Global Knowledge®

Expert Reference Series of White Papers

Introduction to Amazon Auto Scaling

Introduction to Amazon Auto Scaling

Jon M. Gallagher, Global Knowledge Instructor, Certified AWS Solutions Architect, Certified AWS SysOps Administrator, Authorized Amazon Instructor

Introduction

Cloud computing is an exciting suite of technologies that has come to dominate the discussion of computing for two main reasons:

- It can provide the flexibility to quickly deploy computing environments that are not only properly configured for current needs, but that can also expand or contract according to future needs.
- It can help an organization save money.

Cloud computing can deliver flexibility and cost savings because it is uniquely capable of being scaled to the "right size" for a particular environment, no matter how frequently usage of the environment expands or contracts. Taking advantage of this flexibility can be difficult; however, some companies try to do it manually, while others create a custom automated system. Either method has the potential of introducing new challenges into your environment:

- Depending on manual processes to start new servers, stop unneeded systems, or change allocated storage space is expensive, slow, and worst of all, error prone. Any savings from moving environments to the cloud can easily be erased by the costs of manual intervention.
- Creating an automated scaling system customized for your organization's needs takes a long time, almost certainly costs more than planned for, and requires a risky test and deployment phase. Such a system also requires its own hardware, software, and support environment that scales itself for expansion or contraction.

As with so many of its other services and products, Amazon Web Services (AWS) created Auto Scaling to solve its own scaling issues, and now provides the service to its customers for free. On its own, Auto Scaling monitors your environment to ensure that the desired systems stay running. What is even more powerful is that you can tie the Amazon CloudWatch monitoring service into Auto Scaling, which allows your environment to automatically scale up or down based on current conditions:

- As load increases, Amazon CloudWatch can call Auto Scaling to add new computing capacity.
- As load decreases, Amazon CloudWatch can trigger Auto Scaling to shed computing capacity and reduce cost.

This paper describes what Auto Scaling is, when to use it, and provides an example of setting up Auto Scaling.

What is Amazon Auto Scaling?

Using AWS to meet your business needs is easy because of free services like Auto Scaling. Without Auto Scaling, you may have no problems configuring an initial environment in AWS, but dynamic growth can quickly exceed your existing team's ability to respond. Hiring enough operators to respond to your computing environment's conditions 24/7 is too expensive. Guessing the proper environment size means that the systems are probably over-provisioned, which wastes money. Finally, ensuring that all the systems you want running are actually operational can be overwhelming, especially if an Availability Zone (AZ) fails.

Auto Scaling allows you to specify a server group, called an Auto Scaling Group (ASG), in which you define:

- A minimum number of servers to run (in this context, *servers* refers to EC2 instances configured to run your software)
- A maximum number of servers to run
- Optionally, an initial number of servers to run
- The AZs in which you want the servers to run

When you complete the ASG definition, Auto Scaling starts the number of servers you specified. The system distributes the servers as evenly as possible across the designated AZs.

If any server stops working, the ASG replaces the server. If an AZ goes out of service, the remaining AZs have their servers increased to the specified amount.

The ASG definition can also include rules that adjust the number of servers running and the minimum/maximum number of servers based on system conditions or schedules. For example, you may want your environment to respond to changes in CPU percentage over the whole group, or you may want to ensure a certain number of servers are running at the start of business hours. Note that these rules exist to not only expand your environment as needed, but to contract it: you may want to decrease the number of servers running at the end of business hours. Expansion and contraction are equally important, and taking advantage of both can lead to real cost savings.

Auto Scaling consists of the following four components, which are described in the following subsections:

- Launch Configuration (LC)
- Auto Scaling Group (ASG)
- Auto Scaling Policy (ASP)
- Scheduled Action (SA)

Launch Configuration (LC)

The Launch Configuration (LC) tells Auto Scaling how to configure the EC2 instances it launches (not only the initial instances, but any it might add to increase capacity), including which type of EC2 instance to start, which Amazon Machine Image (AMI) to load onto the EC2 instance, how to configure the software running on the EC2 instance once it boots, how to secure the EC2 instance, and any information you might want to associate with the EC2 instance (such as a name, tags, and so on).

Because an LC is just a description of an EC2 runtime, one LC can be used in multiple ASGs. A particular ASG can have only one active LC, but the LC can be changed at any time, even while servers are running.

Auto Scaling Group (ASG)

An Auto Scaling Group (ASG) defines where the EC2 instances described in a LC will run, what the minimum number of instances must be, and what the maximum number of instances must be. An ASG can also specify the number of instances the ASG should start with (what the Auto Scaling documentation refer so as the desired number of instances).

In defining where to run the instances described by the LC, the ASG lets you specify one or more AZs within a region in which to run. Because each AZ is essentially its own data center, this means the ASG can guarantee that the EC2 instances it controls will always be running. In addition, the ASG can be associated with an Elastic Load Balancer (ELB) and guarantee that the ELB will always have a healthy pool of servers behind it.

Auto Scaling Policy (ASP)

An Auto Scaling Policy (ASP; referred to as *policy* in this white paper) defines how an ASG should be changed, which includes whether to increase or decrease the number of EC2 instances by a certain percentage or by an exact number, or to change to an exact number of EC2 instances.

A policy is usually triggered by an Amazon CloudWatch condition. For example, if the ELB that is associated with an ASG is experiencing increasing latency, Amazon CloudWatch can trigger an alarm that triggers a policy to increase capacity. If ELB latency is below a certain threshold, Amazon CloudWatch can trigger a policy that decreases capacity.

The number of EC2 instances that a policy can add to or delete from an ASG is limited by the maximum and minimums set in the ASG.

Scheduled Action (SA)

A Scheduled Action (SA) is an action that is triggered at a specific time to decrease or increase the number of EC2 instances running in an ASG. An SA can be defined to run at a specific date and time, or to run at a recurring time and date. For example, you can set an SA to run at the end of a workday to run a minimum level of servers, and another SA to run at the beginning of the workday to increase the number of servers to hold load during the workday.

When to Use Auto Scaling

Auto Scaling is appropriate for many diverse scenarios, but these are the most common:

- Auto Scaling for high availability
- Auto Scaling to meet system demand
- Auto Scaling to control costs
- Auto Scaling to deploy systems

These scenarios are described in the following subsections.

Auto Scaling for High Availability

When you define an ASG, it immediately starts the number of servers you defined. When all the servers have started, the ASG monitors them; if any fail, the ASG starts new servers to replace them as defined by the LC.

If you want your servers to be highly available, make sure to define multiple AZs in your ASG. This causes the ASG to start one EC2 instance in each AZ up to the minimum or defined number, and protects you as follows:

- If you run one server and you define multiple AZs, and if the AZ you are currently running in fails, then the ASG starts the single server in a different AZ.
- If you run multiple servers and you define multiple AZs, and if an AZ fails, then the ASG starts new servers up to your minimum or defined value, and distributes them across the remaining defined AZs.

Thus, without any additional configuration, you are guaranteed that the ASG keeps your servers running until it runs out of defined AZs. Because there is no cost or compute burden in having multiple AZs, it is highly recommended that you include all of a region's AZs in every ASG.

Auto Scaling to Meet System Demand

The hardest part of sizing infrastructure is when you are asked to predict the future. Using non-cloud infrastructure, you must predict the future to purchase the equipment to meet your anticipated demand.

Guessing too low means disappointing customers, which can endanger the project or your business. Guessing too high means you spent too much money, which can also endanger the project or your business. But even if you are using cloud computing for your environment, you may run into problems scaling manually or developing systems to scale.

Using Amazon Auto Scaling means that you can guess at an initial usage level and let Auto Scaling make the corrections necessary to meet the actual usage of the system. Simply identify an Amazon CloudWatch metric (such as latency at the ELB, or CPU utilization averaged across the ASG), and use policies to change the ASG to react to the metric.

If you have predicted the wrong EC2 instance size, it's easy to change your EC2 instance type:

- Create a new LC with the new EC2 instance type.
- Associate the new LC with your current ASG.
- Increase the size of your server fleet by 100%. You now have 50% of the servers running under the old LC, and 50% running under the new LC.
- Decrease your fleet by 50%. Auto Scaling automatically stops the servers running under the old LC.

Your environment is now running using the new LC and the new EC2 instance type.

Auto Scaling to Control Costs

Because you can use Auto Scaling to expand or contract your environment as conditions change, you can control costs. As in the previous scenario, you can use Amazon CloudWatch to specify metrics that indicate there is too much capacity, and decrease your environment's size. If the EC2 instances are too expensive for the compute power needed, just change the instance type in the LC and swap out the instances.

Amazon CloudWatch also has metrics that track billing. You can set Amazon CloudWatch to adjust your ASGs based on how much or how quickly costs are increasing.

Auto Scaling to Deploy Systems

As mentioned in the previous sections, you can change the LC associated with an ASG. In addition to the ability to change the EC2 instance type the ASG runs, you can also change the LC to use a different AMI to run on the EC2 instance.

If you distribute your systems as fully integrated AMIs, or as AMIs that can self-configure on boot, you can create a new LC that uses the new AMI and change the ASG to use the new LC. Then, as previously, use a policy that increases the size of your server fleet by 100%, and then once all the instances are booted, decrease the fleet by 50%. By default, the ASG drops the instances that were started with the old LC, meaning that your fleet is completely updated.

An Example of Setting up Auto Scaling

This section provides an example of how to set up Auto Scaling for the most common scenario: using Auto Scaling with an ELB to handle incoming load traffic.

Note: This example assumes that you have some experience in starting EC2 instances. If you do not, please take advantage of the free training videos and labs provided by Amazon at this link: http://aws.amazon.com/training/intro_series/

The Auto Scaling environment created in this example consists of the following elements:

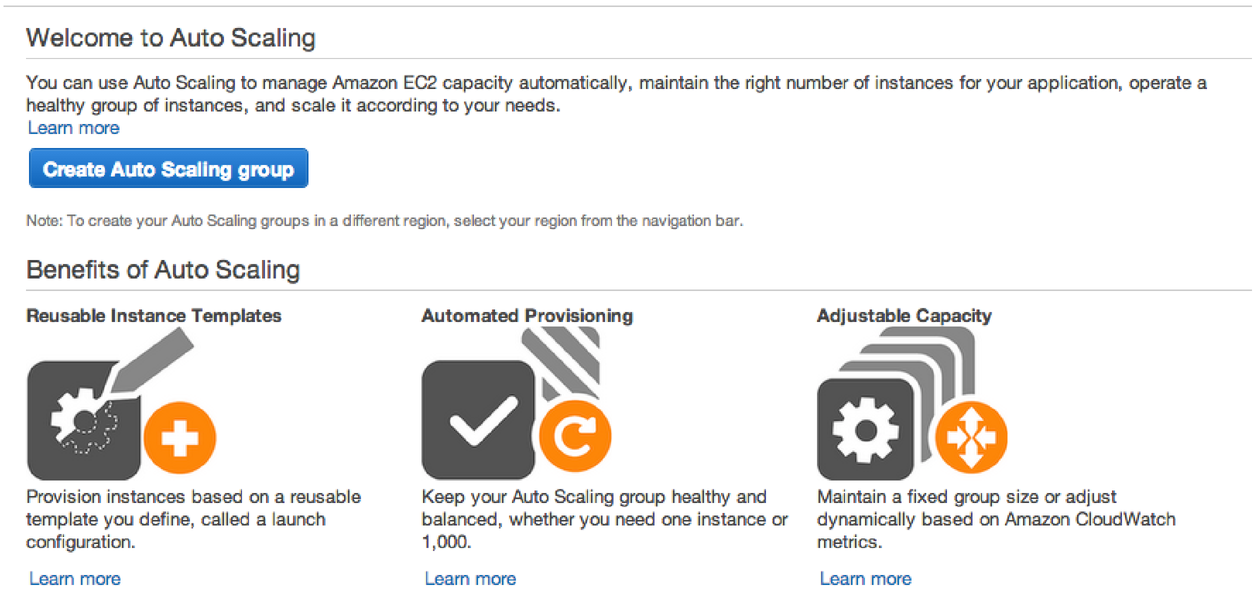
- The ASG will start with two servers, have a minimum of one server, and a maximum of 10 servers
- The policies will increase the number of servers by two, decrease the number of servers by two, and tie these policies to alarms that go off when:
  - There is too much CPU utilization by the ASG (the number of servers will increase)
  - There is too little CPU utilization (the number of servers will decrease)

This example environment is accomplished by creating an LC, an ASG, and Auto Scaling Policies. However, it does not include any Scheduled Actions (SAs).

Note: This example is an illustration of the process so you can see the minimal scope of effort involved. It is not a step-by-step tutorial. For detailed steps, see the Auto Scaling documentation at http://aws.amazon.com/documentation/autoscaling/.

Step 1. Create a Launch Configuration (LC)

To create an LC, this example starts the AWS Management Console and displays the EC2 screen for its region. At the bottom left of the screen, either Launch Configurations or Auto Scaling Groups is clicked, which launches the Welcome To Auto Scaling screen:



Welcome to Auto Scaling

You can use Auto Scaling to manage Amazon EC2 capacity automatically, maintain the right number of instances for your application, operate a healthy group of instances, and scale it according to your needs.
Learn more

Note: To create your Auto Scaling groups in a different region, select your region from the navigation bar.

Benefits of Auto Scaling

Reusable Instance Templates

Provision instances based on a reusable template you define, called a launch configuration.
Learn more

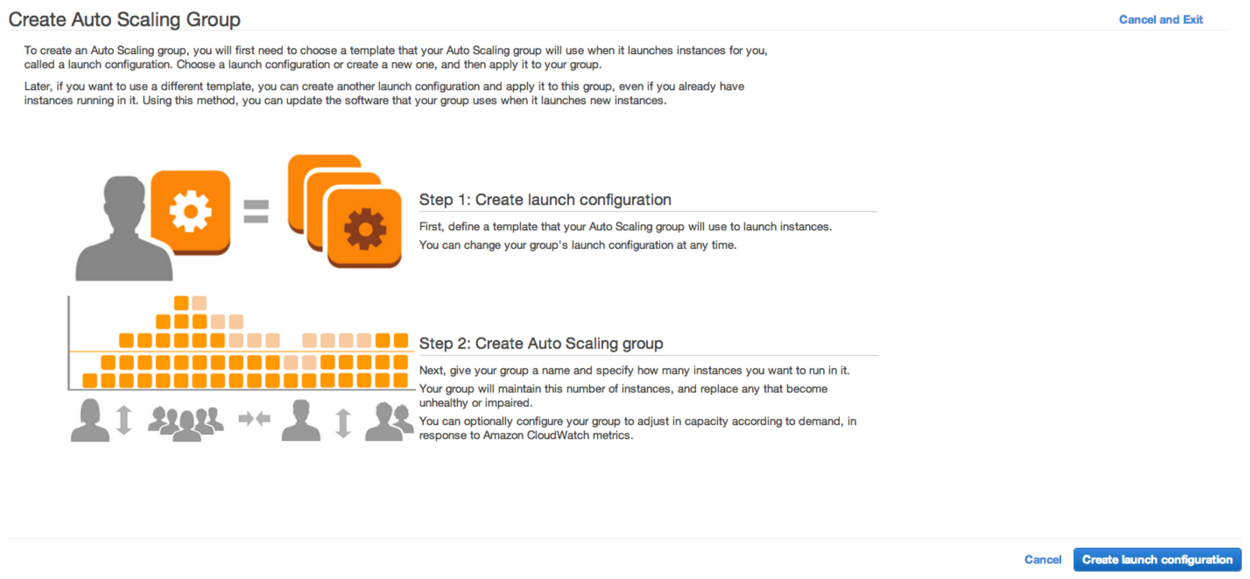
Automated Provisioning

Keep your Auto Scaling group healthy and balanced, whether you need one instance or 1,000.
Learn more

Adjustable Capacity

Maintain a fixed group size or adjust dynamically based on Amazon CloudWatch metrics.
Learn more

The Create Auto Scaling Group button is clicked, which displays the Create Auto Scaling Group screen:


Create Auto Scaling Group

To create an Auto Scaling group, you will first need to choose a template that your Auto Scaling group will use when it launches instances for you, called a launch configuration. Choose a launch configuration or create a new one, and then apply it to your group.

Later, if you want to use a different template, you can create another launch configuration and apply it to this group, even if you already have instances running in it. Using this method, you can update the software that your group uses when it launches new instances.

Step 1: Create launch configuration

First, define a template that your Auto Scaling group will use to launch instances. You can change your group's launch configuration at any time.

Step 2: Create Auto Scaling group

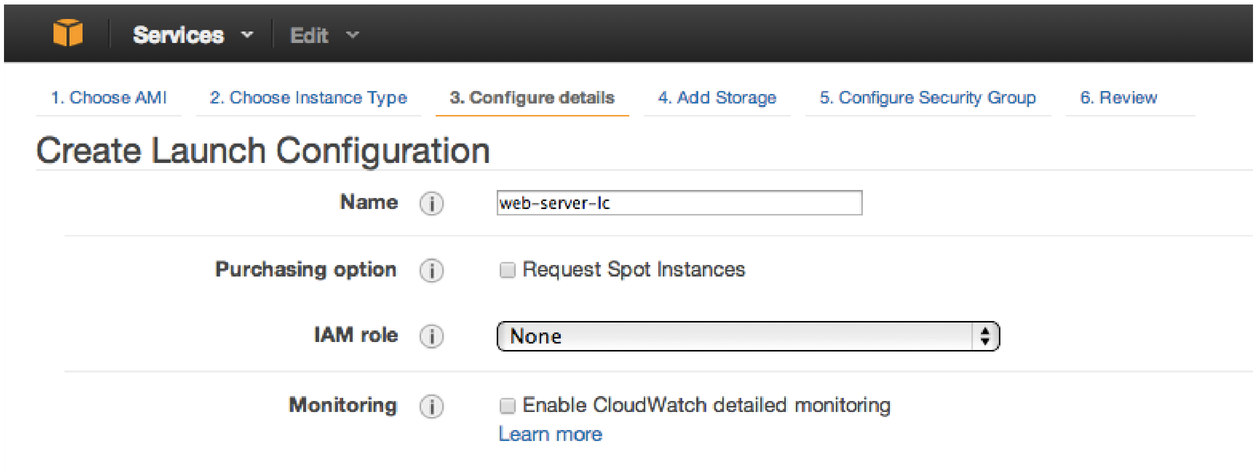
Next, give your group a name and specify how many instances you want to run in it. Your group will maintain this number of instances, and replace any that become unhealthy or impaired.

You can optionally configure your group to adjust in capacity according to demand, in response to Amazon CloudWatch metrics.

The *Create launch configuration* option is clicked. In the next screens, the following choices are made:

- For the AMI, the 32-bit Ubuntu Server 12.04.x LTS is chosen (note that the exact version number changes on a regular basis)
- For the EC2 instance type, a micro instance is chosen

In the Configure Details screen, the LC is named *web-server-lc*:



Create Launch Configuration

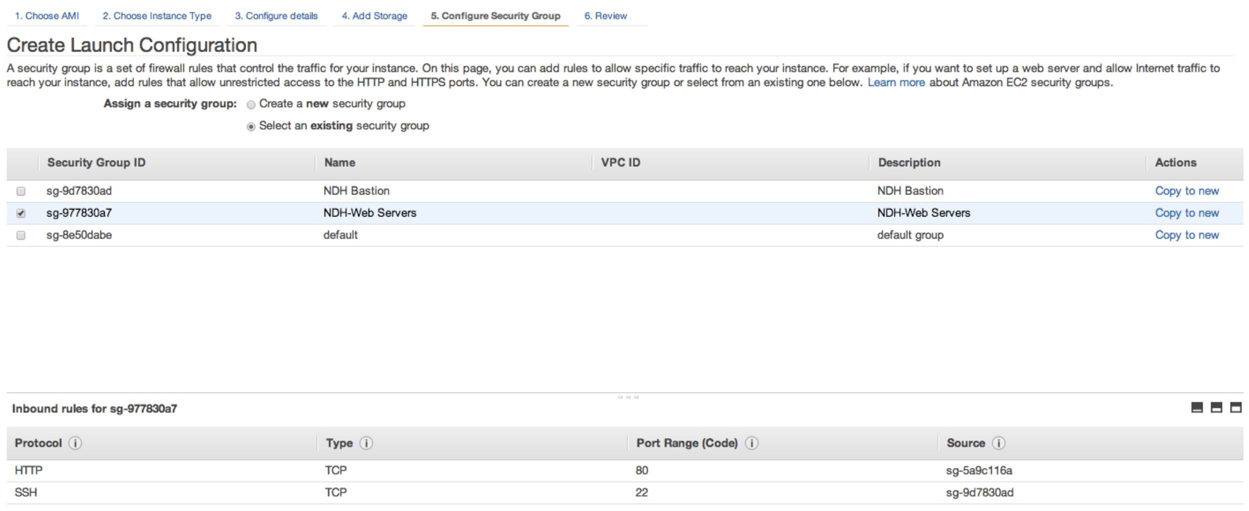
Name: web-server-lc

Purchasing option: Request Spot Instances

IAM role: None

Monitoring: Enable CloudWatch detailed monitoring

The Configure Security Group screen lets you configure who can access the EC2 instances. In this example, access to these instances is limited to EC2 instances that are members of the ELB security group (over port 80, designated by the string *sg-5a9c116a*) or the NDH Bastion group (over port 22, designated by the string *sg-9d7830ad*):



Create Launch Configuration

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

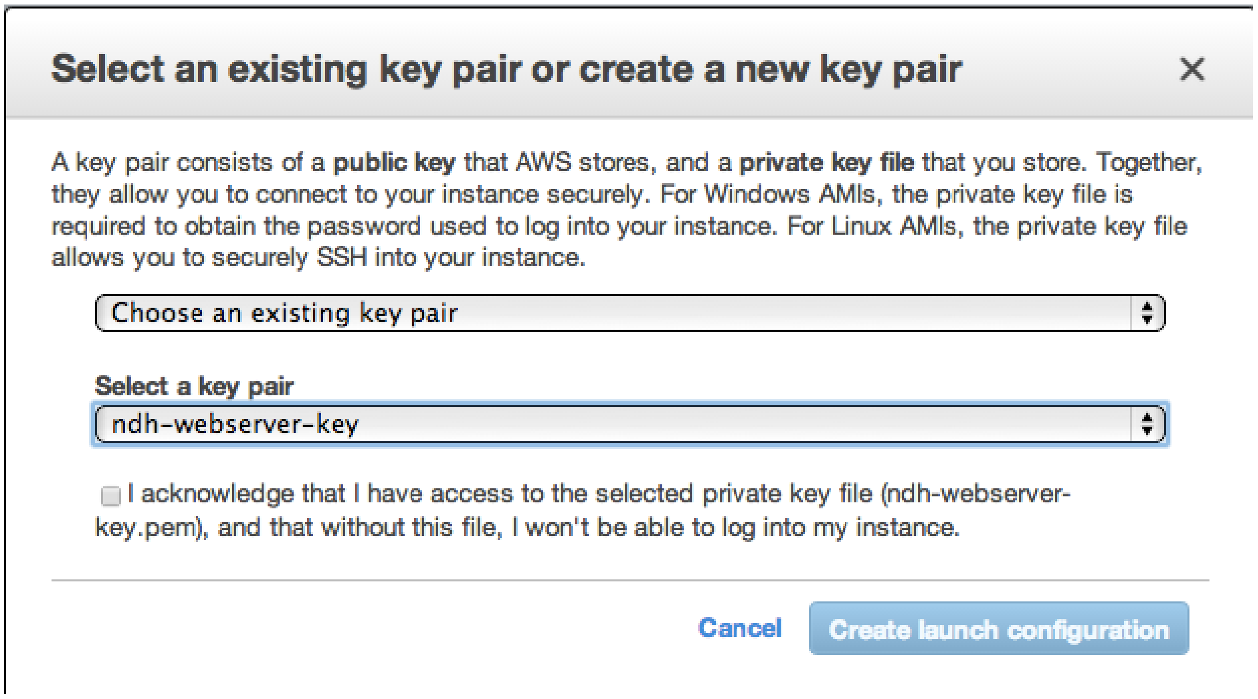
Assign a security group: Create a **new** security group [✓ Select an **existing** security group]

	Security Group ID	Name	VPC ID	Description	Actions
☐	sg-9d7830ad	NDH Bastion		NDH Bastion	Copy to new
✓	sg-977830a7	NDH-Web Servers		NDH-Web Servers	Copy to new
☐	sg-8e50dabe	default		default group	Copy to new

Inbound rules for sg-977830a7

Protocol	Type	Port Range (Code)	Source
HTTP	TCP	80	sg-5a9c116a
SSH	TCP	22	sg-9d7830ad

The Review button is clicked so that the LC's setup can be reviewed, and then the Create Launch Configuration button is clicked. When prompted to provide the name of the key to start the EC2 instances in this LC, the key pair named *ndh-webserver-key* is chosen:



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

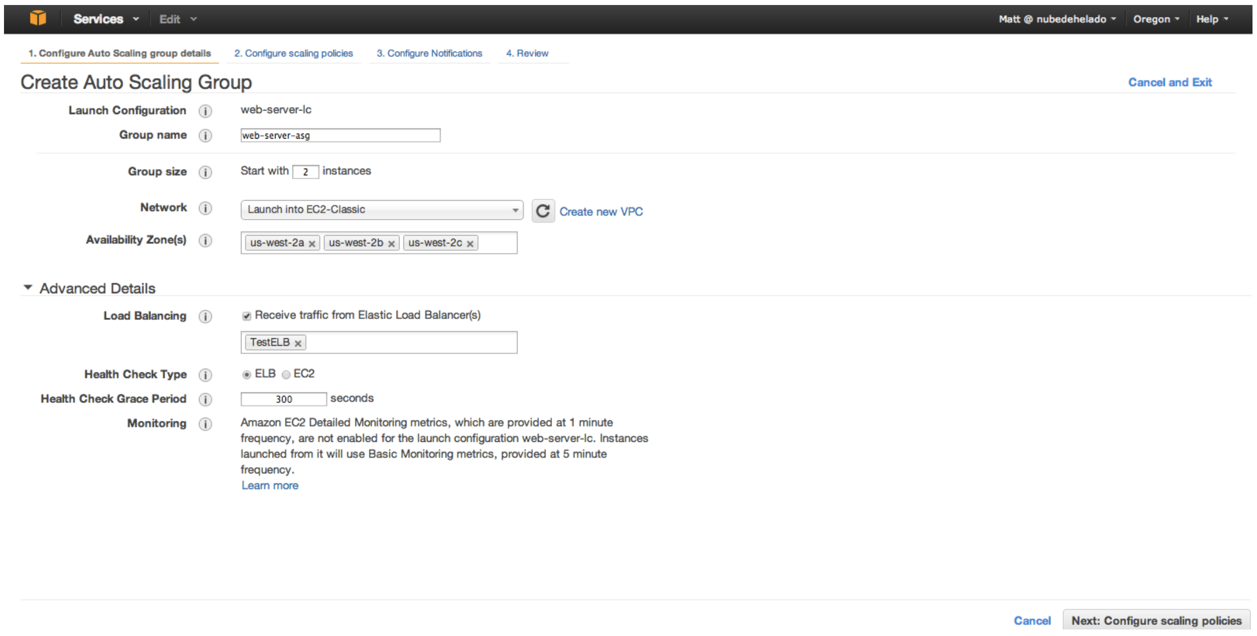
Choose an existing key pair

Select a key pair
ndh-webserver-key

☐ I acknowledge that I have access to the selected private key file (ndh-webserver-key.pem), and that without this file, I won't be able to log into my instance.

Step 2. Create an Auto Scaling Group (ASG) and Auto Scaling Policies (ASPs)

After the LC is created, the Create Auto Scaling Group wizard begins, starting with the Configure Auto Scaling Group Details screen. In this example, the previously created *web-server-lc* is used as the LC; the ASG is named *web-server-asg*; the number of instances to start with is *2*; the AZs to run in are *us-west-2a*, *us-west-2b*, and *us-west-2c*; and the ELB to attach to is *TestELB*.


Services
Edit
Matt @ nubedehelado
Oregon
Help
1. Configure Auto Scaling group details
2. Configure scaling policies
3. Configure Notifications
4. Review
Cancel and Exit
Create new VPC
Cancel
Next: Configure scaling policies

Create Auto Scaling Group

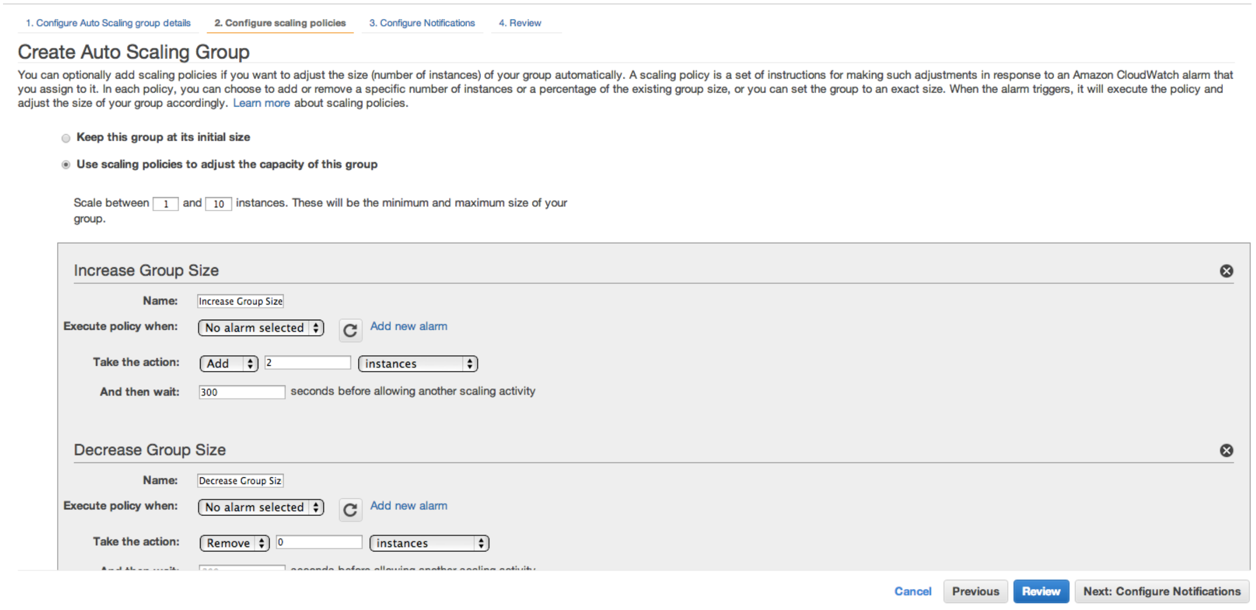
Launch Configuration: web-server-lc
Group name: web-server-asg
Group size: Start with 2 instances
Network: Launch into EC2-Classic
Availability Zone(s): us-west-2a, us-west-2b, us-west-2c

Advanced Details

Load Balancing: [✓] Receive traffic from Elastic Load Balancer(s)
TestELB
Health Check Type: [✓ ELB] EC2
Health Check Grace Period: 300 seconds
Monitoring: Amazon EC2 Detailed Monitoring metrics, which are provided at 1 minute frequency, are not enabled for the launch configuration web-server-lc. Instances launched from it will use Basic Monitoring metrics, provided at 5 minute frequency.
Learn more

Next, the Configure Scaling Policies screen displays. This is where you make your ASG dynamic, by setting policies for increasing or decreasing the number of servers automatically, and by tying those policies to Amazon CloudWatch conditions.

In this example, the *Use scaling policies to adjust the capacity of this group* button is clicked, the minimum number of servers is set to *1*, and the maximum number of servers is set to *10*. In the Increase Group Size section of the screen, the policy is set to increase the number of instances by *2*.



1. Configure Auto Scaling group details | 2. Configure scaling policies | 3. Configure Notifications | 4. Review

Create Auto Scaling Group

You can optionally add scaling policies if you want to adjust the size (number of instances) of your group automatically. A scaling policy is a set of instructions for making such adjustments in response to an Amazon CloudWatch alarm that you assign to it. In each policy, you can choose to add or remove a specific number of instances or a percentage of the existing group size, or you can set the group to an exact size. When the alarm triggers, it will execute the policy and adjust the size of your group accordingly. Learn more about scaling policies.

Keep this group at its initial size [✓ Use scaling policies to adjust the capacity of this group]

Scale between 1 and 10 instances. These will be the minimum and maximum size of your group.

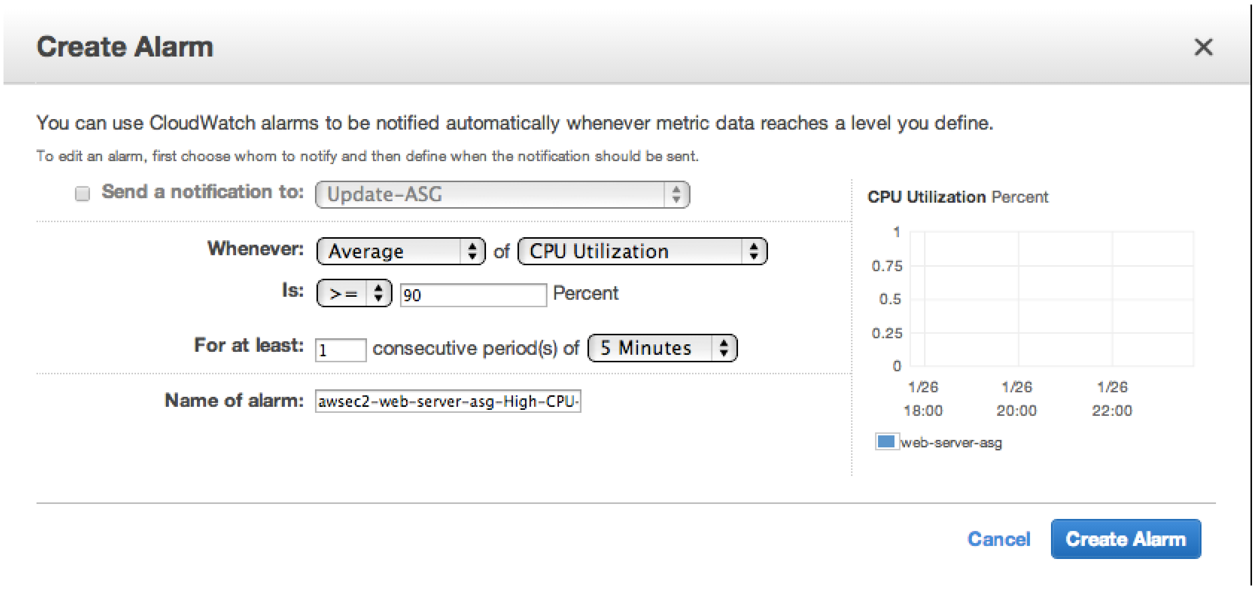
Increase Group Size

Name: Increase Group Size
Execute policy when: No alarm selected | Add new alarm
Take the action: Add 2 instances
And then wait: 300 seconds before allowing another scaling activity

Decrease Group Size

Name: Decrease Group Siz
Execute policy when: No alarm selected | Add new alarm
Take the action: Remove 0 instances

The example continues by electing to add a new alarm to this policy. This alarm triggers if the *Average* of *CPU Utilization* for the ASG is *>=* (greater than or equal to) *90* percent for at least *1* consecutive period of *5 Minutes* (unlike traditional servers, you don't have to leave safety margins for EC2 instances):



Create Alarm

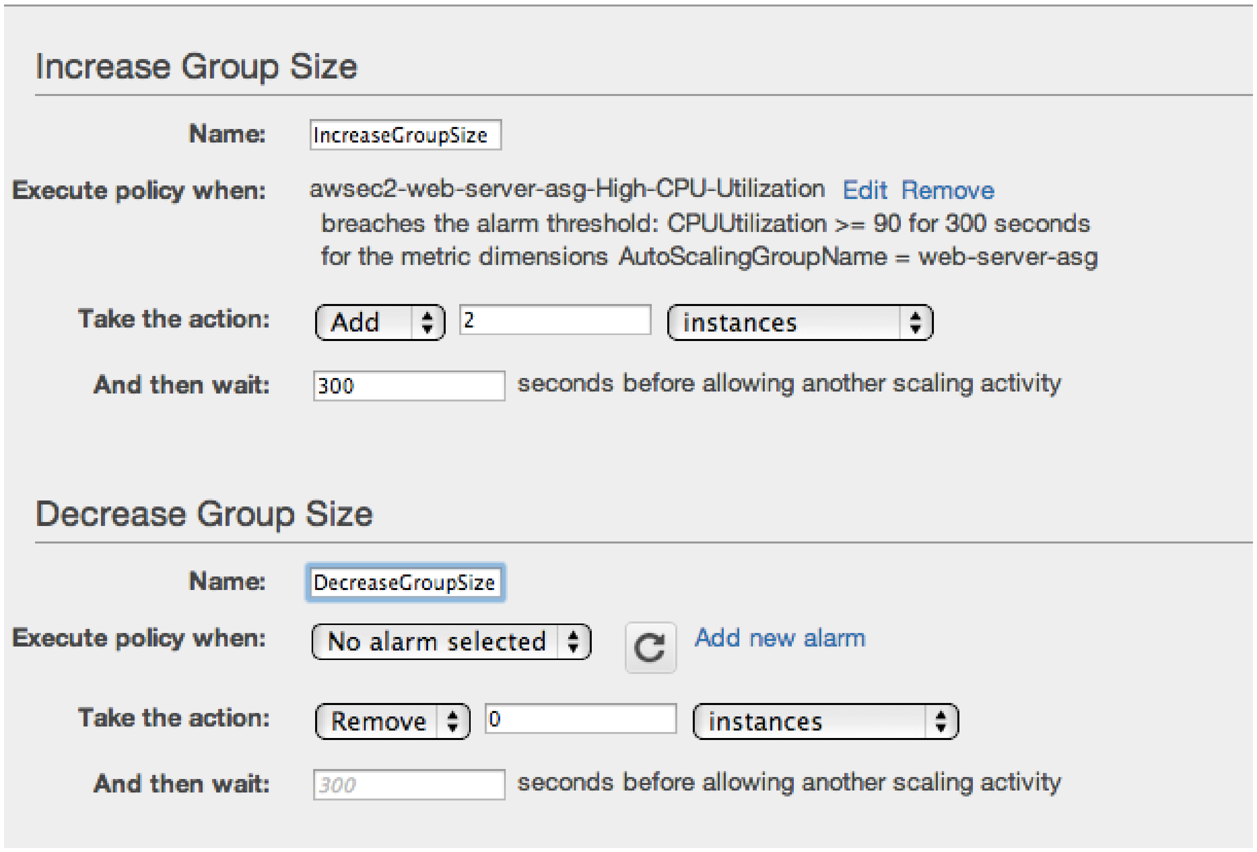
You can use CloudWatch alarms to be notified automatically whenever metric data reaches a level you define.

To edit an alarm, first choose whom to notify and then define when the notification should be sent.

Send a notification to: Update-ASG

Whenever: Average of CPU Utilization
Is: >= 90 Percent
For at least: 1 consecutive period(s) of 5 Minutes
Name of alarm: awsec2-web-server-asg-High-CPU-

Back in the Configure Scaling Policies screen, the Decrease Group Size section of the screen is also used in this example:



Increase Group Size

Name: IncreaseGroupSize

Execute policy when: awsec2-web-server-asg-High-CPU-Utilization
breaches the alarm threshold: CPUUtilization >= 90 for 300 seconds
for the metric dimensions AutoScalingGroupName = web-server-asg

Take the action: Add 2 instances

And then wait: 300 seconds before allowing another scaling activity

Decrease Group Size

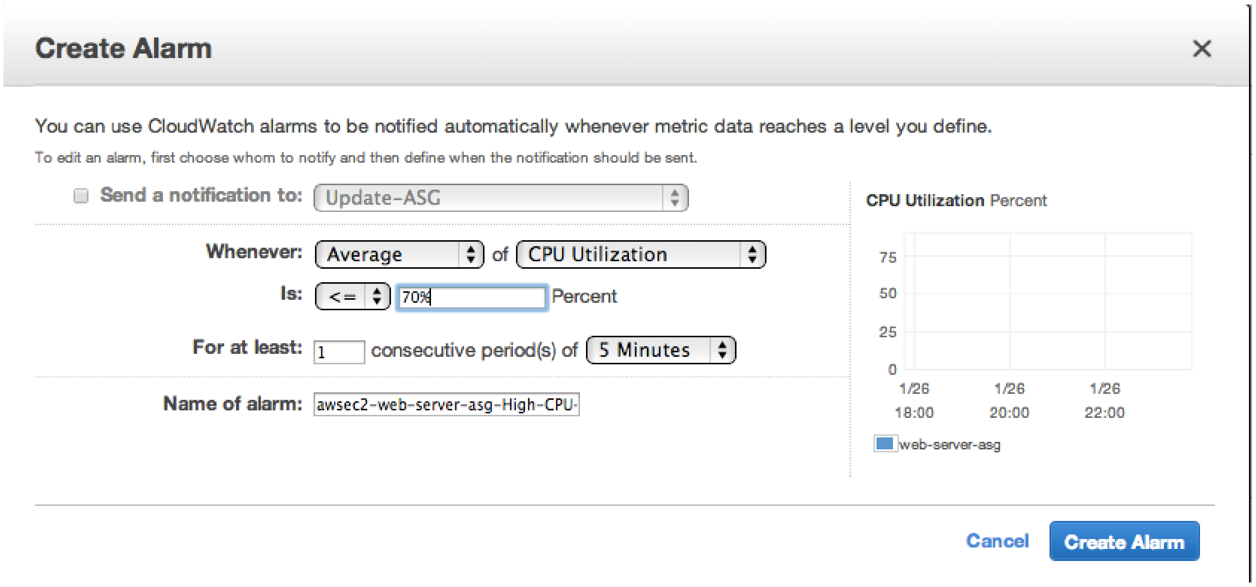
Name: DecreaseGroupSize

Execute policy when: No alarm selected

Take the action: Remove 0 instances

And then wait: 300 seconds before allowing another scaling activity

A new alarm is added to this policy as well. This alarm triggers if the *Average* of *CPU Utilization* for the ASG is *<=* (less than or equal to) *70* percent for at least *1* consecutive period of *5 Minutes*.



Create Alarm

You can use CloudWatch alarms to be notified automatically whenever metric data reaches a level you define.

To edit an alarm, first choose whom to notify and then define when the notification should be sent.

Send a notification to: Update-ASG

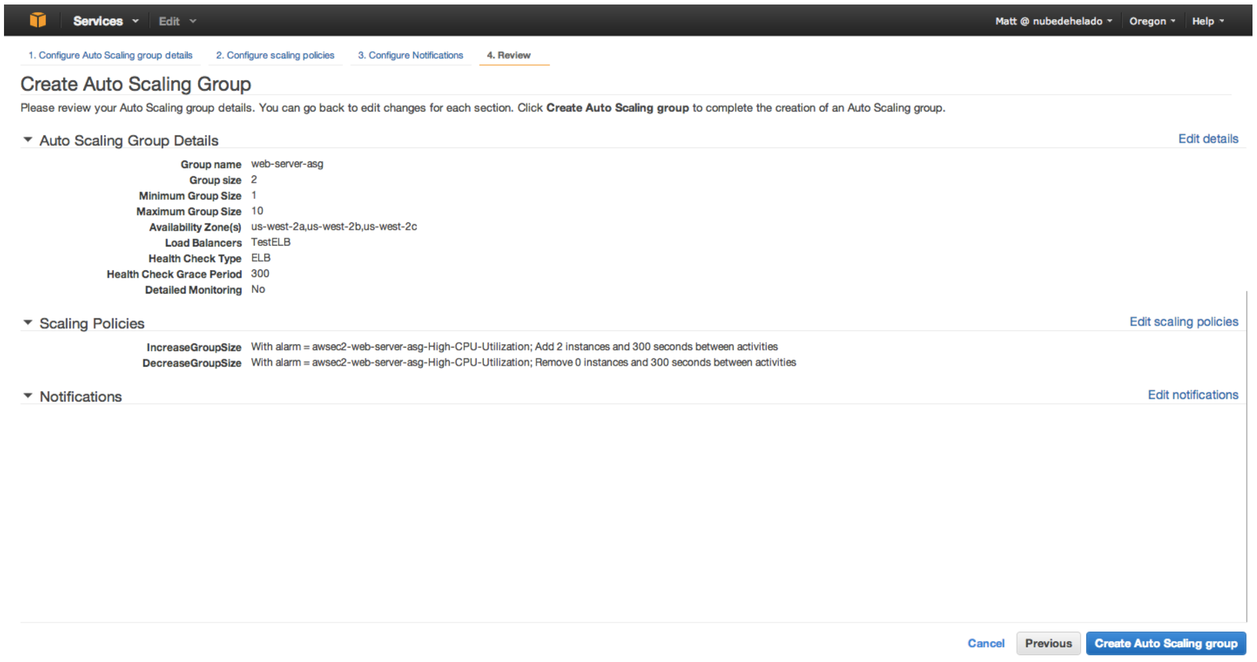
Whenever: Average of CPU Utilization

Is: <= 70% Percent

For at least: 1 consecutive period(s) of 5 Minutes

Name of alarm: awsec2-web-server-asg-High-CPU-

At the end of the process, the Review screen is used to check the ASG before the Create Auto Scaling Group button is clicked, which launches the ASG. The example's Review screen looks like this:


Services
Edit
Matt @ nubedehelado
Oregon
Help
1. Configure Auto Scaling group details
2. Configure scaling policies
3. Configure Notifications
4. Review
Edit details
Edit scaling policies
Edit notifications
Cancel
Previous
Create Auto Scaling group

Create Auto Scaling Group

Please review your Auto Scaling group details. You can go back to edit changes for each section. Click **Create Auto Scaling group** to complete the creation of an Auto Scaling group.

Auto Scaling Group Details

Group name	web-server-asg
Group size	2
Minimum Group Size	1
Maximum Group Size	10
Availability Zone(s)	us-west-2a,us-west-2b,us-west-2c
Load Balancers	TestELB
Health Check Type	ELB
Health Check Grace Period	300
Detailed Monitoring	No

Scaling Policies

IncreaseGroupSize	With alarm = awsec2-web-server-asg-High-CPU-Utilization; Add 2 instances and 300 seconds between activities
DecreaseGroupSize	With alarm = awsec2-web-server-asg-High-CPU-Utilization; Remove 0 instances and 300 seconds between activities

Notifications

When this ASG starts, two instances start up and are distributed among the three AZs that were specified. Then Auto Scaling monitors CPU utilization and increases or decreases the number of servers as necessary, based on the two policies that were created. Additionally, Auto Scaling monitors the running servers: if one dies, or the AZ is lost, Auto Scaling starts a new server until all the lost servers are replaced.

Conclusion

Cloud computing promises to revolutionize computing environments primarily by allowing those environments to meet demand in flexible and inexpensive ways. If it's needed only once in a while, it can be easy to manually spin up more capacity to meet demand, or to manually eliminate unneeded capacity. However, doing so quickly and accurately every time it should occur in your environment can become costly: you may need to hire a staff of operators, or you may need to develop software to manipulate your cloud capacity.

AWS has introduced Auto Scaling so that you can take advantage of cloud computing without having to incur the costs of adding more personnel or building your own software. You can use Auto Scaling to scale for high availability, to meet increasing system demand, or to control costs by eliminating unneeded capacity. You can also use Auto Scaling to quickly deploy software for massive systems, using testable, scriptable processes to minimize risk and cost of deployment.

Learn More

Learn more about how you can improve productivity, enhance efficiency, and sharpen your competitive edge through training.

AWS Essentials
Architecting on AWS
Architecting on AWS - Advanced Concepts
Architecting on AWS - Fast Track

Visit **www.globalknowledge.com** or call **1-800-COURSES (1-800-268-7737)** to speak with a Global Knowledge training advisor.

About the Author

Jon M. Gallagher has decades of experience creating and running large-scale software systems for the web and for enterprises. He has been using Amazon Web Services since its introduction in 2006.