# Efficient and Resilient Workloads with Amazon EC2 Auto Scaling

1. You can view and [download the CloudFormation template](https://raw.githubusercontent.com/awslabs/ec2-spot-workshops/master/content/efficient-and-resilient-ec2-auto-scaling/files/efficient-auto-scaling-quickstart-cnf.yml) from GitHub. Take a moment to review the CloudFormation template so you understand the resources it will be creating.

***Tip:****Right click the link and****'Save Link As'****, to download the file*

1. Browse to the [AWS CloudFormation](https://console.aws.amazon.com/cloudformation) console and click **'Create stack'**, then **'With new resources(standard)'**.
2. In the **'Specify template'** section, select **'Upload a template file'**. Click **'Choose file'** and select the template you downloaded in step 1 and click **'Next'**.
3. In the **'Specify stack details'** section, enter a Stack name and click **'Next'**. (Tip: The stack name cannot contain spaces, use **myEC2Workshop** for example.)
4. In **'Configure stack options'**, you don’t need to make any changes and click **'Next'**.
5. Review the information for the stack. At the bottom under **'Capabilities'**, select **'I acknowledge that AWS CloudFormation might create IAM resources'**. When you’re satisfied with the settings, click **'Create'** stack.

# 

# 

# Log into the AWS cloud9 IDE

1. In the AWS [CloudFormation](https://console.aws.amazon.com/cloudformation/home) console, select the **efficient-auto-scaling-quickstart-cnf** (or the name you used if you built it yourself) stack in the list.
2. In the stack details pane, click the **'Events'** tab.
3. Click the refresh icon to update the events in the stack creation.
4. When AWS CloudFormation has successfully created the stack you will see **CREATE\_COMPLETE**
5. In the stack details pane, click the **'Outputs'** tab.
6. Click on the url of the AWS Cloud9 environment; it’s the value of **Cloud9 IDE** in the CloudFormation stack outputs.

# 

# In side the cloud9 IDE

1. In the Cloud9 IDE, check the folders in left navigation. Look for folder named **ec2-spot-workshops** and confirm it exists.
2. In the Cloud9 terminal, change into the workshop directory

cd ec2-spot-workshops/workshops/efficient-and-resilient-ec2-auto-scaling

# 

# Amazon EC2 Auto Scaling

1. You will create an Auto Scaling group using the **asg.json** configuration file, open the file to review the configuration.

**cat ./asg.json**

1. Create the Auto Scaling group using the command below. This command does not return any output if it is successful.

**aws autoscaling create-auto-scaling-group --cli-input-json file://asg.json**

1. Then run this command to enable CloudWatch metrics collection for the Auto Scaling group, which will help you in monitoring the capacity in the group.

**aws autoscaling enable-metrics-collection \**

**--auto-scaling-group-name ec2-workshop-asg \**

**--granularity "1Minute"**

1. Browse to the [Auto Scaling console](https://console.aws.amazon.com/ec2/home#AutoScalingGroups:) and check out your newly created auto scaling group. At this step of the workshop, the auto scaling group will have no instances running, as the desired number of instances is set to 0.

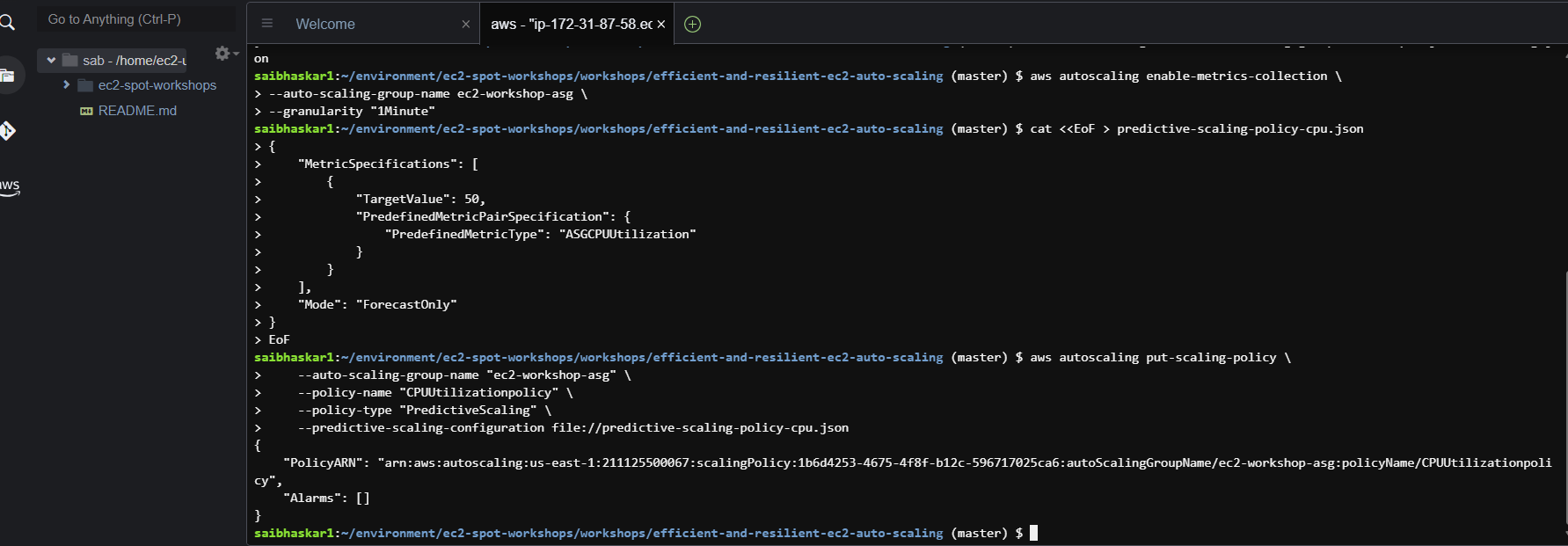
# 

# 

# Pridictive scaling

1. In the Cloud9 terminal, run this command to create a simple configuration file for a predictive scaling policy based on a predefined metrics pair.
2. **cat <<EoF > predictive-scaling-policy-cpu.json**
3. **{**
4. **"MetricSpecifications": [**
5. **{**
6. **"TargetValue": 50,**
7. **"PredefinedMetricPairSpecification": {**
8. **"PredefinedMetricType": "ASGCPUUtilization"**
9. **}**
10. **}**
11. **],**
12. **"Mode": "ForecastOnly"**
13. **}**

**EoF**

1. Create the predictive scaling policy and attach it to the Auto Scaling group.
2. **aws autoscaling put-scaling-policy \**
3. **--auto-scaling-group-name "ec2-workshop-asg" \**
4. **--policy-name "CPUUtilizationpolicy" \**
5. **--policy-type "PredictiveScaling" \**
6. **--predictive-scaling-configuration** [**file://predictive-scaling-policy-cpu.json**](file://predictive-scaling-policy-cpu.json)
7. 

# Verify predicting scaling policy in AWS console

1. Navigate to the [Auto Scaling console](https://console.aws.amazon.com/ec2/home#AutoScalingGroups:), click on Auto Scaling group **ec2-workshop-asg**
2. Click on the tab **'Automatic scaling'**

# 

# Working with Custom Metrics

As mentioned in the previous chapter, you wouldn't normally have 24 hours data for predictive scaling to start forecasting immediately after creating the policy. For the benefit of this lab, as part of the included CloudFormation stack, a script has been executed to create and populate two CloudWatch custom metrics which can be used in creating the predictive scaling policy.

## [Verify in CloudWatch metrics using AWS Console](https://catalog.us-east-1.prod.workshops.aws/workshops/20c57d32-162e-4ad5-86a6-dff1f8de4b3c/en-US/30-predictive/31-metrics" \l "verify-in-cloudwatch-metrics-using-aws-console)

Verify scaling and load metrics data in CloudWatch.

1. Navigate to [Amazon CloudWatch Console](https://console.aws.amazon.com/cloudwatch).
2. Make sure the correct region is selected in the AWS Console.
3. From left side navigation, click on **'Metrics'** then **'All metrics'**.
4. In the **'Browse'** tab select **'EC2 Workshop Custom Metrics'** under Custom namespaces
5. Select **'AutoScalingGroupName'**, then select the two metrics attached with **ec2-workshop-asg**. This should add the metrics to the graph.
6. To view all metrics data, from the time window filter select **'3d'** to view data of the last 3 days
7. Note the workload pattern in the custom metrics graph, this makes it a good use case for predictive scaling.

# 

# Create a Policy

Following up with your scenario, one of the requirements is to reduce the amount of time the application takes to become ready.

## [Create the predictive scaling policy](https://catalog.us-east-1.prod.workshops.aws/workshops/20c57d32-162e-4ad5-86a6-dff1f8de4b3c/en-US/30-predictive/32-policies" \l "create-the-predictive-scaling-policy)

1. In the Cloud9 terminal, check you’re in the directory **ec2-spot-workshops/workshops/efficient-and-resilient-ec2-auto-scaling**
2. Review the policy configuration file and note how the custom metrics have been used in it.

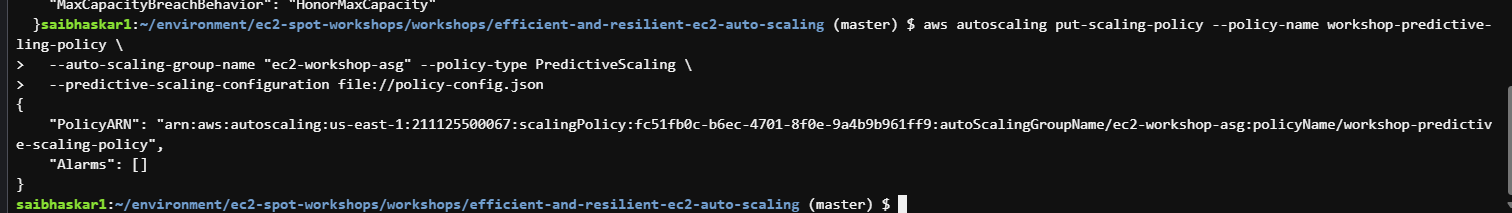
**cat ./policy-config.json**

1. Run this command to create the policy with the custom metrics and attach it to the Auto Scaling group.

**aws autoscaling put-scaling-policy --policy-name workshop-predictive-scaling-policy \**

**--auto-scaling-group-name "ec2-workshop-asg" --policy-type PredictiveScaling \**

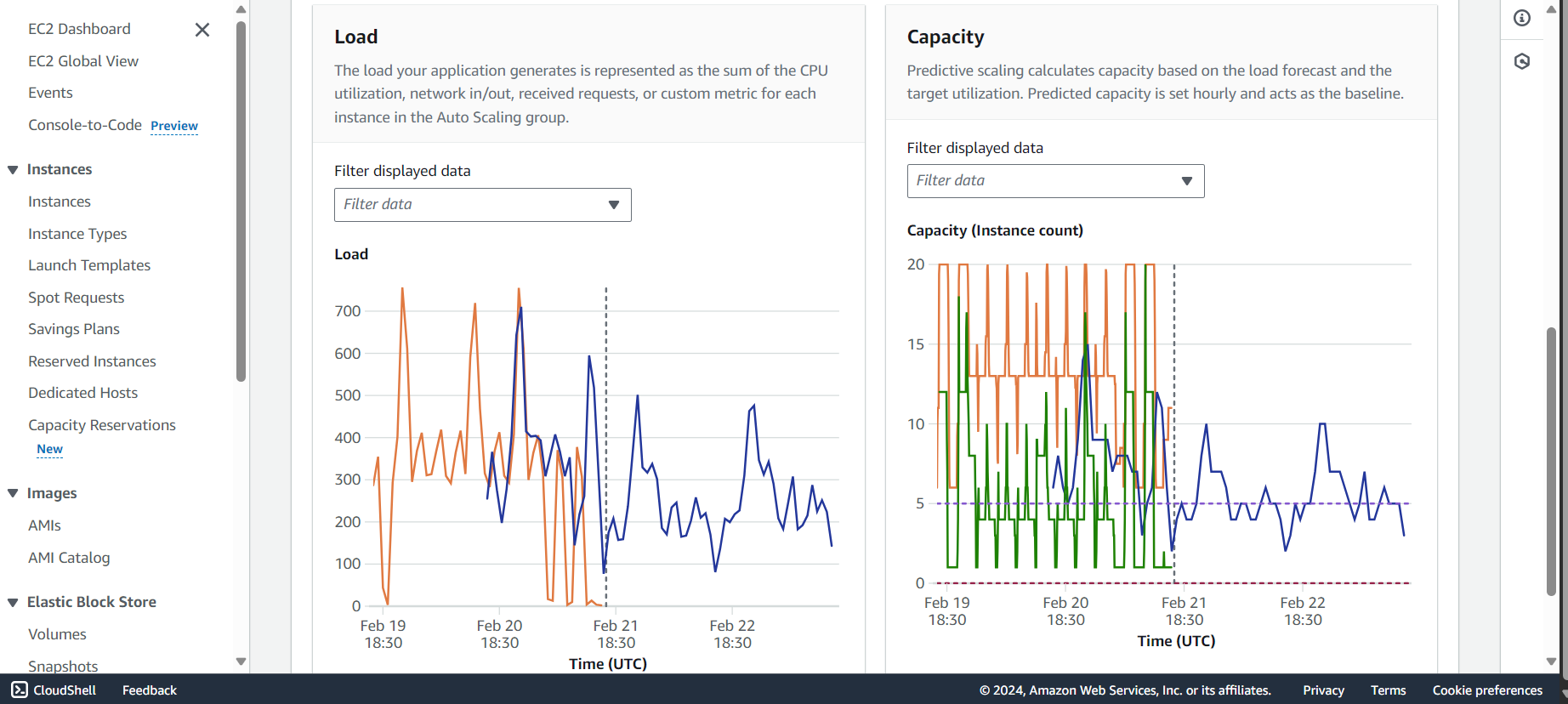
**--predictive-scaling-configuration file://policy-config.json**

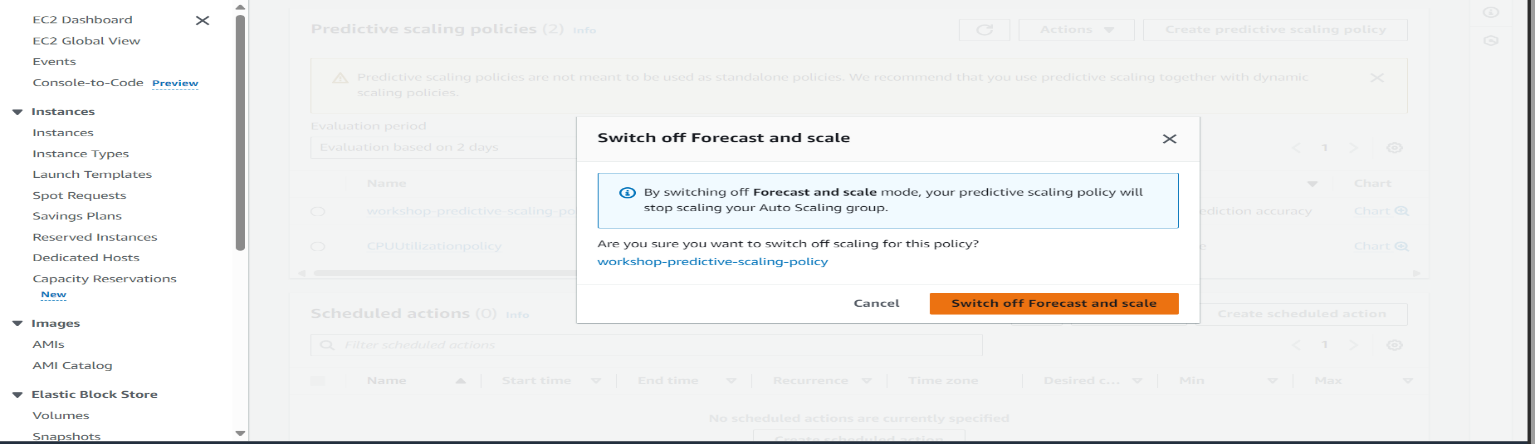


# Review the Outcome

## [Verify predictive scaling policy in AWS Console](https://catalog.us-east-1.prod.workshops.aws/workshops/20c57d32-162e-4ad5-86a6-dff1f8de4b3c/en-US/30-predictive/33-outcome" \l "verify-predictive-scaling-policy-in-aws-console)

1. Navigate to the [Auto Scaling console](https://console.aws.amazon.com/ec2/home#AutoScalingGroups:), click on Auto Scaling group **ec2-workshop-asg**
2. Click on the tab **'Automatic scaling'**
3. A new policy has been created under **'Predictive scaling policies'**





# Dynamic Scaling

1. **Review this command to understand the options, then run it**
2. **cat <<EoF > asg-automatic-scaling.json**
3. **{**
4. **"AutoScalingGroupName": "ec2-workshop-asg",**
5. **"PolicyName": "automaticScaling",**
6. **"PolicyType": "TargetTrackingScaling",**
7. **"EstimatedInstanceWarmup": 300,**
8. **"TargetTrackingConfiguration": {**
9. **"PredefinedMetricSpecification": {**
10. **"PredefinedMetricType": "ASGAverageCPUUtilization"**
11. **},**
12. **"TargetValue": 75,**
13. **"DisableScaleIn": false**
14. **}**
15. **}**
16. **EoF**
17. **Apply the scaling policy**:

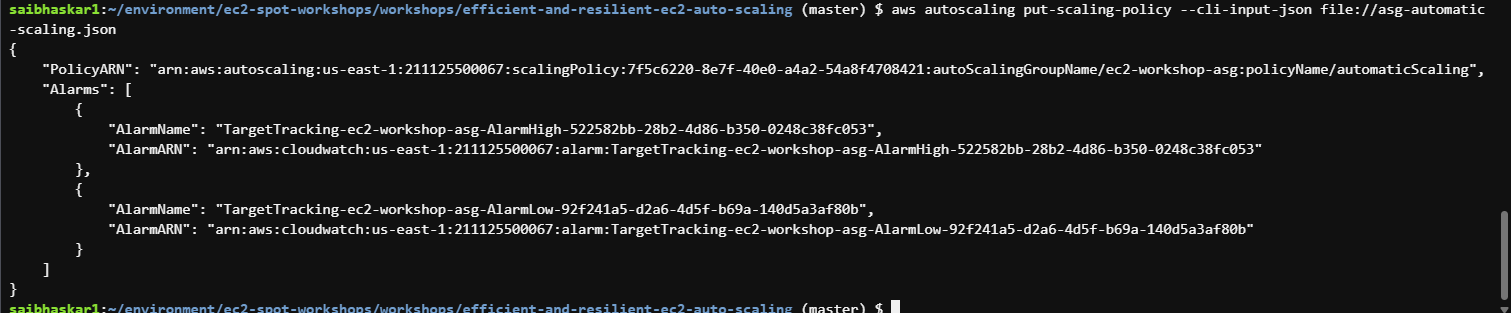
**aws autoscaling put-scaling-policy --cli-input-json file://asg-automatic-scaling.json**

1. Navigate to the [Auto Scaling console](https://console.aws.amazon.com/ec2/home#AutoScalingGroups:) and check out your newly created scaling policy in the Scaling Policies tab.
2. The dynamic scaling policy reduces the capacity at the times with low demand using the same average CPU utilization metric that predictive scaling uses.

*Because predictive scaling starts launching instances only at the beginning of each hour, you have to wait for the end of the hour to see dynamic scaling in action alongside predictive scaling. If you are learning on your own, grab a snack and watch it happen. If, however, you're at an AWS event it is probably best to proceed.*

Congratulations, you now have your Auto Scaling group configured with both:

* a predictive scaling policy in place to ensure application responsiveness ahead of the times of increased demand
* a dynamic policy to
  + scale in and reduce costs when demand is low
  + scale out if the load exceeds prediction



# Instance Lifecycles

**aws autoscaling put-lifecycle-hook \**

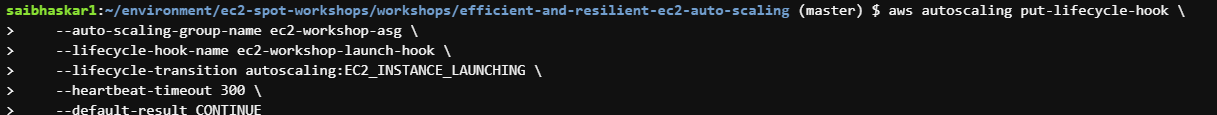
**--auto-scaling-group-name ec2-workshop-asg \**

**--lifecycle-hook-name ec2-workshop-launch-hook \**

**--lifecycle-transition autoscaling:EC2\_INSTANCE\_LAUNCHING \**

**--heartbeat-timeout 300 \**

**--default-result CONTINUE**

****

# Measure Launch Speed

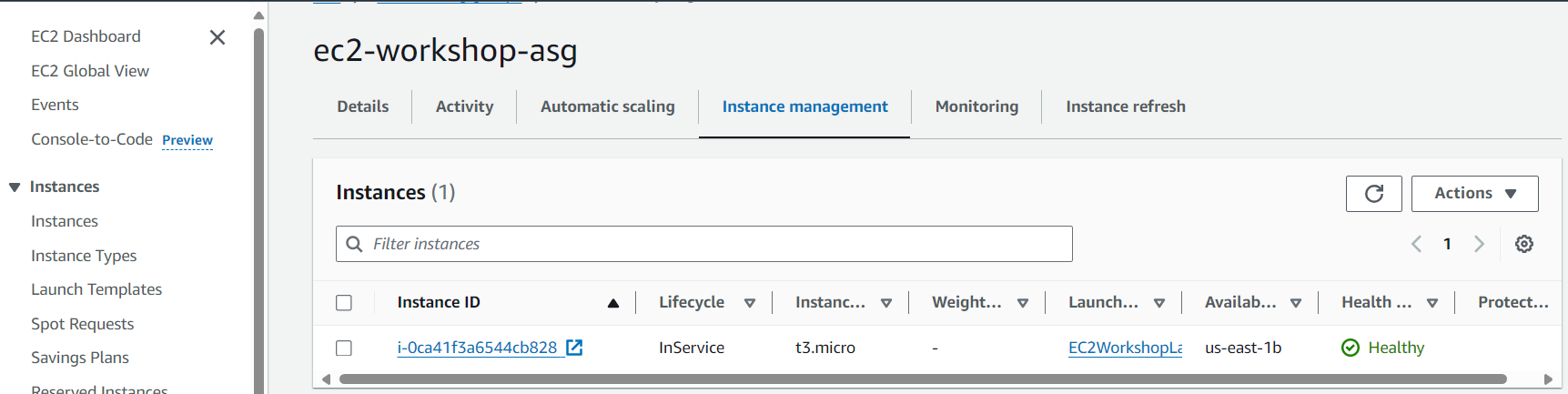
1. In AWS Console window, navigate to [EC2 Auto Scaling](https://console.aws.amazon.com/ec2/home#AutoScalingGroups:).
2. Click on tab **'Automatic scaling'**
3. Under **'Predictive scaling policies'**, find policy **workshop-predictive-scaling-policy** and toggle off **'Forecast and scale'**.
4. Click **'Switch off Forecast and scale'** in the confirmation message.
5. To ensure the accuracy of the next steps, you need to stop instances that are already running. In the Cloud9 terminal, run this command to ensure current capacity is set to 0 instances and wait a moment for the instances to start shutting down.

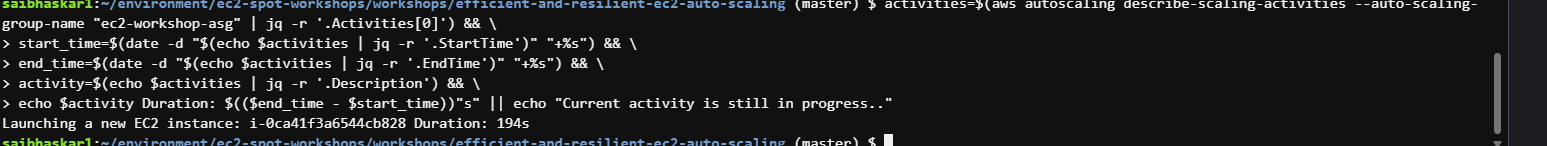
**aws autoscaling set-desired-capacity --auto-scaling-group-name "ec2-workshop-asg" --desired-capacity 0**

# 

**aws autoscaling set-desired-capacity --auto-scaling-group-name "ec2-workshop-asg" --desired-capacity 1**

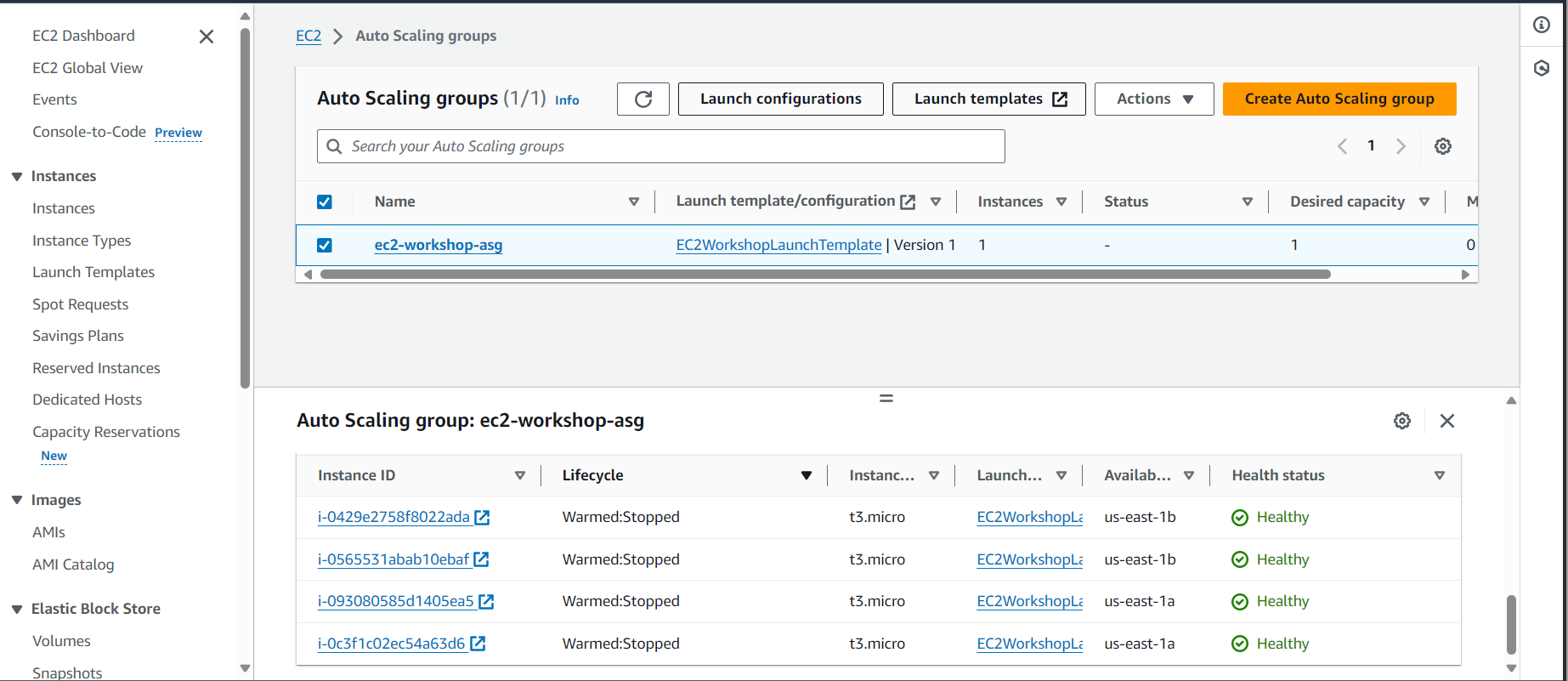
1. Navigate to the Auto Scaling console, click on Auto Scaling group **ec2-workshop-asg**
2. Click on the **'Instance management'** tab





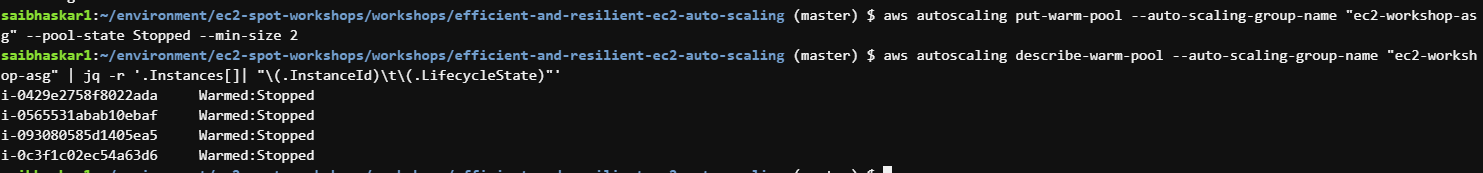
# Warm Pools

1. Navigate to the [Auto Scaling console](https://console.aws.amazon.com/ec2/home?AutoScalingGroups#AutoScalingGroups:), click on Auto Scaling group **ec2-workshop-asg**
2. Click on tab **'Instance management'**
3. A warm pool has been created and the instances have started initializing, note the current lifecycle for the instances is **Warmed:Pending:Wait**
4. Once the instances are initialized, the lifecycle will be changed to **Warmed:Stopped**



# How Warm Pools Work

**aws autoscaling describe-warm-pool --auto-scaling-group-name "ec2-workshop-asg" | jq -r '.Instances[]| "\(.InstanceId)\t\(.LifecycleState)"'**



**Observe launch speed into warm pool**

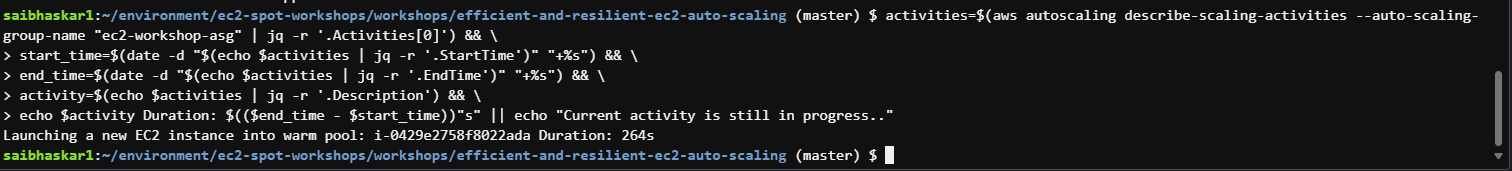
**activities=$(aws autoscaling describe-scaling-activities --auto-scaling-group-name "ec2-workshop-asg" | jq -r '.Activities[0]') && \**

**start\_time=$(date -d "$(echo $activities | jq -r '.StartTime')" "+%s") && \**

**end\_time=$(date -d "$(echo $activities | jq -r '.EndTime')" "+%s") && \**

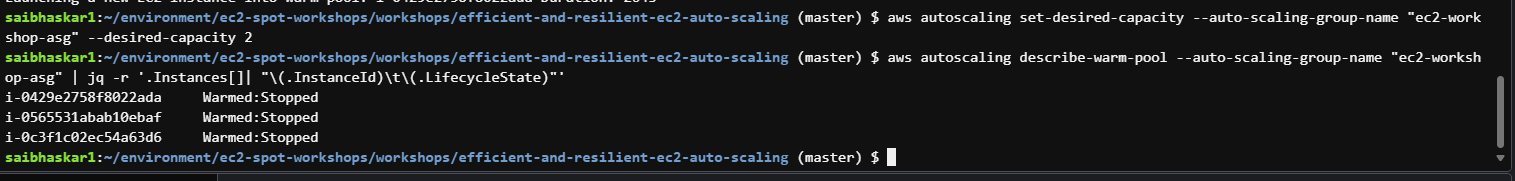
**activity=$(echo $activities | jq -r '.Description') && \**

**echo $activity Duration: $(($end\_time - $start\_time))"s" || echo "Current activity is still in progress.."**



**aws autoscaling set-desired-capacity --auto-scaling-group-name "ec2-workshop-asg" --desired-capacity 2**

**aws autoscaling describe-warm-pool --auto-scaling-group-name "ec2-workshop-asg" | jq -r '.Instances[]| "\(.InstanceId)\t\(.LifecycleState)"'**

****

**Measure launch speed**

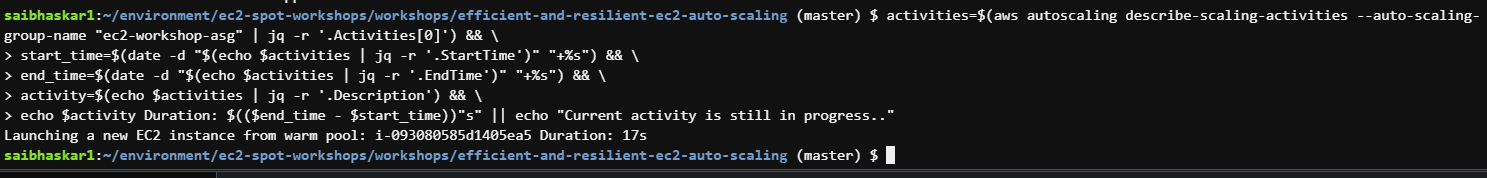
activities=$(aws autoscaling describe-scaling-activities --auto-scaling-group-name "ec2-workshop-asg" | jq -r '.Activities[0]') && \

start\_time=$(date -d "$(echo $activities | jq -r '.StartTime')" "+%s") && \

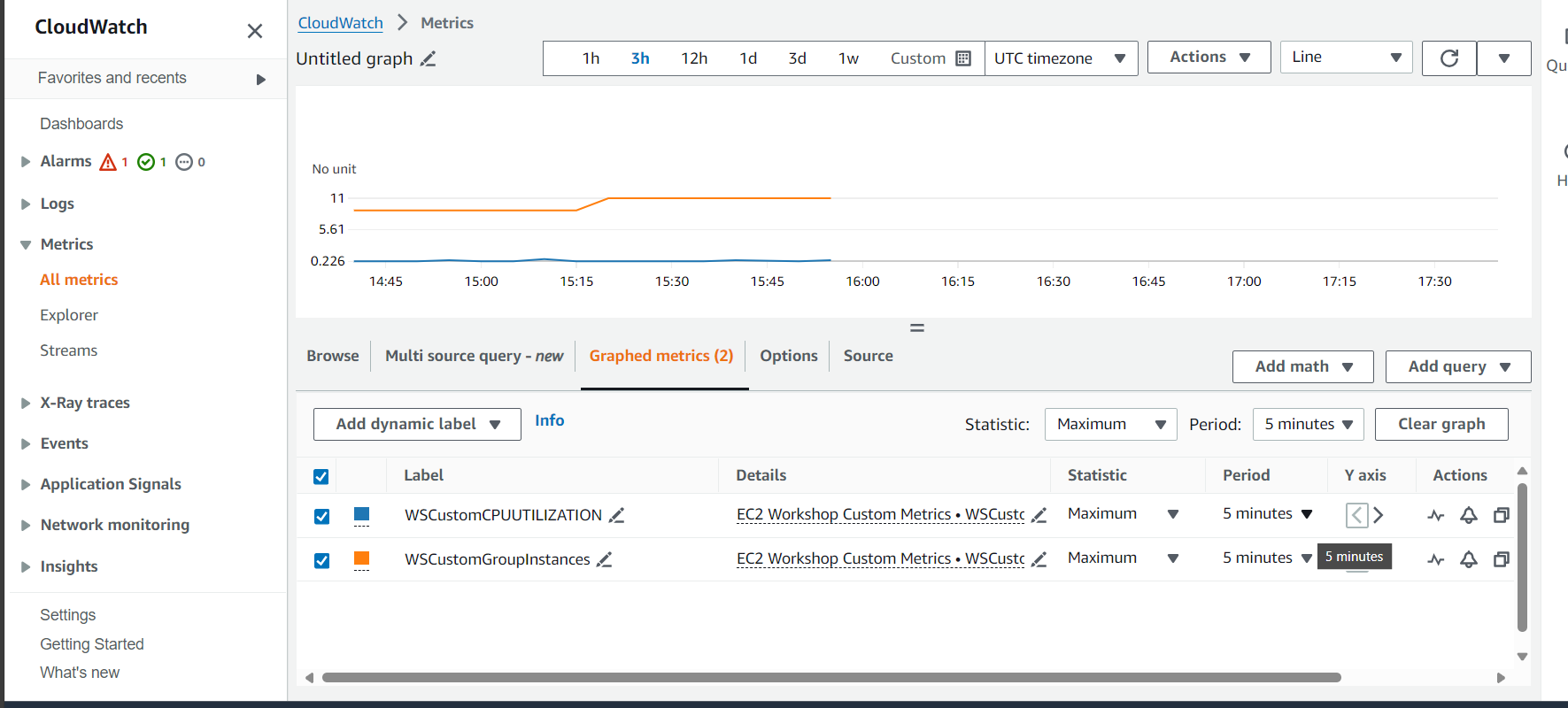
end\_time=$(date -d "$(echo $activities | jq -r '.EndTime')" "+%s") && \

activity=$(echo $activities | jq -r '.Description') && \

echo $activity Duration: $(($end\_time - $start\_time))"s" || echo "Current activity is still in progress.."

****

1. Navigate to [Amazon CloudWatch Console](https://console.aws.amazon.com/cloudwatch).
2. From the left side navigation, click on **'Metrics'** then All metrics.
3. In the **'Browse'** tab select **'Auto Scaling'** under **'AWS namespaces'**
4. Select **'Group Metrics'**, then select these two metrics: **GroupDesiredCapacity** and **GroupInServiceCapacity** attached with **ec2-workshop-asg**. This should add the metrics to the graph.
5. Switch to **'Graphed metrics'** tab and change **'Granularity Period'** to 1 minute for both metrics.
6. Note the difference in time between **'DesiredCapacity'** and **'InServiceCapacity'** before and after enabling warm pools.



**Cleanup**

**aws autoscaling delete-auto-scaling-group --auto-scaling-group-name ec2-workshop-asg --force-delete**

**aws ec2 delete-key-pair --key-name asgworkshop**

**aws cloudformation delete-stack --stack-name $stack\_name**

**aws cloudformation wait stack-delete-complete --stack-name $stack\_name**