EKS Monitoring Labs

Prerequisites

For this module, we need to download the eksctl binary:

```
curl --silent --location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(
uname -s)_amd64.tar.gz" | tar xz -C /tmp

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

Confirm the eksctl command works:

```
eksctl version
```

Enable eksctl bash-completion

```
eksctl completion bash >> ~/.bash_completion
. /etc/profile.d/bash_completion.sh
. ~/.bash_completion
```

Set your environment :

```
export ACCOUNT_ID=$(aws sts get-caller-identity --output text --query
Account)
export AWS_REGION=us-west-2
export AZS=($(aws ec2 describe-availability-zones --query
'AvailabilityZones[].ZoneName' --output text --region $AWS_REGION))
export MASTER_KEY_ARN=$(aws kms create-key --query KeyMetadata.Arn
--output text)
```

Launch EKS

Create an eksctl deployment file (eksworkshop.yaml) use in creating your cluster using the following syntax:

```
cat << EOF > eksworkshop.yaml
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: eksworkshop-eksctl
  region: ${AWS_REGION}
  version: "1.17"

availabilityZones: ["${AZS[1]}", "${AZS[2]}", "${AZS[3]}"]

managedNodeGroups:
- name: nodegroup
  desiredCapacity: 3
  instanceType: t3.small
  ssh:
    enableSsm: true

# To enable all of the control plane logs, uncomment below:
# cloudWatch:
#  clusterLogging:
#    enableTypes: ["*"]

secretsEncryption:
  keyARN: ${MASTER_KEY_ARN}
EOF
```

Next, use the file you created as the input for the eksctl cluster creation.

```
eksctl create cluster -f eksworkshop.yaml
```

Test the Cluster

Confirm your nodes:

```
kubectl get nodes
```

Export the Worker Role Name for use throughout the workshop:

```
STACK_NAME=$(eksctl get nodegroup --cluster eksworkshop-eksctl -o json |
jq -r '.[].StackName')
ROLE_NAME=$(aws cloudformation describe-stack-resources --stack-name
$STACK_NAME | jq -r '.StackResources[] |
select(.ResourceType=="AWS::IAM::Role") | .PhysicalResourceId')
echo "export ROLE_NAME=${ROLE_NAME}" | tee -a ~/.bash_profile
```

Install WordPress

In your terminal you just need to run the following commands to deploy WordPress and its database.

```
# Create a namespace wordpress
kubectl create namespace wordpress-cwi

# Add the bitnami Helm Charts Repository
helm repo add bitnami https://charts.bitnami.com/bitnami

# Deploy WordPress in its own namespace
helm -n wordpress-cwi install understood-zebu bitnami/wordpress
```

This chart will create:

- Two persistent volumes claims..
- Multiple secrets.
- One StatefulSet for MariaDB.
- One Deployment for Wordpress.

You can follow the status of the deployment with this command

```
kubectl -n wordpress-cwi rollout status deployment
understood-zebu-wordpress
```

Accessing Wordpress

You'll need the URL for your WordPress site. This is easily accomplished by running the command below from your terminal window.

```
export SERVICE_URL=$(kubectl get svc -n wordpress-cwi
understood-zebu-wordpress --template "{{ range (index
.status.loadBalancer.ingress 0) }}{{.}}{{ end }}")

echo "Public URL: http://$SERVICE_URL/"
```

You should see the *Hello World* WordPress welcome page.

Preparing to Install Container Insights

Add the necessary policy to the IAM role for your worker nodes

In order for CloudWatch to get the necessary monitoring info, we need to install the CloudWatch Agent to our EKS Cluster.

First, we will need to ensure the Role Name our workers use is set in our environment:

```
test -n "$ROLE_NAME" && echo ROLE_NAME is "$ROLE_NAME" || echo ROLE_NAME
is not set
```

We will attach the policy to the nodes IAM Role:

```
aws iam attach-role-policy \
  --role-name $ROLE_NAME \
  --policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy
```

Finally, let's verify that the policy has been attached to the IAM ROLE:

```
aws iam list-attached-role-policies --role-name $ROLE_NAME | grep
CloudWatchAgentServerPolicy || echo 'Policy not found'
```

Output

```
"PolicyName": "CloudWatchAgentServerPolicy",
"PolicyArn": "arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy"
```

Now we can proceed to the actual install of the CloudWatch Insights.

To complete the setup of Container Insights, you can follow the quick start instructions in this section.

From your Terminal you will just need to run the following command.

```
curl -s
https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-containe
r-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daem
onset/container-insights-monitoring/quickstart/cwagent-fluentd-quickstar
t.yaml | sed
"s/{{cluster_name}}/eksworkshop-eksctl/;s/{{region_name}}/${AWS_REGION}/
" | kubectl apply -f -
```

The command above will:

- Create the Namespace amazon-cloudwatch.
- Create all the necessary security objects for both DaemonSet:
    - SecurityAccount.
    - ClusterRole.
    - ClusterRoleBinding.
- Deploy Cloudwatch-Agent (responsible for sending the **metrics** to CloudWatch) as a DaemonSet.
- Deploy fluentd (responsible for sending the **logs** to Cloudwatch) as a DaemonSet.
- Deploy ConfigMap configurations for both DaemonSets.

You can find the full information and manual install steps here.

You can verify all the DaemonSets have been deployed by running the following command.

```
kubectl -n amazon-cloudwatch get daemonsets
```

Verify CloudWatch Container Insights is working

To verify that data is being collected in CloudWatch, launch the CloudWatch Containers UI in your browser using the link generated by the command below

```
echo "
Use the URL below to access Cloudwatch Container Insights in
$AWS_REGION:

https://console.aws.amazon.com/cloudwatch/home?region=${AWS_REGION}#cw:d
ashboard=Container;context=~(clusters~'eksworkshop-eksctl~dimensions~(~)
~performanceType~'Service)"
```

From here you can see the metrics are being collected and presented to CloudWatch. You can switch between various drop downs to see EKS Services, EKS Cluster and more.

Preparing your Load Test

Install Siege for load testing on your Workspace

```
sudo yum install siege -y
```

Verify Siege is working by typing the below into your terminal window.

```
siege --version
```

Running the Load Test

Run Siege to Load Test your Wordpress Site

Now that Siege has been installed, we're going to generate some load to our Wordpress site and see the metrics change in CloudWatch Container Insights.

From your terminal window, run the following command.

```
export WP_ELB=$(kubectl -n wordpress-cwi get svc
understood-zebu-wordpress -o
jsonpath="{.status.loadBalancer.ingress[].hostname}")

siege -v -t 15S -c 200 -i http://${WP_ELB}
```

This command tells Siege to run 200 concurrent connections to your Wordpress site at varying URLS for 15 seconds.

Viewing our collected metrics

Now let's navigate back to CloudWatch Container Insights browser tab to view the data we've generated.

From here you can choose a number of different views. We're going to narrow down our timelines to a custom time range of just 30 minute so we can zoom into our recently collected insights.

To do so go to the Time Range option at the top right of The CloudWatch Container Insights windows and selecting 30 minutes.

Once zoomed in on the time frame we can see the large spike in resource usage for the load we just generated to the Wordpress service in our EKS Cluster.

As mentioned previous you can view some different metrics based on the Dropdown menu options. Let's take a quick look at some of those items.

Viewing our collected logs

Now that we have a good understanding of the load, let's explore the logs generated by WordPress and sent to Cloudwatch by the *Fluentd* agent.

From the CloudWatch Container Insights browser tab:

- Scroll down to the *Pod performance* section.
- Select the WordPress pod.
- Select *application logs* from the *Action* menu.

The last action will open the *CloudWatch Logs Insights* UI in another tab.

Clic*Fluentd* has split the JSON files into multiple fields that could be easily parsed for debugging or to be included into Custom Application Dashboard.k the *Run query* button and expand one of log line to look at it.

CloudWatch Logs Insights enables you to explore, analyze, and visualize your logs instantly, allowing you to troubleshoot operational problems with ease. You can learn more about CloudWatch Logs Insights here.

Using CloudWatch Alarms

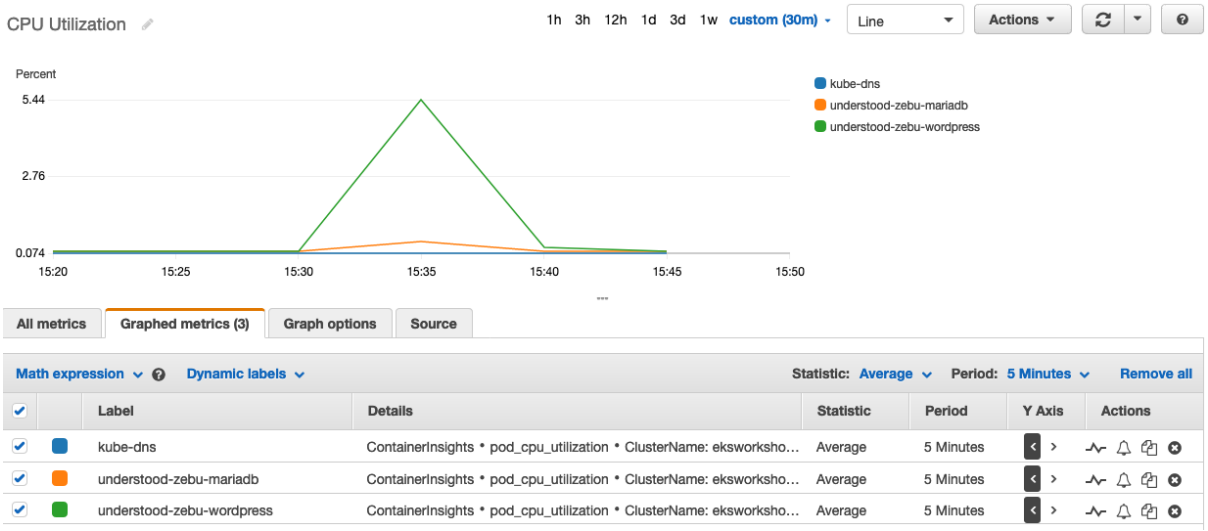
You can use the CloudWatch metrics to generate various alarms for your EKS Cluster based on assigned metrics.

In *CloudWatch Container Insights* we're going to drill down to create an alarm using CloudWatch for CPU Utilization of the Wordpress service.

To do so:

- Click on the three vertical dots in the upper right of the CPU Utilization box.
- Select View in Metrics.

This will isolate us to a single pane view of CPU Utilization for the eksworkshop-eksctl cluster.



From this window we can create alarms for the understood-zebu-wordpress service so we know when it's under heavy load.

For this lab we're going to set the threshold low so we can guarantee to set it off with the load test.

To create an alarm, click on the small bell icon in line with the Wordpress service. This will take you to the metrics alarm configuration screen.

Cleanup your Environment

Let's clean up Wordpress so it's not running in your cluster any longer.

```
helm -n wordpress-cwi uninstall understood-zebu
```

```
kubectl delete namespace wordpress-cwi
```

Run the following command to delete Container Insights from your cluster.

```
curl -s
https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-containe
r-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daem
onset/container-insights-monitoring/quickstart/cwagent-fluentd-quickstar
t.yaml | sed
"s/{{cluster_name}}/eksworkshop-eksctl/;s/{{region_name}}/${AWS_REGION}/
" | kubectl delete -f -
```

Delete the SNS topic and the subscription.

```
# Delete the SNS Topic
aws sns delete-topic \
  --topic-arn
arn:aws:sns:${AWS_REGION}:${ACCOUNT_ID}:wordpress-CPU-Alert

# Delete the subscription
aws sns unsubscribe \
  --subscription-arn $(aws sns list-subscriptions | jq -r
'.Subscriptions[].SubscriptionArn')
```

Finally we will remove the *CloudWatchAgentServerPolicy* policy from the Nodes IAM Role

```
aws iam detach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
  --role-name ${ROLE_NAME}
```

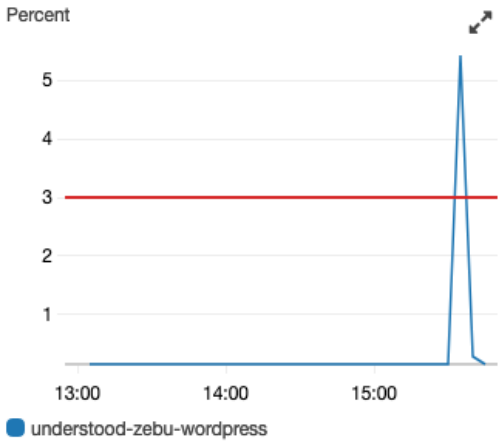
Specify metric and conditions

Metric

Edit

Graph

This alarm will trigger when the blue line goes above the red line for 1 datapoints within 5 minutes.



Namespace

ContainerInsights

Metric name

pod_cpu_utilization

ClusterName

eksworkshop-eksctl

Service

understood-zebu-wordpress

Namespace

wordpress-cwi

Statistic

Average

Period

5 minutes

Conditions

Threshold type

- (•) Static — Use a value as a threshold
- ( ) Anomaly detection — Use a band as a threshold

Whenever pod_cpu_utilization is...

Define the alarm condition.

- (•) Greater — > threshold
- ( ) Greater/Equal — >= threshold
- ( ) Lower/Equal — <= threshold
- ( ) Lower — < threshold

than...

Define the threshold value.

3

Must be a number

▶ Additional configuration

Cancel Next

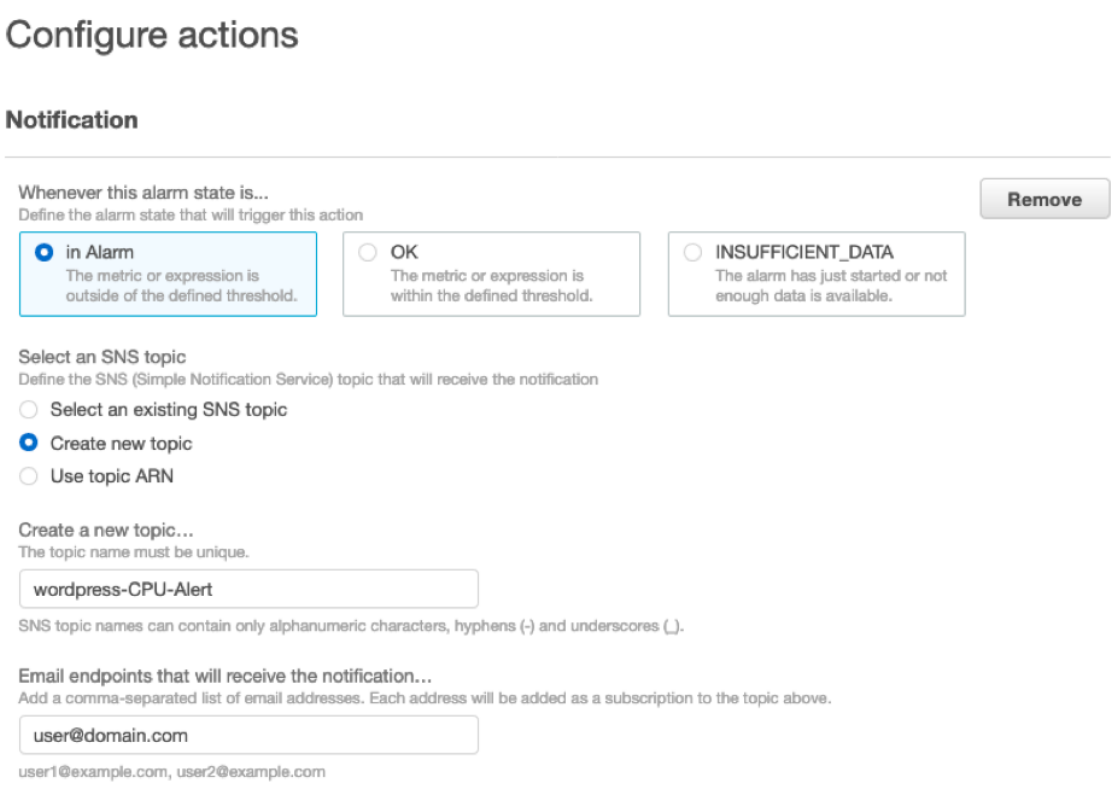
As we can see from the screen we peaked CPU at over 6 % so we're going to set our metric to 3% to assure it sets off an alarm. Set your alarm to 50% of whatever you max was during the load test on the graph.

Click next on the bottom and continue to *Configure Actions*.

We're going to create a configuration to send an SNS alert to your email address when CPU gets above your threshold.

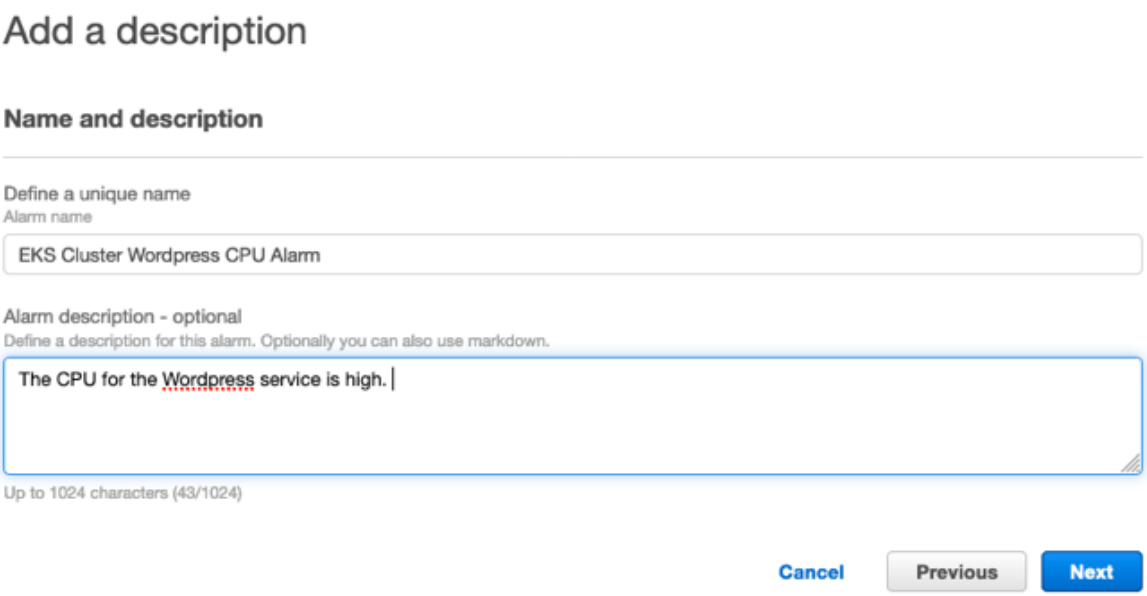
On the Configure Action screen:

- Leave default of *in Alarm*.
- Select *Create new topic* under Select and SNS Topic.
- In *Create new topic…* name it wordpress-CPU-alert.
- In Email Endpoints enter your email address.
- Click create topic.



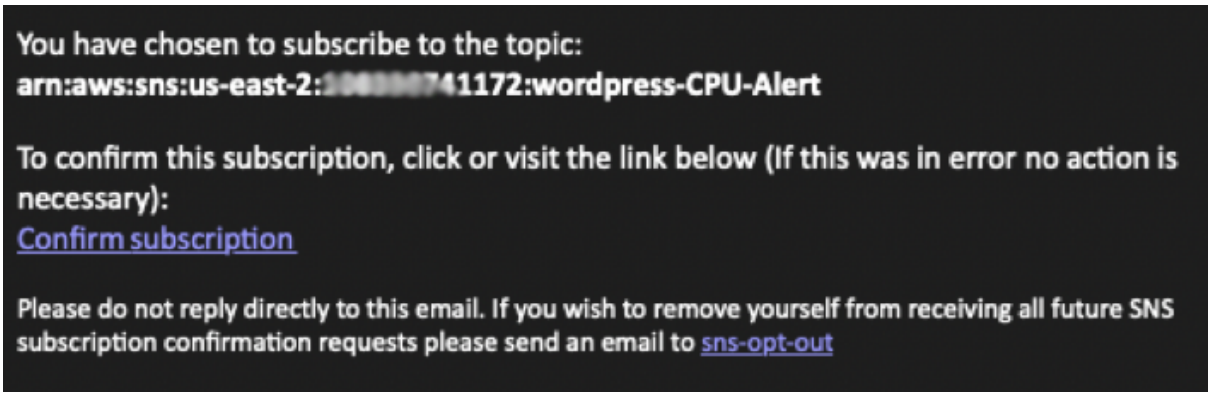
Once those items are set, you can click Next at the bottom of the screen.

On the next screen we'll add a unique name for our alert, and press Next.



The next screen will show your metric and the conditions. Make sure to click create alarm.

After creating your new SNS topic you will need to verify your subscription in your email.



Testing your alarm

For the last step of this lab, we're going to run one more load test on our site to verify our alarm triggers. Go back to your terminal and run the same commands we can previously to load up our Wordpress site.

i.e.

```
export WP_ELB=$(kubectl -n wordpress-cwi get svc
understood-zebu-wordpress -o
jsonpath="{.status.loadBalancer.ingress[].hostname}")

siege -q -t 15S -c 200 -i ${WP_ELB}
```

In a minute or two, you should receive and email about your CPU being in alert. If you don't verify your SNS topic configuration and that you've accepted the subscription to the topic.

You are receiving this email because your Amazon CloudWatch Alarm "EKS Cluster Wordpress CPU Alarm" in the US East (Ohio) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [12.228512401024775 (31/10/19 19:08:00)] was greater than the threshold (3.0) (minimum 1 datapoint for OK -> ALARM transition)." at "Thursday 31 October, 2019 19:13:10 UTC".

View this alarm in the AWS Management Console:
https://us-east-2.console.aws.amazon.com/cloudwatch/home?region=us-east-2#s=Alarms&alarm=EKS%20Cluster%20Wordpress%20CPU%20Alarm

Alarm Details:
- Name: EKS Cluster Wordpress CPU Alarm
- Description: The CPU for the Wordpress service is high.
- State Change: INSUFFICIENT_DATA -> ALARM
- Reason for State Change: Threshold Crossed: 1 out of the last 1 datapoints [12.228512401024775 (31/10/19 19:08:00)] was greater than the threshold (3.0) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp: Thursday 31 October, 2019 19:13:10 UTC
- AWS Account: 1[redacted]1172

Threshold:
- The alarm is in the ALARM state when the metric is GreaterThanThreshold 3.0 for 300 seconds.

Monitored Metric:
- MetricNamespace: ContainerInsights
- MetricName: pod_cpu_utilization
- Dimensions: [Service = understood-zebu-wordpress] [ClusterName = eksworkshop-eksctl] [Namespace = default]
- Period: 300 seconds
- Statistic: Average
- Unit: not specified
- TreatMissingData: missing

State Change Actions:
- OK:
- ALARM: [arn:aws:sns:us-east-2:1[redacted]741172:wordpress-CPU-Alert]
- INSUFFICIENT_DATA:

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-2.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-2:1[redacted]741172:wordpress-CPU-Alert:5b3e55b7-62d5-4e5d-9246-bffcd99c559d&Endpoint=r[redacted]azon.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

Cleanup your Environment

Let's clean up Wordpress so it's not running in your cluster any longer.

```
helm -n wordpress-cwi uninstall understood-zebu

kubectl delete namespace wordpress-cwi
```

Run the following command to delete Container Insights from your cluster.

```
curl -s
https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-containe
r-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daem
onset/container-insights-monitoring/quickstart/cwagent-fluentd-quickstar
t.yaml | sed
"s/{{cluster_name}}/eksworkshop-eksctl/;s/{{region_name}}/${AWS_REGION}/
" | kubectl delete -f -
```

Delete the SNS topic and the subscription.

```
# Delete the SNS Topic
aws sns delete-topic \
  --topic-arn
arn:aws:sns:${AWS_REGION}:${ACCOUNT_ID}:wordpress-CPU-Alert

# Delete the subscription
aws sns unsubscribe \
  --subscription-arn $(aws sns list-subscriptions | jq -r
'.Subscriptions[].SubscriptionArn')
```

Finally we will remove the *CloudWatchAgentServerPolicy* policy from the Nodes IAM Role

```
aws iam detach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
  --role-name ${ROLE_NAME}
```