15-319 / 15-619
Cloud Computing

Overview 5

September 28th, 2021

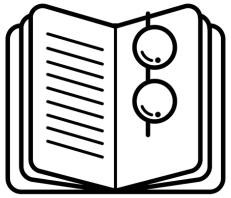
Reflection of Last Week

- Conceptual content on OLI
  - Modules 5, 6 and Quiz 3
- Finished P1 Elasticity
  - Horizontal Scaling using AWS API to reach target RPS
  - Auto Scaling using AWS API to reach target RPS
  - Auto Scaling using Terraform
- Project theme - **Containers: Docker and Kubernetes**
  - Docker Intro / Embedded Profile Service
  - Intro to Helm Charts / Deploying MySQL
  - WeCloud Chat Microservices Architecture
    - Autoscaling, Multi-Cloud and Fault Tolerance to Azure

This Week

- **OPE Training Session**
  - Due on **Sunday**, October 3rd, 2021, 11:59PM ET
- **Quiz 4 (OLI Modules 7, 8 & 9)**
  - Due on **Friday**, October 1st, 2021, 11:59PM ET
- **Project 2**
  - Due on **Sunday**, October 3rd, 2021, 11:59PM ET
- **Team Project Phase 1**
  - Released this week!

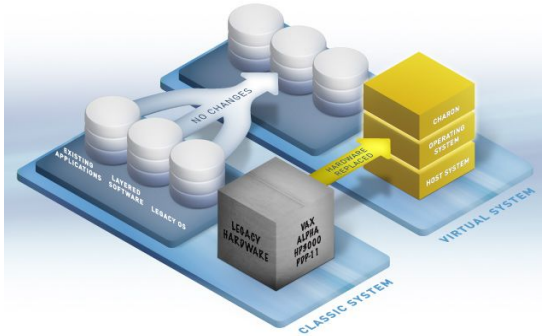
This Week: Conceptual Content



- OLI, Unit 3: Cloud Infrastructure
  - 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

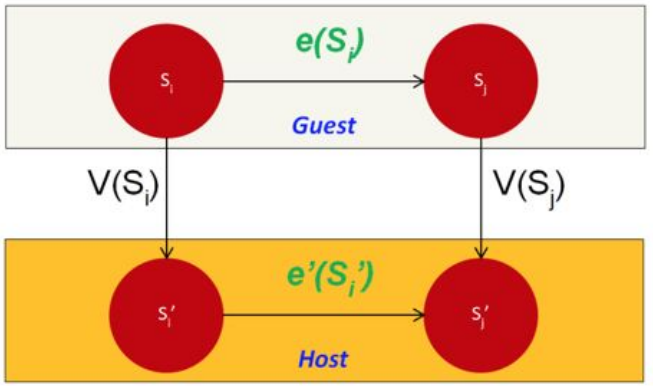
OLI Module 7 - Virtualization Introduction and Motivation

- Why virtualization?
  - Elasticity
  - Resource sandboxing
  - Mixed OS environment
  - Resource sharing
  - Improved system utilization and reduced costs



OLI Module 8 - Virtualization

- What is Virtualization?
  - Involves the construction of an isomorphism that maps a virtual guest system to a real (or physical) host system
  - Sequence of operations *e* modify guest state
  - Mapping function *V(Si)*



- Virtual Machine Types
  - Process Virtual Machines
  - System Virtual Machines

OLI Module 9
Resource Virtualization - CPU

- Steps for CPU Virtualization
  - Multiplexing a physical CPU among virtual CPUs
  - Virtualizing the ISA (Instruction Set Architecture) of a CPU
- Code Patch, Full Virtualization and Paravirtualization
- Emulation (Interpretation & Binary Translation)
- Virtual CPU

TEAM PROJECT
Twitter Data Analytics

Team Project

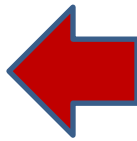
Twitter Analytics Web Service

- Given ~1TB of Twitter data
- Build a performant web service to analyze tweets
- Explore web frameworks
- Explore and optimize database systems
- Explore cloud-native development and deployment approaches

Team Project

- Phase 1:
  - Microservice 1
  - Microservice 2
  - Microservice 3 (checkpoint, MySQL **or** HBase)

- **Input your team account ID and GitHub username on the Sail() Platform**

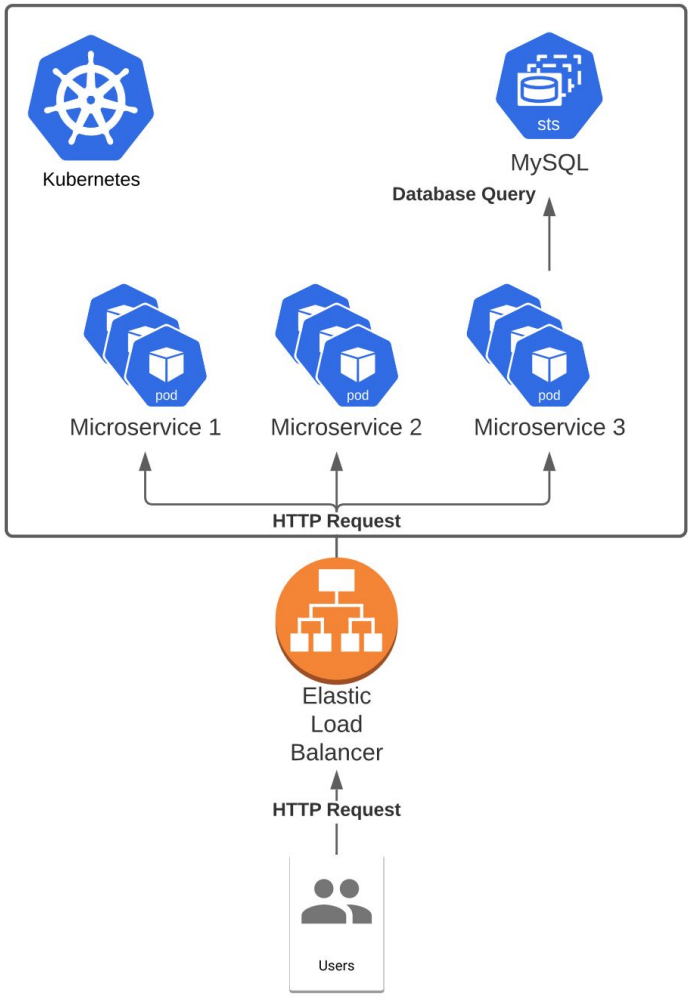


- Phase 2 - Live Test!
  - Microservice 1, 2
  - Microservice 3 (full performance, MySQL **or** HBase)

- Phase 3 - Live Test!
  - Microservice 1, 2, 3 (Managed Cloud Services)

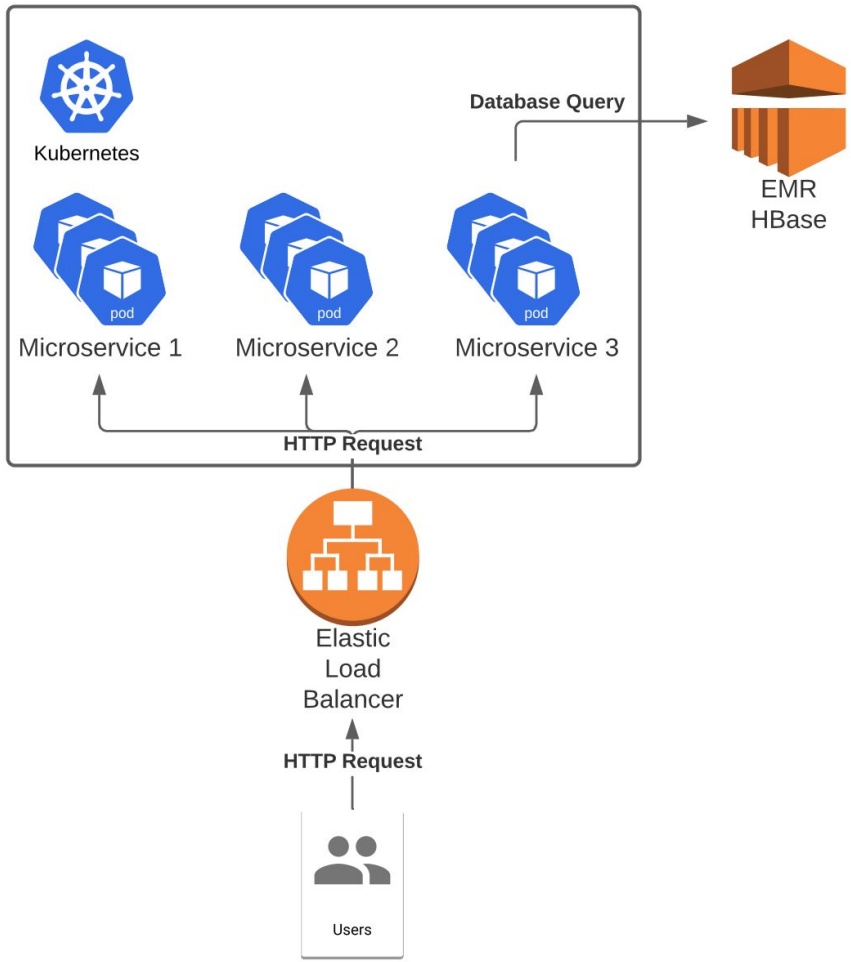
Team Project

- Kubernetes cluster
  - Self-managed on EC2 instances
- Three Microservices
  - Each has a web-tier
  - M3 is the Twitter Analysis microservice which has a storage tier

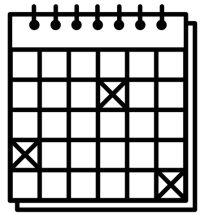


Team Project

- Choose between MySQL and HBase for M3 storage tier
- Same RPS target, higher hourly budget for EMR
  ⟶ Not allowed to host web-tier on EMR cluster!

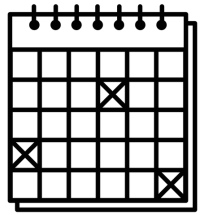


Team Project Time Table



Phase	Deadline (11:59PM ET)
Phase 1 (20%) **- M1** **- M2** **- M3 (ckpt)**	**• M1 CKPT (5%): Sun, 10/3** **• M1 CKPT Report (5%): Sun, 10/3** • M1 FINAL (10%): Sun, 10/10 • M2 CKPT (5%): Sun, 10/10 • M2 FINAL (50%): Sun, 10/17 • M3 CKPT (5%): Sun, 10/17 • Final Report + Code (20%): Tue, 10/19 **BONUSES:** **• M1 Early Bird Bonus (5%): Sun, 10/3** • M2 Early Bird Bonus (5%): Sun, 10/10 • M2 Correctness Penalty Waiver: Sun, 10/10

Team Project Time Table



Phase	Deadline (11:59PM EST)
Phase 2 (30%) **- M1** **- M2** **- M3 (full)** **- Live Test!**	● Live Test on Sun, 10/31
Phase 3 (50%) **- Managed Services for Microservice 1-3** **- Live Test!**	● Live Test on Sun, 11/14

Team Project Deadlines - Phase 1

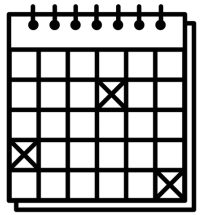
- Writeup and released on Monday 9/27.
- Phase 1 milestones:
  - M1 Checkpoint: **Sunday, 10/3**
    - A successful 10-min submission for M1
  - M1 Checkpoint Report: **Sunday, 10/3**
  - M1 Final due: **Sunday, 10/10**
    - Achieve the M1 target
  - M2 Checkpoint: **Sunday, 10/10**
    - A successful 10-min submissions for M2
  - M2 Final due: **Sunday, 10/17**
    - Achieve the M2 target
  - M3 Checkpoint: **Sunday, 10/17**
    - A successful 10-min submissions for M3 (MySQL **or** HBase)
  - Phase 1, code and report: **Tuesday, 10/19**

Successful: having a non-zero score

Team Project Deadlines - Phase 1

- Phase 1 **bonus milestones**:
  - [5%] M1 Early Bird Bonus: **Sunday, 10/3**
    - Reach target 40000 for M1 (10-min submission)
    - Fill in the checkpoint report
  - [5%] M2 Early Bird Bonus: **Sunday, 10/10**
    - Reach target 25000 for M2 (10-min submission)
  - [Penalty Waiver] M2 Correctness bonus: **Sunday, 10/10**
    - One 10-min submission with correctness above 95%
- Start early, read the report and earn bonus points!
- Remember to fill in the report to claim your bonus!

Suggested Tasks for Phase 1



Phase 1 weeks	Tasks	Deadline
Week 1 ● **9/27 - 10/3**	● **Team meeting** ● **Read Write Up & Report** ● **Complete M1 code & achieve correctness** ● **Start writing M2 solution** ● **Think about M3 database schema**	● **M1 Checkpoint due on 10/3** ● **Checkpoint Report due on 10/3**
Week 2 ● 10/4 - 10/10	● Optimize for M1 performance ● Complete correct M2 code ● **Start ETL process for M3**	● M1 final target due on 10/10 ● M2 Checkpoint due on 10/10
Week 3 ● 10/10 - 10/17	● Optimize for M2 performance ● Finish M3 ETL process ● Complete M3 code & achieve correctness	● M2 final target due on 10/17 ● M3 Checkpoint due on 10/17 ● Final Report due on 10/19

Microservice 1 - QR code

Submission Budget: $0.70/h

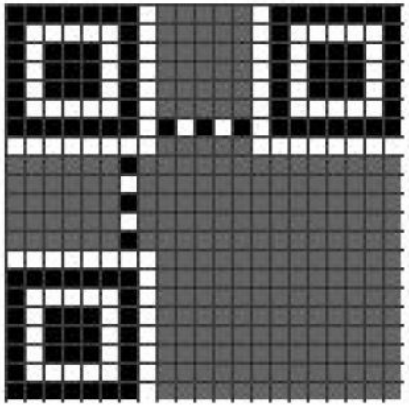
- M1 does not require a database
  - Implement encoding and decoding of QR code
    - A simplified version of QR
  - You must explore different web frameworks
    - Get at least 2 different web frameworks working
    - Techempower will be a good resource to explore
    - Select the framework with the better performance
    - Provide evidence of your experimentation

QR Code

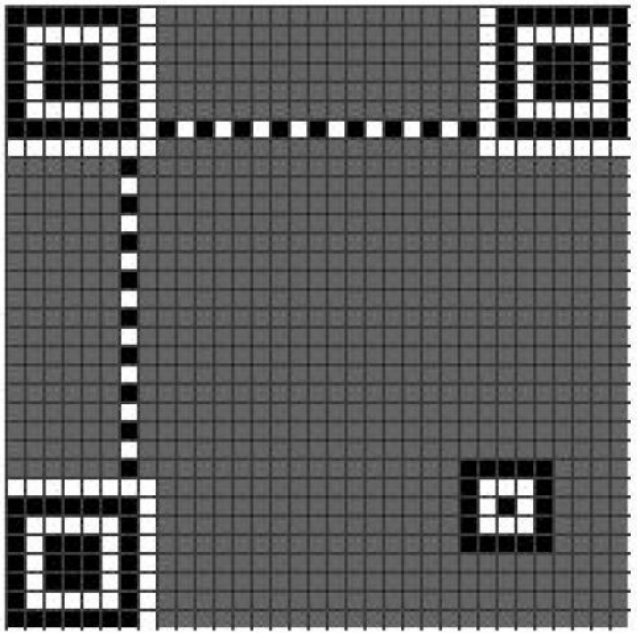
QR Code


ME
FRIENDS
TrainerNIA
& Larvitar
40
LEVEL
BUDDY
JOURNAL
STYLE
TRAINER NAME :
CMont13
OK
ME
FRIENDS
ADD FRIEND
GONIATRAINER
GONIATRAINER caught a Magikarp!
TRAINER
TRAINER caught a Pikachu!
TrainerNIA3
TrainerNIA3 caught a Larvitar!
TrainerNIA
TrainerNIA caught a Magikarp!
Trainer3

QR Code

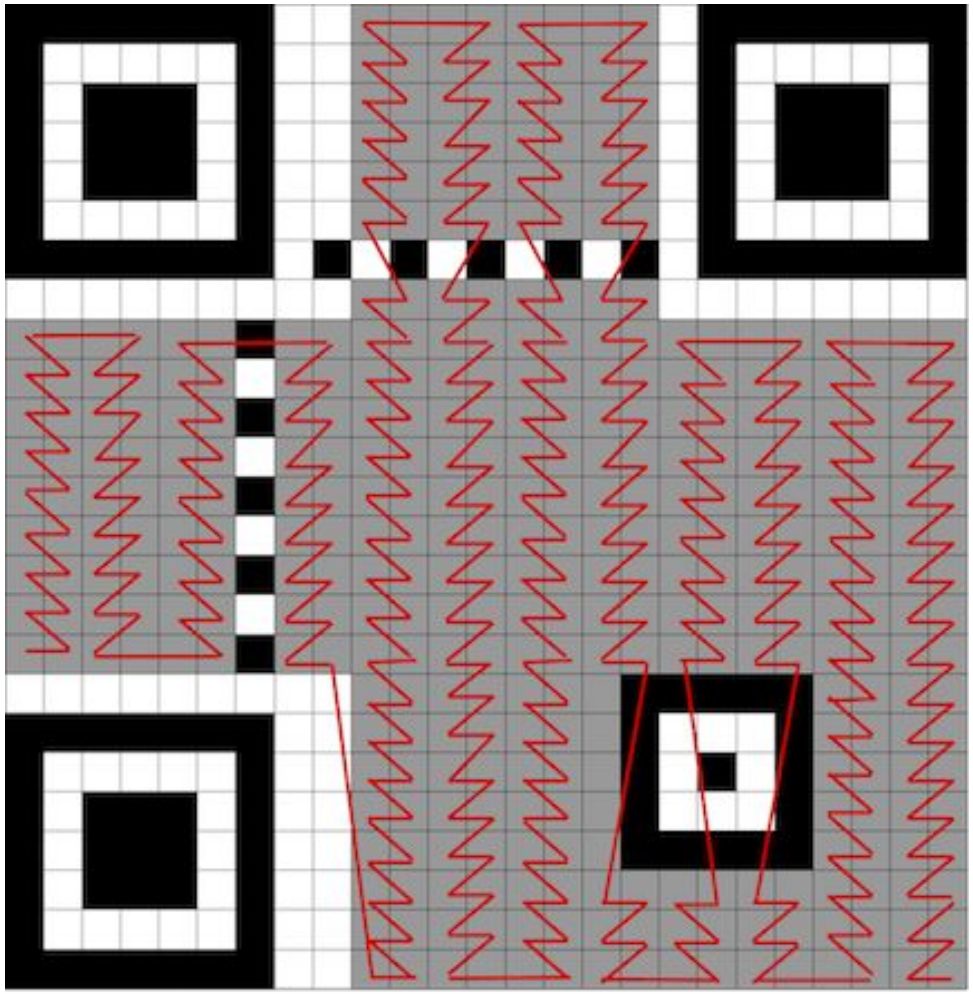


Version 1 (21 * 21)

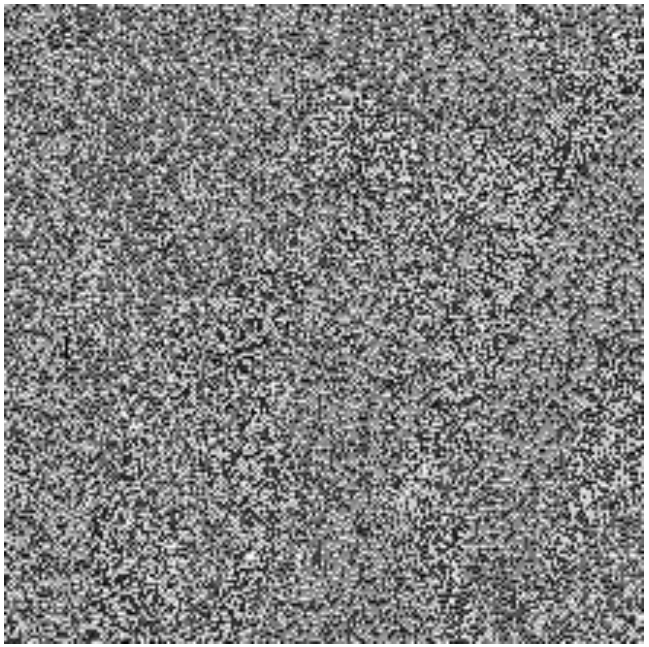


Version 2 (25 * 25)

QR Code



Logistic Map



Microservice 2 - CloudCoin

Submission Budget: $0.70/h

- M2 does not require a database
- Implement a web service that verifies and updates blockchains.
- You must explore different web frameworks
  - Provide evidence of your experimentations
- More computationally intensive than M1
  - Need to profile and optimize your code

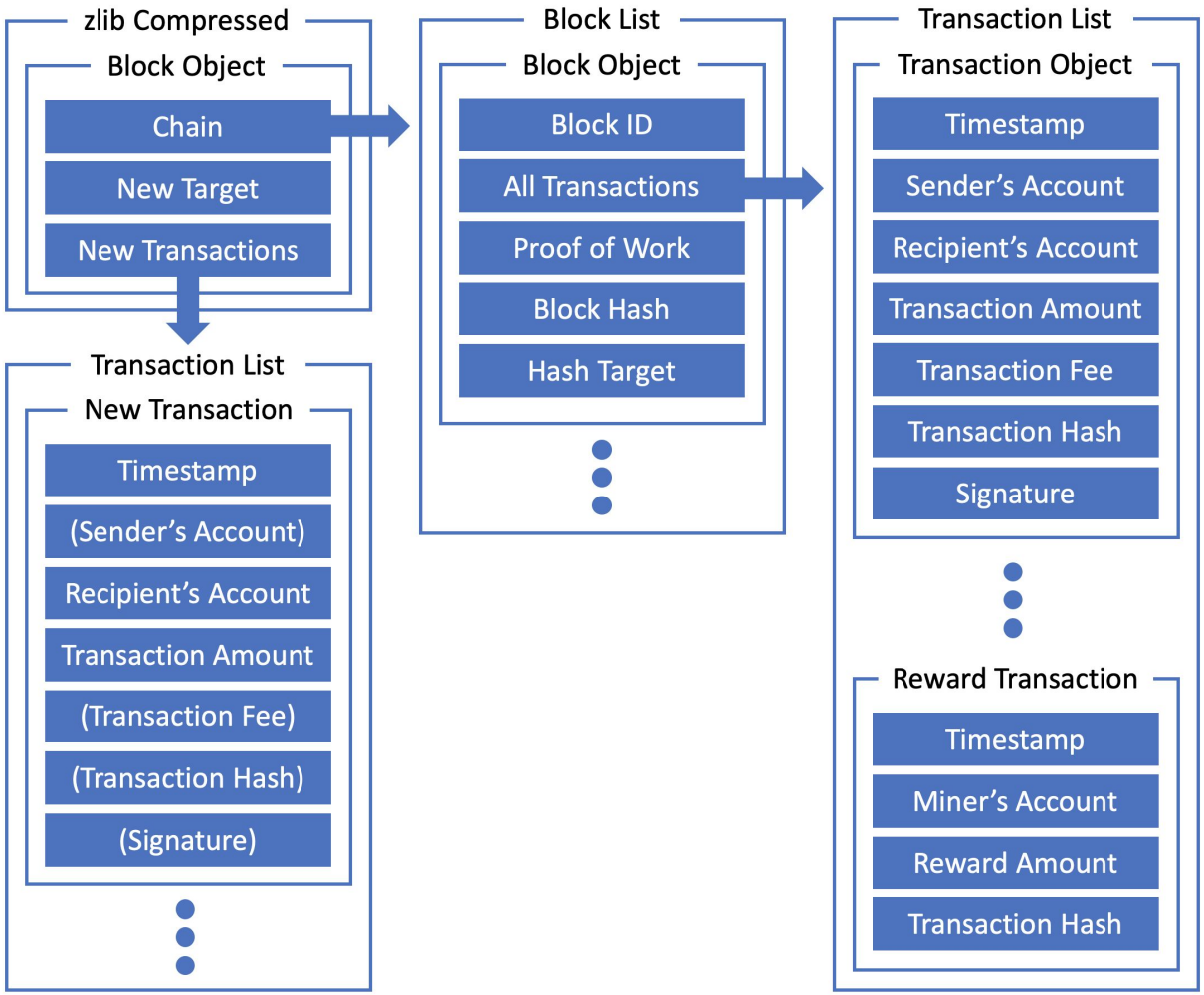
- **Read the report template first!!!**

What is a blockchain, though?

- Data structure that supports digital currency.
- Designed to be untamperable.
- Distributed. Shared among all user nodes.
  - Decentralized
  - Fault Tolerant
- Consists of chained blocks.
- Each block consists of transactions.

M2 Example

- **M2 input:**




```
{
 "chain": [
   {
     "all_tx": [{
       "recv": 895456882897,
       "amt": 500000000,
       "time": "1582520400000000000",
       "hash": "4b277860"
     }],
     "pow": "0",
     "id": 0,
     "hash": "07c98747",
     "target": "1"
   },
   {
     "all_tx": [
       {
         "sig": 1523500375459,
         "recv": 831361201829,
         "fee": 2408,
         "amt": 126848946,
         "time": "1582520454597521976",
         "send": 895456882897,
         "hash": "c0473abd"
       },
       {
         "recv": 621452032379,
         "amt": 500000000,
         "time": "1582521002184738591",
         "hash": "ab56f1d8"
       }
     ],
     "pow": "202",
     "id": 1,
     "hash": "0055fd15",
     "target": "01"
   },
   {
     "all_tx": [
       {
         "sig": 829022340937,
         "recv": 905790126919,
         "fee": 78125,
         "amt": 4876921,
         "time": "1582521009246242025",
         "send": 831361201829,
         "hash": "46b61f8e"
       },
       {
         "sig": 295281186908,
         "recv": 1097844002039,
         "fee": 0,
         "amt": 83725981,
         "time": "1582521016852310220",
         "send": 895456882897,
         "hash": "b6c1b10f"
       },
       {
         "recv": 905790126919,
         "amt": 250000000,
         "time": "1582521603026667063",
         "hash": "b0750555"
       }
     ],
     "pow": "12",
     "id": 2,
     "hash": "00288a38",
     "target": "0a"
   }
 ],
 "new_target": "007",
 "new_tx": [
   {
     "sig": 160392705122,
     "recv": 658672873303,
     "fee": 3536,
     "amt": 34263741,
     "time": "1582521636327155516",
     "send": 831361201829,
     "hash": "1fb48c71"
   },
   {
     "recv": 895456882897,
     "amt": 34263741,
     "time": "1582521645744862608"
   }
 ]
}
```

M2 Example

- **M2 input:**
  - Several blocks, each with numerous transactions
  - A new transaction to be processed by you
- **M2 output:**
  - Check each block and transaction for validity
  - If valid, find a POW that satisfy the hash requirement.
  - Add the completed transaction and new block into the blockchain.

(You could verify your understanding using our reference server)

M2 Example

```
{
  "all_tx": [...],
  "pow": "cloud",
  "id": 2,
  "hash": "09288a38",
  "target": "0a"
}
```

- **Block:**
  - Created by “miners”.
  - Has a list of transactions.
  - Block hash encapsulates
    - all transaction info and block Metadata, as well as the hash of the previous block, plus a PoW chosen by the miner.
  - Miner finds a PoW (Proof of Work) through brute forcing, to make the block hash lexicographically smaller than the hash target.
  - Block hash formula:

```
CCHash(SHA-256("block_id|previous_block_hash|tx1_hash|tx2_hash|tx3_hash...") + PoW)
```

M2 Example

- **Transfer Transaction:**
  - Signature is computed with hash value using RSA.
    - sig=RSA(hash, key)
  - Hash value computed using all info in the blue box.
  - Transaction hash formula:

```
CCHash("timestamp|sender|recipient|amount|fee")
```


```
{
  "send": 831361201829,
  "recv": 905790126919,
  "amt": 4876921,
  "fee": 78125,
  "time": "15825210092462425",
  "sig": 829022340937,
  "hash": "46b61f8e"
},
```

M2 Example

- **Reward Transaction:**
  - Special type of transaction.
  - Created by miner.
  - Is the last transaction in
    - the block's transaction list.
  - Reward amount determined by block id, 500,000,000 for the first two blocks, halved for any two following blocks.


```
{
  "recv": 905790126919,
  "amt": 250000000,
  "time": "15825216030266670 63",
  "hash": "b0750555"
}
```

M2 Example

- **New transactions:**
  - Contains transactions made by your team or by some other accounts.
  - Transaction made by some other account has the same format as any non-reward transaction in the block list.
  - For the transactions made by your team, you need to fill in missing fields and sign it using the key given to you.


```
"new_tx": [
  {
    "sig": 160392705122,
    "recv": 658672873303,
    "fee": 3536,
    "amt": 34263741,
    "time": "1582521636327155516",
    "send": 831361201829,
    "hash": "1fb48c71"
  },
  {
    "recv": 895456882897,
    "amt": 34263741,
    "time": "15825216457448626O8"
  }
]
```

M2 Example

- **M2 Output:**
  - Collect the new transactions.
  - Create a reward transaction.
  - Include these transactions in a new block.
  - Compute a PoW that makes the new block hash satisfies the new hash target.
  - Append the block to the chain.
  - Respond with the zlib compressed and Base64 encoded new JSON.

M2 Example

- **M2 Output:**
  - There will be malicious attempts to break the blockchain.
  - You need to check the validity of the chain.
  - If the chain is not valid, return a string that starts with INVALID.
  - You can append any debug info you want. Just make sure it does not start a new line.
  - E.g., INVALID|any_debug_info_you_like

M3 - User Recommendation System

Submission Budget: $0.70/h MySQL, $1.10/h HBase

Use Case: When you follow someone on twitter, recommend close friends.

Three Scores:

- Interaction Score - closeness
- Hashtag Score - common interests
- Keywords Score - to match interests

Final Score: Interaction Score * Hashtag Score * Keywords Score

Query:

GET /twitter?
user_id=<ID>&
type=<TYPE>&
phrase=<PHRASE>&
hashtag=<HASHTAG>

Response:

<TEAMNAME>,<AWSID>\n
uid\tname\tdescription\ttweet\n
uid\tname\tdescription\ttweet

M3 Example

```
GET /twitter?
user_id=100123&
type=retweet&
phrase=hello%20cc&
hashtag=cmu

TeamCoolCloud,1234-0000-0001
100124\tAlan\tScientist\tDo machines think?\n
100125\tKnuth\tprogrammer\thello cc!
```

M3 ETL Recommendations

- Do the filtering on the first part of the dataset and make sure the result is **exactly the same** as the reference answer. (See **Reference the ETL result of a small dataset** at the end of the Microservice 3 write-up.)
- Start ETL on the mini dataset in GCP/Azure, pick some queries as the test cases and compare your result against the mini reference server (See **Phase1 - Reference Server** in the write-up)
  - You can also test locally for your result (See **Continuous testing** section)
- Plan and test your data loading technique ahead of time!

Phase 1 Budget

- Your web service should not cost more than **$0.70/hour (M1, M2 and M3 MySQL) and $1.10/hour (M3 HBase)** this includes:
  - EC2 cost (Even if you use spot instances, we will calculate your cost using the **on-demand** instance price)
  - **EBS cost**
  - **ELB cost**
  - We will not consider the cost of data transfer and EMR software
  - See writeup for details
- AWS total budget of $80 for Phase 1

Git Workflow

- Commit your code to the private repo we set up
  - Update your GitHub username in the Sail() Platform!
- Make changes on a new branch
  - Work on this branch, commit as you wish
  - Open a pull request to merge into the **master** branch
  - Make sure your final phase 1 code and reports are in **master** branch
- Code review
  - Someone else needs to review and accept (or reject) your code changes
  - This process will allow you to capture bugs and remain informed on what others are doing

Heartwarming Tips from Your Beloved TAs

1. Design your architecture early and apply for limit increase.
2. Test with single VM before spinning up the cluster!
3. Spot Instance can be your friend or enemy.
4. EC2 VM is not the only thing that costs money.
5. Different CPU architecture requires different AMI.
6. Primers and individual project writeups are helpful.
7. You don't need all your hourly budget to reach M1 target.
8. Coding is the least time consuming part.
9. Think before you do esp. for ETL (Azure, GCP, or AWS).
10. Divide workload appropriately. Take up your responsibility.
11. Read the write-up.
12. Read the write-up again.
13. Read the report.
14. Start early. You cannot make-up the time lost
15. I'm not kidding. Drama happens frequently.
16. Don't forget to use the reference server to verify you result!

Best Wishes!!!