15-319 / 15-619 Cloud Computing

Course Overview 2

September 07, 2021

Accessing the Course

- Open Learning Initiative (OLI) Course
  - Access via canvas.cmu.edu
- The Sail() Platform (access through canvas)
  - choose CMU as the identity provider
  - AWS Account Setup (AWS, Azure, GCP)
  - Update your course profile with AWS, Azure & GCP info
  - Complete the Primers on AWS, Azure and GCP
- Piazza

Amazon Web Services (AWS) Account

- ONLY IF YOU HAVEN'T DONE SO ALREADY
- Log on to the Sail() platform through Canvas and make sure you follow the instructions in the Account Setup Primer
- Wait to receive Consolidated Billing Request email from Amazon
  - The linking email is sent automatically, waiting time varies
  - You need to **manually** accept the linking request
- When you receive the linking email, click the link to verify the linked billing
  - Many students have not clicked on the link yet!
    - Check your **SPAM** folder
  - You won't be able to complete the projects without a linked account.

Azure Account

- ONLY IF YOU HAVEN'T DONE SO ALREADY
- Do **not** use your @andrew.cmu.edu or other CMU issued email address.
- Update your course profile and set up Azure subscription

Google Cloud Platform (GCP) Account

- ONLY IF YOU HAVEN'T DONE SO ALREADY
- Follow the instructions in the primer.
- Receive a $50 coupon on the Sail() platform
- Redeem the coupon

Piazza

- Suggestions for using Piazza
  - Discussion forum for a learning community
  - Contribute good questions and answers
- When you have a (project-specific) problem, follow the order below!
  - Try to solve the problem by yourself (Search, Stack Overflow)
  - Read Piazza questions & answers carefully to avoid duplicates
    - Visit TA OHs: TA office hours are posted on Piazza and Google calendar
    - Create a piazza post
- Please note:
  - Try to ask a public question if possible
  - Don't ask a public question about a quiz question
  - Read the Piazza Post Guidelines before asking
  - Show the effort you have made to solve the problem
  - Practice how to communicate effectively in a technical setting
  - The key to effective communication is to provide the full context.

Piazza - Provide the full context

- Which project module are you working on?
- Which task/section are you working on?
- If relevant, please provide the information of the cloud account and resources.
- Example error message in the plain text format (if you are reporting programmatic issues) or screenshot (if you are reporting UI/UX issues)
  - Please provide example error messages **in the plain text format**, **NEVER** share code/text as screenshots which are not parsable!
  - Use screenshots (only) for UI/UX issues
- How to reproduce?
- Expected behavior v.s. actual behavior
- Environment summary
- What you have tried?

Piazza - Articulate technical questions

- There are common patterns to communicate effectively in a technical setting.
- Our course not only aims at building your technical skills, but also training your communication skills.
- We created a template for you to structure your questions.

Reflecting on Last Week

- AWS, Azure and GCP accounts
- Cloud resources
  - AWS EC2, S3, CloudWatch
  - Azure Compute, Azure Storage
  - GCP Compute, GCP Storage
- Interface
  - Web console, CLI, SDK
- Basic SSH skills
- Jupyter Notebook primer
- Infrastructure as Code (Terraform) primer
  - Read it if you have not done so
  - Required by many projects including Project 1

Reflecting on Last Week (cont.)

In Project 0:

- You experimented with cloud resource provisioning with multiple cloud service providers.
- You experiment with the cloud-based development and deployment workflow.
- You quickly studied diverse topics within a short timeframe (i.e., a week), and transferred your learning to complete hands-on tasks with real-world scenarios:
  - tools (e.g., cloud platforms, Maven, Terraform, JUnit, JaCoCo, Jupyter Notebook, Pandas, Linux tools such as awk and grep, etc.)
  - practices (e.g., test-driven development, code coverage, encoding-aware I/O, etc.)
  - processes (e.g., budget, tagging and lifecycle management for cloud resources, etc.)

Programming Experience Expected

- **Strong proficiency** in at least one of the following, with some fair comprehension of the others:
  - **Java 8**
  - **Python 3**
  - **Bash**
- **Java and Python are required** to complete parts of Projects.
- Use the time now to brush up
- Please read Maven primer!
- Do not fear bash/python scripting, it will make your life easier!

Completing Projects in this Course

- Provision AWS, Azure or GCP Resources
  - Use the Cloud VM Images we provide for the project
  - Tag all instances!
- Plan and monitor your cost
  - Calculate costs before you provision!
- Complete tasks for each project
  - Each project writeup has several sections unlocked by AssessMe
- Submit your work
  - Check the score and feedback in the submission tab on the Sail() platform
- Terminate all resources when you have verified your score and kept a copy of your work (e.g., git **private** repo)

Tagging

- Tag *all* tag-able resources on AWS
  - Before you make a resource request, read the docs/specifications to find out if tagging is supported
  - We will specify which resources are required to be tagged in each project
  - Apply the tags during resource provisioning
  - We need tags to track usage, a grade penalty will be applied automatically if you do not tag!
  - Spot instances
    - Tags of a spot request do NOT propagate to the VMs!
    - AWS EC2 Fleet is the remedy
- Tagging Format
  - `Key: project`
  - `Value: getting-started-with-cloud-computing, vm-scaling, containers, etc.`

Budgets and Penalties

- No proper tags ➔ 10% grade penalty
- Provision resources in regions other than us-east-1 ➔ 10% grade penalty
- Budget
  - For P1, each student's budget is **$20**
  - Exceeding Budget ➔ 10% project penalty
  - Exceeding Budget x 2 ➔ 100% project penalty (no score)
  - You can see Cost and Penalties in the Sail() platform
- No exceptions
- We give you an opportunity to learn in Project 0 without affecting your grade
- We will enforce these penalties automatically starting from Project 1

Academic Integrity Violation

- Cheating ➔ the lowest penalty is a 200% penalty & potential dismissal
  - Other students, previous students, Internet (e.g. StackOverflow)
  - Do not work on code together
  - This is about you struggling with something and learning
  - Penalty for cheating is SEVERE – don't do it!
  - Ask us if you are unsure

Compromised Accounts

- People are scanning publicly available files for cloud credentials.
  - They compromise your account and launch resources in other regions.
- If you put any of your credentials in files on
  - Github, Dropbox, Google Drive, Box, etc.
  - You are vulnerable to getting your account compromised.
  - Going over 2x the project budget ⇒ 100% penalty!

Deadlines!

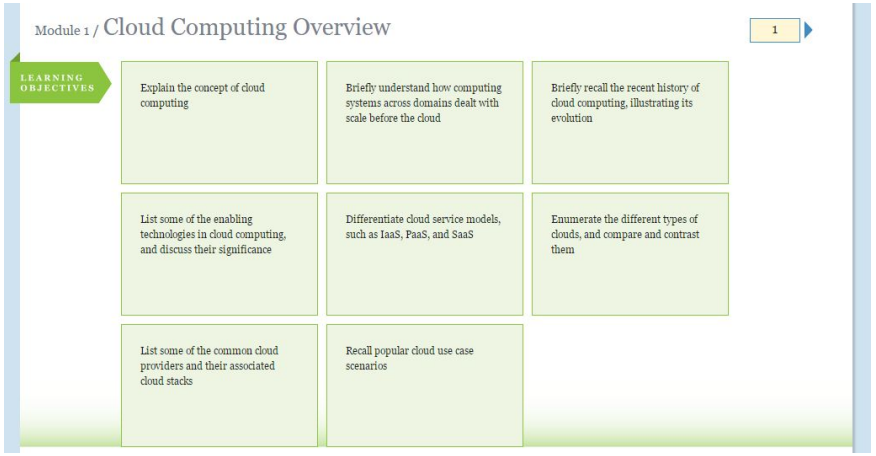
- **Hard** Deadlines
  - No late days, no extensions
  - Start early!
  - Plan your activities, interviews and other commitments around the deadlines.
  - **No exceptions!**
- Projects are typically due on Sundays at 23:59 ET
- Quizzes are typically due on Fridays at 23:59 ET

Deadlines

- **Project deadlines**
  - **On the Sail() Platform**
- **Quiz deadlines**
  - **On OLI**

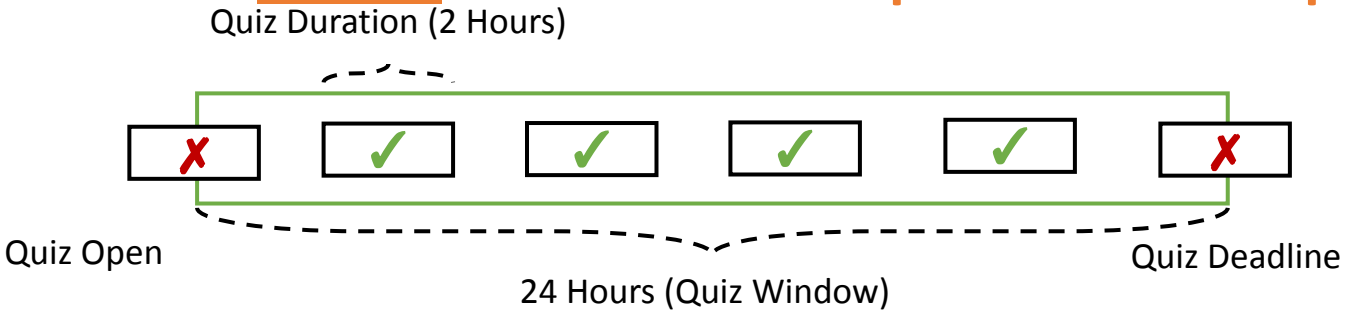
Quiz 1 Preparation

- Tests your understanding in Modules 1 and 2
  - Cloud computing fundamentals, service models, economics, SLAs, security
  - Use the activities in each page for practice.
  - You will be tested on you ability to perform the stated learning objectives on OLI:



Quiz 1 Logistics

- Quiz 1 will be open for 24 hours, Friday, Sep 10
  - All quizzes are **open-book** tests.
  - Quiz 1 becomes available on **Sep 10, 00:00 AM ET.**
  - Deadline for submission is **Sep 10, 11:59 PM ET.**
  - Once open, you have **120 min** to complete the quiz.
  - You may not start the quiz after the deadline has passed.
  - Every 15 minutes you will be prompted to save.
  - **Maintain your own timer from when you start the quiz.**
  - **Click submit before deadline passes. No Exceptions!**



Submit Before Deadline

- When you start the Quiz, you cannot stop the clock.
  - You have 120 minutes to click on submit.
  - You have to keep track of the time yourself.
  - If you don’t click on submit you will not receive a grade.

YOU MUST SUBMIT
WITHIN 120 MINUTES
AND
BEFORE THE DEADLINE

Do not collaborate on quizzes

- In previous semesters, there is always a significant minority who decided to collaborate on quizzes, especially at the semester start and when the team project began.
- **We have to emphasize again that unauthorized collaboration on quizzes is also AIV.**

Quality of Service (QoS)

Quantitatively Measure QoS

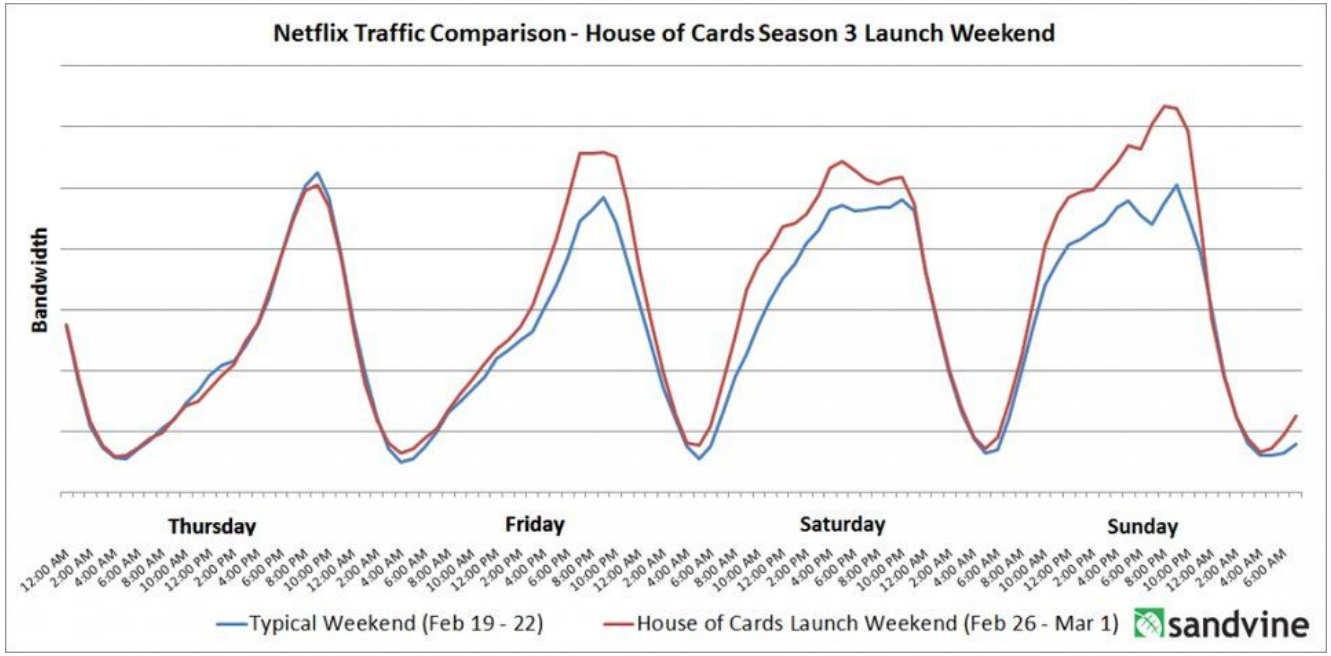
- Performance: Throughput, Latency
  (Very helpful in Project 1 & Team Project)
- Availability: the probability that a system is operational at a given time *(Project 1)*
- Reliability: the probability that a system will produce the correct output

QoS Matters

- Amazon found every 100ms of latency cost them 1% in sales (~$1B)

Traffic patterns in the real-world

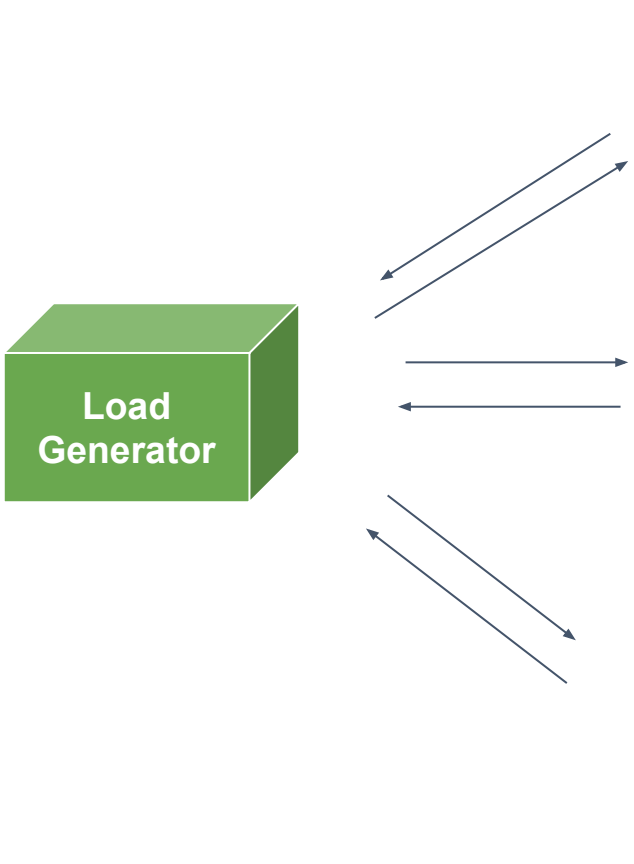
- Daily
- Weekly
- Monthly
- Yearly
- ...



Cloud Comes to the Rescue!

Scaling!

Vertical Scaling



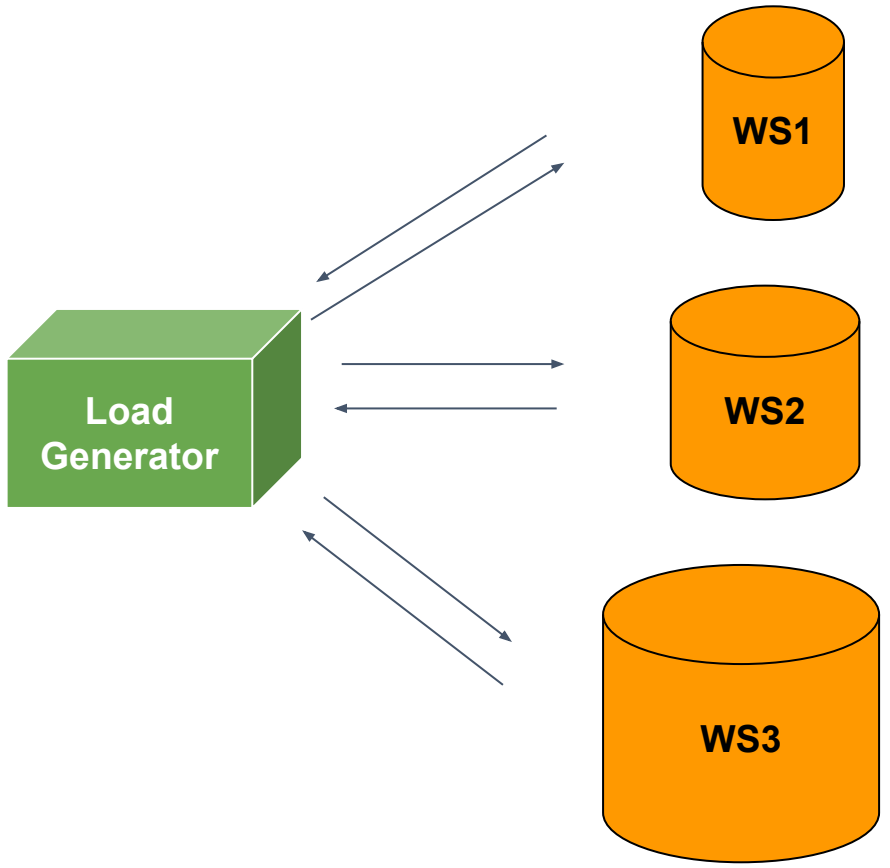
Small Data Center Instance
Serves 100 requests per second

Medium Data Center Instance
Serves 1000 requests per second

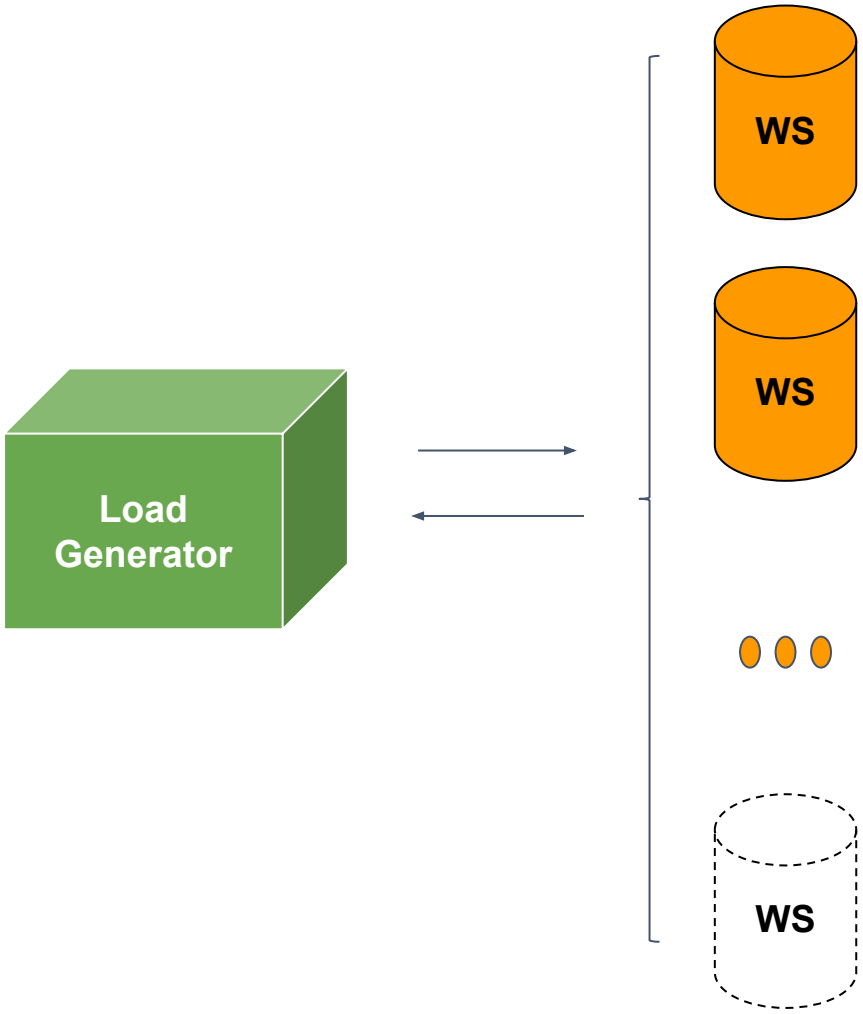
Large Data Center Instance
Serves 10000 requests per second

Vertical Scaling Limitation

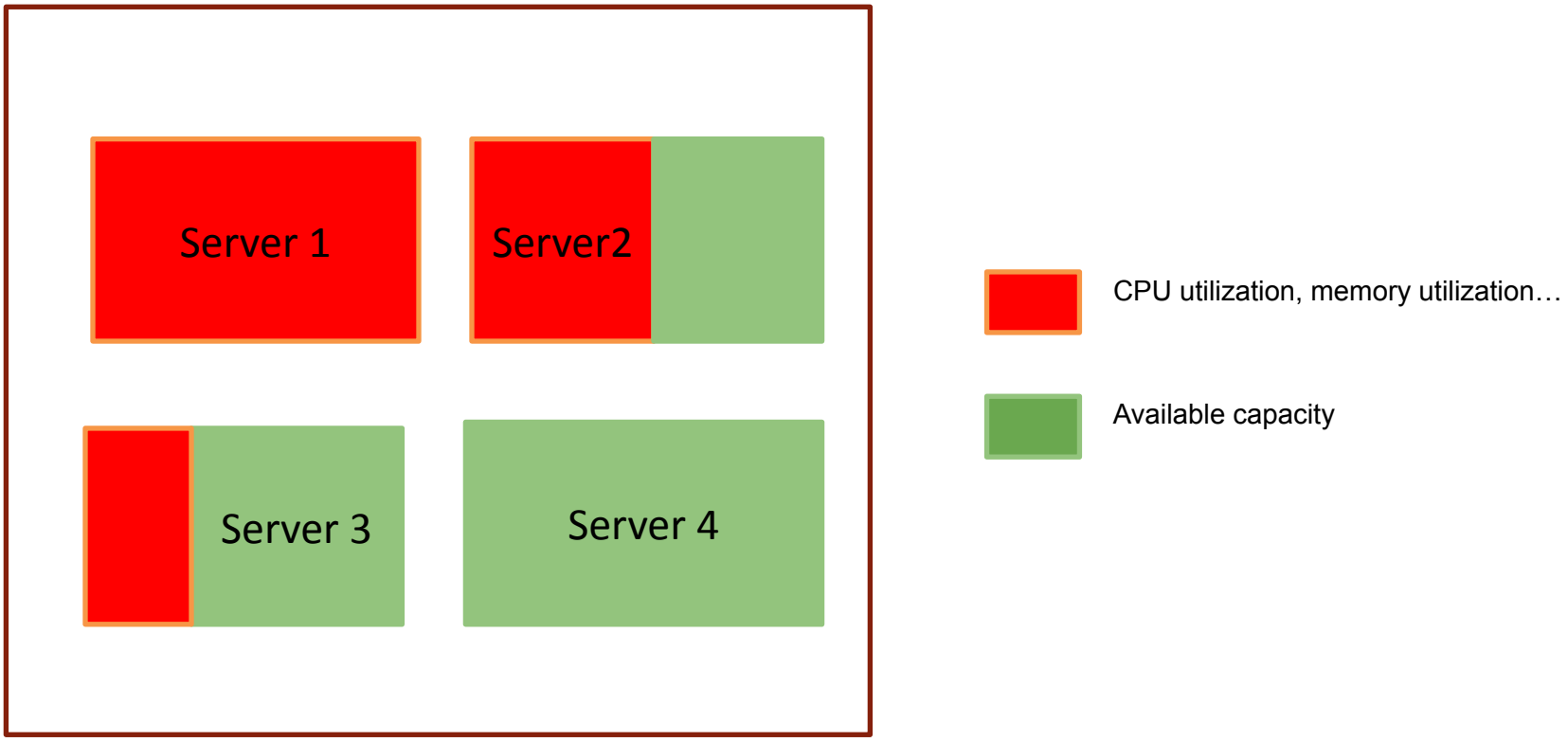
- However, one instance will always have limited resources
- Reboot/Downtime



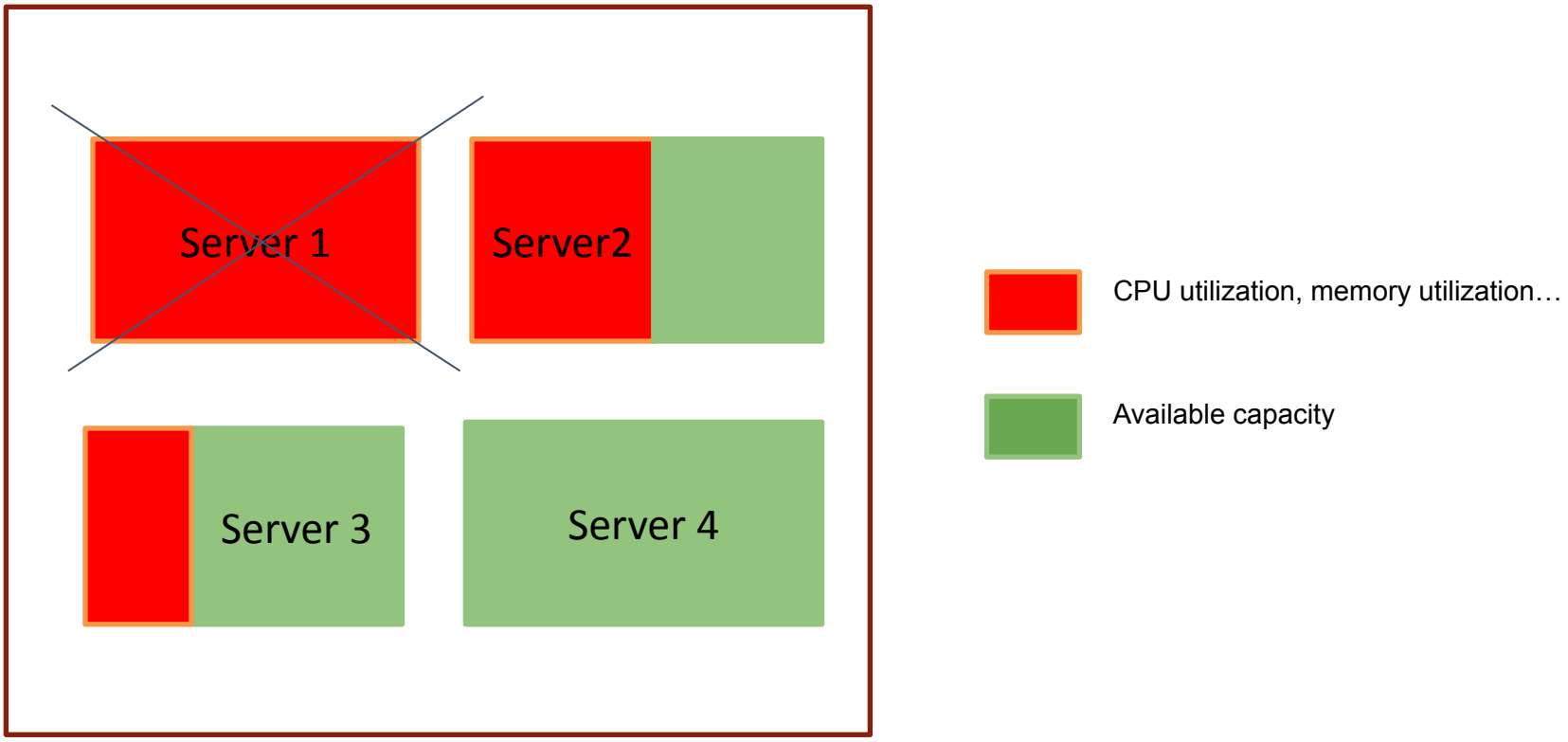
Horizontal Scaling



How do we distribute load?



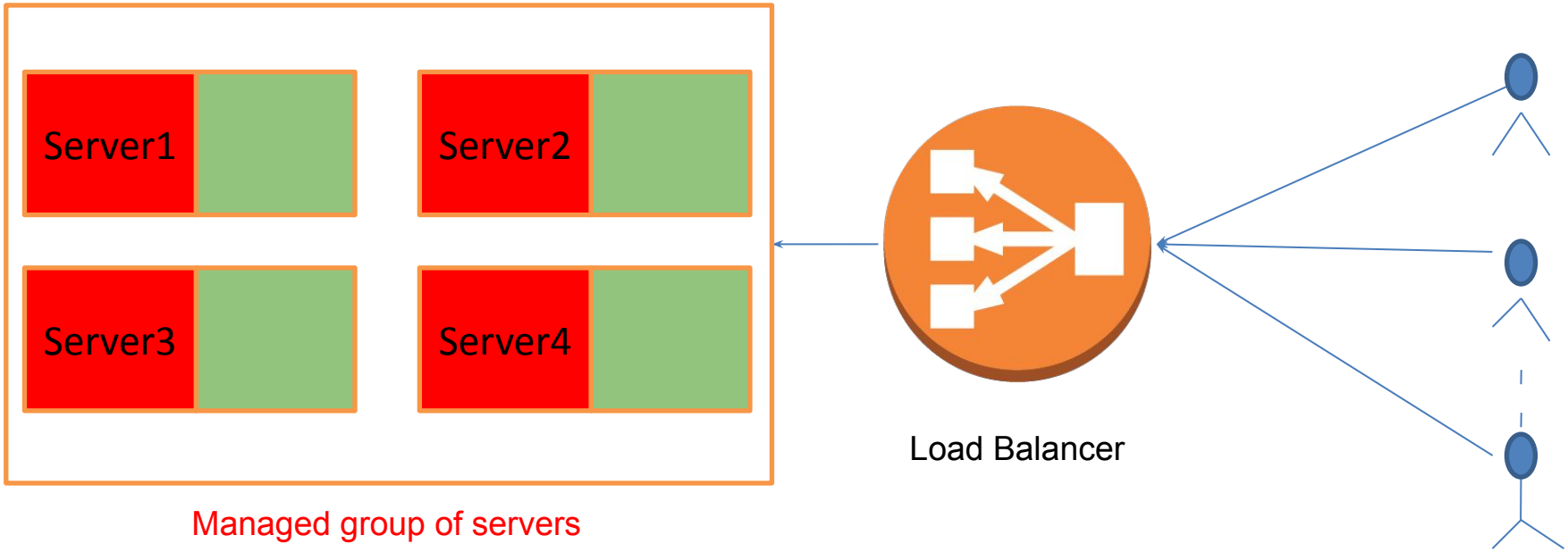
Instance Failure?



What You Need

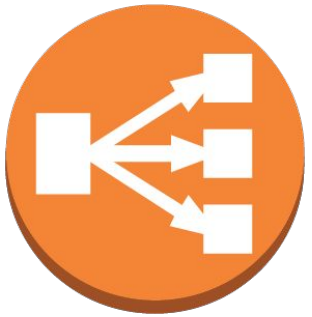
- Make sure that the workload is even on each server
- Do not assign load to servers that are down
- Add/remove servers according to a changing load

How does a cloud service provider help resolve these problems?



Load balancer

- “Evenly” distribute the load
- A simple distribution strategy
  - Round Robin
- Load check
- Health check



Load Balancer

- *What if the Load Balancer becomes the bottleneck?*
  - Elastic Load Balancer (ELB)
    - Scale up based on load
  - Elastic, but it still takes time
    - Require the warm-up process

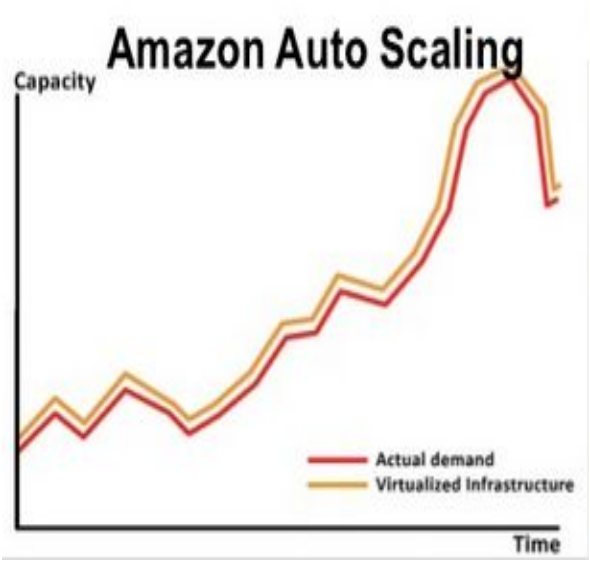
Scaling

Manual Scaling:

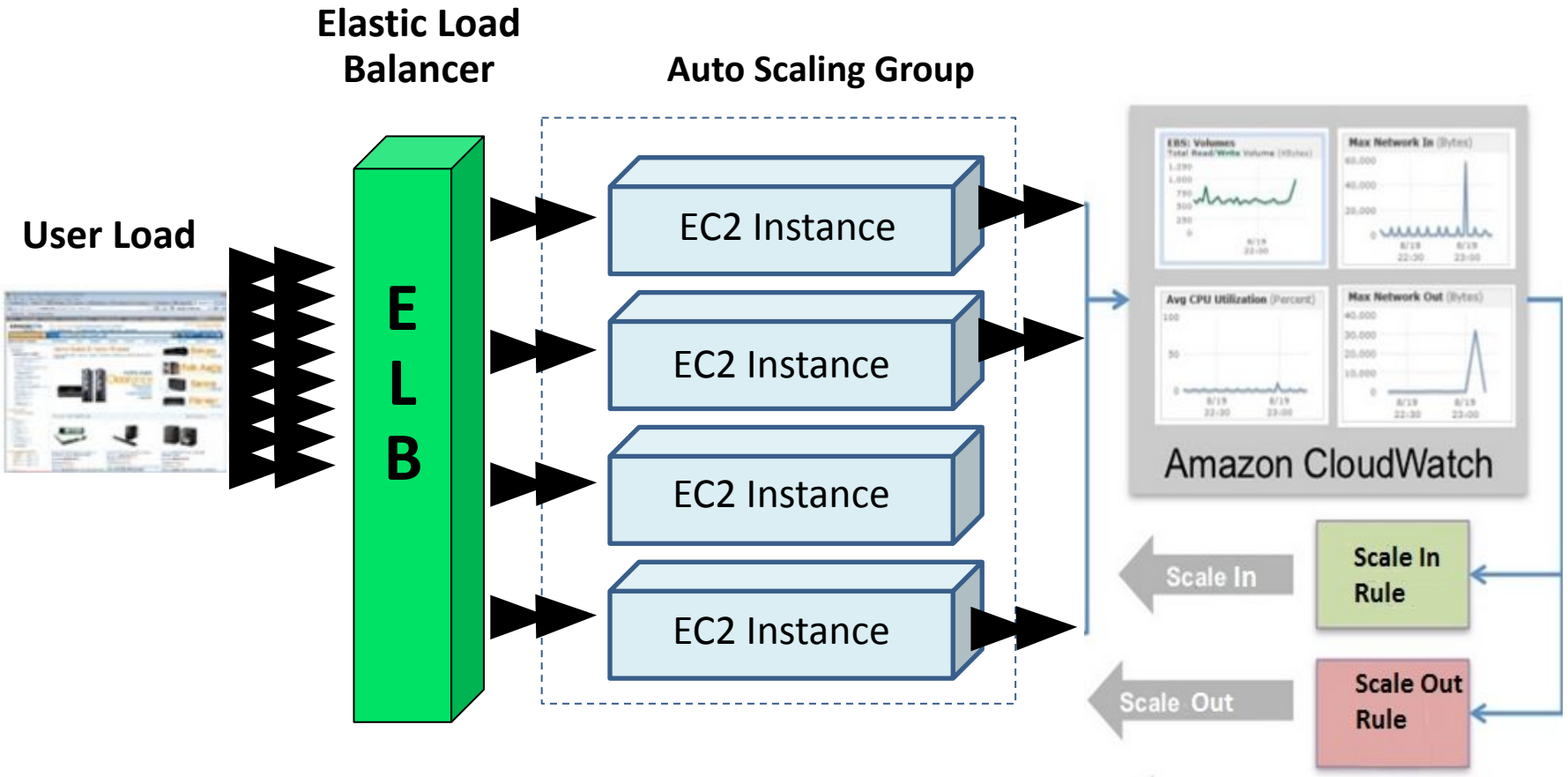
- Tend to lead to over-provisioning and low-utilization
- Tend to lead to insufficient capacity and lose customers
- Expensive on manpower

Autoscaling:

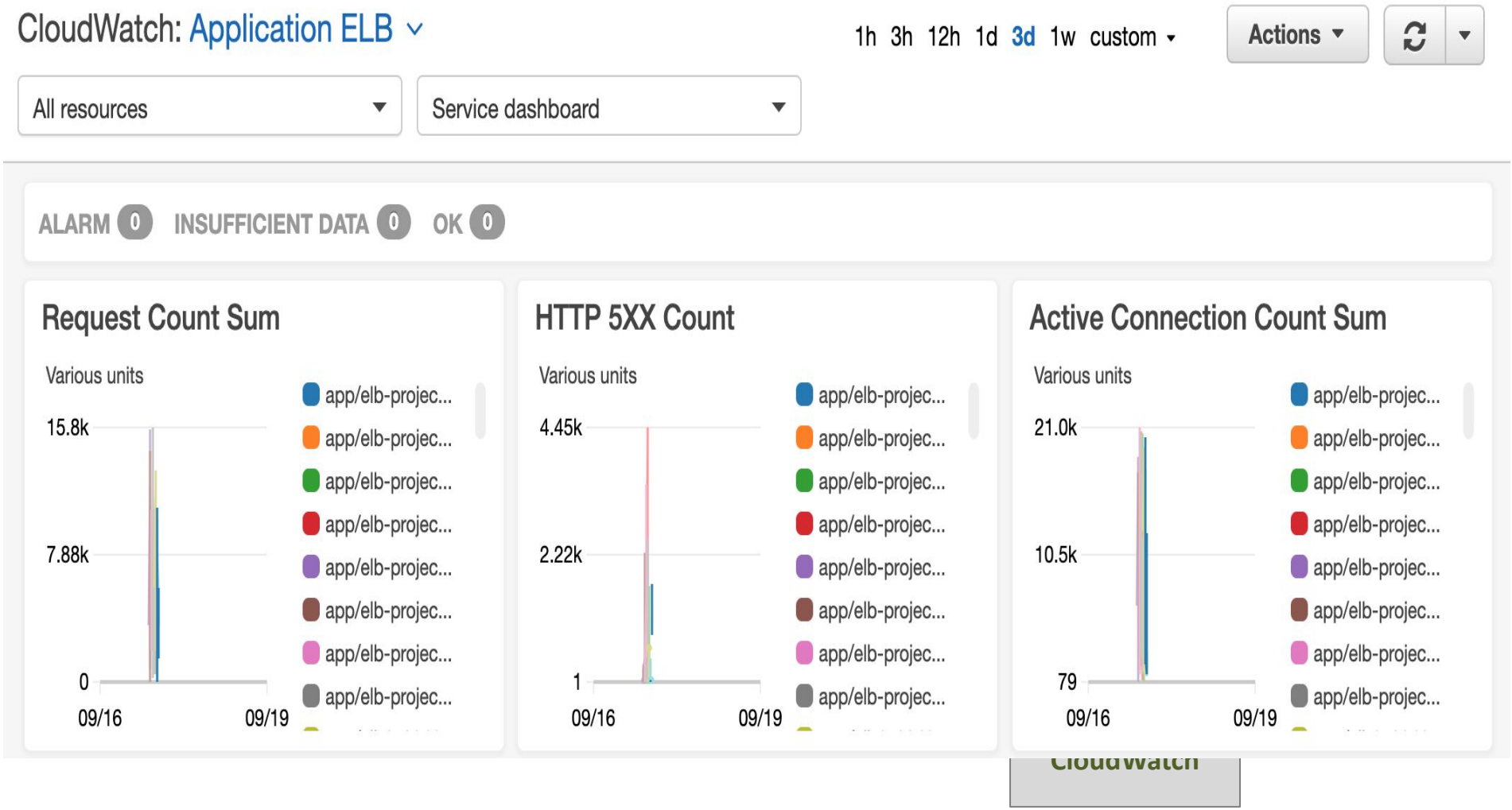
- Automatically adjust the capacity based on metrics and rules
- Save cost



Amazon Auto Scaling Group

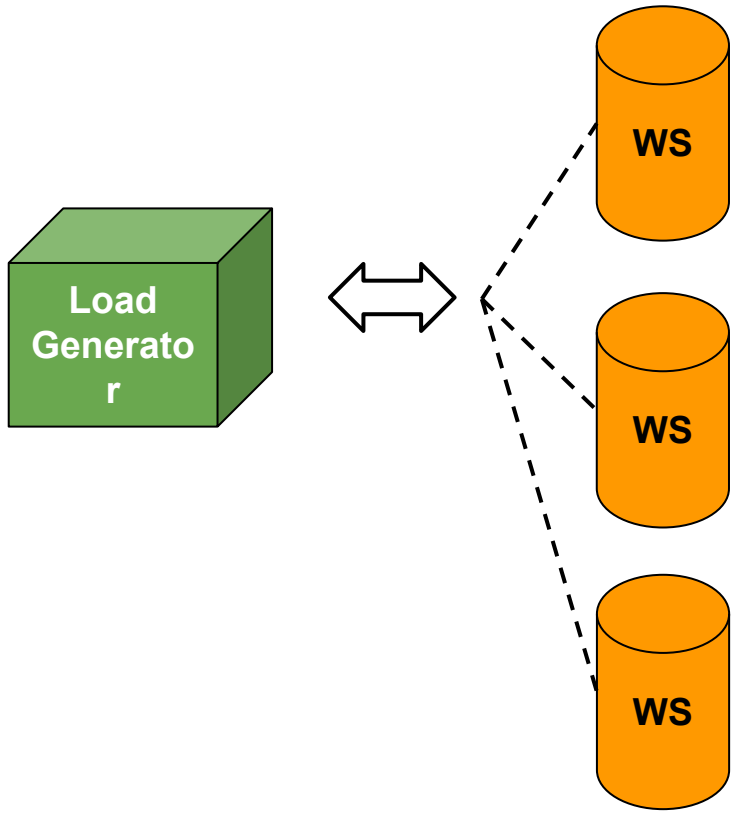


Amazon CloudWatch Alarm



Project 1 Hands-on Tasks

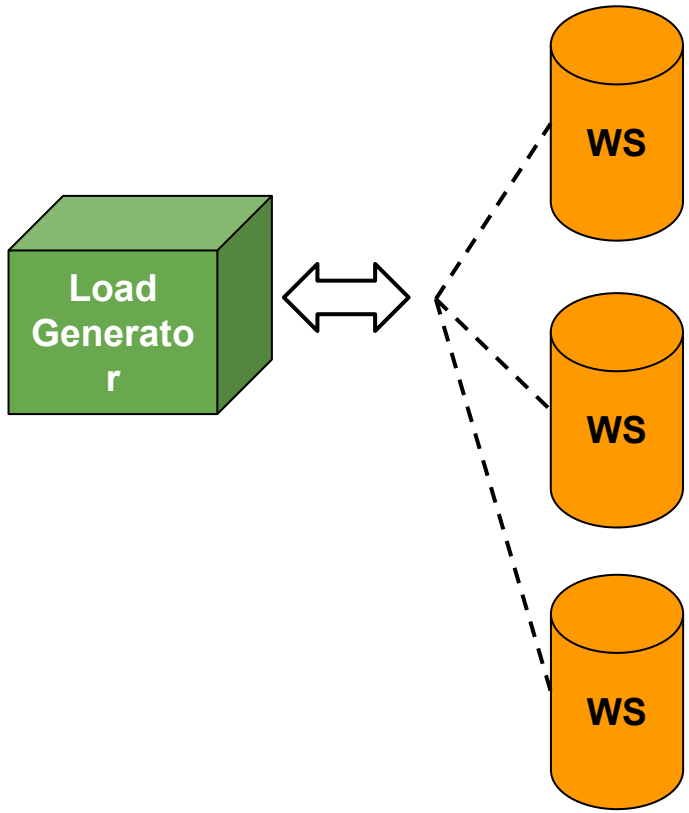
- **Task 1**
  - **AWS Horizontal Scaling**
- **Task 2**
  - **AWS Auto Scaling**
- **Task 3**
  - **AWS Auto Scaling with Terraform**



Project 1 Scaling on AWS

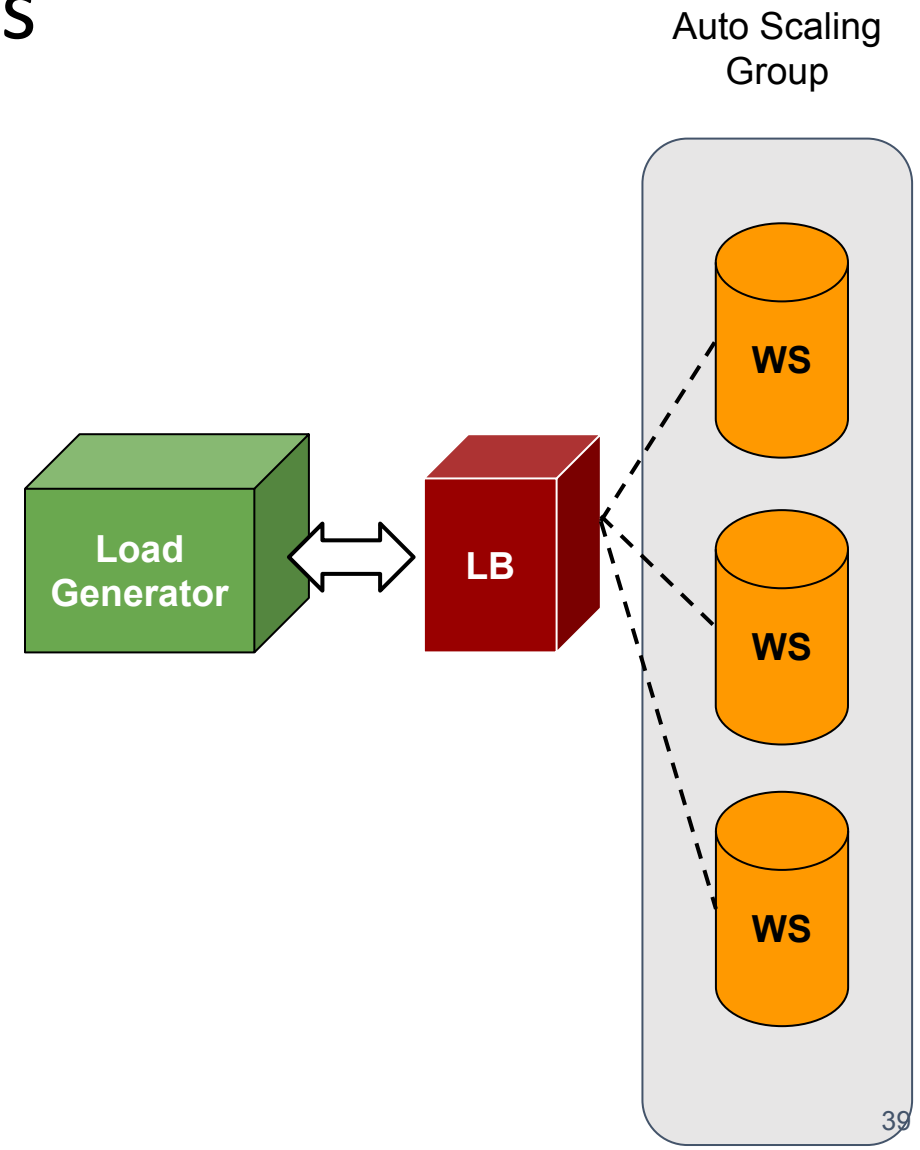
Task 1 - AWS Horizontal Scaling

- Implement Horizontal Scaling in AWS
- Write a program that launches web service instances and ensures that the target total RPS is reached
- Your program should be fully automated: launch LG → submit password → Launch WS → start test → parse log → add more WS...



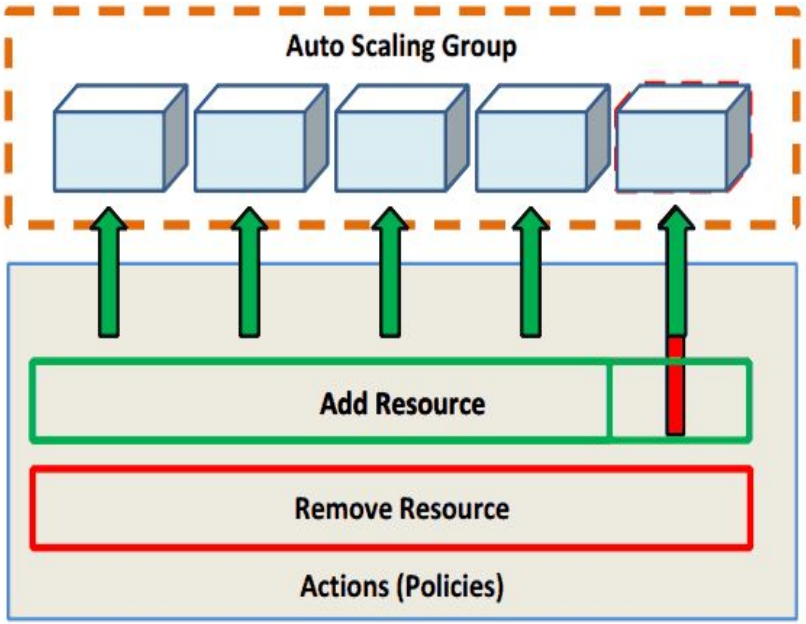
Project 1 Hands-on Tasks

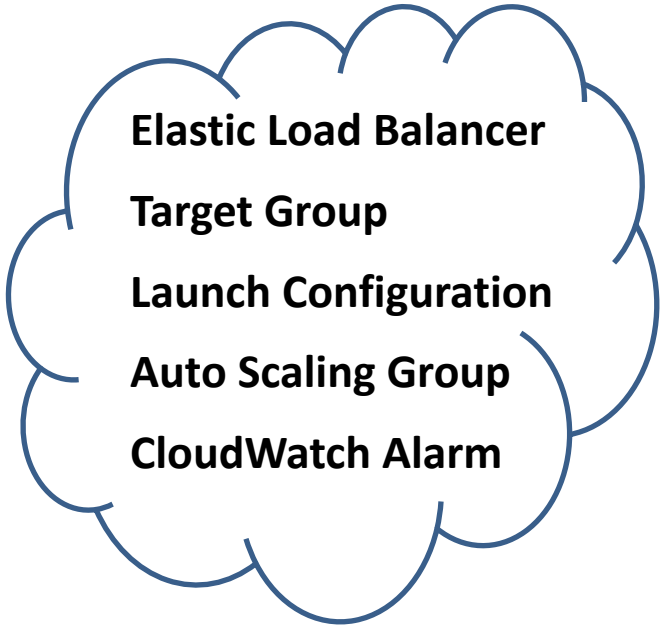
- **Task 1**
  - **AWS Horizontal Scaling**
- **Task 2**
  - **AWS Auto Scaling**
- **Task 3**
  - **AWS Auto Scaling with Terraform**



Project 1 Task 2 AWS Autoscaling

- Programmatically create LG, Application Load Balancer (ALB), Auto-Scaling Group (ASG) with Auto Scaling Policies and launch configuration
- Fine-tune Scale-Out and Scale-In policies
- Your solution also needs to be fault tolerant
- Health configurations are important



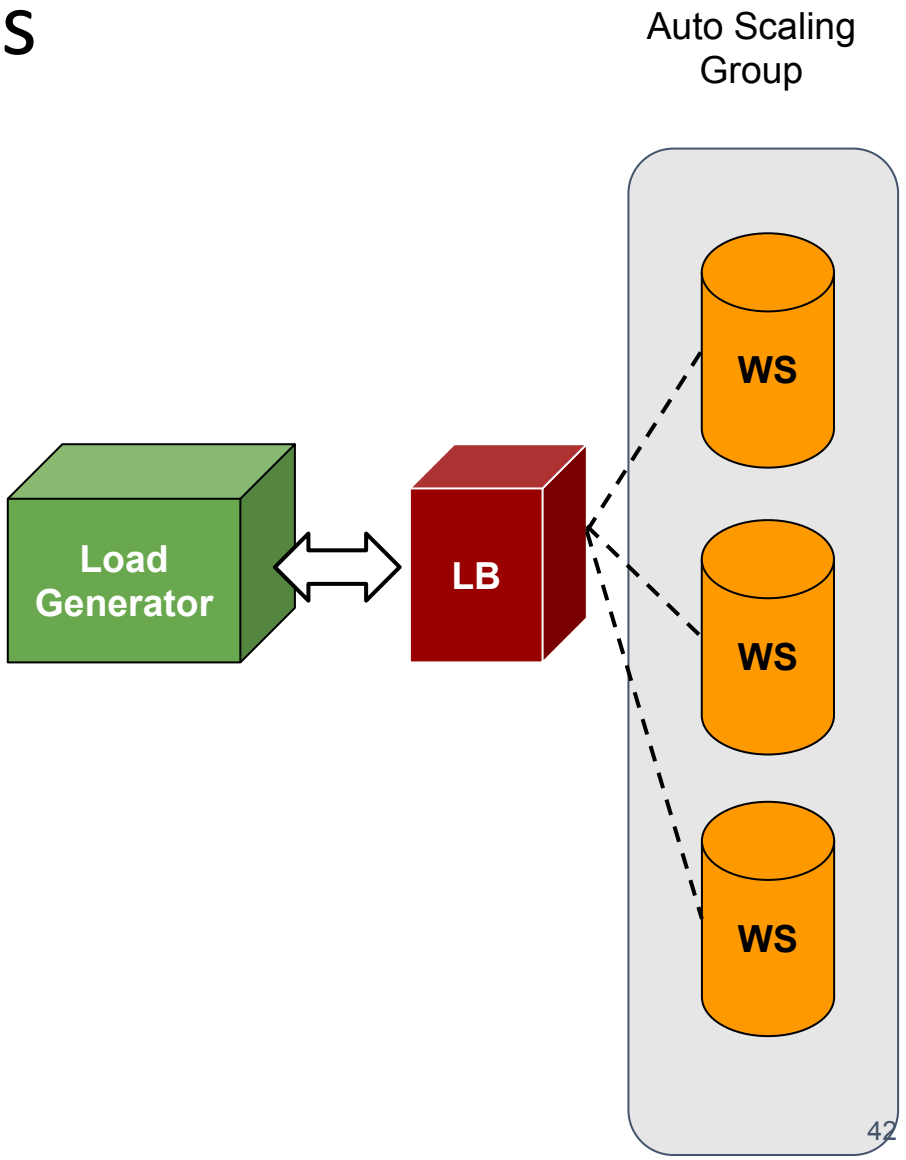


Hints for Project 1 AWS Autoscaling

- Do a dry run via the web console to make sure you understand the workflow
- The Autoscaling test could be expensive!
  - On-demand, charged by per second, do not blindly launch tests
- CloudWatch monitoring is helpful for policy tuning
- Observe and analyze the pattern, experiment with a policy, collect data to verify why it achieved a certain performance, and iterate until you achieve your goal
- **You may need a lot of time to understand the AWS SDK docs**

Project 1 Hands-on Tasks

- **Task 1**
  - **AWS Horizontal Scaling**
- **Task 2**
  - **AWS Auto Scaling**
- **Task 3**
  - **AWS Auto Scaling with Terraform**



Project 1 Task 3 AWS Autoscaling with Terraform

- Read the “Infrastructure as Code” primer to learn about infrastructure automation
- Make sure that **terraform plan** generates the required resources

Penalties for Project 1

Violation	Penalty of the project grade
Spending more than $20 for this project phase on AWS	-10%
Spending more than $40 for this project phase on AWS	-100%
Failing to tag all your resources in either parts (EC2 instances, ELB, ASG) for this project with the tag: key=project, value=vm-scaling	-10%
Submitting your cloud/submission credentials or any Personal Identifiable Information (PII) in your code for grading	-100%
Using instances other than m5.large for Horizontal scaling/Autoscaling on AWS	-100%

Penalties for Project 1 (cont.)

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

Project 1 Workflow

- Launch EC2 instance with the VM Image provided by us
  - The Terraform template to provision EC2 in provided in Project 0
- Complete the Horizontal Scaling Task
- Complete the Autoscaling Task
  - Submit the patterns.pdf file
- Complete the Autoscaling with Terraform Task
- Submit your code for grading
  - Complete the `references` file for citation
  - Execute submitter to submit your code
- Finish Project Reflection (graded) before the deadline
- Finish Project Discussion (graded) within 7 days **after** the project deadline
  - Reply and provide feedback to 3 reflection posts

Grading of Your Projects

- Code submissions are auto-graded
- Scores will be available on the Sail() platform submission tab
  - it may take several minutes for your score to show
  - the submissions table is updated with every submission
- We will grade all the code (both auto and manually graded)
- **Hard to read code** of poor quality will lead to a loss of points during manual grading.
- Lack of **comments**, especially in complicated code, will lead to a loss of points during manual grading.
- Poor **indentation** will lead to a loss of points during manual grading
  - Preface each function with a header that describes what it does
    - Use descriptive variable and function names
    - Use Checkstyle, PEP8, or other tools to check your coding style
- The idea is also NOT to comment every line of code

Reminder: Deadlines

- Sep 10 at 23:59 ET
  - Quiz 1
- Sep 19 at 23:59 ET
  - Project 1 (including Project Reflection)
- Sep 26 at 23:59 ET
  - Project 1 Project Discussion
- ASAP, at the latest 9/13/2021 at 23:59 ET
  - Academic Integrity Course Quiz