Lab 5 – AutoScaling

EC2 Autoscaling – In EC2, Auto Scaling is the ability to scale resources based upon the demand for your application. As the workload increases, Auto Scaling can provision additional resources to absorb requests or other workloads to ensure the performance and availability of your application meets pre-defined minimum standards. As that workload decreases, Auto Scaling can decrease capacity to reduce costs while maintaining a minimal viable configuration. Auto Scaling is also used to help ensure the health and availability of your fleet and ensuring that applications are still accessible in the event of an AWS outage.

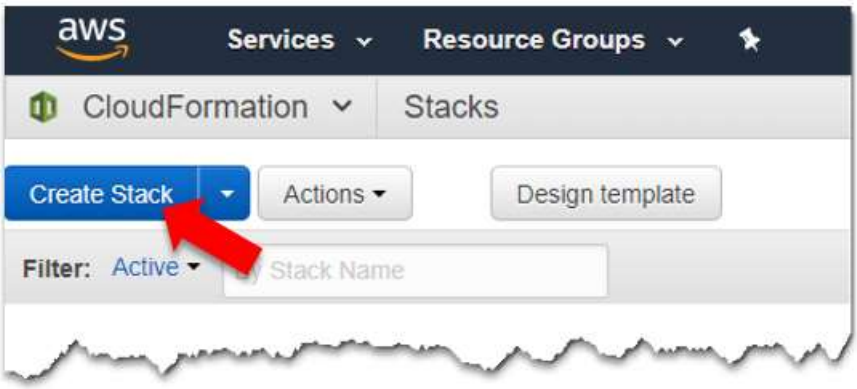
Goal - In this session, users will learn how to deploy an application in a highly scalable fashion to support the scaling of resources on-demand. Participants will configure autoscaling for a web application and experience how changes in the environment can trigger the addition, or removal, of resources as required by the application.

Prerequisites – A VPC with public subnets that allow internet access for resources with public IP addresses. If you've completed Lab 1, you're all set to go.

---

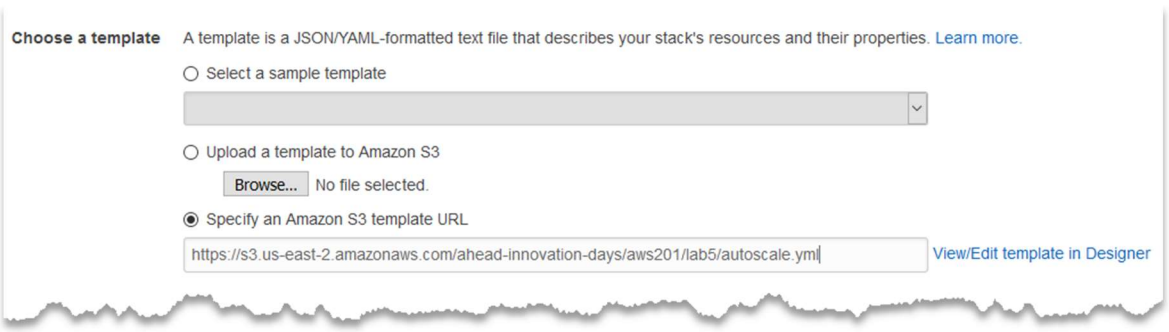
1. Navigate to the **CloudFormation** console.

2. Create a new stack by clicking the **Create Stack** button at the top.



3. Click the radio button next to **Specify an Amazon S3 template URL**. Enter the following URL in the text box.

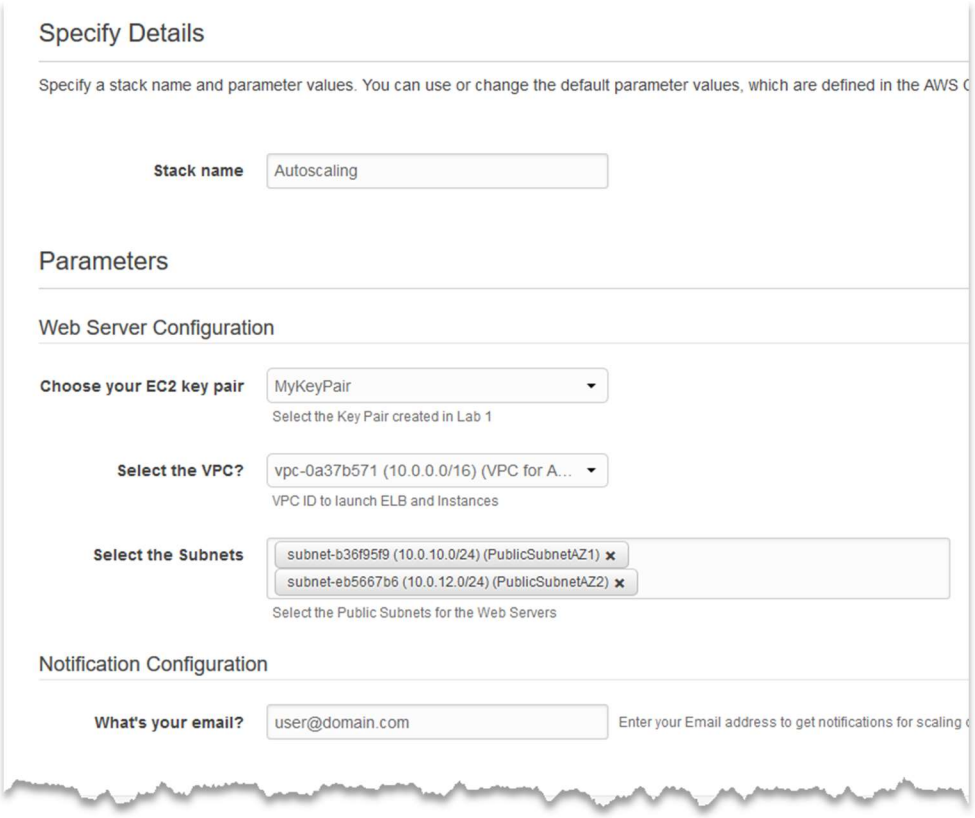
https://s3.us-east-2.amazonaws.com/ahead-innovation-days/aws201/lab5/autoscale.yml



4. In the Stack name text box, type **Autoscaling**.

5. In the Parameters section, provide the values as listed below:

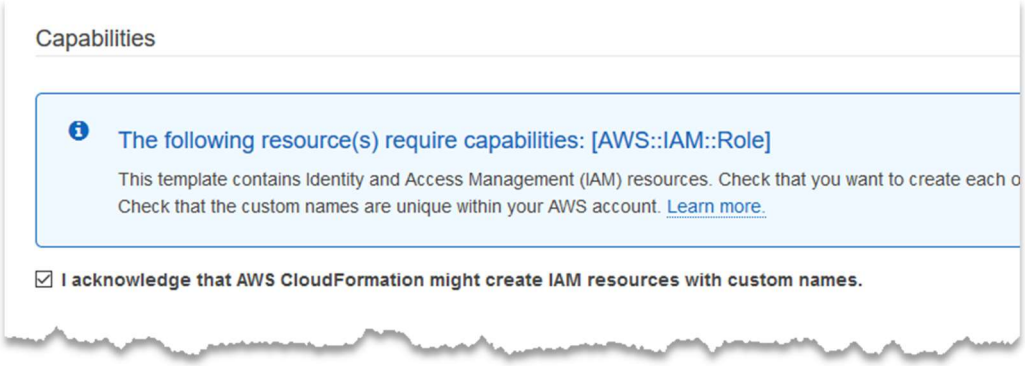
- **Web Server Configuration**
  - **EC2 Key Pair –** Select the Key Pair created at the beginning of Lab 1
  - **VPC:** Select 'VPC for AHEAD Innovation Days' from the list
  - **Instance Subnets:** Select 'PublicSubnetAZ1' and 'PublicSubnetAZ2' from the list
- **Notification Configuration**
  - **Email Address –** Enter your email address to receive alerts for autoscaling operations.



6. Click **Next**.

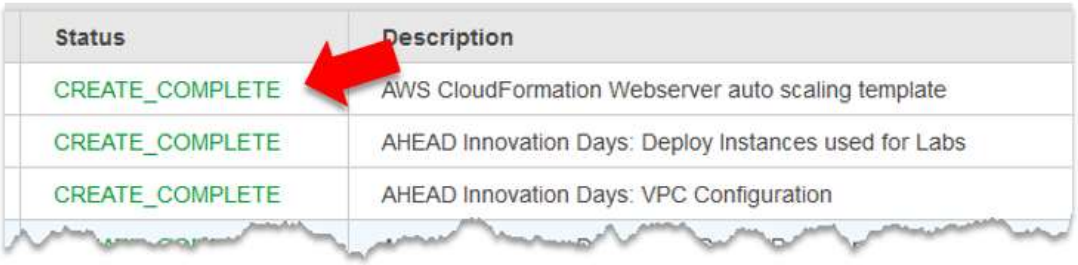
7. On the Options page, leave the defaults and select **Next**.

8. On the Review page, **click the checkbox** to acknowledge that the template will create an IAM role.

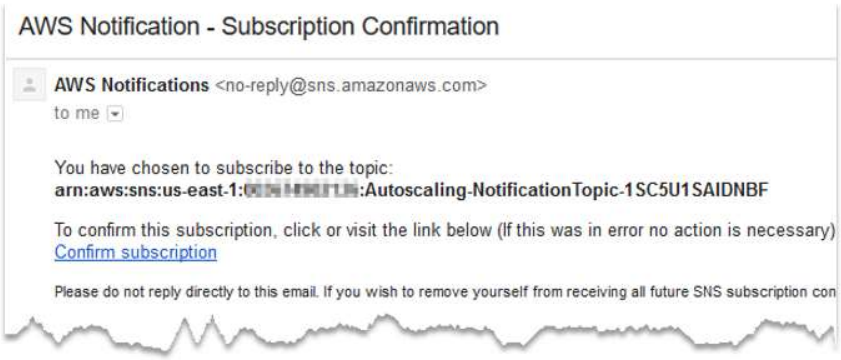


9. Click **Create** to launch the stack.

10. In the CloudFormation console, **refresh** the window until the status for the Autoscaling stack changes to CREATE_COMPLETE.



11. Once the template successfully deploys, make sure to **confirm the SNS topic subscription** created by the template using the email you received during the deployment.



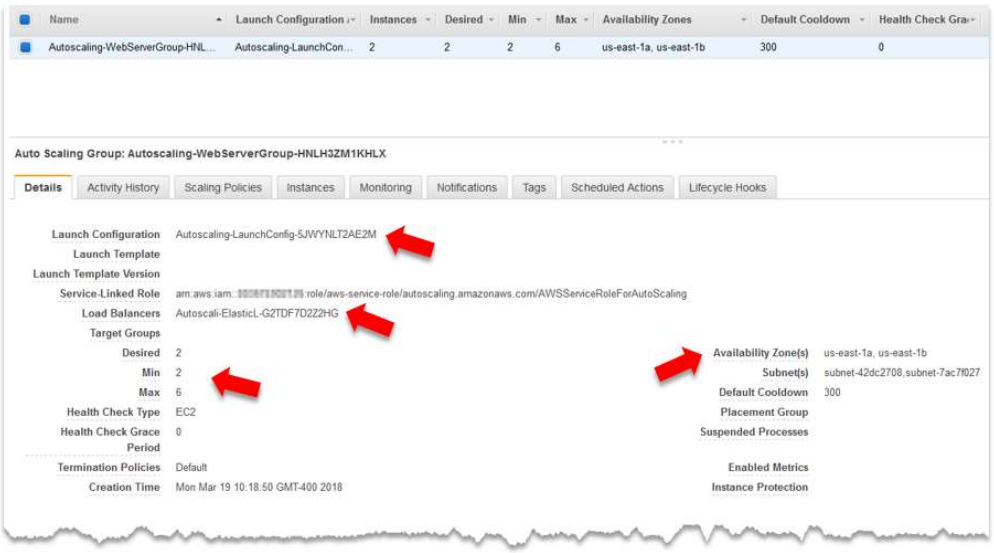
12. Before you make any changes, let's take a glance at what the CloudFormation template created. Browse to the **EC2** console and click on **Auto Scaling Groups** in the AUTO SCALING section. Take a look at the configuration for this Auto Scaling Group in the lower pane, starting with the Details tab:

- **Details Tab:** Notice that this Auto Scaling Group is using the Launch Configuration that was created from the CloudFormation template. It also uses an Elastic Load Balancer for the instances to provide load balancing and redundancy for incoming traffic. The Auto Scaling Group will provision instances across two Availability Zones and will start with a minimum of two (2) instances, scaling up to a maximum of six (6).

- **Activity History Tab:** Notice that the ASG has already created the two (2) instances as the minimum deployment.

- **Scaling Policies Tab:** There are two (2) simple auto scaling policies for this lab, one that scales up if any of the existing instance's CPU Utilization breaches 40% for over 60 seconds. For this example, assume that this means a spike in traffic and more resources are needed to properly serve the application. The second policy is for scaling down – if the instance's CPU Utilization is below 20% for 120 seconds, assume demand has decreased, and Auto Scale can reduce the number of instances needed to serve the application.

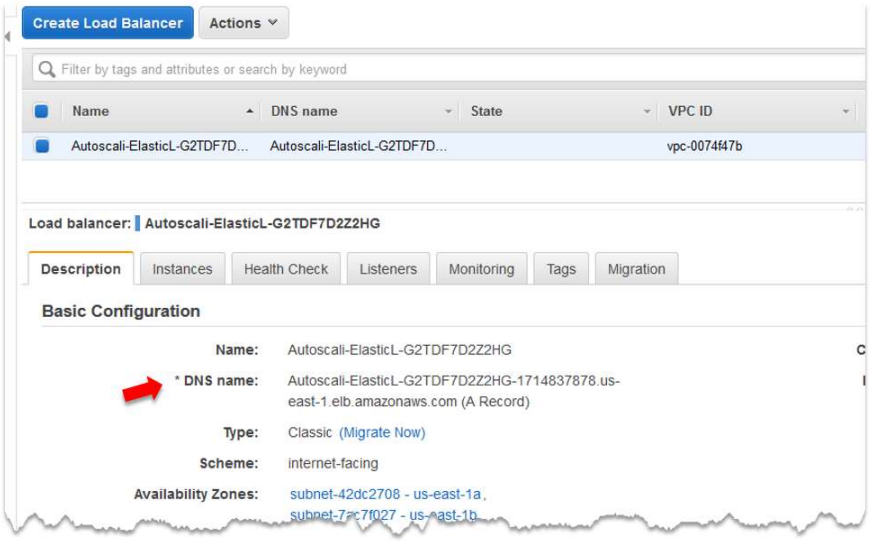
- **Instances Tab:** Information on the two initial instances currently running. Notice that they are in separate Availability Zones to provide redundancy.



Now that you understand what was deployed, let's put it to work to see how Auto Scaling can dynamically respond to events in the environment.

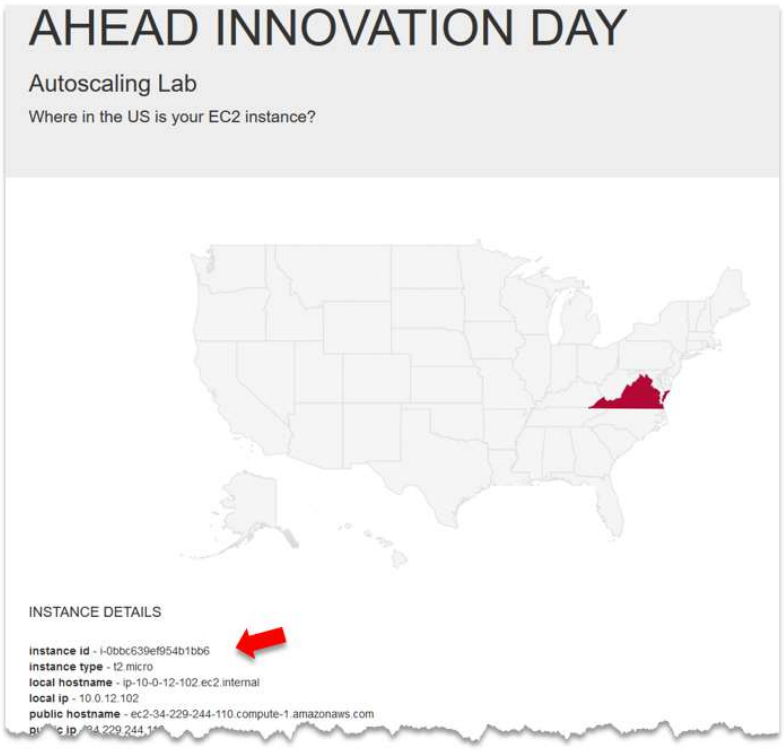
13. In the EC2 console, select **Load Balancers** on the left navigation pane.

14. **Select the box** next to the load balancer, if not already selected, and view the **Description** tab in the lower pane. Copy the **URL** listed for the **DNS Name**.



15. **Paste the URL into your web browser** and view the website that displays. The website is the test application that you want to ensure uptime and availability for using the Auto Scaling Group.

    ***Note**: As stated, there are two (2) instances serving this application. Feel free to refresh your browser and see how the Instance Details at the bottom change as the request gets directed back and forth between the supporting instances.*



Now that you've seen that the deployed infrastructure works as deployed, you're going to see EC2 Auto Scale in action. To do this, you're going to use a **stress** tool that was installed on the instances during the provisioning process. You're going to use the tool to create artificial workloads for the CPU and see how Auto Scale responds.

16. In the EC2 console, click **Instances** on the left. Again, notice there are only two (2) instances running the application right now – both named **WebServer**.

17. In the navigation pane, click **Run Command** under the SYSTEMS MANAGER SERVICES section.

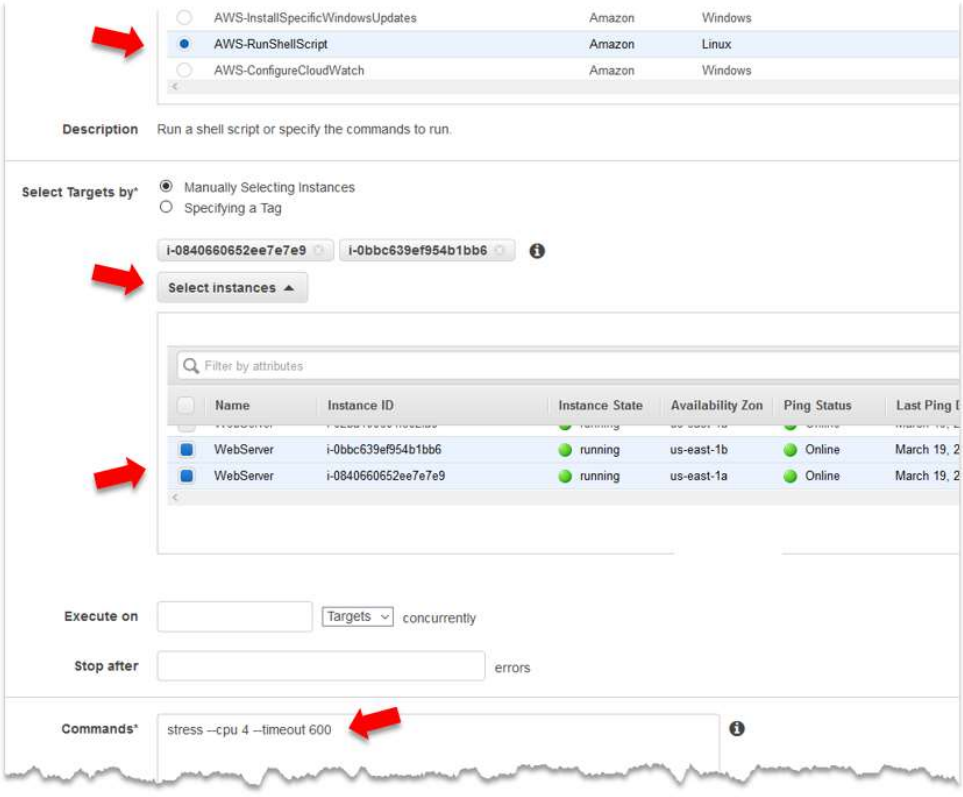
18. Click the **Run a command** button.

19. In the command document section, select the radio button next to **AWS-RunShellScript** from the list.

20. For the **Select Targets by*** section, select **Manually Selecting Instances** and click the **Select Instances** button. Select the two instances named **WebServer**.

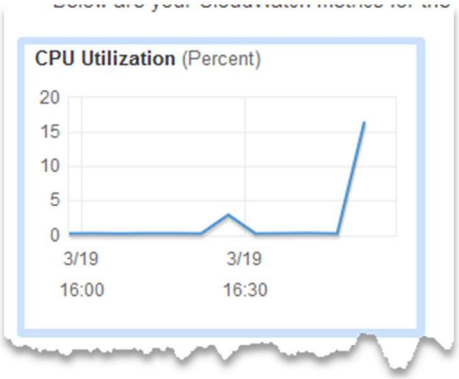
21. In the **Commands*** section, type the following command:

```
stress --cpu 4 --timeout 600
```



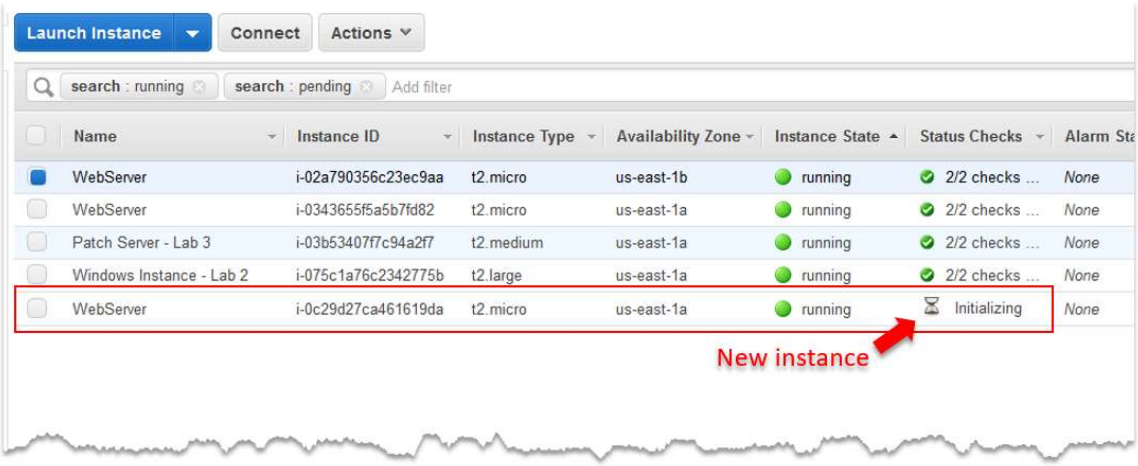
22. Click **Run**.

23. In the EC2 console, select **Instances** and select one of the **WebServer** instances. Click the **Monitoring** tab at the bottom and **refresh** the graph until you see the CPU Utilization spike.



24. At the top, **refresh** the list of instances running in your account. Continue to refresh until you see additional EC2 instances being provisioned. As configured, Auto Scaling is reacting to the CPU spike by provisioning additional resources to absorb the workload that you artificially created on the hosts.

    Note: Auto Scale will eventually scale up to the maximum number of instances for the Auto Scaling Group, which was defined as **six (6).**

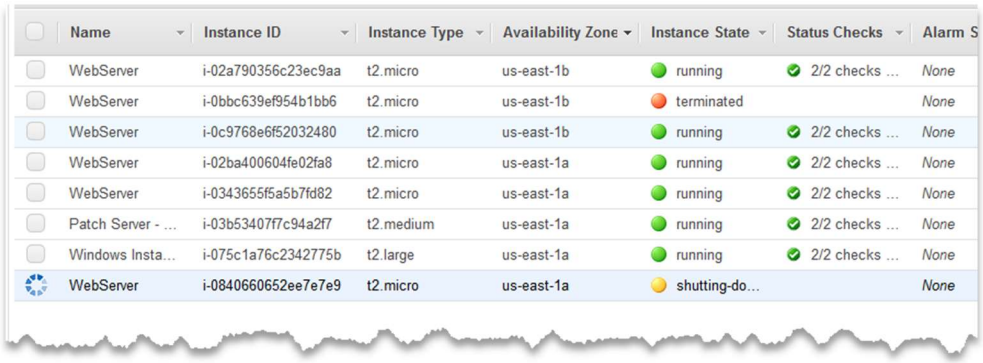


25. On the left, click **Load Balancers**. Click on the load balancer and select the **Instances** tab. Notice that the new instances are automatically being placed behind the load balancer so they can receive web requests for the application.

26. In your web browser, click **refresh** on the application webpage to see that requests are now being handled by six (6) different hosts, rather than just the original two instances that were initially provisioned.

    Now that you've seen Auto Scale react to an increase in workloads you're going to see what happens when the workload is **reduced** and returns to normal conditions. The command that was run against the instances earlier had a timeout, meaning that it only stressed the CPU for 5 minutes and stopped afterward. The timeout will cause the CPU to drop back down to almost 0% CPU Utilization in which the scale down policy will be executed.

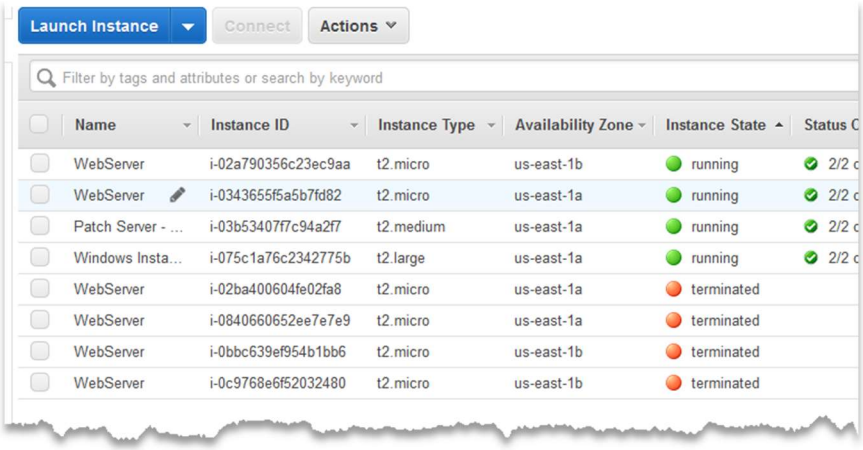
27. In the EC2 console, click **Instances**. Continue to **refresh** the list of instances. After a few minutes, you'll see that Auto Scale starts to **Shutdown and Terminate** one instance at a time.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm S
WebServer	i-02a790356c23ec9aa	t2.micro	us-east-1b	running	2/2 checks ...	None
WebServer	i-0bbc639ef954b1bb6	t2.micro	us-east-1b	terminated		None
WebServer	i-0c9768e6f52032480	t2.micro	us-east-1b	running	2/2 checks ...	None
WebServer	i-02ba400604fe02fa8	t2.micro	us-east-1a	running	2/2 checks ...	None
WebServer	i-0343655f5a5b7fd82	t2.micro	us-east-1a	running	2/2 checks ...	None
Patch Server - ...	i-03b53407f7c94a2f7	t2.medium	us-east-1a	running	2/2 checks ...	None
Windows Insta...	i-075c1a76c2342775b	t2.large	us-east-1a	running	2/2 checks ...	None
WebServer	i-0840660652ee7e7e9	t2.micro	us-east-1a	shutting-do...		None

Eventually, Auto Scale will **terminate** all but **two (2)** instances, which is the minimum for the application. Keep in mind that the remaining two instances serving the application will not likely be the same instances that you originally provisioned. In this case, you shouldn't care, as they are all serving the same application.

Feel free to **refresh** the webpage again to see that it's now being supported by the two remaining EC2 instances that remain from the Auto Scale event.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status C
WebServer	i-02a790356c23ec9aa	t2.micro	us-east-1b	running	2/2 c
WebServer	i-0343655f5a5b7fd82	t2.micro	us-east-1a	running	2/2 c
Patch Server - ...	i-03b53407f7c94a2f7	t2.medium	us-east-1a	running	2/2 c
Windows Insta...	i-075c1a76c2342775b	t2.large	us-east-1a	running	2/2 c
WebServer	i-02ba400604fe02fa8	t2.micro	us-east-1a	terminated	
WebServer	i-0840660652ee7e7e9	t2.micro	us-east-1a	terminated	
WebServer	i-0bbc639ef954b1bb6	t2.micro	us-east-1b	terminated	
WebServer	i-0c9768e6f52032480	t2.micro	us-east-1b	terminated	

You Did It. You are done with this Lab.

Conclusion: In this lab, participants used EC2 Auto Scale to provision a new application. Using a stress tool pre-installed on the instances, participants created artificial workload to watch how Auto Scale reacted, scaling up resources as the workload increased and scaled down when the workload decreased.