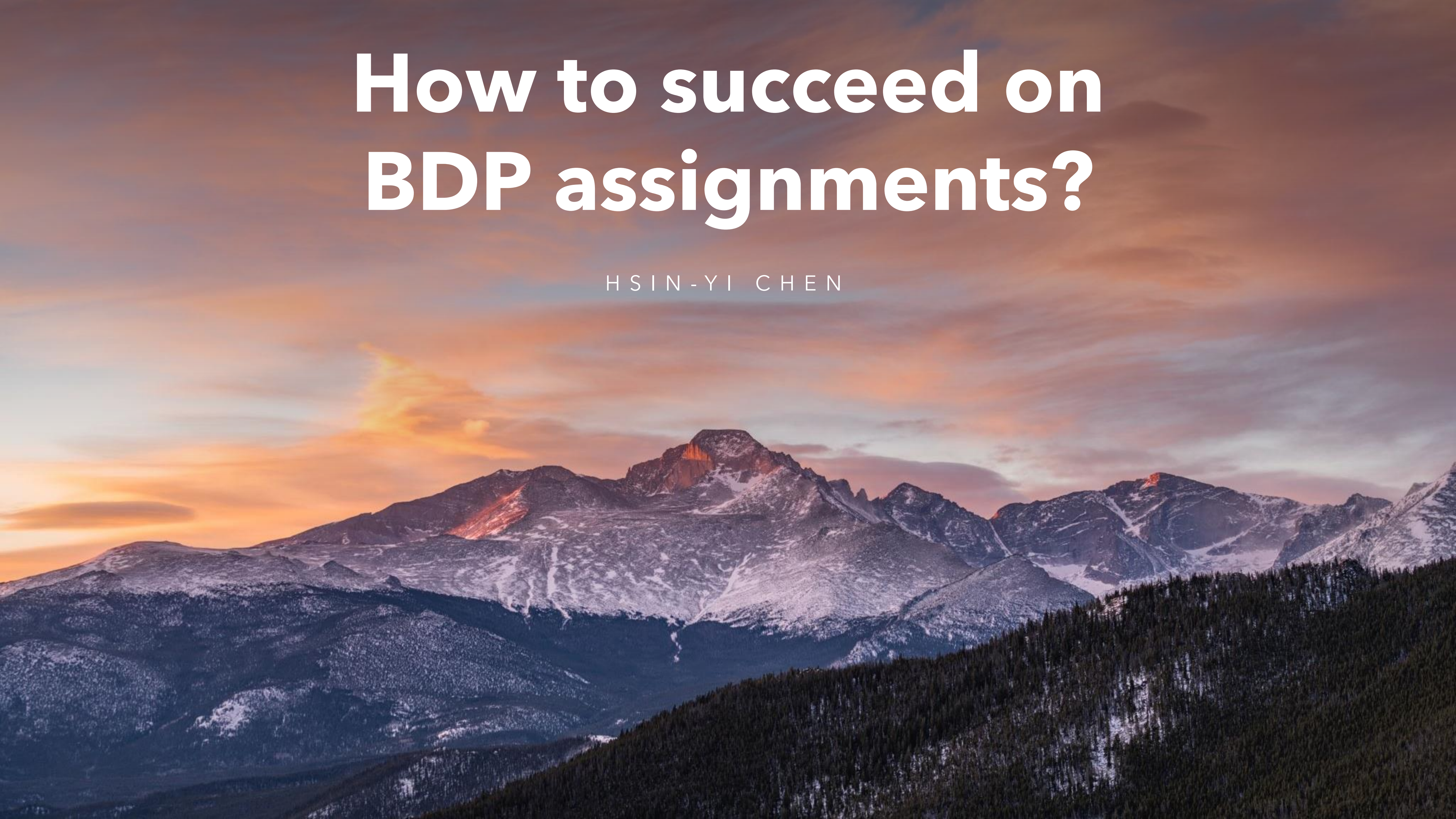
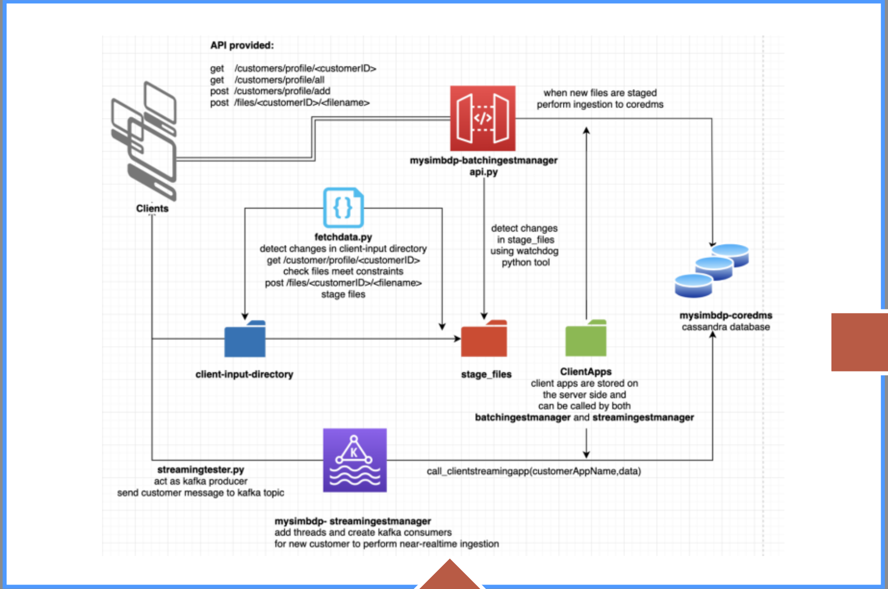


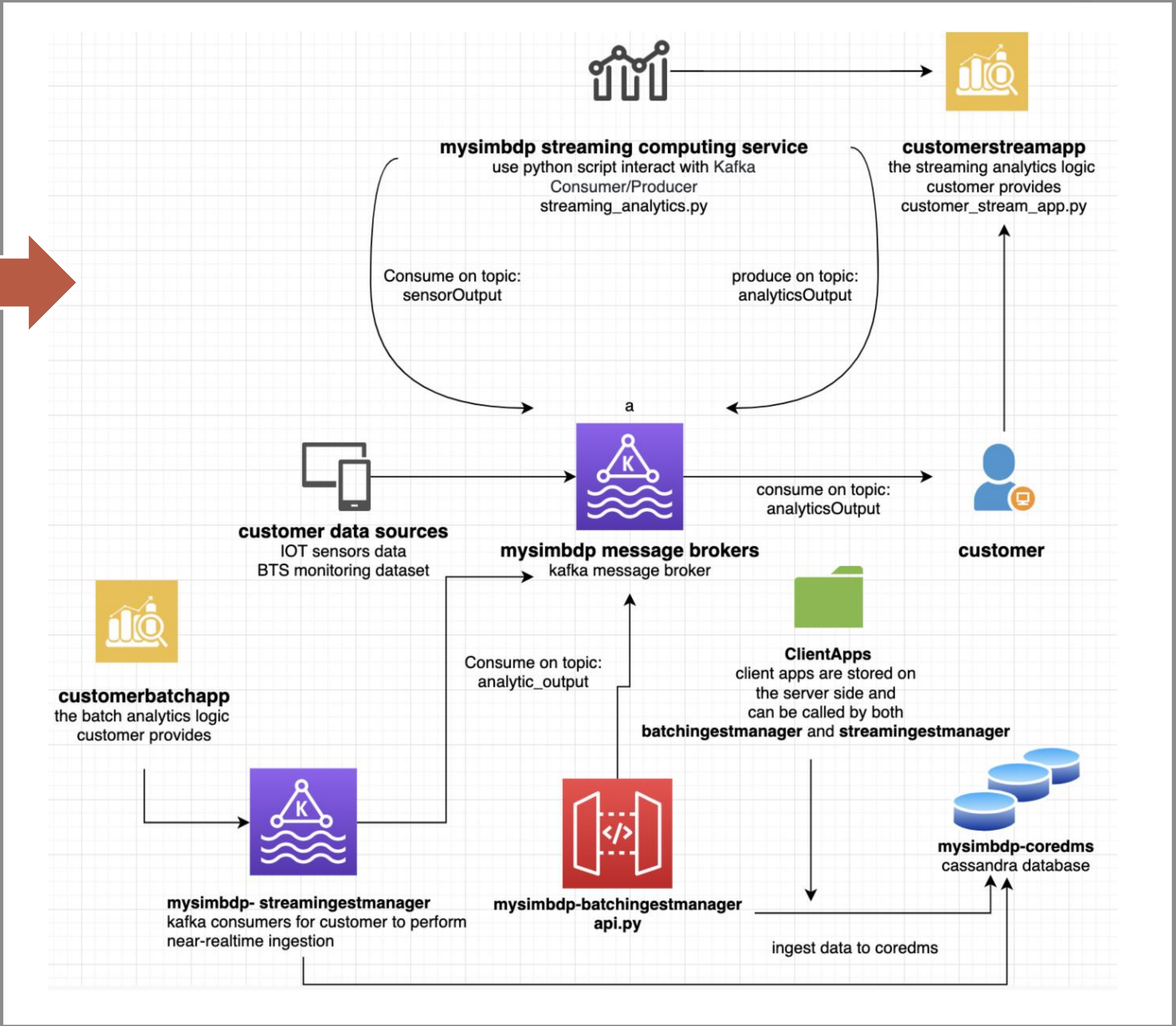
How to succeed on BDP assignments?

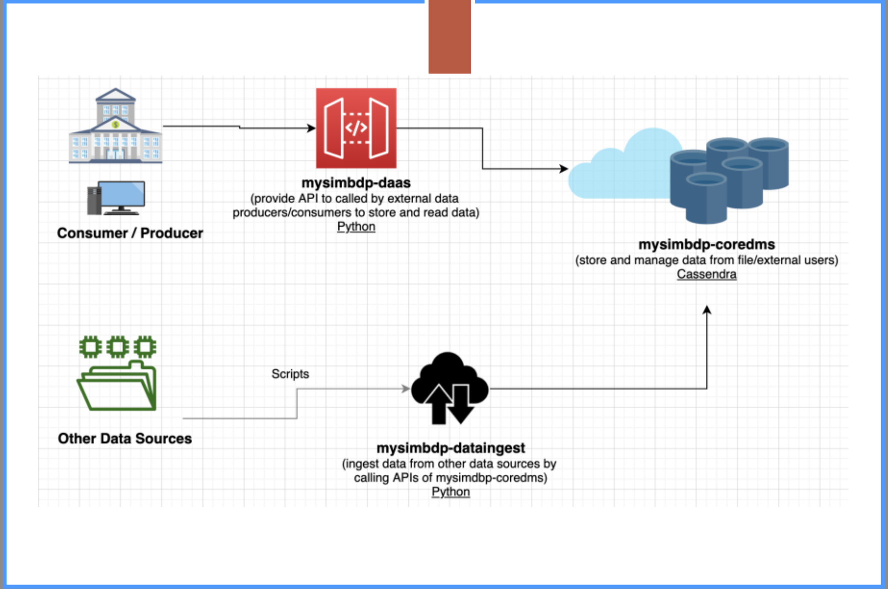
HSIN-YI CHEN

Assignment 2


API provided:
get /customers/profile/<customerID>
get /customers/profile/all
post /customers/profile/add
post /files/<customerID>/<filename>
Clients
mysimbdp-batchingestmanager
api.py
when new files are staged
perform ingestion to coredms
fetchdata.py
detect changes in client-input directory
get /customer/profile/<customerID>
check files meet constraints
post /files/<customerID>/<filename>
stage files
detect changes
in stage_files
using watchdog
python tool
mysimbdp-coredms
cassandra database
client-input-directory
stage_files
ClientApps
client apps are stored on
the server side and
can be called by both
batchingestmanager and streamingestmanager
streamingtester.py
act as kafka producer
send customer message to kafka topic
call_clientstreamingapp(customerAppName,data)
mysimbdp- streamingestmanager
add threads and create kafka consumers
for new customer to perform near-realtime ingestion

Assignment 3



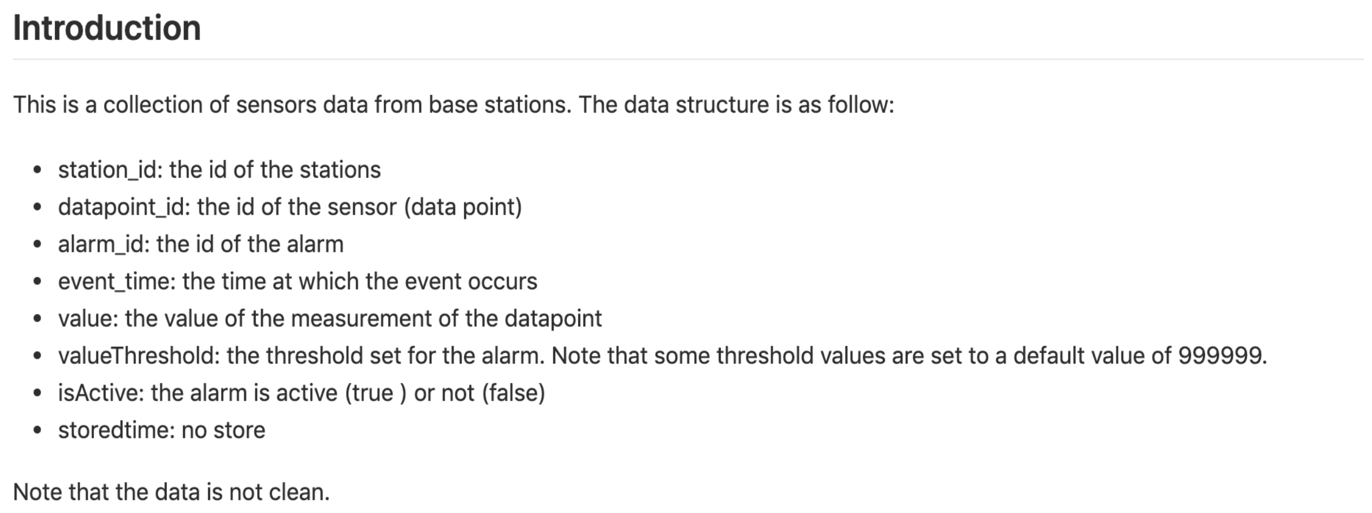

Consumer / Producer
mysimbdp-daas
(provide API to called by external data
producers/consumers to store and read data)
Python
mysimbdp-coredms
(store and manage data from file/external users)
Cassendra
Other Data Sources
Scripts
mysimbdp-dataingest
(ingest data from other data sources by
calling APIs of mysimdbp-coredms)
Python

Assignment 1

Before you start, think of the real-world examples you are interested in

- Look at the provided dataset, think of the associate scenario, and choose what you want to work with based on your interest.
- Example: BTS alarm data

$$\begin{tikzcd}
 & \text{Streaming Ingestion} \arrow[r] & \text{Streaming Analytics} \arrow[r] & \text{Alert} \\
\text{Example: BTS alarm data} \arrow[ur] \arrow[r] & \text{Batch Ingestion} \arrow[r] & \text{Batch Analytics} \arrow[r] & \text{Monthly Report}
\end{tikzcd}$$



Introduction

This is a collection of sensors data from base stations. The data structure is as follow:

- station_id: the id of the stations
- datapoint_id: the id of the sensor (data point)
- alarm_id: the id of the alarm
- event_time: the time at which the event occurs
- value: the value of the measurement of the datapoint
- valueThreshold: the threshold set for the alarm. Note that some threshold values are set to a default value of 999999.
- isActive: the alarm is active (true) or not (false)
- storedtime: no store

Note that the data is not clean.

1. Start Early Enough

- About 2.5 weeks (Design + Implementation + Reporting)
- **Design** - Takes time to understand the requirement and constraints, and research in different technologies and what to choose
- **Implementation** – Takes time to read documentation and understand how certain libraries works
  - Main architecture
  - Testing
  - Deploying
- **Reporting** – Make notes for the report along your design and implementation! Don’t only write everything in the end. You also need to answer some theoretical/extensibility questions.

2. Read the whole assignment before you start to design

- Assignment Structure
  - Introduction - Goal
  - Constraint and Input – Components and Dataset
    - **Component**, **Action**, **Stakeholders**, **Interaction**
    - E.g. a key component, called **mysimbdp-dataingest**, to **read data from data sources** (files/external databases) of **the tenant/user** and then **store the data by calling APIs of mysimbdp-coredms.**
  - Requirement and Delivery
    - Design
    - Development and Deployment
    - Extension

The questions help you understand the concerns you have to take into account when desigining your solution.

3. Design your whole solution before you start to code

- Think about **why** you're using a certain technology before using it in your design.
    - Be honest based on your knowledge
    - Certain technology strength fits the dataset
    - My example: Database=>Cassandra (reason: one of the most popular big database, high scalability)
- Write down your assumptions and design choices while designing and show in the report

4. We don't need to build a complete product, but a proof of concept

- Resource constraints => we are not looking for perfect performance
- Not too simple, even we don’t have the machines, but keep in mind the intend environment and the size of datasets. E.g. Reflect scalability issue
- Example : RestAPI copy file => one thread, only test with small dataset and limit clients => adding concurrency
- But! Do not write "random answer" just with a hope to get some points

5. Reuse Your Code and Environment

- Design and write your code in a modular way from the beginning since you'll most probably need to reuse for the later assignments
- Be consistent through your implementation (e.g. environment - local, google cloud, other resources…) -> save your time
- Resource constraint?
    - Free online resources: google cloud/AWS student, MongoDB Atlas, online message-broker server
    - Local: Microservices in different containers

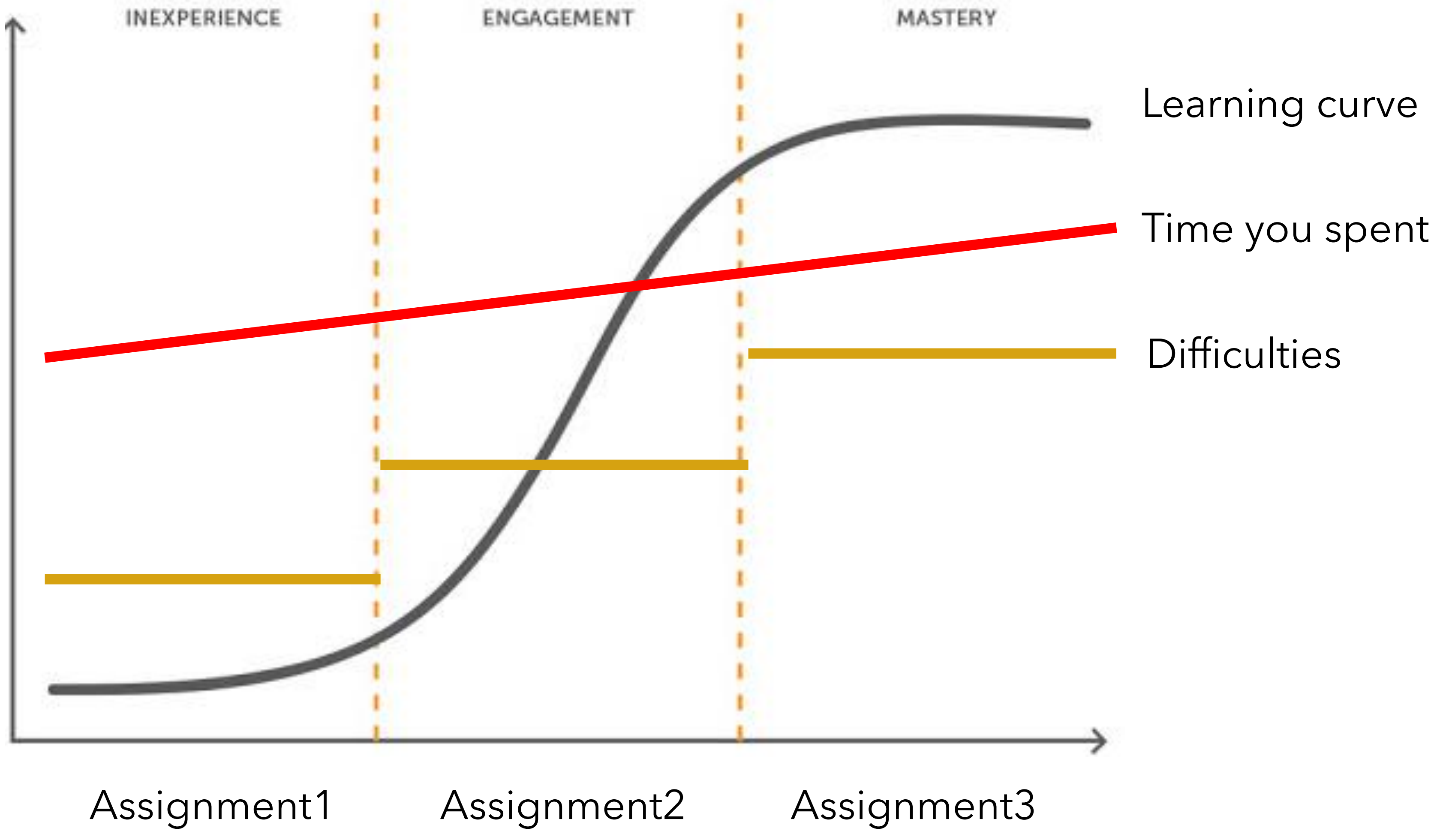
6. Report nicely

- Installation and deployment
  - Download the git, run the application simply by following the instructions
- Use graph to show your design and evaluation (real numbers!)
  - Monitoring tools: Prometheus with Docker and Kafka (open source)
- Not too long but explains your design choices, dataflow, and results. One good review is to see if others can understand your code just by reading your report.
- Report problem even if you don’t know why (e.g. performance went down dramatically when clients number increase)
- Honesty => if you reuse exist component/code, specify clearly

7. Submission – git repository

- Use git to manage your progress
- Do not put external libraries in your repository! Should automatically download somehow. (e.g. pip3 install -r requirements.txt)
- Packaging using docker is not mandatory, but a good practice

8. Your learning experience



9. Ask in the channel if there's any unclarity

- Do communicate with TA/prof etc. for unclear points
- Help answer other people's question

Q&A