

## Contents

Lab 5: Auto Scaling .....	2
Task Break down .....	2
Task 1: Create a Launch Template .....	2
Task 3: Stress Test .....	10
Task 4: Manage Auto Scaling using CLI .....	11

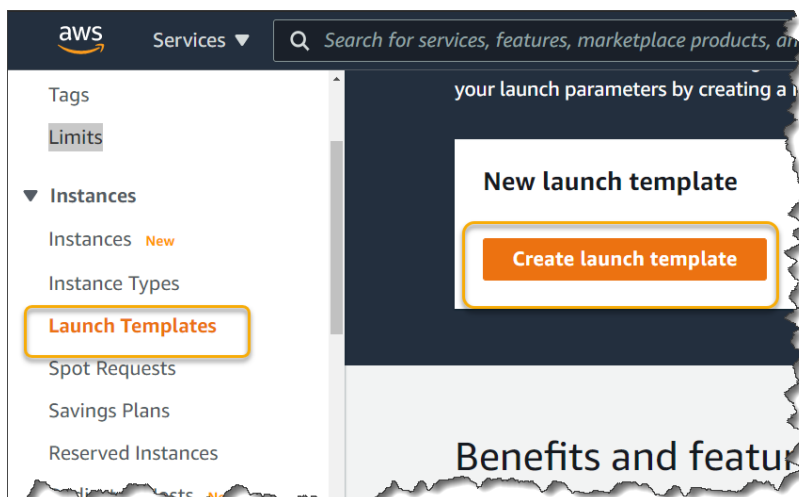
## Lab 5: Auto Scaling

### Task Break down

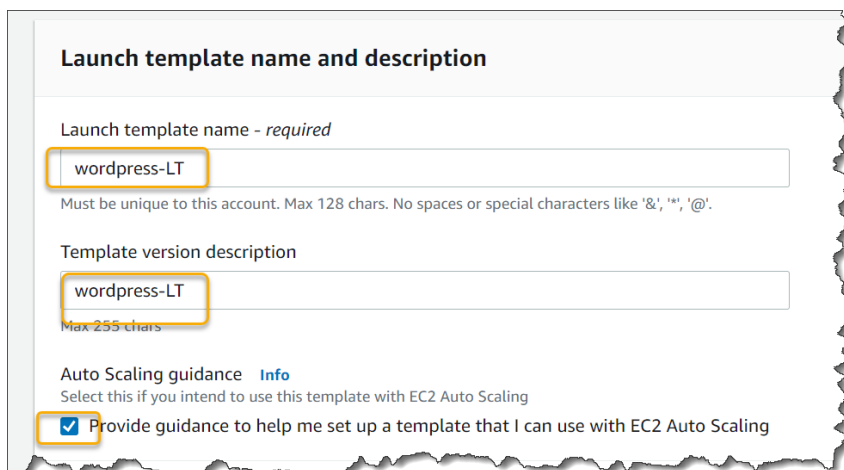
- Create a Launch Template
- Create an Auto Scaling Group and Create Scaling Policies
- Stress Test
- Perform Auto Scaling using CLI

### Task 1: Create a Launch Template

1. On the main EC2 dashboard, on the left-hand side of the screen navigate to **Launch Templates** and click **Create launch template**



2. Enter **wordpress-LT** as the Launch template name, provide a **description** and check **Autoscaling guidance**.



**Launch template name and description**

Launch template name - *required*

wordpress-LT

Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '\*', '@'.

Template version description

wordpress-LT

Max 255 chars

Auto Scaling guidance [Info](#)

Select this if you intend to use this template with EC2 Auto Scaling

☒ Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

- Under **Launch template contents**, Select **wordpress-image** as your **AMI**, select **t2.micro** as your **Instance type** and **hybridskill-app-server-key** as your **Key-pair**

### Launch template contents

Specify the details of your launch template below. You can create a new launch template or select an existing one.

#### Amazon machine image (AMI) - required

AMI - required

wordpress-image

ami-0598f743df0d6cd6c

Catalog: My AMIs architecture: 64-bit (x86) virtural

#### Instance type [Info](#)

Instance type

t2.micro

Family: t2 1 vCPU 1 GiB Memory

On-Demand Linux pricing: 0.0124 USD per Hour

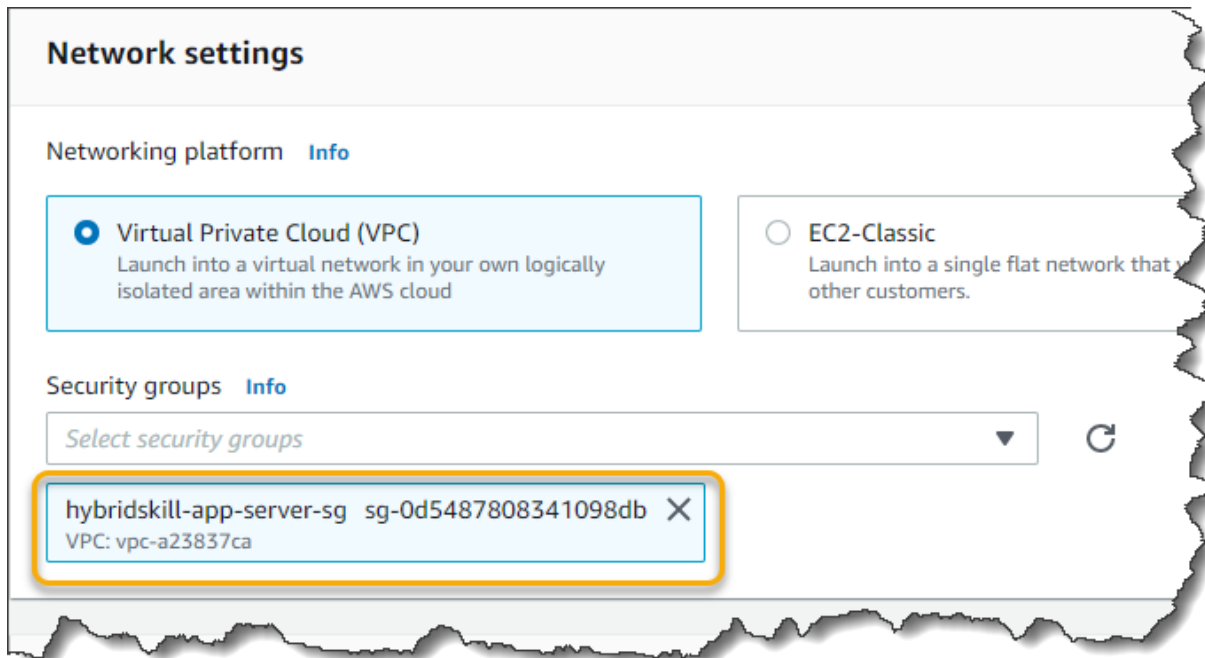
On-Demand Windows pricing: 0.017 USD per Hour

#### Key pair (login) [Info](#)

Key pair name

hybridskill-app-server-key

- Under **Network Settings** select **hybridskill-app-server-sg** as your **Security group**



**Network settings**

Networking platform [Info](#)

☒ Virtual Private Cloud (VPC)  
Launch into a virtual network in your own logically isolated area within the AWS cloud

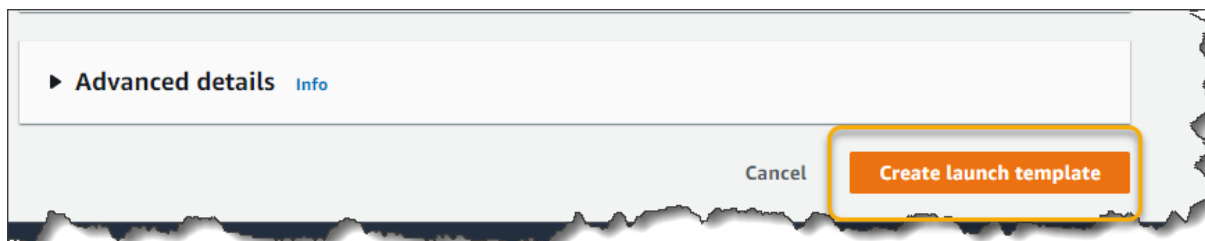
☐ EC2-Classic  
Launch into a single flat network that other customers.

Security groups [Info](#)

Select security groups ▼

hybridskill-app-server-sg sg-0d5487808341098db ✕  
VPC: vpc-a23837ca

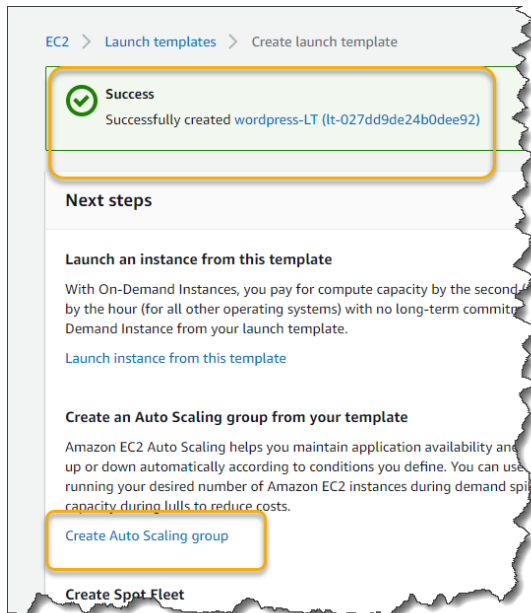
- Leave all other settings as default and click **Create launch template**



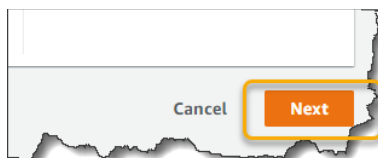
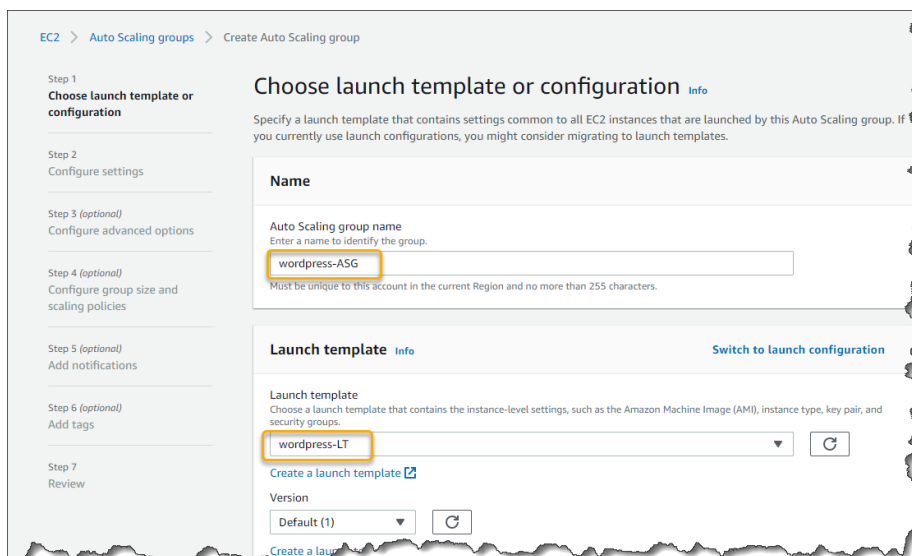
► **Advanced details** [Info](#)

Cancel **Create launch template**

6. Once your template is created, click **Create Autoscaling group**



7. Give **wordpress-ASG** as the name of your autoscaling group and make sure your launch template is selected. Click **Next**



## 8. Select subnets for availability zones **a** and **b** and click **Next**

**Instance purchase options** Info

Use the launch template to create a uniform configuration among all of the instances in the group. Or define options to accommodate a wide variety of requirements, such as launching Spot and On-Demand Instances.

☒ **Adhere to launch template**  
The launch template determines the purchase option (On-Demand or Spot) and instance type.

☐ **Combine purchase options and instance types**  
Specify how much On-Demand and Spot capacity to launch and multiple instance types (optional). This choice is most helpful for optimizing the scale and cost for a fleet of instances.

---

**Network** Info

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

VPC  
vpc-a23837ca (Default)  
172.31.0.0/16 Default

[Create a VPC](#)

Subnets  
Select subnets

ap-south-1a | subnet-332c185b  
172.31.32.0/20 Default

ap-south-1b | subnet-54264318  
172.31.0.0/20 Default

[Create a subnet](#)

Cancel Previous Skip to review **Next**

## 9. Select **Attach to existing load balancer** and select the target group that you created earlier. Click **Next**

**Configure advanced options** Info

Choose a load balancer to distribute incoming traffic for your application across instances to make it more reliable and easily scalable. You can also set options that give you more control over health check replacements and monitoring.

**Load balancing - optional** Info

Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

☐ **No load balancer**  
Traffic to your Auto Scaling group will not be fronted by a load balancer.

☒ **Attach to an existing load balancer**  
Choose from your existing load balancers.

☐ **Attach to a new load balancer**  
Quickly create a basic load balancer to attach to your Auto Scaling group.

---

**Attach to an existing load balancer**

Select the load balancers that you want to attach to your Auto Scaling group.

☒ **Choose from Application or Network Load Balancer target groups**

☐ **Choose from Classic Load Balancers**

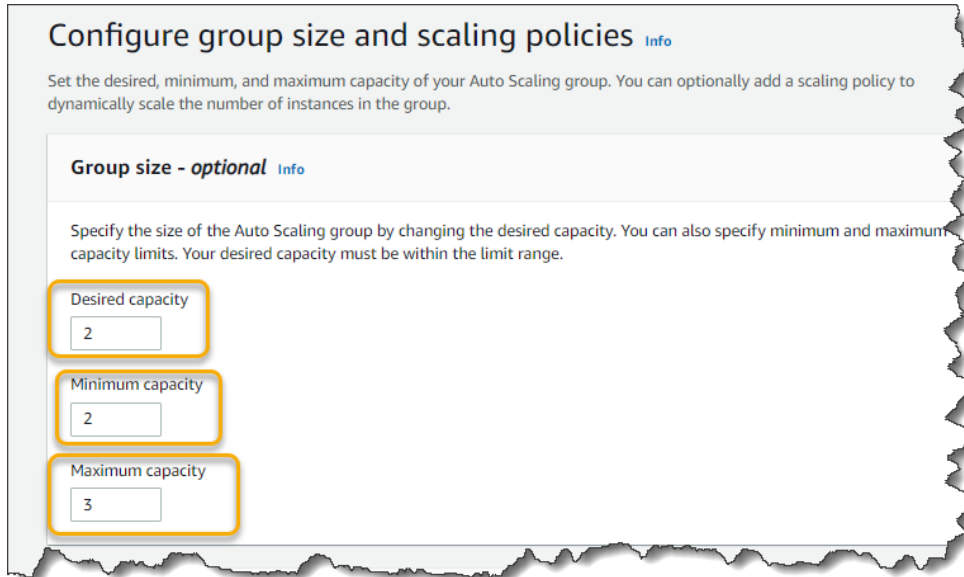
Application or Network Load Balancer target groups  
Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

Select target groups

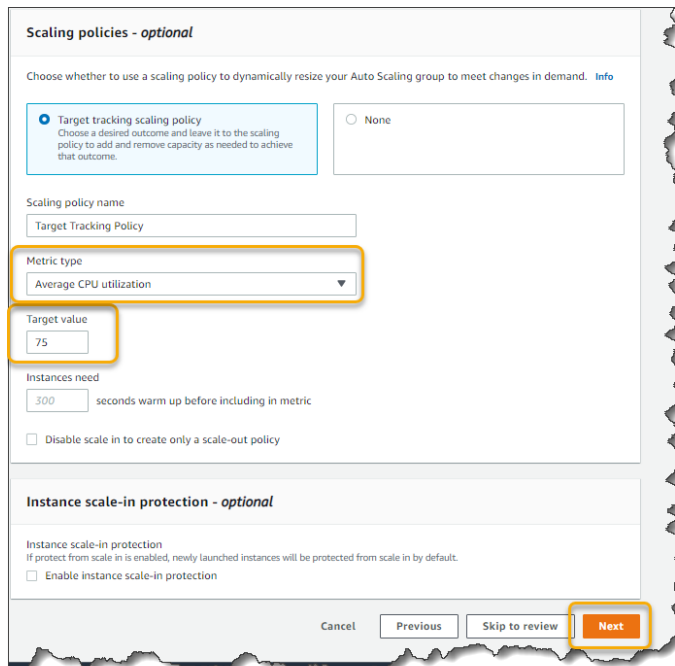
hybridskill-tg | HTTP  
Application Load Balancer: hybridskill-tb

Cancel Previous Skip to review **Next**

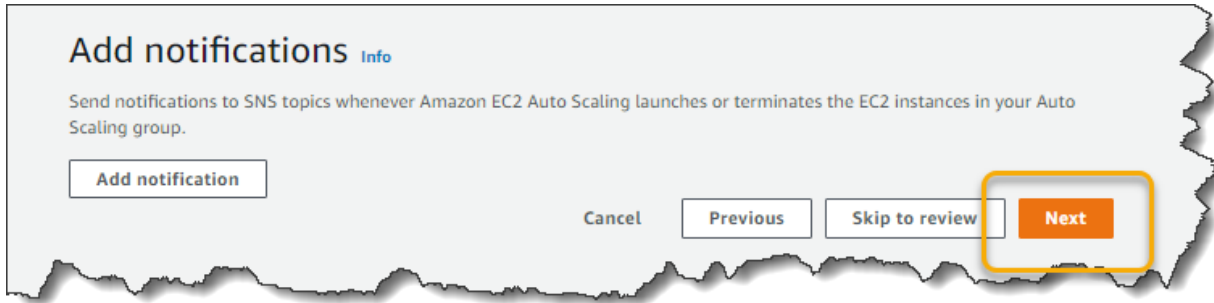
10. Enter **2** as the **Desired capacity**, **2** as the **Minimum capacity** and **3** as the **Maximum capacity**



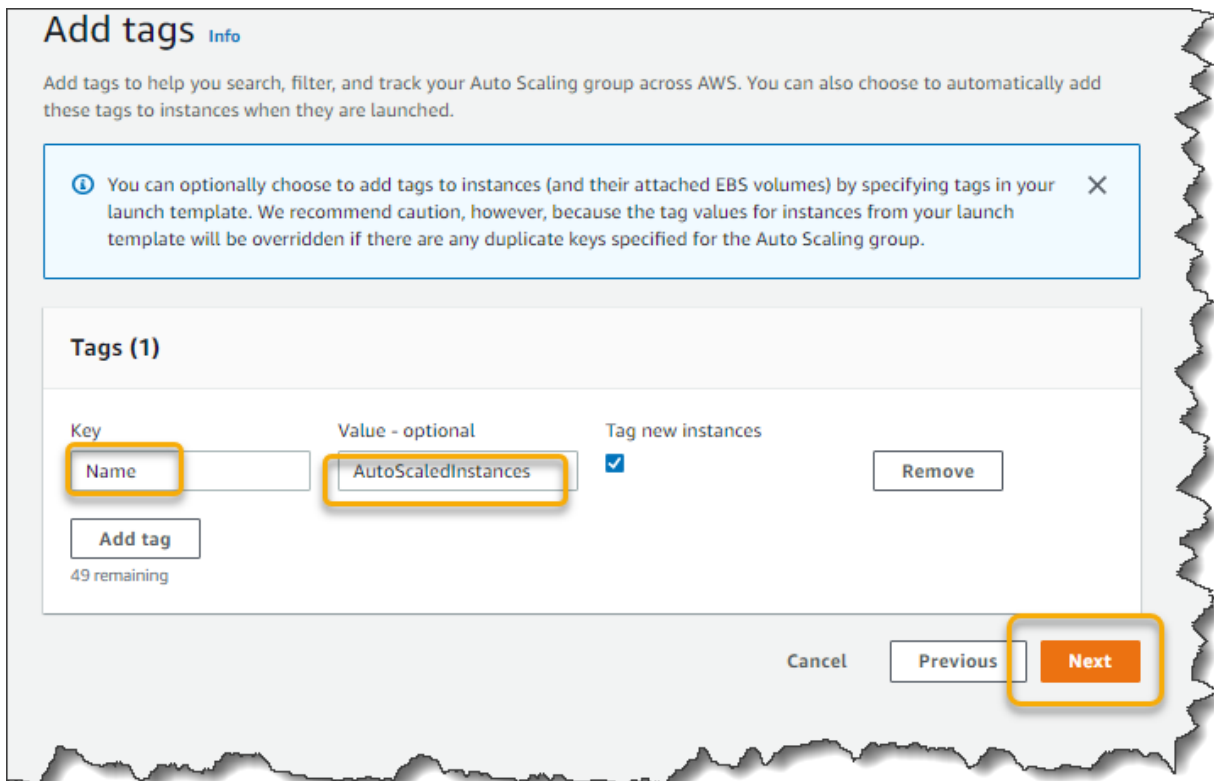
Select **Target tracking scaling policy**, choose the metric as **Average CPU utilization** and Enter the **Target value** as **75**. Click **Next**



11. Don't add any notifications and click **Next**



12. Add a tag with **Key = Name** and **Value = AutoScaledInstances**. Click **Next**





13. Finally click **Create AutoScaling group**

Step 6: Add tags Edit

**Tags (1)**

Key	Value	Tag new instances
Name	AutoScaledInstances	Yes

Cancel Create Auto Scaling group

✔ wordpress-ASG, 1 Scaling policy created successfully

14. On the lower tab of your screen, click **Instances Management** and you should see 2 new instances created to satisfy the minimum group size of 2 requirement.

EC2 > Auto Scaling groups

**Auto Scaling groups (1/1)**

Search your Auto Scaling groups

<input checked="" type="checkbox"/>	Name	Launch template/configuration	Instan...	St...
<input checked="" type="checkbox"/>	wordpress-ASG	wordpress-LT   Version Default	2	-

Details | Activity | Automatic scaling | **Instance management** | Monitoring | Instance refresh

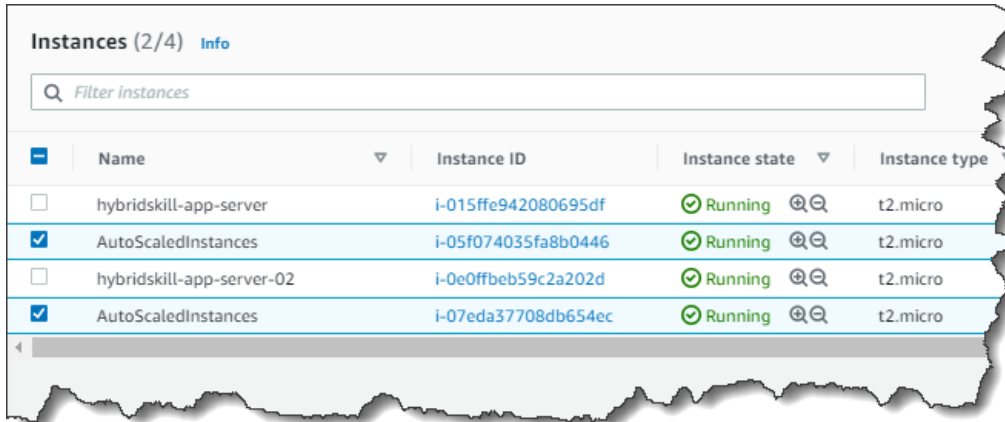
**Instances (2/2)**

Filter instances

<input checked="" type="checkbox"/>	Instance ID	Lifecycle	Instance type	Weighted capacity
<input checked="" type="checkbox"/>	i-05f074035fa8b0446	Pending	t2.micro	-
<input checked="" type="checkbox"/>	i-07eda37708db654ec	Pending	t2.micro	-

## Task 3: Stress Test

We are next going to simulate a **Scale up** action. For this we need to spike the Avg CPU load to > 75%. We will use a tool called **stress** for this. This tool has to be installed on both the machines.



	Name	Instance ID	Instance state	Instance type
<input type="checkbox"/>	hybridskill-app-server	i-015ffe942080695df	Running	t2.micro
<input checked="" type="checkbox"/>	AutoScaledInstances	i-05f074035fa8b0446	Running	t2.micro
<input type="checkbox"/>	hybridskill-app-server-02	i-0e0ffbeb59c2a202d	Running	t2.micro
<input checked="" type="checkbox"/>	AutoScaledInstances	i-07eda37708db654ec	Running	t2.micro

1. Log into both the EC2 machines and run the following commands to install stress.

```
1. sudo wget http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
2. sudo rpm -ivh epel-release-latest-7.noarch.rpm
3. sudo yum --enablerepo=epel install stress
```

2. Next run stress to squeeze the cores of the servers. Log in both the EC2 machines and run the following commands to trigger stress.

```
1. stress -c 2
```

As soon as you do this you should see the third machine come up in response to autoscaling Scale up being fired.

Log back into both the machines and stop stress by doing a Ctrl-C

The CPU should drop down again, and Scale down should happen which will remove an instance.

## Task 4: Manage Auto Scaling using CLI

Now that we have explored Auto Scaling through the console, let's do the same through the CLI. Run the following commands on the command line interface that you had setup earlier.

### 1. Create Launch configuration:

```
:~$ aws autoscaling create-launch-configuration --launch-configuration-name autoscaling-for-wp-app-test --key-name hybridskill-test --image-id ami-0e356a61 --instance-type t2.micro --security-groups hybridskill-sg-test
```

### 2. Launch Auto Scaling Group

```
:~$ aws autoscaling create-auto-scaling-group --auto-scaling-group-name wordpress-app-autoscaling-test --min-size 1 --max-size 2 --launch-configuration-name autoscaling-for-wp-app-test --load-balancer-names wp-test-elb --health-check-type ELB --health-check-grace-period 60 --vpc-zone-identifier subnet-3f312b72
```

### 3. Create scaling out policy

```
:~$ aws autoscaling put-scaling-policy --auto-scaling-group-name wordpress-app-autoscaling-test --policy-name cpu_scaling_out --policy-type StepScaling --adjustment-type ChangeInCapacity --step-adjustments MetricIntervalLowerBound=10,ScalingAdjustment=1

{
  "PolicyARN": "arn:aws:autoscaling:ap-south-1:123456789123:scalingPolicy:53ca70c8-8a35-4dc8-8e3a-b5077f1b6daf:autoScalingGroupName/wordpress-app-autoscaling-test:policyName/cpu_scaling_out",
  "Alarms": []
}
```

### 4. Create CPU alert for Scaling out policy

```
:~$ aws cloudwatch put-metric-alarm --alarm-name cpu_scaling_out --alarm-description "Alarm when >=60" --metric-name CPUUtilization --namespace AWS/AutoScaling --statistic Average --period 300 --threshold 60 --comparison-operator GreaterThanOrEqualToThreshold --dimensions "Name=AutoScalingGroupName,Value=wordpress-app-autoscaling-test" --evaluation-periods 1 --alarm-actions "arn:aws:autoscaling:ap-south-1:123456789123:scalingPolicy:13247c9d-2e6a-44df-9a07-338282aacec9:autoScalingGroupName/wordpress-app-autoscaling-test:policyName/cpu_scaling_out" --unit Percent
```

### 5. Create scaling in policy:

```
:~$ aws autoscaling put-scaling-policy --auto-scaling-group-name wordpress-app-autoscaling-test --policy-name cpu_scaling_in --policy-type StepScaling --adjustment-type ChangeInCapacity --step-adjustments MetricIntervalUpperBound=-20,ScalingAdjustment=-1

{
  "PolicyARN": "arn:aws:autoscaling:ap-south-1:123456789123:scalingPolicy:185f28ab-9b98-42a3-b3ce-261cab918037:autoScalingGroupName/wordpress-app-autoscaling-test:policyName/cpu_scaling_in",
  "Alarms": []
}
```

## 6. Create scaling in CPU alert:

```
:~$ aws cloudwatch put-metric-alarm --alarm-name cpu_scaling_in --alarm-description  
"Alarm when <=40" --metric-name CPUUtilization --namespace AWS/AutoScaling --statistic  
Average --period 300 --threshold 60 --comparison-operator  
LessThanOrEqualToThreshold --dimensions "Name=AutoScalingGroupName,Value=wordpress-  
app-autoscaling-test" --evaluation-periods 1 --alarm-actions "arn:aws:autoscaling:ap-  
south-1:123456789123:scalingPolicy:185f28ab-9b98-42a3-b3ce-  
261cab918037:autoScalingGroupName/wordpress-app-autoscaling-  
test:policyName/cpu_scaling_in" --unit Percent
```

## Important: Cleanup of all Resources

Next let's follow this checklist make sure all resources are cleaned up. to prevent billing to your account.

- **Autoscaling Groups**
- **Launch Templates**
- **Elastic load balancers**
- **RDS instances**
- **EC2 instances**