Assignment #08

Due Date

Section	Due Date	Grading Deadline
Saturday & Online	09:00pm on 08/10/2019	09:00pm on 08/15/2019

Getting Help

Info

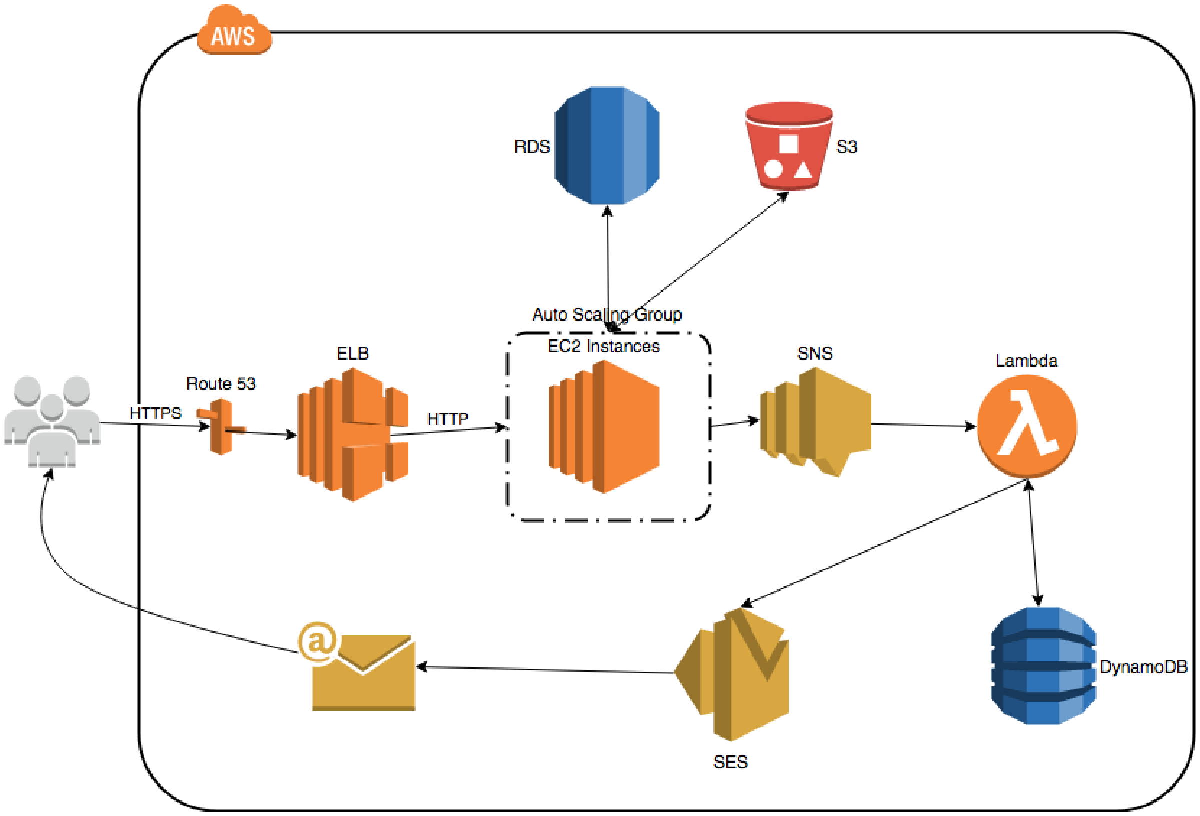
Ask all your questions on Piazza. Assign **assignment8** tag to your posts.

Assignment Weightage

Individual Assignment Weightage on Course Grade of this assignment is **10%**.

Objectives

This assignments builds on top of all previous assignments. Our objective in this assignment is put the final pieces together for a higly available, load balanced and secure infrastructure. This will require you to modify the application CloudFormation template that you have been working with so far. No changes should be needed for the web application itself. You may make changes to CI/CD template as needed.



IAM Users, Roles & Policies Updates

- Add IAM roles & policies needed to meet the assignment objectives to the appropriate CloudFormation stack.

Security Group Updates

Tip

For testing & debugging, you may manually modify security groups to allow direct access to the instances.

- Modify web application security group ingress so that the source is load balancer for the web application port. Use the port you application is accessible on.

- Make sure that the database security group now only allows traffic from web application security group. Your database should not be accessible from anywhere else.

The Auto Scaling Application Stack

So far our web application has been accessible by IP address in plain text (HTTP). We will now disable direct access to our web application. The web application will now only be accessible from load balancer.

1. Create a new application stack called `csye6225-cf-auto-scaling-application.json` by cloning your current application stack `csye6225-cf-application.json`. Remove the EC2 resource from CloudFormation application and replace it with auto-scaling group. Clone `csye6225-aws-cf-create-application-stack.sh` and `csye6225-aws-cf-terminate-application-stack.sh` scripts to create `csye6225-aws-cf-create-auto-scaling-application-stack.sh` and `csye6225-aws-cf-terminate-auto-scaling-application-stack.sh`. Requirements for script are same as those of other CloudFormation stacks we have built.
2. You will now use `csye6225-cf-auto-scaling-application.json` to setup your application stack instead of `csye6225-cf-application.json`.

Setup Autoscaling for EC2 Instances

Instead of launching EC2 instances standalone, we are now going to launch them in auto-scaling group with minimum of **3** instances and maximum of **7**. Use default or value you prefer for properties not listed below.

Launch Configuration

Key	Value
`ImageId`	Your custom AMI
`Instance Type`	t2.micro
`KeyName`	*YOUR_AWS_KEYNAME*
`AssociatePublicIpAddress`	True
`UserData`	*SAME_USER_DATA_AS_CURRENT_EC2_INSTANCE*
`IAM Role`	*SAME_AS_CURRENT_EC2_INSTANCE*
`Resource Name`	asg_launch_config
`Security Groups`	Updated web security group.

Auto Scaling Group

Note

You need to tag [https://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/ASTagging.html] (AutoScalingGroup TagProperty [https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-as-tags.html]) EC2 instances in your Auto Scaling Group so that CodeDeploy will find them and update them when new code deployment is triggered.

Parameter	Value
`Cooldown`	60
`LaunchConfigurationName`	asg_launch_config
`MinSize`	3
`MaxSize`	7
`DesiredCapacity`	3

AutoScaling Policies

Create following policies:

1. Scale `up` policy when `average CPU usage` is above `5%` . Increment by `1` .
2. Scale `down` policy when `average CPU usage` is below `3%` . Decrement by `1` .

SAMPLE AUTOSCALING POLICY

```
"WebServerScaleUpPolicy": {
  "Type": "AWS::AutoScaling::ScalingPolicy",
  "Properties": {
    "AdjustmentType": "ChangeInCapacity",
    "AutoScalingGroupName": {
      "Ref": "WebServerGroup"
    },
    "Cooldown": "60",
    "ScalingAdjustment": "1"
  }
},
"WebServerScaleDownPolicy": {
  "Type": "AWS::AutoScaling::ScalingPolicy",
  "Properties": {
    "AdjustmentType": "ChangeInCapacity",
    "AutoScalingGroupName": {
      "Ref": "WebServerGroup"
    },
    "Cooldown": "60",
      "ScalingAdjustment": "-1"
    }
  },
  "CPUAlarmHigh": {
    "Type": "AWS::CloudWatch::Alarm",
    "Properties": {
      "AlarmDescription": "Scale-up if CPU > 90% for 10 minutes",
      "MetricName": "CPUUtilization",
      "Namespace": "AWS/EC2",
      "Statistic": "Average",
      "Period": "300",
      "EvaluationPeriods": "2",
      "Threshold": "90",
      "AlarmActions": [
        {
          "Ref": "WebServerScaleUpPolicy"
        }
      ],
      "Dimensions": [
        {
          "Name": "AutoScalingGroupName",
          "Value": {
            "Ref": "WebServerGroup"
          }
        }
      ],
      "ComparisonOperator": "GreaterThanThreshold"
    }
  },
  "CPUAlarmLow": {
    "Type": "AWS::CloudWatch::Alarm",
    "Properties": {
      "AlarmDescription": "Scale-down if CPU < 70% for 10
minutes",
      "MetricName": "CPUUtilization",
      "Namespace": "AWS/EC2",
      "Statistic": "Average",
      "Period": "300",
      "EvaluationPeriods": "2",
      "Threshold": "70",
      "AlarmActions": [
        {
          "Ref": "WebServerScaleDownPolicy"
        }
      ],
      "Dimensions": [
        {
          "Name": "AutoScalingGroupName",
          "Value": {
            "Ref": "WebServerGroup"
          }
        }
      ],
      "ComparisonOperator": "LessThanThreshold"
    }
  }
```

Setup Application Load Balancer For Your Web Application

- EC2 instances launced in the auto-scaling group should now be load balanced.
- Add load balancing resource to CloudFormation template.
- Route53 resource record for your domain name should now be an alias for your load balancer application.
- Update CodeDeploy so that code changes can be deployed to all instances.
- Setup Application load balancer to accept HTTPS traffic and forward it to your application instances.
- Loadbalancer should not respond to any HTTP requests.

Securing Infrastructure with SSL Certificates

- Configure your load balancer to use SSL (Secure Sockets Layer) certificate to protect your web application. Update CloudFormation template as required.
- You can get SSL certificates using AWS Certificate Manager [https://aws.amazon.com/certificate-manager/] which might be easiest way to get the certificates and set them up. See https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-certificatemanager-certificate.html [https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-certificatemanager-certificate.html].

- Namecheap [https://www.namecheap.com/security/ssl-certificates.aspx] offers one year SSL certificate for free with Github Student Developer pack [https://education.github.com/pack].
- You can also get SSL certificates for free from Let's Encrypt [https://letsencrypt.org/].

DNS Update

Your CloudFormation template should configure Route53 so that your domain points to your load balancer and your web application is accessible thru **https://your-domain-name.tld/ [https://your-domain-name.tld/]**.

Web Application Firewall

- Deploy AWS WAF [https://aws.amazon.com/waf/] to the Application Load Balancer (ALB) that fronts your web servers running on EC2.
- Use AWS WAF to Mitigate OWASP's Top 10 Web Application Vulnerabilities [https://aws.amazon.com/about-aws/whats-new/2017/07/use-aws-waf-to-mitigate-owasps-top-10-web-application-vulnerabilities/].
- All WAF resources and web security rules should be added to your application cloudformation stack.

Penetration Testing

- Identify and test your application against at least `2` attact vectors that **do not exploit UI** vulnerabilities.
- You will document your findings in a PDF (Google doc exported as PDF) and commit it to your Github repository.
- Your report should be as detailed as possible.
- **You will document attacks on your own web application with and without the AWS WAF in place.**

Your report should provide details on following along with screenshots:

1. Attack Vector
2. Result
3. Why did you choose this specific attack vector?

Create JMeter Load Testing Script

> **Tip**
>
> Create a static file with usernames and passwords and use that as input for all JMeter run. Clear your database before each run so you can create same users again.

Using Apache JMeter [http://jmeter.apache.org] create tests that can be run against your application APIs. Your scripts should focus on testing following:

1. `1000` concurrent users creating new accounts in your application. See JMeter [http://jmeter.apache.org/usermanual/functions.html#__RandomString] documentation for details on auto-generating values for form.
2. `1000` concurrent authenticated user creating new transactions and attaching attachments to these transactions. You can use the users created in first step.

Documentation

- Sample Auto Scaling CloudFormation Template [https://s3-us-east-2.amazonaws.com/cloudformation-templates-us-east-2/AutoScalingMultiAZWithNotifications.template]
- Auto Scaling AutoScalingGroup TagProperty [https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-as-tags.html]
- AWS::Route53::RecordSet CloudFormation [https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-route53-recordset.html]

- Load Testing Web Servers with Siege [https://linode.com/docs/tools-reference/tools/load-testing-with-siege/]
- How To Use Apache JMeter To Perform Load Testing on a Web Server [https://www.digitalocean.com/community/tutorials/how-to-use-apache-jmeter-to-perform-load-testing-on-a-web-server]
- Fill Forms and Submit With JMeter [https://dzone.com/articles/fill-forms-and-submit-with-jmeter-made-easy]
- Using CSV DATA SET with JMeter [https://guide.blazemeter.com/hc/en-us/articles/206733689-Using-CSV-DATA-SET-CONFIG]

Penetration Testing Tools

- Kali Linux [https://www.kali.org/]
- Kali Linux Penetration Testing Tools [http://tools.kali.org/]
- Automatic SQL injection and database takeover tool [https://github.com/sqlmapproject/sqlmap]
- XSSer [http://tools.kali.org/web-applications/xsser]
- Grabber [http://rgaucher.info/beta/grabber/]
- Wapiti [http://wapiti.sourceforge.net/]
- W3af [http://w3af.org/]
- Wireshark [https://www.wireshark.org/]

Submission

> **Danger**
>
> Assignment will be considered late if commits are made to `master` and `feature` branch after due date.

1. All work for this assignment must be done on **assignment8** feature branch and merged to master when you are dev complete.
2. All team member’s feature and master branches must be in-sync.

Grading Guidelines

⚠ **Warning**

Following guidelines are for information only. They are subject to change at the discretion of the instructor and TA.

- TBD