Lecture #08

Agenda

* Review Assignment #05
* Load Balancing
* High Availability
  + Availability Zones
  + Auto Scaling
* Discuss Assignment #06

Demo

* Load Balancers
* Auto Scaling

Reading

JMeter

* [JMeter User Manual](https://jmeter.apache.org/usermanual/index.html)
* [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)

Auto Scaling & Load Balancing

* [AWS AutoScaling](https://aws.amazon.com/autoscaling/)
* [Auto Scaling in the Amazon Cloud - Netflix](http://techblog.netflix.com/2012/01/auto-scaling-in-amazon-cloud.html)
* [AWS Elastic Load Balancing](https://aws.amazon.com/elasticloadbalancing/)
* Classic Load Balancer
* [Application Load Balancer](https://aws.amazon.com/elasticloadbalancing/applicationloadbalancer/)
* [Netflix Software Based Load Balancer (Zuul)](https://github.com/Netflix/zuul)
* <https://f5.com/glossary/load-balancer>

Infrastructure at Scale

* [How Twitter Handles 3,000 Images Per Second](http://highscalability.com/blog/2016/4/20/how-twitter-handles-3000-images-per-second.html)
* [The Infrastructure Behind Twitter: Scale](https://blog.twitter.com/2017/the-infrastructure-behind-twitter-scale)
* [Caching at Reddit](https://redditblog.com/2017/1/17/caching-at-reddit/)
* [How Uber Manages A Million Writes Per Second Using Mesos And Cassandra Across Multiple Datacenters](http://highscalability.com/blog/2016/9/28/how-uber-manages-a-million-writes-per-second-using-mesos-and.html)
* [Instant Messaging at LinkedIn: Scaling to Hundreds of Thousands of Persistent Connections on One Machine](https://engineering.linkedin.com/blog/2016/10/instant-messaging-at-linkedin--scaling-to-hundreds-of-thousands-)
* [Big Data in Real-Time at Twitter](https://www.slideshare.net/nkallen/q-con-3770885)
* [https://red](https://redditblog.com/2017/04/13/how-we-built-rplace/)
* [ditblog.com/2017/04/13/how-we-built-rplace/](https://redditblog.com/2017/04/13/how-we-built-rplace/)

High Availability & Fault Tolerance Testing

* [Netflix Chaos Monkey](https://github.com/Netflix/chaosmonkey)
* [Netflix Simian Army](https://github.com/Netflix/SimianArmy/wiki)

CloudFormation Template from DEMO

AWSTemplateFormatVersion: "2010-09-09"  
Description: CSYE 6225 Auto Scaling Demo  
Resources:  
  asgLaunchTemplate:  
    Type: AWS::EC2::LaunchTemplate  
    Properties:  
      LaunchTemplateName: !Sub '${AWS::StackName}-launch-template'  
      LaunchTemplateData:  
        BlockDeviceMappings:  
        - DeviceName: /dev/sda1  
          Ebs:  
            VolumeSize: 20  
            VolumeType: "gp3"  
            DeleteOnTermination: true  
        ImageId: ami-07e89c735e8b886d7  
        InstanceType: t3.micro  
        KeyName: csye6225  
        SecurityGroupIds:  
          - sg-0efe1159f9ca25998  
  
  asgWebApp:  
    Type: AWS::AutoScaling::AutoScalingGroup  
    Properties:  
      AutoScalingGroupName: !Sub ${AWS::StackName}-asg  
      AvailabilityZones:  
        - us-east-1a  
        - us-east-1b  
        - us-east-1c  
      Cooldown: '300'  
      DesiredCapacity: '1'  
      HealthCheckGracePeriod: 90  
      HealthCheckType: EC2  
      LaunchTemplate:  
        LaunchTemplateId: !Ref asgLaunchTemplate  
        Version: !GetAtt asgLaunchTemplate.LatestVersionNumber  
      # LoadBalancerNames:  
      #   - String  
      MaxSize: '3'  
      MinSize: '1'  
      NewInstancesProtectedFromScaleIn: true  
      Tags:  
        - Key: Application  
          Value: WebApp  
          PropagateAtLaunch: "true"  
        - Key: Name  
          Value: !Sub ${AWS::StackName}-asg  
          PropagateAtLaunch: "true"  
      TargetGroupARNs:  
        - !Ref loadBalancerTargetGroup  
      VPCZoneIdentifier:  
        - subnet-0fa5ba21  
        - subnet-1730905a  
        - subnet-9c928fc0  
  
  asgCPUPolicy:  
    Type: AWS::AutoScaling::ScalingPolicy  
    Properties:  
      AutoScalingGroupName: !Ref asgWebApp  
      PolicyType: TargetTrackingScaling  
      TargetTrackingConfiguration:  
        PredefinedMetricSpecification:  
          PredefinedMetricType: ASGAverageCPUUtilization  
        TargetValue: 90.0  
  
  loadBalancer:  
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer  
    Properties:  
      IpAddressType: ipv4  
      LoadBalancerAttributes:  
        - Key: idle\_timeout.timeout\_seconds  
          Value: '60'  
      Name: !Sub ${AWS::StackName}-load-balancer  
      Scheme: internet-facing  
      SecurityGroups:  
        - sg-0efe1159f9ca25998  
      Subnets:  
        - subnet-0fa5ba21  
        - subnet-1730905a  
        - subnet-9c928fc0  
      Tags:  
        - Key: Application  
          Value: WebApp  
        - Key: Name  
          Value: !Sub ${AWS::StackName}-load-balancer  
      Type: application  
  
  loadBalancerTargetGroup:  
    Type: AWS::ElasticLoadBalancingV2::TargetGroup  
    Properties:  
      HealthCheckIntervalSeconds: 90  
      HealthCheckPath: '/'  
      HealthCheckPort: '8080'  
      HealthCheckProtocol: 'HTTP'  
      HealthCheckTimeoutSeconds: 60  
      HealthyThresholdCount: 2  
      IpAddressType: ipv4  
      Name: !Sub ${AWS::StackName}-lb-tg  
      Port: 8080  
      Protocol: 'HTTP'  
      Tags:  
        - Key: Name  
          Value: !Sub ${AWS::StackName}-lb-tg  
      TargetGroupAttributes:  
        - Key: deregistration\_delay.timeout\_seconds  
          Value: '20'  
      TargetType: instance  
      UnhealthyThresholdCount: 2  
      VpcId: vpc-3c250946  
  
  loadBalancerListener:  
    Type: AWS::ElasticLoadBalancingV2::Listener  
    Properties:  
      DefaultActions:  
        - Type: forward  
          TargetGroupArn: !Ref loadBalancerTargetGroup  
      LoadBalancerArn: !Ref loadBalancer  
      Port: 80  
      Protocol: HTTP