# **Logging(EFK Stack):**

## Fluentd**:**

Fluentd allows you to unify data collection and consumption for a better use and understanding of data.

## ElasticSearch:

Elasticsearch is a distributed, RESTful search and analytics engine capable of solving a growing number of use cases.

## Kibana:

Kibana lets you visualize your Elasticsearch data and navigate the Elastic Stack. Kibana gives the freedom to select the way you give shape to your data.

### Configure IAM Policy:

The fluentd log daemon will collect logs and forward to CloudWatch Logs. This will require the nodes to have permissions to send logs and create log groups and log streams. This can be accomplished with an IAM user, IAM role

* Need to ensure the Role Name for the workers use the same set in the environment

“**test -n "$ROLE\_NAME" && echo ROLE\_NAME is "$ROLE\_NAME" || echo ROLE\_NAME is not set**”

**NOTE**: If the result is “**ROLE\_NAME is not set**” then one need to export the Role\_name with the below command

**“INSTANCE\_PROFILE\_NAME=$(aws iam list-instance-profiles | jq -r '.InstanceProfiles[].InstanceProfileName' | grep nodegroup)**

**INSTANCE\_PROFILE\_ARN=$(aws iam get-instance-profile --instance-profile-name $INSTANCE\_PROFILE\_NAME | jq -r '.InstanceProfile.Arn')**

**ROLE\_NAME=$(aws iam get-instance-profile --instance-profile-name $INSTANCE\_PROFILE\_NAME | jq -r '.InstanceProfile.Roles[] | .RoleName')**

**echo "export ROLE\_NAME=${ROLE\_NAME}" >> ~/.bash\_profile**

**echo "export INSTANCE\_PROFILE\_ARN=${INSTANCE\_PROFILE\_ARN}" >> ~/.bash\_profile”**

This will export the role to be used across the eks cluster

* IAM Policy to export logs to elastic search

**cat <<EoF > awapps-eks-logs.json**

**{**

**"Version": "2012-10-17",**

**"Statement": [**

**{**

**"Action": [**

**"logs:DescribeLogGroups",**

**"logs:DescribeLogStreams",**

**"logs:CreateLogGroup",**

**"logs:CreateLogStream",**

**"logs:PutLogEvents"**

**],**

**"Resource": "\*",**

**"Effect": "Allow"**

**}**

**]**

**}**

**EoF**

* Attach the policy to the EKS Cluster

**“aws iam put-role-policy --role-name $ROLE\_NAME --policy-name Logs-Policy-For-Worker --policy-document** [**file://~/environment/iam\_policy/awapps-eks-logs.json**](file://~/environment/iam_policy/awapps-eks-logs.json)**”**

* Verify if the policy is attached to the EKS cluster

**“aws iam get-role-policy --role-name $ROLE\_NAME --policy-name Logs-Policy-For-Worker”**

### ElasticSearch Cluster:

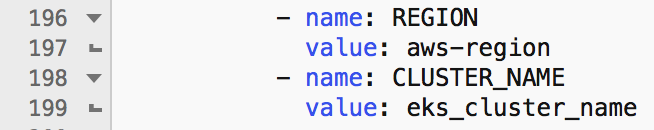
* AM Water had been setup to use the elastic search cluster in the shared account.

### Deploy FluentD:

* Download fluentd

**“ wget** [**https://eksworkshop.com/logging/deploy.files/fluentd.yml**](https://eksworkshop.com/logging/deploy.files/fluentd.yml) **“**

* Fluentd will be deployed as a DaemonSet, i.e. one pod per worker node
* Open the downloaded fluentd.yml file and update the Region and Cluster\_Name at lines 197 & 199



* Deploy the fluentd daemon on the eks cluster with the below command

“**kubectl apply -f fluentd.yml**”

* Check the status of the running pods with the command

“**kubectl get pods -w --namespace=kube-system**”

* Wait till all the pods are up and running. Once the pods are up and running we can see the logs in CloudWatch

### CloudWatch Logs Subscription:

* CloudWatch Logs can be delivered to other services such as Amazon Elasticsearch for custom processing. This can be achieved by subscribing to a real-time feed of log events. A subscription filter defines the filter pattern to use for filtering which log events gets delivered to Elasticsearch, as well as information about where to send matching log events to.
* Lambda IAM Execution Role

Following command will create an IAM Role which is required to push the logs from CloudWatch to elastic search cluster

**cat <<EoF > awapps\_eks\_logs\_lambda.json**

**{**

**"Version": "2012-10-17",**

**"Statement": [**

**{**

**"Effect": "Allow",**

**"Principal": {**

**"Service": "lambda.amazonaws.com"**

**},**

**"Action": "sts:AssumeRole"**

**}**

**]**

**}**

**EoF**

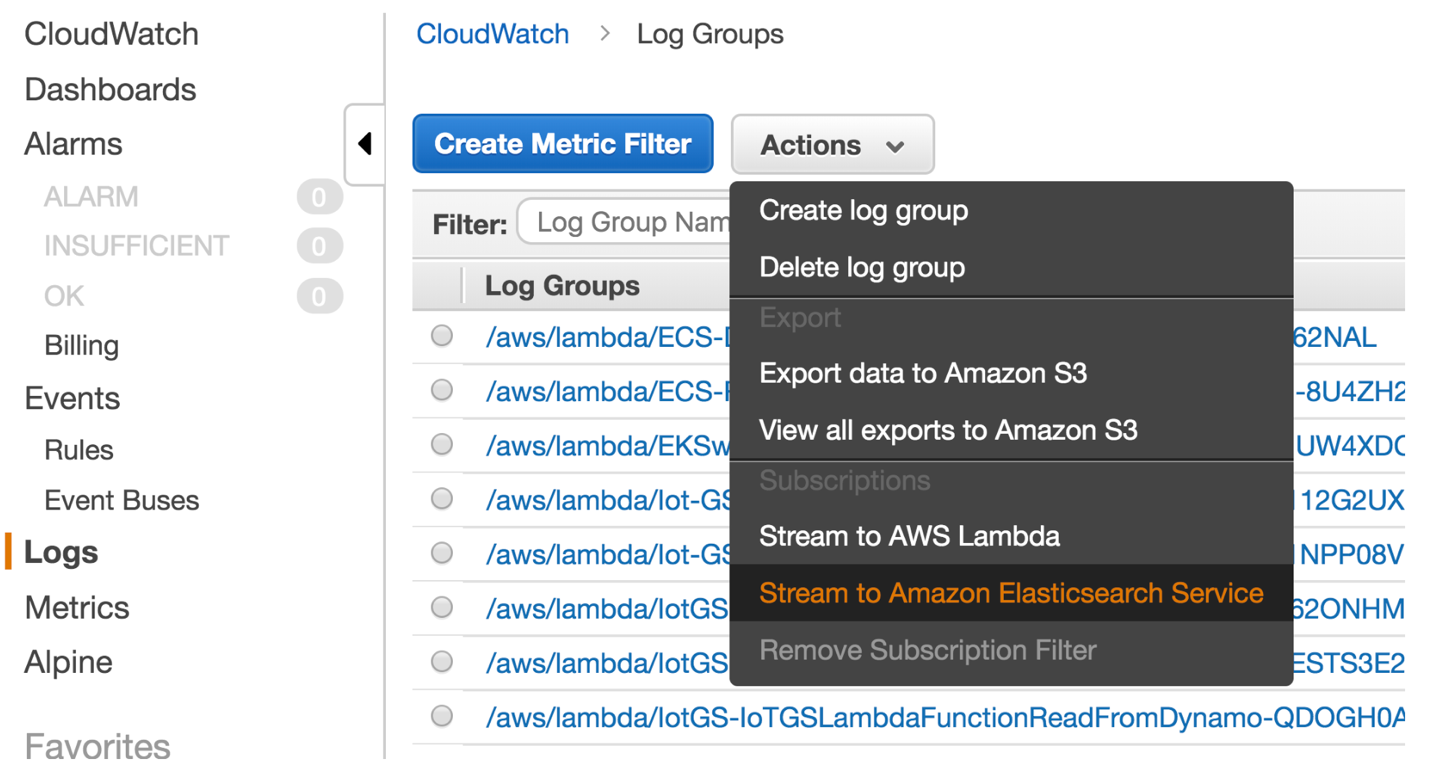
**aws iam create-role --role-name lambda\_basic\_execution --assume-role-policy-document file:awapps\_eks\_logs\_lambda**

* Attach the role policy with the below command

**“aws iam attach-role-policy --role-name lambda\_basic\_execution --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole”**

* Login to the CloudWatch Logs [console](https://console.aws.amazon.com/cloudwatch/home)

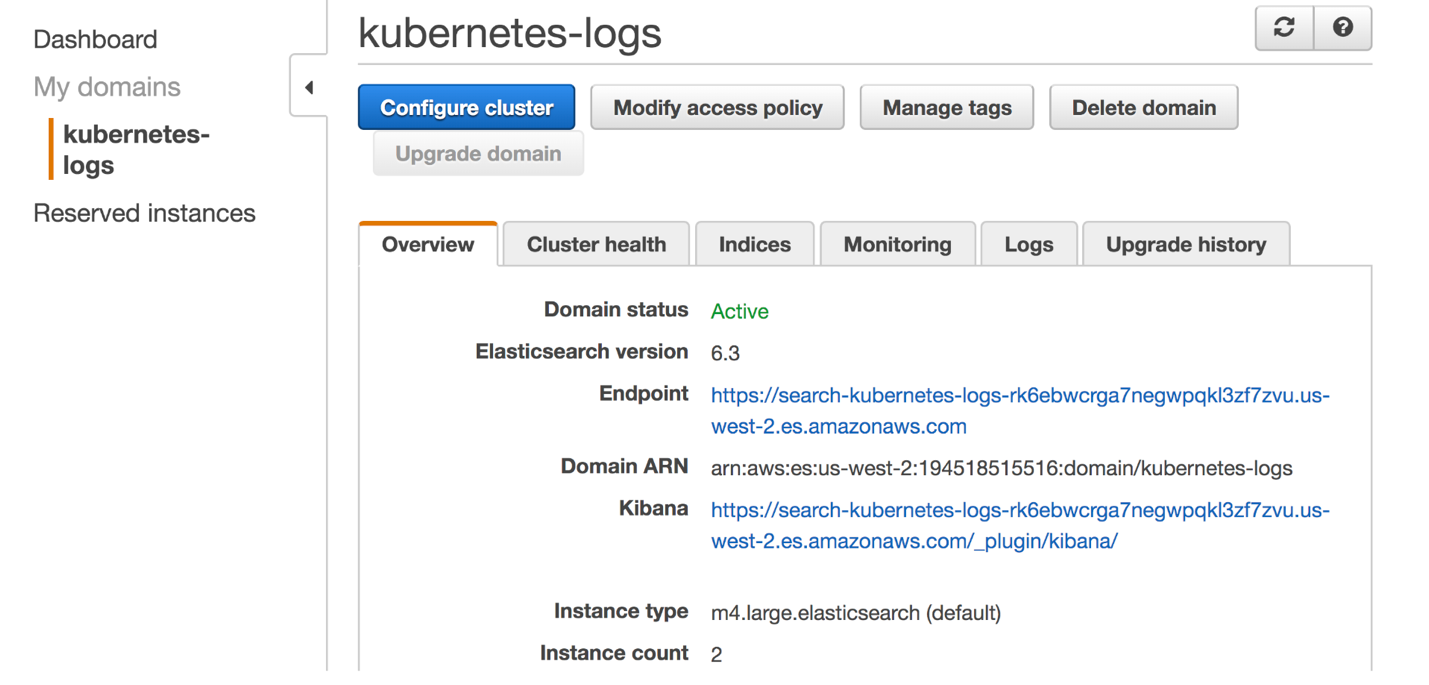
Select the log group similar to **/eks/ekscluster-eksctl/containers** click on **Actions** and select Stream to **Amazon Elastic Search Service**



* Select the cluster and IAM role and click next
* Select **Common Log Format** and click **Next**
* Review the configuration and click on **Start Streaming**
* CloudWatch page is refreshed to show that the filter was successfully created

Configure Kibana:

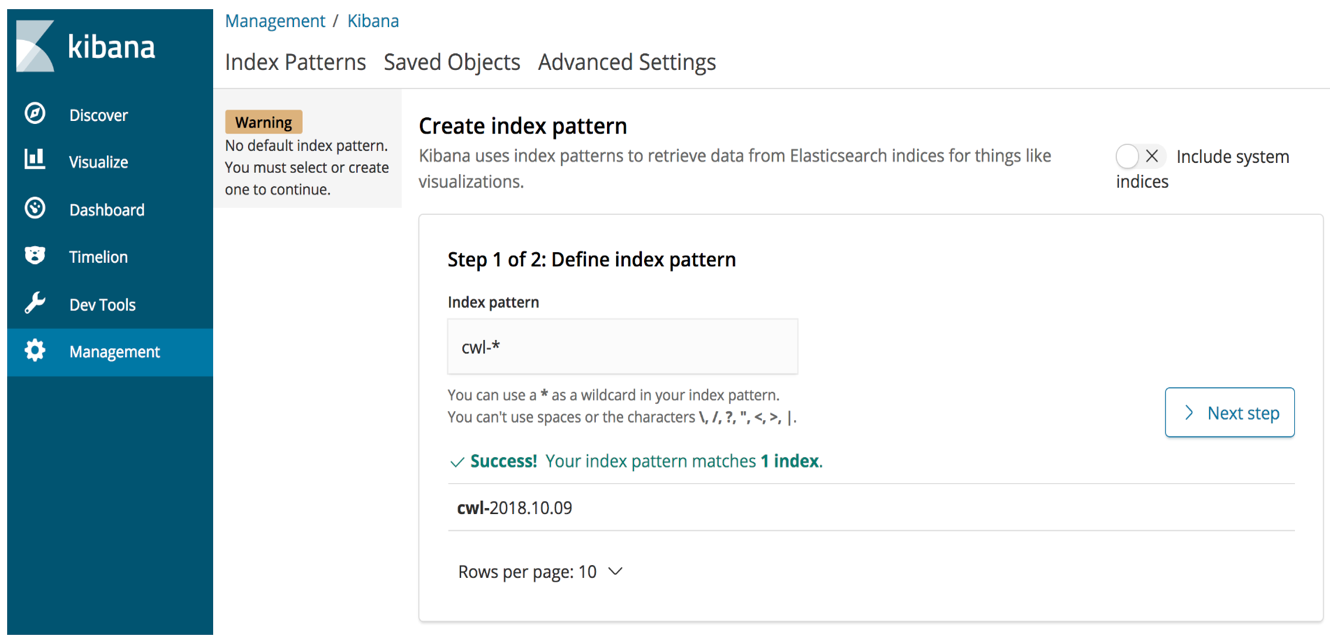
* Login to the AWS ElasticSearch [console](https://console.aws.amazon.com/es/home) and choose kubernetes-logs
* Open the Kibana dashboard from the link. After a few minutes, records will begin to be indexed by ElasticSearch. You’ll need to configure an index pattern in Kibana.



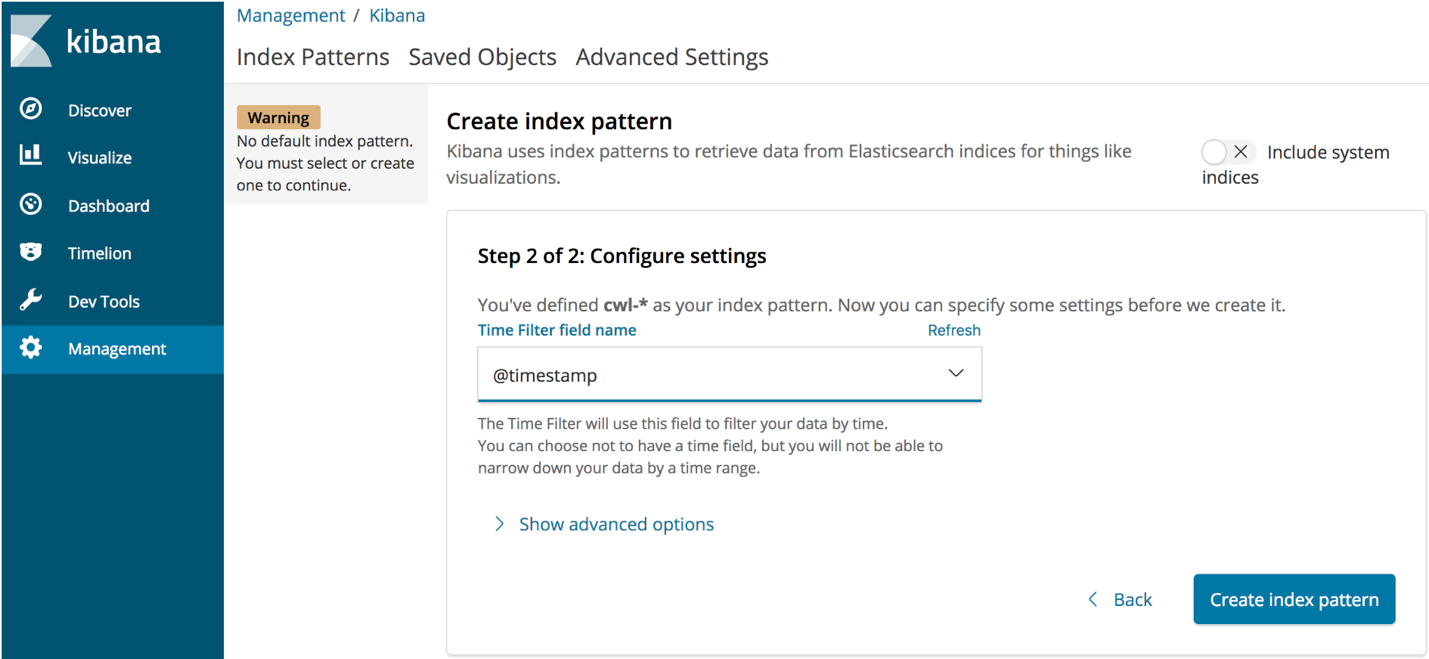
Kibana dashboard endpoint

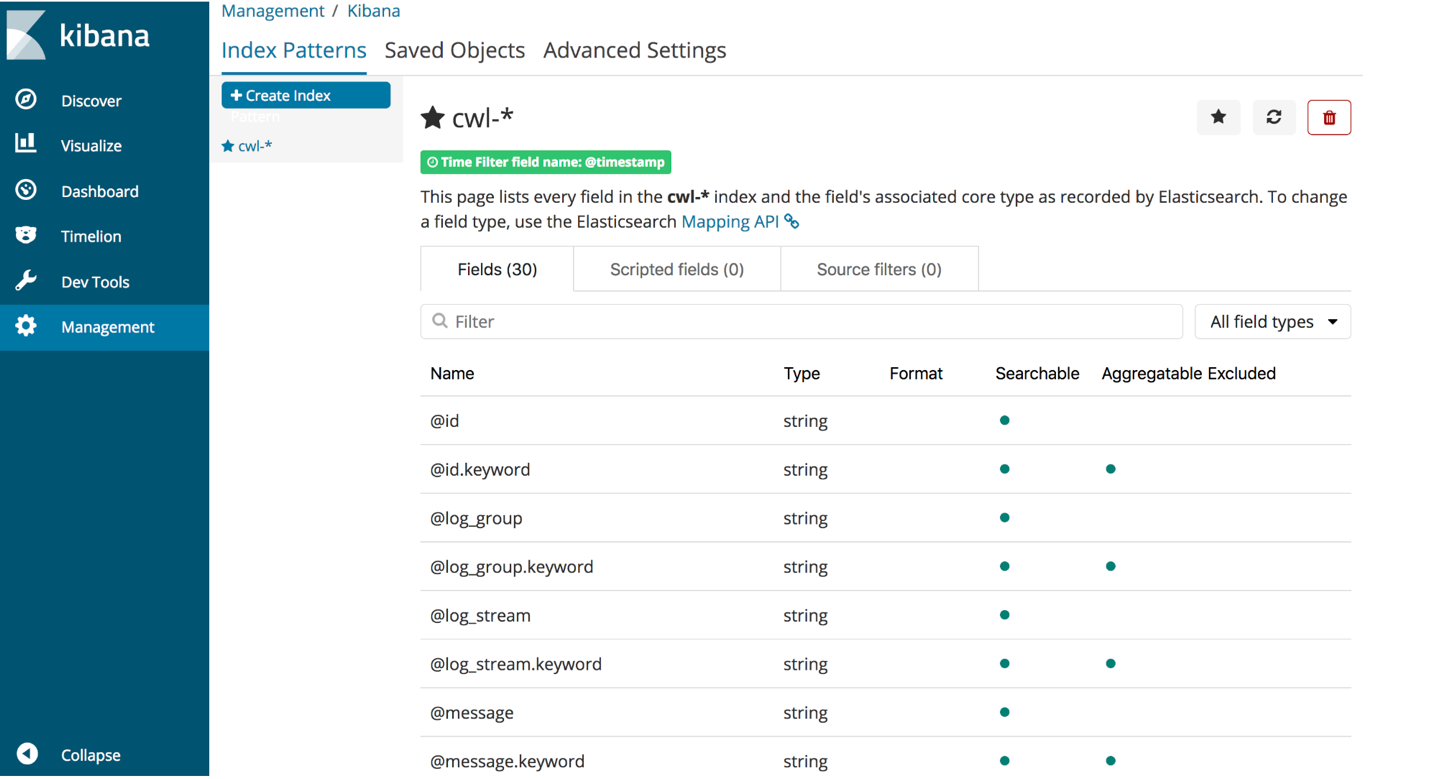
ElasticSearch cluster endpoint

* Click on index pattern as **cwl-\*** and click **Next**

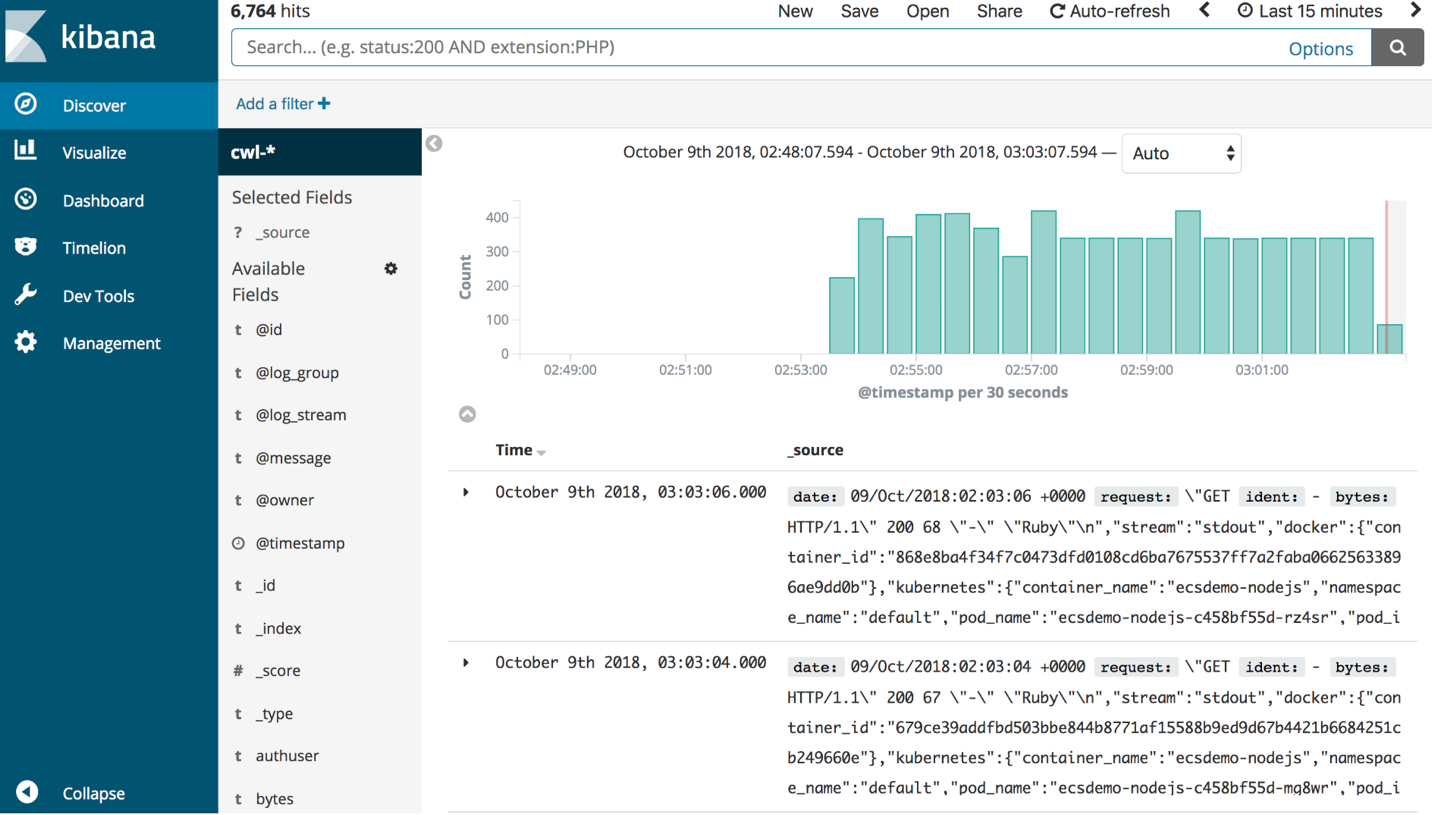


* Select **@timestamp** from dropdown list and select **Create index pattern**





Click on **Discover** and explore the logs



# **EKS Cluster Monitoring:**

## Prerequisites:

* Install helm

Helm helps you manage Kubernetes applications — Helm Charts help you define, install, and upgrade even the most complex Kubernetes application.

Install helm with the following steps

* + cd ~/environment
  + curl https://raw.githubusercontent.com/kubernetes/helm/master/scripts/get > get\_helm.sh
  + chmod +x get\_helm.sh
  + ./get\_helm.sh

**NOTE**: Do not run ‘helm init’ yet

* Configure helm with RBAC

**cat <<EoF > ~/environment/rbac.yaml**

**---**

**apiVersion: v1**

**kind: ServiceAccount**

**metadata:**

**name: tiller**

**namespace: kube-system**

**---**

**apiVersion: rbac.authorization.k8s.io/v1beta1**

**kind: ClusterRoleBinding**

**metadata:**

**name: tiller**

**roleRef:**

**apiGroup: rbac.authorization.k8s.io**

**kind: ClusterRole**

**name: cluster-admin**

**subjects:**

**- kind: ServiceAccount**

**name: tiller**

**namespace: kube-system**

**EoF**

* Now apply the configuration with the below command

**kubectl apply -f ~/environment/rbac.yaml**

* Now install tiller using helm tooling

**helm init --service-account tiller**

## Deploy Prometheus:

Prometheus collects metrics from monitored targets by scraping metrics HTTP endpoints on these targets. Since Prometheus also exposes data in the same manner about itself, it can also scrape and monitor its own health.

* Deploy Prometheus with the below commands
  + **kubectl create namespace Prometheus**
  + **helm install stable/prometheus \**

**--name prometheus \**

**--namespace prometheus \**

**--set alertmanager.persistentVolume.storageClass="io1" \**

**--set server.persistentVolume.storageClass="io1"**

**NOTE: Replace “gp2” with “io1” for better performance and production deployments**

* Copy the Prometheus end point to clipboard
* To check if the Prometheus components are deployed use the following command
  + **kubectl get all -n Prometheus**
* Access the application by port forwarding from 8080 to 9090 with the below command
  + **kubectl port-forward -n prometheus deploy/prometheus-server 8080:9090**
* In cloud9 click on Tools -> Preview -> Preview Running Applications. Scroll to the end and append ‘/targets’ to the URL
* A web UI will be displayed to show all the targets and metrics that are being monitored

## Deploy Grafana:

Grafana is an open source visualization tool that can be used on top of a variety of different data stores

* Deploy Grafana with the below commands
  + **kubectl create namespace Grafana**
  + **helm install stable/grafana \**

**--name grafana \**

**--namespace grafana \**

**--set persistence.storageClassName="io1" \**

**--set adminPassword="\*\*\*\*\*\*\*\*\*\*" \**

**--set datasources."datasources\.yaml".apiVersion=1 \**

**--set datasources."datasources\.yaml".datasources[0].name=Prometheus \**

**--set datasources."datasources\.yaml".datasources[0].type=prometheus \**

**--set datasources."datasources\.yaml".datasources[0].url=http://prometheus-server.prometheus.svc.cluster.local \**

**--set datasources."datasources\.yaml".datasources[0].access=proxy \**

**--set datasources."datasources\.yaml".datasources[0].isDefault=true \**

**--set service.type=LoadBalancer**

* Successful deployment can be checked with below command
  + **kubectl get all -n Grafana**
* Copy the Grafana URL and export the same with the below command
  + **export ELB=$(kubectl get svc -n grafana grafana -o jsonpath='{.status.loadBalancer.ingress[0].hostname}')**
  + **echo** [**http://$ELB**](http://$ELB)
* Login to the application with username as admin and get the hash password with below command
  + **kubectl get secret --namespace grafana grafana -o jsonpath="{.data.admin-password}" | base64 --decode ; echo**

**NOTE:** For the password as a part of best practices and future enhancements use SSO with Grafana