15-319 / 15-619
Cloud Computing

Recitation 4

February 2nd, 2016

Administrative Issues

- Make use of office hours
  - We will have to make sure that you have tried yourself before you ask
- Monitor AWS expenses regularly
- Always do the cost calculation before launching services
- Terminate your instances when not in use
- Stopping instances still has an EBS cost ($0.1/GB-Month)
- Make sure spot instances are tagged right after launch

Important Notice

- **DON’T EVER EXPOSE YOUR AWS CREDENTIALS!**
  - **Github**
  - **Bitbucket**
  - **Anywhere public…**
- **DON’T EVER EXPOSE YOUR AWS CREDENTIALS!**
- **DON’T EVER EXPOSE YOUR Azure CREDENTIALS!**
  - **ApplicationId, ApplicationKey**
  - **StorageAccountKey, EndpointUrl**

Reflection

- Last week’s reflection
  - Project 1.2, Quiz 2
- Theme - **Big data analytics**
  - P1.1: Sequential Analysis of **418MB** wikipedia data
  - P1.2: Parallel Analysis of **64GB** compressed (**300 GBs** uncompressed) wikipedia data
- Power of parallel analysis
  - Amount of work done remains the same
  - Span is reduced significantly

Reflection

- You should have learned
  - How to process big data sets with MapReduce
    - How MapReduce works
    - How to write a Mapper and a Reducer
    - Performance/cost tradeoff
    - How to debug MapReduce
  - How to save overall cost by testing using small data sets
- Don’t forget about MapReduce just yet!
  - Will be relevant in 15619Project and Project 4

This Week

- **Quiz 3 (OLI Modules 5 & 6)**
  - Due on **Friday**, Feb 5th, 2016, 11:59PM ET

- **Project 2.1**
  - Due on **Sunday**, Feb 7th, 2016, 11:59PM ET

OLI Module 5 - Cloud Management

Cloud Software stack - enables provisioning, monitoring and metering of virtual user “resources” on top of the Cloud Service Provider’s (CSP) infrastructure.

- Cloud middleware
- Provisioning
- Metering
- Orchestration and automation
- Case Study: Openstack - Open-source cloud stack implementation

OLI Module 6 - Cloud Software Deployment Considerations

- Programming the cloud
- Deploying applications on the cloud
- Build fault-tolerant cloud services
- Load balancing
- Scaling resources
- Dealing with tail latency
- Economics for cloud applications

Project 2 Overview

MSB Interview

- **2.1 Scale, Scale, Scale**
  - AWS and Azure APIs
  - Horizontal scaling in/out, load balancing, failure detection, and cost management
- **2.2 Explore load balancing**
  - Distribute load evenly among your servers
- **2.3 Explore caching strategies**
  - Speedup your server through different cache strategies

Project 2.1 Learning Objectives

- Use cloud APIs to programmatically control resources in response to demand.
- Be able to deploy web services that account for failure, cost and performance constraints.
- Compare and contrast the APIs used in AWS and Azure.
- Be able to identify and explain the need for handling resource failures.
- Be able to explain the need for distributing load evenly among all resources.
- Configure and deploy a Load Balancer along with an VM Scale Set on Azure.
- Configure and deploy an Elastic Load Balancer along with an Auto Scaling Group on AWS.
- Develop solutions that manage cloud resources with the ability to deal with resource failure.
- Account for cost as a constraint when provisioning cloud resources.
- Analyze the trade off between maximizing performance & reliability and the effect on cost.

Quality of Service (QoS)

Quantitatively Measure QoS

- Performance: Throughput, Latency
  (Very helpful in Projects 2 & 15619Project)
- Availability: the probability that a system is operational at a given time *(Projects P2.1 and P2.2)*
- Reliability: the probability that a system will produce a correct output up to a given time *(Project P2.1)*

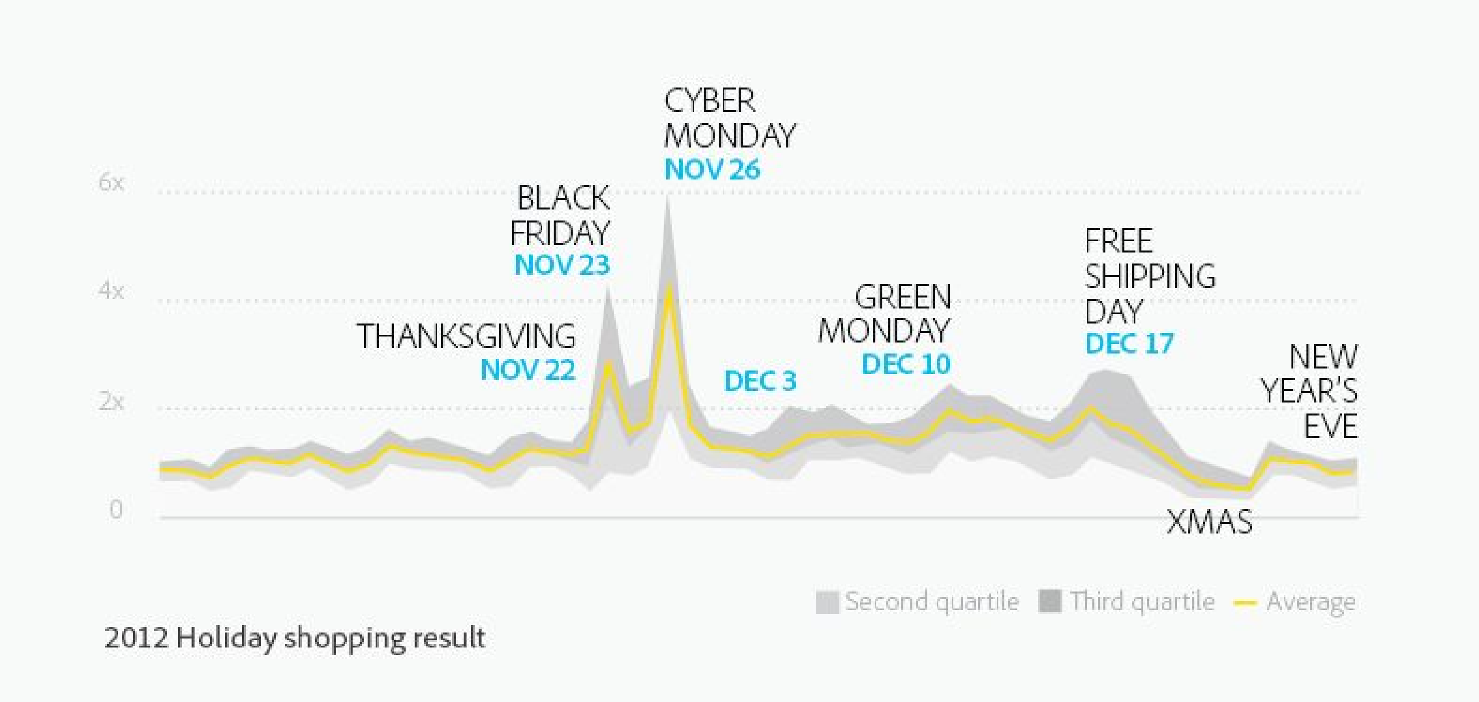
QoS Matters:

- Amazon found every 100ms of latency cost them 1% in sales.

FACT

Reality, human patterns...

- Daily
- Weekly
- Monthly
- Yearly
- ...

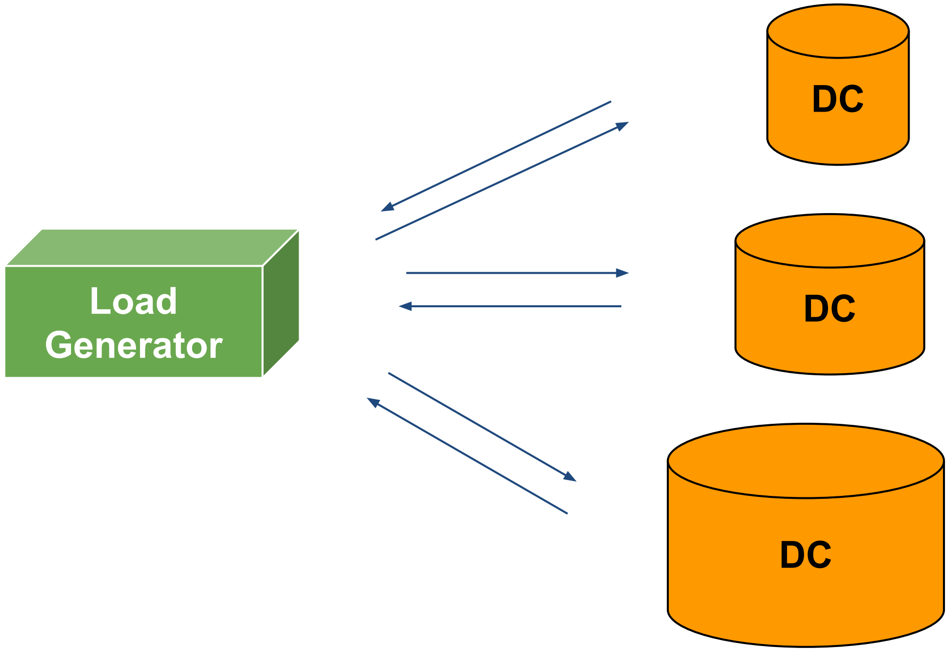


2012 Holiday shopping result

sapient.com

Cloud Comes to the Rescue!
Scaling!

P0: Vertical Scaling

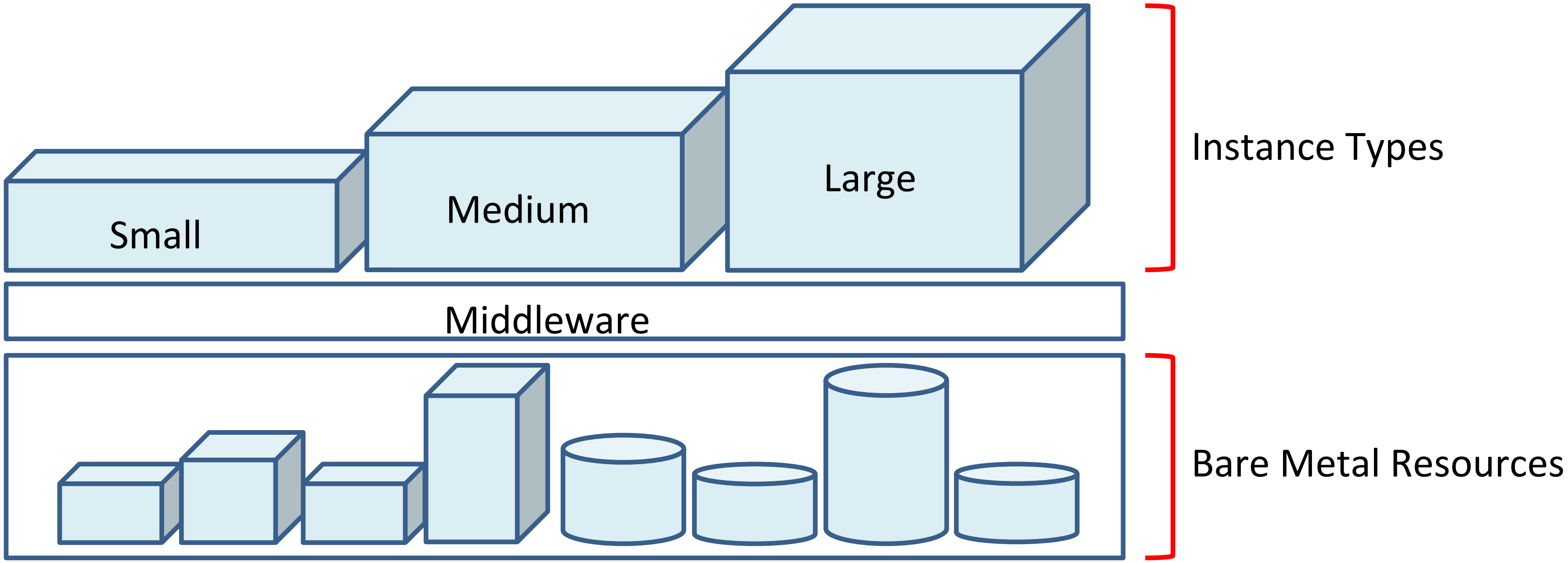


Small Data Center Instance
Serves 100 requests per second

Medium Data Center Instance
Serves 1000 requests per second

Large Data Center Instance
Serves 10000 requests per second

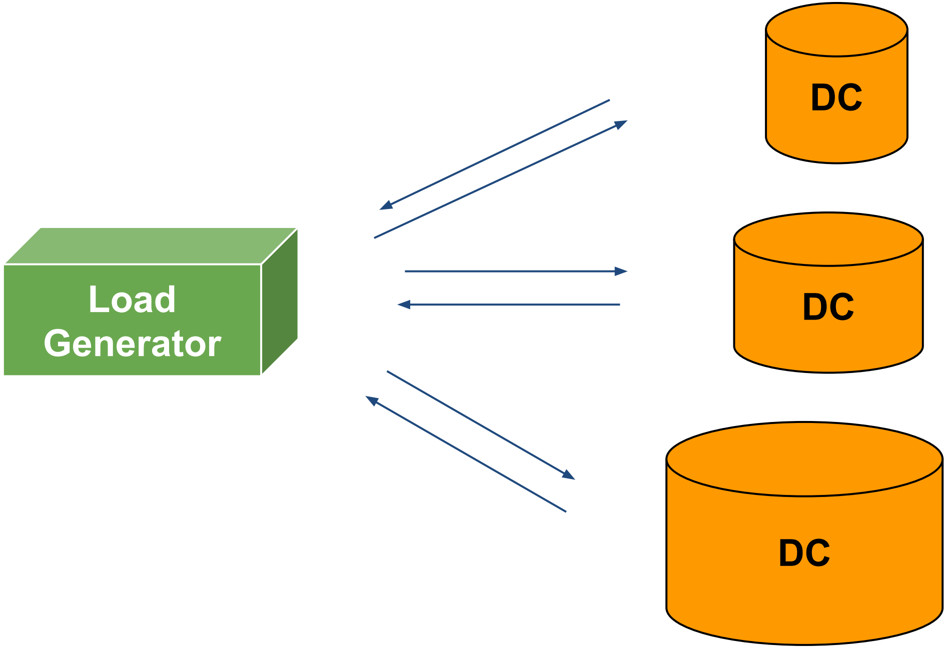
Resources in Cloud Infrastructure



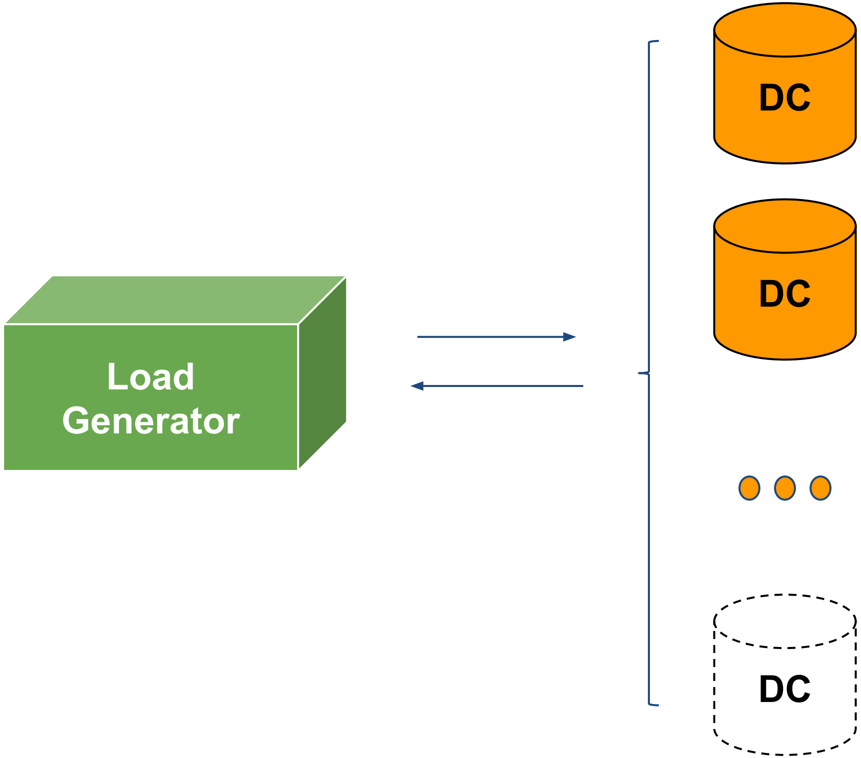
P0: Vertical Scaling Limitation

- However, one instance will always have limited resources.

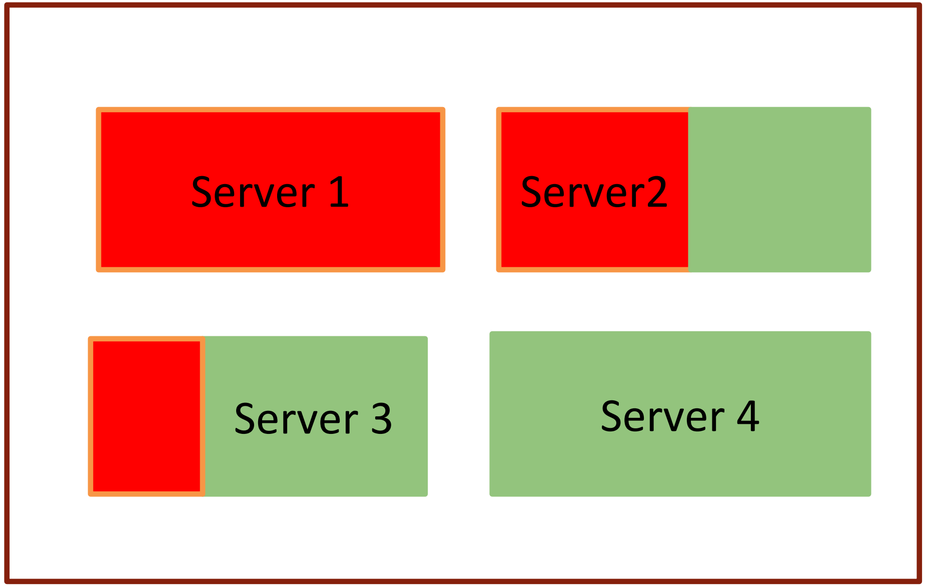
- Reboot/Downtime.



Horizontal Scaling



How do we distribute load?



Server 1

Server2

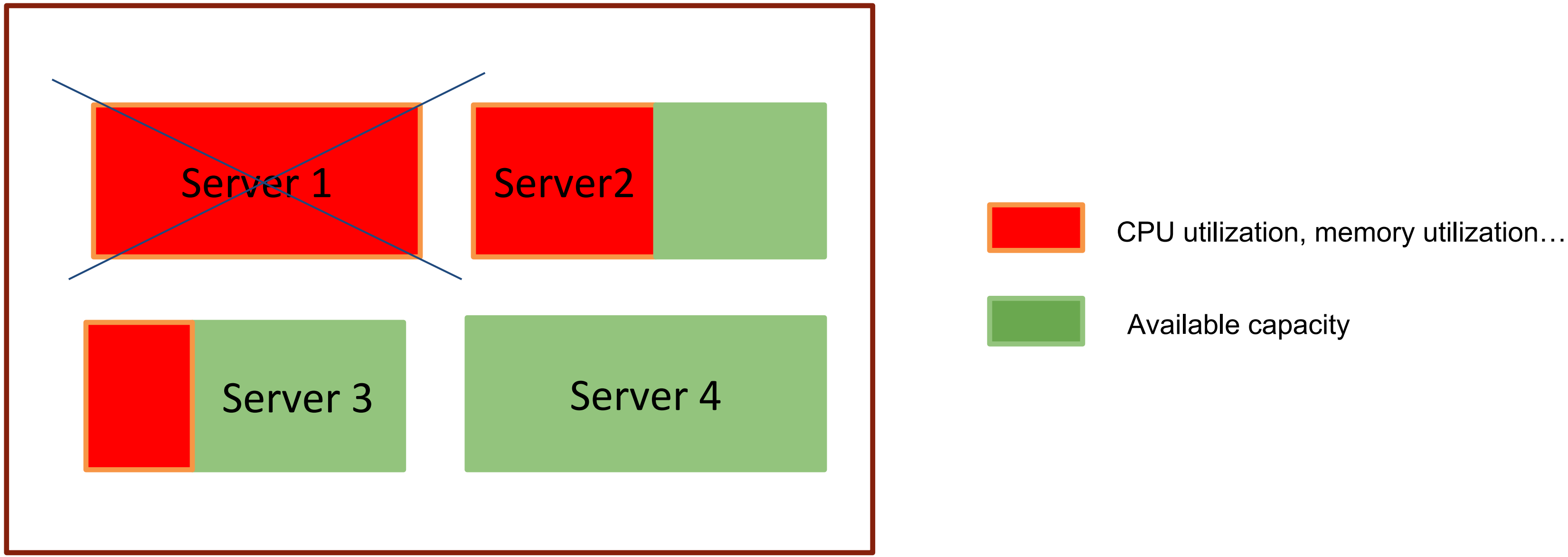
Server 3

Server 4

🟥 CPU utilization, memory utilization…

🟩 Available capacity

Instance Failure?



Server 1

Server2

Server 3

Server 4

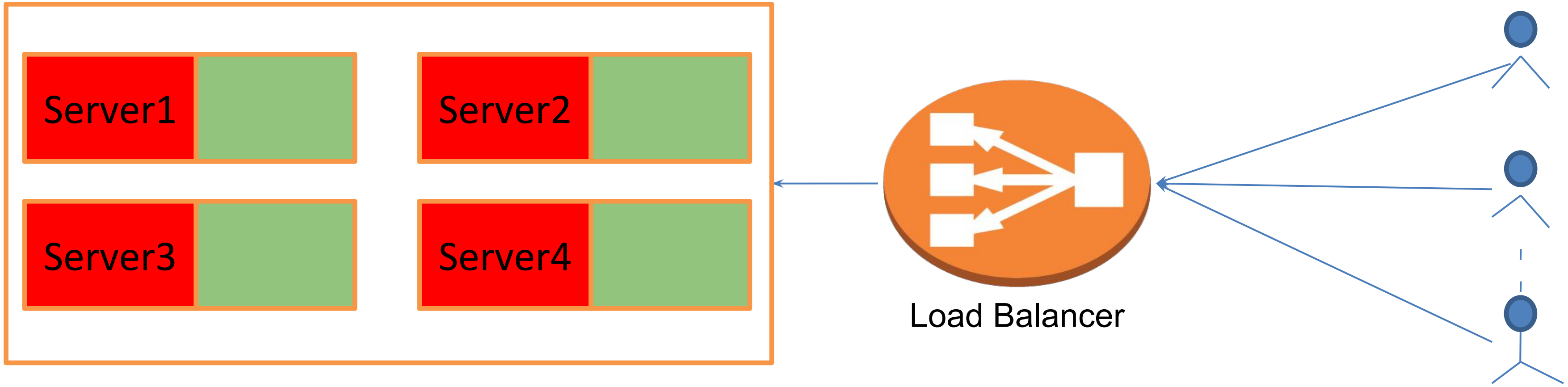
🟥 CPU utilization, memory utilization…

🟩 Available capacity

What You Need

- Make sure that workload is even on each server
- Do not assign load to servers that are down
- Increase/Remove servers according to changing load

How does a cloud service help solve these problems?



Managed group of servers

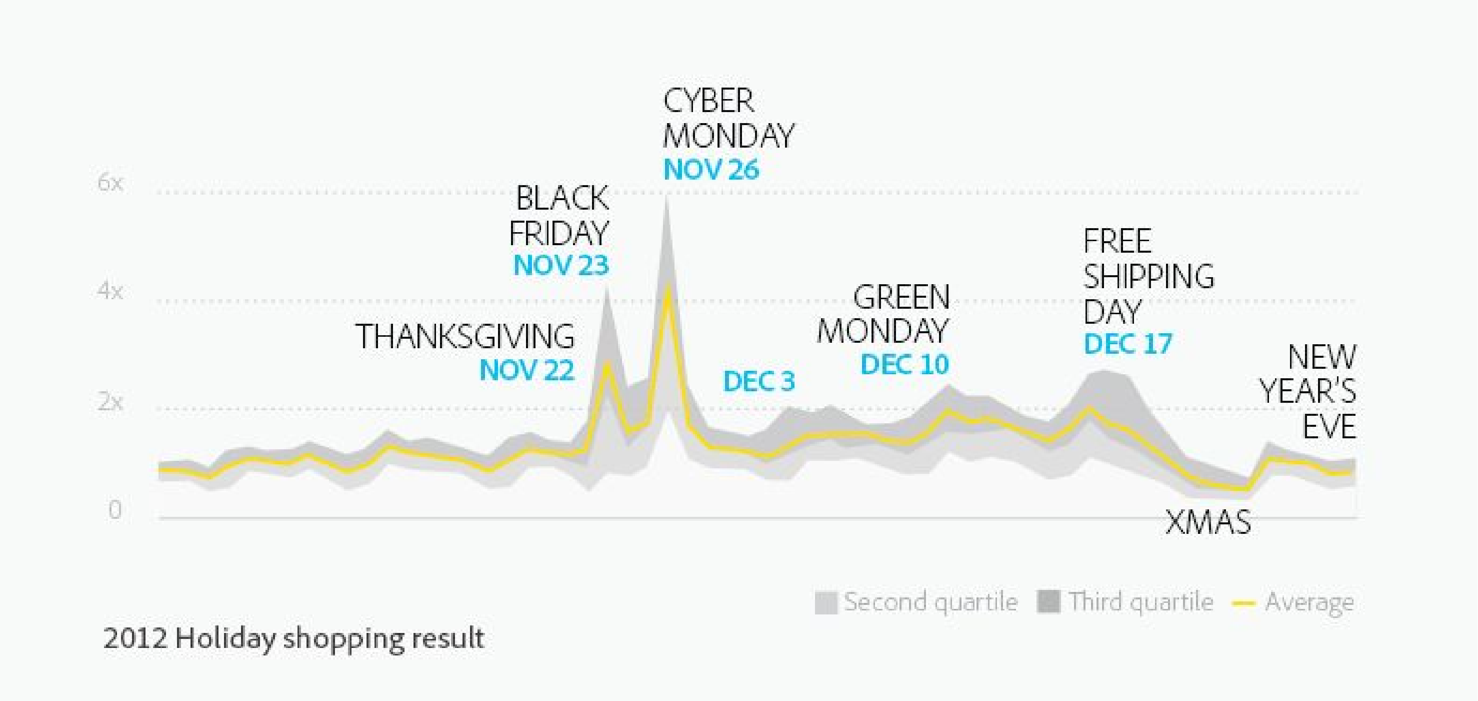
Load balancer

- “Evenly” distribute the load
- Simplest distribution strategy
    - Round Robin
- Health Check

Load Balancer

- What if the Load Balancer becomes the bottleneck?
    - Elastic Load Balancer
        - Could scale up based on load
    - Elastic, but it takes time
        - Through the warm-up process

Reality...



2012 Holiday shopping result

sapient.com

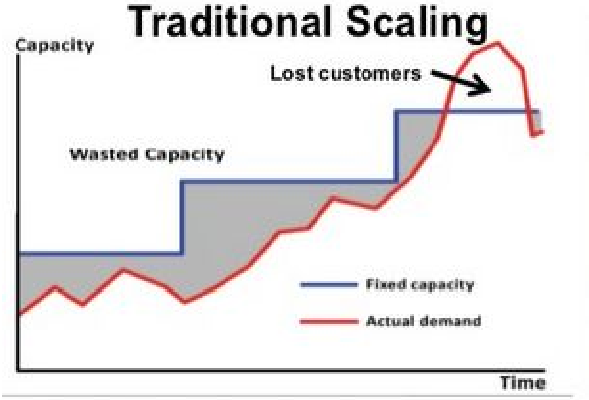
Scaling

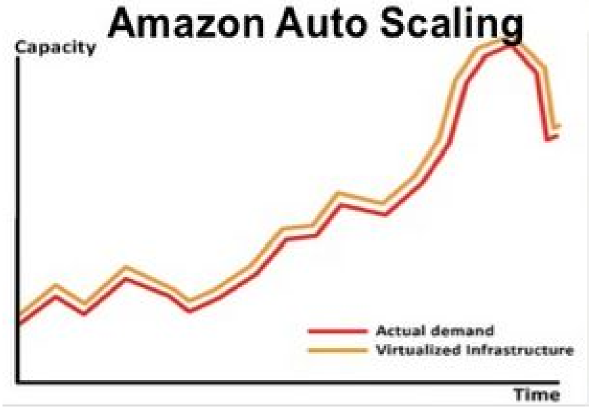
Manual Scaling:

- Expensive on manpower
- Low utilization or over provisioning
- Manual control
- Lose customers

Autoscaling:

- Automatically adjust the size based on demand
- Flexible capacity and scaling sets
- Save cost





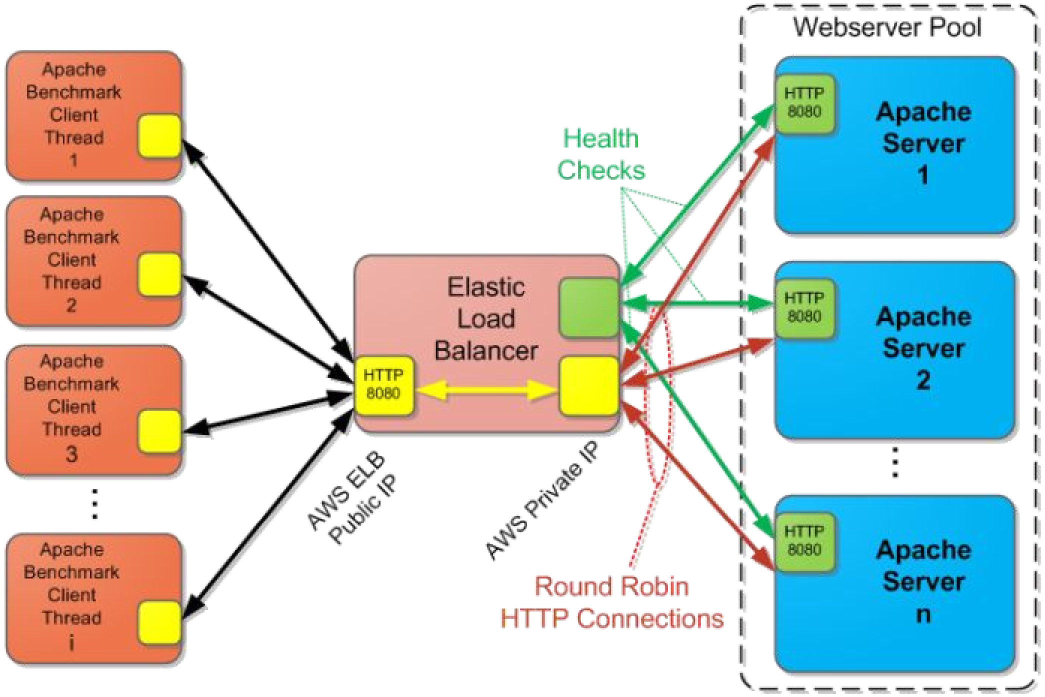
AWS Autoscaling

Auto Scaling on AWS

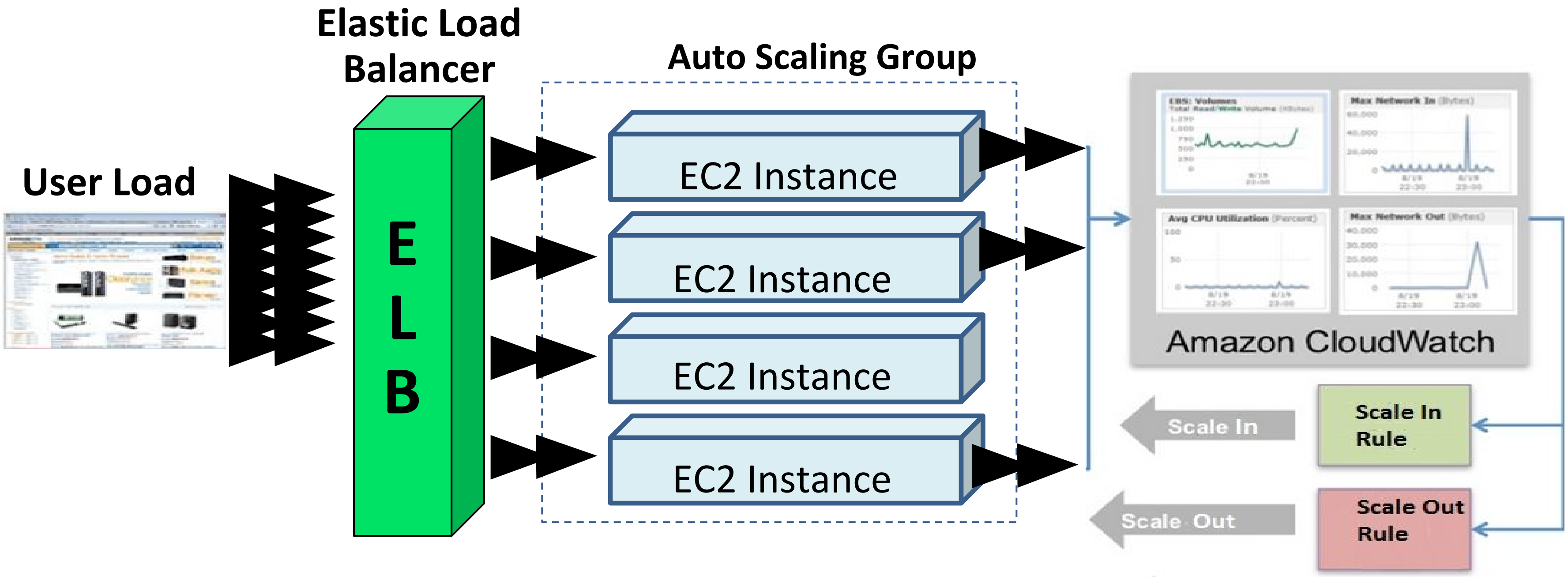
Using the AWS APIs:

- **CloudWatch**
- **ELB**
- **Auto Scaling Group**
- **Auto Scaling Policy**
- **EC2**

You can build a load balanced auto-scaled web service.

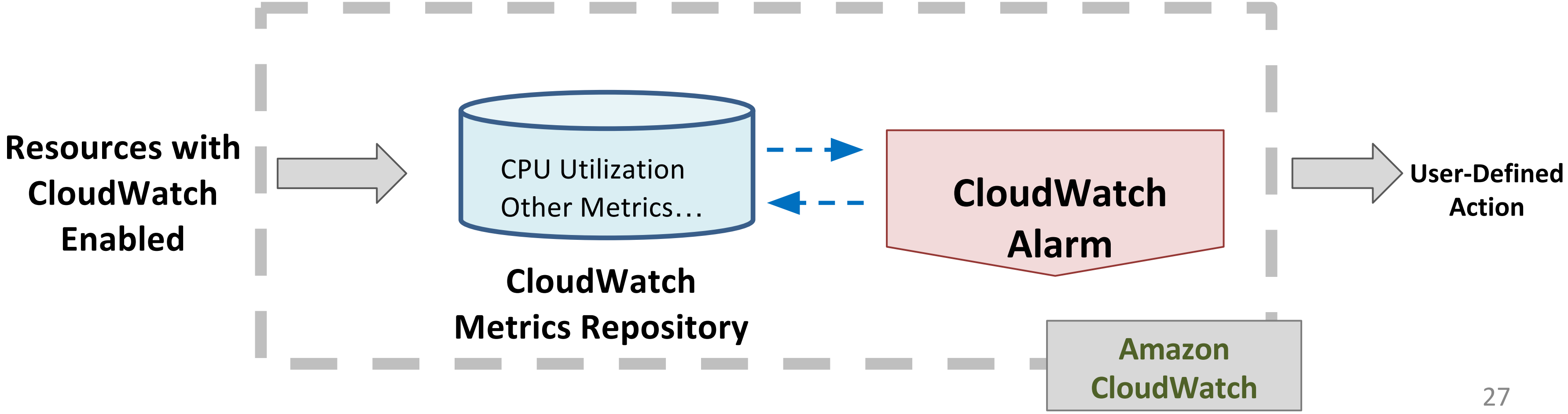


Amazon Auto Scaling Group



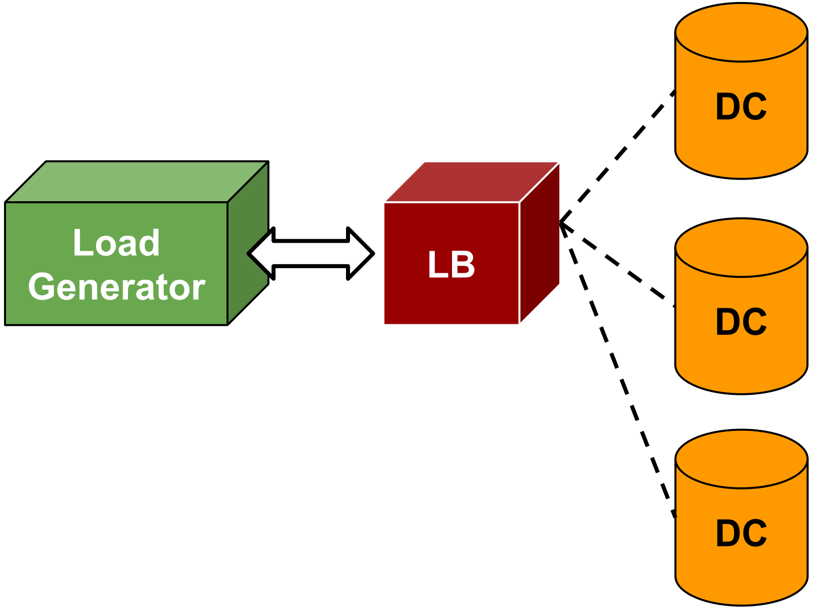
Amazon’s CloudWatch Alarm

- Monitor CloudWatch metrics for some specified alarm conditions
- Take automated action when the condition is met



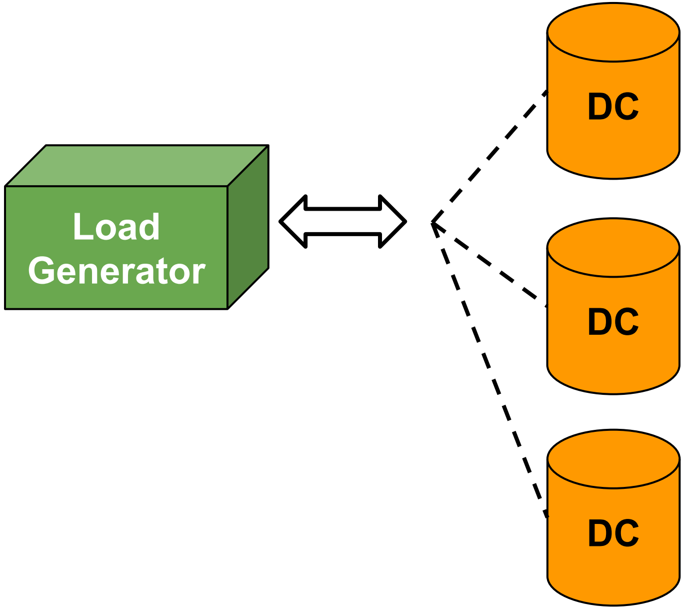
Project 2.1 Scaling on Azure and AWS

- Step 1
  - Azure Horizontal Scaling
- Step 2
  - Azure Auto Scaling
- Step 3
  - AWS Horizontal Scaling
- ❗ Step 4
  - AWS Auto Scaling



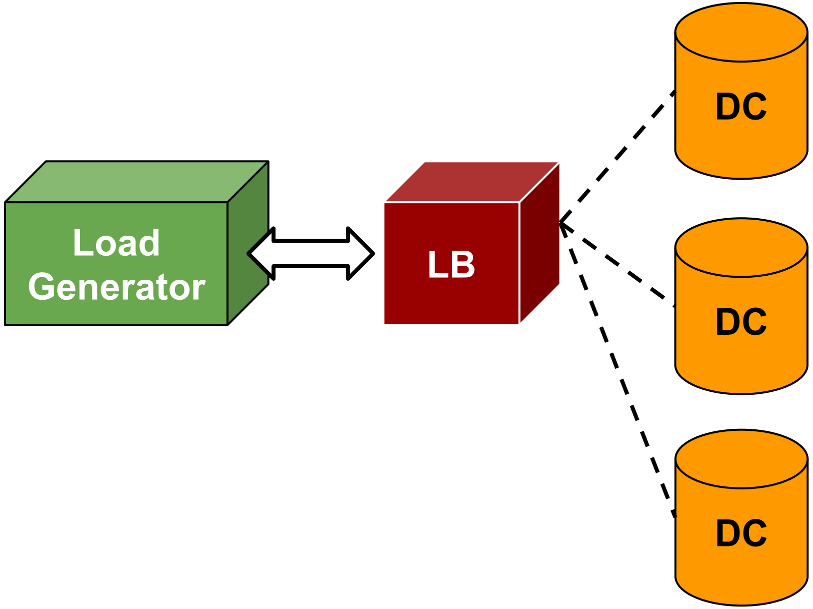
Project 2.1 Scaling on Azure and AWS

- Step 1 Azure Horizontal Scaling
- Implement Horizontal Scaling in Azure.
- Write a program that launches the data center instances and ensures that the target total RPS is reached.
- Your program should be fully automated: launch LG->submit password-> Launch DC-> start test-> check log -> add more DC...



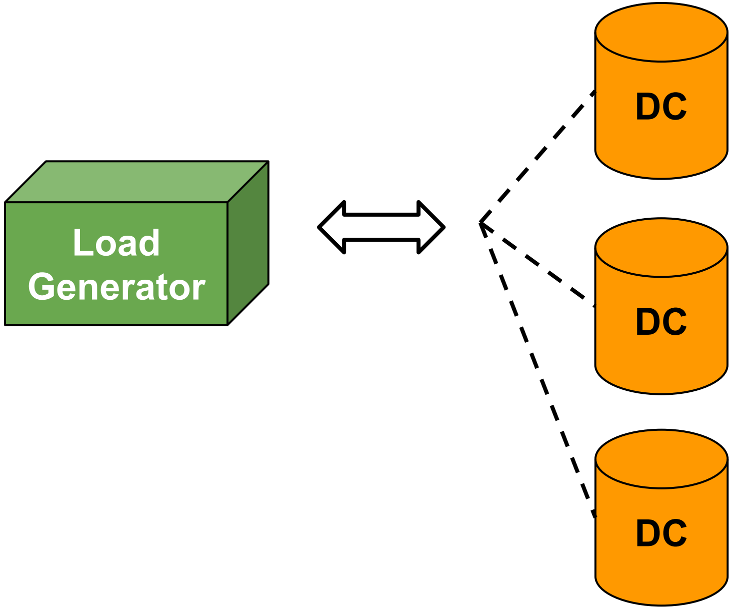
Project 2.1 Scaling on Azure and AWS

- Step 2
  - Azure Auto Scaling
- Deploy a VM Scale Set in Azure using the Json configuration files. You need to configure the policies (condition, action).
- You can start your test manually.
- VM Scale Set should be able to scale up when the load goes up.



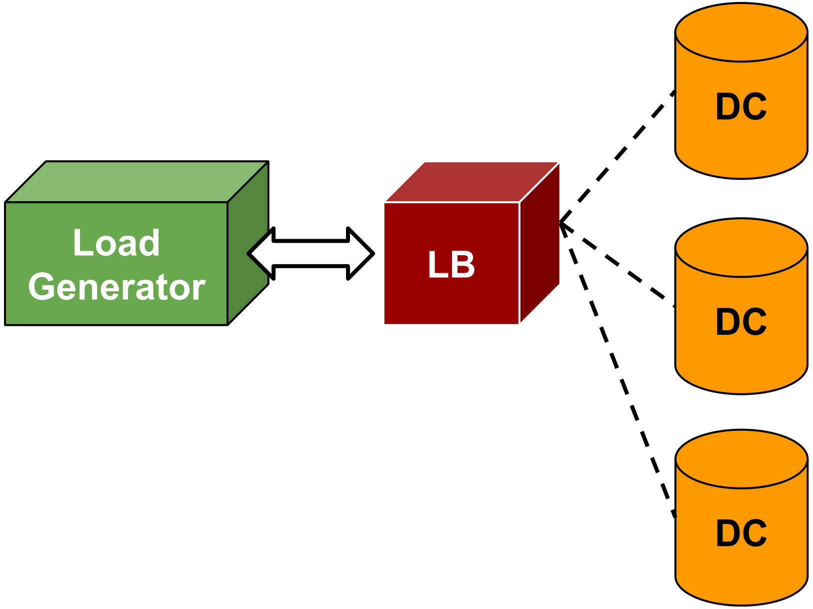
Project 2.1 Scaling on Azure and AWS

- Step 3
  - AWS Horizontal Scaling
- Very similar to Horizontal Scaling in Azure.
- Difference?
  - You need to use AWS API.



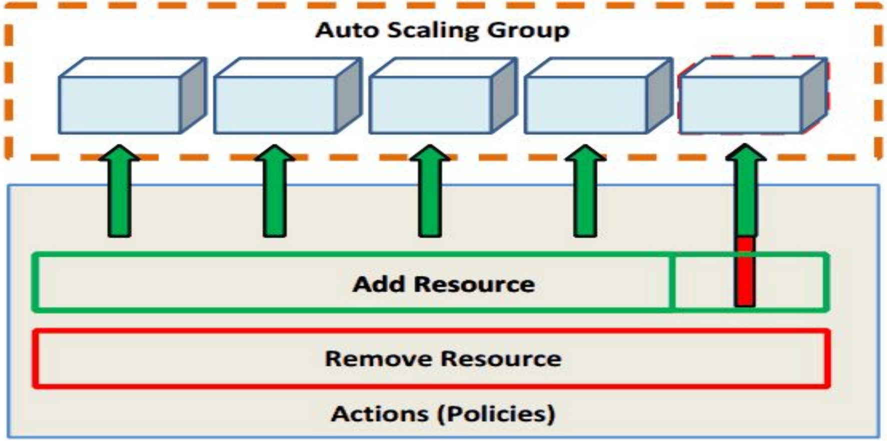
Project 2.1 Scaling on Azure and AWS

- Step 4
    - AWS Auto Scaling



P2.1, Step 4 - Your Tasks

- Programmatically create an Elastic Load Balancer (ELB) and an Auto Scaling Policy. Attach the policy to Auto-Scaling Group (ASG) and link ASG to ELB.
- Test by submitting a URL request and observe logs, ELB, and CloudWatch.
- Decide on the Scale-Out and Scale-In policies
- Mitigate failure



Elastic Load Balancer

Auto Scaling Group

CloudWatch Alarm

Hints for Project 2.1 AWS Autoscaling

Step 4 - AWS Auto Scaling

- Autoscaling Test could be very EXPENSIVE!
  - on-demand and charged by the hour
- Try your code on cheap instances (maybe t1.micro) first
- Creating and deleting security groups can be tricky
- CloudWatch and monitoring in ELB is helpful
- Explore ways to check if your instance is ready
- Understanding the API documents could take time
- Finish parts 1-3 first, the experience will help

Project 2.1 Code Submission

Azure:

- Submit on Azure’s load generator (LG) VM
- The code for the horizontal scaling task

AWS:

- Submit on AWS’s load generator instance
- Include code for both, the horizontal scaling task and the autoscaling task as a single zip file
- Add a readme file if necessary

Penalties for 2.1

Violation	Penalty of the project grade
Spending more than $20 for this project phase on AWS	-10%
Spending more than $35 for this project phase on AWS	-100%
Failing to tag all your resources in either parts (EC2 instances, ELB, ASG) for this project (AWS only)	-10%
Submitting your AWS credentials in your code for grading	-100%
Submitting the Azure part with AWS instances or the AWS part with Azure VMs	-100%
Using instances other than m3.medium or m3.large for Autoscaling on AWS	-100%
Using virtual machines other than Standard_A1(DC) and Standard_D1(LG) in the Azure part	-100%

Cloud APIs

AWS APIs:

- Command Line Interface API Tools (link)
- AWS SDK for Java (link)
- AWS SDK for Python (link)

Azure APIs:

- Command Line Interface API Tools (link)
- Azure SDK for Java (link)
- Azure SDK for Python (link)

15619Project Team Up

15-619 Students:

- Start to form your teams
  - Choose carefully as you cannot change teams
  - Look for a mix of skills in the team
    - Front end
    - Back end
    - ETL
- Create an AWS account only for the team project
- Wait for our post on Piazza to submit your team information

This Week’s Deadlines

- **Quiz 3 (OLI Modules 5 & 6)**
  - Due on **Friday**, Feb 5th, 2016, 11:59PM ET

- **Project 2.1**
  - Due on **Sunday**, Feb 7th, 2016, 11:59PM ET

DEMO

1. Copy images to your storage account in Azure
2. Create VM using the Azure API
3. Deploy Azure Virtual Machine Scale Set using JSON configuration files

Questions?