Takeaway and Challenges in Part 1

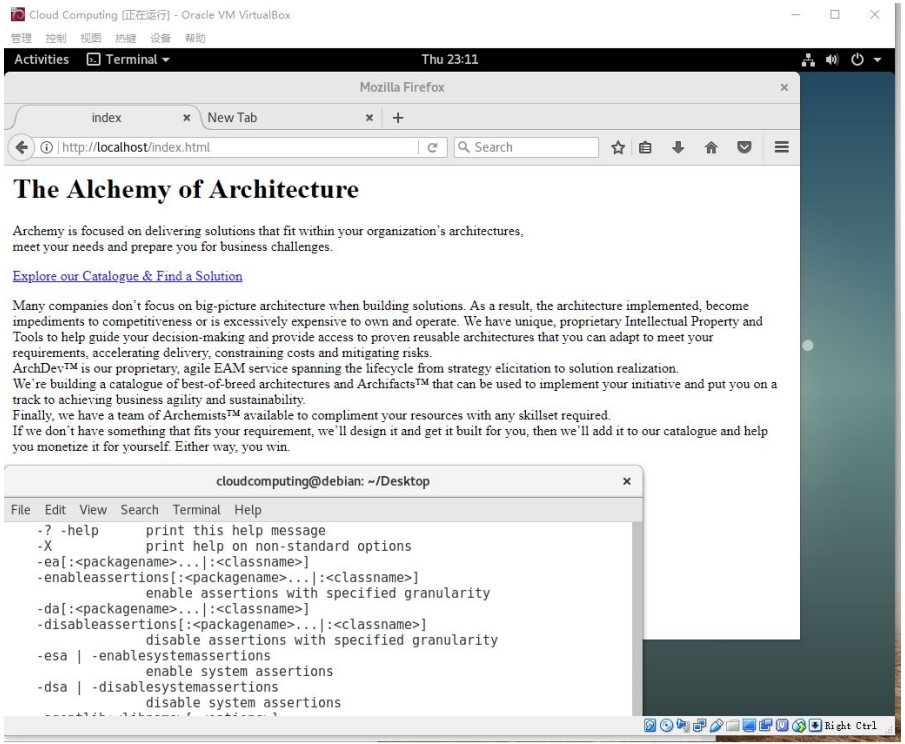
Cloud Computing Final Project

Haoran Ma, Chun-Yi Yang, Lizi Chen, Le Wang

Background

In this project, we use the ArchNav application as a complex business cloudification and migration case study.

We first deployed our ArchNav applications without scalability running on our local environment. After confirming everything is going properly, we decided to migrate our application to cloud computing platform.



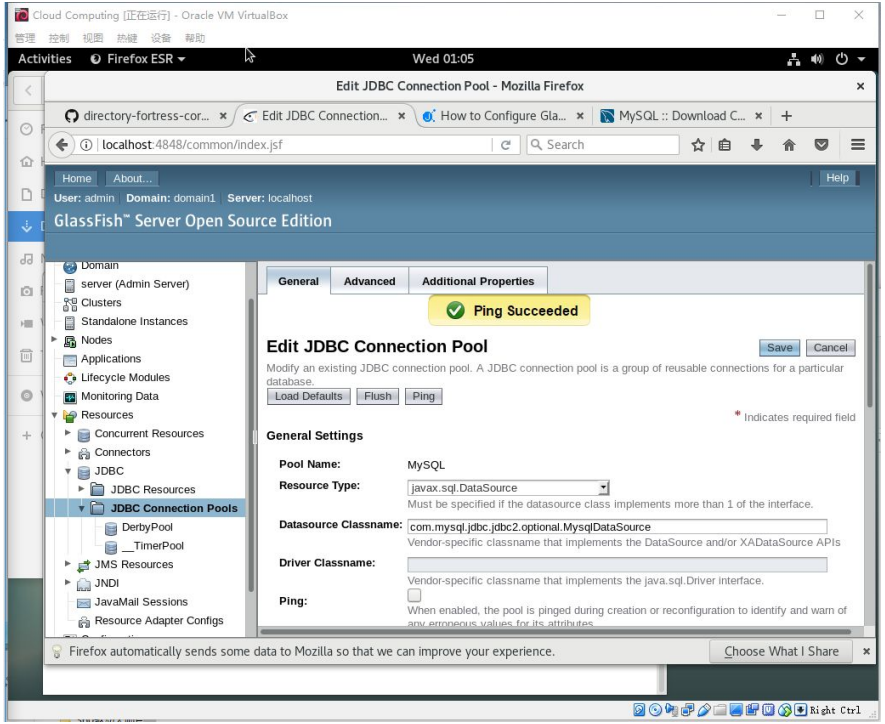
Troubleshooting: MySQL

Challenge #1:

Remote access to MySQL

Challenge #2:

Glassfish cannot work properly with latest MySQL



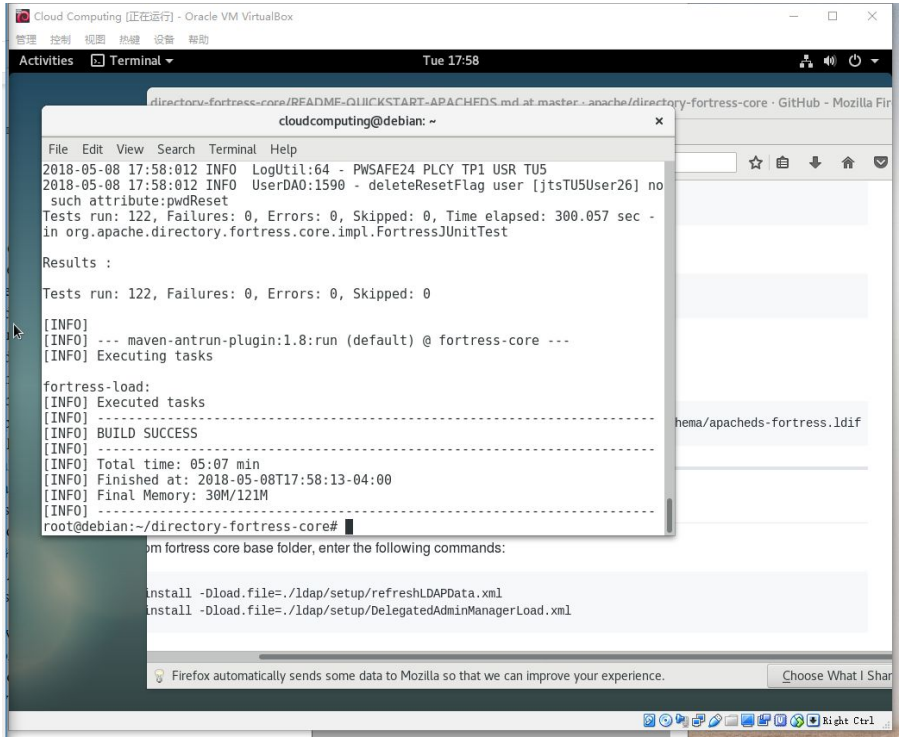
Troubleshooting: Apache

Challenge #1:

The latest version of Apache-fortress-core cannot be deployed on Ubuntu 18

Challenge #2:

Maven 3 needs the latest version of Java (Java 8) while ArchNav needs Java 7



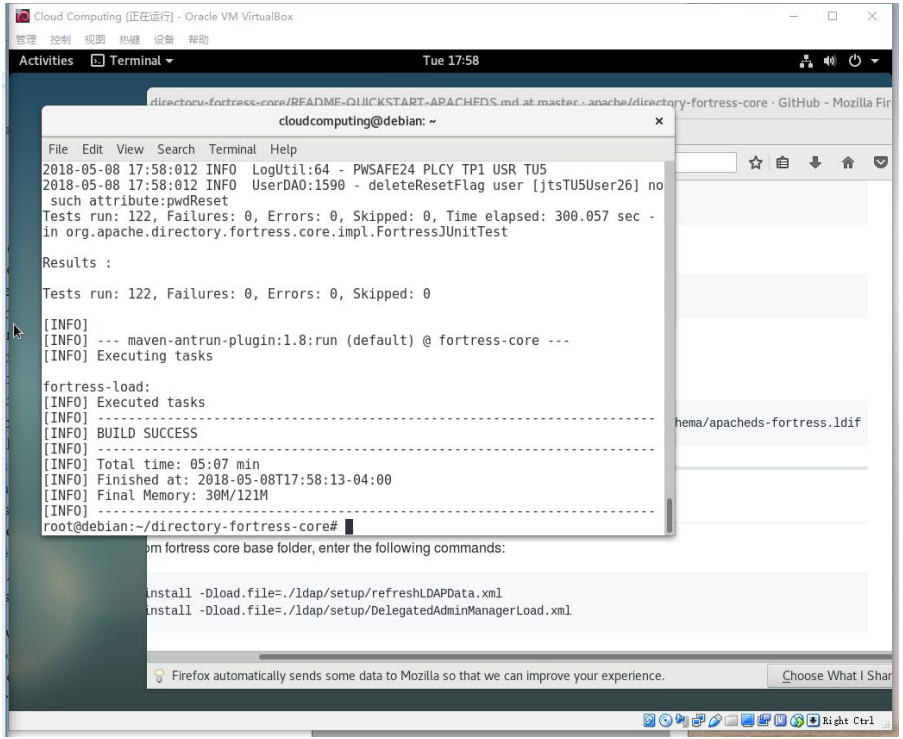
Troubleshooting: Tomcat

Challenge #1:

Tomcat cannot work with default sample security policy download from GitHub

Challenge #2:

There is a fault in Tomcat Instruction Page about shutdown.sh and startup.sh. We should to config init.d and restart the system to load the security policy.



Part 2
Personalized Twitter Audio

Cloud Computing Final Project

Haoran Ma, Chun-Yi Yang, Lizi Chen, Le Wang

Project Proposal

An Application enables users to listen to feeds from friends and channels they followed on social networks (Twitter).

Application Features

- Transfer tweets to audios
  - Eleberated audio that integrates information from twitter timeline
- Update periodically
  - Regular monitoring and scraping in a specific period
- Classify audio
  - A CNN support method to classify content into different category. Ex. sport, business, etc

Presentation Process

Cloud Architecture Design	Machine Learning	Application
• Tools • Services Introduction	• CNN learning on text classification • Model implementation	• Application deployment • Application showcase

Cloud Service - AWS

Amazon EC2

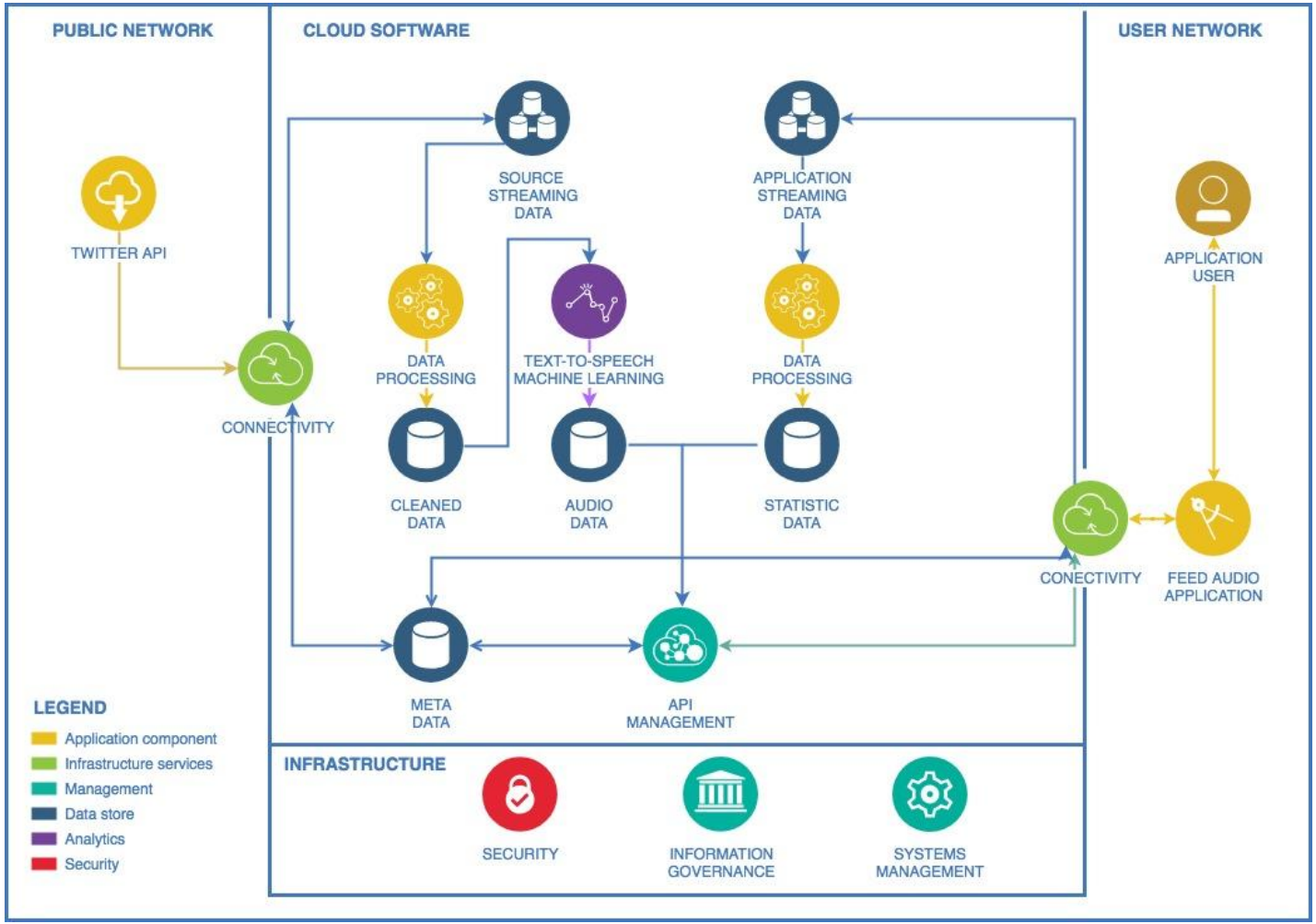
- Virtual computing environments
- Various configurations of CPU, memory, storage, and networking capacity
- Static IPv4 addresses for dynamic cloud computing

Amazon DynamoDB

- A NoSQL database service,
- Automatic data replication over three zones
- Designed for massive scalability
- DynamoDB delivers highly predictable performance

Amazon S3

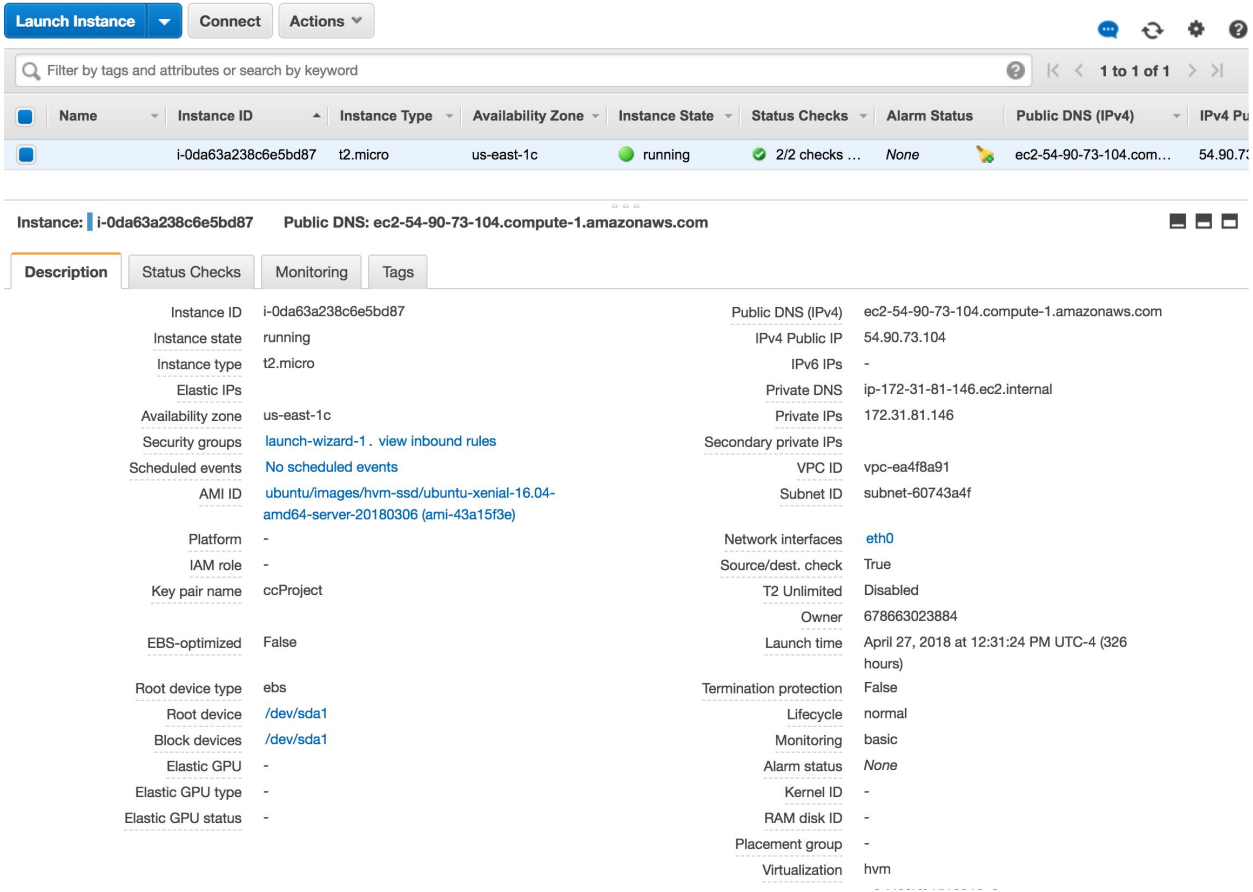
- An online storage service
- Durability, availability and scalability
- Comprehensive security and compliance capability
- Flexible management, easy data transfer


PUBLIC NETWORK
CLOUD SOFTWARE
USER NETWORK
TWITTER API
SOURCE STREAMING DATA
APPLICATION STREAMING DATA
APPLICATION USER
CONNECTIVITY
DATA PROCESSING
TEXT-TO-SPEECH MACHINE LEARNING
DATA PROCESSING
CLEANED DATA
AUDIO DATA
STATISTIC DATA
CONECTIVITY
FEED AUDIO APPLICATION
META DATA
API MANAGEMENT
LEGEND
Application component
Infrastructure services
Management
Data store
Analytics
Security
INFRASTRUCTURE
SECURITY
INFORMATION GOVERNANCE
SYSTEMS MANAGEMENT

EC2

Ubuntu 16.04-amd64-server

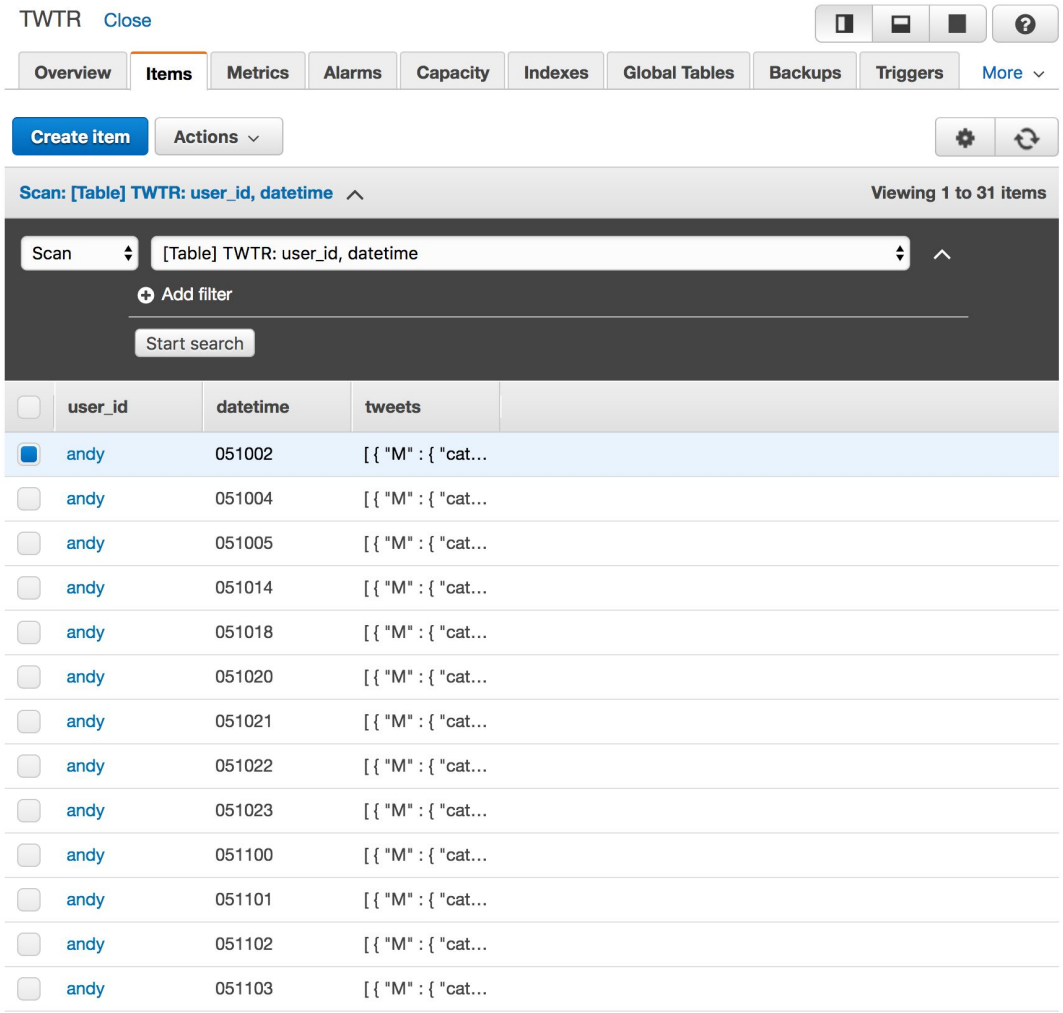
t2.micro



Dynamodb

Twitter Schema

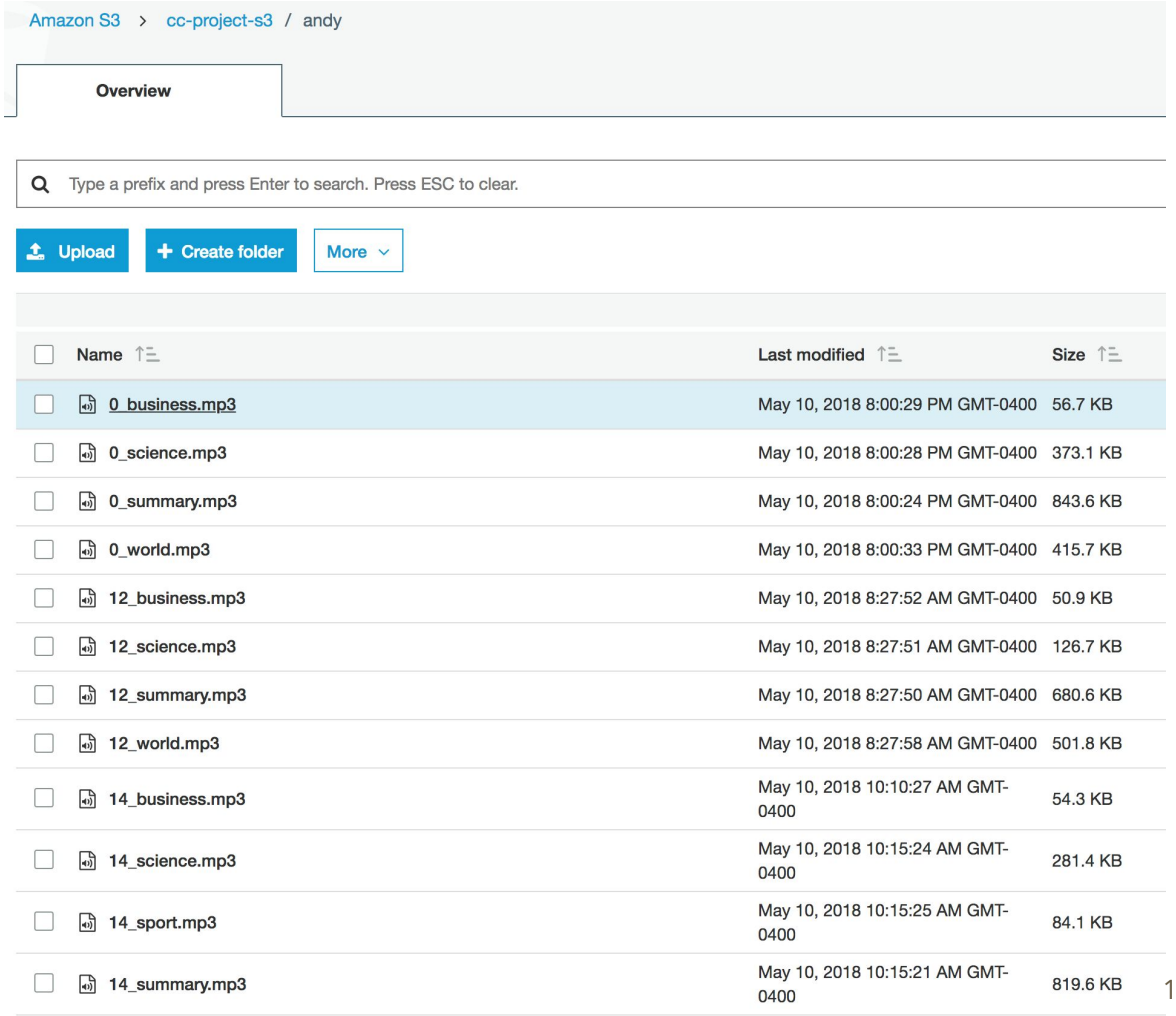
```
[
  "user_id":"andy",
  "datetime":"051002"  // MONTH/DAY/HOUR
  "tweets":[
      {
        "category":"0",
        "created_at":"Thu May 10 02:45:02 +0000 2018",
        "favorate_count":"3",
        "quote_count":"6",
        "reply_count":"2",
        "retweet_count":"10",
        "screenNanme":"The Wall Street Journal",
        "text":"David Mayman has helped make sci-fi a
        "tweet_id":"994407939959148546"
      },
      ...
    ] <!--END OF TWEEt LIST-->
      ...
  } <!--END OF USER LISt-->
]
```



S3

Bucket: cc-project-s3

Folder: user_id / time_category



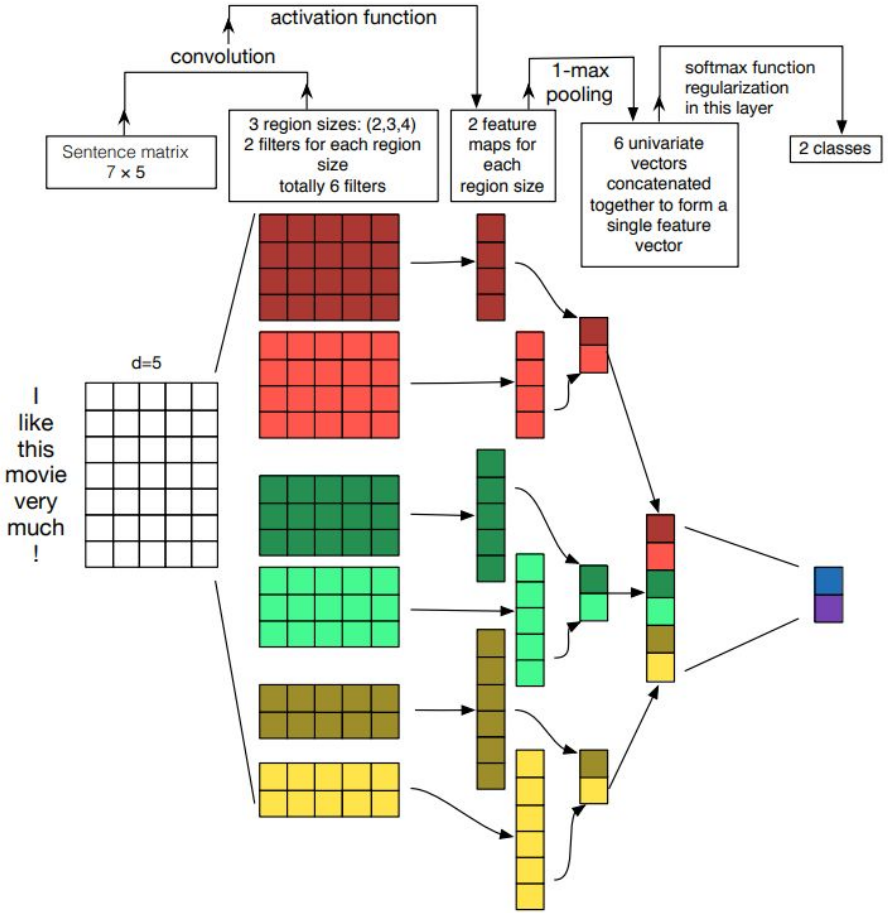
News Classifier

- World
- Sports
- Business
- Sci/Tech

- 300,000 Training set
- 100,000 Testing set

- Precision: 92%
- Recall: 91%

Further Improvement:
Use Tensorflow Serving as a constant running service.



Deployment on AWS EC2

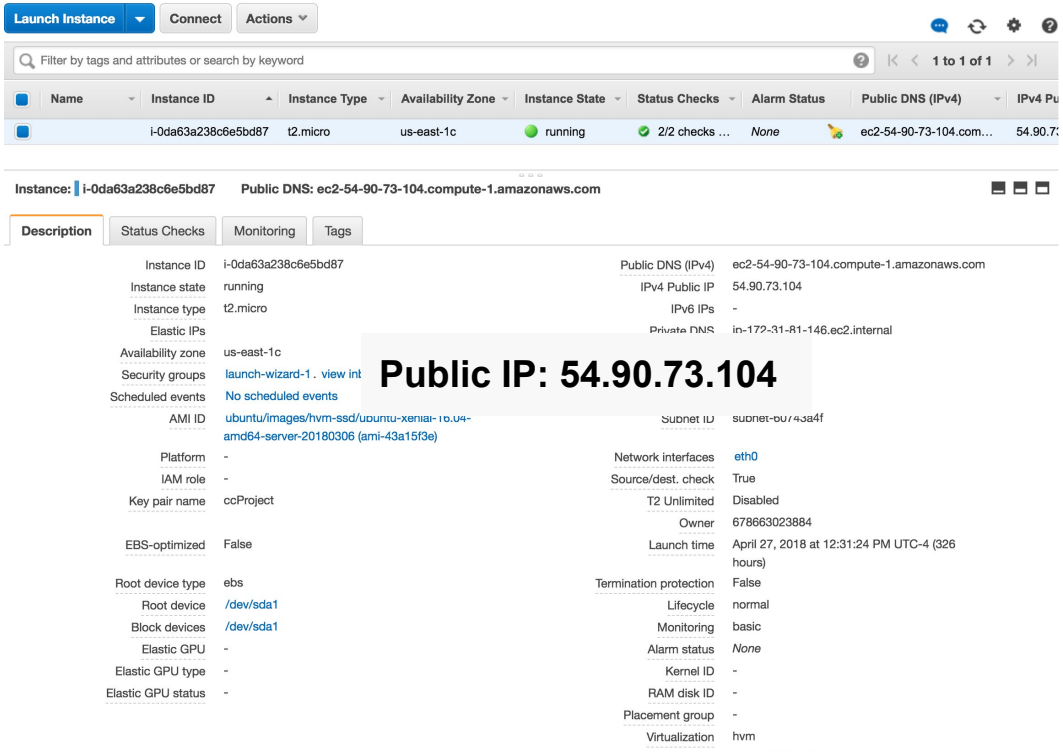
1. Setup EC2 on AWS
2. Connect to EC2

Code sample for flask deployment

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=80)
```



Demo