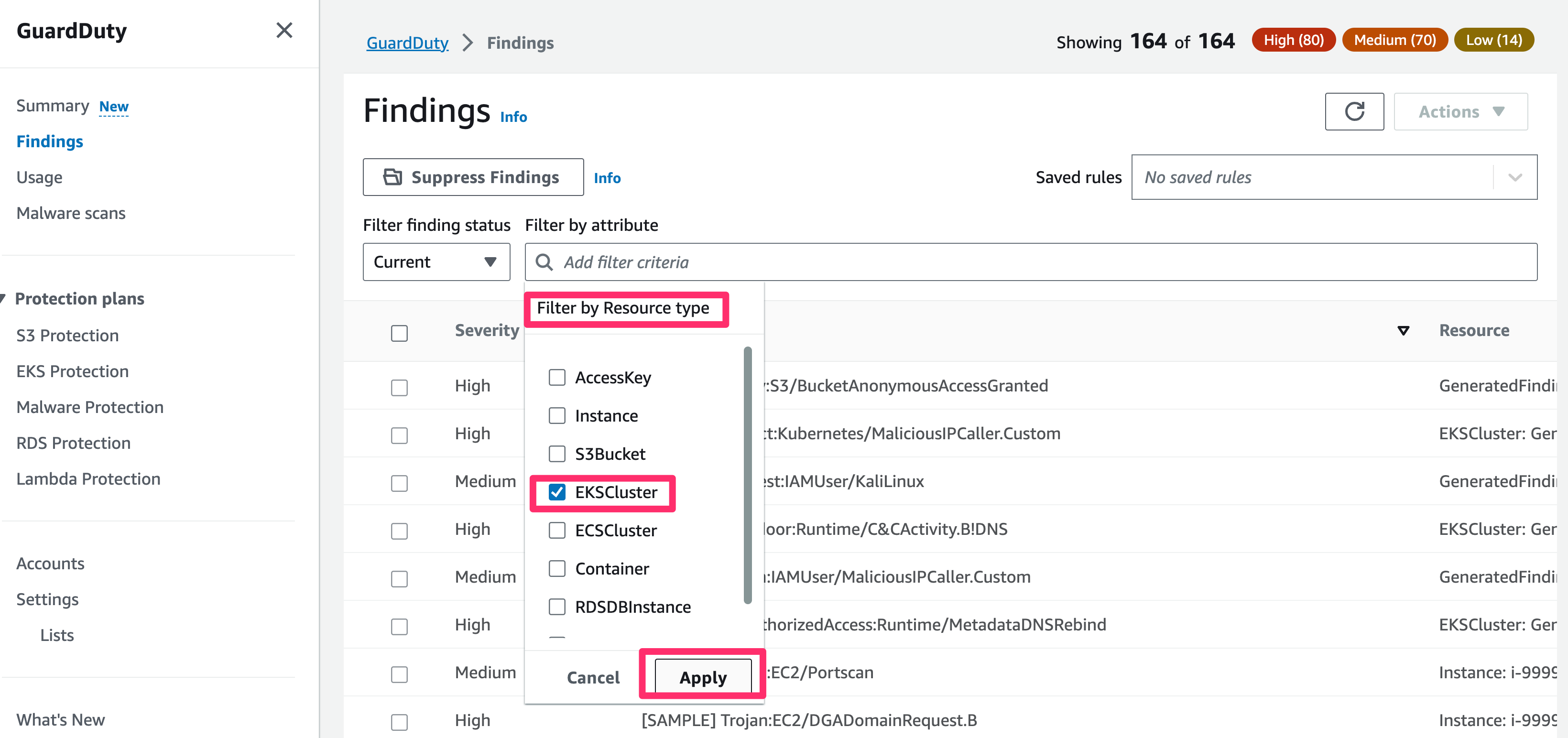
Use the following procedure to generate sample findings. This process generates one sample finding for each GuardDuty finding type.

1. Open the [GuardDuty console](https://console.aws.amazon.com/guardduty/" \t "_blank)
2. In the navigation pane, choose **Settings**.
3. On the **Settings** page, under **Sample findings**, choose **Generate sample findings**. 
4. In the navigation pane, choose **Findings**. The sample findings are displayed on the **Current findings** page with the prefix **[SAMPLE]**. 

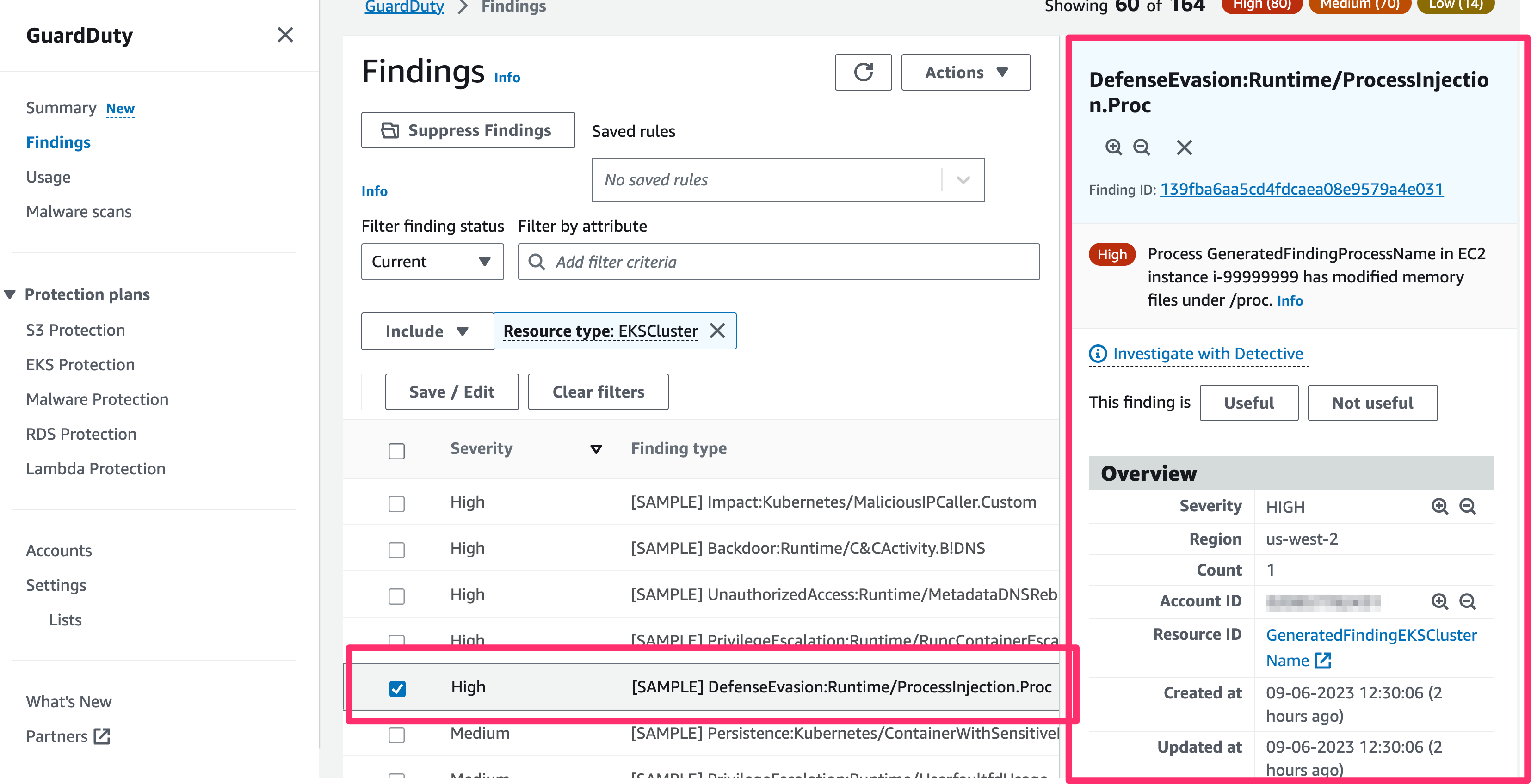
**Note**

Note the above step generate Sample Findings for all the resource types. Let us apply some filters to see Findings for Amazon EKS

1. In the filters list, select key **Resource Type**, select **EKS Cluster** and Click **Apply**



1. Select one of the sample Findings and click on it. You will see Finding Summary details on the right side.



# Generate Kubernetes Findings using Kubectl in Cloud9 Instance

In this section, we will generate some Kubernetes findings in your Amazon EKS cluster using your Cloud9 instance.

Go to your Cloud9 terminal and run the following commands to generate the sample findings.

### [**Policy:Kubernetes/AnonymousAccessGranted**](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-kubernetes.html#policy-kubernetes-anonymousaccessgranted)

This finding means **The**system:anonymous**user was granted API permission on a Kubernetes cluster.**

Run the following in your terminal to create the YAML manifest that has a ClusterRole and a ClusterRoleBinding definition.

cd ~/environment

cat << EoF > anonymous.yaml

### Finding type: Policy:Kubernetes/AnonymousAccessGranted

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: anonymous-admin

subjects:

- kind: User

name: system:anonymous

namespace: default

roleRef:

kind: ClusterRole

name: view

apiGroup: rbac.authorization.k8s.io

EoF

As you can see, we are granting the user system:anonymous with access to the view ClusterRole. This will allow an anonymous user to view all objects in your cluster using the kubernetes API. This is generally an unexpected configuration and should be reviewed. Run kubectl apply to apply this configuration. You can see more details fo this finding [here](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-kubernetes.html#policy-kubernetes-anonymousaccessgranted)

kubectl apply -f anonymous.yaml

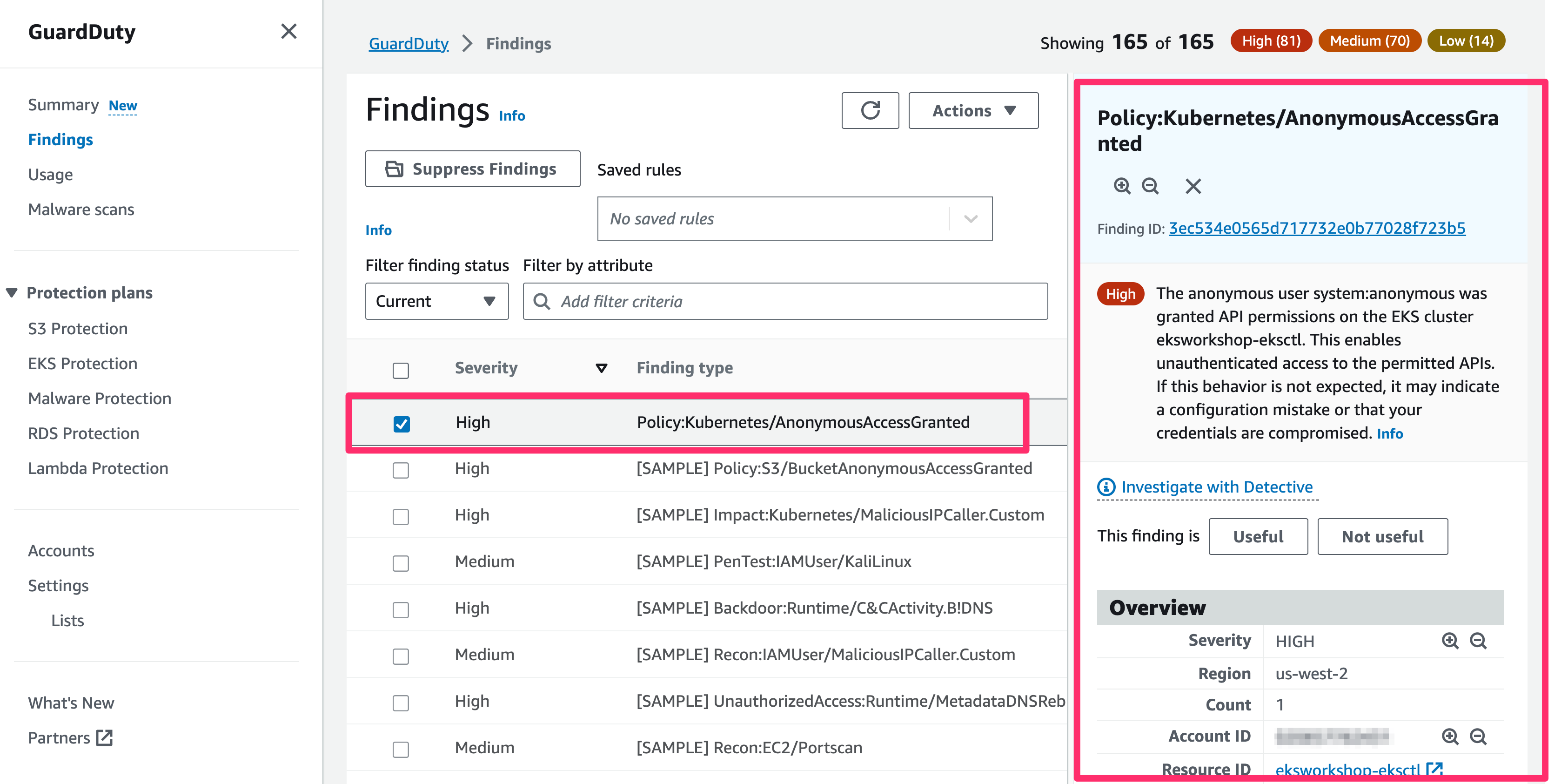
Check Output

clusterrolebinding.rbac.authorization.k8s.io/anonymous-admin created

Go back [AWS GuardDuty console](javascript:void(0)) and check that a finding is generated for this.

**Note**

If the finding doesn’t appear immediateley in the GuardDuty Console, try changing the name under metadata (ex: **anonymous-admin2**) in the anonymous.yaml file and re-run the kubectl apply -f anonymous.yaml. Also keep refreshing the page since it make take few minutes to to generate the Kubernetes Findings



### [**Policy:Kubernetes/AdminAccessToDefaultServiceAccount**](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-kubernetes.html#policy-kubernetes-adminaccesstodefaultserviceaccount)

This finding means **The default service account was granted admin privileges on a Kubernetes cluster.**

From your terminal, run the command below to create the YAML manifest for the finding.

cd ~/environment

cat << EoF > elevate.yaml

### Finding type: Policy:Kubernetes/AdminAccessToDefaultServiceAccount

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: default-service-acct-admin

subjects:

- kind: ServiceAccount

name: default

namespace: default

roleRef:

kind: ClusterRole

name: cluster-admin

apiGroup: rbac.authorization.k8s.io

EoF

Here you can see, we have a ClusterRole and a ClusterRoleBinding definition that is binding the default service account in the default namespace to the cluster-admin ClusterRole. Any pod not associated to a specific service account make use of the default service account in their namespace. This configuration can provide unintentional elevated admin privileges to pods. More information about this [here](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-kubernetes.html#policy-kubernetes-adminaccesstodefaultserviceaccount) . Run kubectl apply to apply this configuration.

kubectl apply -f elevate.yaml

Check Output

clusterrolebinding.rbac.authorization.k8s.io/default-service-acct-admin created

Go back AWS GuardDuty console and check that a finding is generated for this.



### [**PrivilegeEscalation:Kubernetes/PrivilegedContainer** and **Persistence:Kubernetes/ContainerWithSensitiveMount**](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-kubernetes.html)

These findings means **A privileged container with root level access was launched on your Kubernetes cluster.** and **A container was launched with a sensitive external host path mounted inside.**

cd ~/environment

cat << EoF > pod\_with\_sensitive\_mount.yaml

### Finding type: PrivilegeEscalation:Kubernetes/PrivilegedContainer

### Finding type: Kubernetes/ContainerWithSensitiveMount Incident

apiVersion: apps/v1

kind: Deployment

metadata:

name: ubuntu-privileged-with-mount

spec:

selector:

matchLabels:

app: ubuntu-privileged-with-mount

replicas: 1

template:

metadata:

labels:

app: ubuntu-privileged-with-mount

spec:

containers:

- name: ubuntu-privileged-with-mount

image: nginx

securityContext:

privileged: true

volumeMounts:

- mountPath: /test-pd

name: test-volume

volumes:

- name: test-volume

hostPath:

path: /etc

type: Directory

EoF

This yaml file generates 2 findings using a single Kubernetes deployment spec. The first is related to privileged container with root level access. This is accomplished through the privileged: true setting in the securityContext configuration of the nginx container spec. Secondly, the container mounts /etc directory on the host as a writable volume. More information about the two findings here - [sensitive mount](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-kubernetes.html#persistence-kubernetes-containerwithsensitivemount) and [privileged container](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-kubernetes.html#privilegeescalation-kubernetes-privilegedcontainer)

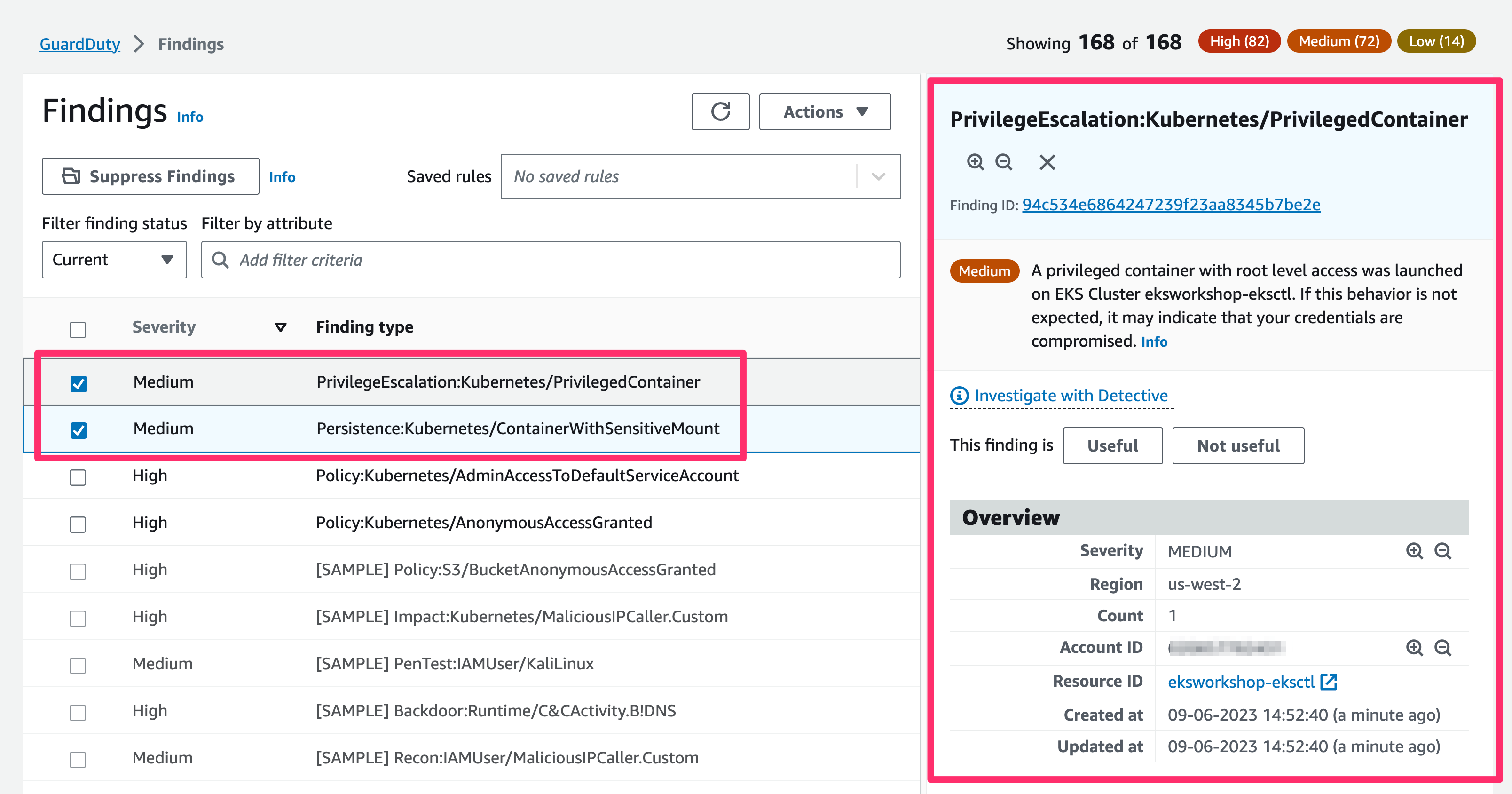
Run kubectl apply to apply this configuration.

kubectl apply -f pod\_with\_sensitive\_mount.yaml

Check Output

deployment.apps/ubuntu-privileged-with-mount created

Go to AWS GuardDuty Console to check the findings.



Let's take a moment to review findings' detail. Click on each finding in the GuardDuty console to open its detail.

As an example, click PolicyKubernetes/AdminAccessToDefaultServiceAccount finding. In the finding details, examine the Action section.

Click the body of Parameters to determine when the finding was generated, what kind of API object was the target for the API call.

### [**Policy:Kubernetes/ExposedDashboard**](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-kubernetes.html#persistence-kubernetes-containerwithsensitivemount)

This finding means **The dashboard for a Kubernetes cluster was exposed to the internet**

This finding informs you that Kubernetes dashboard for your cluster was exposed to the internet by a Load Balancer service. An exposed dashboard makes the management interface of your cluster accessible from the internet and allows adversaries to exploit any authentication and access control gaps that may be present.

To simulate this we'll need to expose the Kubernetes dashboard to the Internet with service type LoadBalancer.

First off, we'll install the Kubernetes dashboard component. We'll be using the version v2.7.0 of the dashboard, which is compatible with Kubernetes cluster v1.25 based on the [release notes](https://github.com/kubernetes/dashboard/releases/tag/v2.7.0).

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml

**Check Output**

namespace/kubernetes-dashboard created

serviceaccount/kubernetes-dashboard created

service/kubernetes-dashboard created

secret/kubernetes-dashboard-certs created

secret/kubernetes-dashboard-csrf created

secret/kubernetes-dashboard-key-holder created

configmap/kubernetes-dashboard-settings created

role.rbac.authorization.k8s.io/kubernetes-dashboard created

clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created

rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created

clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created

deployment.apps/kubernetes-dashboard created

service/dashboard-metrics-scraper created

deployment.apps/dashboard-metrics-scraper created

Let us patch the kubernetes-dashboard service to be type LoadBalancer.

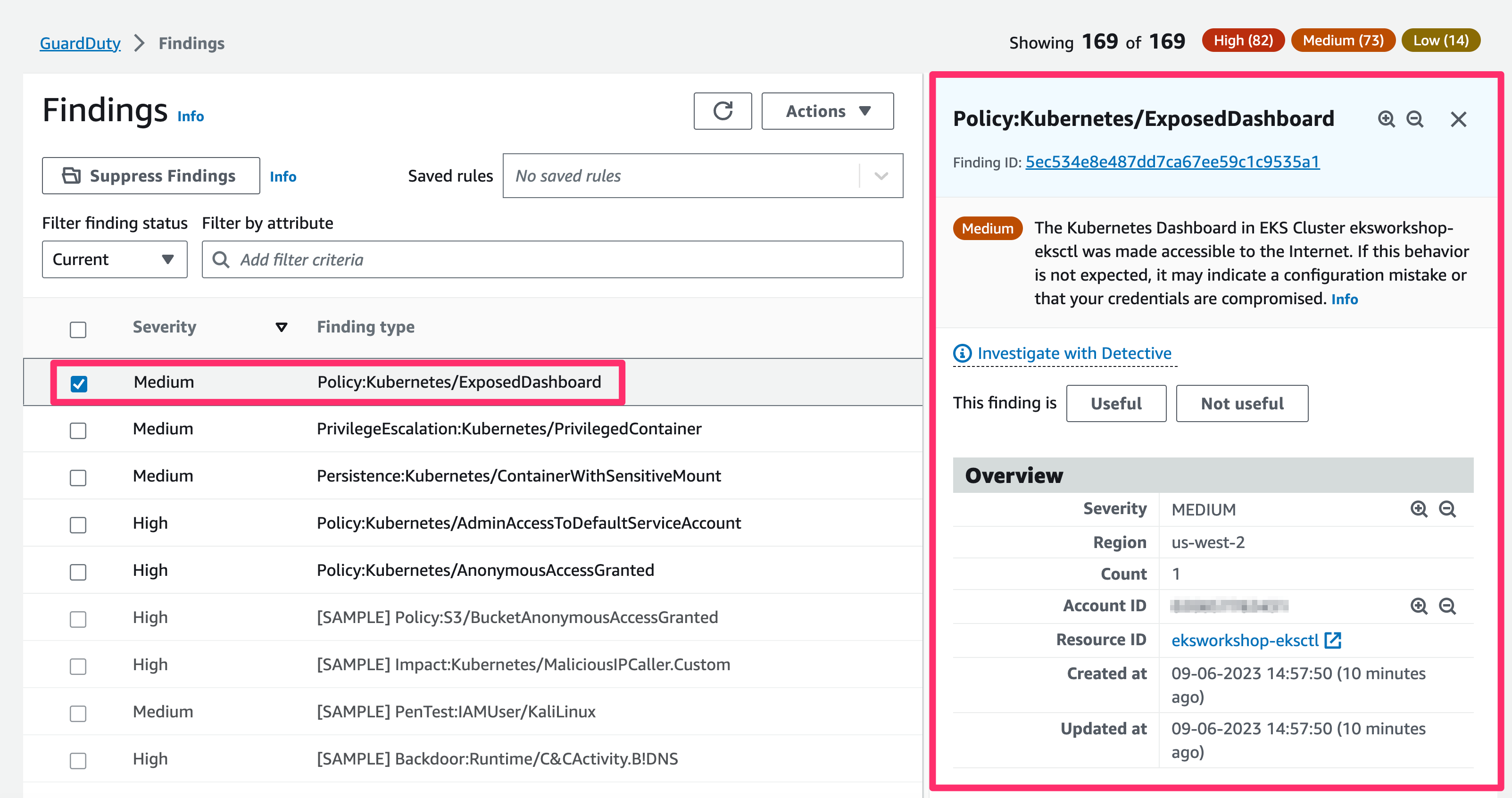
1

kubectl patch svc kubernetes-dashboard -n kubernetes-dashboard -p='{"spec": {"type": "LoadBalancer"}}'

Check Output

1

service/kubernetes-dashboard patched



Within a few minutes we'll see the finding Policy:Kubernetes/ExposedDashboard in the GuardDuty portal.

# Setup EventBridge and SNS

In this section we will look at how to be notified for any findings detected by Amazon GuardDuty in your EKS cluster. To do this, we will create an Amazon EventBridge rule to filter specific Amazon GuardDuty Kubernetes events and send messages to an Amazon SNS topic, which has an email subscription to get an automated email notification.

We will start by creating the Amazon SNS topic and subscribe it with your email address.

* Using AWS CLI
* Using AWS Console

#### [Create a SNS Topic](https://catalog.us-east-1.prod.workshops.aws/workshops/165b0729-2791-4452-8920-53b734419050/en-US/5-detective-controls/1-guardduty-protection-for-eks/eks-audit-logs/setup-eventbridge-and-sns" \l "create-a-sns-topic)

Run the following command to create a SNS Topic

SNS\_TOPIC\_ARN=$(aws sns create-topic --name K8sAudit | jq -r '.TopicArn')

echo $SNS\_TOPIC\_ARN

The output will look like below

arn:aws:sns:us-west-2:XXXXXXXX:K8sAudit

#### [Create a Subscription for the SNS Topic](https://catalog.us-east-1.prod.workshops.aws/workshops/165b0729-2791-4452-8920-53b734419050/en-US/5-detective-controls/1-guardduty-protection-for-eks/eks-audit-logs/setup-eventbridge-and-sns" \l "create-a-subscription-for-the-sns-topic)

Set below environment variable to your email id

1

export MY\_EMAIL\_ID=<my-email@example.com>

Run below command to create a SNS Subscription for the above topic

aws sns subscribe \

--topic-arn $SNS\_TOPIC\_ARN \

--protocol email \

--notification-endpoint $MY\_EMAIL\_ID

The output looks like below

{

"SubscriptionArn": "pending confirmation"

}

**Note**

**Check your email you have specified above and confirm the subscription to the SNS topic.**

#### [Create an Amazon EventBridge rule](https://catalog.us-east-1.prod.workshops.aws/workshops/165b0729-2791-4452-8920-53b734419050/en-US/5-detective-controls/1-guardduty-protection-for-eks/eks-audit-logs/setup-eventbridge-and-sns" \l "create-an-amazon-eventbridge-rule)

Run below command to create an EventBridge Event Rule with event pattern set to Amazon GuardDuty Findings.

aws events put-rule \

--name EKSAuditRoute \

--event-pattern \

'{

"source": ["aws.guardduty"],

"detail-type": ["GuardDuty Finding"]

}

'

The output looks like below

{

"RuleArn": "arn:aws:events:us-west-2:XXXXXXXX:rule/EKSAuditRoute"

}

Then run below command to update the above Rule to attach a target for the above SNS topic.

aws events put-targets --rule EKSAuditRoute --targets "Id"="1","Arn"="$SNS\_TOPIC\_ARN"

The output looks like below

{

"FailedEntryCount": 0,

"FailedEntries": []

}

Finally, EventBridge must be granted permissions to publish messages to SNS. Run the following command to update the resource policy for the SNS topic:

aws sns set-topic-attributes --topic-arn $SNS\_TOPIC\_ARN \

--attribute-name Policy \

--attribute-value "{\"Version\":\"2012-10-17\",\"Id\":\"\_\_default\_policy\_ID\",\"Statement\":[{\"Sid\":\"\_\_default\_statement\_ID\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"\*\"},\"Action\":[\"SNS:GetTopicAttributes\",\"SNS:SetTopicAttributes\",\"SNS:AddPermission\",\"SNS:RemovePermission\",\"SNS:DeleteTopic\",\"SNS:Subscribe\",\"SNS:ListSubscriptionsByTopic\",\"SNS:Publish\"],\"Resource\":\"$SNS\_TOPIC\_ARN\"}, {\"Sid\":\"PublishEventsToMyTopic\",\"Effect\":\"Allow\",\"Principal\":{\"Service\":\"events.amazonaws.com\"},\"Action\":\"sns:Publish\",\"Resource\":\"$SNS\_TOPIC\_ARN\"}]}"

# Test Incident Notification

Now, let’s generate another GuardDuty finding in your Cloud9 terminal to check if the EventBridge rule routes events from GuardDuty to the target SNS topic, and sends an email notification to you.

#### [**Execution:Kubernetes/ExecInKubeSystemPod**](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-kubernetes.html#execution-kubernetes-execinkubesystempod)

This time we will generate a new finding related to a command being executed inside a pod within the kube-system namespace. More information about this [here](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-kubernetes.html#execution-kubernetes-execinkubesystempod).

Run the below commands to generate this finding. Note, the exact pod name varies in the second command. Use the pod name you see for the aws-node pod as displayed in the output of the first command.

AWS\_NODE\_POD=`kubectl get pods -n kube-system -l k8s-app=aws-node -o name | head -n 1`

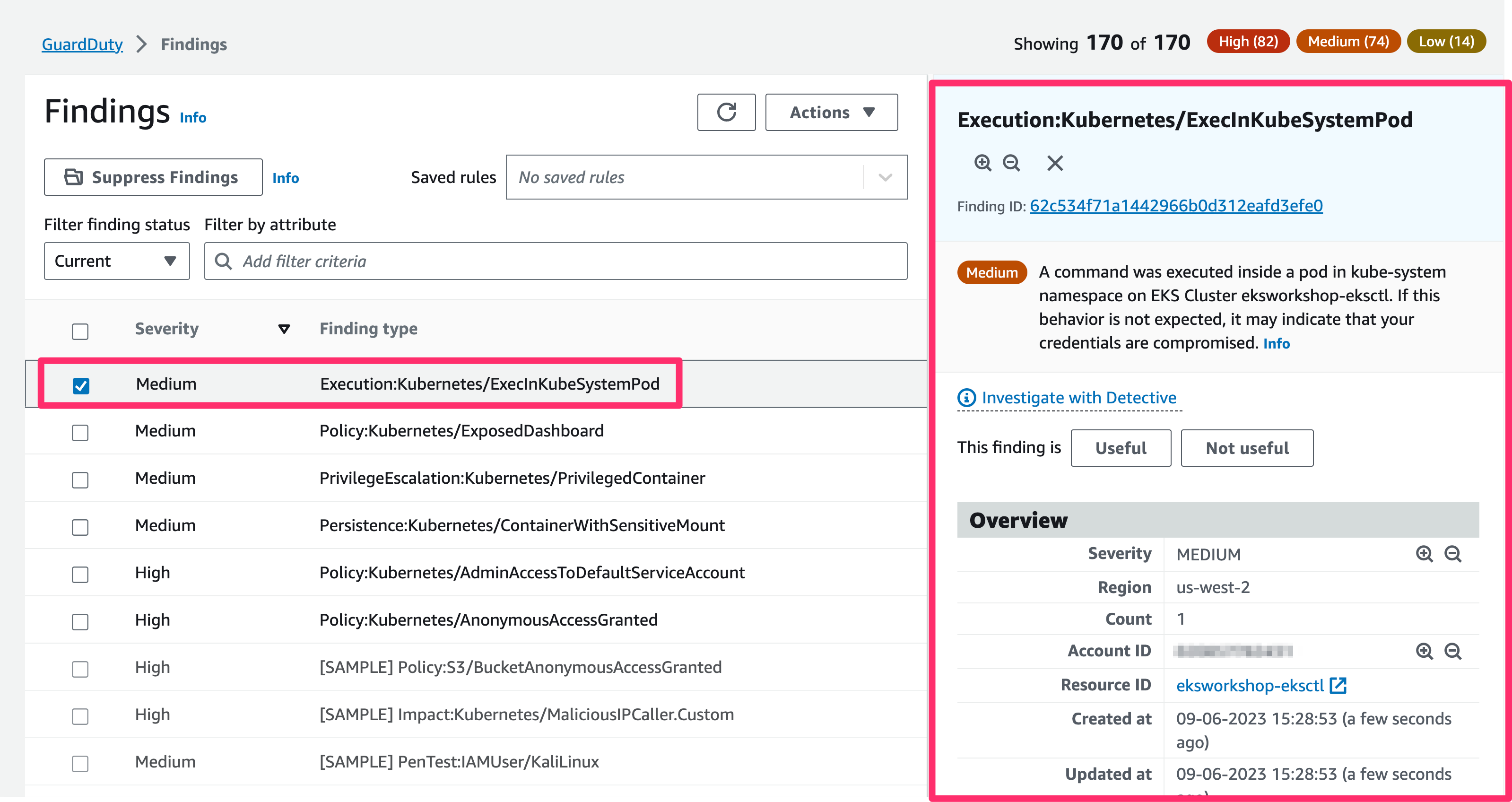
kubectl -n kube-system exec $AWS\_NODE\_POD -- /app/grpc-health-probe -addr=:50051

The output looks like below

Defaulted container "aws-node" out of: aws-node, aws-vpc-cni-init (init)

{"level":"info","ts":"2023-04-03T06:41:13.817Z","caller":"/root/sdk/go1.19.2/src/runtime/proc.go:250","msg":"status: SERVING"}

Go back to the [Amazon GuardDuty console](https://console.aws.amazon.com/guardduty/home) to check if a finding is generated.



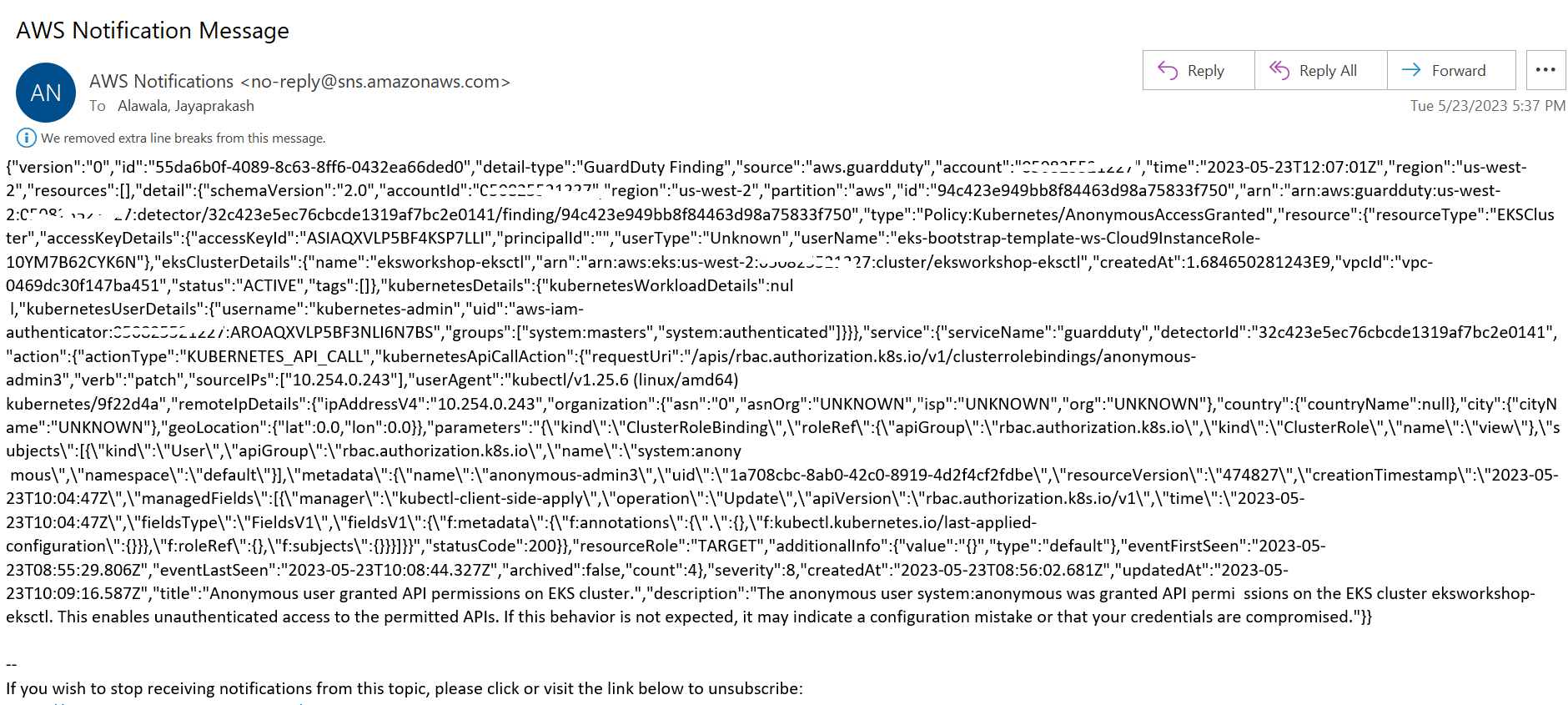
Amazon GuardDuty sends a notification to the Eventbridge.

After the event is emitted from GuardDuty and an Eventbridge rule is triggered, you should receive an email through the subscription on the SNS topic shortly after.

**Note**

Sometime, the Email Notification may take longer time. In that case, you can check your Email later.

The Email Notification will like below once it is arrived.



You can monitor Eventbridge metrics by CloudWatch. See details [here](https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-monitoring.html)

You can try re-generating the other finding types by modifying the name of resource in the yaml files.

# Cleanup

Once you have completed this chapter, you can cleanup the files and resources you created by issuing the following commands:

kubectl delete -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml

kubectl delete -f pod\_with\_sensitive\_mount.yaml

kubectl delete -f elevate.yaml

kubectl delete -f anonymous.yaml