Deploying Amazon CloudWatch Observability and Cost Management for EKS Clusters Using Helm Charts and Container Insights

By

Mahendran Selvakumar

https://devopstronaut.com/

https://www.linkedin.com/in/mahendran-selvakumar/

What is Amazon Cloudwatch Container Insights?

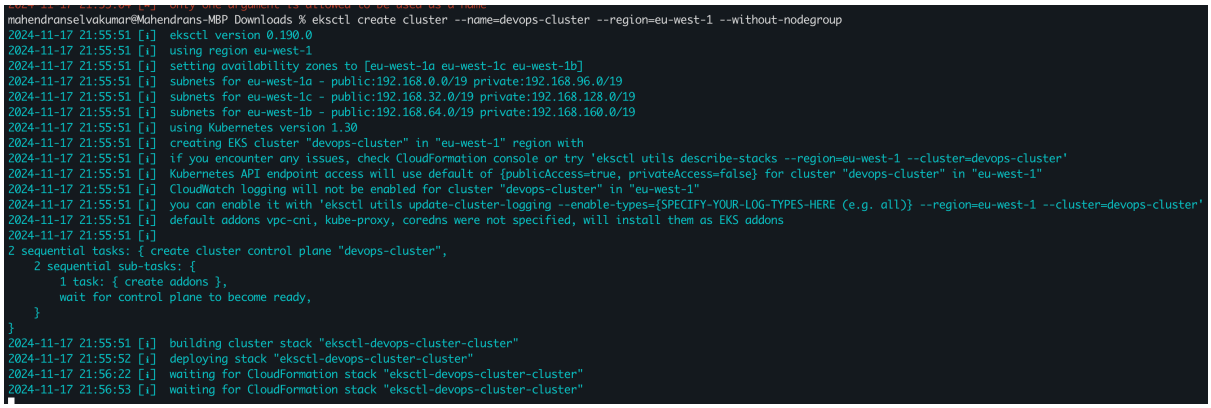
Amazon CloudWatch Container Insights is a feature within AWS that enables monitoring and troubleshooting of containerized applications and microservices. It collects, aggregates, and summarizes metrics and logs from services such as Amazon Elastic Container Service (ECS), Amazon Elastic Kubernetes Service (EKS), and Kubernetes platforms running on Amazon EC2 instances

Key Features:

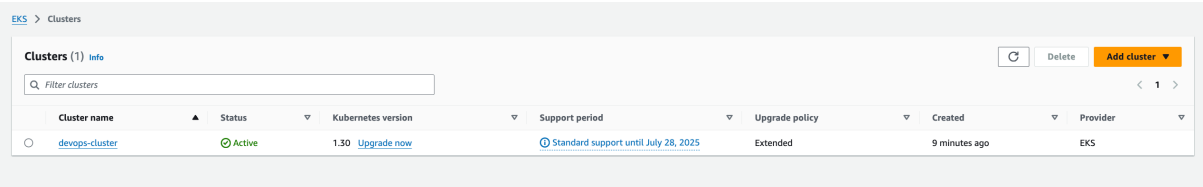
- **Resource Metrics:** Tracks CPU, memory, disk, and network usage for clusters, nodes, pods, tasks, and services.
- **Health Monitoring:** Provides data on container health, including restart failures and performance issues.
- Dashboards: Includes prebuilt dashboards in CloudWatch to view metrics in real-time.
- **Alerts:** Enables setting alarms based on collected metrics to proactively address potential issues.
- **Enhanced Observability:** For EKS on EC2, detailed metrics and curated dashboards are available for deeper insights into control plane and container-level performance.

Step 1: Create the EKS Cluster Without Any Node Groups

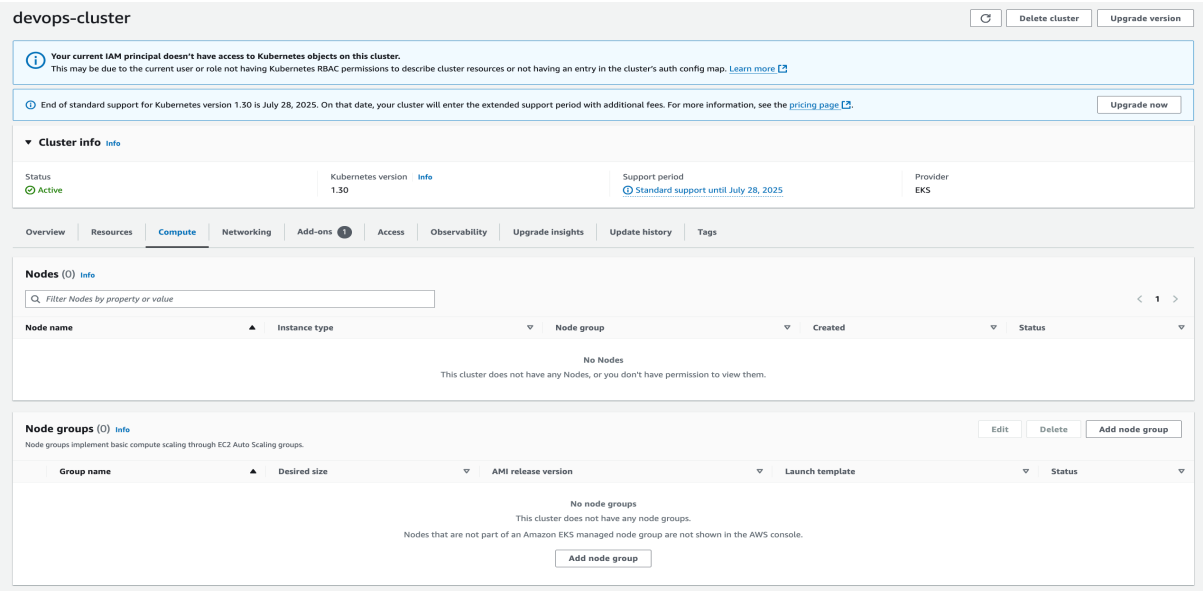
Create an EKS cluster without a node group using the eksctl command **(eksctl create cluster --name=devops-cluster --region=eu-west-1 --without-nodegroup)**. By default, eksctl creates a node group with m5.large instances, so we used the — without-nodegroup option to skip creating a default node group



Go to the EKS console to verify that the cluster was successfully created using eksctl



If you view the created EKS cluster in the console, you'll see that it was created without any node groups



Step 2: Create Node group and Nodes:

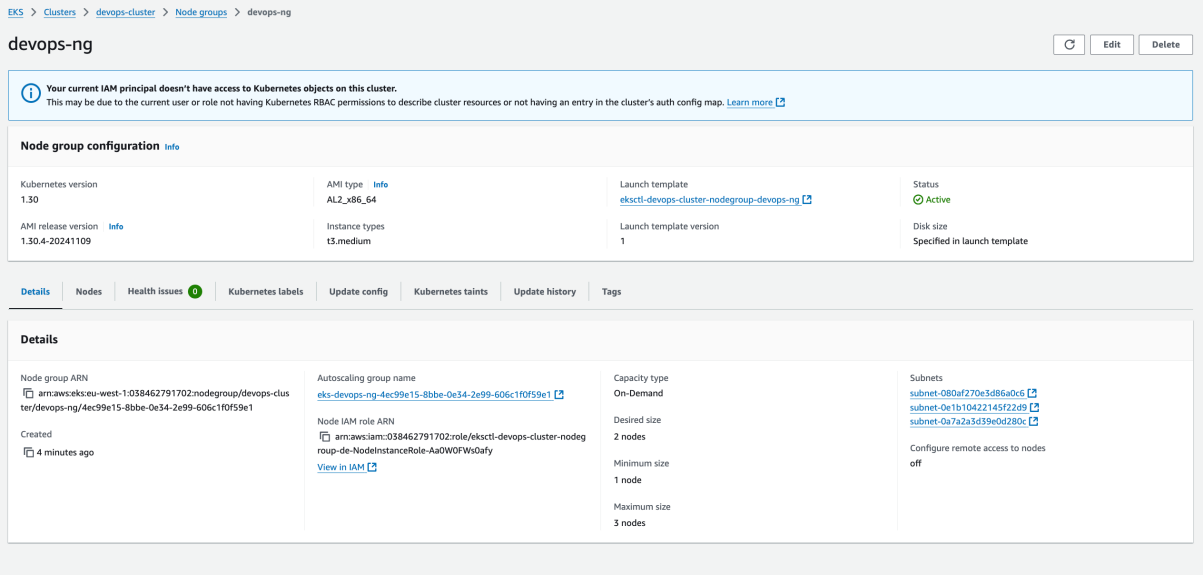
Add a node group using the following separate eksctl command**: eksctl create nodegroup --name devops-ng --cluster devops-cluster --region eu-west-1 --nodes 2 --nodes-min 1 --nodes-max 3 --node-type t3.medium**

```
mahendranselvakumar@Mahendrans-MBP Downloads % eksctl create nodegroup --name devops-ng --cluster devops-cluster  --region eu-west-1 --nodes 2 --nodes
-min 1 --nodes-max 3 --node-type t3.medium
2024-11-17 22:07:33 [ℹ]  will use version 1.30 for new nodegroup(s) based on control plane version
2024-11-17 22:07:35 [ℹ]  nodegroup "devops-ng" will use "" [AmazonLinux2/1.30]
2024-11-17 22:07:35 [ℹ]  1 nodegroup (devops-ng) was included (based on the include/exclude rules)
2024-11-17 22:07:35 [ℹ]  will create a CloudFormation stack for each of 1 managed nodegroups in cluster "devops-cluster"
2024-11-17 22:07:35 [ℹ]
2 sequential tasks: { fix cluster compatibility, 1 task: { 1 task: { create managed nodegroup "devops-ng" } }
}
2024-11-17 22:07:35 [ℹ]  checking cluster stack for missing resources
2024-11-17 22:07:36 [ℹ]  cluster stack has all required resources
2024-11-17 22:07:36 [ℹ]  building managed nodegroup stack "eksctl-devops-cluster-nodegroup-devops-ng"
2024-11-17 22:07:36 [ℹ]  deploying stack "eksctl-devops-cluster-nodegroup-devops-ng"
2024-11-17 22:07:36 [ℹ]  waiting for CloudFormation stack "eksctl-devops-cluster-nodegroup-devops-ng"
2024-11-17 22:08:06 [ℹ]  waiting for CloudFormation stack "eksctl-devops-cluster-nodegroup-devops-ng"
2024-11-17 22:08:48 [ℹ]  waiting for CloudFormation stack "eksctl-devops-cluster-nodegroup-devops-ng"
2024-11-17 22:10:17 [ℹ]  waiting for CloudFormation stack "eksctl-devops-cluster-nodegroup-devops-ng"
2024-11-17 22:10:17 [ℹ]  no tasks
2024-11-17 22:10:17 [✔]  created 0 nodegroup(s) in cluster "devops-cluster"
2024-11-17 22:10:18 [ℹ]  nodegroup "devops-ng" has 2 node(s)
2024-11-17 22:10:18 [ℹ]  node "ip-192-168-7-113.eu-west-1.compute.internal" is ready
2024-11-17 22:10:18 [ℹ]  node "ip-192-168-89-183.eu-west-1.compute.internal" is ready
2024-11-17 22:10:18 [ℹ]  waiting for at least 1 node(s) to become ready in "devops-ng"
2024-11-17 22:10:18 [ℹ]  nodegroup "devops-ng" has 2 node(s)
2024-11-17 22:10:18 [ℹ]  node "ip-192-168-7-113.eu-west-1.compute.internal" is ready
2024-11-17 22:10:18 [ℹ]  node "ip-192-168-89-183.eu-west-1.compute.internal" is ready
2024-11-17 22:10:18 [✔]  created 1 managed nodegroup(s) in cluster "devops-cluster"
2024-11-17 22:10:18 [ℹ]  checking security group configuration for all nodegroups
2024-11-17 22:10:18 [ℹ]  all nodegroups have up-to-date cloudformation templates
mahendranselvakumar@Mahendrans-MBP Downloads %
```

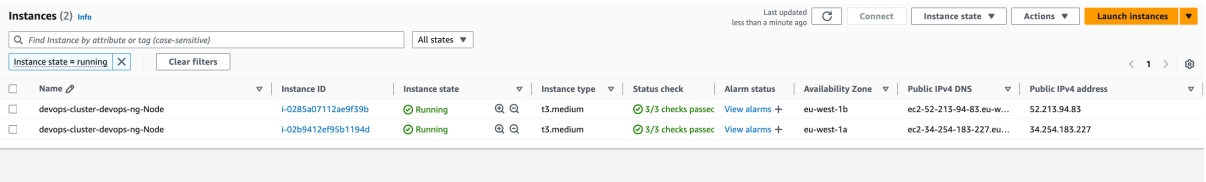
Explanation of the flags:

- --cluster: Specifies the name of the existing EKS cluster to which the node group will be added.
- --name: Names the node group for easy identification.
- --region: Specifies the AWS region.
- --nodes: Sets the initial desired number of nodes (in this case, 2).
- --nodes-min and — nodes-max: Define the minimum and maximum number of nodes for auto-scaling.
- --node-type: The EC2 instance type for the nodes (e.g., m5.large)

If you check in the console, you'll see that the node group has been created



In the instances section of the EC2 console, you can view the Instances created for the node group



Step 3: Configure Context for EKS Cluster:

Set the Kubernetes context for the EKS cluster using the following command: **aws eks --region eu-west-1 update-kubeconfig --name devops-cluster**

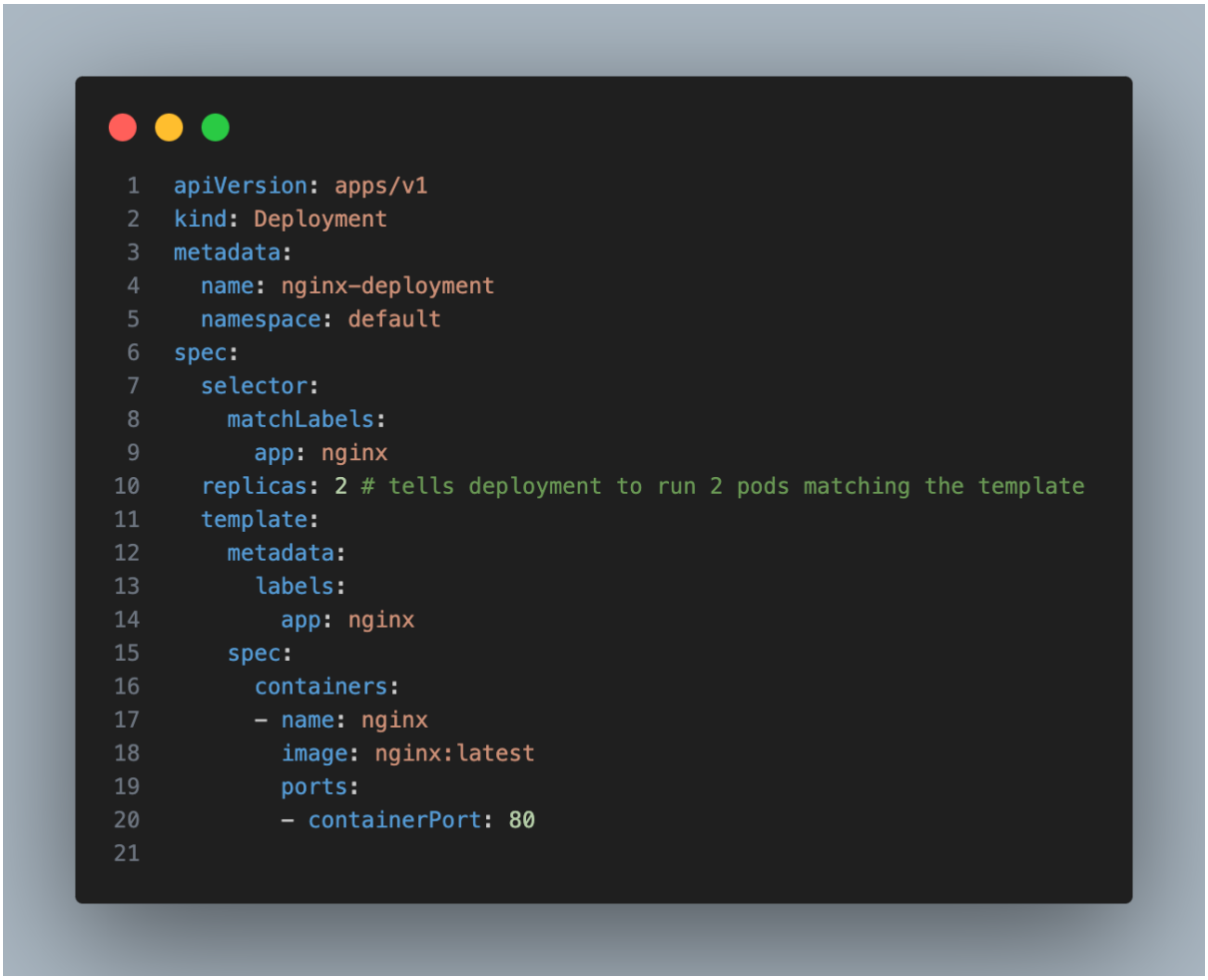
```
mahendranselvakumar@Mahendrans-MBP Downloads % aws eks --region eu-west-1 update-kubeconfig --name devops-cluster
Added new context arn:aws:eks:eu-west-1:038462791702:cluster/devops-cluster to /Users/mahendranselvakumar/.kube/config
mahendranselvakumar@Mahendrans-MBP Downloads %
```

Use **kubectl get ns** to view namespaces and **kubectl get nodes** to check the status of the nodes in the cluster, verifying that the setup is complete

```
mahendranselvakumar@Mahendrans-MBP Downloads % kubectl get ns
NAME              STATUS   AGE
default           Active   13m
kube-node-lease   Active   13m
kube-public       Active   13m
kube-system       Active   13m
mahendranselvakumar@Mahendrans-MBP Downloads % kubectl get nodes
NAME                                           STATUS   ROLES    AGE     VERSION
ip-192-168-7-113.eu-west-1.compute.internal    Ready    <none>   5m52s   v1.30.4-eks-a737599
ip-192-168-89-183.eu-west-1.compute.internal   Ready    <none>   5m49s   v1.30.4-eks-a737599
mahendranselvakumar@Mahendrans-MBP Downloads %
```

Step 4: Deploy nginx application

Deploy nginx application in your EKS cluster with below code in nginx-deployment.yaml file



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  namespace: default
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80

```

- **Namespace:** If omitted, the default namespace is used, but for real-world projects, it's best to use namespaces for separation (e.g., dev, staging, prod).

- **Replicas:** Set this value based on your application's scalability and high-availability requirements.
- **Selectors and Labels:** Ensure these match to link the Deployment with the pods it manages.
- **Container Configuration:**
    - **Image:** Specify a versioned tag (nginx:1.14.2) instead of latest for stability.
    - **Ports:** Define only the necessary ports to limit attack surfaces.

Apply the configuration using this command **kubectl apply -f nginx-deployment.yaml**

```
mahendranselvakumar@Mahendrans-MBP Downloads % kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
mahendranselvakumar@Mahendrans-MBP Downloads %
```

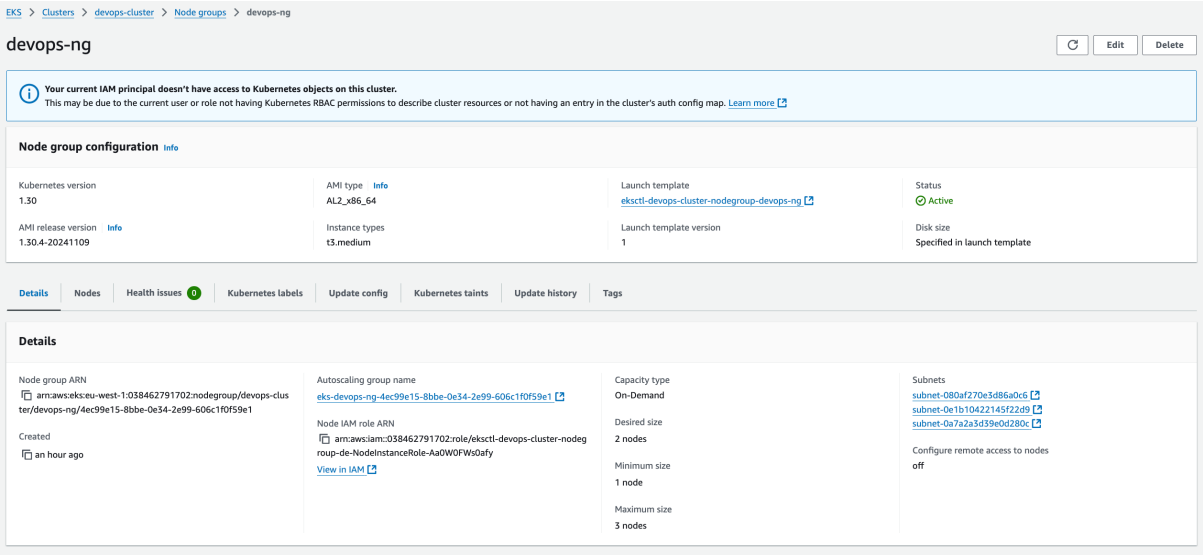
Run kubectl get pods to check the status of the pods

```
mahendranselvakumar@Mahendrans-MBP Downloads % kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-576c6b7b6-64pjk    1/1     Running   0          38s
nginx-deployment-576c6b7b6-78tvt    1/1     Running   0          38s
mahendranselvakumar@Mahendrans-MBP Downloads %
```

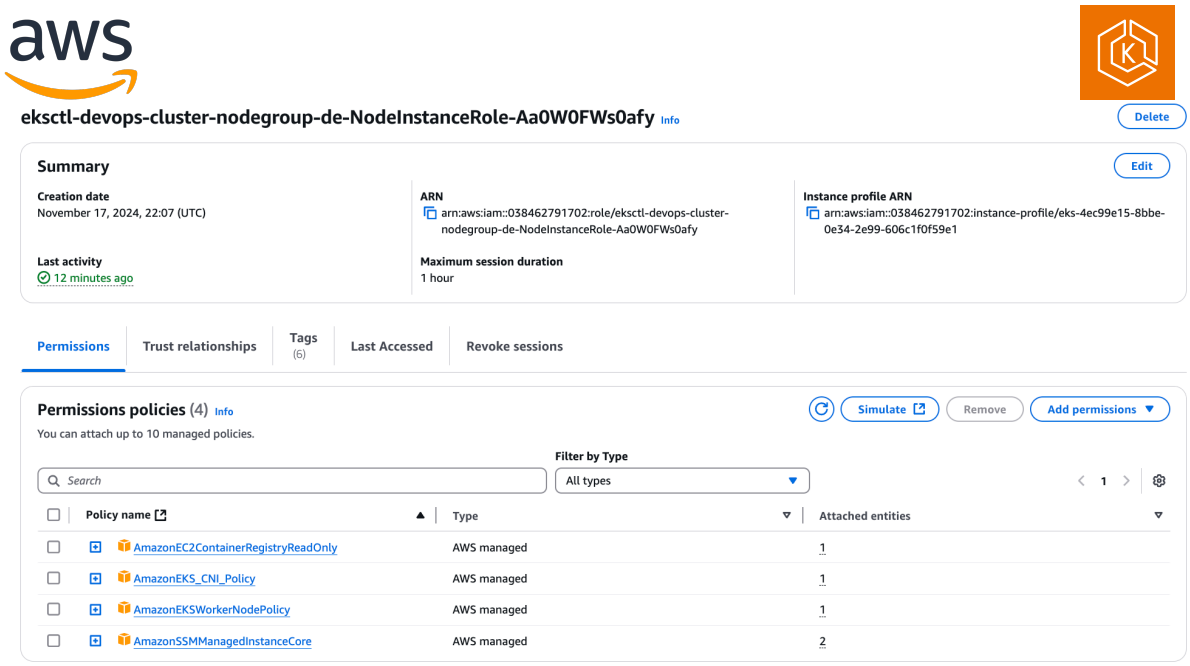
Step 5: Install with IAM permissions on worker nodes

Associate the CloudWatch policy with the IAM role assigned to your EKS worker nodes to configure CloudWatch observability

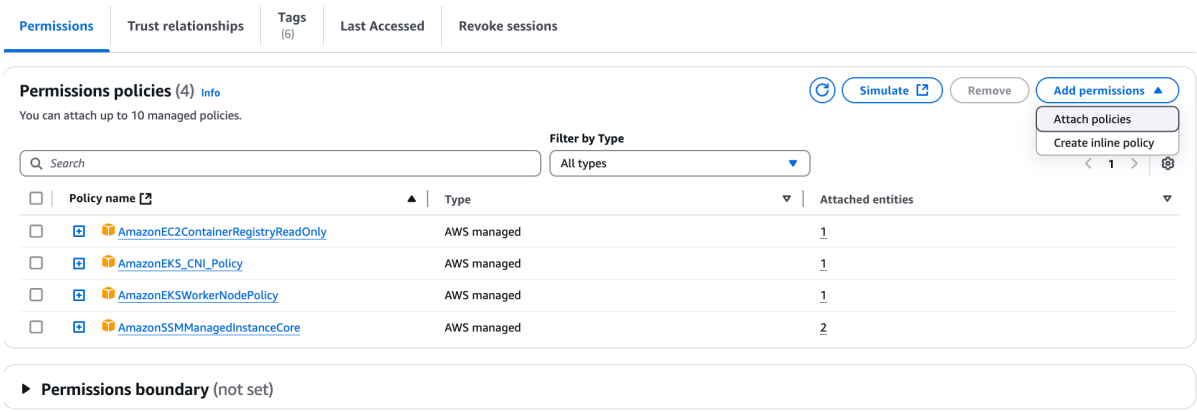
Go to the Node Group and click 'View in IAM' under the Node IAM Role ARN



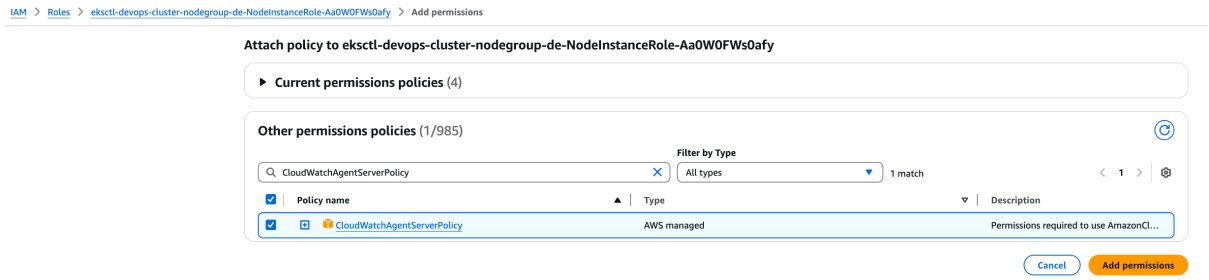
Click **Add permissions**



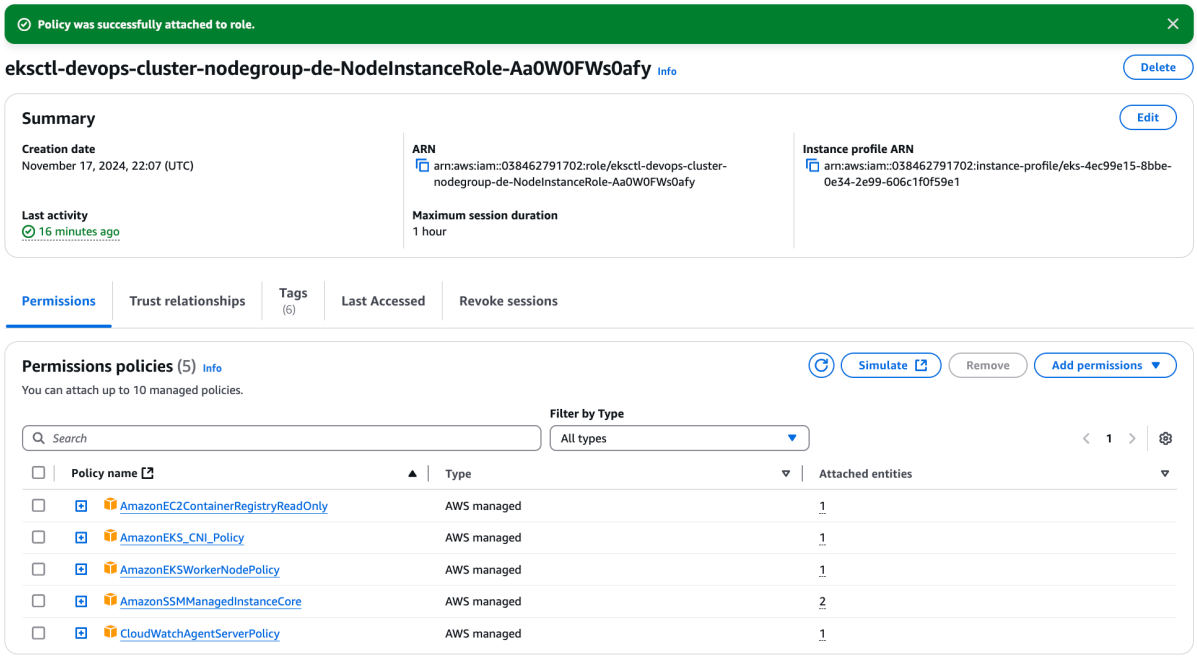
Click **Attach Policies**



Search for CloudWatchAgentServerPolicy, select the policy, and click **Add permissions**



The policy has now been successfully attached to the NodeGroup role



Step 6: Install the CloudWatch Agent and the Fluent-bit agent on an Amazon EKS cluster using Amazon CloudWatch Observability Helm chart

You can use Amazon CloudWatch Observability Helm chart to install the CloudWatch Agent and the Fluent-bit agent on an Amazon EKS cluster

Add the Helm Repository:

Add the AWS Observability Helm repository using this command **helm repo add aws-observability** https://aws-observability.github.io/helm-charts

```
mahendranselvakumar@Mahendrans-MBP Downloads % helm repo add aws-observability https://aws-observability.github.io/helm-charts
"aws-observability" has been added to your repositories
mahendranselvakumar@Mahendrans-MBP Downloads %
```

Update the repo using this command **helm repo update aws-observability**

```
mahendranselvakumar@Mahendrans-MBP Downloads % helm repo update aws-observability
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "aws-observability" chart repository
Update Complete. ⎈Happy Helming!⎈
mahendranselvakumar@Mahendrans-MBP Downloads %
```

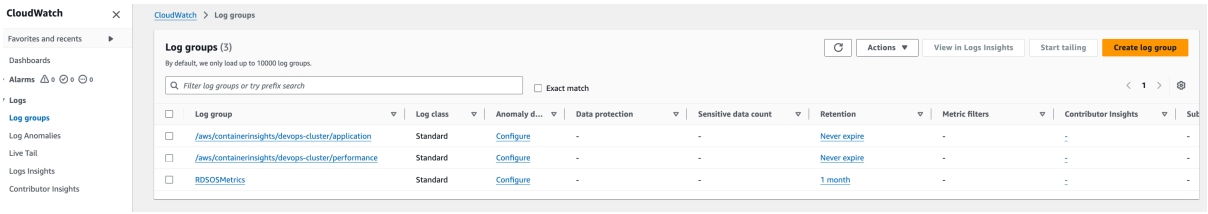
Install the aws observability helm chart using this command **helm install --wait --create-namespace --namespace amazon-cloudwatch amazon-cloudwatch-observability aws-observability/amazon-cloudwatch-observability --set clusterName=devops-cluster --set region=eu-west-1**

```
mahendranselvakumar@Mahendrans-MBP Downloads % helm install --wait --create-namespace --namespace amazon-cloudwatch amazon-cloudwatch-observability aw
s-observability/amazon-cloudwatch-observability --set clusterName=devops-cluster --set region=eu-west-1
NAME: amazon-cloudwatch-observability
LAST DEPLOYED: Sun Nov 17 22:28:54 2024
NAMESPACE: amazon-cloudwatch
STATUS: deployed
REVISION: 1
TEST SUITE: None
mahendranselvakumar@Mahendrans-MBP Downloads %
```

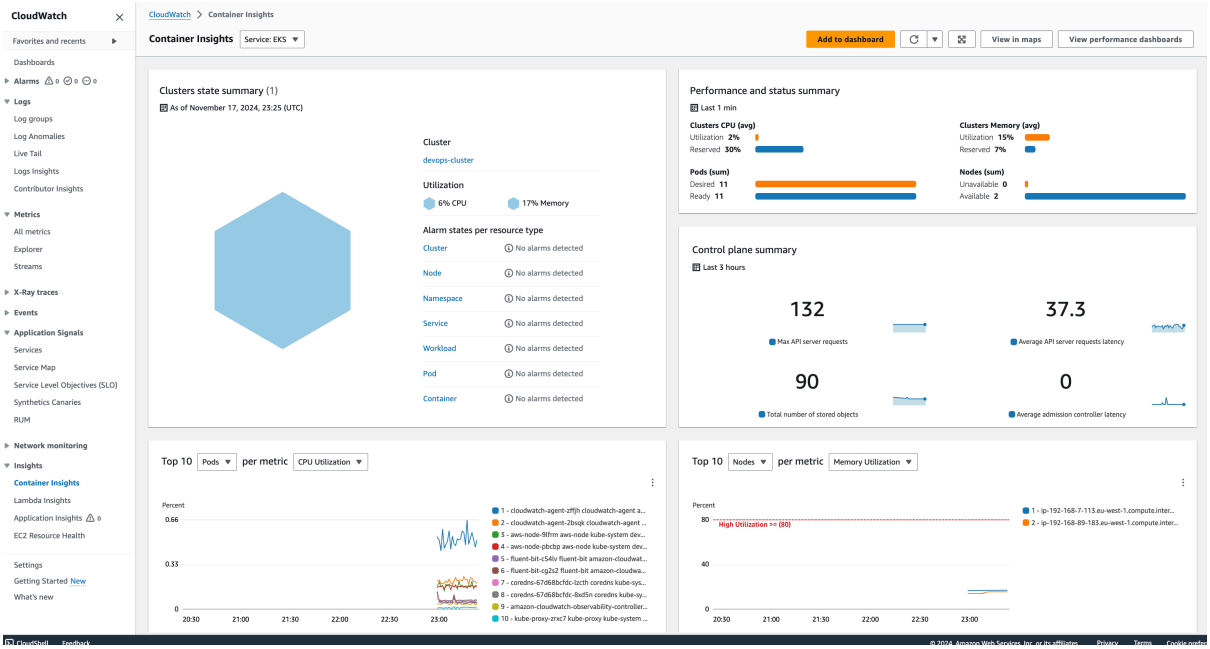
Run **kubectl get pods -n amazon-cloudwatch** to ensure the agent pods are running

```
mahendranselvakumar@Mahendrans-MBP Downloads % kubectl get pods -n amazon-cloudwatch
NAME                                                READY   STATUS    RESTARTS   AGE
amazon-cloudwatch-observability-controller-manager-78fdbc7s9vh7   1/1     Running   0          87s
cloudwatch-agent-2bsqk                              1/1     Running   0          79s
cloudwatch-agent-zffjh                              1/1     Running   0          79s
fluent-bit-c54lv                                    1/1     Running   0          87s
fluent-bit-cg2s2                                    1/1     Running   0          87s
mahendranselvakumar@Mahendrans-MBP Downloads %
```

Go to CloudWatch, select **CloudWatch log groups**, you will be able to see the log groups associated with your EKS cluster

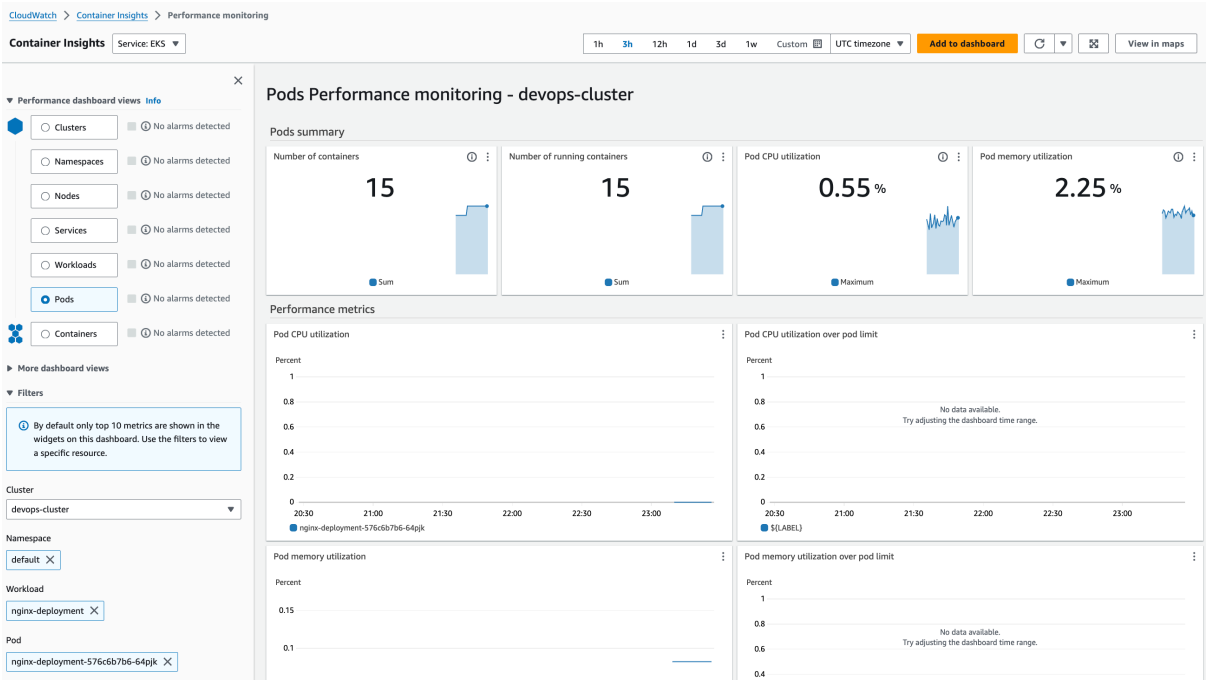


In Container Insights, you will be able to view the performance and status of your EKS cluster.

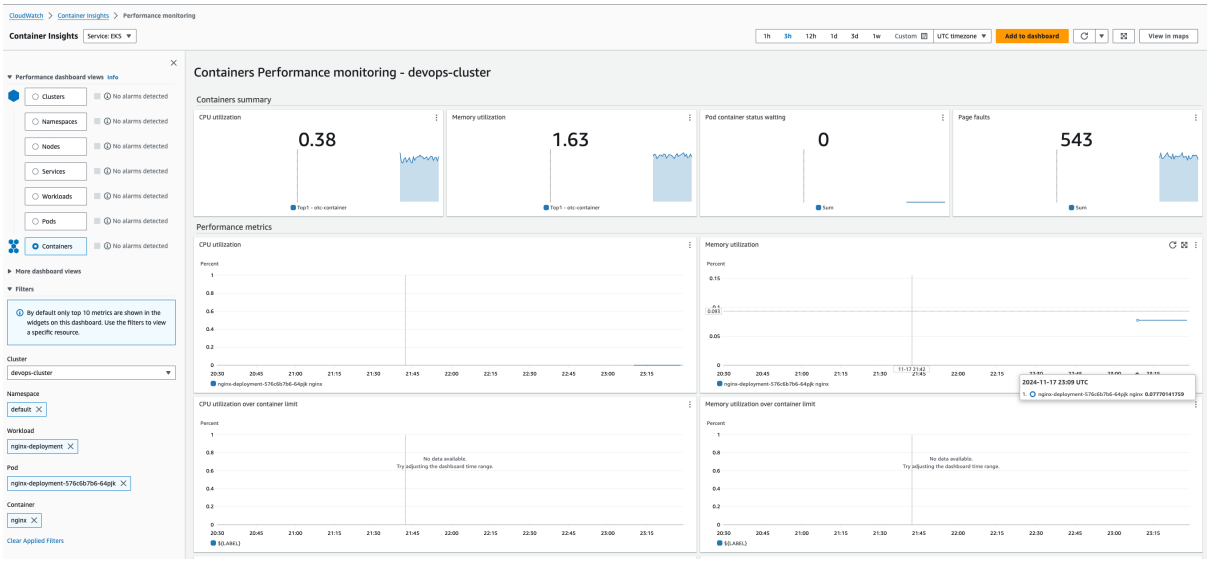


Click Created cluster under the Cluster Overview in Container Insights

To view pod performance monitoring, select **Pods**, and in the Filters section, choose the **Cluster**, **Namespace**, **Workload** and **Pod** in the Performance Dashboard view section



To view container performance monitoring, select **Containers**, and in the Filters section, choose the **Cluster**, **Namespace**, **Workload**, **Pod**, and **Container** in the Performance Dashboard view section

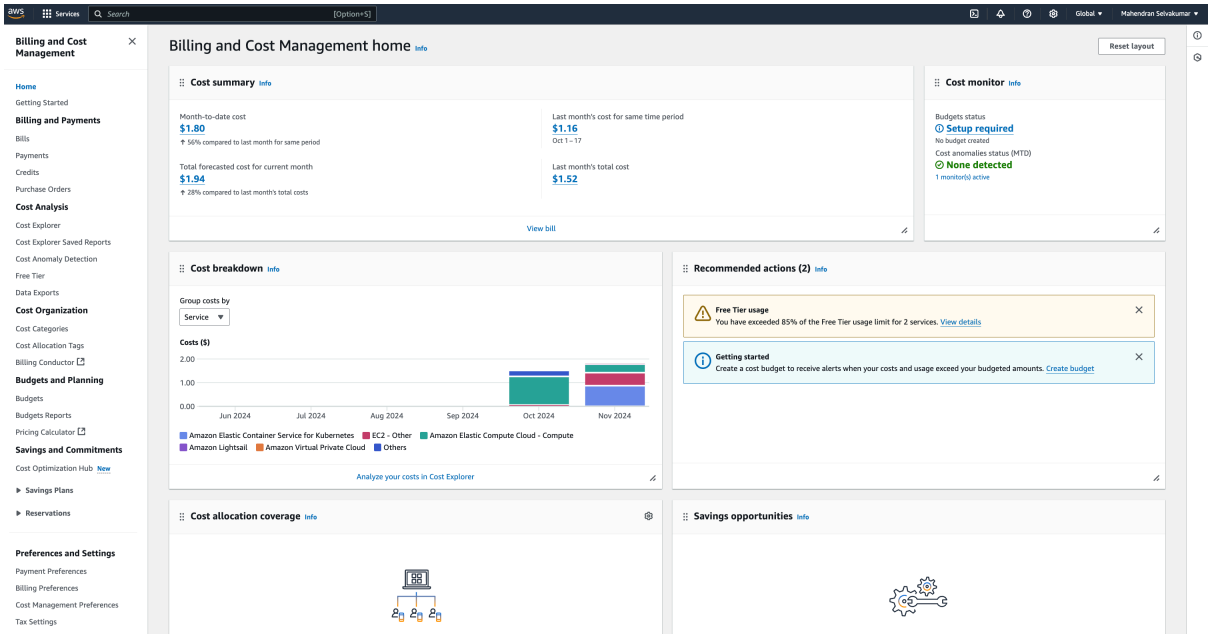


https://www.linkedin.com/in/mahendran-selvakumar/

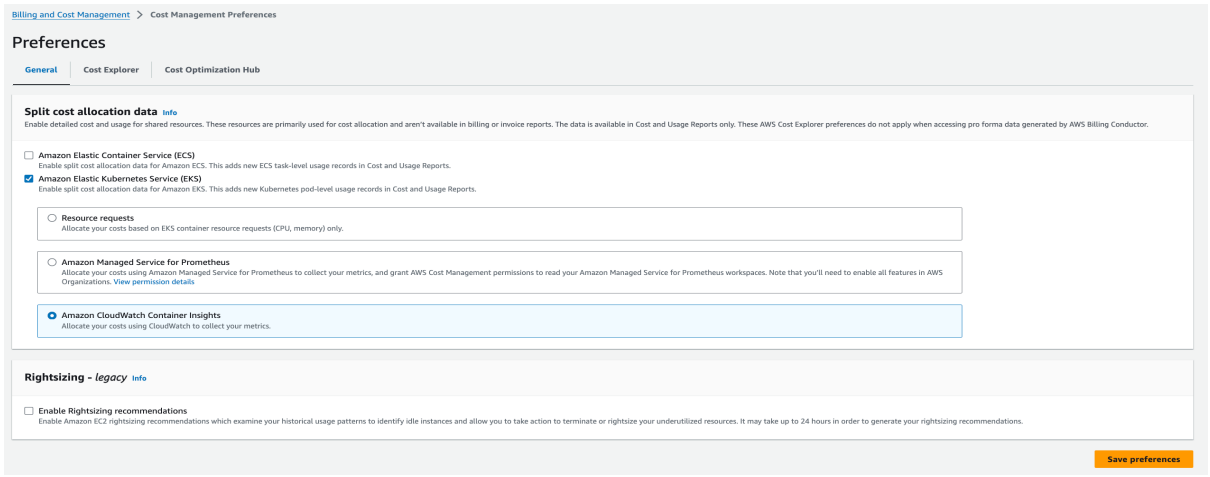
Step 7: Enable Split cost allocation data for Amazon EKS

It is a prerequisite to opt in to split cost allocation data through the Cost Management preferences

Go to **Billing and Management** and select **Cost Management Preferences** under **Preferences and Settings**



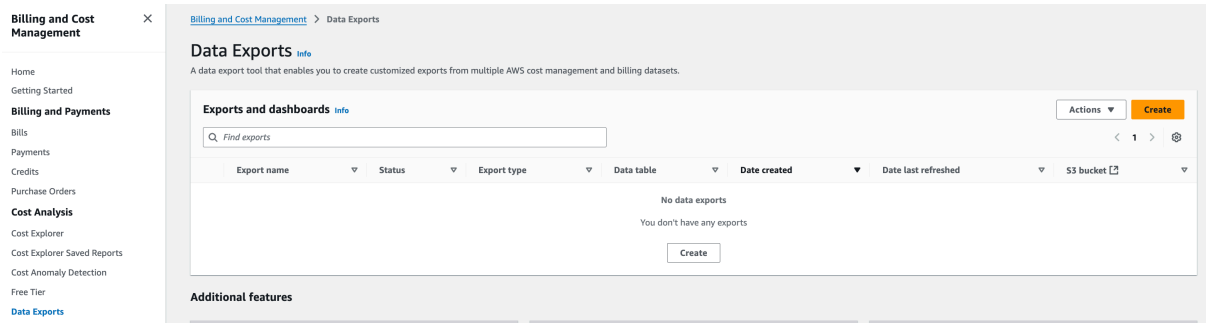
Select Amazon EKS, choose Amazon Container Insights, and click Save Preferences



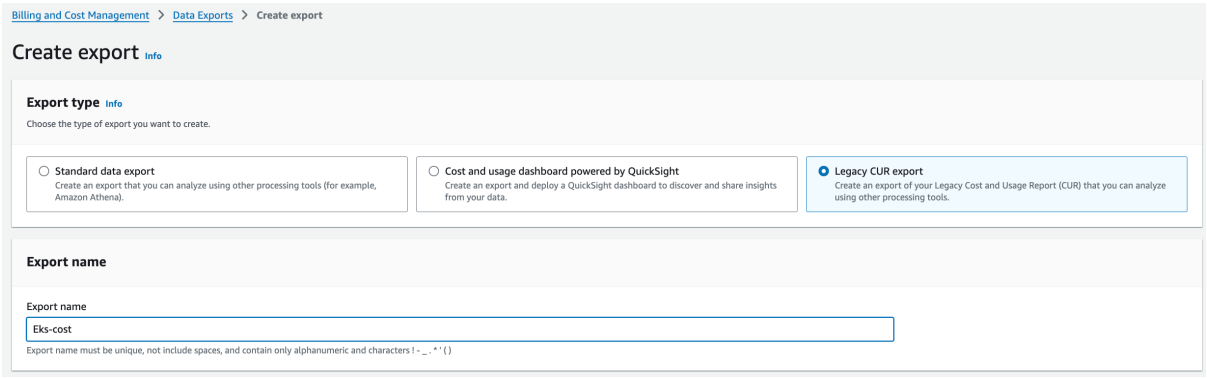
Step 8: Include cost and usage data in your report

Once you've opted in, you can choose to have cost and usage data for container-level resources included in your report during step one of report creation.

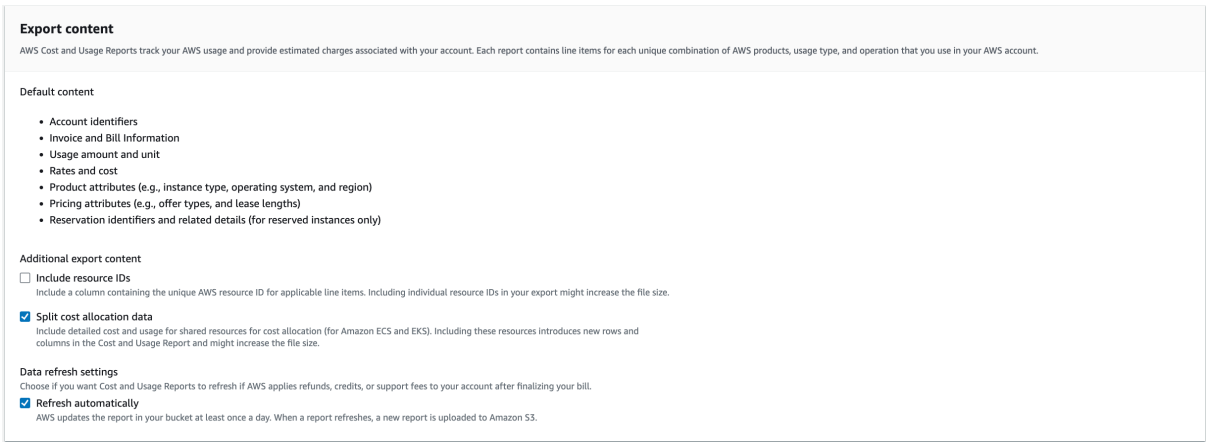
Go to **Data Exports** under **Cost Analysis** and click **Create**



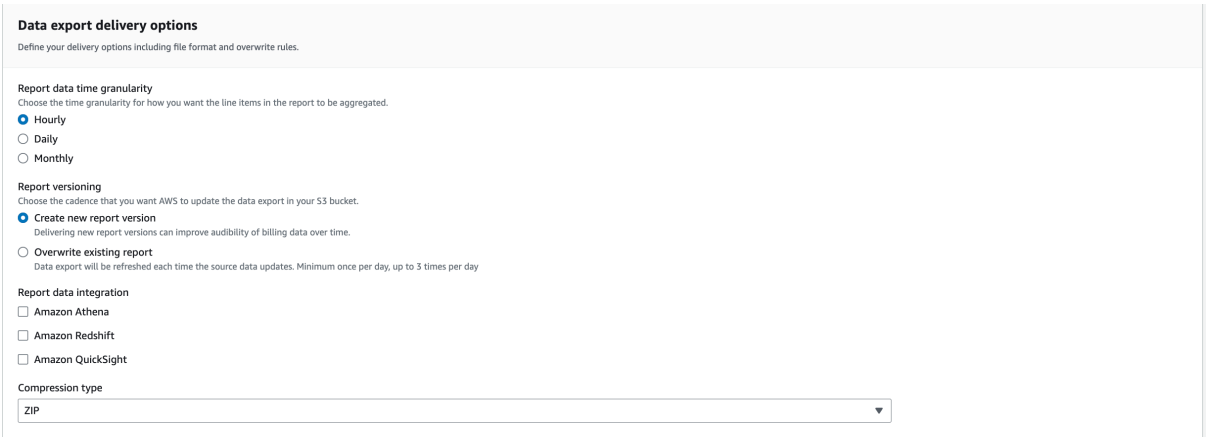
Select **Legacy CUR Export and** provide the export name



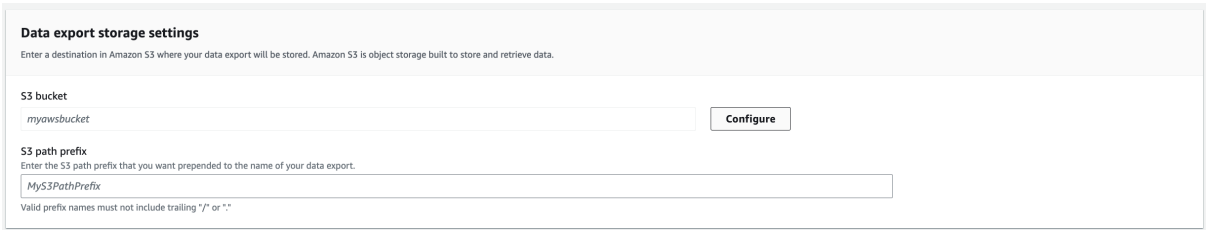
Select Split cost allocation data and enable Refresh automatically in the Data Refresh Settings



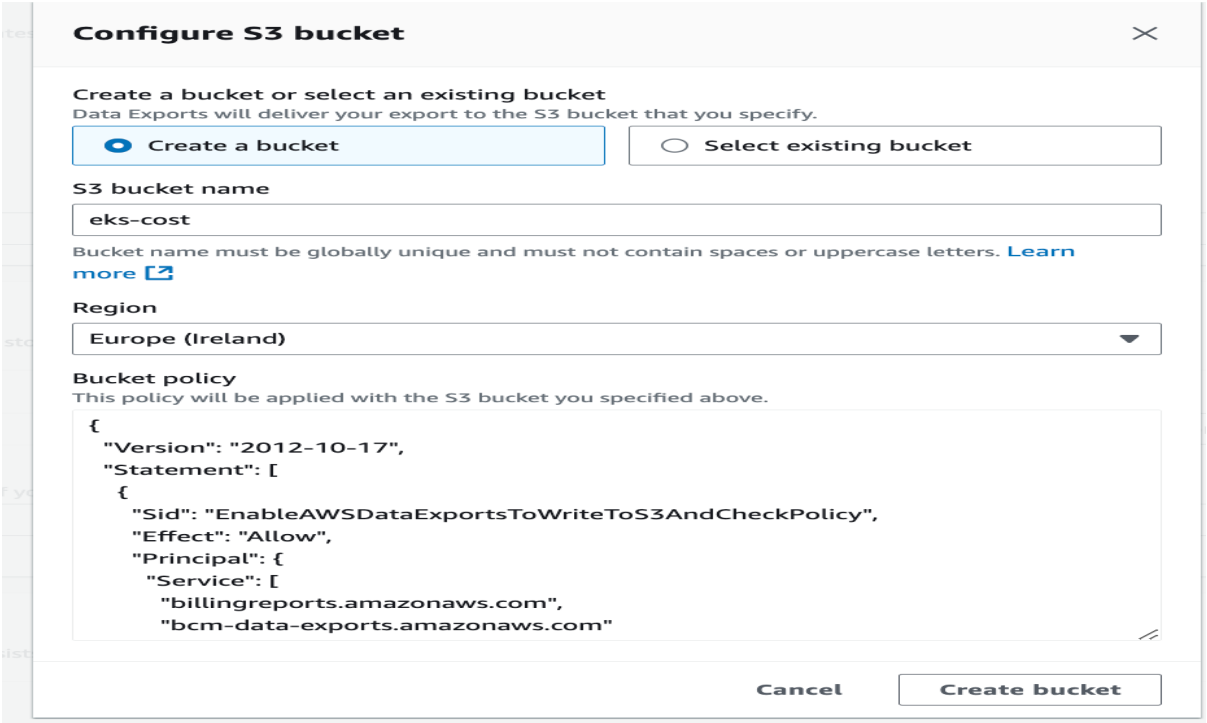
Choose the report data time granularity and set the report version to Create new report version



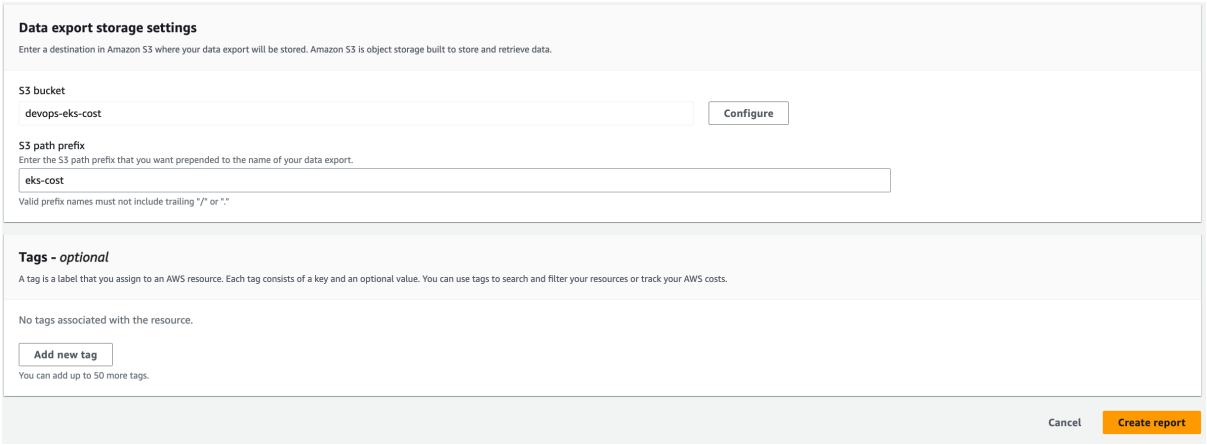
Click Configure to set up Data Export Storage



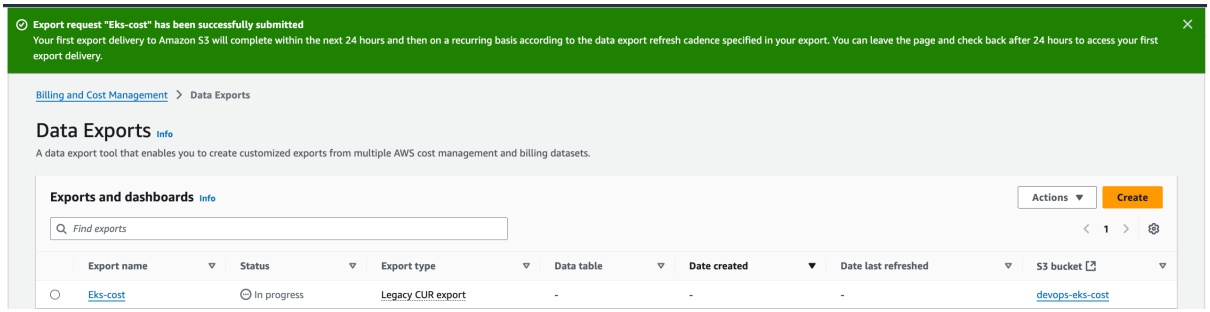
Provide the S3 bucket name, select the Region, and click Create bucket



Provide the S3 Path Prefix and click Create Report



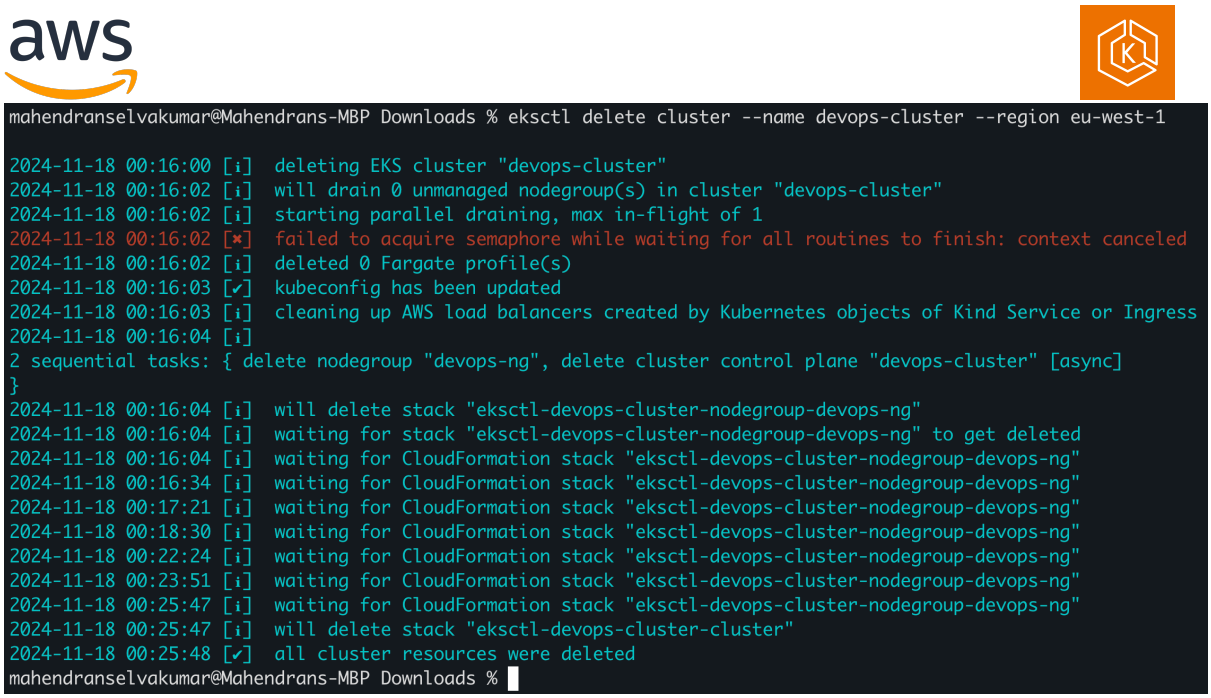
The data export has been successfully created, and it will take up to 24 hours to complete



Once the feature is enabled, the pod-level usage data will be available in CUR within 24 hours

Step 9: Delete the EKS Cluster

Run this command to delete EKS cluster **eksctl delete cluster --name devops-cluster --region eu-west-1**



All resources dependent on the EKS cluster have now been deleted.

Conclusion:

This guide explained how to set up observability and cost management for an Amazon EKS cluster. It included attaching the CloudWatchAgentServerPolicy to the worker nodes' IAM role, using the aws-observability Helm repository to monitor performance through CloudWatch Container Insights, and configuring cost management settings for detailed tracking. Finally, it covered cleaning up associated resources for proper management

Keep Learning, Keep EKS-ing!!

Feel free to reach out to me, if you have any other queries or suggestions Stay connected on LinkedIn: Mahendran Selvakumar

Stay connected on Medium: https://devopstronaut.com/