AWS Auto Scaling

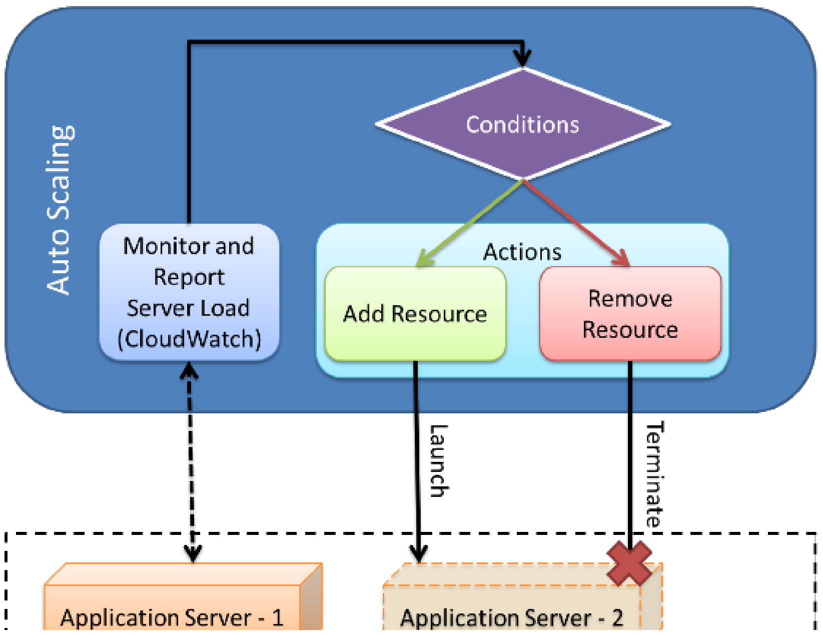
Develop programs that automatically manage cloud resources

Amazon Auto Scaling

Amazon's Auto Scaling service automatically adds or removes computing resources allocated to an application, by responding to changes in demand. In the AWS Auto Scaling context, **scaling** here refers to **horizontal scaling**, which is the act of increasing or decreasing the compute capacity of an application by changing the number of servers assigned to it. This is in contrast to **vertical scaling**, which is achieved by changing the size of servers in response to demand, and is currently not supported by AWS Auto Scaling. Amazon Auto Scaling can be accessed through command-line tools or through APIs in the SDK.

Auto Scaling Introduction

A business case for Auto Scaling is dynamic infrastructure management. For example, a web application that has variable traffic can benefit from Auto Scaling. As traffic increases and the load on the servers increase, the number of servers can be increased dynamically. Similarly, as the traffic decreases and the load on the servers decreases, the number of servers can be decreased dynamically.



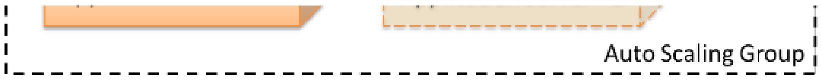


Figure 3.3: Auto Scaling Architecture in Amazon EC2.

Amazon's Auto Scaling provides the following deployment modes on a group of EC2 instances:

- Maintain a fixed number of running EC2 instances at all times, by performing periodic health checks on instances and automatically launching new instances when one or more are unhealthy;
- Manually scale instances by invoking Auto Scaling manually, which will increase or decrease the number of running instances on your behalf;
- Predictable scaling based on a developer-specified schedule (for example, a condition could specify to decrease the servers every Friday at 5 p.m. and to increase them every Monday at 8 a.m);
- Dynamically scale based on conditions specified by the developer (for example, CPU utilization of your EC2 instances).

In this project, we will be focusing on the last deployment mode, i.e. to program Amazon Auto Scaling to dynamically scale the size of the cluster, based on certain conditions. In our case, we will program Auto Scaling to respond to changes in CPU load of provisioned servers.

Working with Auto Scaling

Using Amazon's Auto Scaling in a dynamic fashion requires, at minimum, the following:

- An Auto Scaling **group** should be defined, with minimum, maximum and desired number of instances defined during the group creation. More information can be found here on creating auto scaling groups.
- A **launch configuration** template should be defined, which includes the AMI ID, instance types, key pairs and security group information, among others. As Auto Scale is meant to work automatically to scale based on application demands, the instance should be configured to automatically start the required application services and work seamlessly without user interaction once launched.
- An Auto Scaling **policy** should be created, which defines the set of actions to perform when an event, such as a CloudWatch alarm, is triggered.

The full documentation on auto-scaling is available in the documentation.

Initial Setup

By default, Amazon autoscaling events happen automatically without any notification . For this project, you should setup Amazon's Simple Notification Service (SNS) to automatically send you email regarding Auto Scale events, to keep track of your instances within an Auto Scaling group:

1. In the AWS Management Console, click on **SNS**.
2. Click on **Create New Topic**, specify a name and description when prompted.
3. Once the Topic is created, note down the **Amazon Resource Number (ARN)**.
4. Subscribe to the topic using your email address and confirm the subscription when you receive an email.

Auto Scaling in the AWS Console

AWS provides three methods to launch and configure Auto Scaling: You can use AWS Console, CLI tools or write a program using AWS API. In this part,we will introduce how to launch Auto Scaling by using AWS Console. For CLI tools, you can find detailed installation instructions here.

1. Click “Auto Scaling Group” on the left side of EC2 dashboard and click “Create Auto Scaling Group”
2. Choose a launch configuration or create a new one, and then apply it to your group. This is the template that your Auto Scaling group will use when it launches instances for you.
3. Steps in launch configuration are very similar as launching new EC2 instance. And you need to specify

configuration name, spot request or not.

4. After create a new launch configuration, you need to configure Auto Scaling group. In this step, you will specify the group name, initial instance number. In the advanced options, you can select a load balancer and receive traffic from it
5. Click next to configure auto scaling policies, you can select the group as fixed size or automatically scaling.
6. If you use scaling policies to adjust the capacity of this group, you need to specify the minimum and maximum instance numbers.
7. Next is to configure scaling up and scaling down policies. Choose an Amazon CloudWatch alarm to associate with this policy. The alarm will automatically execute the policy when its threshold is breached.
8. Add notification. You will receive email notifications when scaling group executes scale up or scaling down operations.

Then a new scaling group is created. You can check its status and instances attached to it by selecting the created group and modify policies if needed. Remember to delete the group once you are finished, or it will keep launching instances if you attempt to manually terminate the instances within the ASG.

Writing your own Auto Scaler

In this part of the project, you will write a program that uses the Amazon Auto Scale API in the Autoscale CLI or SDKs to create an **auto scaling group** that expands and shrinks automatically based on CPU utilization. You will use a custom AMI that we have built that creates variable CPU load at periodic intervals.

Note: For this checkpoint, assign the tag with Key: `Project` and Value: `2.4` for all resources

1. Write a program that utilizes the Auto Scaling API and does the following:
    a. Create an ELB with the parameters similar to the ones from the last part of the project. Make sure your program outputs the ELB DNS name and record it for future reference
    b. Create a **Launch Configuration** for the instances that will become part of the auto scaling group, with the following parameters:
        a. AMI ID: `ami-99e2d4f0`
        b. Instance Type: `m1.small`
        c. Detailed Monitoring: `enabled`

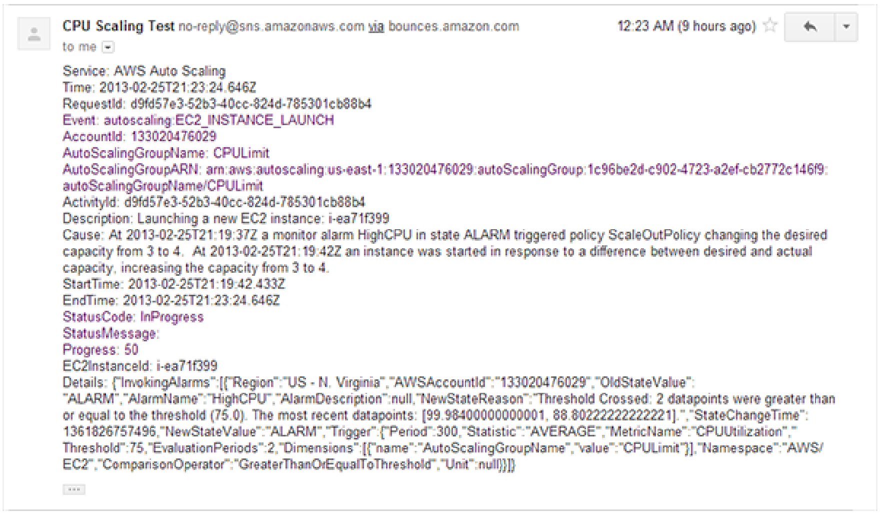
    c. Create an **Auto Scaling Group** with the following Parameters:
        a. Minimum Instance Size: **2**
        b. Maximum Instance Size: **5**
        c. Desired Instance Capacity: **2**
        d. ELB Name: <your ELB name>

    d. Create the following Auto Scale **Policies**:
        - Create a `ScaleOut` policy that automatically adds a single instance to the auto scaling group.
        - Create a `ScaleIn` policy that automatically removes a single instance from the auto scaling group.

    e. Create CloudWatch Alarms that invoke the appropriate policy for the following scenarios:
        - Scale up when the group's CPU load exceeds 80% on average over a 5 minute interval.
        - Scale down when the group's CPU load is below 20% on average over a 5 minute interval.

    f. Configure your Auto Scaling Group to notify you during scale up and scale down events using the ARN you obtained while creating the SNS topic.
    g. Pass all the requests to AWS to launch the auto-scaling group.

2. After the auto scaling group is created, you can monitor your instances and your CloudWatch Alarms in AWS Console. You should receive an email every time there is an activity on your Autoscaling group. (If you don't receive them, check your junk box)



```
CPU Scaling Test no-reply@sns.amazonaws.com via bounces.amazon.com        12:23 AM (9 hours ago)
to me

Service: AWS Auto Scaling
Time: 2013-02-25T21:23:24.646Z
RequestId: d9fd57e3-52b3-40cc-824d-785301cb88b4
Event: autoscaling:EC2_INSTANCE_LAUNCH
AccountId: 133020476029
AutoScalingGroupName: CPULimit
AutoScalingGroupARN: arn:aws:autoscaling:us-east-1:133020476029:autoScalingGroup:1c96be2d-c902-4723-a2ef-cb2772c146f9:
autoScalingGroupName/CPULimit
ActivityId: d9fd57e3-52b3-40cc-824d-785301cb88b4
Description: Launching a new EC2 instance: i-ea71f399
Cause: At 2013-02-25T21:19:37Z a monitor alarm HighCPU in state ALARM triggered policy ScaleOutPolicy changing the desired
capacity from 3 to 4.  At 2013-02-25T21:19:42Z an instance was started in response to a difference between desired and actual
capacity, increasing the capacity from 3 to 4.
StartTime: 2013-02-25T21:19:42.433Z
EndTime: 2013-02-25T21:23:24.646Z
StatusCode: InProgress
StatusMessage:
Progress: 50
EC2InstanceId: i-ea71f399
Details: {"InvokingAlarms":[{"Region":"US - N. Virginia","AWSAccountId":"133020476029","OldStateValue":
"ALARM","AlarmName":"HighCPU","AlarmDescription":null,"NewStateReason":"Threshold Crossed: 2 datapoints were greater than
or equal to the threshold (75.0). The most recent datapoints: [99.98400000000001, 88.8022222222221].","StateChangeTime":
1361826757496,"NewStateValue":"ALARM","Trigger":{"Period":300,"Statistic":"AVERAGE","MetricName":"CPUUtilization","
Threshold":75,"EvaluationPeriods":2,"Dimensions":[{"name":"AutoScalingGroupName","value":"CPULimit"}],"Namespace":"AWS/
EC2","ComparisonOperator":"GreaterThanOrEqualToThreshold","Unit":null}}]}
```

Figure 3.4: Amazon SNS email indicating an Autoscale operation.

3. Once you have setup everything, open a browser to point to your ELB's DNS name and verify that the ELB is connected to your instances and web pages are accessible.
4. When you are ready to run the benchmark, provision an `m1.medium` instance using the ami: `ami-99e2d4f0` to act as the launching instance. Run the program `cd benchmark; sudo ./seige_bench.sh <your ELB DNS>` to start the benchmark. The benchmark should take about 45 minutes to complete.
5. Monitor the EC2 dashboard as well as your inbox for incoming messages regarding state changes with your AutoScaling group. Keep the emails safe for the checkpoint below.
6. Once you have finished verifying that your autoscale program works correctly, delete your auto scaling group in AWS Console. You must delete your auto scaling group, and not just terminate the instances launched by it, as they will keep re-launching more EC2 instances automatically.

   Note: Do NOT attempt to manually terminate instances from the AWS Management Console. The AutoScale group will assume a fault and relaunch instances.
7. Copy your source code (as a plain text file) onto your Amazon S3 submision bucket that is accessible to the instructors.

Once you have completed all the steps above, proceed to the checkpoint quiz below:

checkpoint

Auto Scaling