Contents

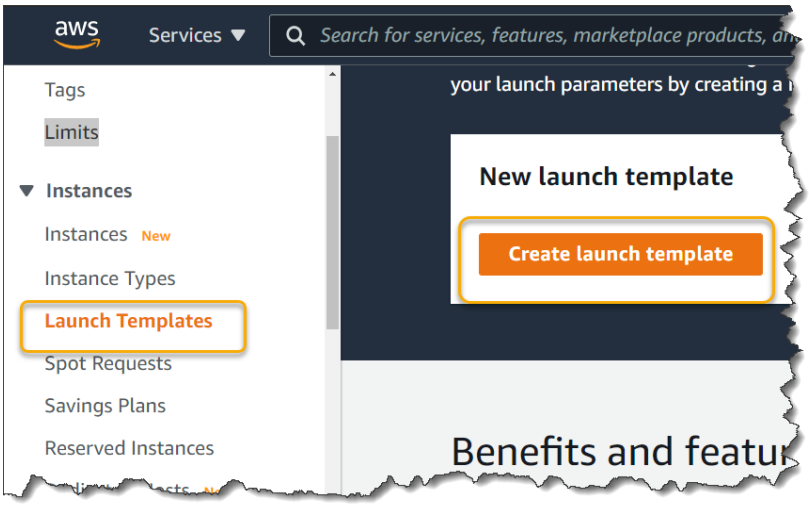
Lab 5: Auto Scaling

Task Break down

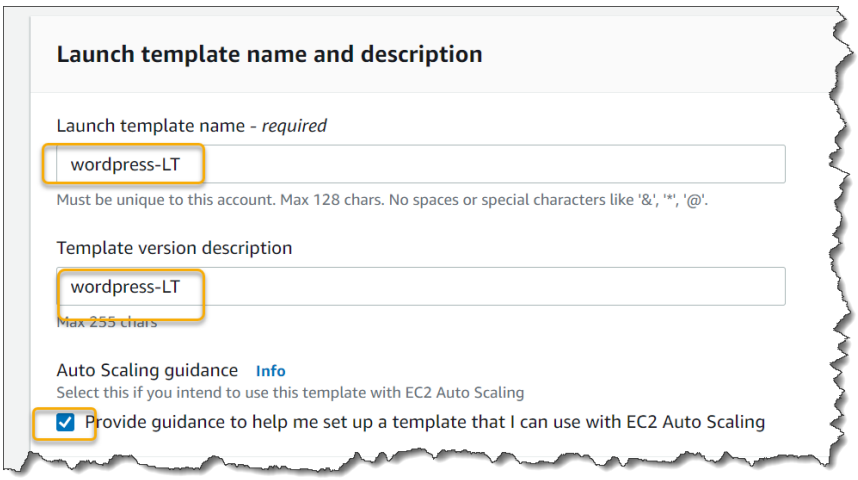
- Create a Launch Template
- Create an Auto Scaling Group and Create Scaling Policies
- Stress Test
- Perform Auto Scaling using CLI

Task 1: Create a Launch Template

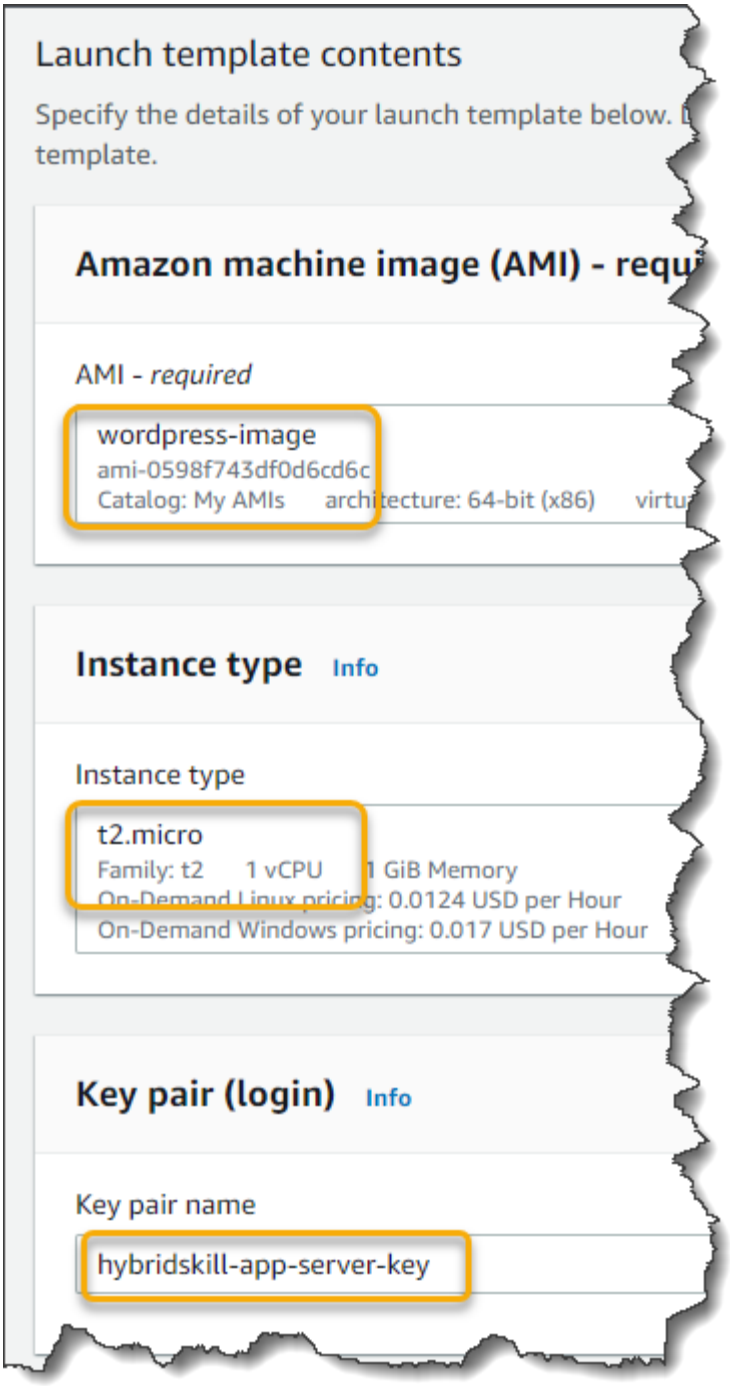
1. On the main EC2 dashboard, on the left-hand side of the screen navigate to **Launch Templates and click Create launch template**



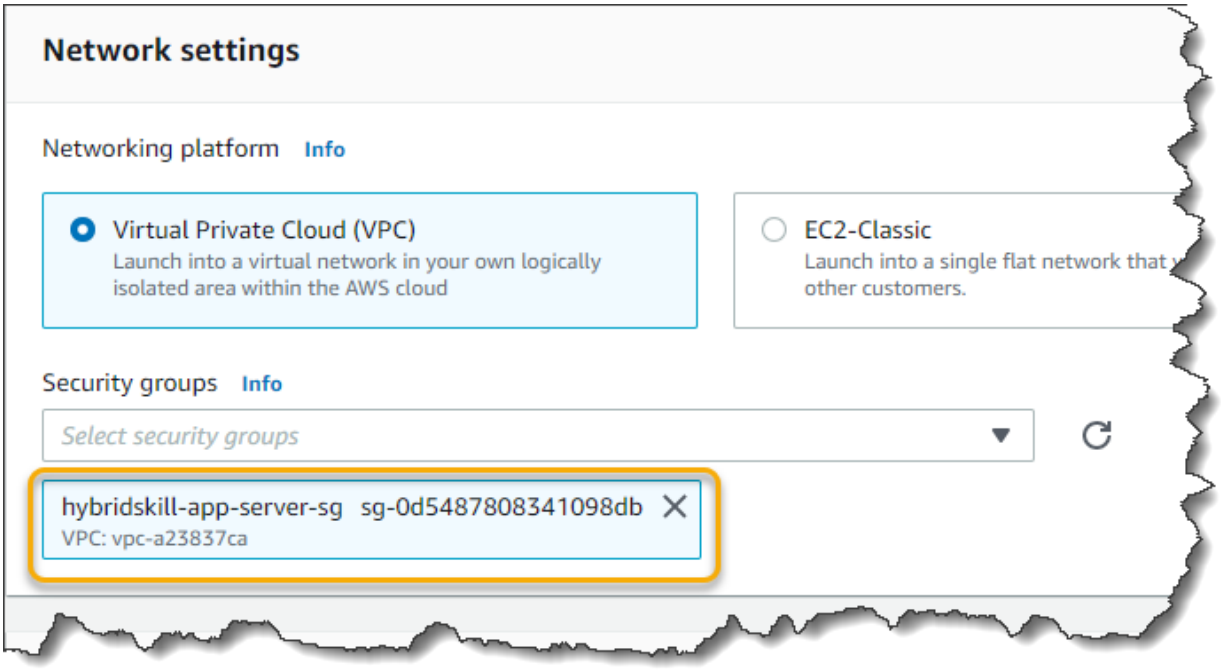
2. Enter **wordpress-LT** as the Launch template name, provide a **description** and check **Autoscaling guidance.**



3. 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**

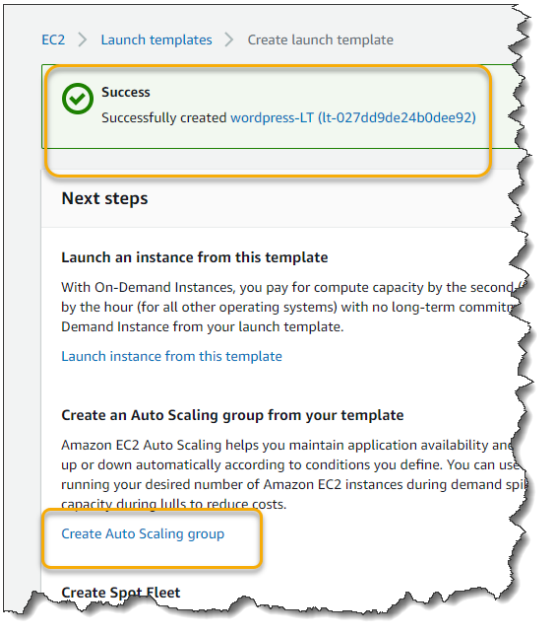


4. Under **Network Settings** select **hybridskill-app-server-sg** as your **Security group**

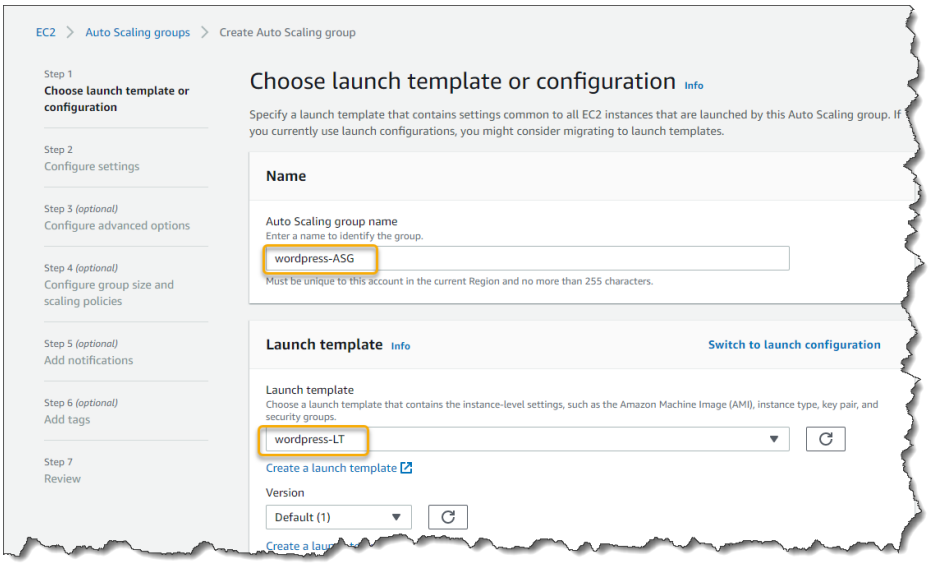


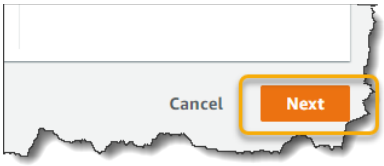
5. Leave all other settings as default and click **Create launch template**

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

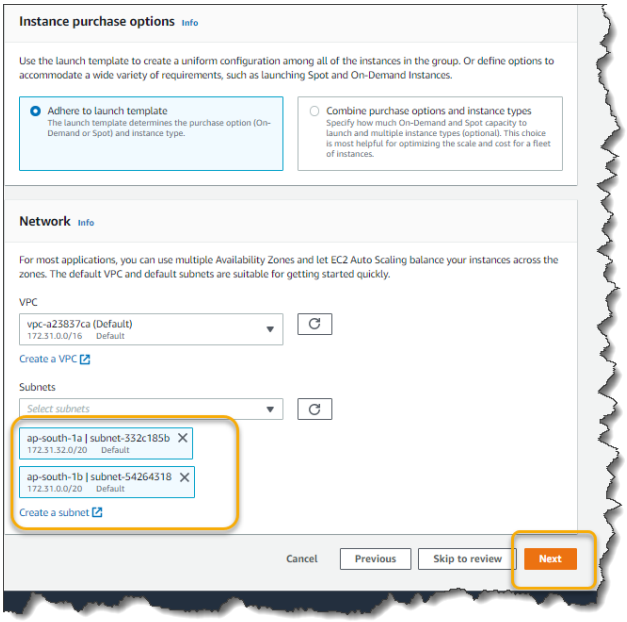


7. Give **wordpress-ASG** as the name of your autoscaling group and make sure your launch template is selected. Click **Next**

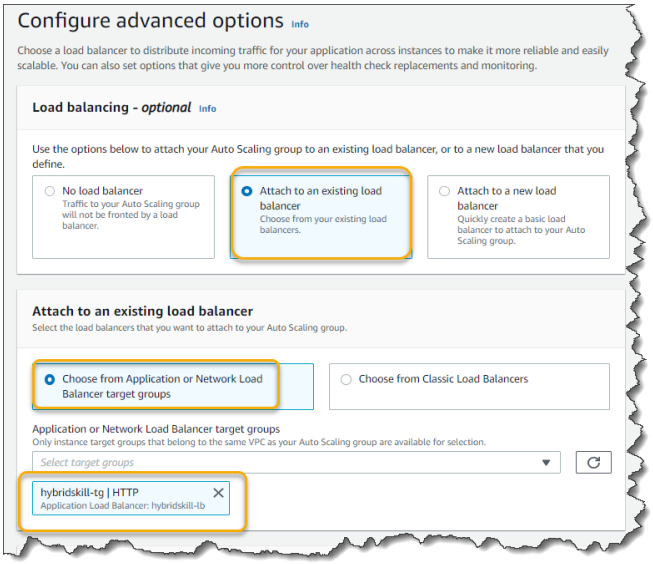


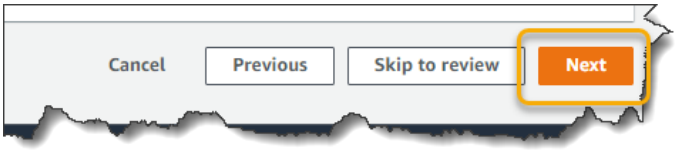


8. Select subnets for availability zones **a** and **b** and click **Next**

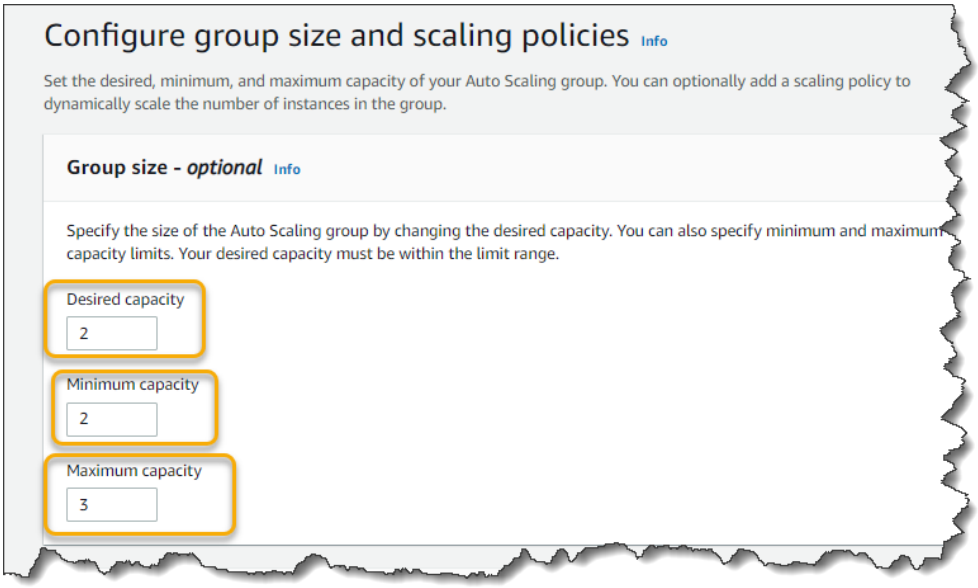


9. Select **Attach to existing load balancer** and select the target group that you created earlier. Click **Next**

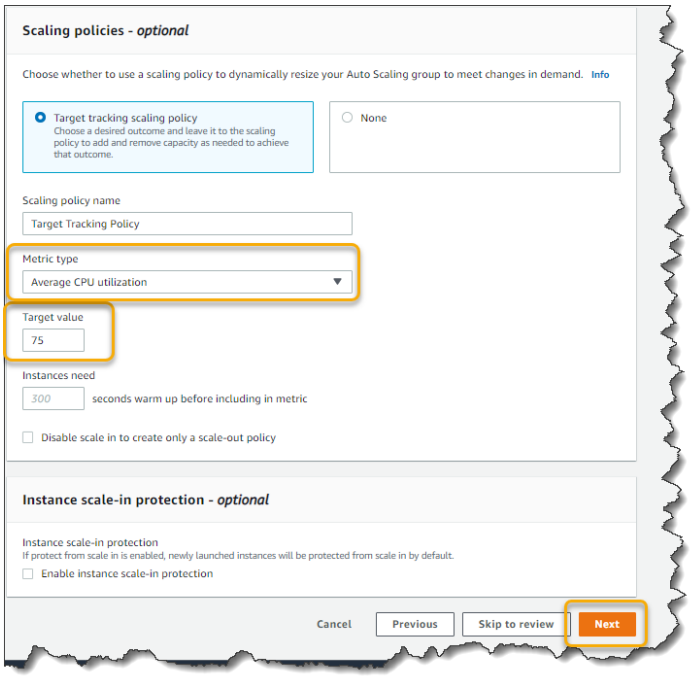




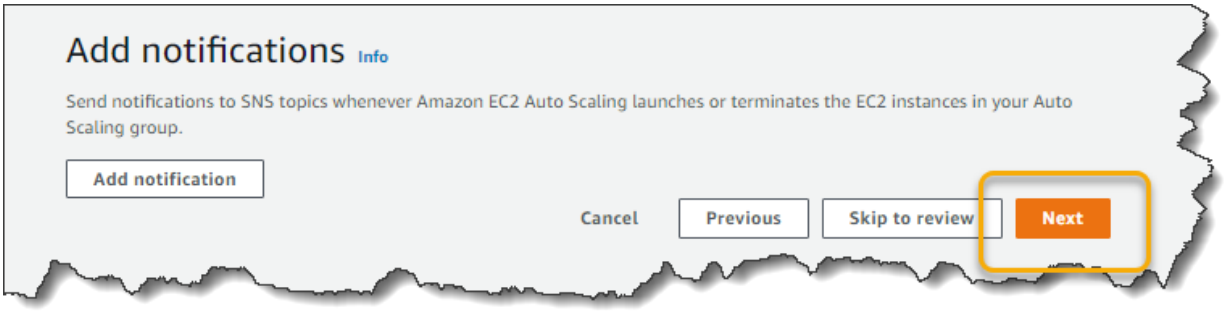
10. Enter **2** as the **Desired capacity**, **2** as the **Minimum capacity** and **3** as the **Maximum capacity**



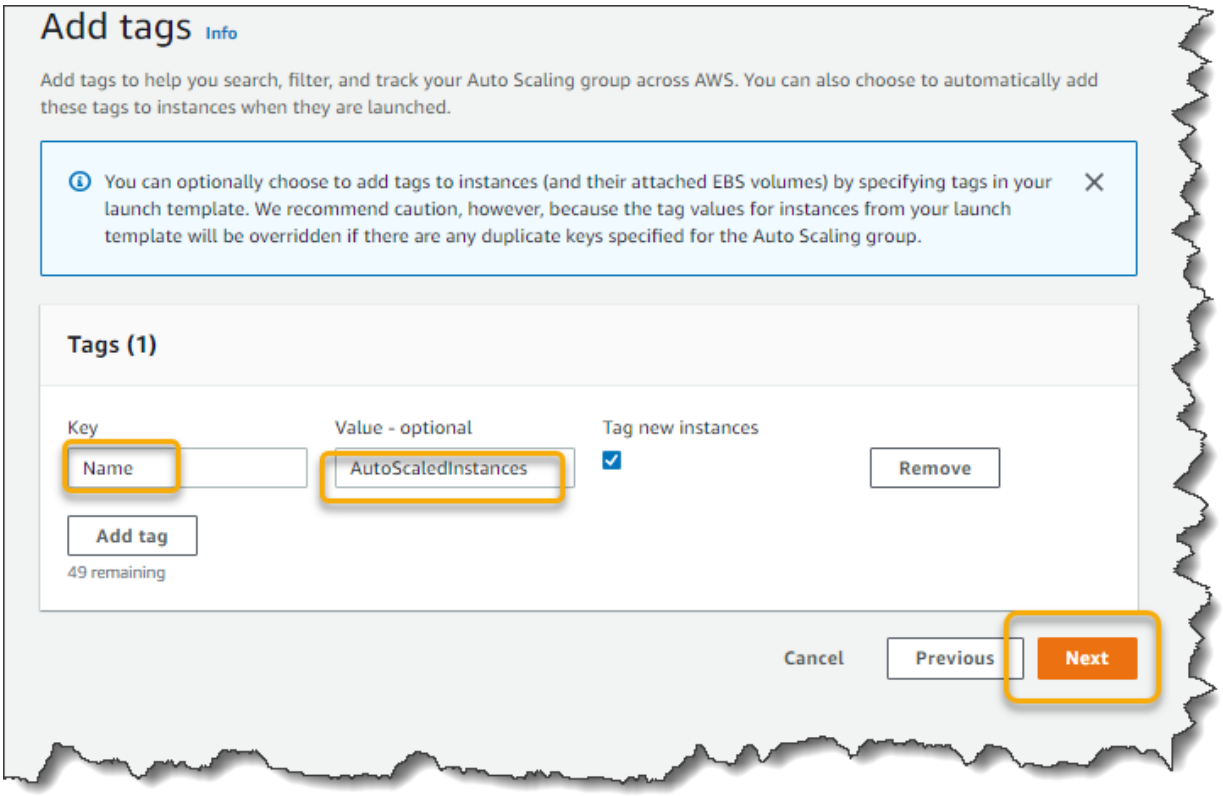
Select **Target tracking scaling policy**, choose the metric as **Average CPU utilization** and Enter the **Target value** as **75**. Click **Next**



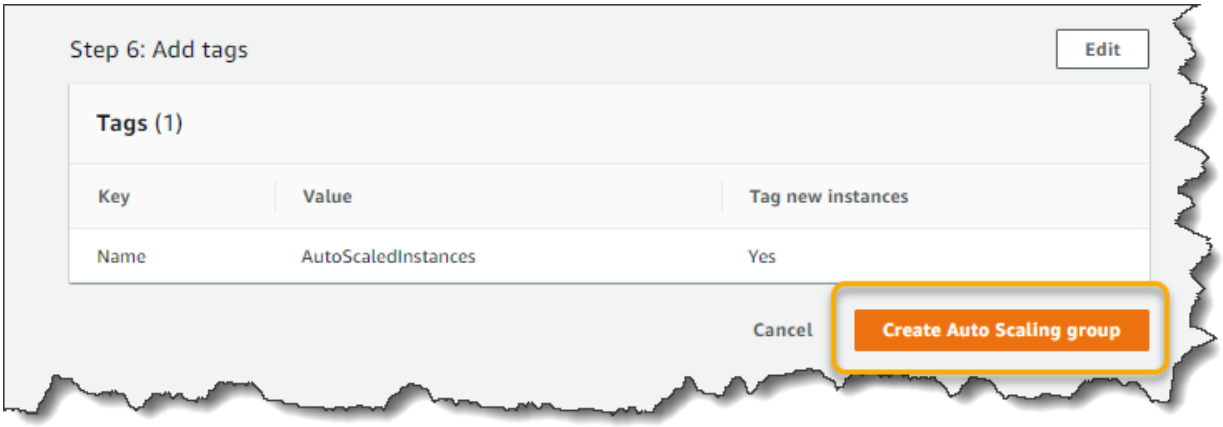
11. Don't add any notifications and click **Next**

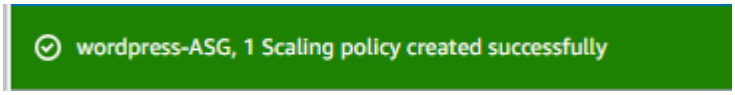


12. Add a tag with **Key = Name** and **Value = AutoScaledInstances.** Click **Next**

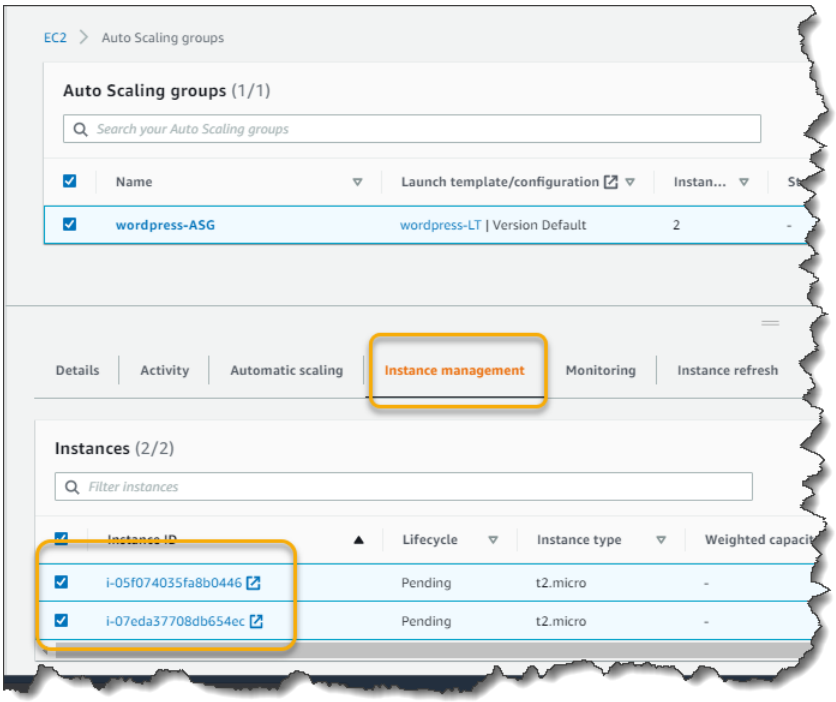


13. Finally click **Create AutoScaling group**



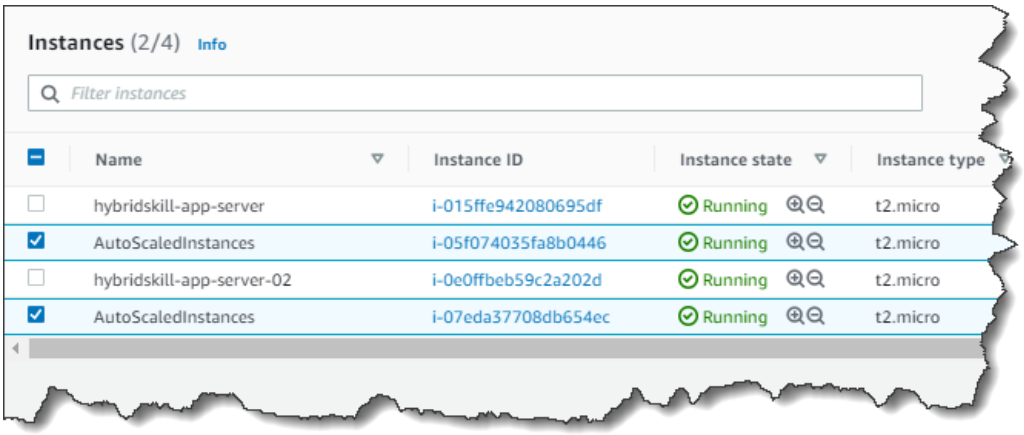


14. On the lower tab of your screen, click **Instances Management** and you should see 2 new instances created to satisfy the minimum group size of 2 requirement.



Task 3: Stress Test

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



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**