

15-319 / 15-619

Cloud Computing

Course Overview 2

September 6, 2022

Agenda

- Course Logistics
- Project 0 Recap
- Quiz 1 Overview
- Project 1 Overview

Accessing the Course

- Course website
 - <https://www.cs.cmu.edu/~msakr/15619-f22/index.html>
- Open Learning Initiative (OLI) Course
 - Access via canvas.cmu.edu
- [The Sail\(\) Platform](#)
 - Choose CMU as the identity provider
 - Cloud 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
- Follow the instructions in the Account Setup Primer
- Wait to receive a Consolidated Billing Request email from Amazon
 - The linking email is sent automatically, waiting time varies
- When you receive the linking email, click the link to verify the linked billing
 - You need to manually accept the linking request
 - Remember to check your **SPAM** folder
 - You won't be able to complete the projects without a linked account.

Google Cloud Platform (GCP) Account

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

Microsoft Azure Account

- ONLY IF YOU HAVEN'T DONE SO ALREADY
- Do **not** use your @andrew.cmu.edu or other CMU issued email address
 - You can use the GCP email you created
- Update the course profile your Azure email address for the invitation code to set up Azure subscription

Piazza

- Piazza is a discussion forum for our learning community.
 - Asking questions itself is a learning activity in this course.
 - Please contribute good questions and answers!
- Check out [Piazza Post Guidelines](#) and Ask Good Technical Questions Primer for best practices when asking questions
- When you encounter a (project-specific) problem:
 - Attempt to solve the problem by yourself (check the hints and grader feedback, search online, etc.)
 - Check current Piazza questions & answers carefully to avoid duplicates
 - Utilize of Piazza's search and tag features
 - Visit TA OHs: OH schedule are posted on Piazza and [Google calendar](#)
 - Create a Piazza post

Piazza Notes

- Ask a public question **IF** possible
- **DON'T** ask a public question about a quiz question
- Read the [Piazza Post Guidelines](#) before asking
- Show your attempt and effort 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 code/commands/error messages
- How to reproduce?
- Expected behavior v.s. Actual behavior
- Environment summary
- What you have tried to solve the issue?

Do not use screenshots to share text-based information

- Sharing code/error logs/ terminal commands using a screenshot makes it very difficult for the readers to consume
- In Piazza, use the backticks (``) to put code block/error

**Please share
code/commands/error
messages in the plain text
format. NEVER share
code/commands/error
messages using screenshots!**

```
19 File "/Users/<username>/wor
20 self.process_show(cmd)
21 File "/Users/<username>/wor
22 self.app_engine.message =
23 File "/Users/<username>/wor
24 f' ...{"." * max_name - t
25 TypeError: unsupported operand type(s)
```

```
$ pycodestyle app_cli.py
d 2 blank lines, found 1
ected spaces around keyword / parameter equals
ected spaces around keyword / parameter equals
ected spaces around keyword / parameter equals
ected spaces around keyword / parameter equals
ng whitespace after ','
ected spaces around keyword / parameter equals
ected spaces around keyword / parameter equals
too long (100 > 79 characters)
ng whitespace around operator
ng whitespace around operator
ected spaces around keyword / parameter equals
ected spaces around keyword / parameter equals
ng whitespace around operator
ndentation is not a multiple of 4
app_cli.py:61:80: E501 line too long (111 > 79 characters)
app_cli.py:65:20: E201 whitespace after '['
app_cli.py:65:23: E202 whitespace before ']'
app_cli.py:66:27: E203 whitespace before ':'
app_cli.py:68:28: E203 whitespace before ':'
app_cli.py:71:25: E205 missing whitespace around operator
```

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.
- **Each time you ask a question, please mindfully attempt to articulate the question. You will receive feedback from the teaching staff on how your question may be better articulated.**
- We provided a template in the Ask Good Technical Questions Primer for you to structure your questions.

Deadlines

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

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

Project 0 Recap

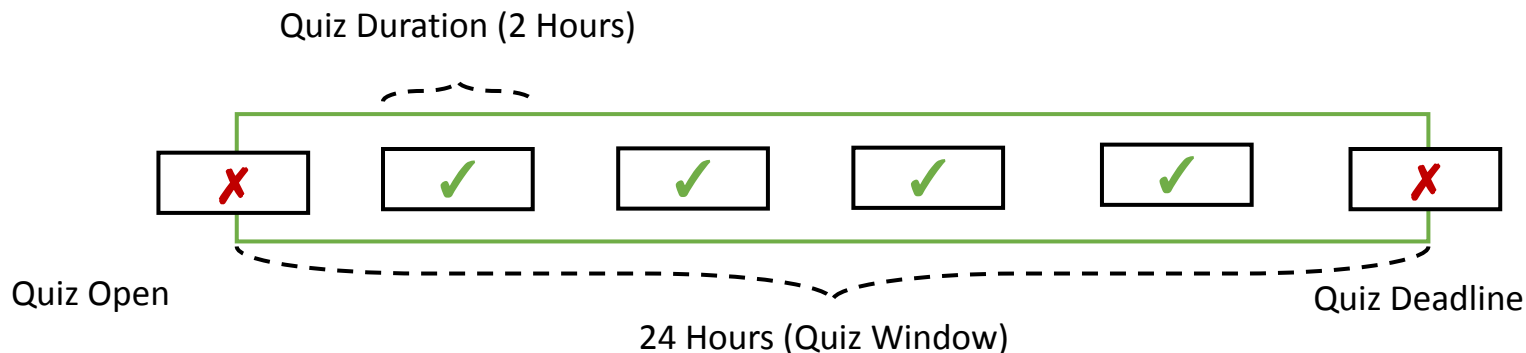
- Learning with the Sail() platform
- Get familiar with AWS, Azure and GCP accounts
 - AWS EC2, S3, CloudWatch
 - Azure Virtual Machines, Azure Storage
 - GCP Compute, GCP Storage
 - Web consoles, CLIs, SDKs
- Basic SSH skills
- Jupyter Notebook primer
- Maven primer (important!)
- Infrastructure as Code (Terraform) primer (important!)
- How to Ask Good Technical Questions primer (important!)

Project 0 Recap

- You experimented with how to complete project-based learning with the solution-feedback cycle
- You experimented with how to provision cloud resources using multiple cloud service providers.
- You experimented 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.)

Quiz 1 Logistics

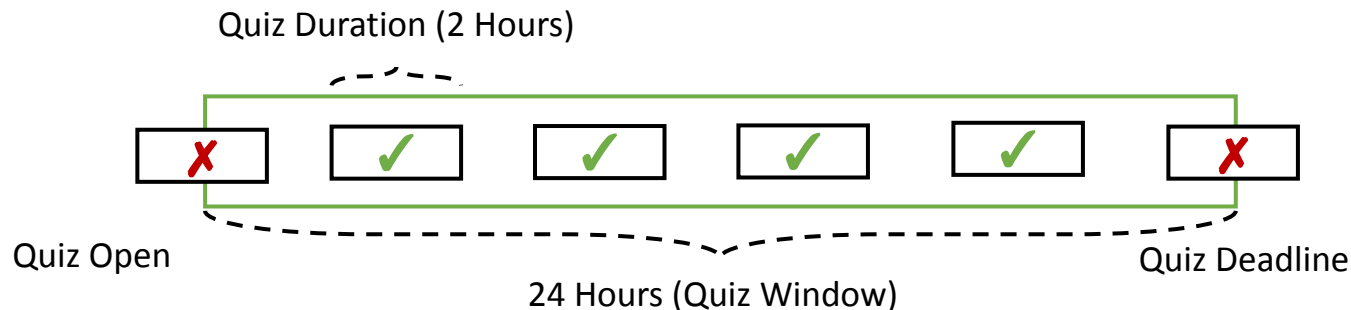
- Quiz 1 will be open for 24 hours, Friday, Sep 9
 - All quizzes are **open-book** tests.
 - Quiz 1 becomes available on **Sep 9, 00:00 AM ET.**
 - Deadline for submission is **Sep 9, 11:59 PM ET.**
 - Once open, you have **120 min** to complete the quiz.
 - You may not start the quiz after the deadline has passed.
 - Every 15 minutes you will be prompted to save.
 - **Maintain your own timer from when you start the quiz.**



Start **BEFORE** Deadline

- After 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, the quiz will be automatically submitted after 2 hours.
 - **You will only receive a grade if the manual/automated submission is before the deadline.**

**THE QUIZ MUST BE SUBMITTED
BEFORE THE DEADLINE**



Quiz 1 Preparation

- Test 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 your ability to perform the stated learning objectives on OLI:

Module 1 / Cloud Computing Overview

1

LEARNING OBJECTIVES

Explain the concept of cloud computing	Briefly understand how computing systems across domains dealt with scale before the cloud	Briefly recall the recent history of cloud computing, illustrating its evolution
List some of the enabling technologies in cloud computing, and discuss their significance	Differentiate cloud service models, such as IaaS, PaaS, and SaaS	Enumerate the different types of clouds, and compare and contrast them
List some of the common cloud providers and their associated cloud stacks	Recall popular cloud use case scenarios	

Module 2 / Economics, Benefits, Risks, Challenges and Solutions

11

LEARNING OBJECTIVES

Discuss some of the advantages and disadvantages of the cloud paradigm	Articulate the economic benefits as well as the issues/risks of the cloud paradigm for users	Articulate the economic benefits as well as the issues/risks of the cloud paradigm for cloud service providers
Define SLAs and SLOs and illustrate their importance in Cloud Computing	Enumerate and explain various threats in cloud security	Enumerate and explain various controls in cloud security

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.**

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 the projects
 - Use the time now to brush up
 - Please read the Maven primer!
- Do not fear Bash/Python scripting, it will make your life easier!

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!
- Tagging Format
 - Key: project/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 gave you an opportunity to learn in Project 0 without affecting your grade
- **We 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!

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

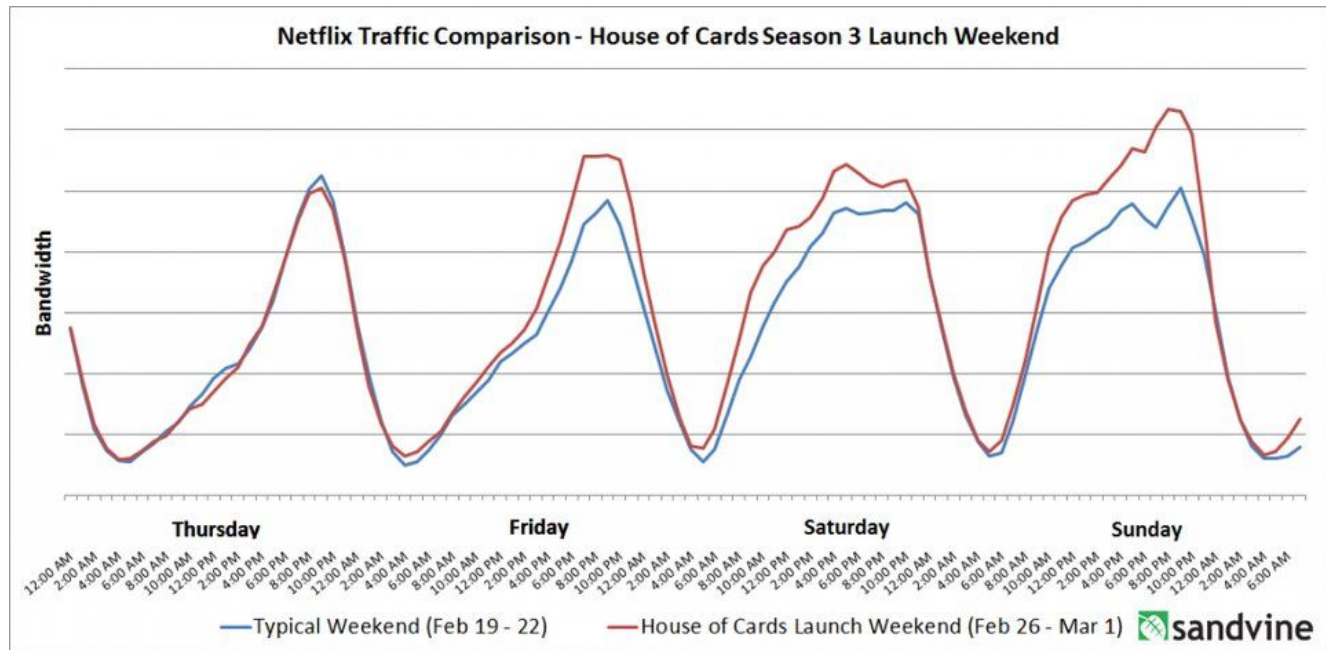
QoS Matters!

- Amazon found every **100ms** of latency cost them **1%** in sales (~\$1B)
- Meta lost about \$65 million in advertising revenue because of a 7-hour outage in 2021



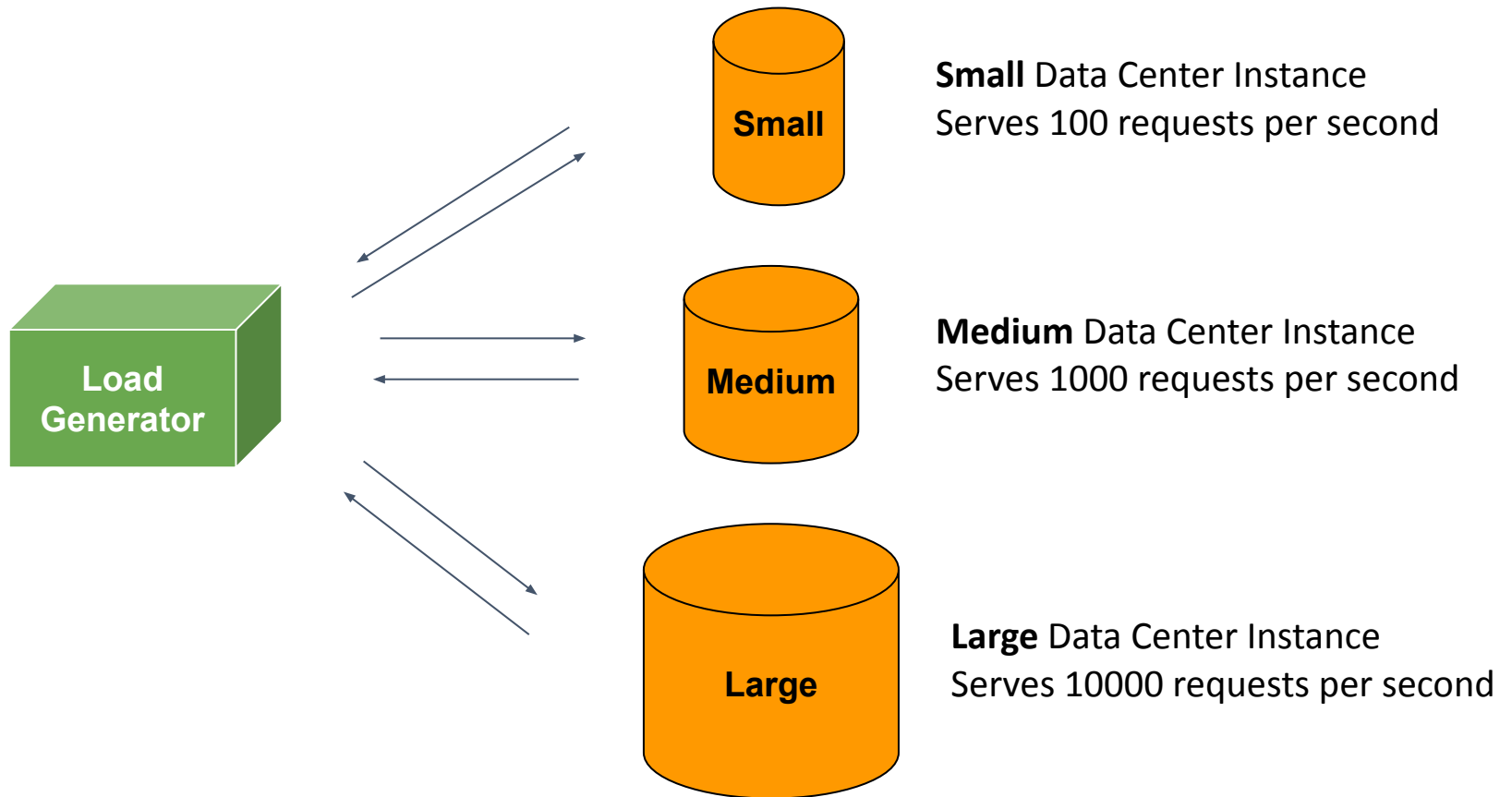
Traffic patterns in the real-world

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



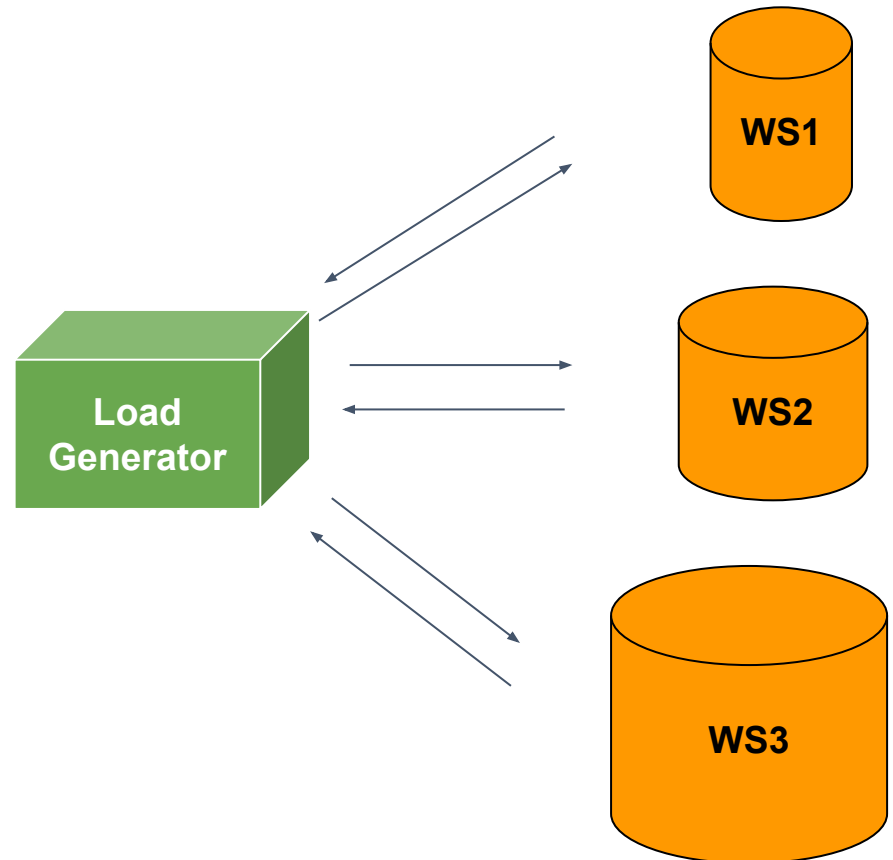
The Ferenstein Wire

Vertical Scaling

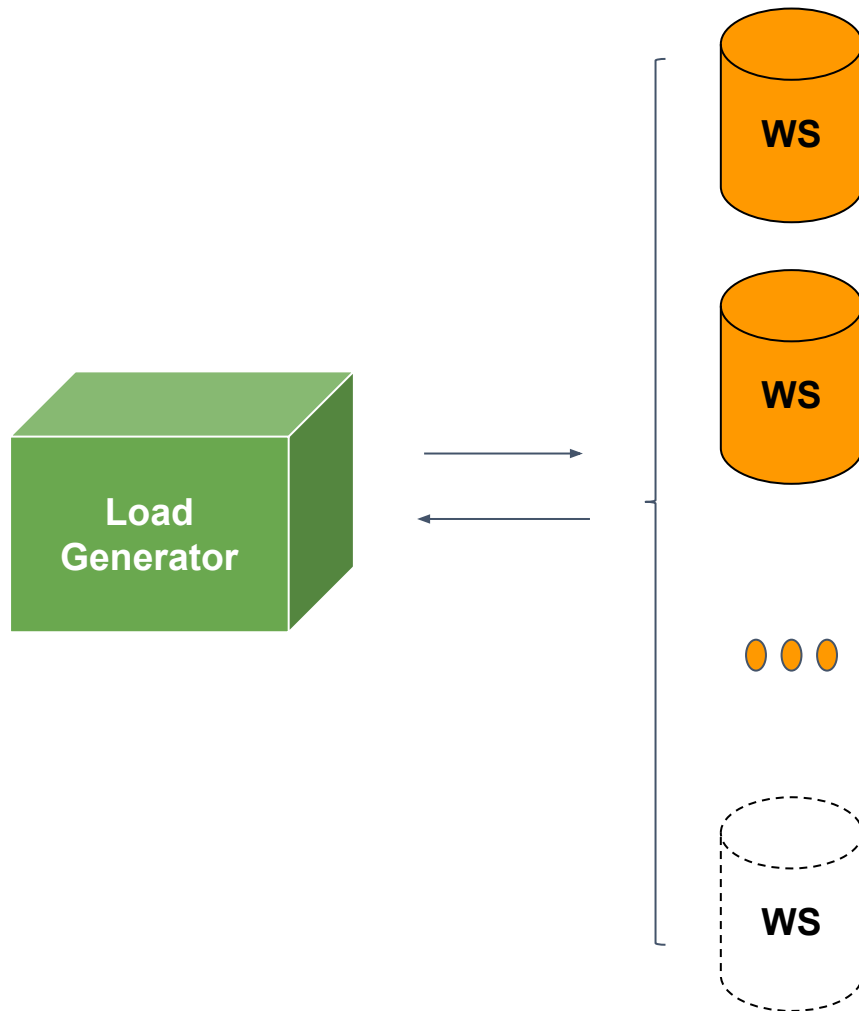


Vertical Scaling Limitation

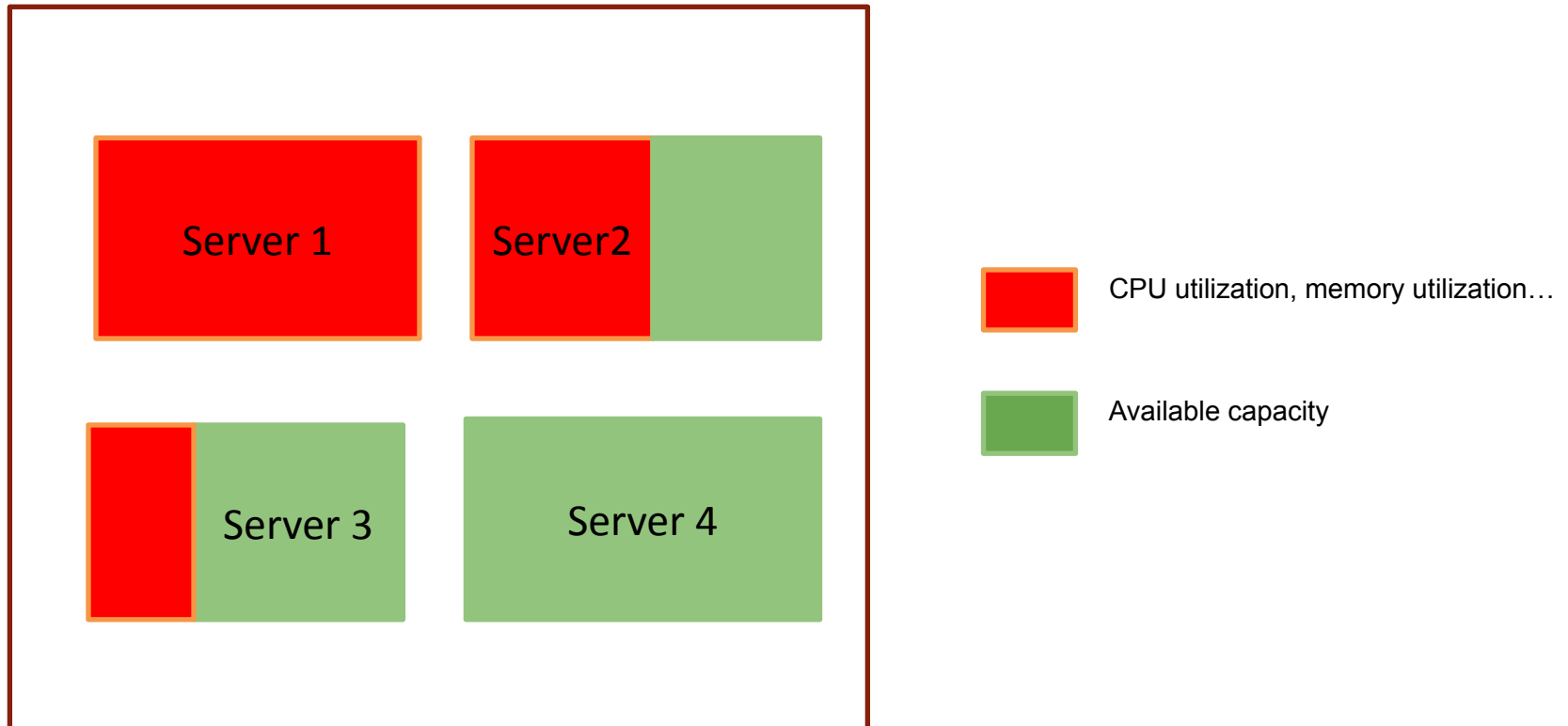
- However, one instance will always have limited resources
- Reboot/Downtime



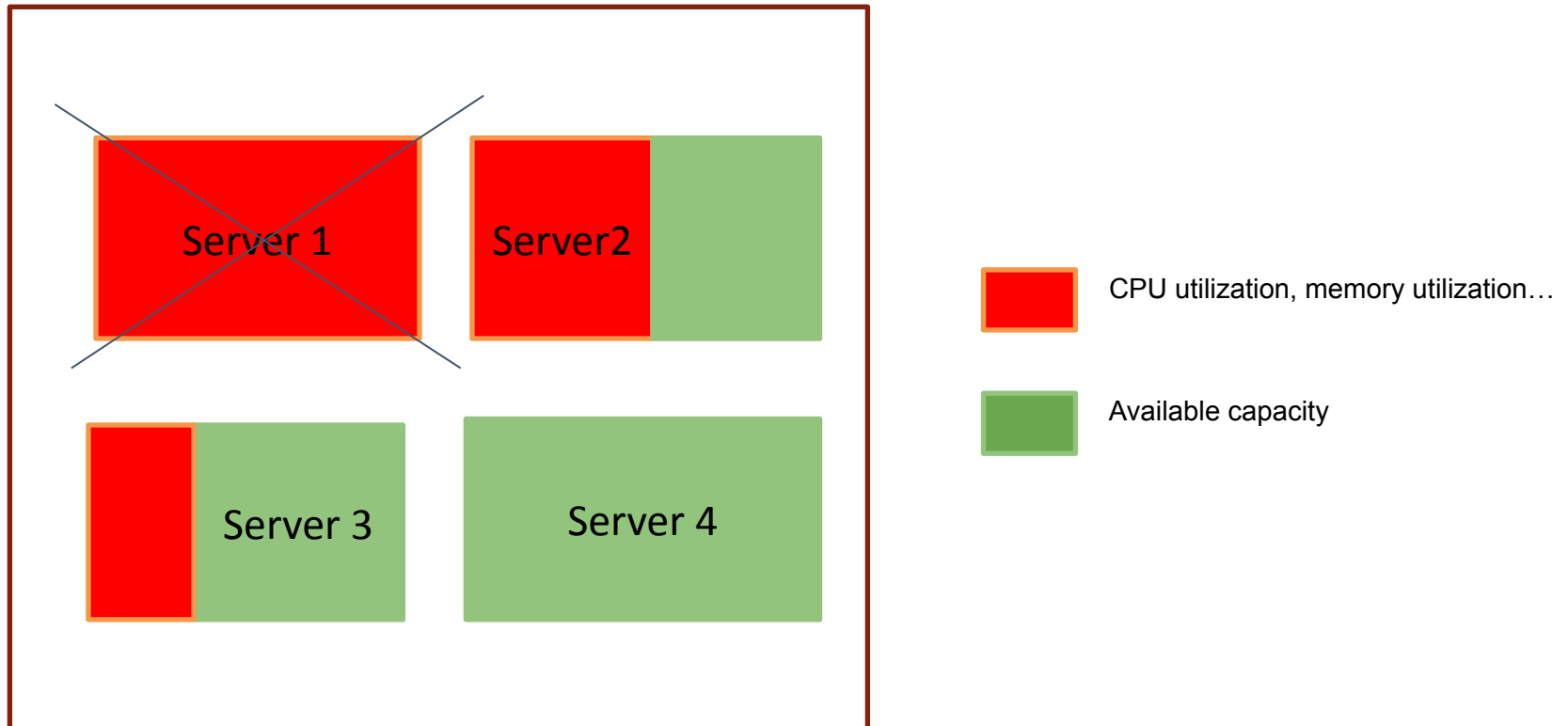
Horizontal Scaling



How do we distribute load?



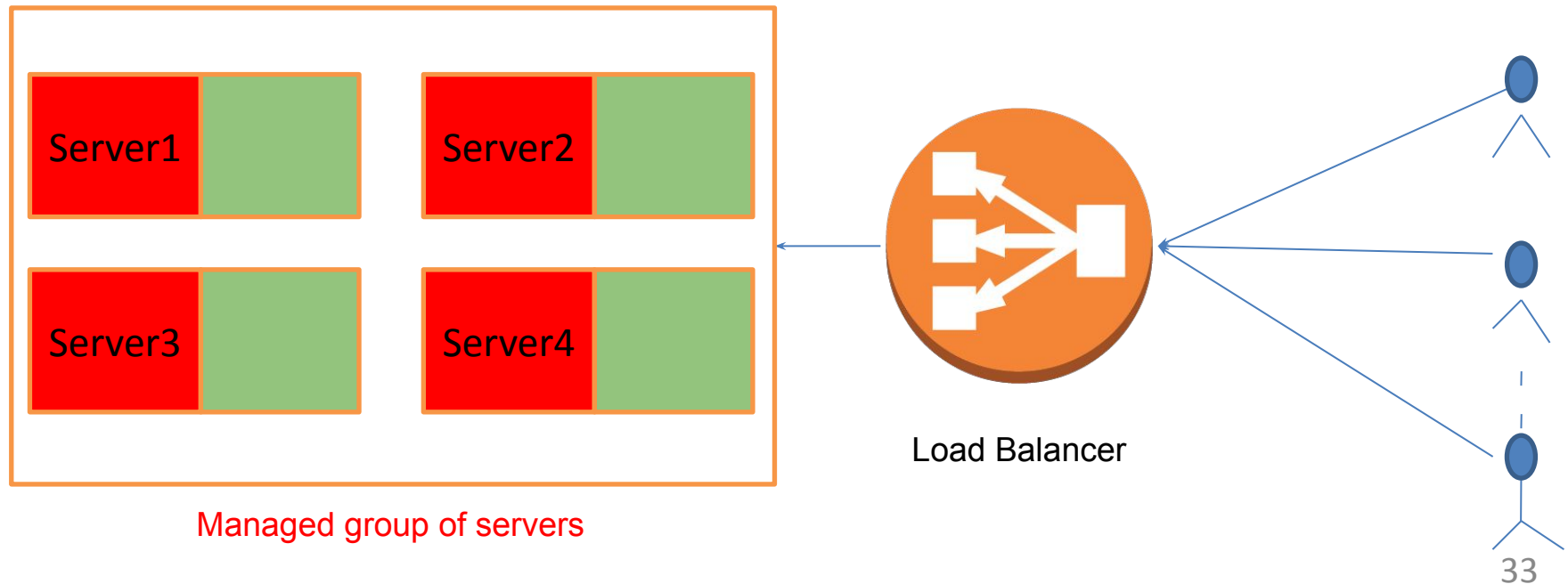
How to handle instance failure?



Make good use of horizontal scaling...

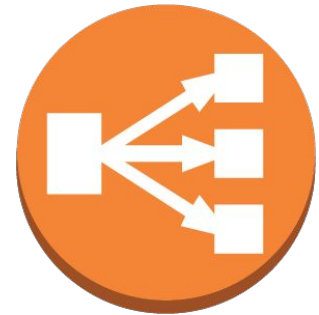
- Make sure that the workload is even on each server
- Do not assign load to servers that are down
- Add/remove servers according to a changing load

How does a cloud service provider help resolve these problems?



Load Balancer

- *“Evenly” distribute the load?*
 - Round Robin distribution
 - Also consider:
 - Load checks
 - Health checks
- *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



Load Balancer

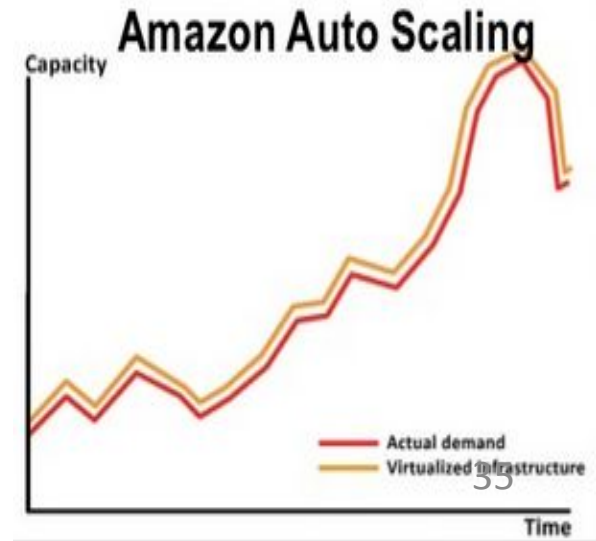
Scaling

Manual Scaling:

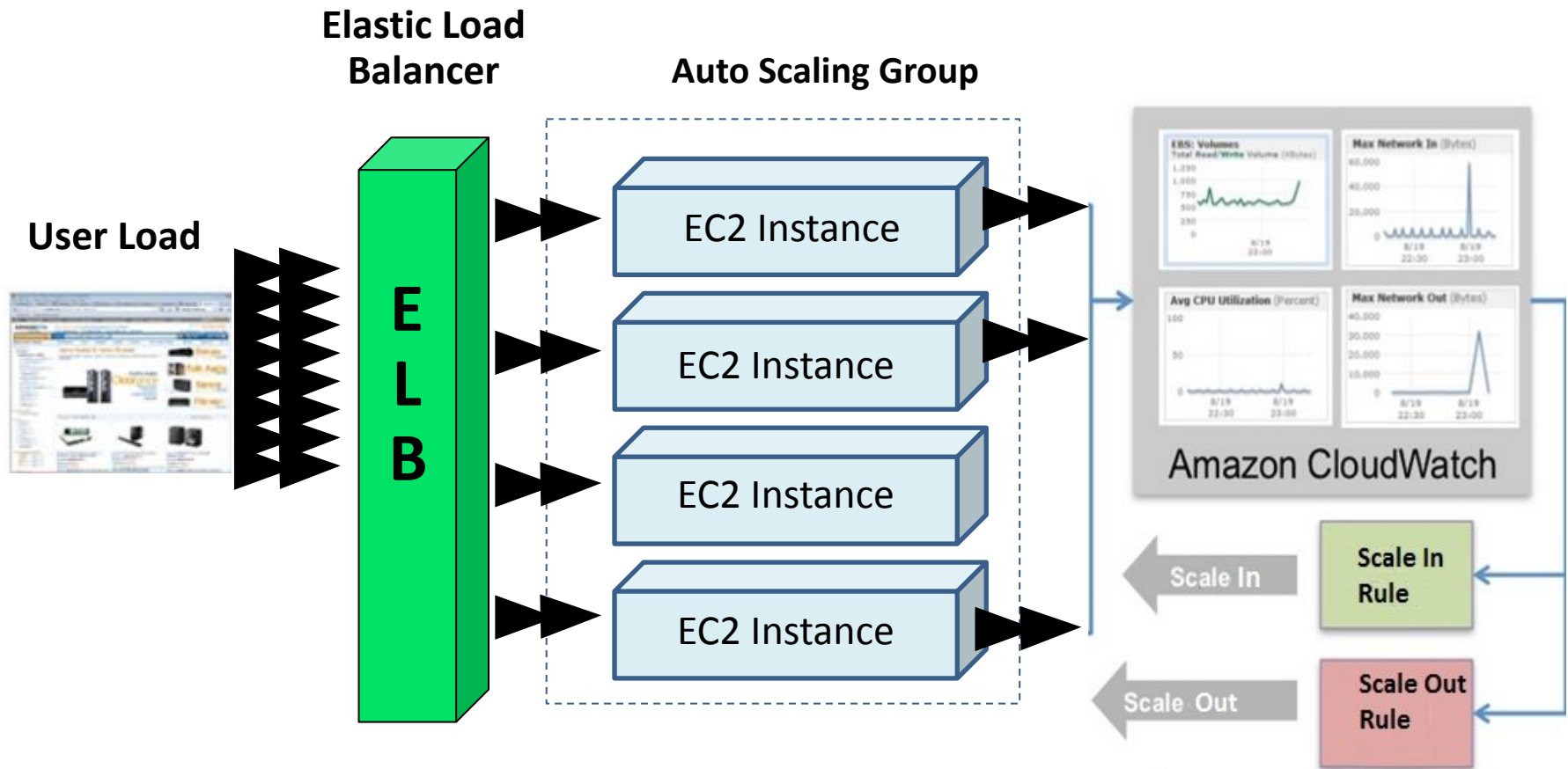
- Tend to lead to over-provisioning and low-utilization
- Tend to lead to insufficient capacity and lose customers
- Expensive on manpower

Autoscaling:

- Automatically adjust the capacity based on metrics and rules

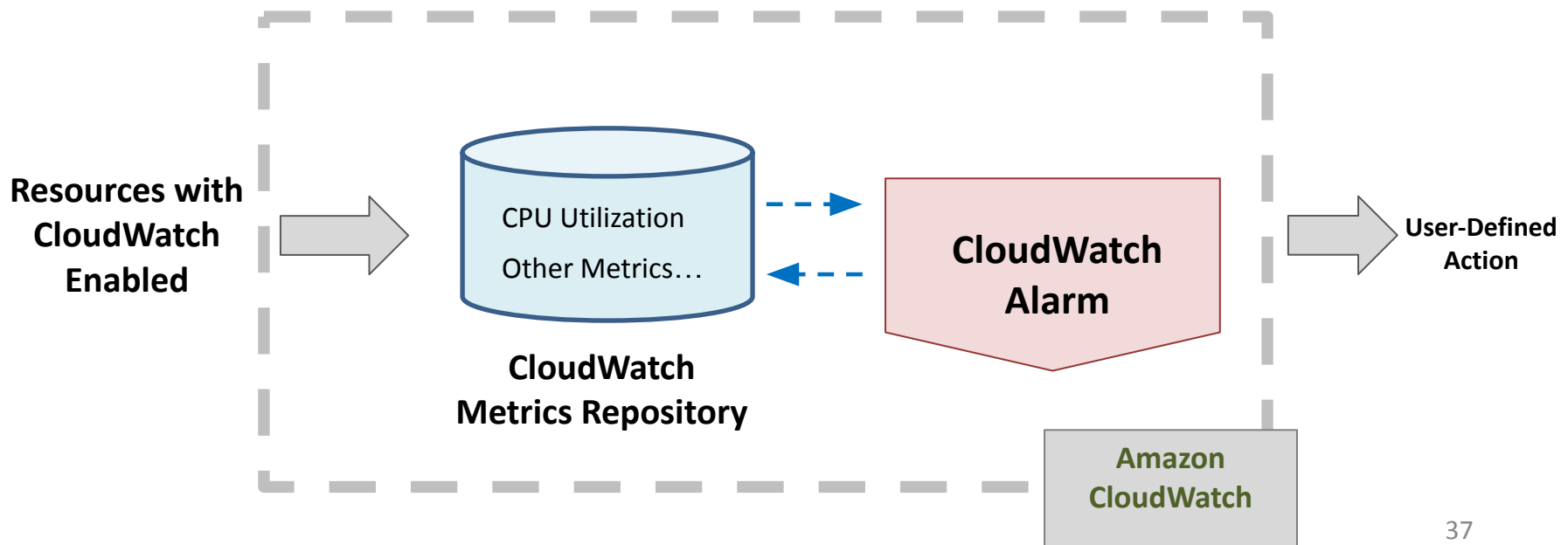


Amazon Auto Scaling Group



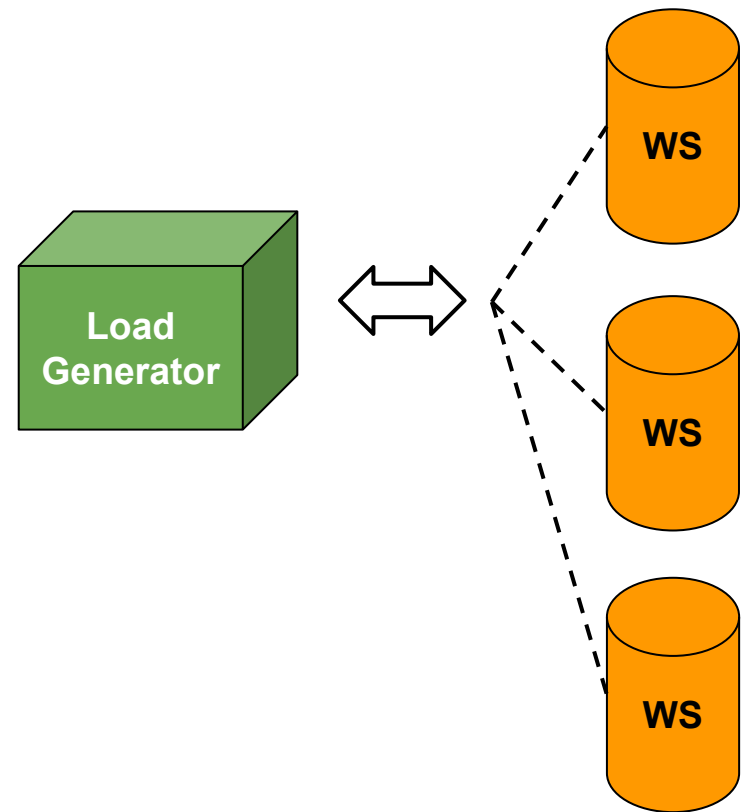
Amazon CloudWatch Alarm

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



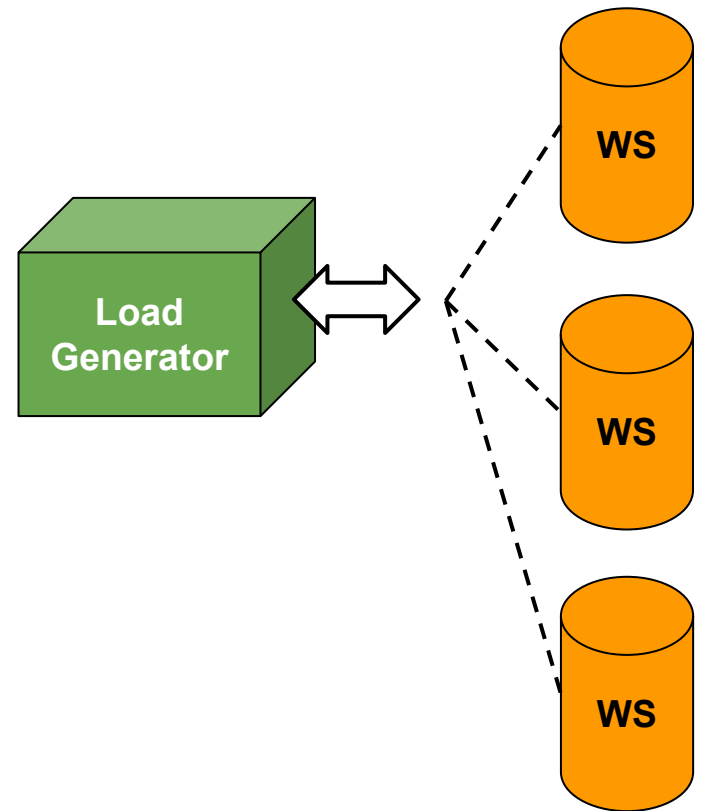
Project 1 Hands-on Tasks

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



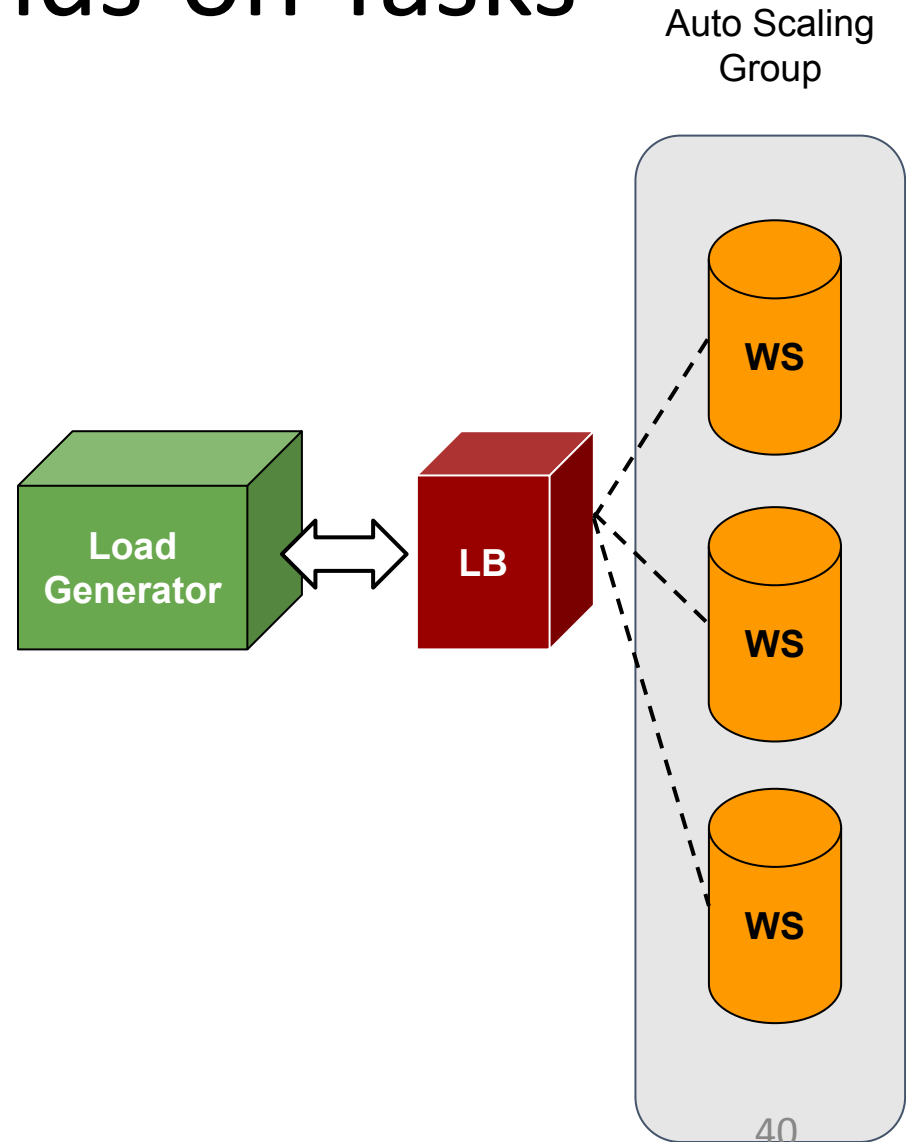
Project 1 Task 1: Horizontal Scaling on AWS

- 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...



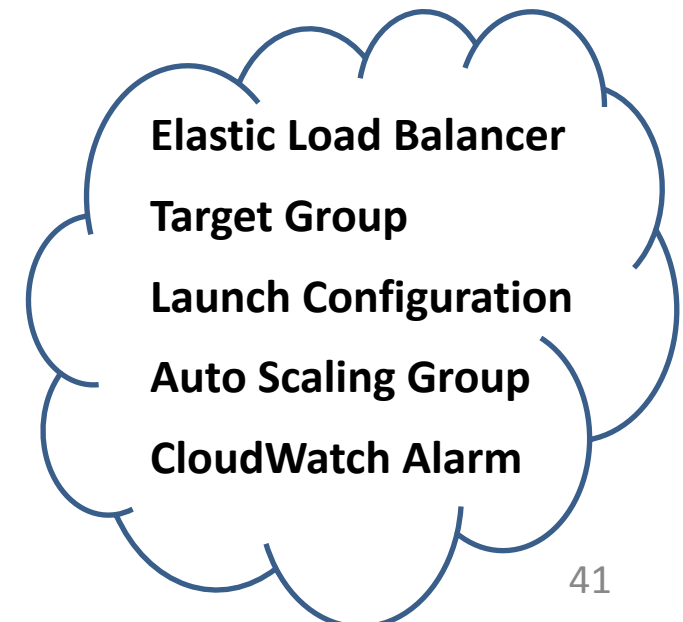
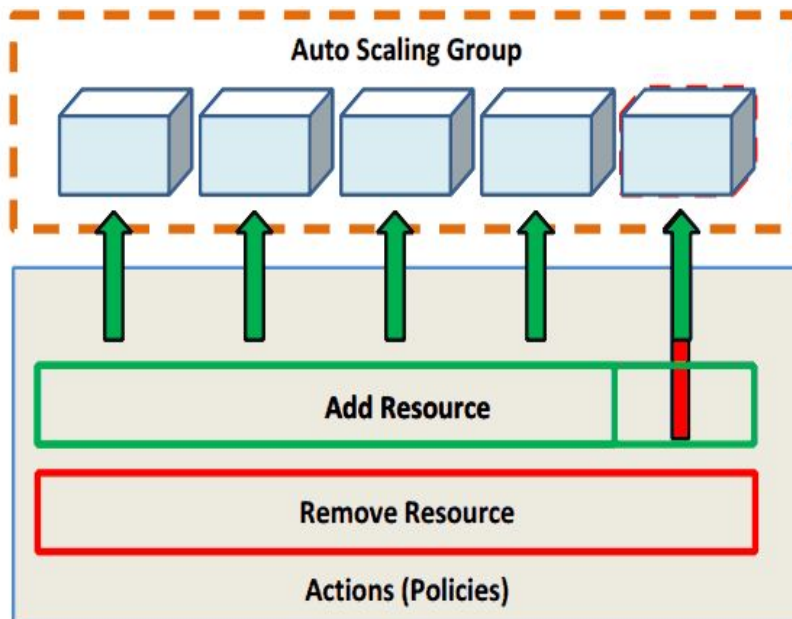
Project 1 Hands-on Tasks

- 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, Load Balancer (ELB), Auto-Scaling Group (ASG) with Auto Scaling Policies and Launch Templates
- Fine-tune Scale-Out and Scale-In policies
- Your solution also needs to be fault tolerant
- Health configurations are important



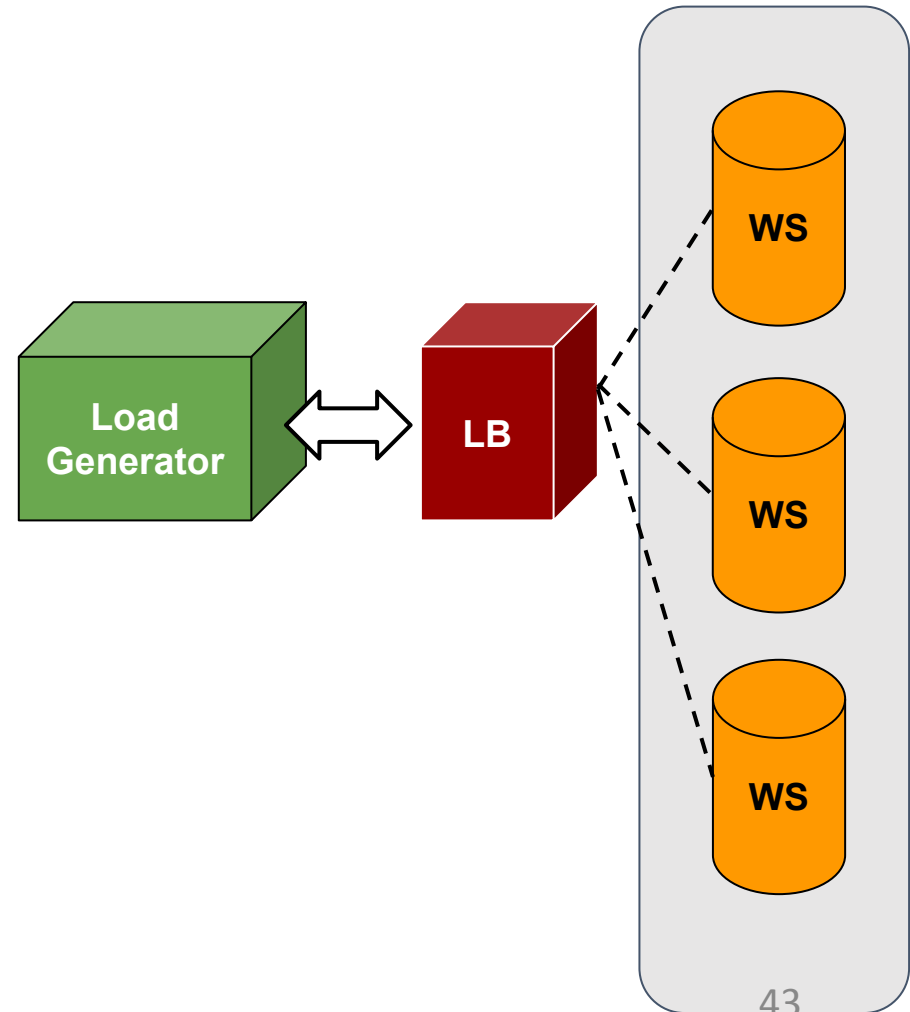
Hints for Project 1 Task 2

- 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 documentations**

Project 1 Hands-on Tasks

Auto Scaling
Group

- 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

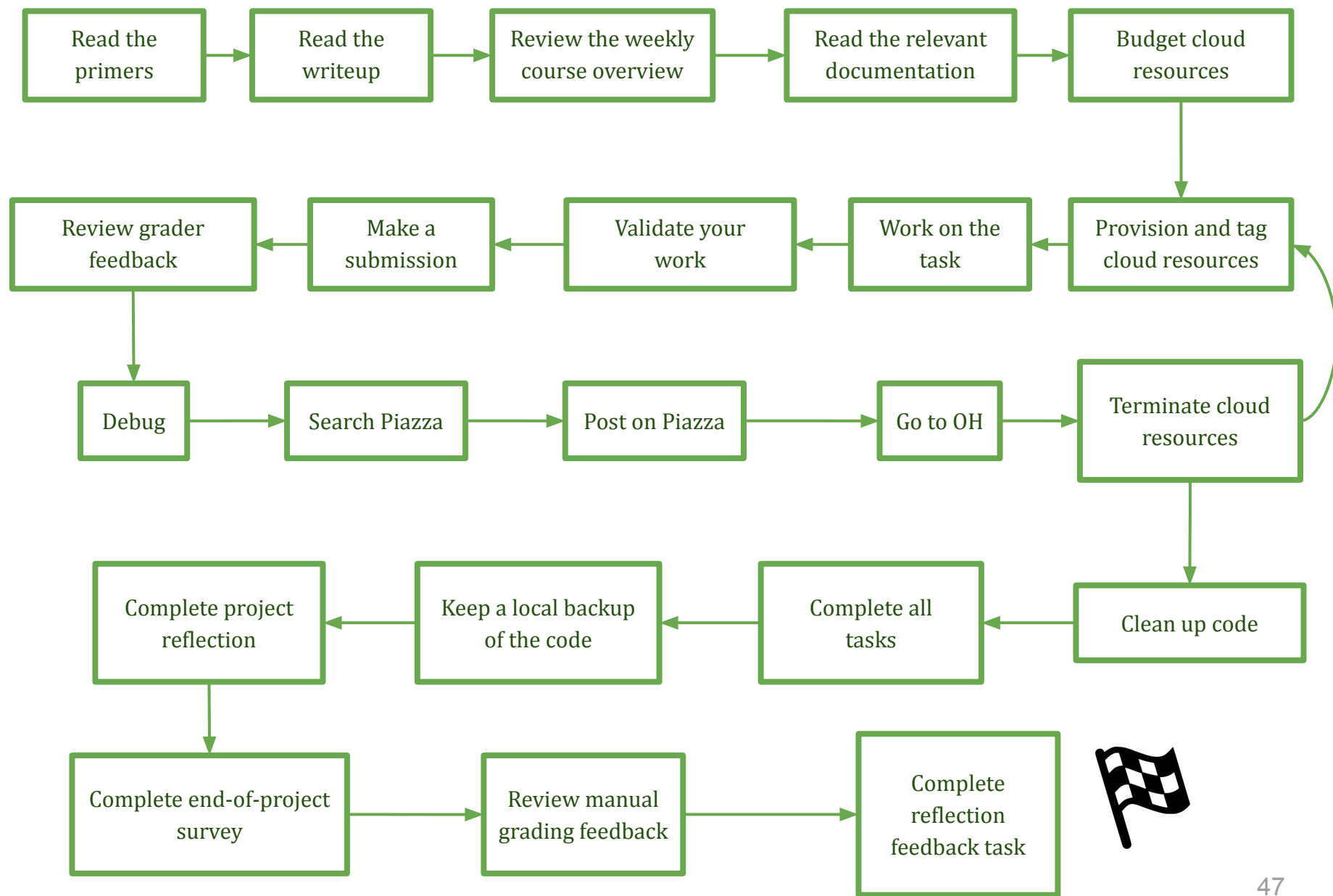
- Read P0. Infrastructure as Code Primer
- Make sure that **terraform plan** generates the required resources
- Consider trying task 3 before spending time fine-tuning your policies for task 2 as task 3 usually takes less time

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 Deliverables

- 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 on your student VM 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)

Manual Grading of Your Projects

- **Hard to read code** of poor quality will lead to a loss of points during manual grading.
- **Poor indentation** 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.
 - The idea is also NOT to comment every line of code!
- **General advice:**
 - Preface each function with a header that describes what it does
 - Use descriptive variable and function names
 - Utilize Checkstyle, PEP8, or other tools to check your coding style

Reminder: Deadlines

- **Sep 9** at 23:59 ET
 - Quiz 1
- **Sep 18** at 23:59 ET
 - Project 1 (including Project Reflection)
- **Sep 25** at 23:59 ET
 - Project 1 Project Discussion
- **ASAP**
 - Academic Integrity Course Quiz