**Project 1:**

**Student ID: 012482392 Student Name: Hemambujam Veeraraghavan**

**Movie Recommendation System**

The application I have developed is a library of movies for movie buffs called “**Movie recommendation system**”. This is a personalized 3 tier web application which provides a list of top rated movies on different genres which the user can browse and have their own personalized review list. This is more of a personally inspired idea as most of us would just browse around for a good movie to rent and before you realize, it would already be mid-night!

This application would list the top-rated movies based on the public votes over the internet from IMDB.com and segregate based on different genres, for the users’ perusal like comedy, horror, drama, etc. Thus, saving a lot of time by narrowing down their search to the specific genres.

Apart from being a web application of movie lists, this also has a customized login where the user can, not just browse through the top-rated movies, but also write reviews and have them uploaded under their personal account. This feature comes in handy when the users aren’t sure if they had watched the movie already. The users can list their reviews and check for themselves if they had watched it already before renting the movie over the internet, thus saving money.

The users can also download their reviews and reminisce their take on those movies they had watched. And if they happened to have a change of opinion, the application allows them to delete their review and upload a fresh one instead.

**Technologies Used:**

1. Coding Language: PHP 5.5.9-1ubuntu4.22 (cli)
2. Webserver: Apache 2.4
3. AWS Cloud Service features:
   * EC2
   * ELB
   * Lambda
   * AutoScaling Group
   * RDS
   * CloudFront
   * S3
   * S3 Transfer Acceleration
   * R53
   * ElastiCache (Optional)
   * CloudWatch
   * SNS
4. HTML, CSS, JSON/.

**Feature List:**

-A web application that allow users to upload their reviews in a file format with a maximum size of 10MB per file and have them stored in the cloud.

- Allow users to browse their uploaded movie review files in the home page with minimal latency.

- A updated list of the all the files which also allow users to update details, if required.

- Ability to delete their files from the web portal which will reflect in their cloud storage instantly.

- A quick-view feature to glance through to the already uploaded files instead of downloading them.

- Allow users to download the files as and when required.

- Help cloud administrators to keep track of the files using alarms and emails/phone number.

- Monitoring service for all the cloud components along with their all logs.

- Enabling cloud storage in multiple locations (Availability Zones) which helps in data redundancy and retrieval.

- Configure the cloud storage with different types of archived storage based on the frequency of data usage and confidentialities. This is in-build in AWS S3 storage.

- Deploy the entire application in cost effective and auto-scaling cloud compute with proper load balance ti manage millions of user requests and auto recovery.

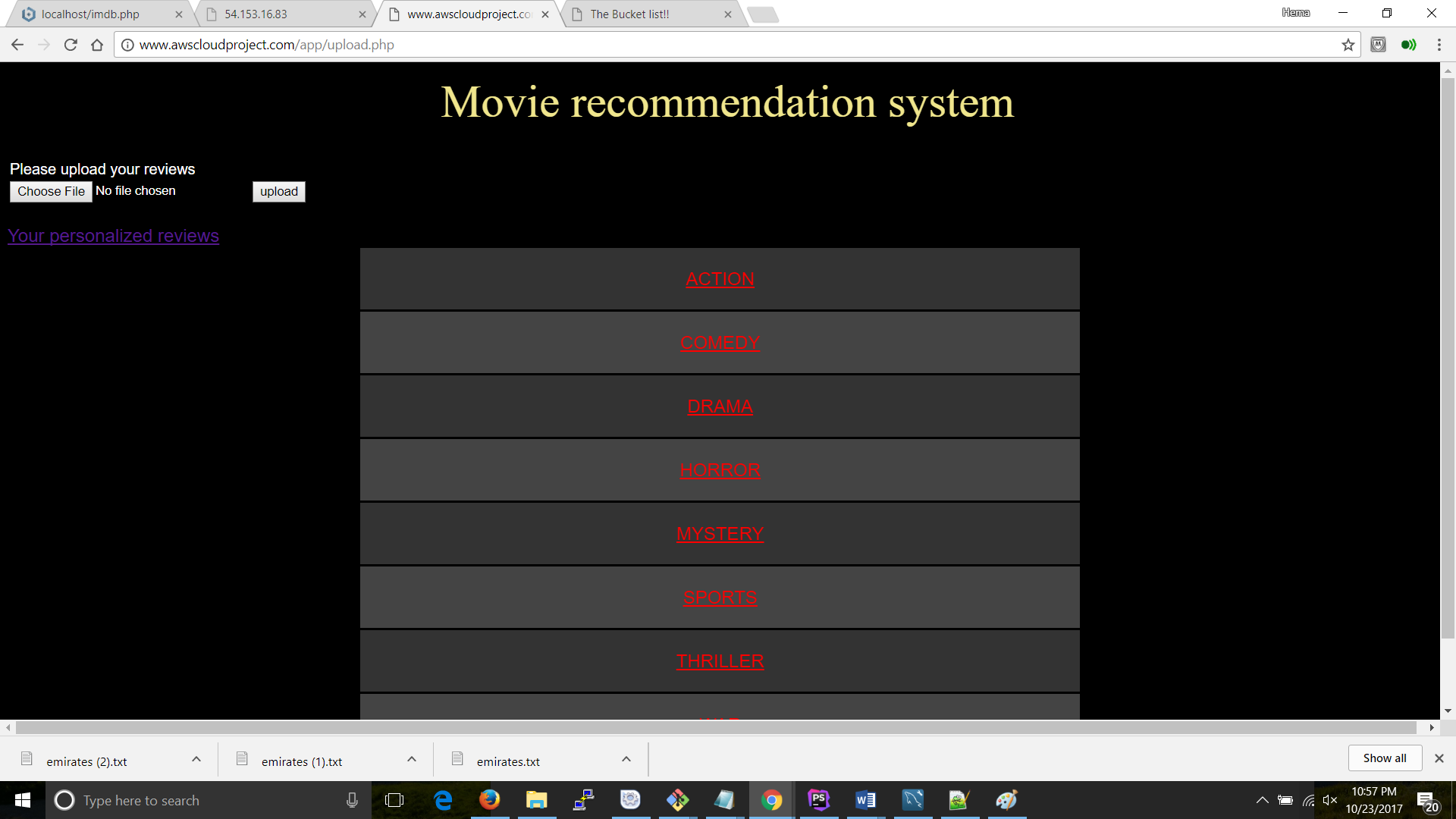
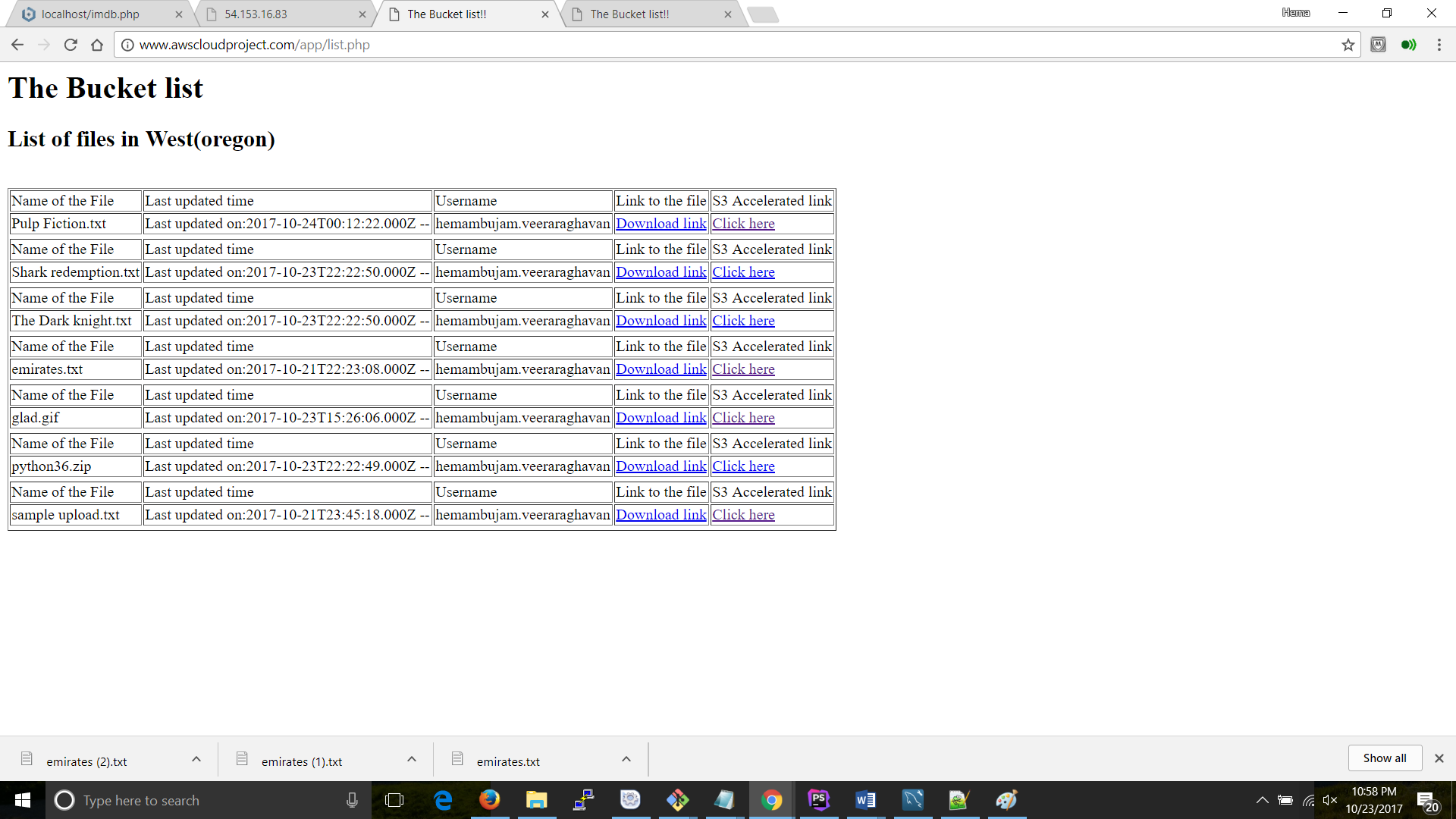
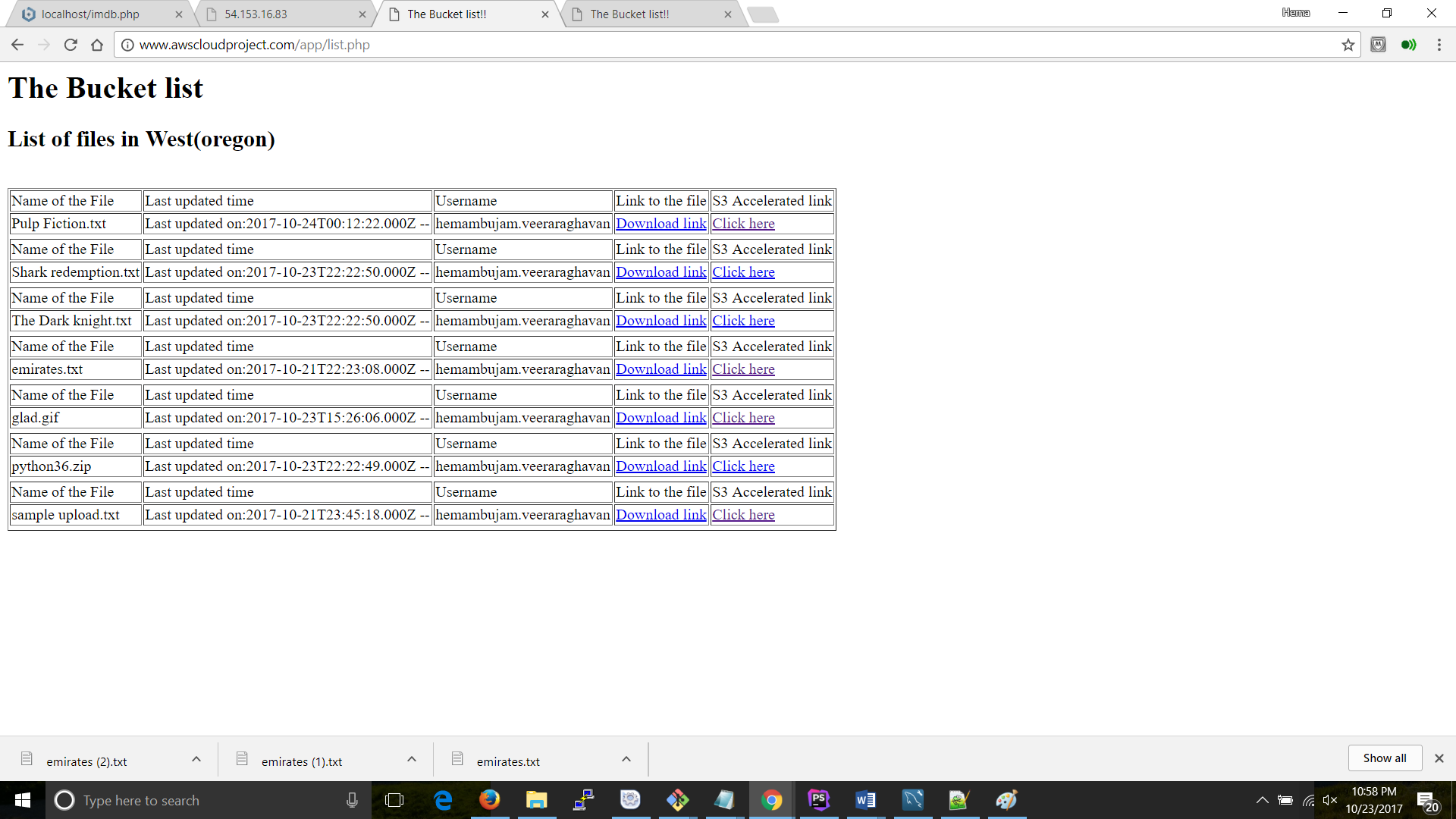
- Register this application with a Domain Name to make it available in public internet.

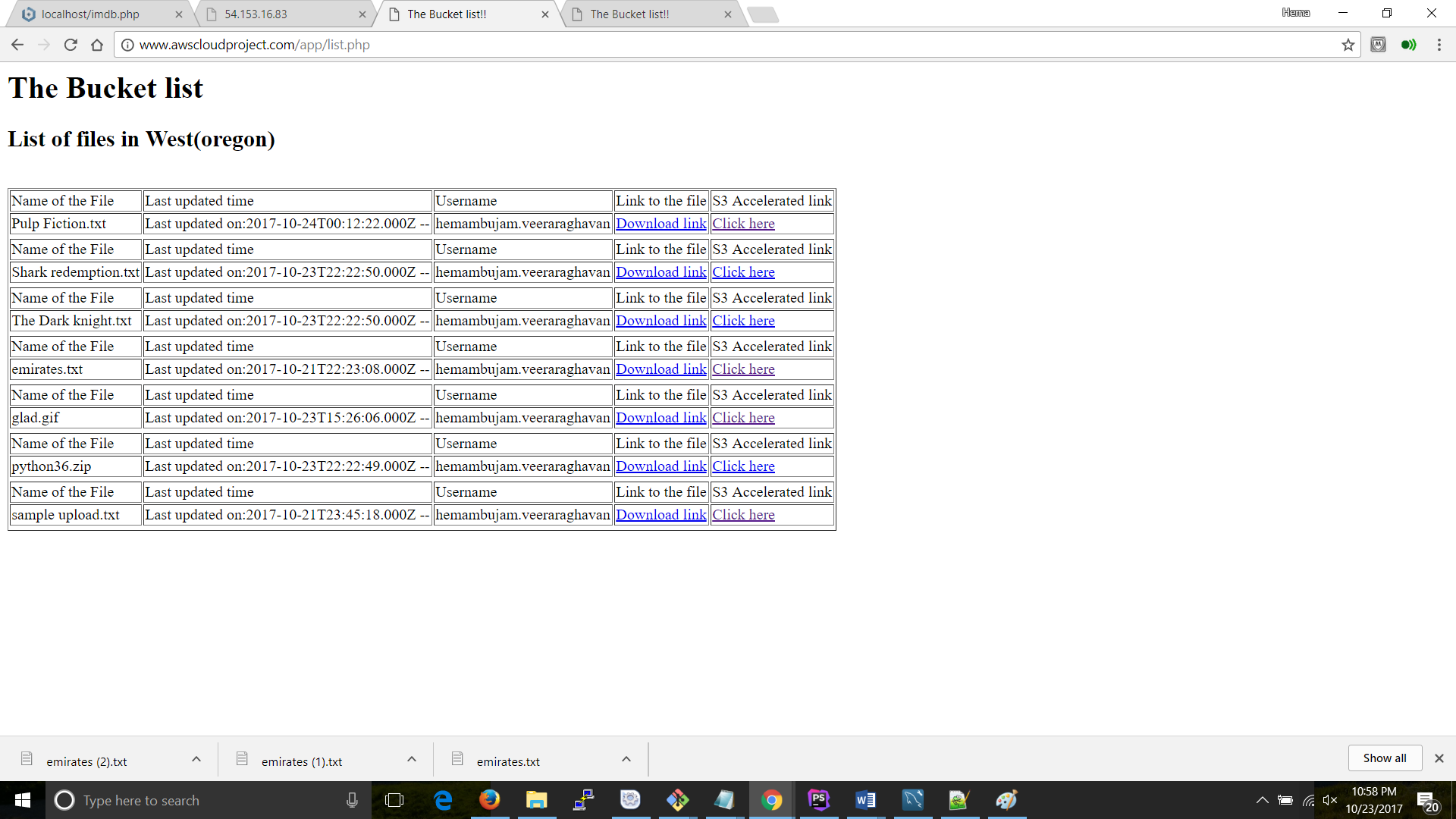
- Throughout the development and production, track the health of the application and notify the admin in case of emergency and alarm the situation for safety.

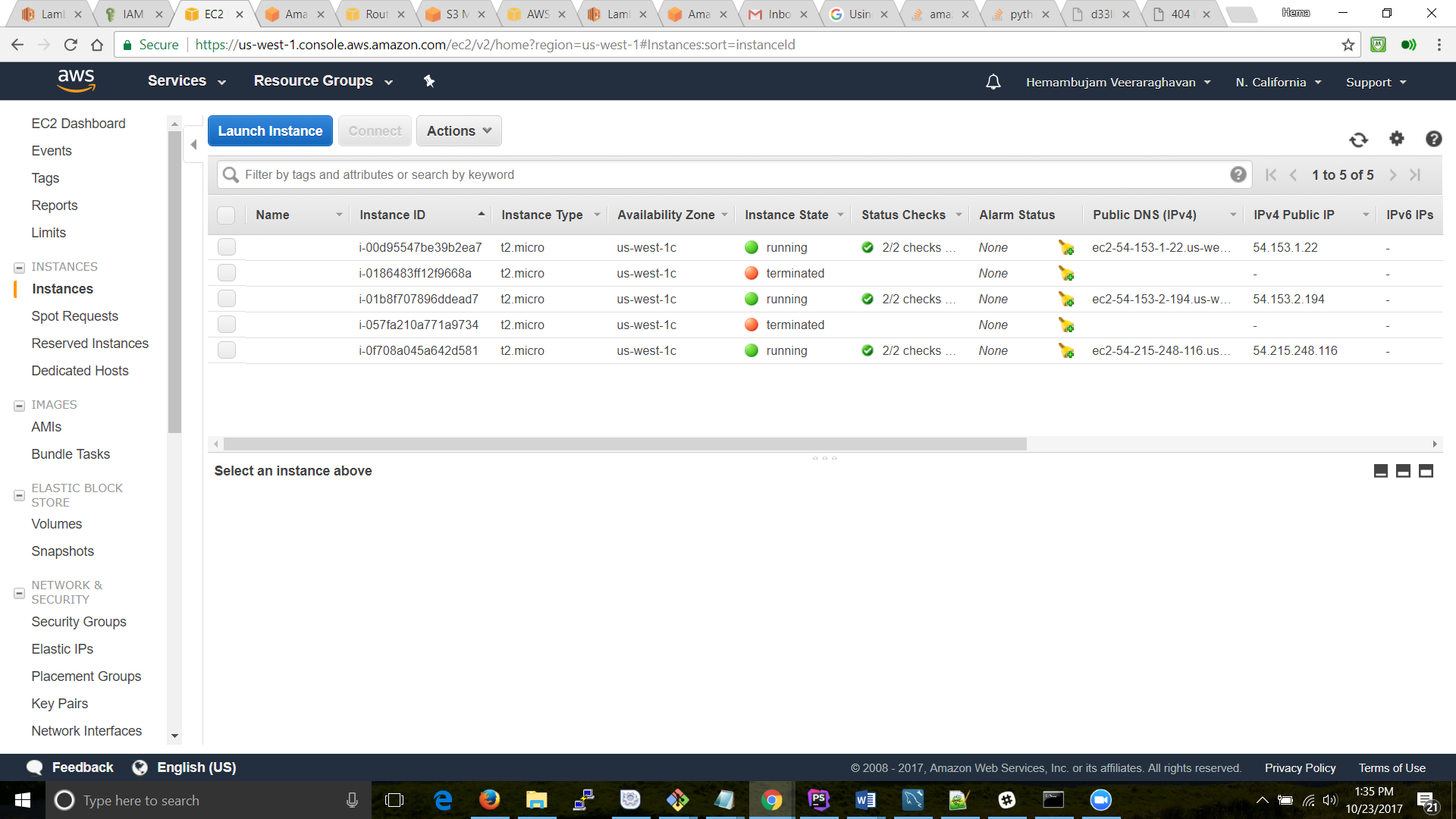
**Solution list:**

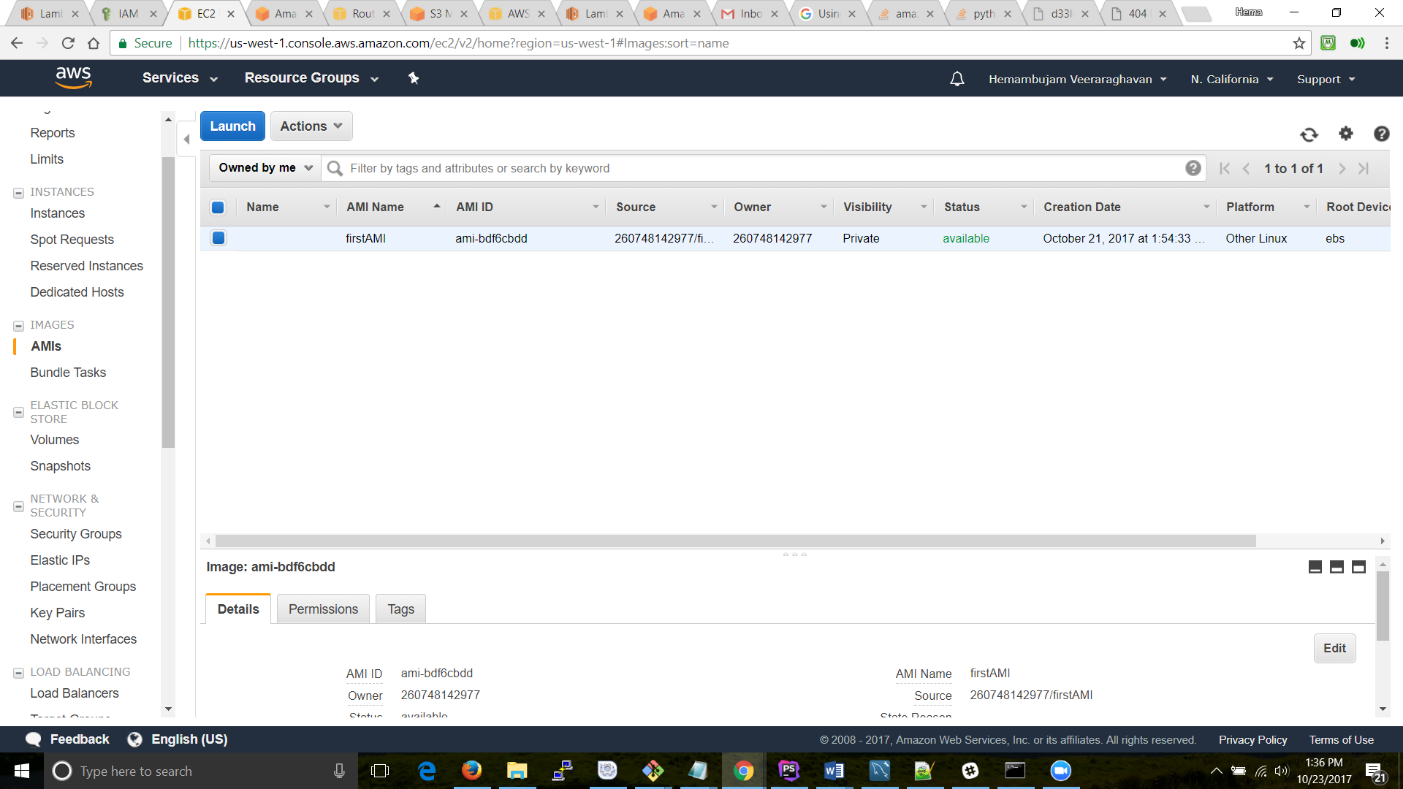
Below are the list of problem statements from the project requirements along with the solutions

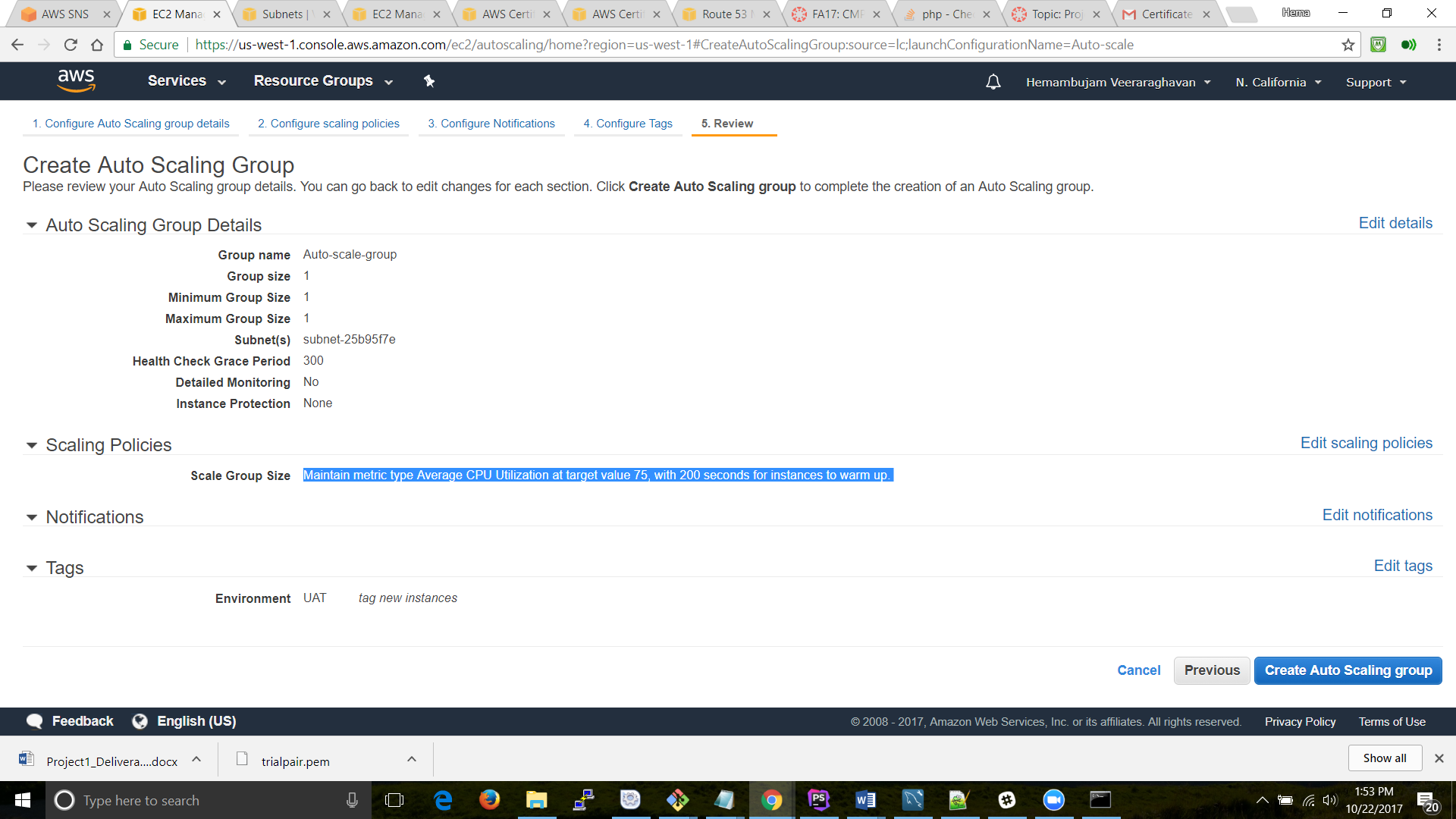
The web application allowing authorized users to perform CURD operations:

* + The upload button helps the users to upload new files
  + The Below screenshot allows the users to browse through already uploaded list of files with each record having a URL to download the fie. 
  + The S3 Accelerated link to have a quick glance on the content of the file: 
  + For each file upload, application tracks File name, the Username and the file updated time, along with a link to download the file:

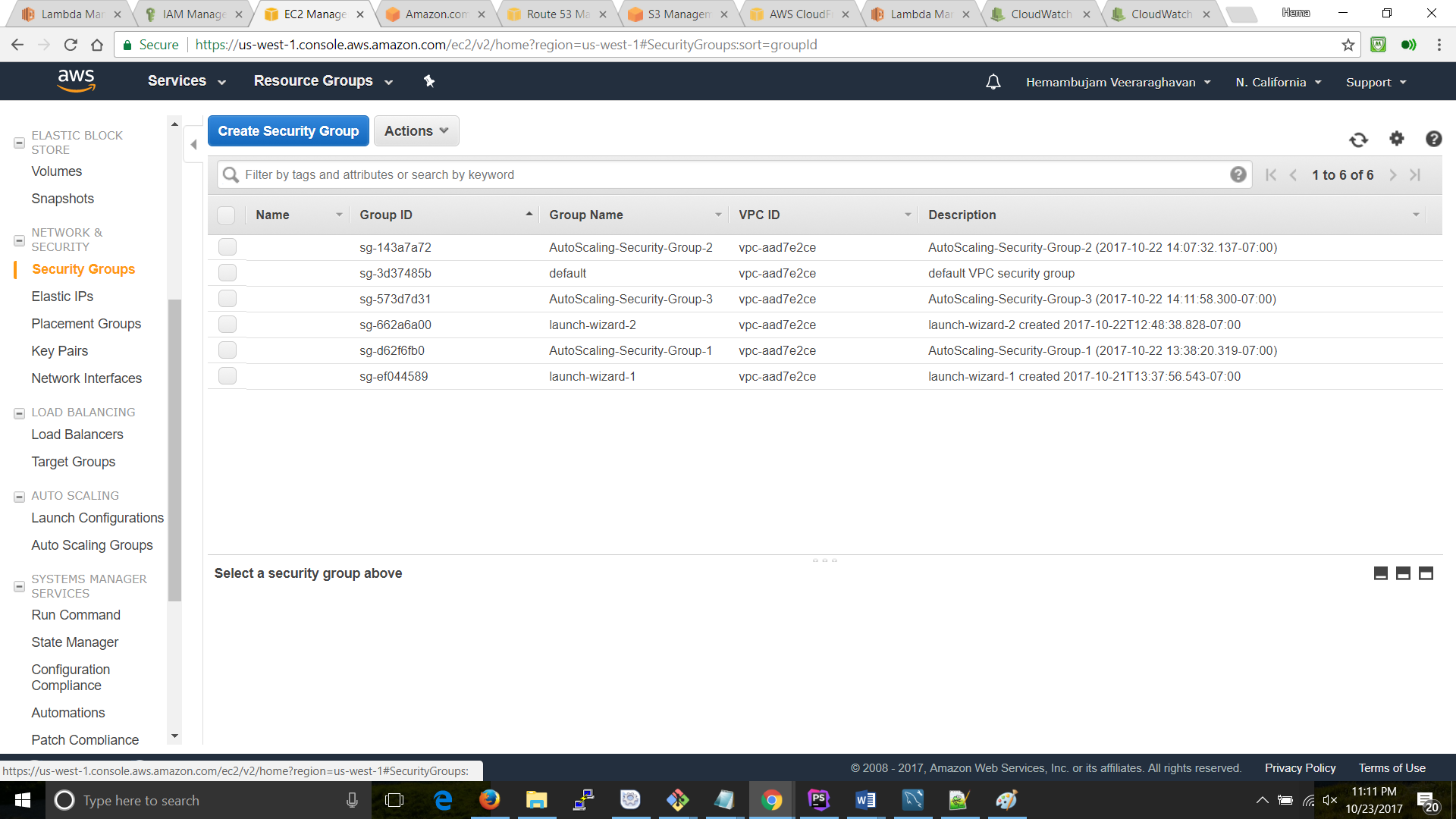


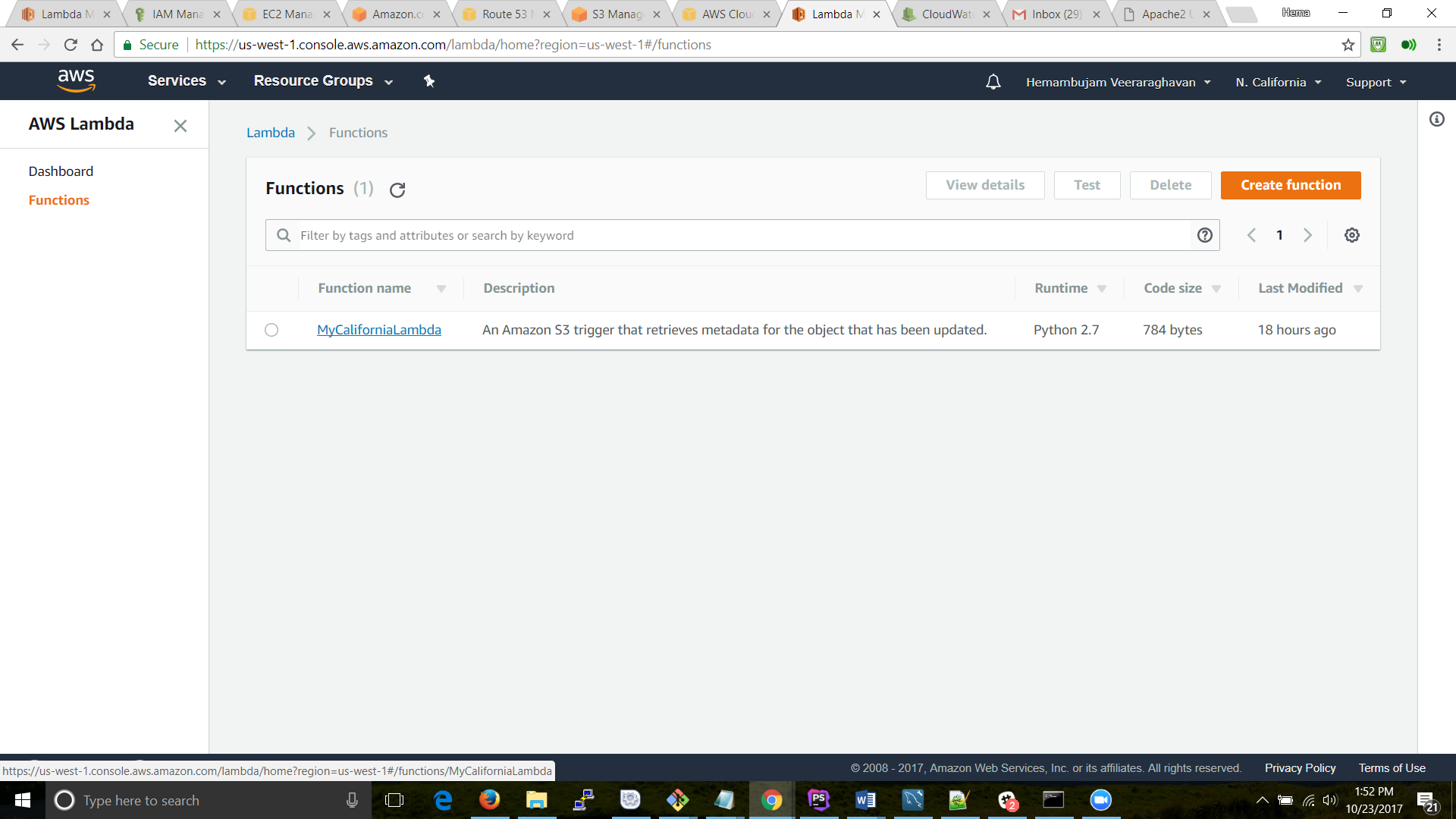
* The architecture diagram depicting the architecture of your project and all appropriate solution components.
* This project leveraged the following AWS services:
  + **EC2:** The entire application deployments rest at the EC2 instance: 
  + Auto-Scaling Group:

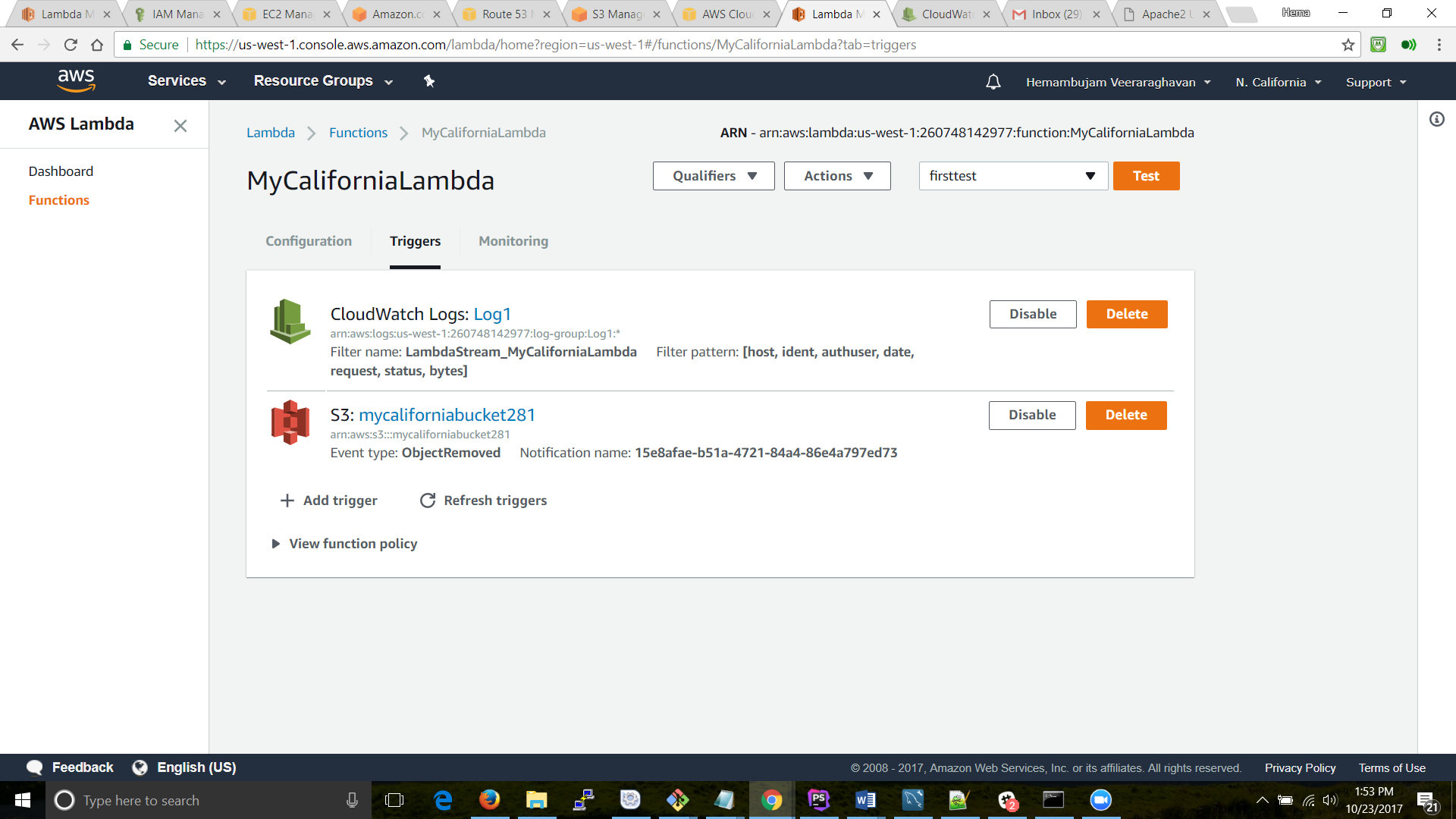
In order to create an auto scaling group, we’ll first have to have an AMI of the existing parent instance and use that image to spin up the new instance: 

This way we save a lot of time in terms of deploying the application on the newly spun instances. Also there are a the required ports are to be opened to be implements for these instnaces: 

Along with these, we will also have to define our security policies for these instances:



* Lambda: a serverless compute service that runs the code in response to events and automatically manages the underlying compute resources. For this application I had created a function in Lambda:

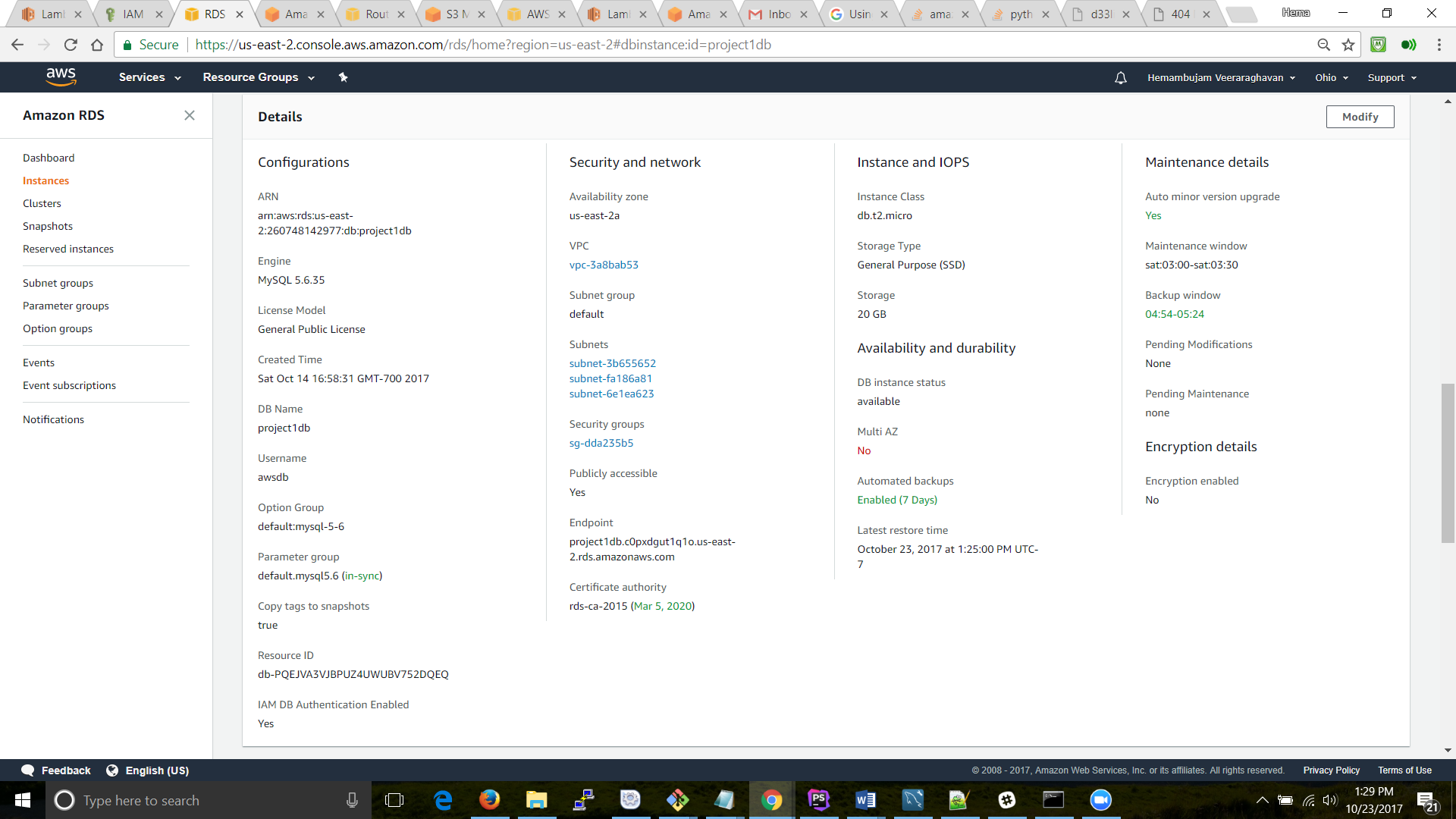
The functions in Lambda are set to send out triggers for the following: 

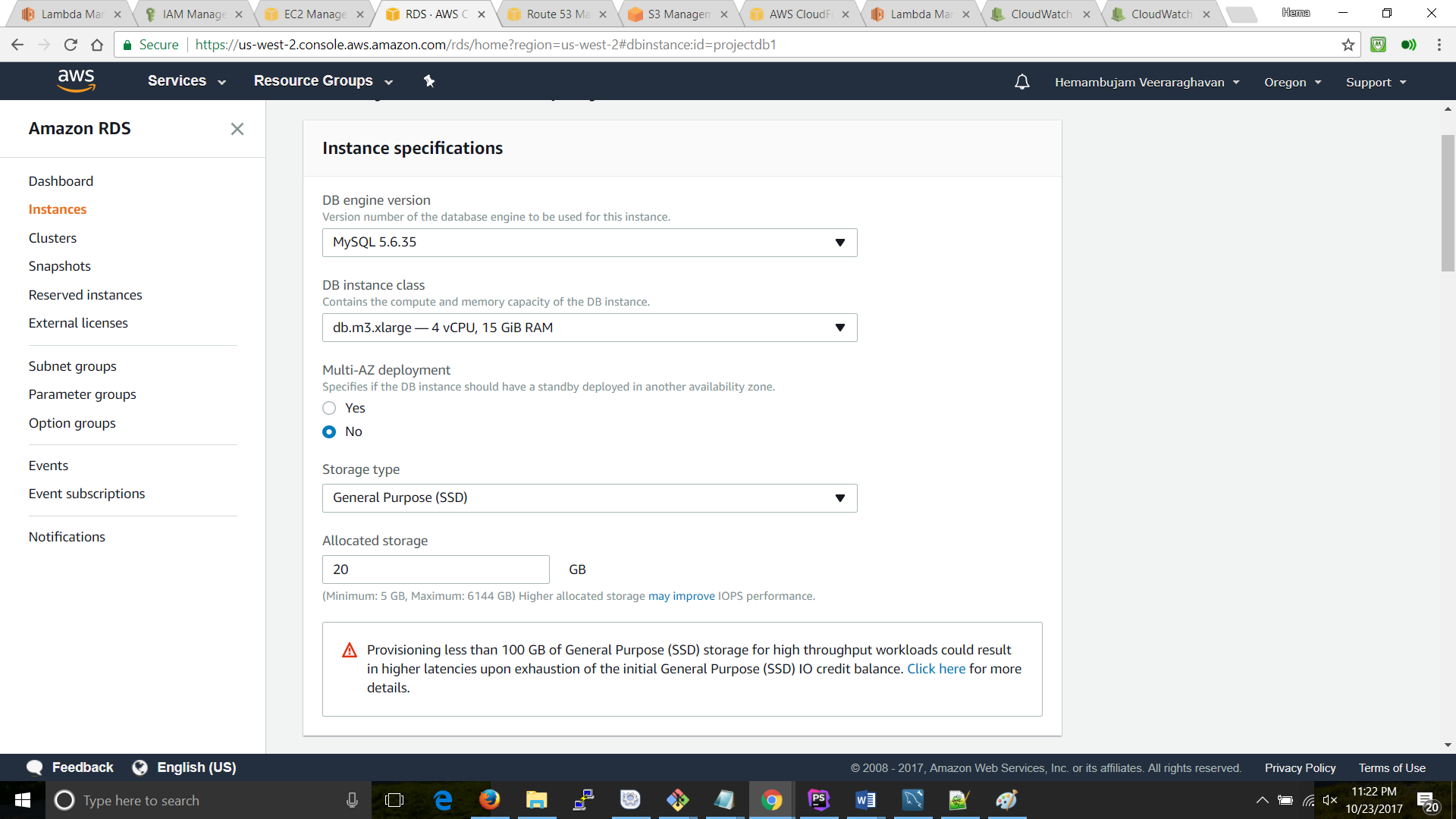
* + ELB:

**Elastic Load Balancing** , as the name suggests balancers the incoming load to the application servers that are hosted in multiple Availability Zones. I have configure health checks on these instances to monitor their health instances so that the load balancer can send requests only to the healthy instances: 

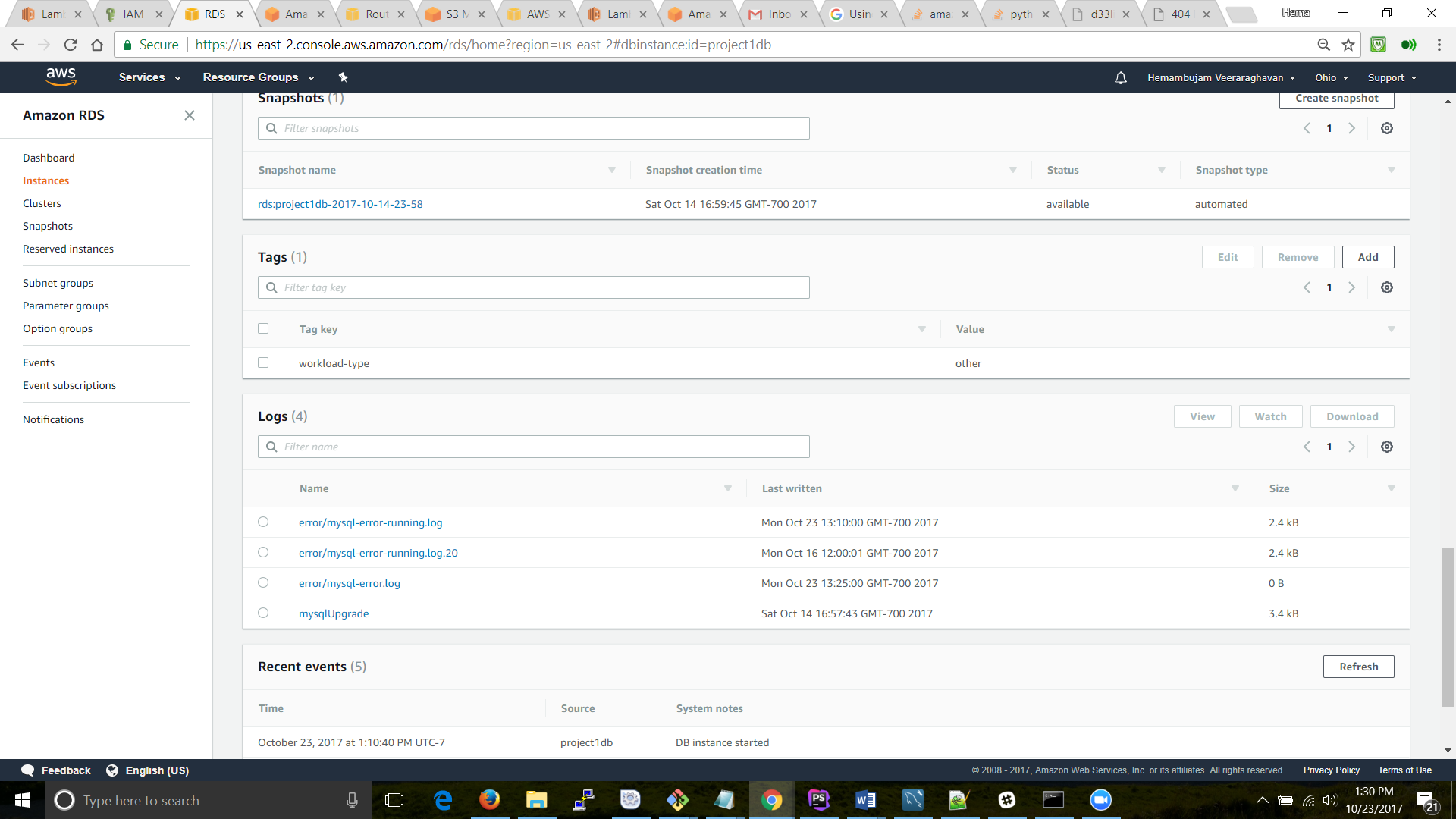
* + RDS:

Amazon **Relational Database Service** (**RDS**) is **a web service** that is used to set up, operate, and scale a relational databases in the cloud. RDS can spin up a number of flavors of DB such as Aurora, Oracle and Mysql. It provides an extremely cost-efficient metho to manage common database administration tasks. (Describe what steps you can take to covert the single AZ DB into multi-AZ deployment as part of your project deliverable):

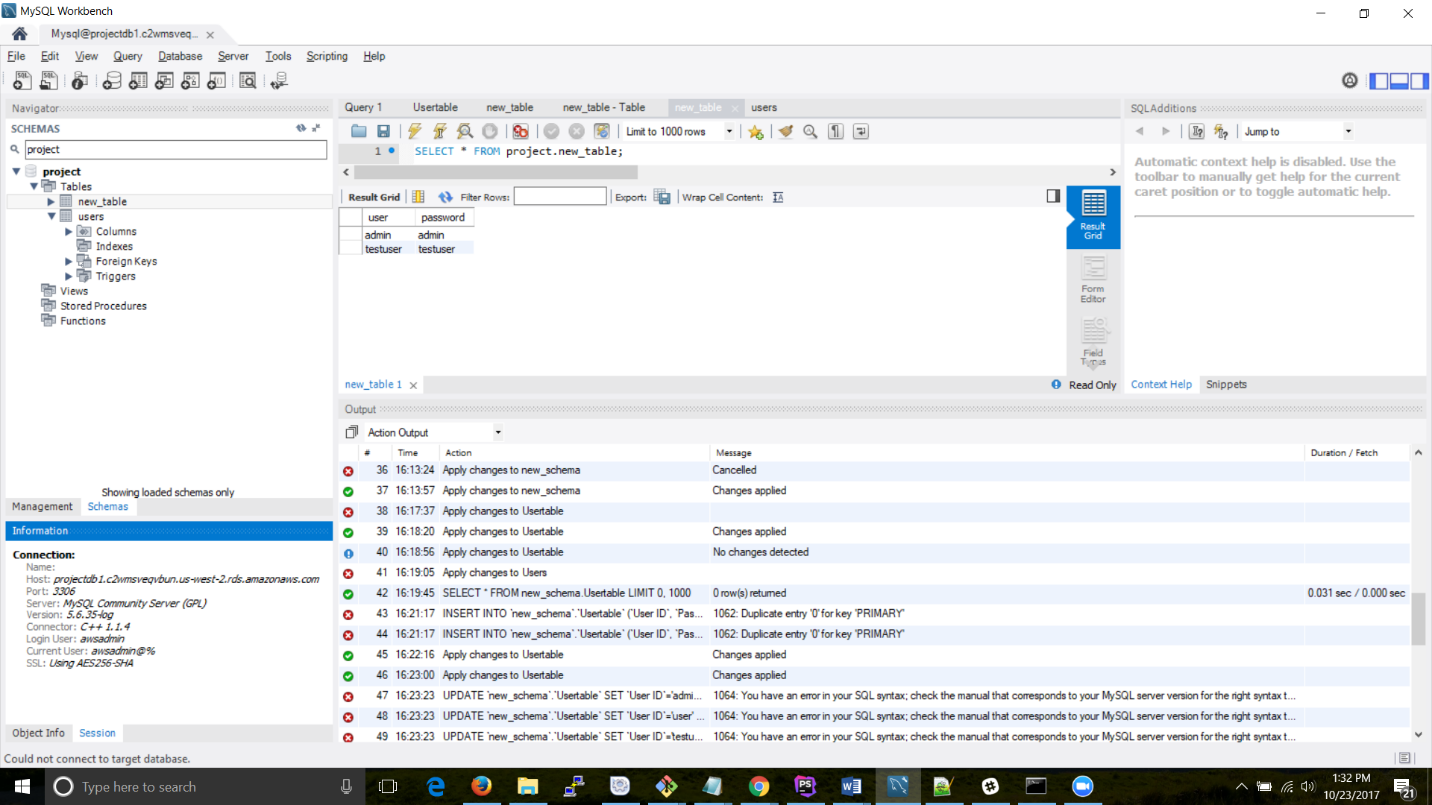


The configuration shows “NO” under Multi AZ. Inorder to convert a single AZ into a multi -AZ, we can modify the instance configuration to host multi-AZs: 

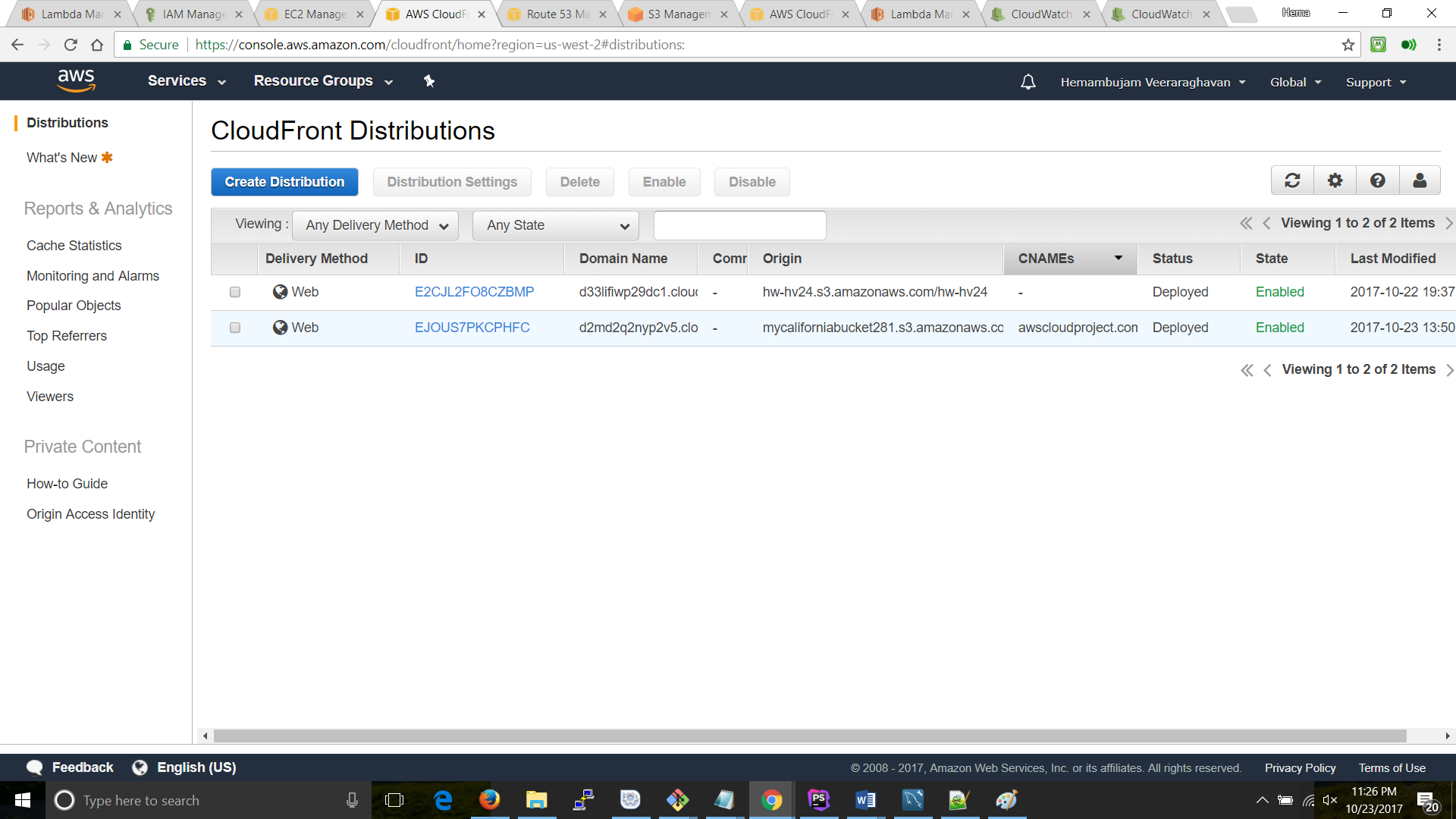
We can also take snapshorts of the data for data redundancy and in times of data loss, we can bring up the DB using the snapshots:



The query is put into the DB using a mysql workbench:

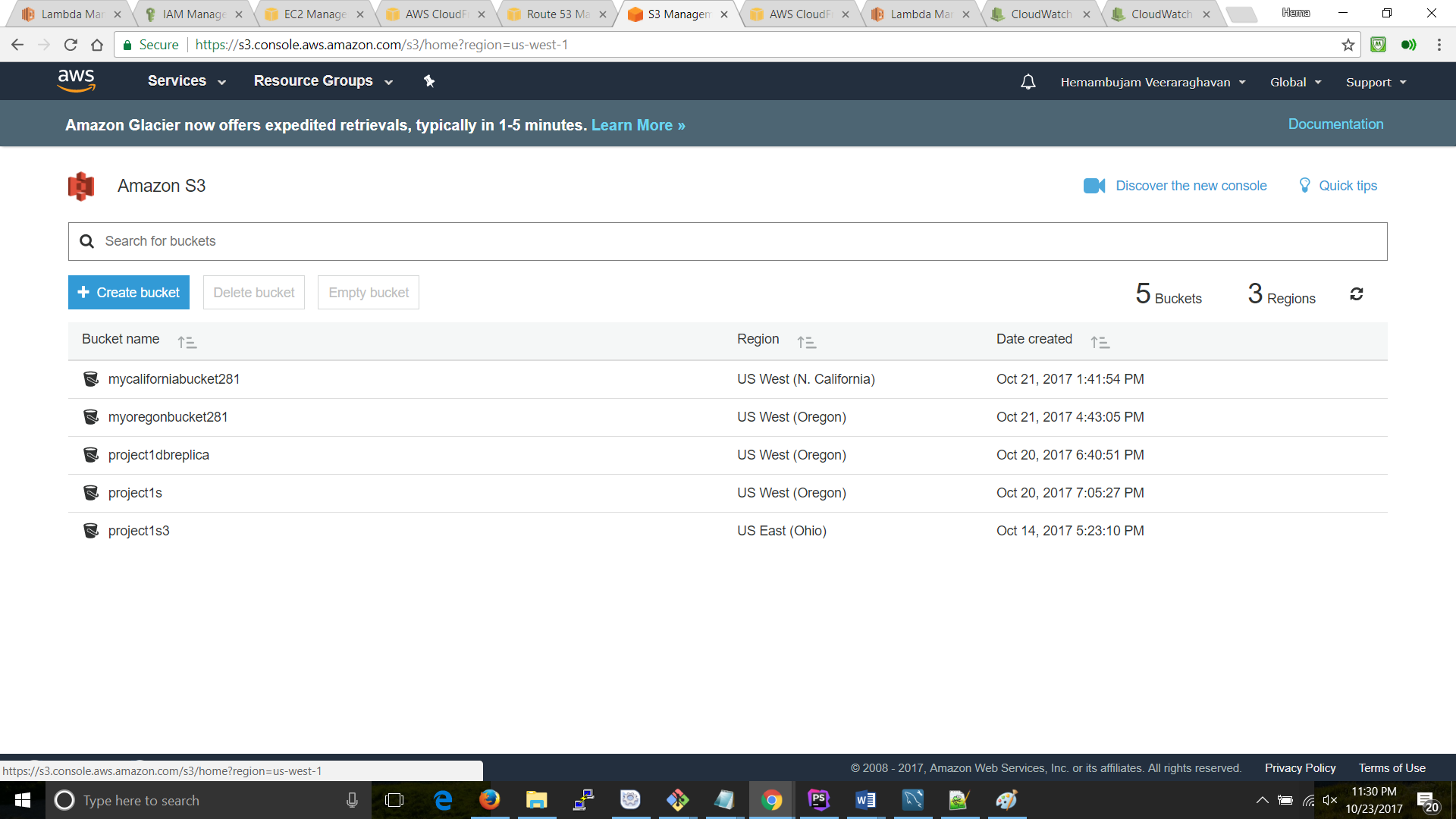


* + CloudFront:

This is a content delivery feature in AWS enabling the servers to cache content and provide them with very minimal latency: 

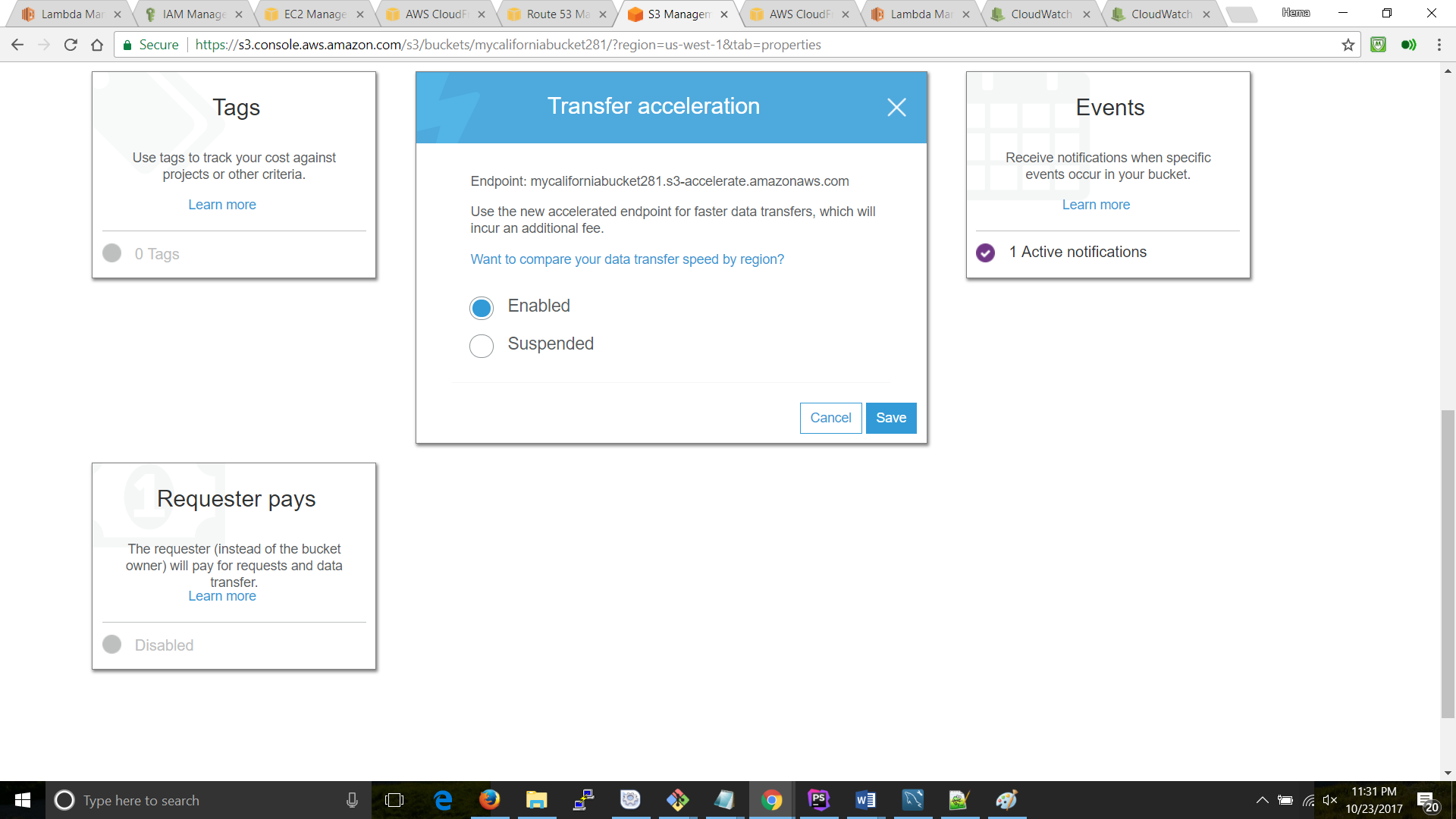
* + S3:

Simple Storage Service has a number of features including storing data, web hosting etc:



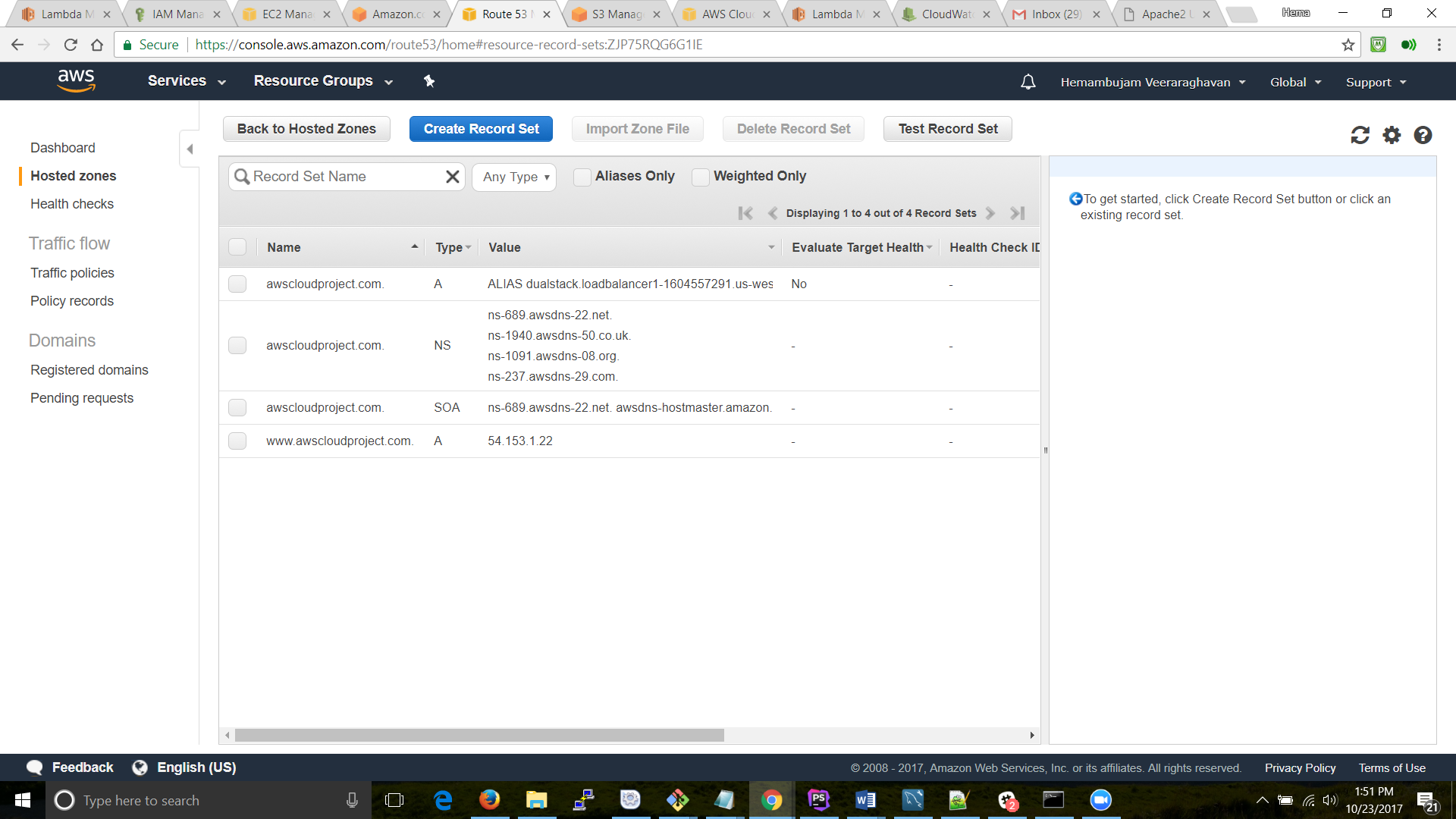
* + S3 Transfer Acceleration:

Under the properties of the bucket, the Transfer acceleration is enabled. Using the new accelerated endpoint there will be faster data transfers.



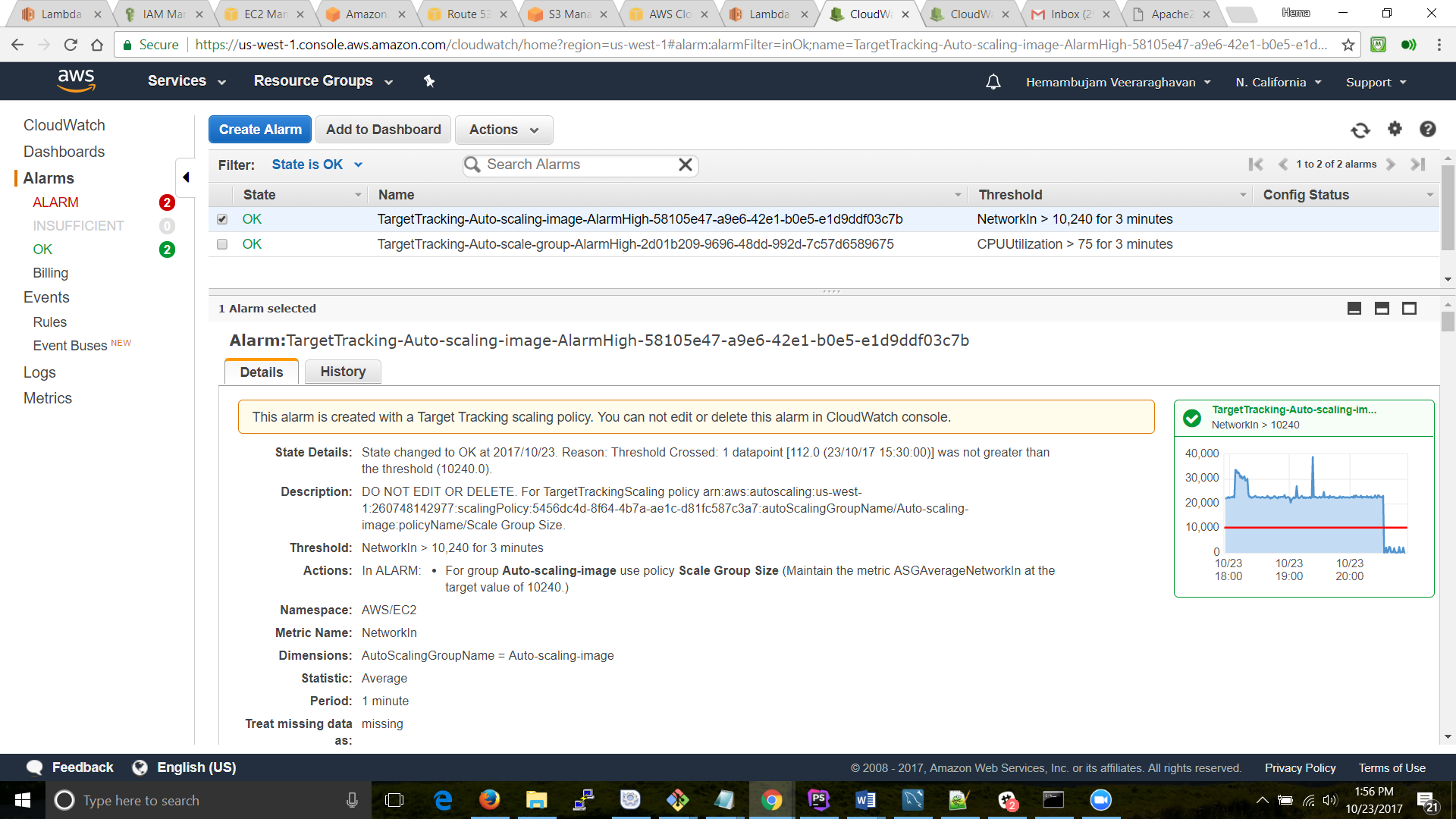
* + R53:

Amazon Route 53 is a highly available Domain Name System (DNS). Please find the configurations below:

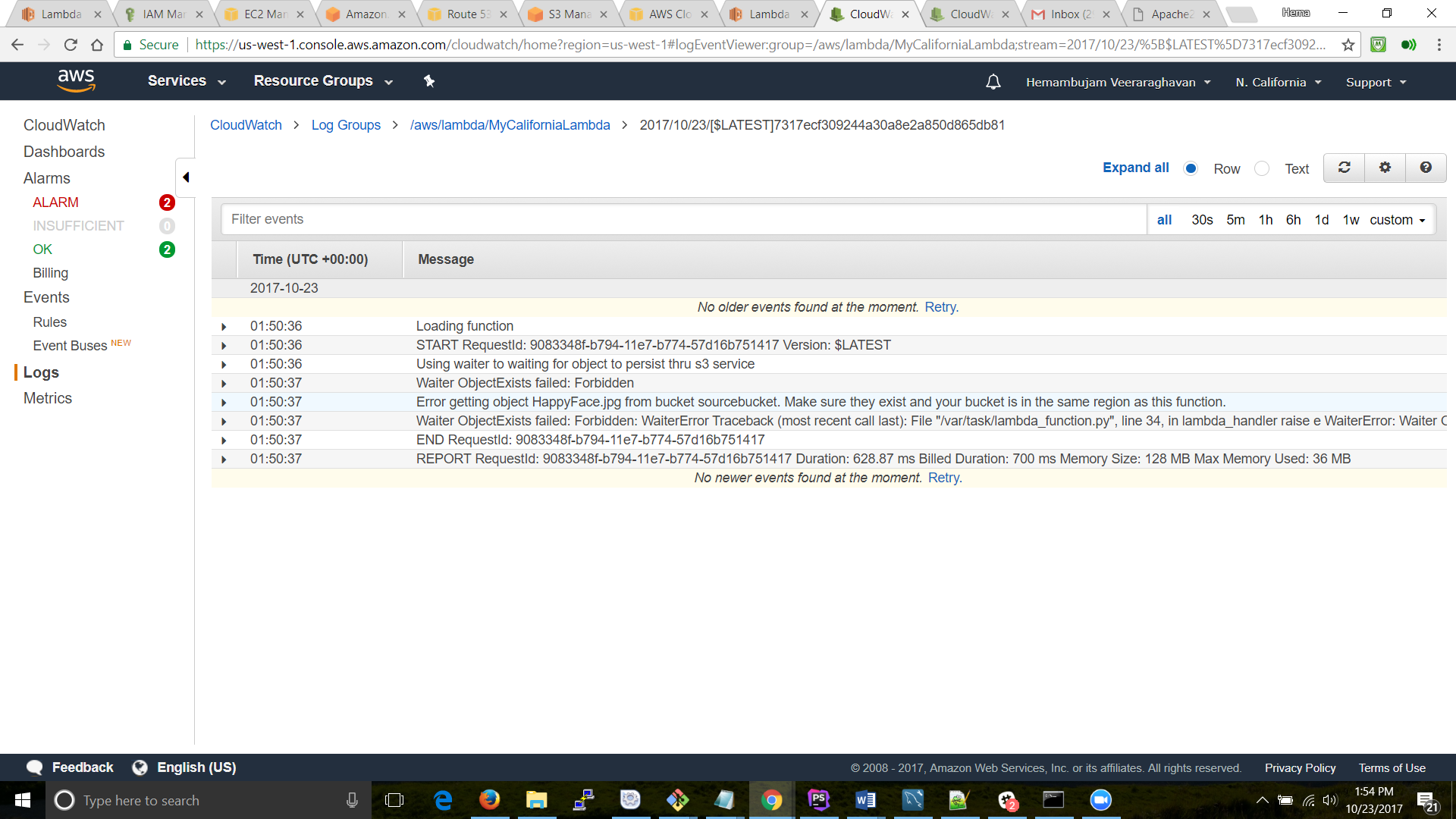


* + CloudWatch:

Amazon CloudWatch is a monitoring service for the cloud resources and the application we had spun up. I used Amazon CloudWatch to collect and track metrics, collect and monitor log files, and set alarms:

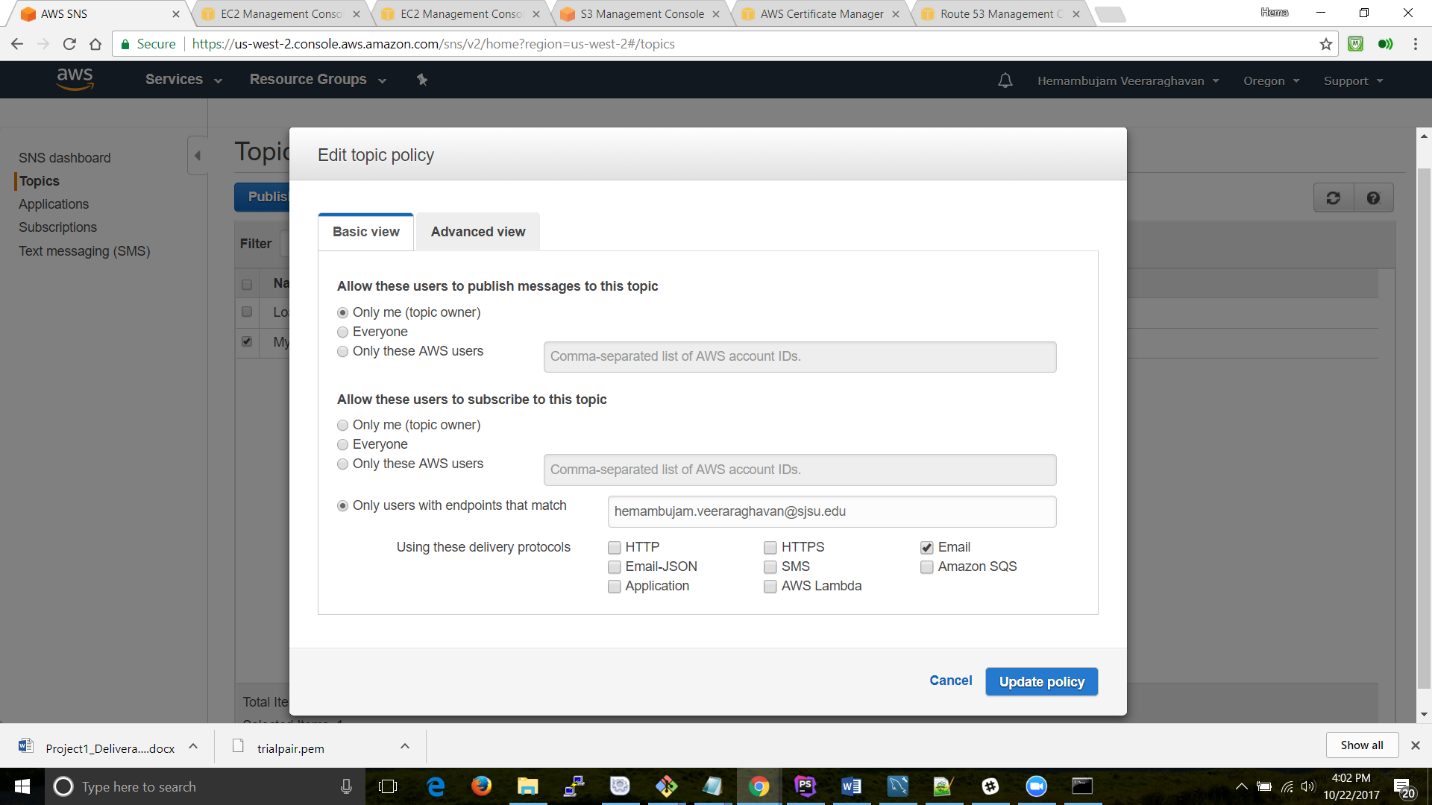


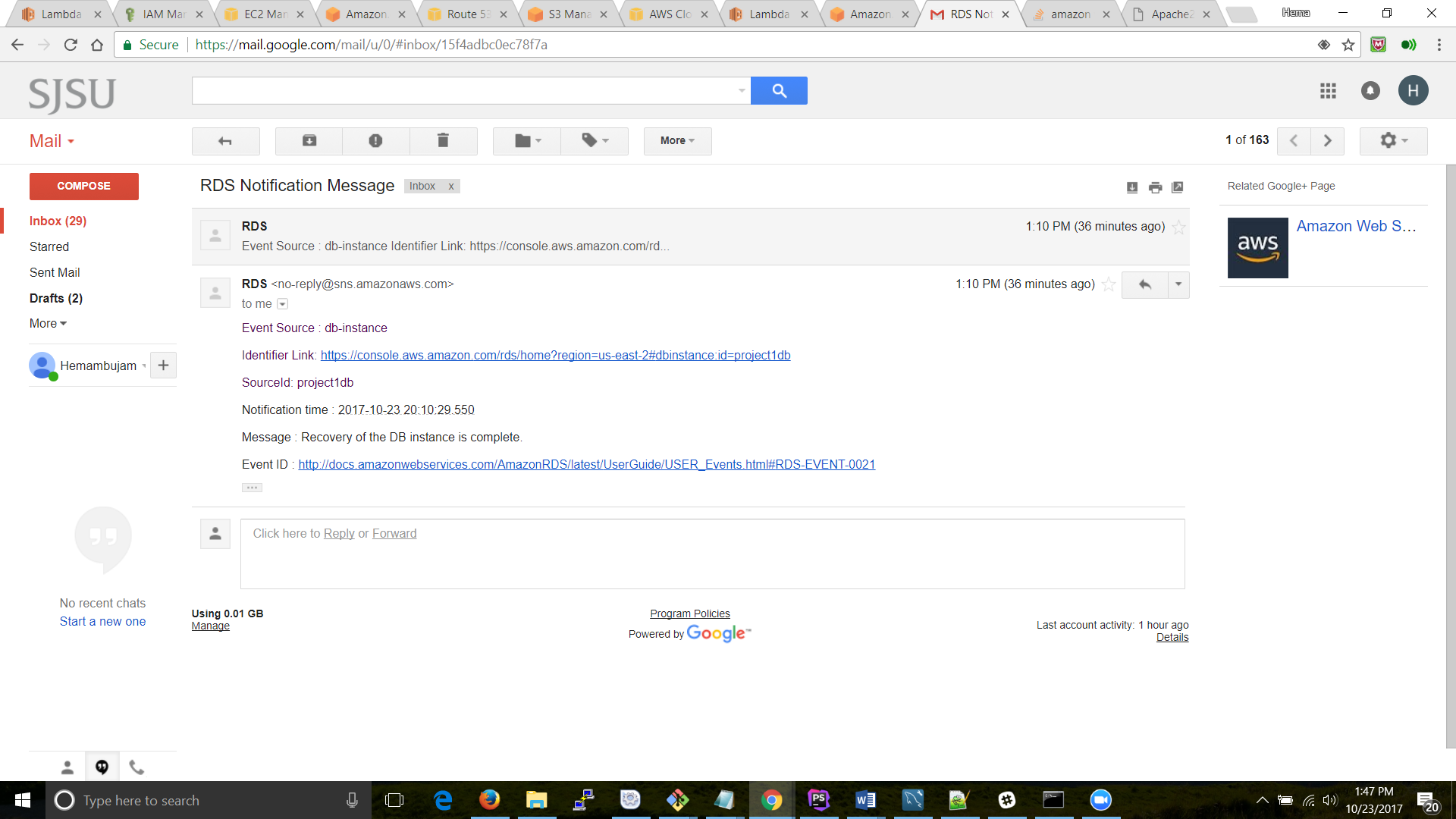
Cloudwatch log files:.



* + SNS:

 Simple Notification Service (SNS) is a notification service which provides a low-cost infrastructure for the mass delivery of messages, predominantly to mobile users:



**Email notification:** 

* Cost effective:

The solution is found to be cost effective by leveraging the cloud such as:

1. Moving data from S3 to IA to Glacier
2. Efficient management of EC2 instances using Auto-scaling.
3. Highly redundant application with a few resources.
4. Scalable and very minimal latency using Transfer Acceleration
5. Highly Available application using ELB

**References:**

* [www.aws.com](http://www.aws.com)
* [www.stacksoverflow.com](http://www.stacksoverflow.com)

Public url: [www.awscloudproject.com](http://www.awscloudproject.com)

Github: <https://github.com/hemambujamV/Aws_cloud_project>

Userid: testuser

Password: testuser