Scaling: Remove Instances

We'll cover the following

Objective

- Replace explicit EC2 instances with Auto Scaling.

Steps

- Remove `Instance` and `Instance2`.

Removing the explicit instances

Now that our new ASG instances are up and are serving requests through our load balancer, we can safely remove our explicit instances without causing any disruption. To do so, we just need to remove every reference to them from our `main.yml` script, and redeploy:

- The entire `Instance` resource.
- The entire `Instance2` resource.
- The entire `Targets` property from the `LoadBalancerTargetGroup` resource.
- The `Ec2TagFilters` property from the `StagingDeploymentGroup` resource.
- The `InstanceEndpoint` and `InstanceEndpoint2` outputs.

Now, let's redeploy our infrastructure by running `deploy-infra.sh`.

```
./deploy-infra.sh


========== Deploying setup.yml ==========

Waiting for changeset to be created..
```

```
No changes to deploy. Stack awsbootstrap-setup is up to date



========== Deploying main.yml ==========

Waiting for changeset to be created..
Waiting for stack create/update to complete
Successfully created/updated stack - awsbootstrap
[
    "http://awsbo-LoadB-13F2DS4LKSCVO-10652175.us-east-1.elb.amazonaws.com:80"
]
```

terminal

If we hit the load balancer endpoint now, we should see our traffic split between the two instances in our ASG. The other two instances have been terminated.

```
for run in {1..20}; do curl -s http://awsbo-LoadB-13F2DS4LKSCVO-10652175.us-east-1.elb.amazonaws.c
9 Hello World from ip-10-0-50-202.ec2.internal
11 Hello World from ip-10-0-68-58.ec2.internal
```

terminal

It's time to checkpoint all our changes and push them to GitHub.

```
git add main.yml
git commit -m "Remove explicitly created EC2 instances"
git push
```

terminal

> **Note:** All the code has been already added and we are pushing it on our repository as well.

This code requires the following API keys to execute:

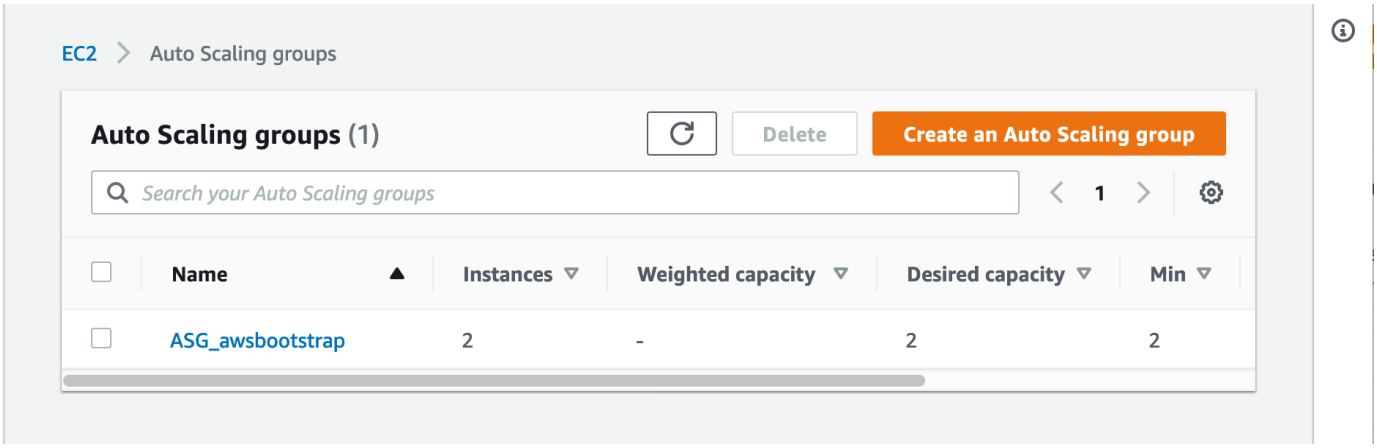
username	Not Specified...
AWS_ACCESS_KE...	Not Specified...
AWS_SECRET_AC...	Not Specified...
AWS_REGION	us-east-1
Github_Token	Not Specified...

```
const { hostname } = require('os');
const http = require('http');
const message = `Hello World from ${hostname()}\n`;
const port = 8080;
const server = http.createServer((req, res) => {
```

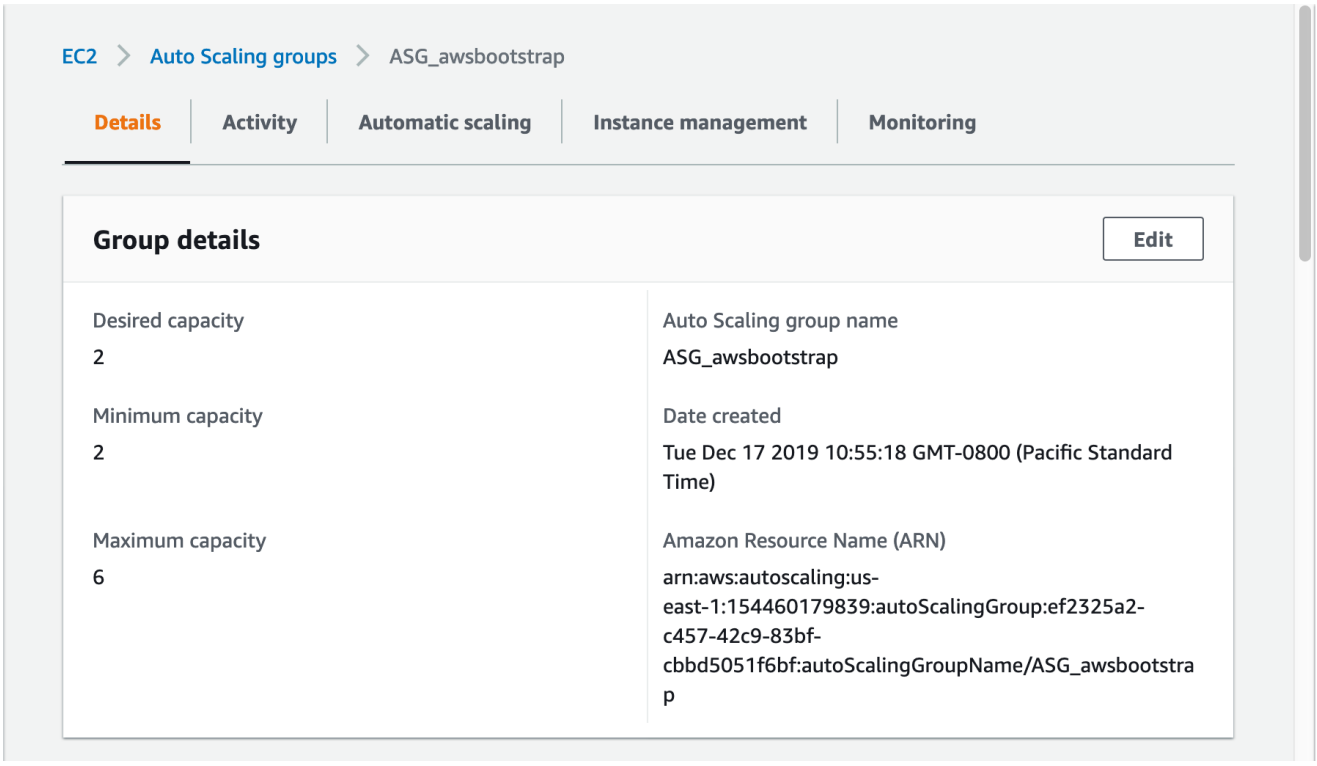
```
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end(message);
});
  server.listen(port, hostname, () => {
    console.log(`Server running at http://${hostname()}:${port}/`);
});
```

Adding capacity in a pinch is much easier now. From the Auto Scaling console (FIGURE 1), you can select your ASG (FIGURE 2) and edit the desired capacity setting (FIGURE 3). The number of EC2 instances will automatically reflect your desired capacity within a few seconds.

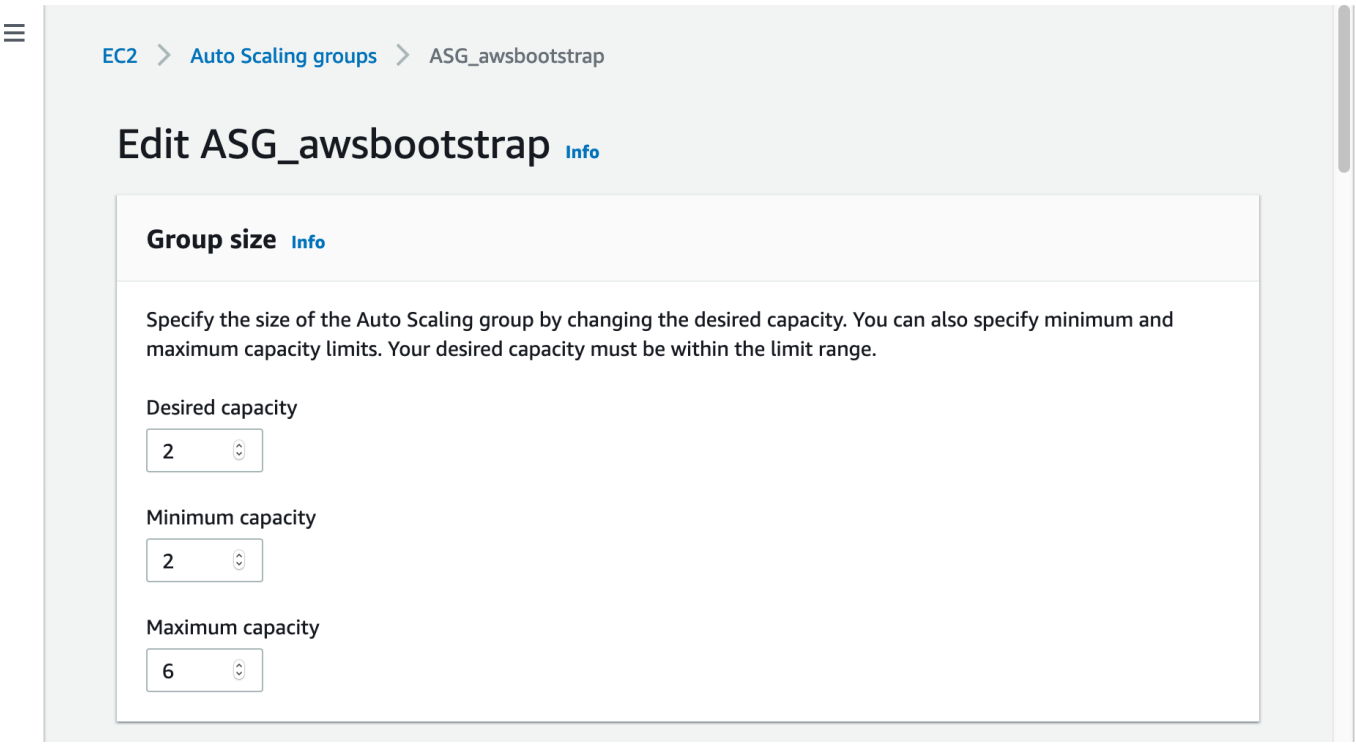
> **NOTE:** All these figures are shown below.



Auto Scaling Groups (FIGURE 1)

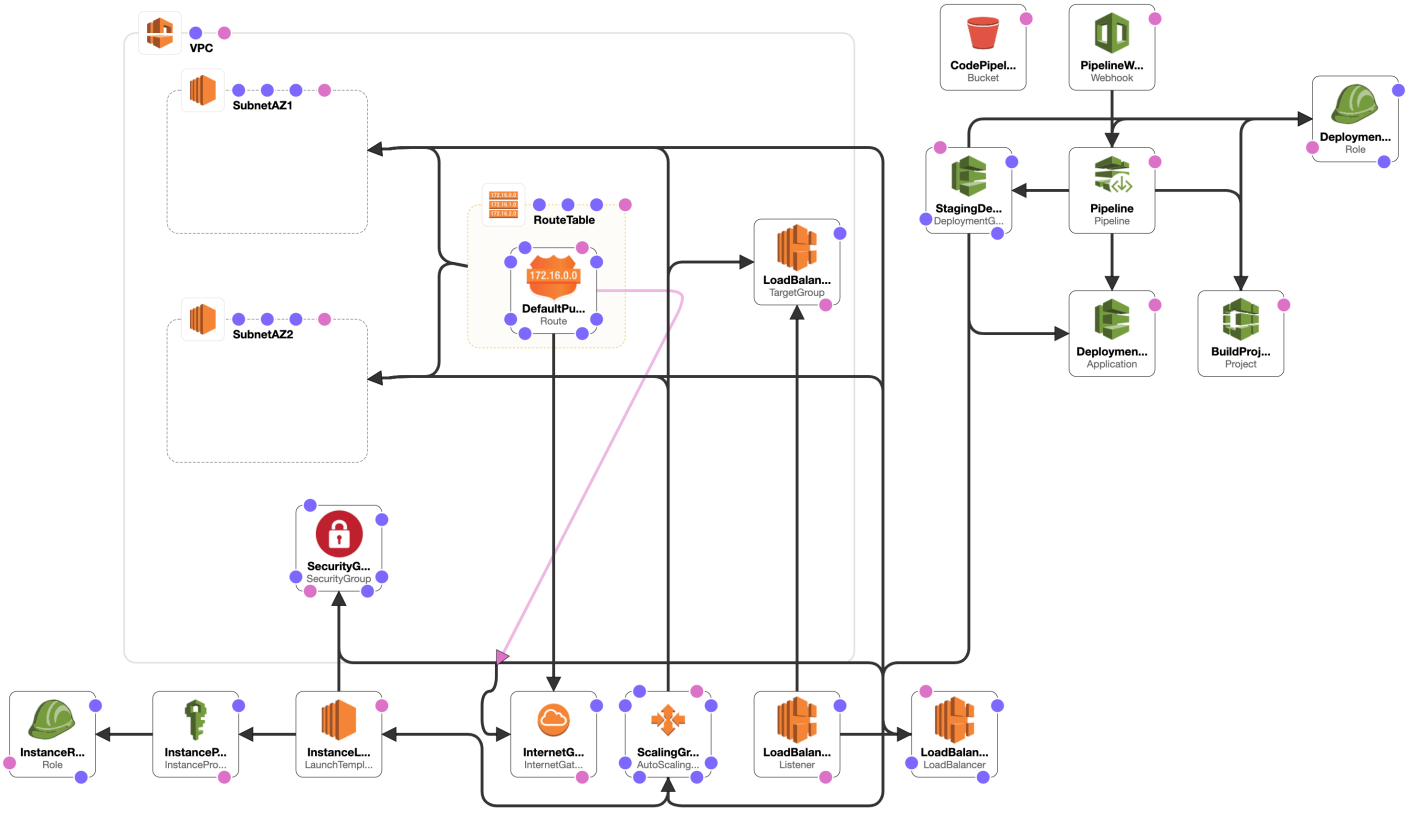


Auto Scaling Details (FIGURE 2)



Auto Scaling Edit Group Size (FIGURE 3)

In order to get a pictorial view of our developed cloudformation stack so far, below is the design view which shows the resources we created and their relationships.



Add an Auto Scaling Group

In the next lesson, we will create separate environments for staging and production.