



Published in Towards AWS



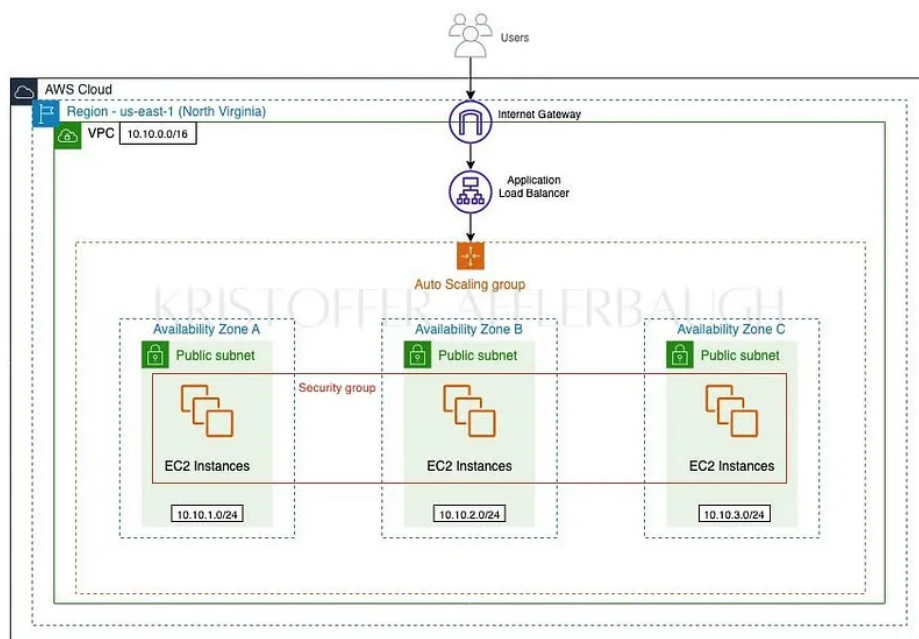
Kristoffer Afflerbaugh

Follow

May 1 · 11 min read · Listen



## Scaling your AWS architecture with EC2 auto scaling groups



The diagram above is what we will create in this project.

### Scenario:

A business has decided to move its entire operation to the cloud, ditching the old model of using on-premises servers. They'll need to use EC2

instances to host web servers in three different availability zones. The EC2 instances will need to be scalable; if traffic spikes more instances should be launched, and if the traffic is low the instances need to be scaled back to reduce cost. In addition to needing to scale their EC2 instances, they'll need a load balancer to distribute internet traffic evenly amongst the instances.

### Prerequisites

- You'll need to have an AWS account set up and be able to work with an IAM user that has admin privileges.
- An understanding of how to set up basic resources using VPC and EC2.

### Glossary

- **VPC (Virtual Private Cloud):** A virtual network in the cloud that resembles a traditional on-premises network. It is private, but hosted within the public cloud.
- **Subnet:** A subnet is a specific range of IP addresses within the VPC. Subnets can be private or public, and can be viewed as smaller VPCs within the main VPC.
- **CIDR:** CIDR stands for Classless Inter-Domain Routing. It is an improved way of assigning IP addresses from the old classful method, which drained the pool of unassigned IP addresses much more. When we refer

to a "CIDR block", we are talking about a range of IP addresses. You can dive deeper into CIDR [here](#).

- **IPv4 (Internet Protocol version 4):** IP addresses are 32 bit addresses divided into 4 separate octets, or 8 bit values. We usually see IPv4 addresses as 4 sets of up to 3 digit numbers ranging from 0 to 255. This [article](#) provides a great explanation of how the numbers we use are converted to binary form, and vice versa.
- **Load Balancer:** A load balancer distributes traffic across a pool of resources. There are many different ways of distributing that traffic. In our case, we will use an application load balancer, which operates using the TCP, HTTP and HTTPS protocols, on layer 7 of the [OSI model](#). The simplest way of using it is to route traffic evenly across different web servers, which in our case will be running on EC2 instances.
- **Auto scaling group:** An auto scaling group launches or terminates EC2 instances based on demand. When the traffic increases, more EC2 instances are created to alleviate the cpu load of the current instances. When there is a decrease in traffic, instances will be terminated in order to save money.
- **Security group:** Security groups are essentially a group of rules that act as a firewall. They will allow only the necessary traffic to AWS resources in an effort to increase security.
- **Launch Template:** A launch template is a template that is pre-configured to launch new EC2 instances quickly. Launch templates can even be pre-

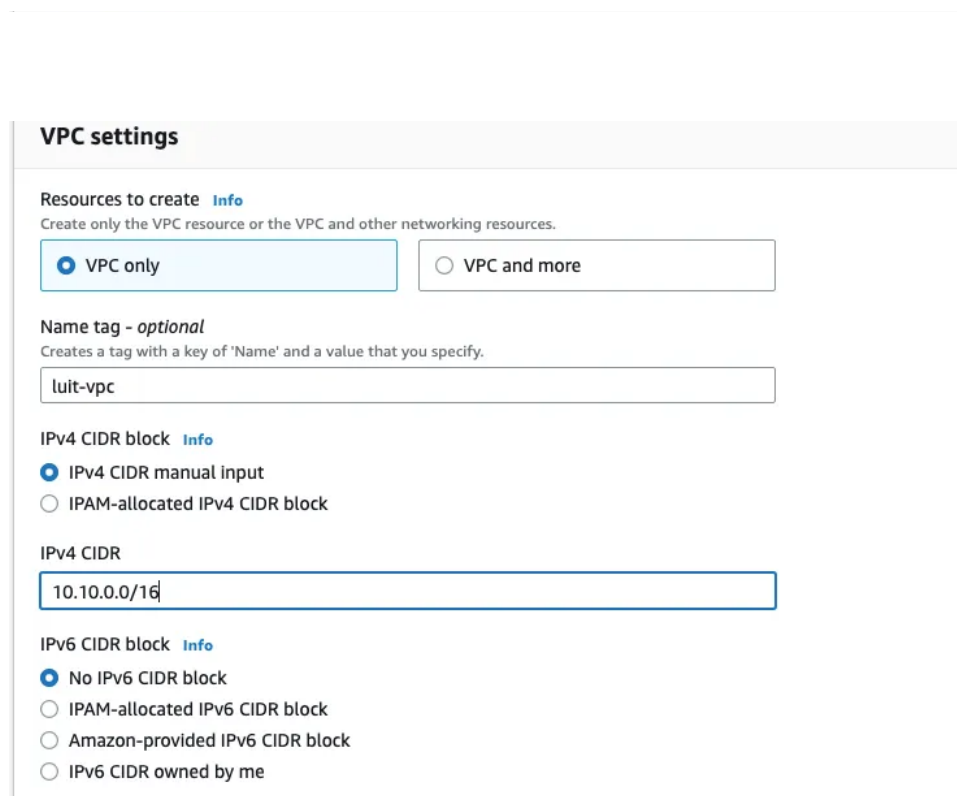
configured with user data, which will run a script upon the creation of the instance. One common use of user data is to install and configure a web server upon creation.

. . .

## Phase 1) Creating the infrastructure: VPC, subnets, load balancer and auto scaling group

### Step 1.)

The first thing we need to do is create our VPC from the VPC dashboard. Choose “VPC only” from the menu and give it a unique name. It makes life much easier if all of your resources have a similar name that refer to each other. For example, I named my VPC “luit-vpc”. When I go to name other resources I will name them all with the prefix “luit”, so I know they are all related to this particular project. The IPv4 CIDR block is going to be 10.10.0.0/16. I created my VPC in the region “us-east-1”. You can use any region for this project, as long as it has at least 3 availability zones.



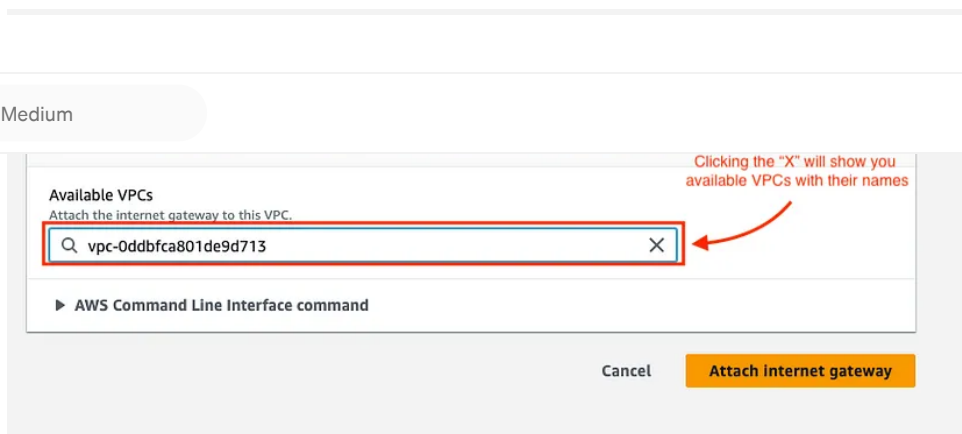
The screenshot shows the 'VPC settings' section of the AWS console. It includes the following fields and options:

- Resources to create:** Two radio buttons. 'VPC only' is selected (indicated by a blue dot). 'VPC and more' is unselected.
- Name tag - optional:** A text input field containing 'luit-vpc'. Below the field is a small description: 'Creates a tag with a key of 'Name' and a value that you specify.'
- IPv4 CIDR block:** Two radio buttons. 'IPv4 CIDR manual input' is selected. 'IPAM-allocated IPv4 CIDR block' is unselected.
- IPv4 CIDR:** A text input field containing '10.10.0.0/16'.
- IPv6 CIDR block:** Four radio buttons. 'No IPv6 CIDR block' is selected. The other three options ('IPAM-allocated IPv6 CIDR block', 'Amazon-provided IPv6 CIDR block', and 'IPv6 CIDR owned by me') are unselected.

### Step 2.)

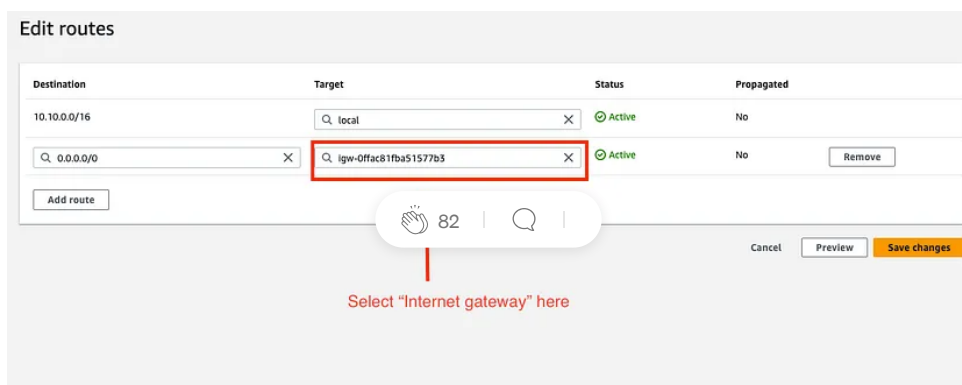
After your VPC is created, we'll need to create an internet gateway and attach it to our VPC. An internet gateway is what allows the VPC to connect to the internet. Once again, give it a name. In keeping with the prefix style naming

convention, I named mine “luit-igw”. After your internet gateway has been created you will be prompted to attach it to a VPC. Attach it to the VPC we just created.



### Step 3.)

Now we need to update the VPC route table to point to the internet gateway. Find the route table for your VPC and click to edit it. We need to click “Edit routes” and then “Add route”. You need to add “0.0.0.0/0” under “Destination” and select our internet gateway for “Target”.



### Step 4.)

Once you’ve saved the changes, now you can move on to creating the subnets. Select the VPC we’ve just created then create your first subnet. I would recommend naming it something that identifies it as a public subnet, as well as tell us which availability zone it is in. I named mine “luit-public-A”, since these are going to be public subnets. For your availability zone choose “a”. The CIDR block will be “10.10.1.0/24”.

**VPC**

VPC ID

Create subnets in this VPC.

vpc-0ddbfa801de9d713 (luit-vpc)

Associated VPC CIDRs

IPv4 CIDRs

10.10.0.0/16

**Subnet settings**

Specify the CIDR blocks and Availability Zone for the subnet.

**Subnet 1 of 1**

**Subnet name**

Create a tag with a key of 'Name' and a value that you specify.

luit-public-A

The name can be up to 256 characters long.

**Availability Zone** [Info](#)

Choose the zone in which your subnet will reside, or let Amazon choose one for you.

US East (N. Virginia) / us-east-1a

**IPv4 CIDR block** [Info](#)

10.10.1.0/24

Now do the same thing for the two remaining subnets. You can create them all from the same menu by selecting “Add new subnet”. They should have

CIDR blocks “10.10.2.0/24” and “10.10.3.0/24”, and be in Availability zones B and C respectively.

## Step 5.)

The next step is to create our target group. This will be the group (of EC2 instances) that our load balancer points to. Go to the EC2 dashboard, select “Target Groups” under the “Load Balancing” menu on the left, and create a new target group. Select “Instances” and give your target group a name. Leave the protocol set to “HTTP” and the port set to “80”. Make sure the VPC we just created is selected, not the default VPC.

Choose a target type

☒ **Instances**

- Supports load balancing to instances within a specific VPC.
- Facilitates the use of [Amazon EC2 Auto Scaling](#) to manage and scale your EC2 capacity.

☐ **IP addresses**

- Supports load balancing to VPC and on-premises resources.
- Facilitates routing to multiple IP addresses and network interfaces on the same instance.
- Offers flexibility with microservice based architectures, simplifying inter-application communication.
- Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.

☐ **Lambda function**

- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

☐ **Application Load Balancer**

- Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
- Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

Target group name

luit-tg

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol Port

HTTP : 80

You will be given the option to register your targets. Since we are using an autoscaling group, we won't do that since we have no way of knowing the instance IDs at this time.

## Step 6.)

Now that our target group has been created, it's time to set up our load balancer. Go to Load Balancers and create an application load balancer. I'll name mine "luit-alb". The "alb" lets me know that it is an application load balancer. The scheme will be "internet facing", which should be pre-selected already. Next select the VPC, and then under "mappings" check the boxes for the three subnets we just created.

**Network mapping** Info  
The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

**VPC** Info  
Select the virtual private cloud (VPC) for your targets. Only VPCs with an internet gateway are enabled for selection. The selected VPC can't be changed after the load balancer is created. To confirm the VPC for your targets, view your [target groups](#).

luit-vpc  
vpc-0ddbfca801de9d713  
IPv4: 10.10.0.0/16

↻

**Mappings** Info  
Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.

☒ **us-east-1a (use1-az1)**

Subnet  
subnet-047e8cc3bc039aa54 luit-public-A ▼

IPv4 settings  
Assigned by AWS

☒ **us-east-1b (use1-az2)**

Subnet  
subnet-09a3adbcebc4acc0 luit-public-B ▼

The next part is very important. We need to create a new security group. The area of security groups is where most people get confused and end up making mistakes. It helps me to know what to do if I just stop and think about what each resource needs to do in plain English. So in the case of this load balancer, it needs to be able to accept traffic via HTTP over the internet, so we need an inbound rule allowing all HTTP traffic. Then it needs to

forward traffic to our target group, which will point to our autoscaling group we will create in step 8.

So for inbound rules, choose HTTP as type and choose anywhere (IPv4) as source. In outbound rules choose HTTP as type and leave the source as anywhere. Make sure both ports are set to 80. Give the security group a name that references your load balancer. I named mine “luit-alb-sg”. Make sure you select the correct VPC.

Back on our load balancer set-up page, hit refresh next to security groups and select the security group we just created. Make sure you unselect the default security group as well.

Next you need to set a listener. Protocol should be set to HTTP, port set to 80, and then under “Default action”, forward to our target group. This is the step that allows the load balancer to forward and distribute traffic to the target group.

**Listeners and routing** Info  
A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener HTTP:80 Remove

Protocol

HTTP ▼

:

Port

80

1-65535

Default action

Info

Forward to

luit-tg

Target type: Instance, IPv4

HTTP ▼

↺

[Create target group](#)

Listener tags - *optional*

Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

Add listener tag

You can add up to 50 more tags.

## Step 7.)

After we've created the load balancer, the next step is to create a launch template. This will serve as a template for the EC2 instances we will deploy with the auto scaling group. Find "Launch Templates" under "Instances" on the left-hand menu.

Create a name and a description for your template. Choose linux 2 as the AMI, and t2.micro as instance type. Select a key pair, or you can create a new one. Under "Network Settings" choose "Don't include in launch template" as "Subnet".

Next we'll create a security group. We'll create an inbound security group rule with HTTP as "type", 80 as "port range", and the source will be the security group of our load balancer. This security group rule will allow HTTP traffic from the load balancer only. Also, make sure you select the correct VPC.



**Firewall (security groups)** [Info](#)  
 A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☐ Select existing security group
 ☒ Create security group

Security group name - *required*

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and .\_-:/()#,@[]+=&:;!\$\*

Description - *required* [Info](#)

VPC - *required* [Info](#)  
  
 10.10.0.0/16

Inbound security groups rules  
 ▼ Security group rule 1 (TCP, 80, sg-0ebbdb65d60bad9f6) Remove

Type [Info](#)   
 Protocol [Info](#)   
 Port range [Info](#)

Source type [Info](#)   
 Source [Info](#)   
 ×

Description - *optional* [Info](#)

*Scroll until you see the security group of your load balancer*

You'll also need to create a second security group rule allowing you to SSH into your instance. We'll need this for the next part of this project, when we stress test an instance. Choose SSH as Type, and My IP as Source.

Then you need to click on "Advanced network configuration" and enable auto-assigning of public IP. You don't need to touch anything else here.

▼ Advanced network configuration Set to "Enable"

**Network interface 1** Remove

Device index [Info](#)   
 Network interface [Info](#)   
 Description [Info](#)

Subnet [Info](#)   
 Not applicable for EC2 Auto Scaling

Security groups [Info](#)   
Show all selected (1)

Auto-assign public IP [Info](#)

Next, move down to "Advanced details" and scroll all the way down to the "User data" field. Paste in this code:

```
echo '<center><h1>This Amazon EC2 instance is located in Availability Zone: AZID  
sed "s/AZID/$EC2AZ/" /var/www/html/index.txt > /var/www/html/index.html
```

This will install and start Apache on our instances. I've also included some code that should display what availability zone our instance is in on the web page. When we test by going to the url of our load balancer, the availability zone that's displayed should change as we refresh.

## Step 8.)

The final step is to create the auto scaling group. You can do this from the launch template dashboard (go to actions), or from the auto scaling option at the bottom of the menu on the left.

As usual, name the auto scaling group, then select the launch template we just created. On the next page select the VPC and then all three subnets we just created. We want our instances deployed to each of these subnets.

**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**  
Choose the VPC that defines the virtual network for your Auto Scaling group.

vpc-0ddbfa801de9d713 (luit-vpc)  
10.10.0.0/16

[Create a VPC](#)

**Availability Zones and subnets**  
Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets

- us-east-1a | subnet-047e8cc3bc039aa54 (luit-public-A)  
10.10.1.0/24
- us-east-1b | subnet-09a3adbce4acc0 (luit-public-B)  
10.10.2.0/24
- us-east-1c | subnet-08e2348019bb8cc7c (luit-public-C)  
10.10.3.0/24

[Create a subnet](#)

Select all three subnets we created

Next, select “Attach to an existing load balancer” and choose the target group we created for the load balancer.

**Load balancing** [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** Choose the target group of the load balancer

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

☒ **Choose from your load balancer target groups**  
This option allows you to attach Application, Network, or Gateway Load Balancers.

☐ **Choose from Classic Load Balancers**

**Existing 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 ▼

luit-tg | HTTP ×  
Application Load Balancer: luit-alb

I would recommend turning on elastic load balancer health checks on that page as well. On the next page we want our minimum capacity to be 2, maximum capacity to be 5, and our desired capacity to be 3. Click next all the way through to review. Make sure everything is correct and create.

## Step 9.)

Now we just need to wait for the auto scaling group to launch and update capacity. It'll take a few moments. Once that happens we should be able to see our EC2 instances if we go back to the instances dashboard. What we want to be able to do is to view our web page from the load balancer url, but not from the public IPv4 address of our individual instances.

Go to the load balancer dashboard and you should see the DNS name of your load balancer. Copy and paste this url into your browser. You should see this page:

**This Amazon EC2 instance is located in Availability Zone: us-east-1a**

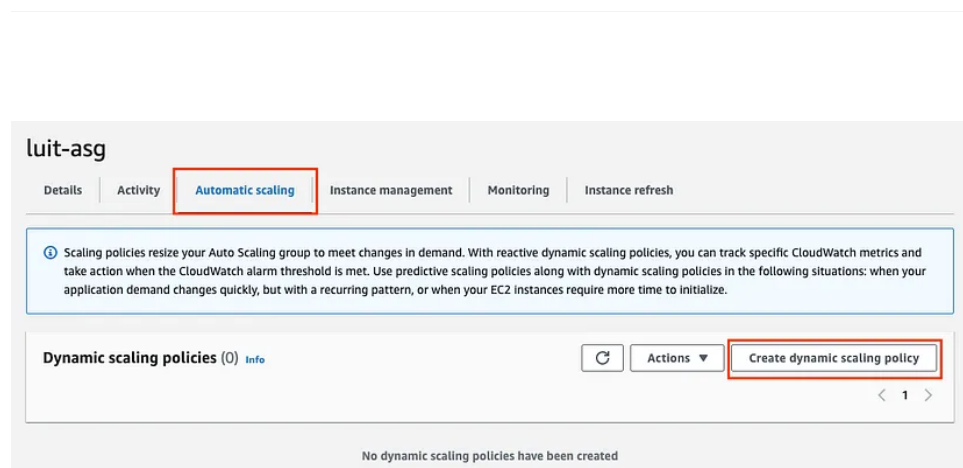
If you keep hitting refresh the zone should change to 1b and 1c, and back to 1a. If this works it means that the load balancer is working correctly. Now, go to your instances and try to copy and paste a public IPv4 address of one of the instances into your browser. You should not be able to reach the web page from that address.

## Phase 2) Add a dynamic scaling policy and test

We can add a scaling policy to the auto scaling group to ensure that it scales (creates new instances) based on certain parameters. In our case we want new instances to be created when cpu utilization of an instances rises above 50%. We'll test this by installing and running a stress tool on one of the instances.

### Step 1.)

To do this let's first create the dynamic scaling policy. Click on your auto scaling group and in the "Automatic scaling" tab go to "Dynamic scaling policies" and create.



Keep the default for policy type and give your policy a name. Metric type should be "Average CPU utilization" with a target value of 50. I also reduced the number of seconds before including in metric from the default 300 to 60, just to speed up monitoring. Then create the policy.

### Create dynamic scaling policy

Policy type  
Target tracking scaling ▼

Scaling policy name  
CPU\_50

Metric type  
Average CPU utilization ▼

Target value  
50

Instances need  
60 seconds warm up before including in metric

☐ Disable scale in to create only a scale-out policy

Cancel Create

## Step 2.)

Next we need to SSH into one of the instances (it doesn't matter which) and install the stress tool. First we need to install the Amazon Linux extras repository. Use the command:

```
sudo amazon-linux-extras install epel -y
```

epel stands for Extra Packages for Enterprise Linux. Now you can install stress with the following command:

```
sudo yum install stress -y
```

## Step 3.)

We're ready to run the stress test. Use the following command:

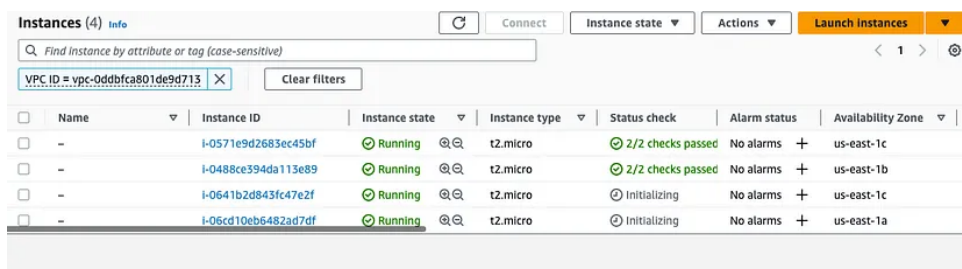
```
stress -c 4 -v
```

Once you see output like below it means your stress test is underway.

```
[ec2-user@ip-10-10-3-145 ~]$ stress -c 4 -v
stress: info: [3646] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
stress: debug: [3646] using backoff sleep of 12000us
stress: debug: [3646] --> hogcpu worker 4 [3647] forked
stress: debug: [3646] using backoff sleep of 9000us
stress: debug: [3646] --> hogcpu worker 3 [3648] forked
stress: debug: [3646] using backoff sleep of 6000us
stress: debug: [3646] --> hogcpu worker 2 [3649] forked
stress: debug: [3646] using backoff sleep of 3000us
stress: debug: [3646] --> hogcpu worker 1 [3650] forked
```

## Step 4.)

Go back to the AWS console and click the monitoring tab for the instance we are stressing. There is a box for CPU utilization, and you can also make that window large. After several minutes you should see the cpu rise above 50%. Once it rises high enough we should be able to go back to instances and see that another instance has been created. Again, this may take a little time, so use the refresh button and be patient.



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	-	i-0571e9d2685ec45bf	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c
<input type="checkbox"/>	-	i-0488ce394da113e89	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b
<input type="checkbox"/>	-	i-0641b2d843fc47e2f	Running	t2.micro	Initializing	No alarms	us-east-1c
<input type="checkbox"/>	-	i-06cd10eb6482ad7df	Running	t2.micro	Initializing	No alarms	us-east-1a

If you see a fourth instance being created it means that the scaling policy worked. Now that we know everything works you can delete the auto scaling group, the load balancer, target group, any security group rules, launch template and VPC (which will also delete subnets, routing table and internet gateway). The only thing you have to delete if you don't want to be charged is the auto scaling group (you can also set min, max and desired capacity to zero if you want to keep the scaling group up without actually creating any instances)

AWS

Aws Ec2

Cloud Computing

DevOps


Auto Scaling Groups

## Sign up for Best of Towards AWS

By Towards AWS

A monthly newsletter by Towards AWS. [Take a look.](#)

By signing up, you will create a Medium account if you don't already have one. Review our [Privacy Policy](#) for more information about our privacy practices.

 Get this newsletter

