15-319 / 15-619
Cloud Computing

Recitation 4

September 19th, 2017

Administrative Issues

- Make use of office hours
  - We will have to make sure that you have tried yourself before you ask
- Monitor AWS, Azure, and GCP 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 GCP 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 **100s MB** of wikipedia data
  - P1.2: Parallel Analysis of **35GB** compressed / **128GB** decompressed 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 the Team Project and Project 4

This Week

- **Quiz 3 (OLI Modules 5 & 6)**
  - Due on **Friday**, Sept 22nd, 2017, 11:59PM ET
- **Project 2.1**
  - Due on **Sunday**, Sept 24th, 2017, 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

Scaling and Elasticity with VMs and Containers

- **2.1 Scaling Virtual Machines**
  - Horizontal scaling in / out
    - AWS and Azure or GCP APIs
  - Load balancing, failure detection, and cost management on AWS
- **2.2 Scaling with Containers**
  - Building your own container-based services. Both Python / Java and Frontend / Backend.
  - Docker containers
  - Manage multiple Kubernetes Cluster
  - Multi Cloud deployments.

Project 2.1 Learning Objectives

- Invoke cloud APIs to programmatically provision and deprovision cloud resources based on the current load.
- Explore and compare the usability and performance of APIs used in AWS, GCP and 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 and analyze the performance tradeoffs due to budget restrictions.

Quality of Service (QoS)

Quantitatively Measure QoS

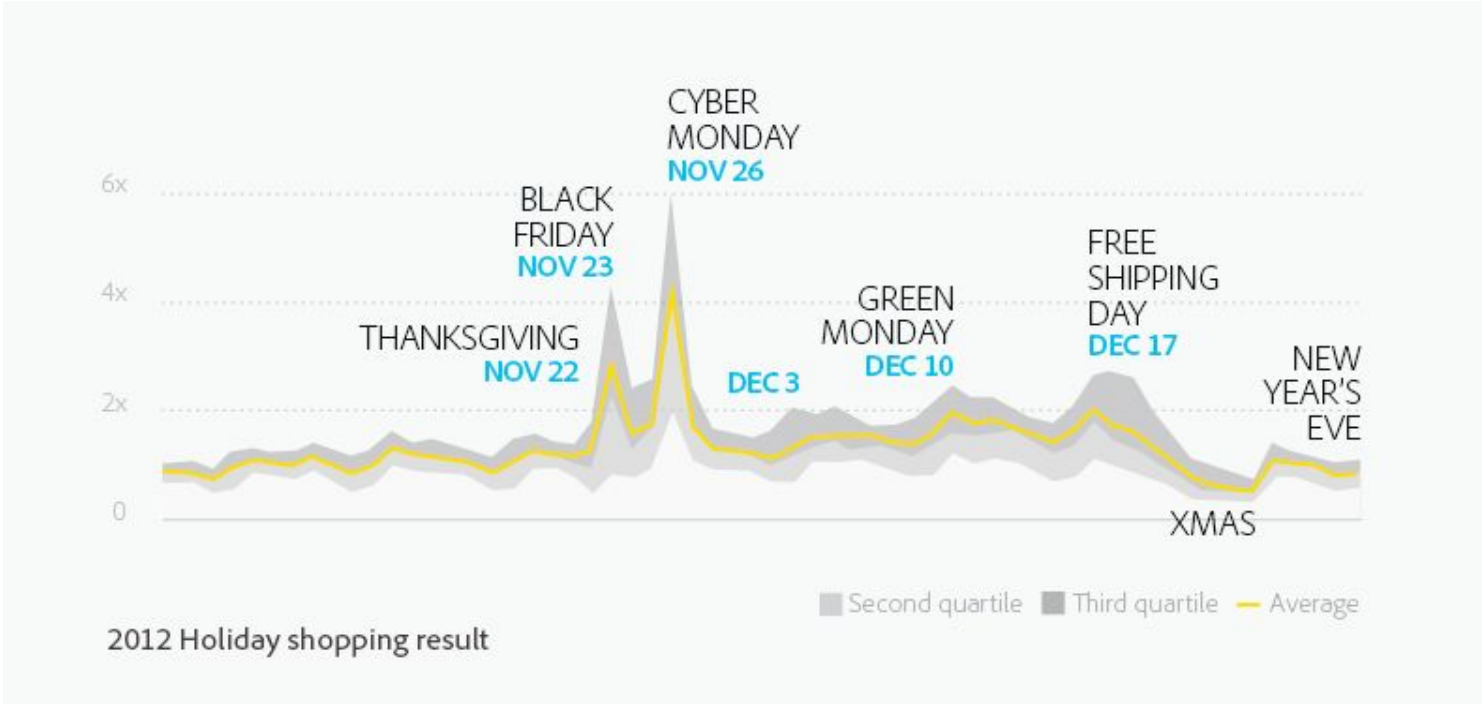
- Performance: Throughput, Latency
  (Very helpful in Projects 2 & Team Project)
- Availability: the probability that a system is operational at a given time
  (Projects P2.1 and P2.2)
- Reliability: the probability that a system will produce a correct output up to a given time *(Project P2.1 and P2.2)*

QoS Matters:

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

Reality, human patterns...

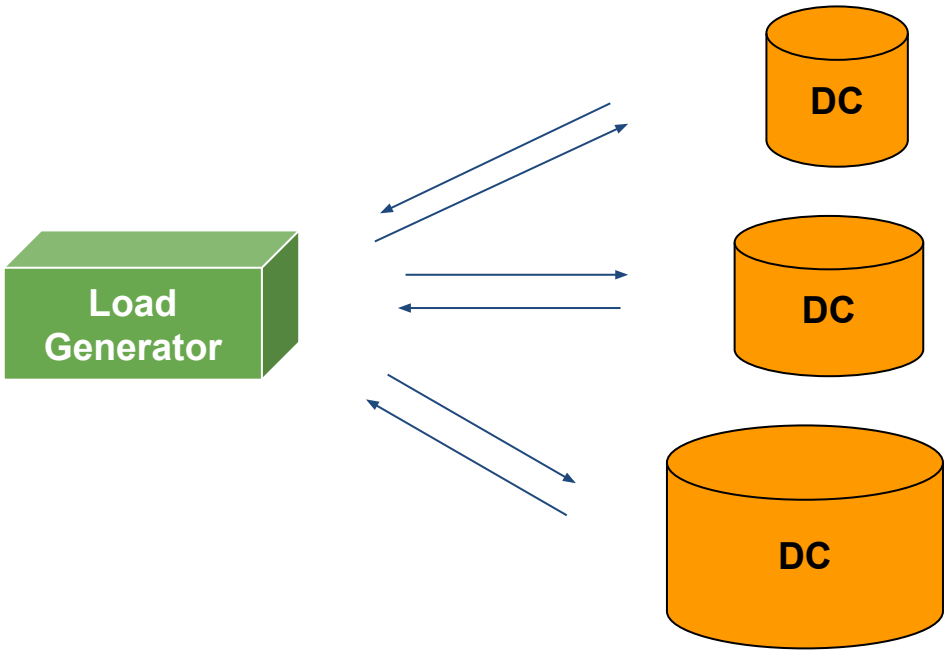
- Daily
- Weekly
- Monthly
- Yearly
- ...



sapient.com

Cloud Comes to the Rescue!
Scaling!

P0: Vertical Scaling

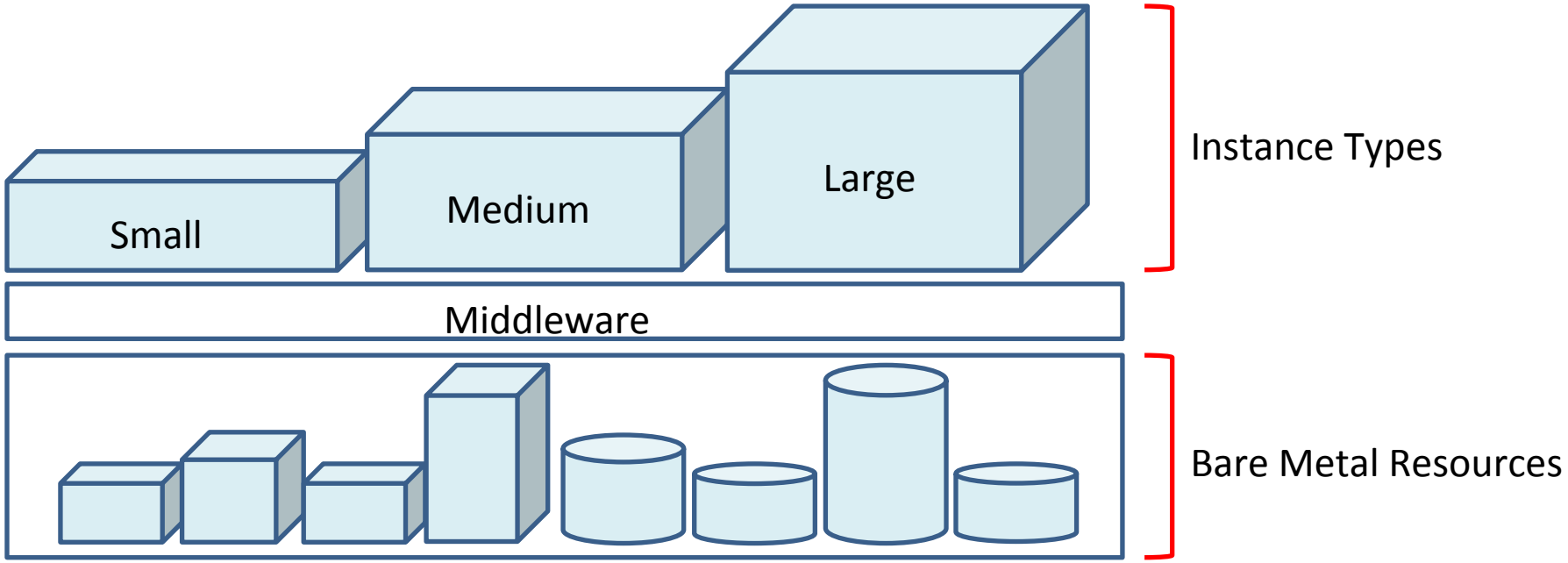


Small Data Center Instance
Serves 100 requests per second

Medium Data Center Instance
Serves 1000 requests per second

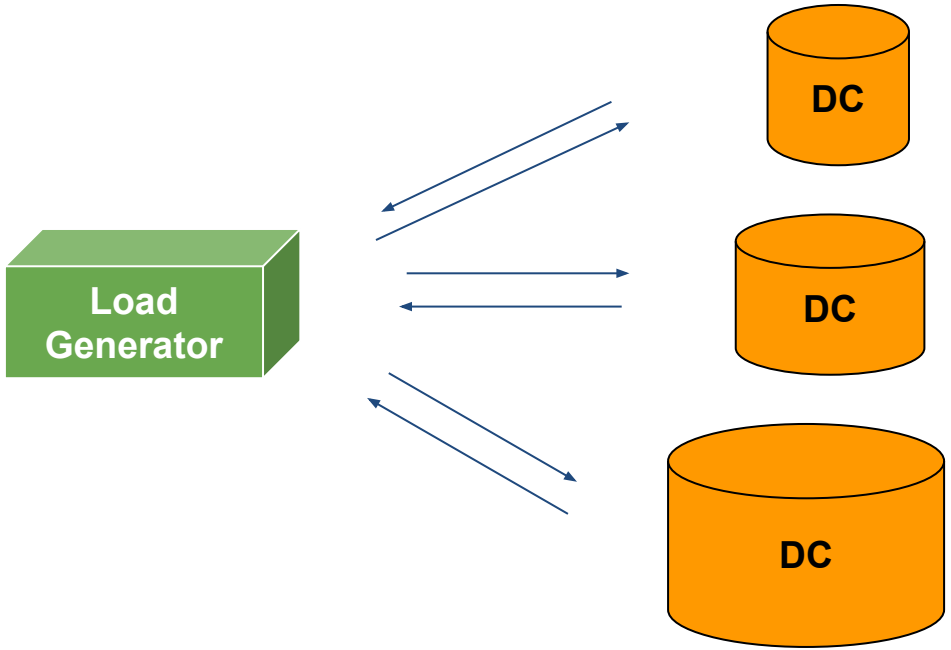
Large Data Center Instance
Serves 10000 requests per second

Resources in Cloud Infrastructure

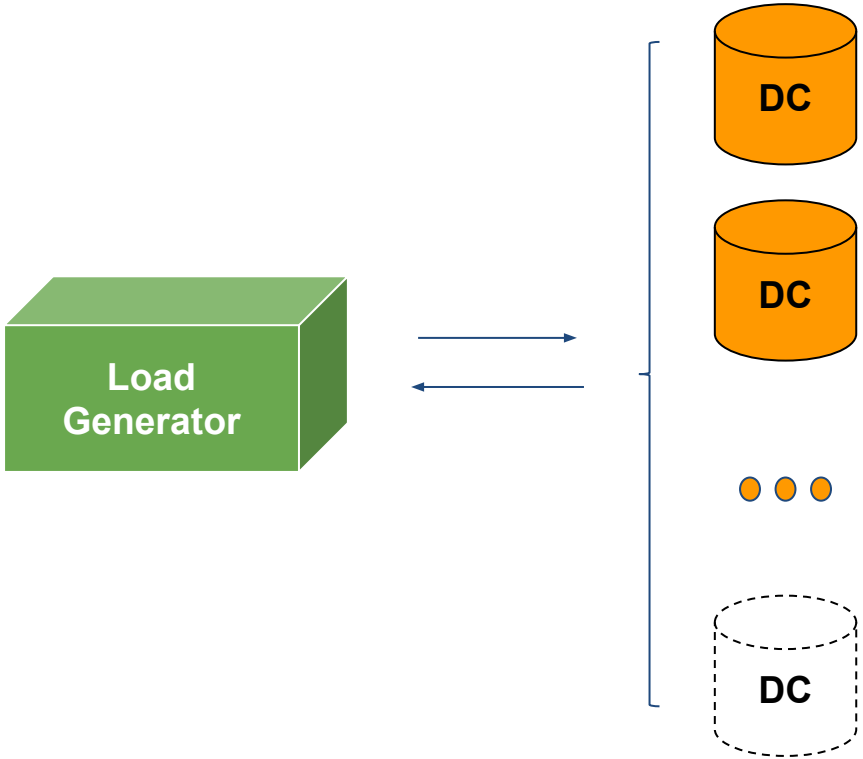


P0: Vertical Scaling Limitation

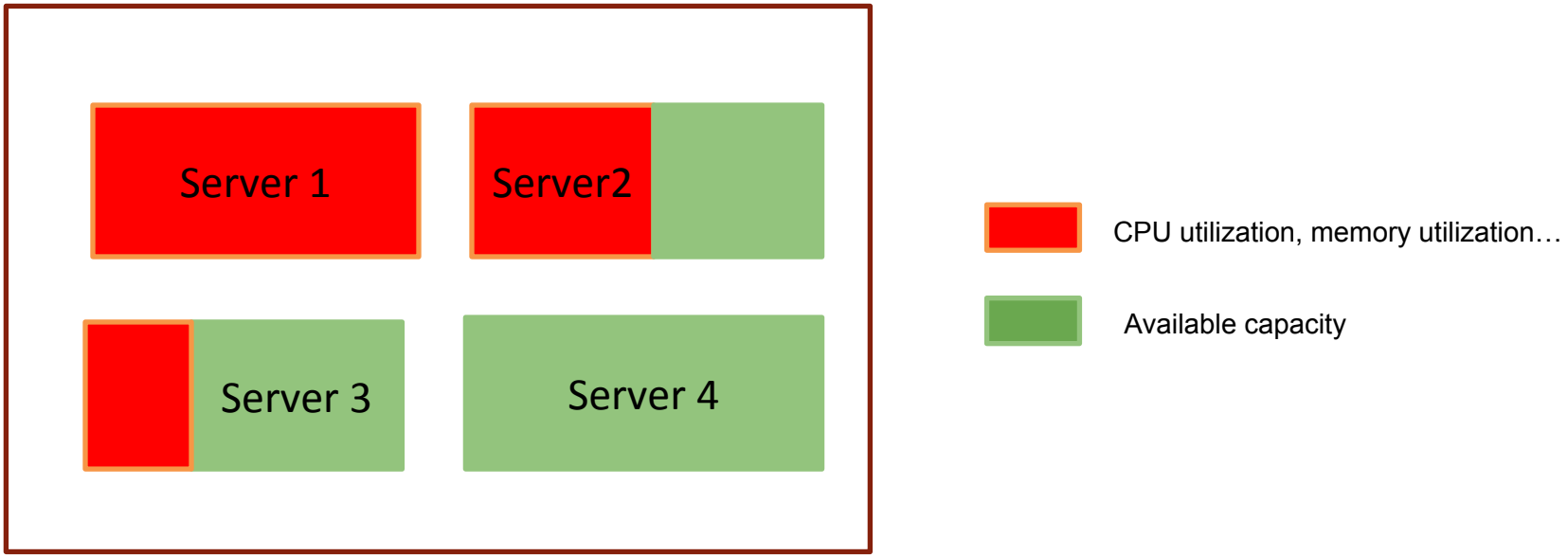
- However, one instance will always have limited resources.
- Reboot/Downtime.



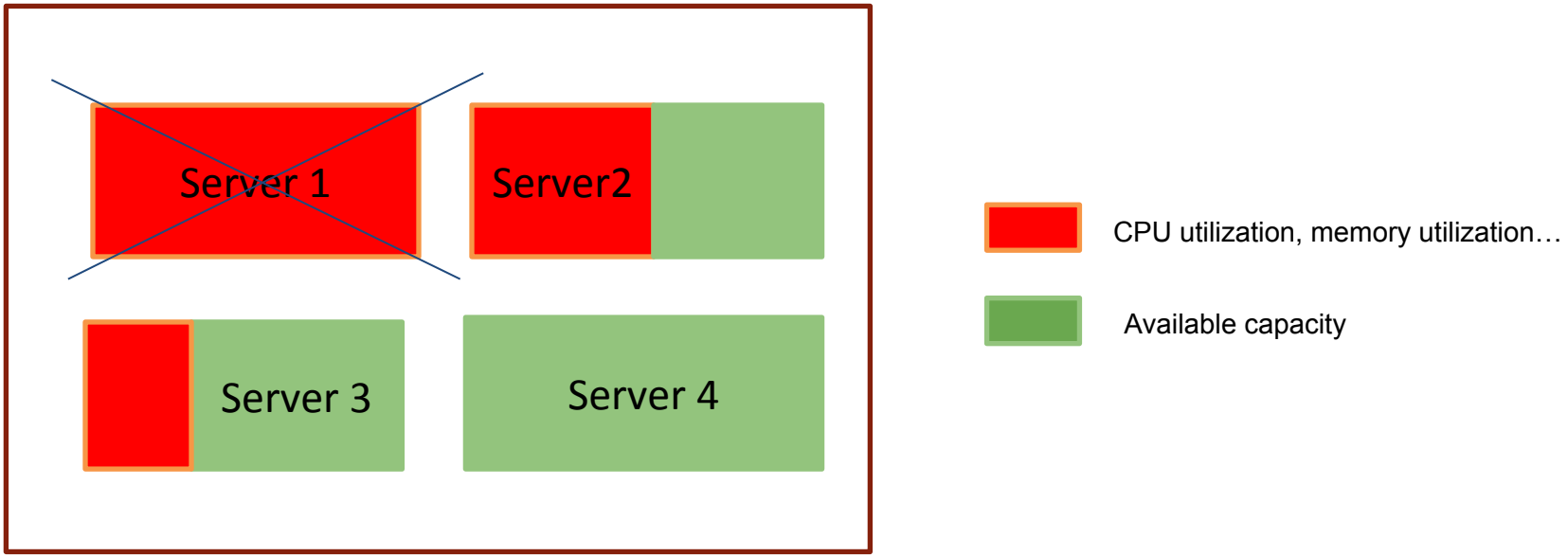
Horizontal Scaling



How do we distribute load?



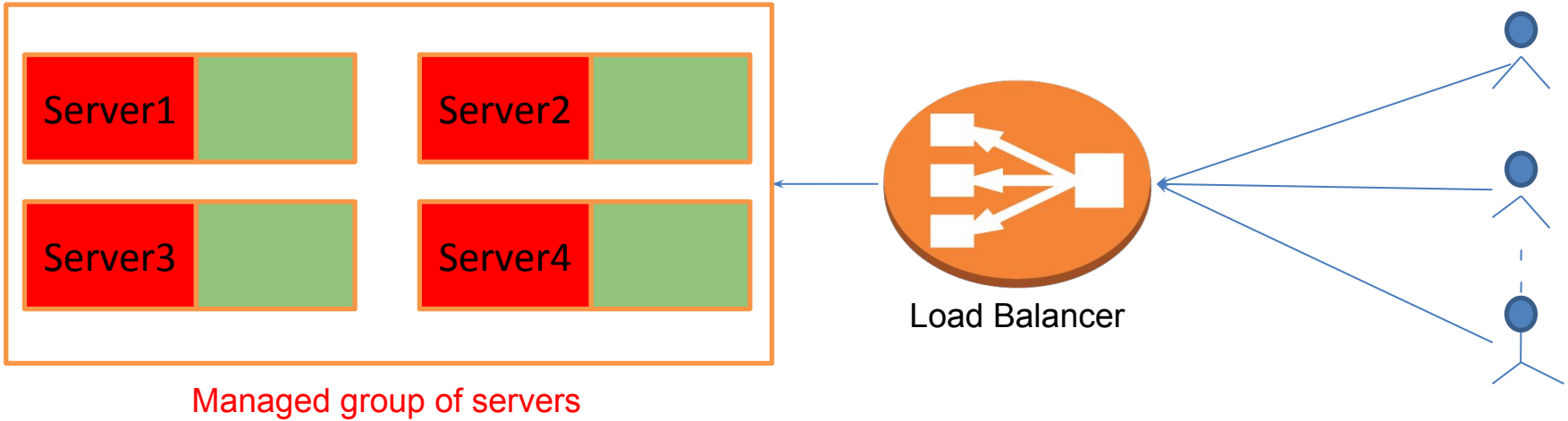
Instance Failure?



What You Need

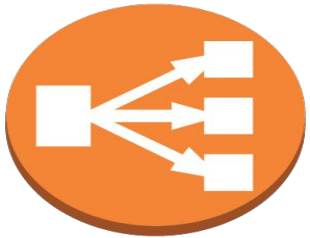
- Make sure that workload is even on each server
- Do not assign load to servers that are down
- Increase/Remove servers according to changing load

How does a cloud service help solve these problems?



Load balancer

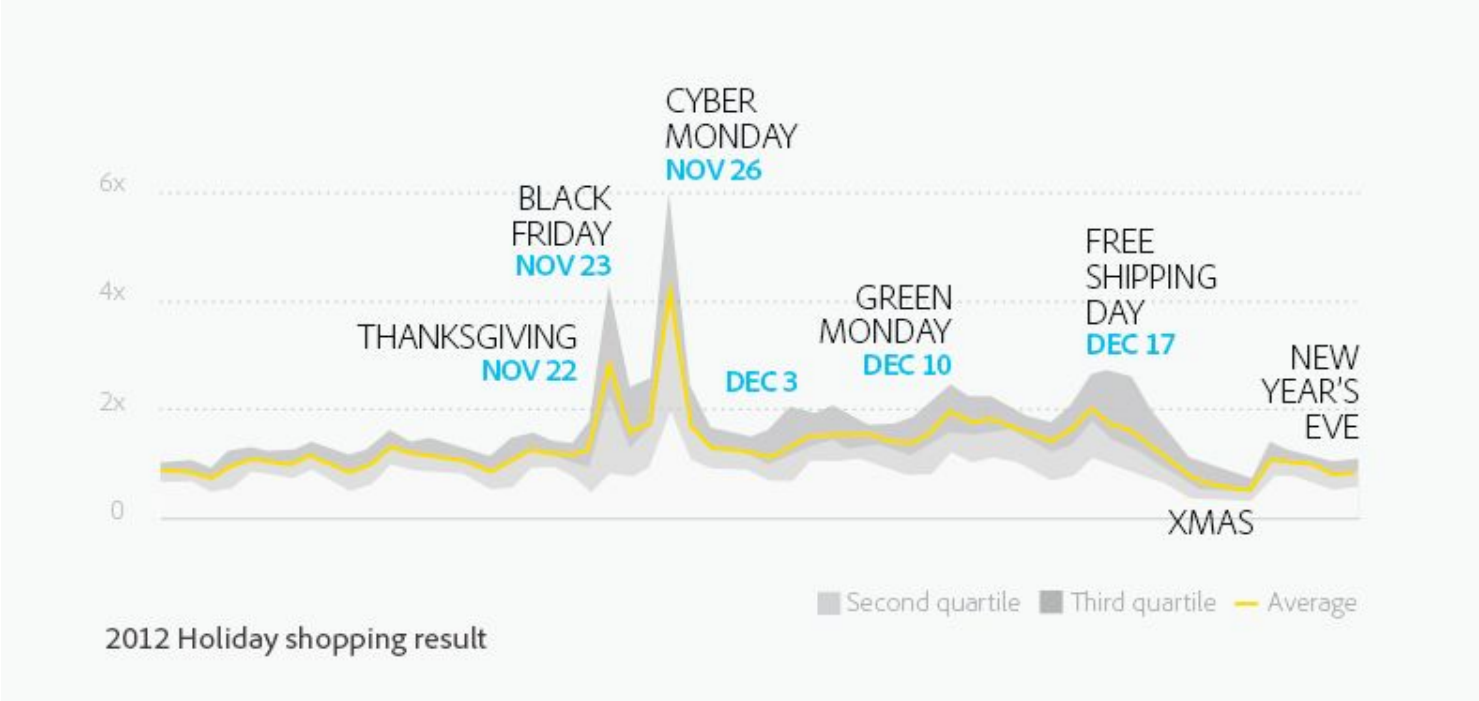
- “Evenly” distribute the load
- Simplest distribution strategy
  - Round Robin
- Health Check



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...



sapient.com

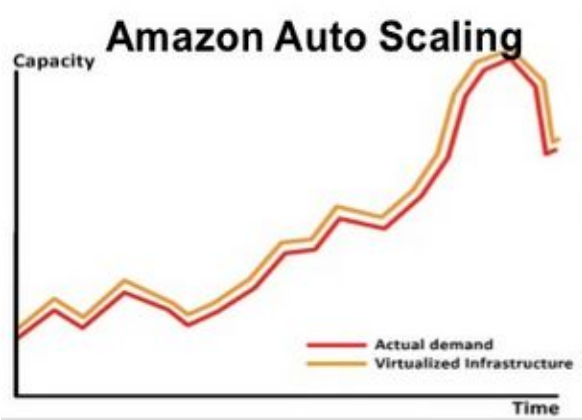
Scaling

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



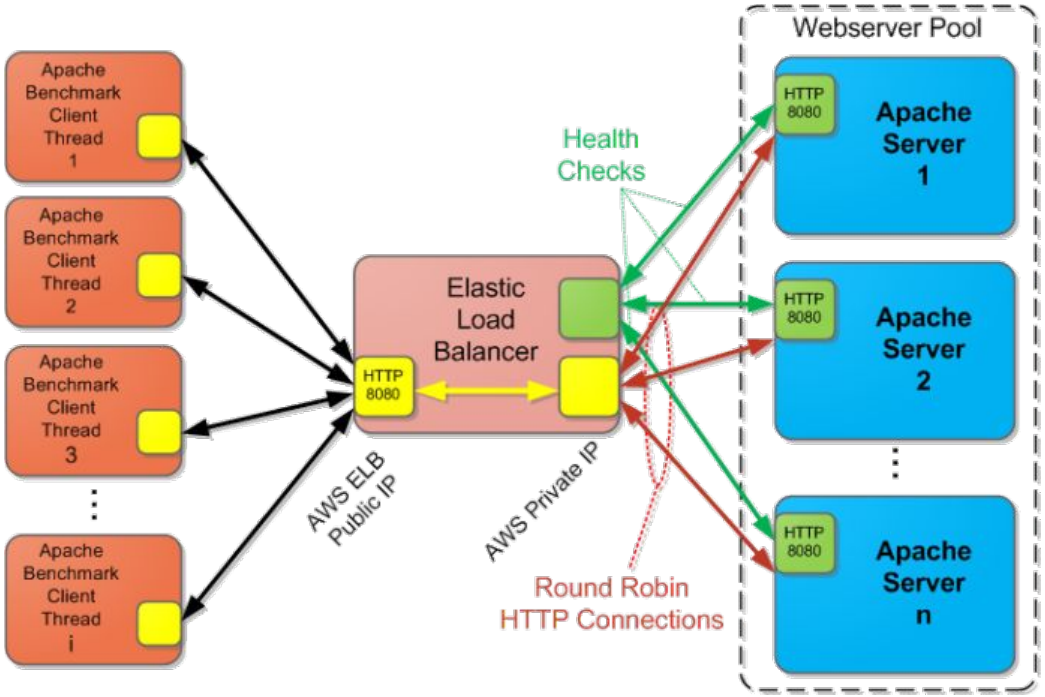
AWS Autoscaling

Auto Scaling on AWS

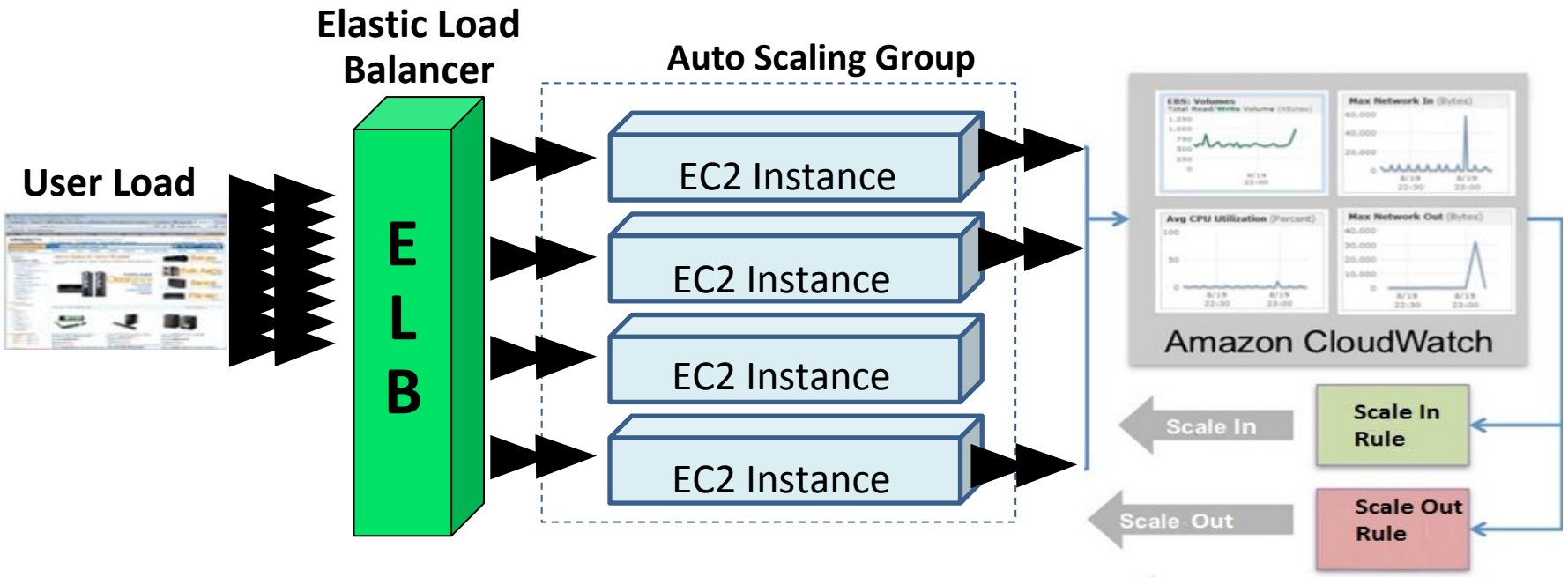
Using the AWS APIs:

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

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

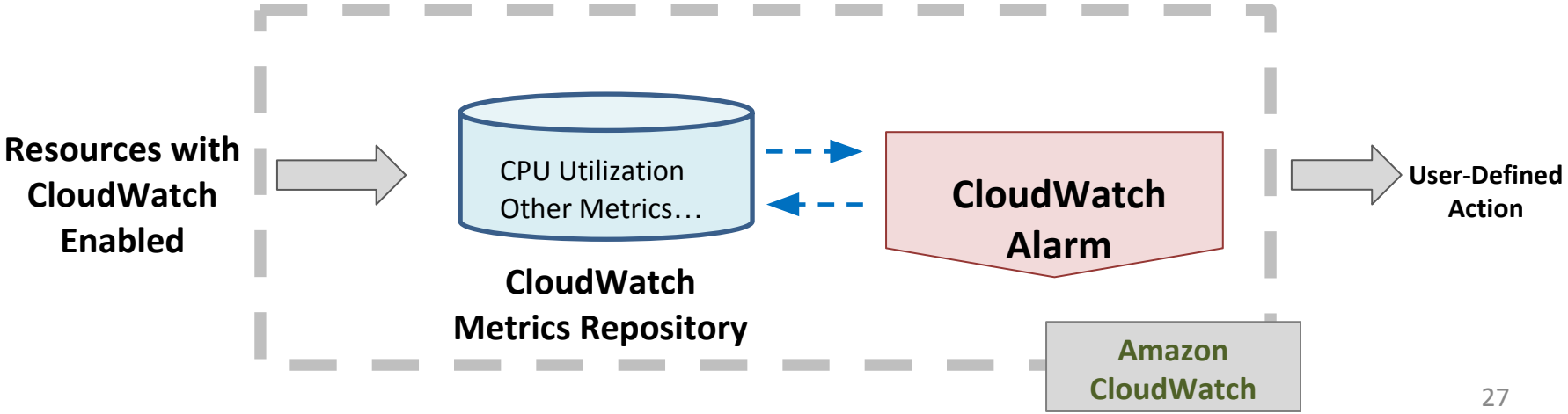


Amazon Auto Scaling Group



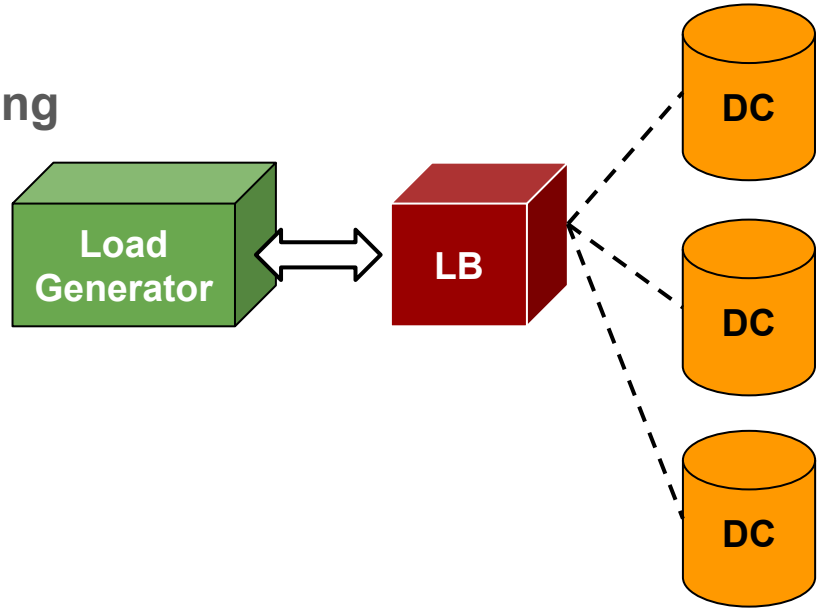
Amazon's CloudWatch Alarm

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



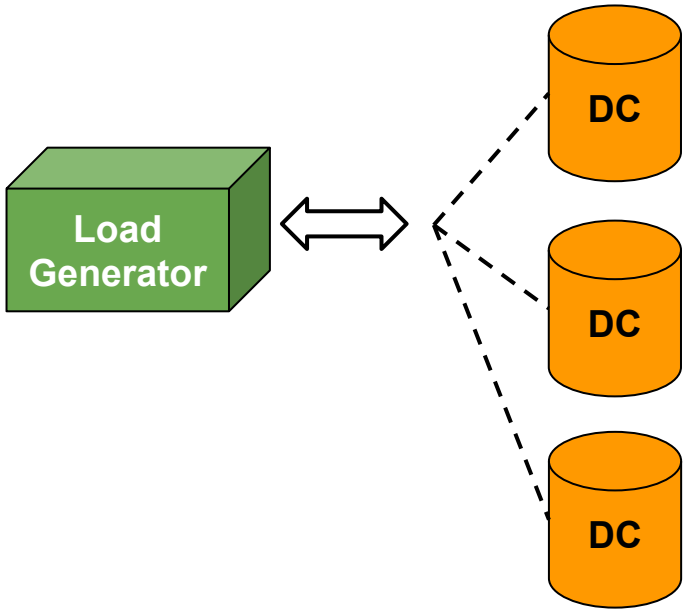
Project 2.1 Scaling on Azure, GCP, and AWS

- Step 1
  - Azure or GCP Horizontal Scaling
- Step 2
  - AWS Horizontal Scaling
- **Step 3**
  - **AWS Auto Scaling**



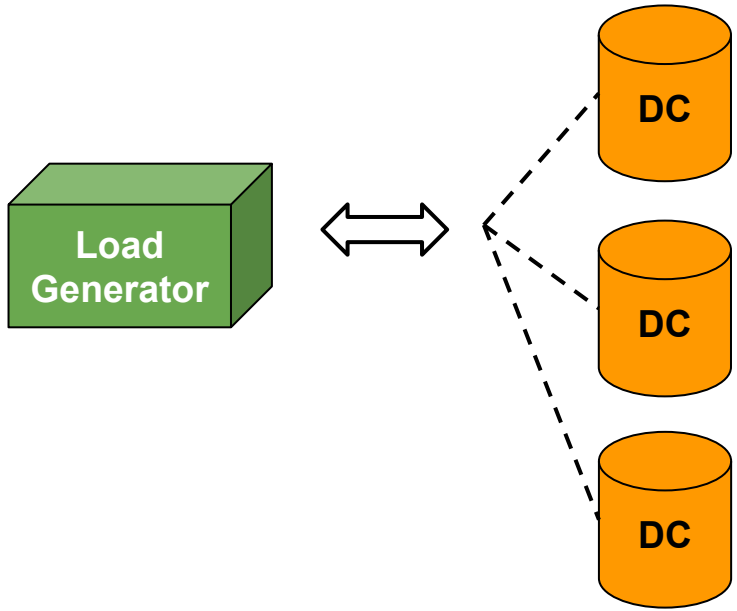
Project 2.1 Scaling on Azure, GCP, and AWS

- Step 1 Azure or GCP Horizontal Scaling
- Implement Horizontal Scaling in Azure or GCP.
- 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...



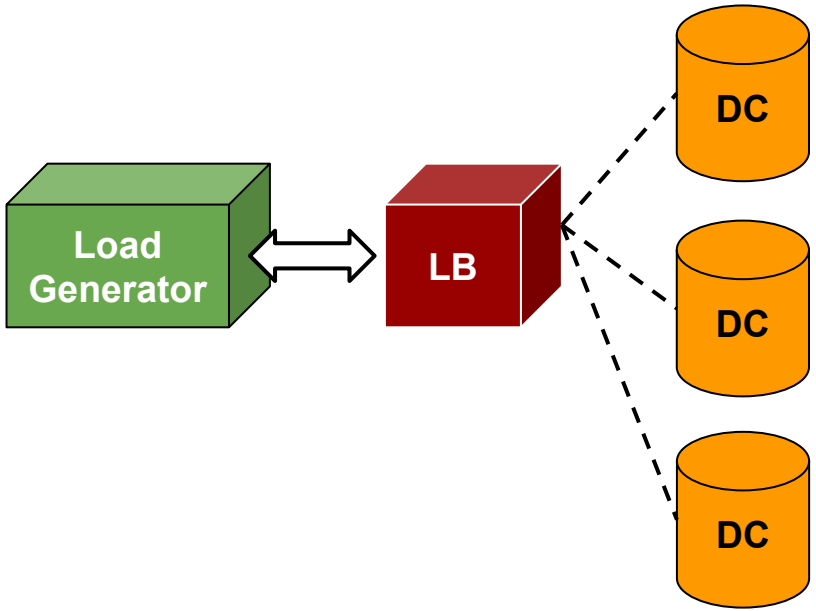
Project 2.1 Scaling on Azure, GCP, and AWS

- Step 2
  - AWS Horizontal Scaling
- Very similar to Horizontal Scaling in Azure and GCP.
- Difference?
  - You need to use AWS API.



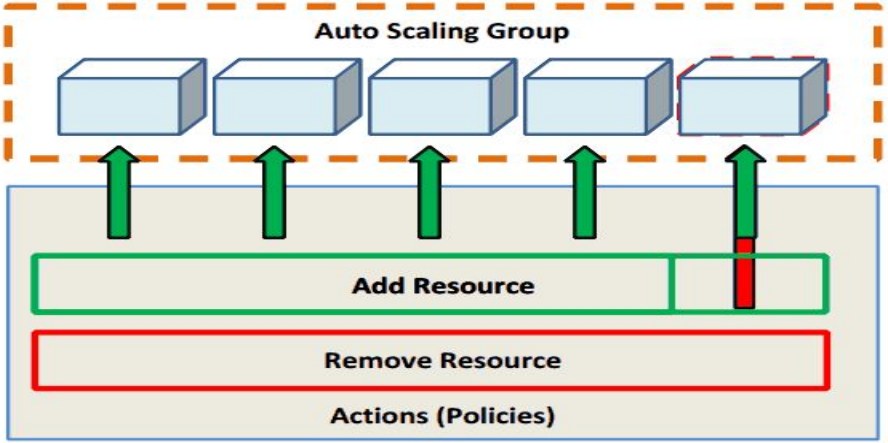
Project 2.1 Scaling on Azure, GCP, and AWS

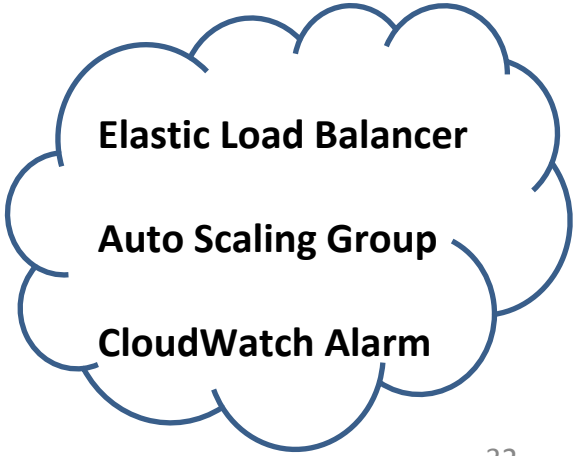
- Step 3
  - AWS Auto Scaling



P2.1, Step 3 - 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





Hints for Project 2.1 AWS Autoscaling

Step 3 - AWS Auto Scaling

- Autoscaling Test could be very EXPENSIVE!
  - on-demand and charged by the hour
- Determine if there is a less expensive means to test your solution
- 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 or GCP:

- Submit on Azure or GCP's load generator (LG) VM
- The code for the horizontal scaling task
- Add a readme file describing the content of your folders

AWS:

- Submit the horizontal scaling task on AWS's load generator (LG) instance
- Submit the autoscaling task to the AWS load generator (LG) instance
- Add a readme file describing the content of your folders

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). You must use tag: key=Project, value=2.1	-10%
Submitting your AWS/GCP/Azure/Andrew credentials in your code for grading	-100%
Completing the test for one cloud with instances from another.	-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%

Penalties for 2.1 cont.

Violation	Penalty of the project grade
Using virtual machines other than n1-standard-1 in the GCP part	-100%
Submitting executables (.jar, .pyc, etc.) instead of human-readable code (.py,.java, .sh, etc.)	-100%
Attempting to hack/tamper the autograder in any way	-100%
Cheating, plagiarism or unauthorized assistance (please refer to the university policy on academic integrity and our syllabus)	-200% & potential dismissal

AWS Cloud APIs

- AWS CLI (link)
- AWS Java SDK (link)
- AWS Python SDK (link)

Azure Cloud APIs

- Microsoft Azure CLI 2.0 (link)
- Azure Java SDK (link)
- Azure Python SDK (link)

GCP Cloud APIs

Google Cloud Platform

- gcloud CLI (link)
- GCP Java Client Libraries (link)
- GCP Python Client Libraries (link)

Team Project - Time to 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**, Sept 22nd, 2017, 11:59PM ET
- **Project 2.1**
  - Due on **Sunday**, Sept 24th, 2017, 11:59PM ET

Questions?