

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



By

Mahendran Selvakumar

<https://devopstronaut.com/>



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

```
mahendralselvakumar@Mahendrals-MBP Downloads % eksctl create cluster --name=devops-cluster --region=eu-west-1 --without-nodegroup
2024-11-17 21:55:51 [i] eksctl version 0.190.0
2024-11-17 21:55:51 [i] using region eu-west-1
2024-11-17 21:55:51 [i] setting availability zones to [eu-west-1a eu-west-1c eu-west-1b]
2024-11-17 21:55:51 [i] subnets for eu-west-1a - public::192.168.0.0/19 private::192.168.96.0/19
2024-11-17 21:55:51 [i] subnets for eu-west-1c - public::192.168.32.0/19 private::192.168.128.0/19
2024-11-17 21:55:51 [i] subnets for eu-west-1b - public::192.168.64.0/19 private::192.168.160.0/19
2024-11-17 21:55:51 [i] using Kubernetes version 1.30
2024-11-17 21:55:51 [i] creating EKS cluster "devops-cluster" in "eu-west-1" region with
2024-11-17 21:55:51 [i] if you encounter any issues, check CloudFormation console or try `eksctl utils describe-stacks --region=eu-west-1 --cluster=devops-cluster`
2024-11-17 21:55:51 [i] Kubernetes API endpoint access will use default of [publicAccess=true, privateAccess=false] for cluster "devops-cluster" in "eu-west-1"
2024-11-17 21:55:51 [i] CloudWatch logging will not be enabled for cluster "devops-cluster" in "eu-west-1"
2024-11-17 21:55:51 [i] you can enable it with `eksctl utils update-cluster-logging --enable-types=[SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)] --region=eu-west-1 --cluster=devops-cluster`
2024-11-17 21:55:51 [i] default addons vpc-cni, kube-proxy, coredns were not specified, will install them as EKS addons
2024-11-17 21:55:51 [i]
2 sequential tasks: { create cluster control plane "devops-cluster",
  2 sequential sub-tasks: {
    1 task: { create addons },
    wait for control plane to become ready,
  }
}
2024-11-17 21:55:51 [i] building cluster stack "eksctl-devops-cluster-cluster"
2024-11-17 21:55:52 [i] deploying stack "eksctl-devops-cluster-cluster"
2024-11-17 21:56:22 [i] waiting for CloudFormation stack "eksctl-devops-cluster-cluster"
2024-11-17 21:56:53 [i] waiting for CloudFormation stack "eksctl-devops-cluster-cluster"
```

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

The screenshot shows the AWS EKS console with the URL <https://console.aws.amazon.com/eks/clusters>. It displays a single cluster named "devops-cluster". The cluster details are as follows:

Cluster name	Status	Kubernetes version	Support period	Upgrade policy	Created	Provider
devops-cluster	Active	1.30 Upgrade now	Standard support until July 28, 2025	Extended	9 minutes ago	EKS

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

The screenshot shows the AWS EKS cluster details page for "devops-cluster". The cluster status is "Active" with Kubernetes version 1.30 and standard support ending on July 28, 2025. The Compute tab is selected, showing the "Nodes (0) Info" section which states "This cluster does not have any Nodes, or you don't have permission to view them." and the "Node groups (0) Info" section which states "This cluster does not have any node groups." Both sections have "Add node group" buttons.

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**

```
mahendralselvakumar@Mahendran-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 [v] 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 [v] 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
mahendralselvakumar@Mahendran-MBP Downloads %
```

Explanation of the flags:

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



- **--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

The screenshot shows the AWS EKS Node Groups console. The URL is [EKS > Clusters > devops-cluster > Node groups > devops-ng](#). The page displays the configuration for the 'devops-ng' node group, which was created 4 minutes ago. It shows details like Kubernetes version (1.30), AMI type (AL2_x86_64), Launch template (eksctl-devops-cluster-nodegroup-devops-ng), and Status (Active). The node group ARN is arn:aws:eks:eu-west-1:038462791702:nodegroup/devops-cluster/devops-ng/4ec99e15-8bbe-0e34-2e99-606c1f0f59e1. The node IAM role ARN is arn:aws:iam:038462791702:role/eksctl-devops-cluster-nodegroup-de-NodeInstanceRole-Aa0WOFWs0afy. The autoscaling group name is eks-devops-ng-4ec99e15-8bbe-0e34-2e99-606c1f0f59e1. Capacity type is On-Demand, and the desired size is 2 nodes. Subnets listed are subnet-080af270e3d86a0c6, subnet-0e1b10422145f22d9, and subnet-0a7a2a3d39e0d280c. The node group is configured to off remote access. The node group has 2 nodes in total, with 1 node running and 1 node pending.

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

The screenshot shows the AWS EC2 Instances console. The URL is [Instances \(2\) Info](#). It lists two instances: 'devops-cluster-devops-ng-Node' and 'devops-cluster-devops-ng-Node'. Both instances are in the 'Running' state, with Instance ID i-0285a07112ae9f39b and i-02b9412ef95b1194d respectively. They are t3.medium instance type, with 3/3 checks passed and alarm status 'View alarms +'. They are located in eu-west-1b and eu-west-1a availability zones, with Public IPv4 DNS ec2-52-215-94-83.eu-west-1.amazonaws.com and ec2-34-254-183-227.eu-west-1.amazonaws.com, and Public IPv4 address 52.213.94.83 and 34.254.183.227.

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**

```
LAST 10 LINES [1] all nodegroups have up to date transformation templates
mahendralselvakumar@Mahendrals-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/mahendralselvakumar/.kube/config
mahendralselvakumar@Mahendrals-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

```
mahendralselvakumar@Mahendrals-MBP Downloads % kubectl get ns
NAME      STATUS  AGE
default   Active  13m
kube-node-lease  Active  13m
kube-public  Active  13m
kube-system  Active  13m
mahendralselvakumar@Mahendrals-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
mahendralselvakumar@Mahendrals-MBP Downloads %
```

Step 4: Deploy nginx application

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

```
● ● ●

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: nginx-deployment
5    namespace: default
6  spec:
7    selector:
8      matchLabels:
9        app: nginx
10   replicas: 2 # tells deployment to run 2 pods matching the template
11   template:
12     metadata:
13       labels:
14         app: nginx
15     spec:
16       containers:
17         - name: nginx
18           image: nginx:latest
19         ports:
20           - containerPort: 80
21
```

- **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**

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

Run kubectl get pods to check the status of the pods

```
mahendralselvakumar@Mahendrals-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
mahendralselvakumar@Mahendrals-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

The screenshot shows the AWS EKS Node Groups configuration for the 'devops-ng' node group. It includes sections for Node group configuration, Details, and a large 'View in IAM' button. The IAM section displays the Node IAM role ARN, which is associated with the CloudWatchLogsFullAccess policy.

Kubernetes version	AMI type	Launch template	Status
1.30	AL2_x86_64	eksctl-devops-cluster-nodegroup-devops-ng	Active
AMI release version	Instance types	Launch template version	Disk size
1.30.4-20241109	t3.medium	1	Specified in launch template

Details

Node group ARN	Autoscaling group name	Capacity type	Subnets
arn:aws:eks:eu-west-1:038462791702:nodegroup/devops-cluster/devops-nger/4ec99e15-8bbe-0e34-2e99-606c1f0f59e1	eks-devops-nger-4ec99e15-8bbe-0e34-2e99-606c1f0f59e1	On-Demand	subnet-080af270e3d86a0c6, subnet-0e1b10422145f22d9, subnet-0a7a3d39e0d280c
Created	Node IAM role ARN	Desired size	Configure remote access to nodes
an hour ago	arn:aws:iam::038462791702:role/eksctl-devops-cluster-nodegroup-de-NodeInstanceRole-AaWOFW9oafy	2 nodes	off
	View in IAM	Minimum size	
		1 node	
		Maximum size	
		3 nodes	

Click **Add permissions**

The screenshot shows the AWS IAM Role details page for the role `eksctl-devops-cluster-nodegroup-de-NodeInstanceRole-Aa0W0FWs0afy`. The ARN is `arn:aws:iam::038462791702:role/eksctl-devops-cluster-nodegroup-de-NodeInstanceRole-Aa0W0FWs0afy`. The instance profile ARN is `arn:aws:iam::038462791702:instance-profile/eks-4ec99e15-8bbe-0e34-2e99-606c1f0f59e1`. The role was created on November 17, 2024, at 22:07 UTC, and it has been active for 12 minutes ago. The maximum session duration is 1 hour. There are four permissions policies attached to this role.

[Permissions](#) [Trust relationships](#) [Tags \(6\)](#) [Last Accessed](#) [Revoke sessions](#)

Permissions policies (4) [Info](#)

You can attach up to 10 managed policies.

Filter by Type

Policy name	Type	Attached entities
AmazonEC2ContainerRegistryReadOnly	AWS managed	1
AmazonEKS_CNI_Policy	AWS managed	1
AmazonEKSWorkerNodePolicy	AWS managed	1
AmazonSSMManagedInstanceCore	AWS managed	2

Click Attach Policies

[Permissions](#) [Trust relationships](#) [Tags \(6\)](#) [Last Accessed](#) [Revoke sessions](#)

Permissions policies (4) [Info](#)

You can attach up to 10 managed policies.

Filter by Type

Policy name	Type	Attached entities
AmazonEC2ContainerRegistryReadOnly	AWS managed	1
AmazonEKS_CNI_Policy	AWS managed	1
AmazonEKSWorkerNodePolicy	AWS managed	1
AmazonSSMManagedInstanceCore	AWS managed	2

▶ Permissions boundary (not set)

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

The screenshot shows the 'Attach policy' step for the role `eksctl-devops-cluster-nodegroup-de-NodeInstanceRole-Aa0W0FWs0afy`. It shows the current permissions policies (4) and the search results for the CloudWatchAgentServerPolicy (1/985). The policy is selected, and the 'Add permissions' button is visible.

The policy has now been successfully attached to the NodeGroup role



Policy was successfully attached to role.

eksctl-devops-cluster-nodegroup-de-NodeInstanceRole-Aa0W0FWs0afy [Info](#)

[Delete](#)

Summary

Creation date
November 17, 2024, 22:07 (UTC)

ARN
[arn:awsiam::038462791702:role/eksctl-devops-cluster-nodegroup-de-NodeInstanceRole-Aa0W0FWs0afy](#)

Instance profile ARN
[arn:awsiam::038462791702:instance-profile/eks-4ec99e15-8bbe-0e34-2e99-606c1f0f59e1](#)

Last activity
[16 minutes ago](#)

Maximum session duration
1 hour

[Edit](#)

[Permissions](#)

[Trust relationships](#)

[Tags \(6\)](#)

[Last Accessed](#)

[Revoke sessions](#)

Permissions policies (5) [Info](#)

You can attach up to 10 managed policies.

[Simulate](#) [Remove](#) [Add permissions](#)

Filter by Type

All types

<input type="checkbox"/> Policy name	Type	Attached entities
<input checked="" type="checkbox"/> AmazonEC2ContainerRegistryReadOnly	AWS managed	1
<input checked="" type="checkbox"/> AmazonEKS_CNI_Policy	AWS managed	1
<input checked="" type="checkbox"/> AmazonEKSWorkerNodePolicy	AWS managed	1
<input checked="" type="checkbox"/> AmazonSSMManagedInstanceCore	AWS managed	2
<input checked="" type="checkbox"/> CloudWatchAgentServerPolicy	AWS managed	1

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>**

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

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

```
mahendralselvakumar@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!*
mahendralselvakumar@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 aws-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-78fdb7s9vh7   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

The screenshot shows the CloudWatch Log Groups page. It lists three log groups under the 'Log groups' section. Each entry includes a checkbox, the log group name, 'Log class' (Standard), 'Configure' button, 'Retention' (Never expire), 'Metric filters' (None), and 'Contributor Insights' (None). The 'RDOSMetrics' entry has a retention of '1 month'.

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

The screenshot shows the Container Insights dashboard for the 'devops-cluster'. The left sidebar navigation includes 'Logs', 'Metrics', 'Events', 'Application Signals', 'Network monitoring', and 'Container Insights'. The main dashboard features several sections: 'Clusters state summary' (last updated November 17, 2024, 23:25 UTC), 'Performance and status summary' (Cluster CPU usage 2%, Reserved 30%, Clusters Memory usage 15%, Reserved 7%), 'Control plane summary' (Max API server requests 132, Average API server requests latency 90ms, Total number of stored objects 0, Average admission controller latency 0ms), and two line charts for 'Top 10 Pods per metric' and 'Top 10 Nodes per metric'. The 'Top 10 Pods' chart shows CPU Utilization over time, with the top pod being 'cloudwatch-agent-2bsqk'. The 'Top 10 Nodes' chart shows Memory Utilization over time, with the top node being 'ip-192-168-7-113.eu-west-1.compute.internal'.

Click Created cluster under the Cluster Overview in Container Insights

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



Clusters overview (1)

Find cluster

Name	Alarm state	Last state update	Max CPU %	Max Memory %
devops-cluster	No alarms detected	-	6%	17%

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

CloudWatch > Container Insights > Performance monitoring

Container Insights Service: EKS

1h 3h 12h 1d 3d 1w Custom UTC timezone Add to dashboard C View in maps

Performance dashboard views Info

Clusters Namespaces Nodes Services Workloads Pods Containers

More dashboard views Filters

By default only top 10 metrics are shown in the widgets on this dashboard. Use the filters to view a specific resource.

Cluster devops-cluster Namespace default Workload nginx-deployment Pod nginx-deployment-576c6b7b6-64pkj

Pods Performance monitoring - devops-cluster

Pods summary

Number of containers	Number of running containers	Pod CPU utilization	Pod memory utilization
15	15	0.55 %	2.25 %

Performance metrics

Pod CPU utilization	Pod CPU utilization over pod limit	Pod memory utilization	Pod memory utilization over pod limit
Percent 1 0.8 0.6 0.4 0.2 0	Percent 1 0.8 0.6 0.4 0.2 0	Percent 1 0.8 0.6 0.4 0.2 0	Percent 1 0.8 0.6 0.4 0.2 0

20:30 21:00 21:30 22:00 22:30 23:00

nginx-deployment-576c6b7b6-64pkj

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

CloudWatch > Container Insights > Performance monitoring

Container Insights Service: EKS

1h 3h 12h 1d 3d 1w Custom UTC timezone Add to dashboard C View in maps

Performance dashboard views Info

Clusters Namespaces Nodes Services Workloads Pods Containers

More dashboard views Filters

By default only top 10 metrics are shown in the widgets on this dashboard. Use the filters to view a specific resource.

Cluster devops-cluster Namespace default Workload nginx-deployment Pod nginx-deployment-576c6b7b6-64pkj Container nginx

Containers Performance monitoring - devops-cluster

Containers summary

CPU utilization	Memory utilization	Pod container status waiting	Page faults
0.38	1.63	0	543

Performance metrics

CPU utilization	Memory utilization
Percent 1 0.8 0.6 0.4 0.2 0	Percent 1 0.8 0.6 0.4 0.2 0

20:30 20:45 21:00 21:15 21:30 21:45 22:00 22:15 22:30 22:45 23:00 23:15

nginx-deployment-576c6b7b6-64pkj nginx

CPU utilization over container limit

Percent 1
0.8
0.6
0.4
0.2
0

No data available. Try adjusting the dashboard time range.

20:30 20:45 21:00 21:15 21:30 21:45 22:00 22:15 22:30 22:45 23:00 23:15

nginx-deployment-576c6b7b6-64pkj nginx 0.0770141759

Memory utilization over container limit

Percent 1
0.8
0.6
0.4
0.2
0

No data available. Try adjusting the dashboard time range.

20:30 20:45 21:00 21:15 21:30 21:45 22:00 22:15 22:30 22:45 23:00 23:15

nginx-deployment-576c6b7b6-64pkj nginx 0.0770141759



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**

The screenshot shows the AWS Billing and Cost Management home page. On the left, there's a sidebar with various navigation options like Home, Getting Started, Billing and Payments, Cost Analysis, Cost Organization, Budgets and Planning, Savings and Commitments, and Preferences and Settings. The main area has sections for Cost summary, Cost breakdown, Recommended actions, and Cost allocation coverage. The Recommended actions section includes a warning about Free Tier usage and a link to create a budget.

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

The screenshot shows the 'Cost Management Preferences' page under 'Billing and Cost Management'. It has tabs for General, Cost Explorer, and Cost Optimization Hub. The General tab is selected. In the 'Split cost allocation data' section, there are three options: 'Amazon Elastic Container Service (ECS)', 'Amazon Elastic Kubernetes Service (EKS)', and 'Amazon CloudWatch Container Insights'. The 'Amazon CloudWatch Container Insights' option is selected. At the bottom right, there is a 'Save preferences' button.



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**

The screenshot shows the 'Data Exports' section of the AWS Billing and Cost Management console. On the left, there's a sidebar with navigation links for Home, Getting Started, Billing and Payments, Bills, Payments, Credits, Purchase Orders, Cost Analysis (with sub-links for Cost Explorer, Cost Explorer Saved Reports, Cost Anomaly Detection, and Free Tier), and Data Exports. The main content area is titled 'Exports and dashboards' and contains a table with no data. The table has columns for 'Export name', 'Status', 'Export type', 'Data table', 'Date created', 'Date last refreshed', and 'S3 bucket'. At the bottom of the table area, there's a 'Create' button.

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

The screenshot shows the 'Create export' page. At the top, it says 'Billing and Cost Management > Data Exports > Create export'. Below that is a section titled 'Create export' with a 'Info' link. Under 'Export type', there are three options: 'Standard data export', 'Cost and usage dashboard powered by QuickSight', and 'Legacy CUR export' (which is selected). Below 'Export type' is a 'Export name' field containing 'Eks-cost'. At the bottom right of the page, there's a 'Create' button.

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

The screenshot shows the 'Export content' settings page. It includes sections for 'Default content' (listing items like Account identifiers, Invoice and Bill Information, Usage amount and unit, Rates and cost, Product attributes, Pricing attributes, and Reservation identifiers), 'Additional export content' (with checkboxes for 'Include resource IDs' and 'Split cost allocation data'), and 'Data refresh settings' (with a checkbox for 'Refresh automatically'). At the bottom right, there's a 'Create' button.



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

Data export delivery options
Define your delivery options including file format and overwrite rules.

Report data time granularity
Choose the time granularity for how you want the line items in the report to be aggregated.
 Hourly
 Daily
 Monthly

Report versioning
Choose the cadence that you want AWS to update the data export in your S3 bucket.
 Create new report version
Delivering new report versions can improve audibility of billing data over time.
 Overwrite existing report
Data export will be refreshed each time the source data updates. Minimum once per day, up to 3 times per day.

Report data integration
 Amazon Athena
 Amazon Redshift
 Amazon QuickSight

Compression type
ZIP

Click Configure to set up Data Export Storage

Data export storage settings
Enter a destination in Amazon S3 where your data export will be stored. Amazon S3 is object storage built to store and retrieve data.

S3 bucket
myawsbucket Configure

S3 path prefix
Enter the S3 path prefix that you want prepended to the name of your data export.
MyS3PathPrefix

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

Configure S3 bucket

Create a bucket or select an existing bucket
Data Exports will deliver your export to the S3 bucket that you specify.

Create a bucket Select existing bucket

S3 bucket name
eks-cost

Bucket name must be globally unique and must not contain spaces or uppercase letters. [Learn more](#)

Region
Europe (Ireland)

Bucket policy
This policy will be applied with the S3 bucket you specified above.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnableAWSDataExportsToWriteToS3AndCheckPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "billingreports.amazonaws.com",
          "bcm-data-exports.amazonaws.com"
        ]
      }
    }
  ]
}
```

Cancel Create bucket



Provide the S3 Path Prefix and click Create Report

Data export storage settings
Enter a destination in Amazon S3 where your data export will be stored. Amazon S3 is object storage built to store and retrieve data.

S3 bucket
devops-eks-cost Configure

S3 path prefix
Enter the S3 path prefix that you want prepended to the name of your data export.
eks-cost

Valid prefix names must not include trailing "/" or ":".

Tags - optional
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

Add new tag
You can add up to 50 more tags.

Cancel Create report

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

Export request "Eks-cost" has been successfully submitted
Your first export delivery to Amazon S3 will complete within the next 24 hours and then on a recurring basis according to the data export refresh cadence specified in your export. You can leave the page and check back after 24 hours to access your first export delivery.

Billing and Cost Management > Data Exports

Data Exports info
A data export tool that enables you to create customized exports from multiple AWS cost management and billing datasets.

Actions	Create					
Exports and dashboards <small>info</small>						
Find exports						
Export name	Status	Export type	Data table	Date created	Date last refreshed	S3 bucket
Eks-cost	In progress	Legacy CUR export	-	-	-	devops-eks-cost

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**



```
mahendranelvakumar@Mahendrants-MBP Downloads % eksctl delete cluster --name devops-cluster --region eu-west-1

2024-11-18 00:16:00 [i] deleting EKS cluster "devops-cluster"
2024-11-18 00:16:02 [i] will drain 0 unmanaged nodegroup(s) in cluster "devops-cluster"
2024-11-18 00:16:02 [i] starting parallel draining, max in-flight of 1
2024-11-18 00:16:02 [*] failed to acquire semaphore while waiting for all routines to finish: context canceled
2024-11-18 00:16:02 [i] deleted 0 Fargate profile(s)
2024-11-18 00:16:03 [✓] kubeconfig has been updated
2024-11-18 00:16:03 [i] cleaning up AWS load balancers created by Kubernetes objects of Kind Service or Ingress
2024-11-18 00:16:04 [i]
2 sequential tasks: { delete nodegroup "devops-ng", delete cluster control plane "devops-cluster" [async]
}
2024-11-18 00:16:04 [i] will delete stack "eksctl-devops-cluster-nodegroup-devops-ng"
2024-11-18 00:16:04 [i] waiting for stack "eksctl-devops-cluster-nodegroup-devops-ng" to get deleted
2024-11-18 00:16:04 [i] waiting for CloudFormation stack "eksctl-devops-cluster-nodegroup-devops-ng"
2024-11-18 00:16:34 [i] waiting for CloudFormation stack "eksctl-devops-cluster-nodegroup-devops-ng"
2024-11-18 00:17:21 [i] waiting for CloudFormation stack "eksctl-devops-cluster-nodegroup-devops-ng"
2024-11-18 00:18:30 [i] waiting for CloudFormation stack "eksctl-devops-cluster-nodegroup-devops-ng"
2024-11-18 00:22:24 [i] waiting for CloudFormation stack "eksctl-devops-cluster-nodegroup-devops-ng"
2024-11-18 00:23:51 [i] waiting for CloudFormation stack "eksctl-devops-cluster-nodegroup-devops-ng"
2024-11-18 00:25:47 [i] waiting for CloudFormation stack "eksctl-devops-cluster-nodegroup-devops-ng"
2024-11-18 00:25:47 [i] will delete stack "eksctl-devops-cluster-cluster"
2024-11-18 00:25:48 [✓] all cluster resources were deleted
mahendranelvakumar@Mahendrants-MBP Downloads %
```

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/>