

# 15-319 / 15-619

# Cloud Computing

Recitation 6

February 19<sup>th</sup>, 2019

# Administrative - OH & Piazza

- Make use of office hours
  - Make sure that you are able to describe the problem and what you have tried so far
  - [Piazza Course Staff](#)
  - [Google calendar in ET](#)
  - [Google calendar in PT](#)
- Suggestions for using Piazza
  - Contribute questions and answers
  - Read the Piazza Post Guidelines ([@6](#)) before asking questions
  - Read Piazza questions & answers carefully to avoid duplicates
  - Don't ask a public question about a quiz question
  - Try to ask a **public** question if possible

# Administrative - Cloud spending

- Suggestion on cloud service usage
  - Monitor AWS expenses regularly
  - Always do the cost calculation before launching services
  - Terminate your instances when not in use
  - Stopped instances have EBS costs (\$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
- **If your account is compromised, contact the CSP immediately, follow their instructions and post privately on Piazza.**

# Reflection of Last Week

- Conceptual content on OLI
  - Modules 7, 8 & 9 and Quiz 4
- Project theme - **Containers: Docker and Kubernetes**
  - **Docker Intro / Profile Service**
    - Develop a Dockerfile using a provided Spring application
  - **Deploying the Profile Service using GCR and GKE**
    - k8s YAML files, deploying applications to a remote cluster
  - **Intro to Helm Charts / Deploying MySQL**
    - Working with Helm / Tiller, releasing applications using charts
  - **WeCloud Chat Microservices**
    - Deploy the Login and Group Chat services to implement a subset of the WeChat microservices architecture
  - **Autoscaling, Multi-Cloud and Fault Tolerance**
    - Replicate the Profile and Login services to Azure. Use HorizontalPodAutoscalers to scale based on CPU utilization.

# This Week

- **Code Review - Project 2.1**
  - Due on Wednesday, Feb 20th, 2019, 11:59PM ET
- **Quiz 5 (OLI Modules 10, 11 & 12)**
  - Due on Friday, Feb 22th, 2019, 11:59PM ET
- **Project 2.3**
  - Due on Sunday, Feb 24th, 2019, 11:59PM ET
  
- **Primers released this week**
  - Team Project - Profiling a Cloud Service
  - P3.1 - Storage I/O benchmarking
  - P3.1 - NoSQL Primer
  - P3.1 - HBase Basics

# 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
    - Web tier: web framework performance
    - Storage tier: deploy and optimize MySQL and HBase
    - Extract, Transform and Load (ETL)
- Create a new AWS account only for the team project

## 15-319 Students:

- You are allowed to participate in the team project
- Once committed to a team, you cannot quit
- Earn a significant bonus for participating in the team project
- If you are a 15-319 student and want to participate in the team project, please email the professor.

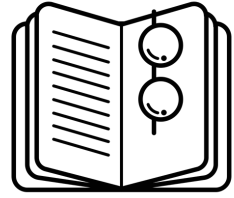
# Team Formation - Deadlines

Follow the instructions in [@1053](#) carefully

- **By Friday 2/22 at 11:59 PM ET**
  - Identify your team members
  - One team member should form a team on TPZ and all other team members should accept the invitation
    - Completing this step will freeze your team
- **By Saturday 2/23 at 11:59 PM ET**
  - Create a new AWS account ⇒ only used for the team project
  - Update the team profile in TPZ with the
    - New AWS ID aws-id
- **By Sunday 2/24 at 11:59 PM ET**
  - Finish reading the Profiling a Cloud Service primer to get yourself prepared for the team project



# This Week: Conceptual Content



- OLI, UNIT 3: Virtualizing Resources for the Cloud
  - Module 7: Introduction and Motivation
  - Module 8: Virtualization
  - Module 9: Resource Virtualization - CPU
  - **Module 10: Resource Virtualization - Memory**
  - **Module 11: Resource Virtualization – I/O**
  - **Module 12: Case Study**
  - Module 13: Storage and Network Virtualization

# OLI, Unit 3: Modules 10, 11, 12

- Understand two-level page mappings from virtual memory to real pages, from real pages to physical memory
- Learn how memory is overcommitted and reclaimed using ballooning
- Study how I/O requests are intercepted at different interfaces
- Map these concepts into your practical exploration with AWS

# OLI Module 10 - Memory Virtualization

- A process that cannot fit in physical memory? To run or not to run?
- Page Table
  - Per process
  - Maps virtual addresses to physical addresses
- One level v/s two levels mapping
- Virtual, Real, Physical address spaces
- Memory reclamation
- Ballooning

# OLI Module 11 - I/O Virtualization

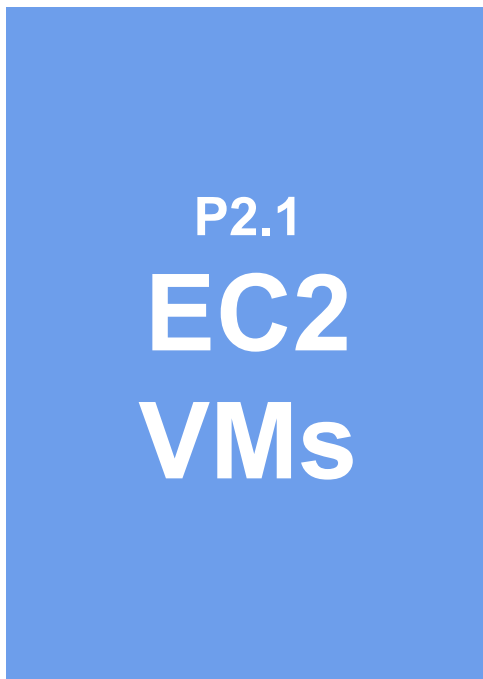
- How?
  - Construct a virtual version of the device
  - Virtualize the I/O activity routed to the device
- System call interface, the device driver interface, and the operation-level interface

## OLI Module 12 - AWS Case Study

# Project 2

Running Theme:

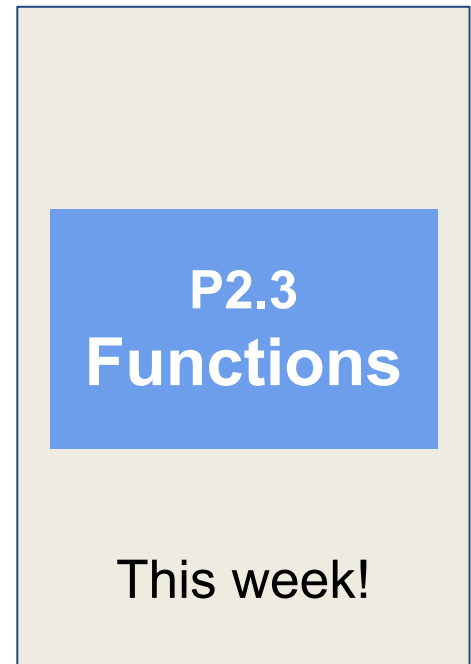
**Automating and scaling distributed systems**



P2.1  
**EC2**  
**VMs**



P2.2  
**Containers**



P2.3  
**Functions**

This week!

# Serverless Computing

- Develop and run applications on servers without having to provision or manage servers.
- Applications can have one or more functions.
- The Cloud Service Provider (CSP) provides the server to run the application.
- The developer designs the application and runs it without having to manage any servers or worry about scaling.
  - Pay-per-invocation model
- Functions-as-a-Service (FaaS) is a use-case of serverless computing

# Why use Cloud Functions

- High availability
- Resiliency
  - Each execution is contained and isolated, and thus have no impact on other executions
- Built in logging and monitoring
  - Easily accessible logs from CloudWatch ensures traceability
- Event-driven
  - Functions can be triggered by events like S3 file uploads

# Cloud Functions

- Possible use cases
  - Chat bots
  - Mobile backends
  - How about Mapreduce?
    - Not suitable for FaaS
    - EMR is PaaS
- FaaS, typically stateless functions
- Pay per number of function invocations + running time
- Scalability is automatic through provider





# AWS Lambda



- AWS FaaS offering
- Only pay for the number of invocations and the compute time (i.e., when your code runs)
- Stateless...
  - Every lambda function has 500MB of non-persistent disk space in its own /tmp dir
- Debugging?
  - CloudWatch is your friend

# AWS Lambda Programming Model

- Handler
  - Execution starts here
  - Triggered by events
- Context
  - Interact w/ AWS Lambda execution environment e.g. `getFunctionName()`, `getLogger()` etc.
- Logging
- Exceptions
- Be stateless!

# Triggers

- Events trigger Lambda functions
- Events are passed to function as input parameter
- Event sources publish events that cause the cloud function to be invoked
  - AWS Lambda: Create object in [S3 bucket](#), SNS topic etc.
  - Azure: HTTPTrigger, BLOBTrigger etc.
  - GCP: File upload to Cloud Storage, message on a Cloud Pub/Sub topic etc.

# Project 2.3 - Overview

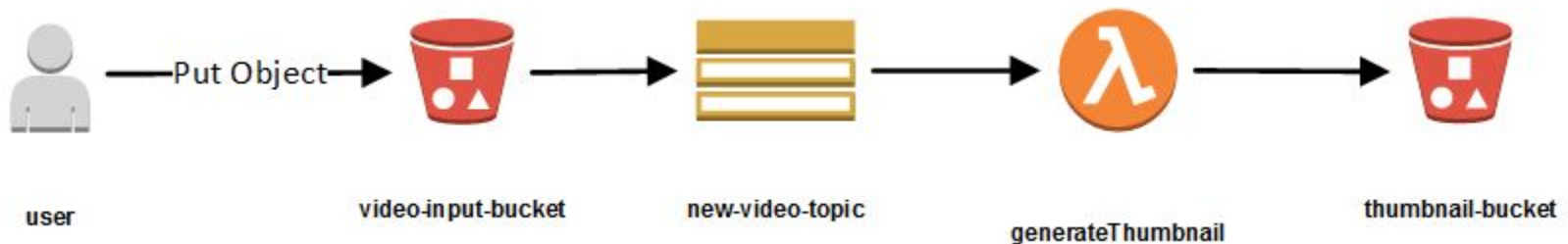
- Task 1
  - Fibonacci function in Azure Functions
  - Power sets function in GCP Cloud Functions
  - CIDR block statistics in AWS Lambda
- Task 2
  - Lambda and FFmpeg to generate thumbnails
- Task 3
  - Rekognition and CloudSearch to label thumbnails and index for video search

# P2.3 Task 1: Cloud Functions

- HTTP triggered functions
- Subtask 1
  - Fibonacci - Azure Functions
- Subtask 2
  - PowerSets - GCP Functions
- Subtask 3
  - CIDR - AWS Lambda
    - Java: use SubnetUtils class from Apache's Commons Net.
    - Python users should consider the python-iptools package or ipaddress from the Python standard library.

# P2.3 Task 2: Event Driven Functions

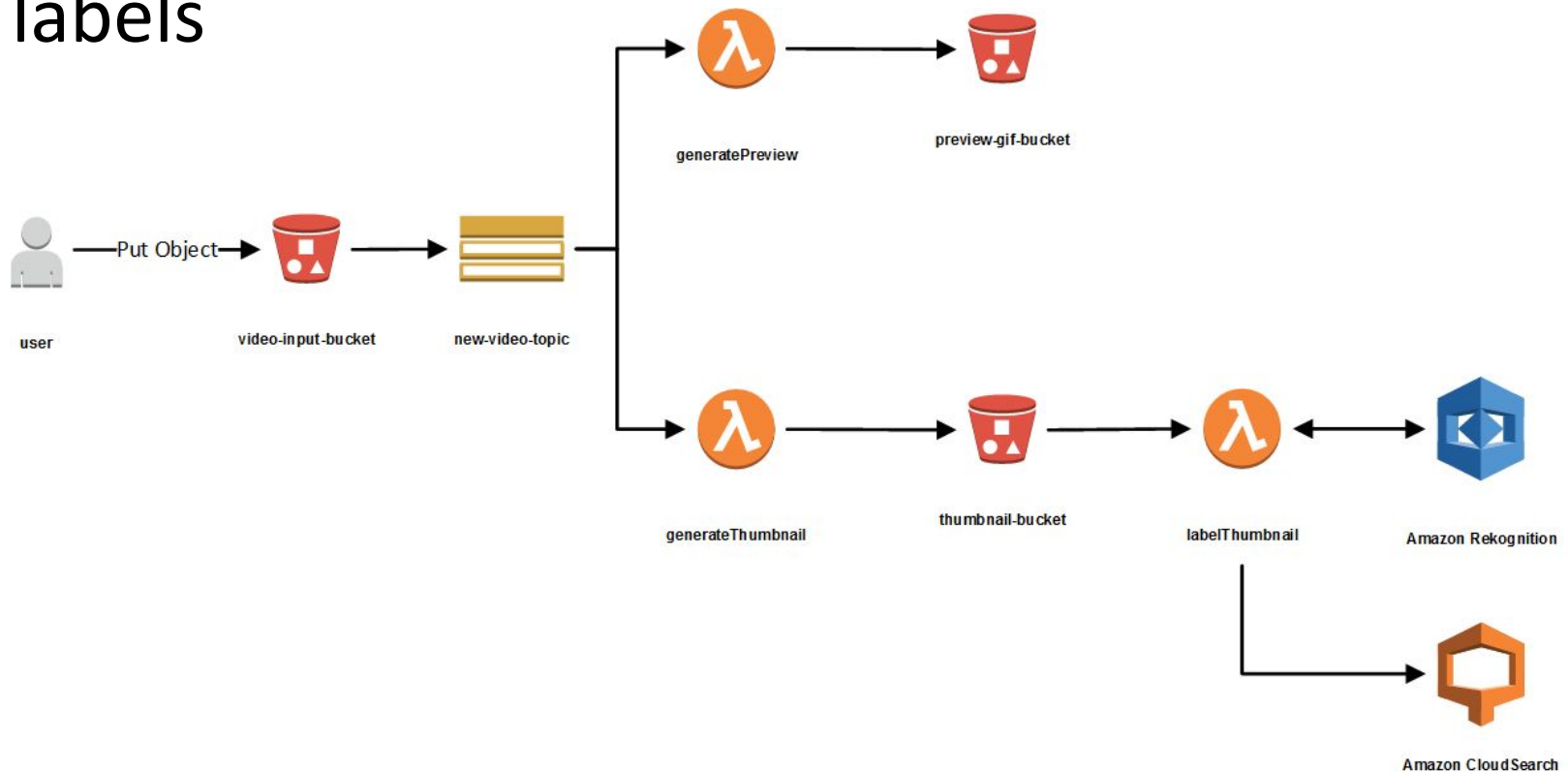
- Event driven functions using AWS Lambda
- Possible event sources:
  - S3, SNS, and CloudWatch Logs
- Event delivery protocol
  - Pull (Kinesis, DynamoDB Streams)
  - Push (S3, API Gateway)
    - S3 does not support notifying multiple Lambda functions for the same event type
    - Use SNS to fanout the event trigger



# Project 2.3 - Task 3

## Video Processing Pipeline

- AWS Lambda and FFmpeg to process videos
- AWS Rekognition for image labeling
- AWS CloudSearch to index videos based on labels





OFF



UPLOAD



0:47 / 1:40



1.00x



#27 ON TRENDING

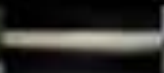
## Ant Man 2 Trailer

6,163,127 views

Video Content Search Results: CAR



47



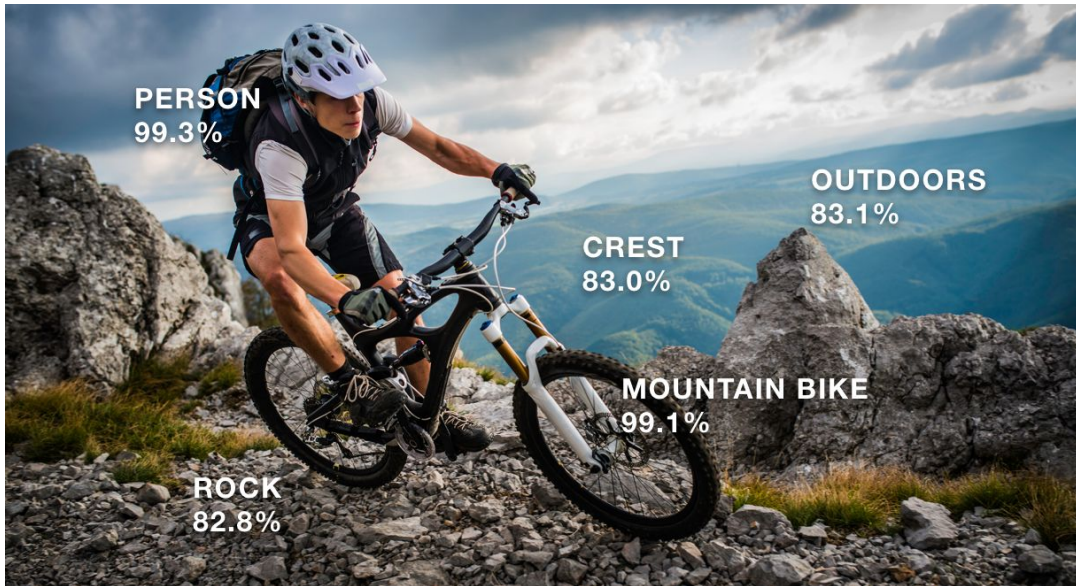


# AWS CloudSearch

- Fully managed AWS service for searching
  - Scaling
    - Disable auto-scaling for this project
  - Cost
    - Use search.m1.small (\$0.059 per hour)
  - Batch uploads (\$0.10 per 1,000 Batch Upload Requests, each batch  $\leq$  5 MB)

# AWS Rekognition

- Image recognition service



- Pricing!!

- \$1.00 per 1,000 images
- Keep budget for testing
- Be careful not to exhaust your budget

# CloudWatch Alarm

- To mitigate overspending on Rekognition, we have provided you with a CloudWatch Alarm terraform template that **you must apply**.
- The submitter will verify that you have not exceeded some threshold of calls to Rekognition before proceeding.
- Useful trick for future projects!

# Project 2.3 - Hints

- Make sure you are authenticated before submitting (EC2 IAM role, az login, gcloud auth)
- Review the suggested libraries
- ffmpeg + ffmpy works well in Python 2.7
- Test your functions and triggers manually first
  - `aws s3 cp video.mp4 s3://your-video-bucket`
- Review the CloudWatch Logs for errors
- Pay attention to the bucket permissions (you may need to change it to public) when submitting task 3 since it has a different permission requirement than task 2
- Pay attention to all your naming throughout the project, make sure you are using the same format as specified in the write-up
- Check for misconfigured permissions
  - S3 -> SNS, SNS -> Lambda, Lambda -> S3 / CS
- Use the /tmp directory for storing images and video during a function execution
  - But.. remember that functions are stateless

# Project 2.3 Penalties

Danger

## Project Grading Penalties

| Violation  | Penalty of the project grade |
|--|------------------------------|
| Spending more than \$15 for this project on AWS  | -10%                         |
| Spending more than \$30 for this project on AWS  | -100%                        |
| Failing to tag all your resources in any task (Lambda Functions, CloudSearch Domains, EC2 Instances) for this project; Key: Project and Value: 2.3 | -10%                         |
| Submitting your AWS, GCP, or Azure credentials, other secrets, or Andrew ID in your code for grading   | -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  |

# Upcoming Deadlines



- **Code Review - Project 2.1**
  - Due on Wednesday, Feb 20th, 2019, 11:59PM ET
- **Quiz 5 (OLI Modules 10, 11 & 12)**
  - Due on Friday, Feb 22th, 2019, 11:59PM ET
- **Team Project - Team Formation**
  - Due on Friday, Feb 22th, 2019, 11:59PM ET
- **Project 2.3**
  - Due on Sunday, Feb 24th, 2019, 11:59PM ET

# Team Formation - Deadlines



Follow the instructions in [@1053](#) carefully

- **By Friday 2/22 at 11:59 PM ET**
  - Identify your team members
  - One team member should form a team on TPZ and all other team members should accept the invitation
    - Completing this step will freeze your team
- **By Saturday 2/23 at 11:59 PM ET**
  - Create a new AWS account ⇒ only used for the team project
  - Update the team profile in TPZ with the new AWS ID `aws-id`
- **By Sunday 2/24 at 11:59 PM ET**
  - Finish reading the Profiling a Cloud Service primer to get yourself prepared for the team project

# Questions?

