Assignment 5 (Activity & Homework) myDropbox

Cloud Computing Technologies
Computer Engineering, Chulalongkorn University

Instructor: Kunwadee Sripanidkulchai, Ph.D.

We will build the myDropbox application

- We will build a Dropbox-like app called myDropbox with these features
  - Upload/Download
  - Sharing
  - Syncing
  - Encryption

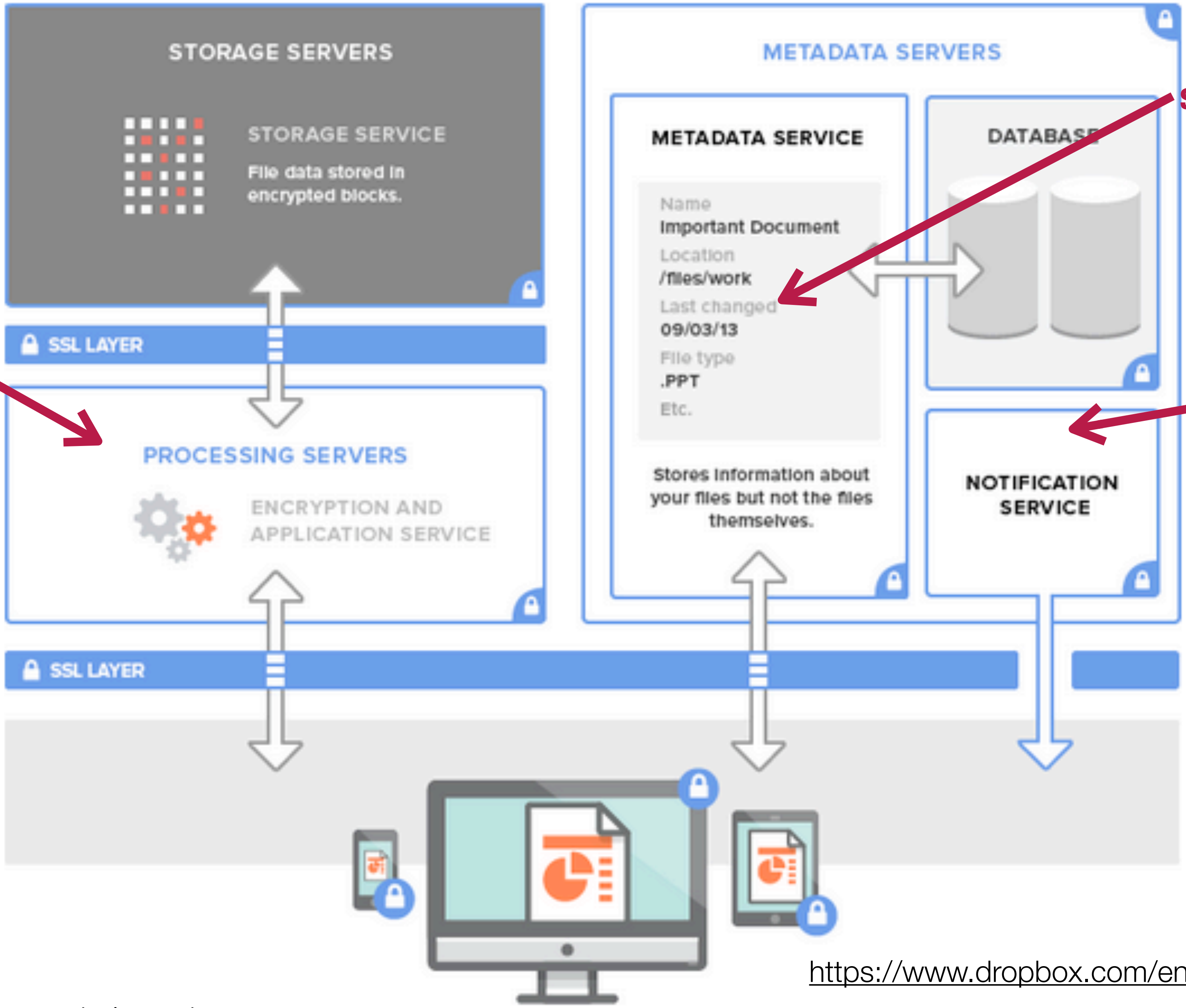
We will build the myDropbox application

- We will build a Dropbox-like app called myDropbox with these features
  - Upload/Download
  - Sharing
  - Syncing X
  - Encryption X

Requires notification-like service

Can simply configure the S3 bucket, so not doing this...

The real Dropbox



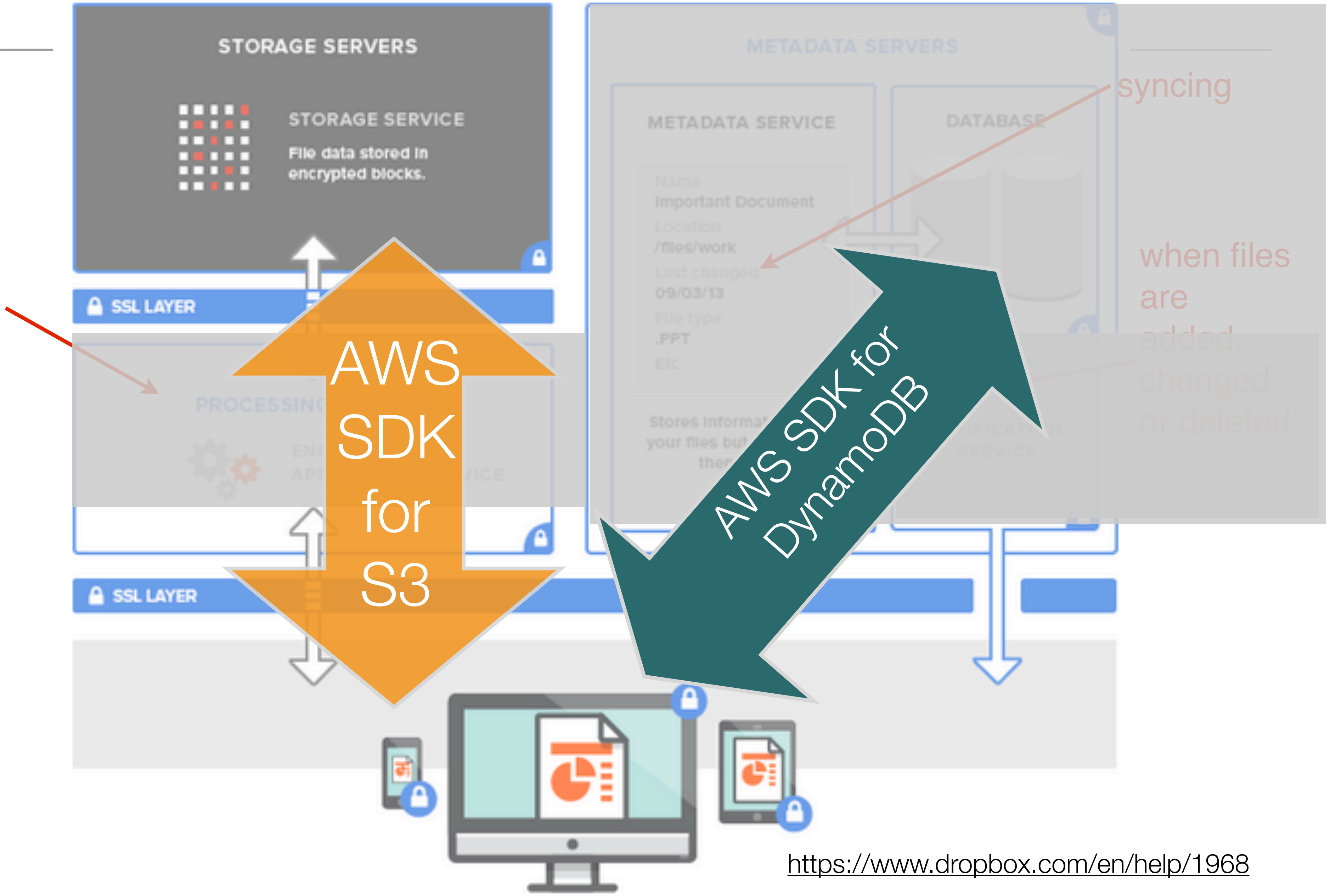
access to files, allow file sharing

syncing

when files are added, changed, or deleted

https://www.dropbox.com/en/help/1968

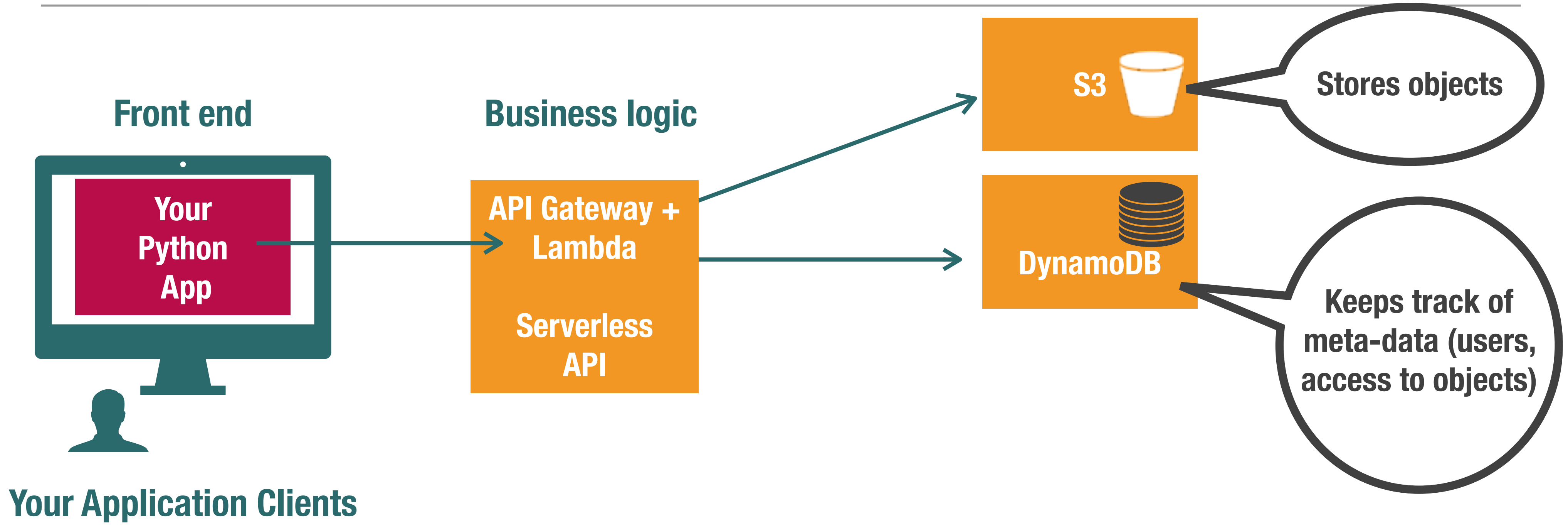
myDropbox



https://www.dropbox.com/en/help/1968

myDropbox

myDropBox Application Back end



What I am expecting: simple command line UI

```
> python myDropbox.py
Welcome to myDropbox Application
=============================================
Please input command (newuser username pass
username password, put filename, get filena
If you want to quit the program just type q
=============================================
>>newuser bob@mail.com password password
OK
>>login bob@mail.com password
OK
>>put tmp.txt
OK
>>view
OK
tmp.txt 5  2016-02-06 14:33:57+00:00 bob@mail.com
tmp2.txt   5  2016-02-06 14:31:57+00:00 bob@mail.com
>>get tmp2.txt bob@mail.com
OK
>>share tmp2.txt alice@mail.com
OK
>>logout
OK
>>quit
======================================================
```

check for existing users; do not allow new user creation if user already exists. passwords need to match.

password needs to match the one used to create new user

show filename, file size, last modified time for files that bob@mail.com owns or files that have been shared with bob

What I am expecting: simple command line UI

```
> python myDropbox.py
Welcome to myDropbox Application
======================================================
Please input command (newuser username password password, login
username password, put filename, get filename, view, or logout).
If you want to quit the program just type quit.
======================================================
>>newuser bob@mail.com password password
OK
>>login bob@mail.com password
OK
>>put tmp.txt
OK
>>view
OK
tmp.txt 5  2016-02-06 14:33:57+00:00 bob@mail.com
tmp2.txt   5  2016-02-06 14:31:57+00:00 bob@mail.com
>>get tmp2.txt bob@mail.com
OK
>>share tmp2.txt alice@mail.com
OK
>>logout
OK
>>quit
======================================================
```

share the object named tmp2.txt with user alice@mail.com

Goals of Assn# 5 myDropbox

- Create a **command-line client** application (to be run on clients' computers) that can wait for a user command: ~~**newuser, login, logout,**~~ **put, view, get,** ~~**share,**~~ **quit**
  - ~~**newuser** create a new user with specified username and password, and store it in a table in DynamoDB~~
  - ~~**login** login to the application, check to make sure that the username and password match with your DynamoDB table~~
  - ~~**logout** logout from the application~~
  - **put** upload one file at a time
  - **view** look at files that users have uploaded themselves
    - ***Make sure users only see objects owned by the user*** ~~***as well as files that others have shared with the user,***~~ ***but no other files.*** You can design your own mechanism for myDropbox user access control. Just make sure that it is embedded directly in your code and it does not require manual manipulation using the AWS web console for S3.
    - Show the following information for each file: filename, lastModifiedDate, size, owner.
  - **get** download one file at a time
    - **Make sure users can get files that are owned by the user** ~~**as well as files that others have shared with the user,**~~ **but not any other file**
    - get filename [username] (where username is the owner of the file; by default the current user is the owner of the file)
  - ~~**share** a file with another user~~
  - **quit** stop using the myDropbox application
- Your client application must not embed/require any of your AWS account access keys/credentials to function.
- Your client should interact directly with your serverless API. Your serverless API should interact directly with S3 ~~and DynamoDB~~ on your client's behalf. If you wish to use pre-signed URLs for S3 (you do not need to do it this way -- it is one of many design choices), your client may contact your S3 bucket directly only after obtaining pre-signed URLs from your Lambda function. If you choose to use pre-signed URLs, also understand the security risks of doing so and try to design a secure solution.

Setting up for Assn #5 (in class)

- You are myDropbox. You have an AWS S3 account that you will use to support multiple users. Your users DO NOT have AWS S3 accounts.
- Your command-line client must be able to be run simultaneously from multiple machines at the same time.
- Manually create a **new bucket** in AWS S3 to which only you (and your IAM roles/users) have access (do not make it public and do not share it with me). You will use this bucket to support all of your myDropbox users.
- ~~Create a new **DynamoDB** table for storing username and password pairs. The table design is given in the following slides.~~

What you need to design for S3

- How your many users will store their files in the same S3 bucket
- How you will know which file belongs to which user
- How you will ensure that if user A uploads a file, and user B uploads another file with the same name, that they are **NOT THE SAME** object in your bucket
- etc.
- **Draw/write up your design as 1-2 slides and include them with your homework submission.**

Lambda functions need to be assigned appropriate roles

- Lambda function should be able to
  - Access (CRUD?) our S3 bucket
  - ~~Access (CRUD?) our DynamoDB table~~
- How do we set this up?
  - Create the S3 bucket, ~~DynamoDB table~~
  - Create the IAM policies (bucket name, ~~table name needed)~~
  - Create IAM role (IAM policy needed)
  - Create Lambda Function (IAM Role needed)
  - Create S3 bucket policy (Lambda function name needed)

When we test…

- Your system must support file sizes up to 10 MB.
- Upload/Download directories
  - Uploads: need to support fully qualified paths
  - Downloads: need to be stored in the current working directory (do not create subdirectories)
- Best practice: open files with least privilege access permissions such as “read only”

Never embed your AWS keys in your source code

- Never embed your AWS keys in your source code (Lambda & client-side app)
- Assign the appropriate least privileged permissions to your Lambda roles in order to give them access to execute and use the required services on AWS

When writing your code...

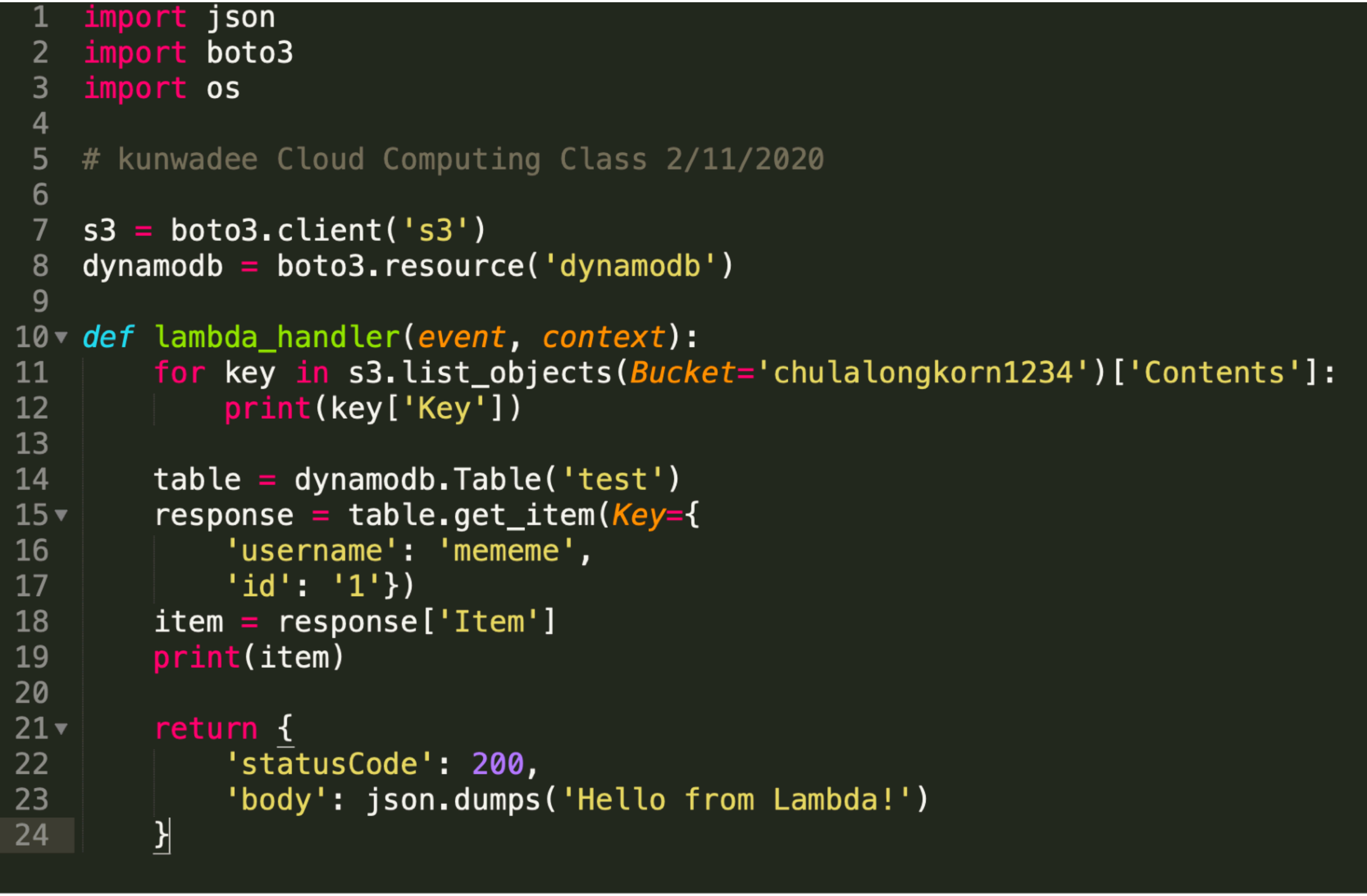
- Python 3.7+ only
- Name your zip file “myDropbox_YourStudentID”
- Make your code readable
- Use descriptive function & variable names
- Each function needs some header that says what it does
- Comment your code so both you and I know what you are trying to do

Submit before the deadline in mcv

- **Create 1 combined .zip file containing the following and upload the .zip to CV**
  - Lambda function (download zip from AWS) named "myDropbox_YourStudentID.py"
  - Source for your python command line client named "myDropboxClient_YourStudentID.py"
  - Your 4-5 page pdf slides (named "myDropbox_YourStudentID.pdf") that describes
    - Your S3 design,
    - ~~Your DynamoDB design,~~
    - A README for all of your code that tells me the name and functionality of each of the source code files you submit,
    - A HOWTO for your API (need to specify http endpoints and request/ response formats)
- **Leave all components, S3, DynamoDB, your deployed serverless API running. We will test your work using your command line client.**

HOW TO use AWS Python SDK with S3 & DynamoDB

- https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
- https://boto3.amazonaws.com/v1/documentation/api/latest/guide/dynamodb.html



```python
import json
import boto3
import os

# kunwadee Cloud Computing Class 2/11/2020

s3 = boto3.client('s3')
dynamodb = boto3.resource('dynamodb')

def lambda_handler(event, context):
    for key in s3.list_objects(Bucket='chulalongkorn1234')['Contents']:
        print(key['Key'])

    table = dynamodb.Table('test')
    response = table.get_item(Key={
        'username': 'mememe',
        'id': '1'})
    item = response['Item']
    print(item)

    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

Set up billing alerts

- https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/free-tier-alarms.html