Project Report

"AWS Media Streamer"

CS 6386.501 Telecommunication Software Design

Department of Computer Science

Erik Jonsson School of Engineering and Computer Science

By,

Shashank Kumar Shankar (sxs141731)

INDEX

1. Introduction
2. Team Information
3. Description
4. Entities and Network
5. Design of Protocol for Communication
6. How do entities work together?!
7. Functionalities of Entities
8. Challenged Faced
9. Contributions from Team Members
10. Learning outcomes from the Project
11. Appendix
12. References

1. Introduction

The “Media Streamer” project has three communication entities to deploy remote access for media files. To facilitate this feature, we made use of cloud infrastructure provided by Amazon Web Services such as EC2, S3 and utilized a few external libraries to support communication over HTTP/2 in an Android application.

The AWS EC2 Server is an Ubuntu Linux machine which acts like a mediator between media database present in AWS S3 and a controller which uses the service. A REST API running on a Jetty HTTP/2 server is communicating with S3 bucket to return list of media content.

Android application runs like a controller, which is used to control the rendered media file. A Virtual Machine is used to implement the renderer functionality. This functionality includes running a Java implementation of VLC media player called “VLCj” in a Swing Utility frame to display the contents.

The communication between all the entities is based on HTTP/2. The reason behind using HTTP/2 is for its low latency achieved by its highly multiplexed nature which prevents head of line blocking.

To support HTTP/2 version of API communication, we have used a Java platform based Jetty Server. The Jetty Server is a light Server machine which runs on Linux machine.

Jersey framework REST API is used for communication between the android application and renderer. The API interaction involves passing HTTP/2 URL for a specific media file from Android application to Renderer.

2. Team Information

Name	**Net ID**
Ajay Chintalapalli Jayakumar	axc142430
Binal Kamani	bxk131030
Lakshmi Deepak Chadalawada	lxc140730
Shashank Kumar Shankar	sxs141731

3. Description

The Media Streamer project is divided into 3 modules:

Entity 1: Server

Server is a Java application running in Jetty HTTP/2 server present in an AWS EC2 instance. The application is a REST API which returns a JSON Object containing the list of Media available on a specified AWS S3 bucket. The communication occurs on top of HTTP/2.

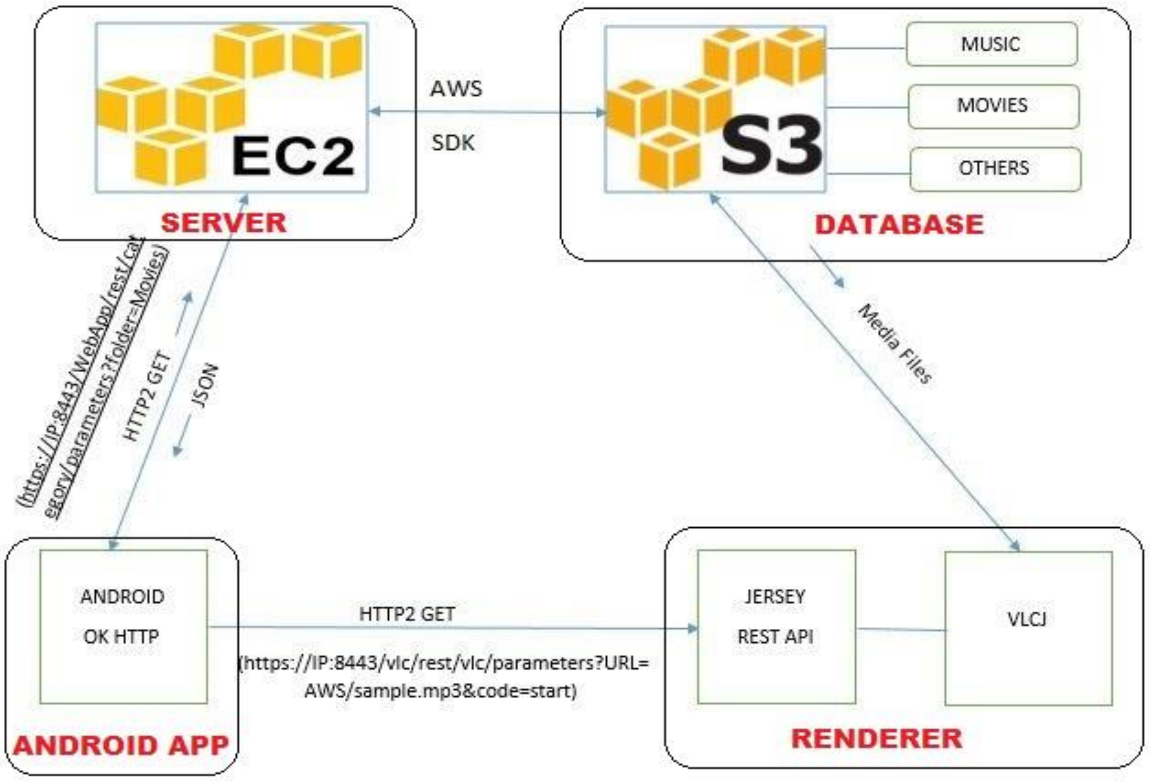
Entity 2: Controller

Controller is an Android application which requests for the list of media files by making HTTP/2 GET requests to the Server. Server sends back a JSON containing the list to the Android app and on receiving the JSON Object, the app parses the JSON and populates a List View for the user to select which media file to be played. The user will also be able to play, pause, resume, forward, rewind and stop the media and watch the results on the renderer.

Entity 3: Renderer

Renderer is chosen to be a Java implementation of VLC (VideoLAN) media player called VLCj. A REST API is developed to provide a control interface to the VLC player. The REST API present in the Renderer receives HTTP/2 requests from Android app, processes them and perform requested tasks e.g. play, pause, forward, backward, etc.

4. Entities and Network


EC2
SERVER
AWS
SDK
S3
MUSIC
MOVIES
OTHERS
DATABASE
Media Files
(https://IP:8443/WebApp/rest/category/parameters?folder=Movies)
HTTP2 GET
JSON
ANDROID
OK HTTP
ANDROID APP
HTTP2 GET
(https://IP:8443/vlc/rest/vlc/parameters?URL=AWS/sample.mp3&code=start)
JERSEY
REST API
VLCJ
RENDERER

[Figure 2.1: Diagram of Entities and Network]

Overview:

Diagram above shows the communication between the different entities. User uses an Android app to request list of media from server. From this list, the android application passes the media to be played, with its URL to the renderer. The renderer accepts the URL of the media and plays it in VLCj by requesting it from Amazon S3 media database.

5. Design of Protocol for communication

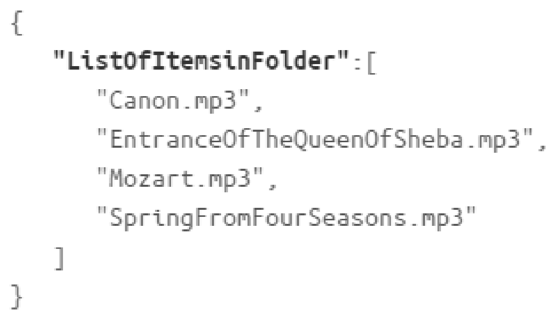
Each entity communicates with each other using customized URLs accessing REST APIs. The communication between the entities occur as follows:

Android to Server communication:

The Android App makes HTTP/2 GET requests to the following URL which is parsed by the REST API present in AWS EC2:

1. URL Call format: https://IPOfServer:8443/WebApp/rest/category/parameters?folder=Music
2. /WebApp/: Name of '.war' file deployed in Jetty Server
3. /rest/category/: URL Path
4. "/parameter?folder=": Selects a category. Values include - Movies, Music and Others. The Server parses this parameter and returns a JSON containing the contents of a particular folder category.

Once the android app does the HTTP/2 request, it receives a JSON. A sample JSON file when requested for "Music" category is shown below,



```
{
   "ListOfItemsinFolder":[
      "Canon.mp3",
      "EntranceOfTheQueenOfSheba.mp3",
      "Mozart.mp3",
      "SpringFromFourSeasons.mp3"
   ]
}
```

Android to Renderer communication:

The Android App makes HTTP/2 GET requests to the following URL which is parsed by the REST API present in the Renderer:

1. URL call format: https://IPOfRenderer:8443/vlc/rest/vlc/parameters?URL=http://s3-us-west-1.amazonaws.com/shank7485/Music/Mozart.mp3&code=start
2. /vlc/: Name of '.war' file deployed in Jetty Server
3. /rest/vlc/: URL Path
4. /parameters?URL=": Passing the URL of Media to renderer

5. "&code=start": Tells VLCj to start playing the passed URL. Values include - stop, resume, forward, rewind, pause.

6. How do the entities work together?!

Android:

An Android application initiates a connection to server by prompting user to enter Server IP address and renderer IP address. Once the fields for IP address have been filled, application then fetches the contents of the files stored in AWS S3 by making HTTP/2 GET requests to server. The files are listed on the application under the tags "Movies", "Music", "Others"

Server:

Server module utilizes storage and computational entities from Amazon Web Services for its functionality. The Elastic Compute Cloud (EC2) instance has a Jetty Server installed on Ubuntu that accepts connections from the Android app and the S3 storage entity has all the media files stored.

The Jetty Server is running in an EC2 which has a unique IP and is running a REST API which accepts HTTP/2 requests from the android application. It returns a JSON containing the media contents present in S3 when the HTTP/2 requests are made.

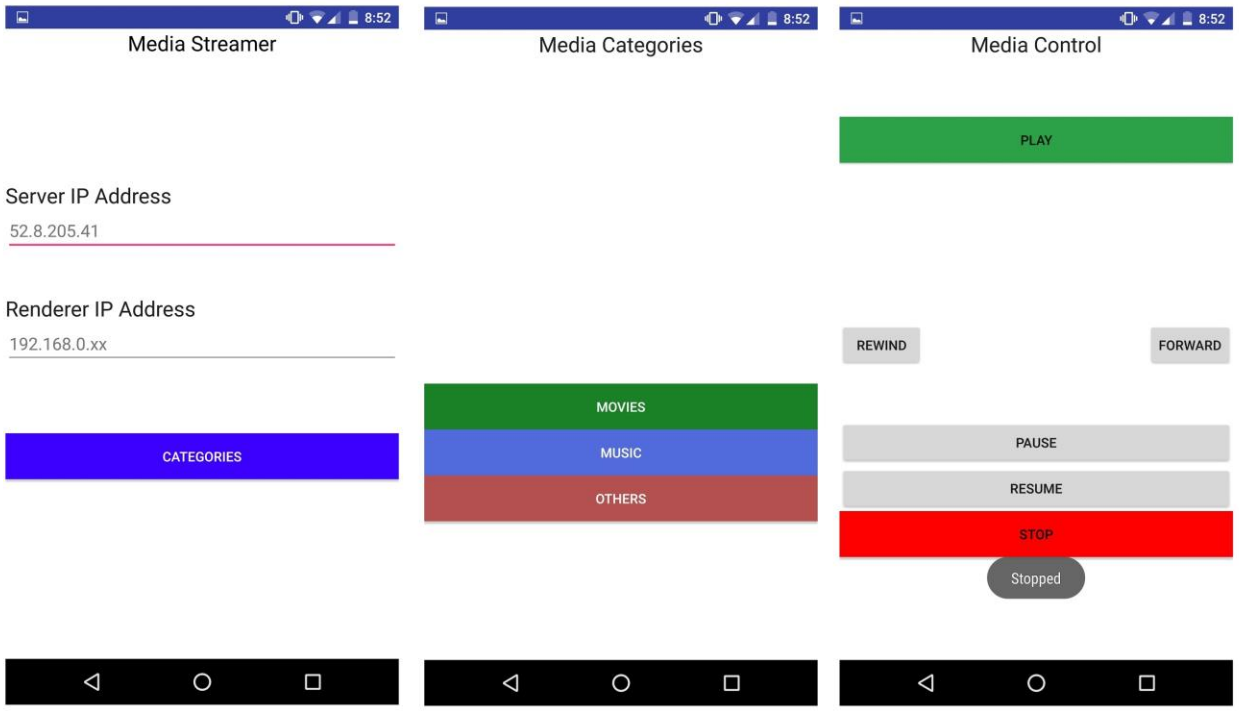
S3 has the media files stored ordered in a group manner. The folders has 3 categories a) Movies b) Music c) Others.

Renderer:

The renderer is a media player, a VLC player. The renderer is backed by a Jetty server running a REST API to control the media player functionalities. The android application sends HTTP/2 GET requests to the Jetty server on renderer side. The server parses the parameter passes in the URL and performs appropriate functions based on the command sent. The server sends out the play, pause, forward, backward and stops messages to the VLC player depending on the command. The player plays the media files selected by the android app by loading the URL and streaming it in a new thread.

7. Functionalities of Entities

Android:



Server IP Address
52.8.205.41

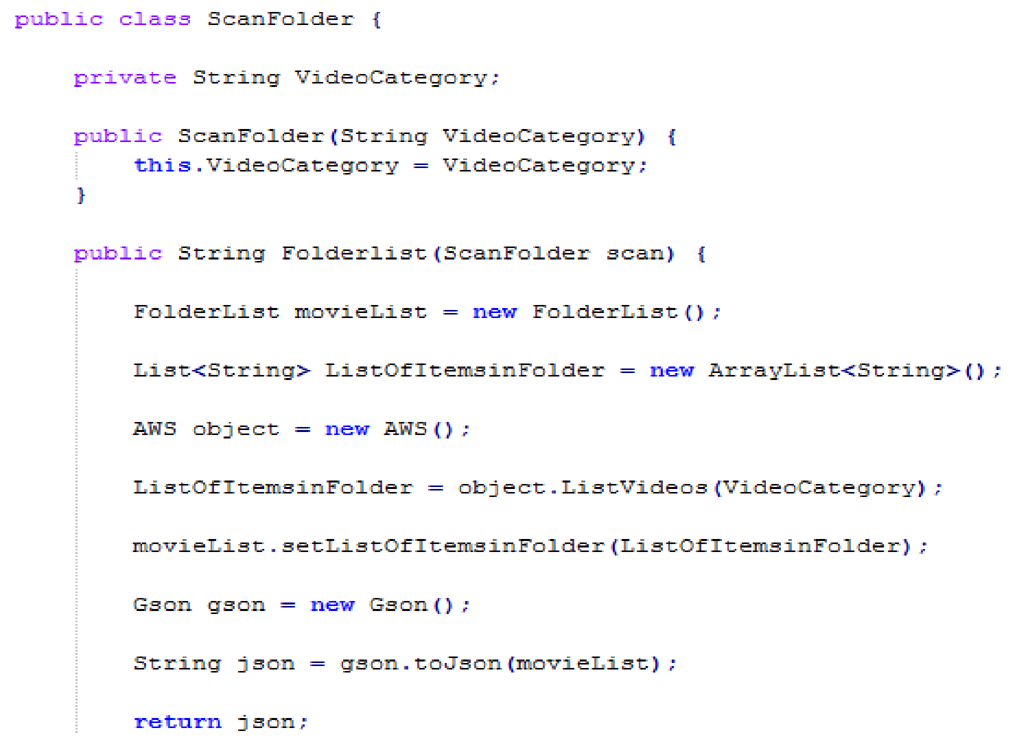
Renderer IP Address
192.168.0.xx

The screenshot shown above contains all the screens of the android application. The application (first image) prompts the user to enter the Server IP Address and Renderer IP Address. Then it displays the categories under which user can request for media files.

The server has to be kept running before we launch the application. Android application talks to AWS EC2 server via GET requests upon HTTP/2 protocol. The EC2 instance has Jetty server running on port 8443. Upon getting a request, the server communicates with AWS S3 using AWS SDK and gets a list of media inside a particular folder. The EC2 server then sends back the list as JSON to the android application and the list is being displayed under specific name tags on the Android app. The user can now select whichever media he wants to play and this will call another HTTP/2 GET on the renderer server and the media starts playing.

Server:

- **AWS EC2/Jetty -** The server is basically an AWS EC2 instance with Jetty server installed on it. Jetty is a Java Servlet engine and Web Server that runs on latest JDKs. This server running, runs the ScanFolder.java file to get the list from AWS S3 bucket.

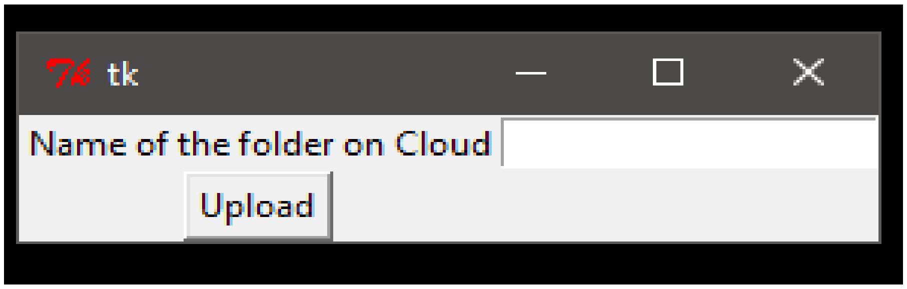


```
public class ScanFolder {

    private String VideoCategory;

    public ScanFolder(String VideoCategory) {
        this.VideoCategory = VideoCategory;
    }

    public String Folderlist(ScanFolder scan) {

        FolderList movieList = new FolderList();

        List<String> ListOfItemsinFolder = new ArrayList<String>();

        AWS object = new AWS();

        ListOfItemsinFolder = object.ListVideos(VideoCategory);

        movieList.setListOfItemsinFolder(ListOfItemsinFolder);

        Gson gson = new Gson();

        String json = gson.toJson(movieList);

        return json;
```

The above screenshot shows how the code gets list of video categories and returns it as JSON.

- **S3 instance:** It is a Simple Storage Service provided by Amazon Web services. This service stores the data/files in groups called Buckets. This feature has been exploited to create categories of media files.

We have implemented an add-on feature to the project with a simple GUI based upload feature. The GUI runs on Tkinter, a python GUI library. The screenshot shown below is the GUI for uploading files to the S3 bucket.



Renderer: It is configured to be on an Ubuntu Linux with the REST API deployed in a Jetty HTTP/2 Server.

Renderer has 2 functionalities;

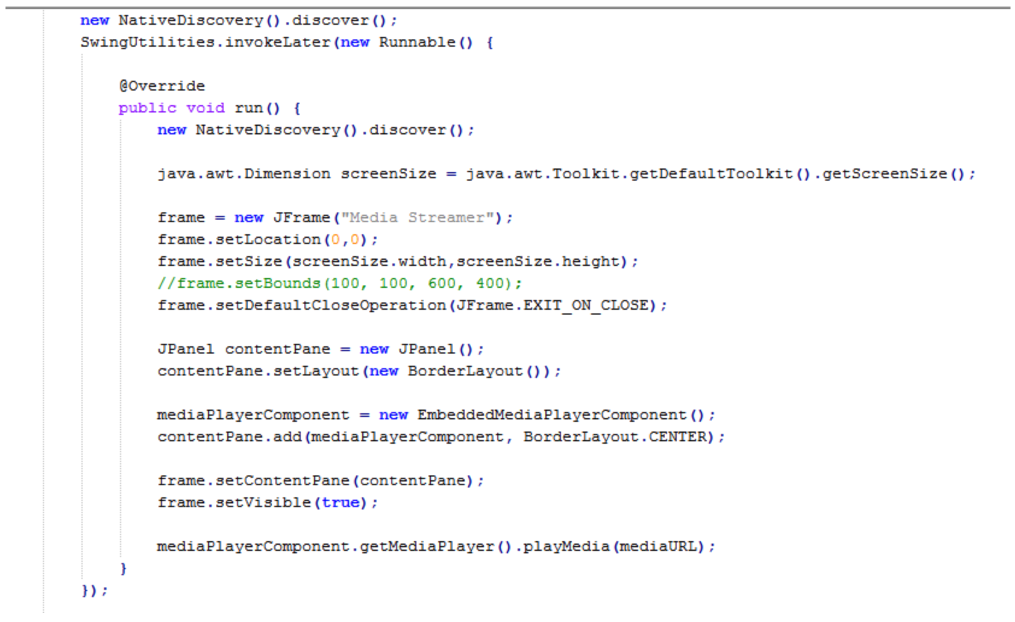
1. Server Functionality
2. Fetch and Play Functionality

Server Functionality:

A Jetty server is installed on the VM. This server plays the mediator functionality which gets the HTTP/2 based URL parameters from Android app and provides it to the VLCj (Java framework for VLC Media Player).

Fetch and Play Functionality:

The parameters received from the Android is the URL (Uniform Resource Locater) and command for the VLCj media player. The URL is used by VLCj to fetch media contents from S3 and command to play, pause, forward etc. in the media player. The renderer plays the media contents in the Swing utility (a GUI Widget toolkit for Java). Swing creates a base frame with full screen dimensions and starts playing the passed URL.



```
new NativeDiscovery().discover();
SwingUtilities.invokeLater(new Runnable() {

    @Override
    public void run() {
        new NativeDiscovery().discover();

        java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

        frame = new JFrame("Media Streamer");
        frame.setLocation(0,0);
        frame.setSize(screenSize.width,screenSize.height);
        //frame.setBounds(100, 100, 600, 400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel contentPane = new JPanel();
        contentPane.setLayout(new BorderLayout());

        mediaPlayerComponent = new EmbeddedMediaPlayerComponent();
        contentPane.add(mediaPlayerComponent, BorderLayout.CENTER);

        frame.setContentPane(contentPane);
        frame.setVisible(true);

        mediaPlayerComponent.getMediaPlayer().playMedia(mediaURL);
    }
});
```

The code shows how the Swing Thread is fetching the content from the URL. The URL is passed as string to the object of class.

8. Challenges Faced

- Researching for a Server which supports HTTP/2
- Setting up Server and understanding Java based Servers/Servlets
- Connection setup from Android to EC2 and Jetty server (on the renderer side).
- Understanding Jersey REST Framework
- Implementing REST APIs using Jersey
- Understanding VLCj java implementation of VLC Media Player
- Developing Android Application and using external libraries to enable HTTP/2 in Android
- Understanding working of cloud services such as AWS EC2 and AWS S3
- Implementing APIs provided by AWS SDK for programmatic access of AWS.

9. Contributions from Team Members

Deepak: He worked on the Android application. He took help from me for setting up the external OkHTTP libraries and also helped him in getting the list of media from server and displaying it as list view in Android Intent.

Ajay: He worked on the Renderer part of the project. He worked on collecting and parsing the URL parameters sent from the android application. I helped him in providing the REST API backend for the VLCj application.

Binal: She helped me in setting up the Jetty HTTP/2 servers on the EC2 instance and on the Renderer. She also helped in explaining how WAR files are deployed in HTTP/2 servers.

Shashank: I see myself as the Team Leader wherein I architected the whole idea of using AWS services in this project and designed the end-to-end software modules. Once the design and protocol was decided, I divided the work and assigned each module work to the team members.

I worked on all of the entities. My major work was in implementing the Java Jersey based REST APIs in the Server and the Renderer and using AWS services for compute and storage. I worked on the Java based AWS SDK to communicate between the EC2 instance and AWS S3 bucket. I also designed the protocol to pass parameters on top of HTTP/2 based URLs to the Server and Render. I also made sure the correct JSON was passed to the Android app whenever a HTTP/2 GET request was made.

I also helped the other team members. I assisted Deepak in setting up OkHTTP library in android gradle to build the .jar files. Also, I helped him in displaying the list in List View in android intent. I helped Ajay in running the VLCj with REST backend and helped him to run the Java Swings in threads. He faced issues in tearing down a thread, I helped him resolve the issue. Since Binal took care of setting up the Jetty HTTP/2 server, I helped her in installing this server on the AWS EC2 instance.

10. Learning outcomes from the Project

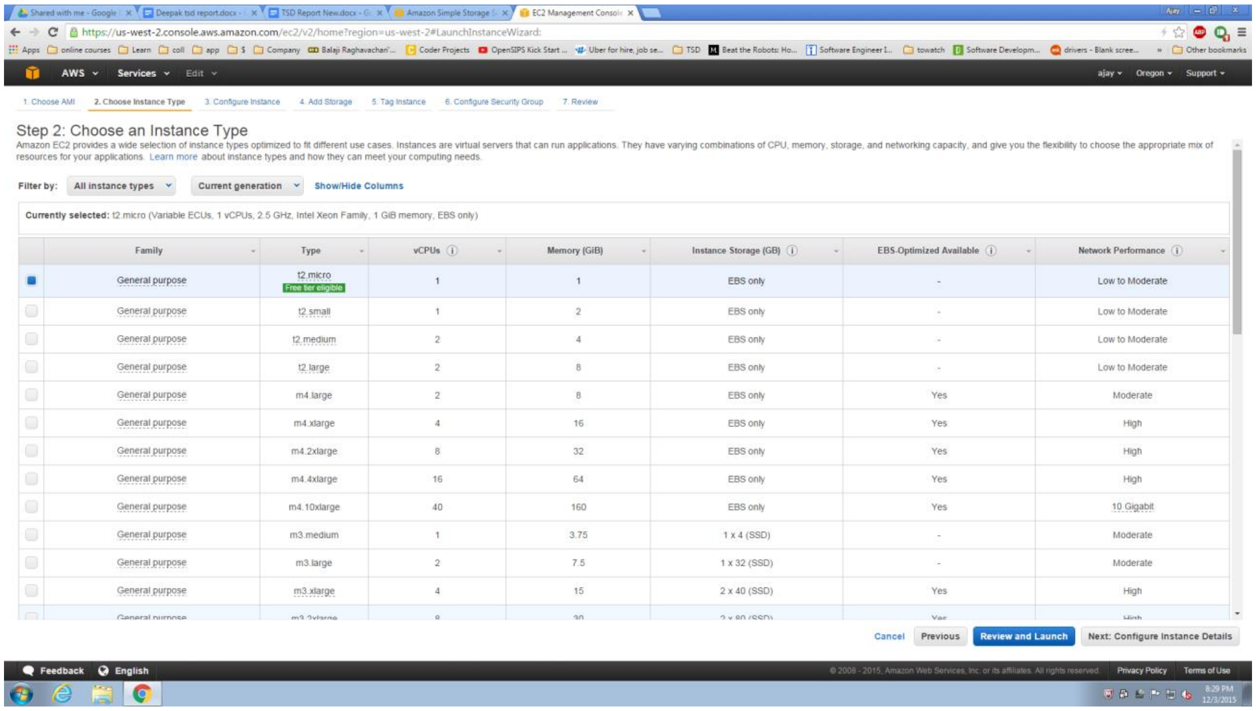
Some of the important aspects learnt from this project are:

- Designing and Implementing HTTP/2 based protocols.
- Programming an Android application which makes uses of external libraries to make HTTP/2 requests.
- Designing and Implementing RESTful API services.
- Returning a JSON and parsing a JSON in Android.
- Creating and stopping worker Threads in Android and in Java.
- Setting up Java based Servers.
- Using cloud services provided by Amazon.
- Implementing the AWS SDK to enable communication between an AWS EC2 instance and an S3 bucket.
- Leadership.
- Teamwork.

11. Appendix

Setting up an EC2 Instance:

- Figure 2: Initially, it is required to have an AWS account by signing up at https://aws.amazon.com/. Once that is done, go into the AWS console and select EC2.Choose the type of Image required for the instance. Image means the type of operating system required to boot up.



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Filter by: All instance types | Current generation | Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

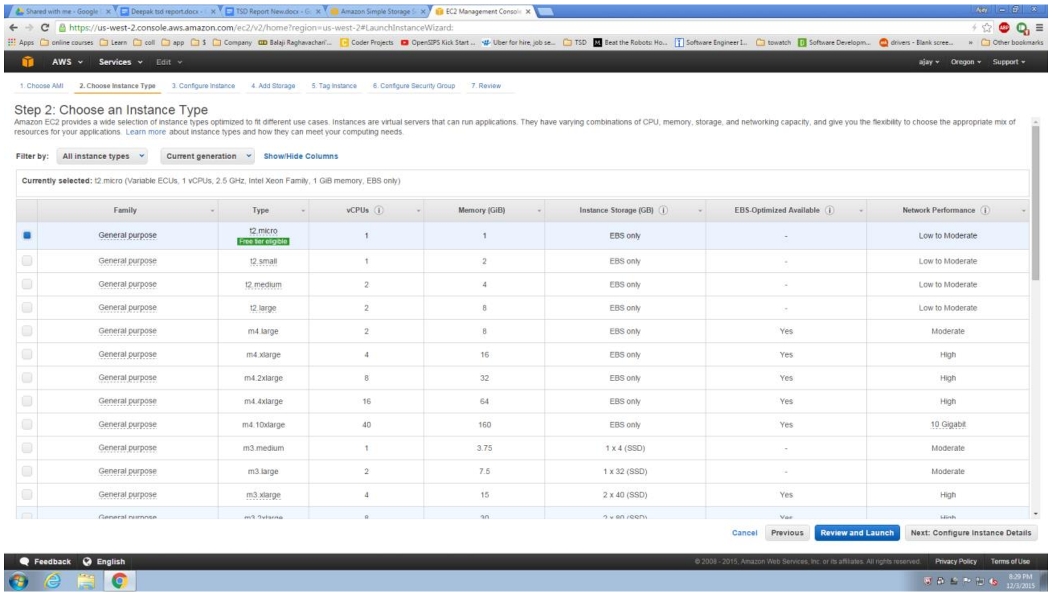
	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
■	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate
☐	General purpose	t2.small	1	2	EBS only	-	Low to Moderate
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
☐	General purpose	t2.large	2	8	EBS only	-	Low to Moderate
☐	General purpose	m4.large	2	8	EBS only	Yes	Moderate
☐	General purpose	m4.xlarge	4	16	EBS only	Yes	High
☐	General purpose	m4.2xlarge	8	32	EBS only	Yes	High
☐	General purpose	m4.4xlarge	16	64	EBS only	Yes	High
☐	General purpose	m4.10xlarge	40	160	EBS only	Yes	10 Gigabit
☐	General purpose	m3.medium	1	3.75	1 x 4 (SSD)	-	Moderate
☐	General purpose	m3.large	2	7.5	1 x 32 (SSD)	-	Moderate
☐	General purpose	m3.xlarge	4	15	2 x 40 (SSD)	Yes	High
☐	General purpose	m3.2xlarge	8	30	2 x 80 (SSD)	Yes	High

© 2008 - 2015, Amazon Web Services, Inc. or its affiliates. All rights reserved.

8:29 PM 12/3/2015

[Figure 2]

- <u>Figure 3:</u> Select the type of Instance required.



https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#LaunchInstanceWizard:

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Filter by: All instance types | Current generation | Show/Hide Columns

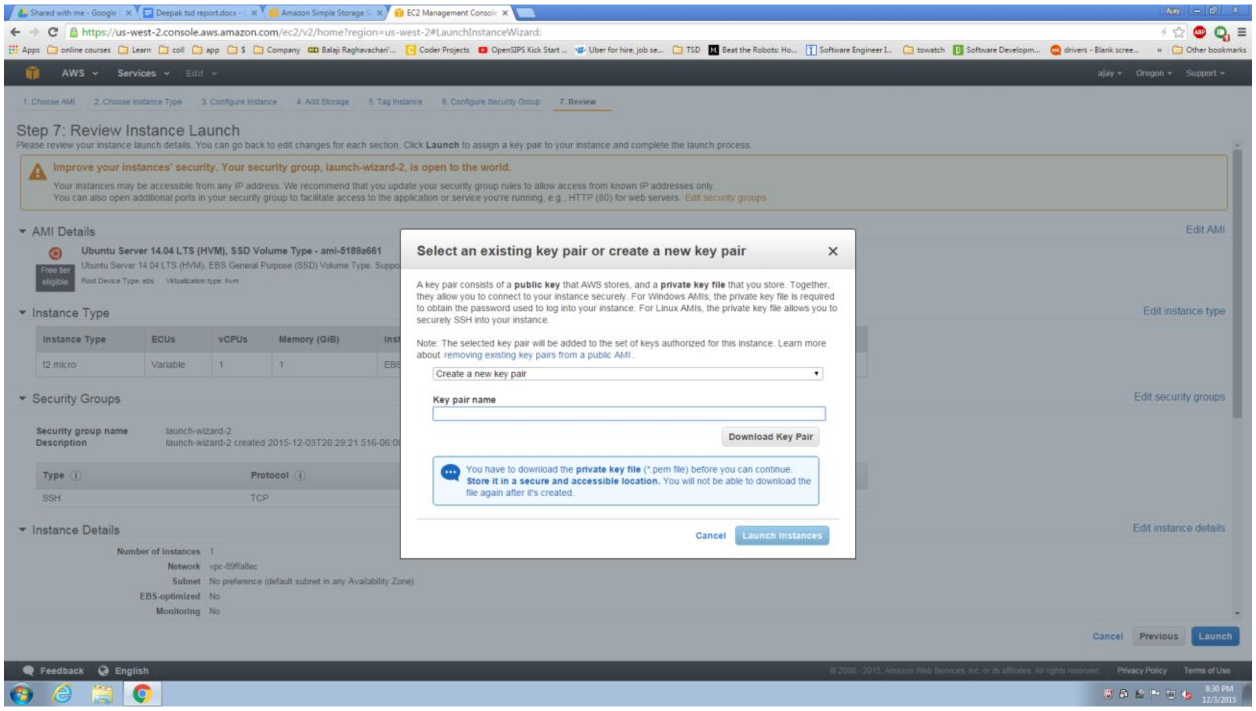
Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
■	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate
☐	General purpose	t2.small	1	2	EBS only	-	Low to Moderate
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
☐	General purpose	t2.large	2	8	EBS only	-	Low to Moderate
☐	General purpose	m4.large	2	8	EBS only	Yes	Moderate
☐	General purpose	m4.xlarge	4	16	EBS only	Yes	High
☐	General purpose	m4.2xlarge	8	32	EBS only	Yes	High
☐	General purpose	m4.4xlarge	16	64	EBS only	Yes	High
☐	General purpose	m4.10xlarge	40	160	EBS only	Yes	10 Gigabit
☐	General purpose	m3.medium	1	3.75	1 x 4 (SSD)	-	Moderate
☐	General purpose	m3.large	2	7.5	1 x 32 (SSD)	-	Moderate
☐	General purpose	m3.xlarge	4	15	2 x 40 (SSD)	Yes	High
☐	General purpose	m3.2xlarge	8	30	2 x 80 (SSD)	Yes	High

Cancel
Previous
Review and Launch
Next: Configure Instance Details
Feedback
English
© 2008 - 2015, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

[Figure 3]

- <u>Figure 4:</u> Generate a Key pair. This is required for accessing the AWS Instance using SSH. It is required to keep the download ".pem" file secure as this is the only way the AWS instance can be accessed.



https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#LaunchInstanceWizard:

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

Improve your instances' security. Your security group, launch-wizard-2, is open to the world.

Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.
You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. Edit security groups

AMI Details — Edit AMI

Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-5189a661
Ubuntu Server 14.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Suppo
Free tier eligible — Root Device Type: ebs — Virtualization type: hvm

Instance Type — Edit instance type

Instance Type	ECUs	vCPUs	Memory (GiB)	Inst
t2.micro	Variable	1	1	EBS

Security Groups — Edit security groups

Security group name launch-wizard-2
Description launch-wizard-2 created 2015-12-03T20:29:21.516-06:0

Type	Protocol
SSH	TCP

Instance Details — Edit instance details

Number of instances 1
Network vpc-89ffa8ec
Subnet No preference (default subnet in any Availability Zone)
EBS-optimized No
Monitoring No

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Create a new key pair

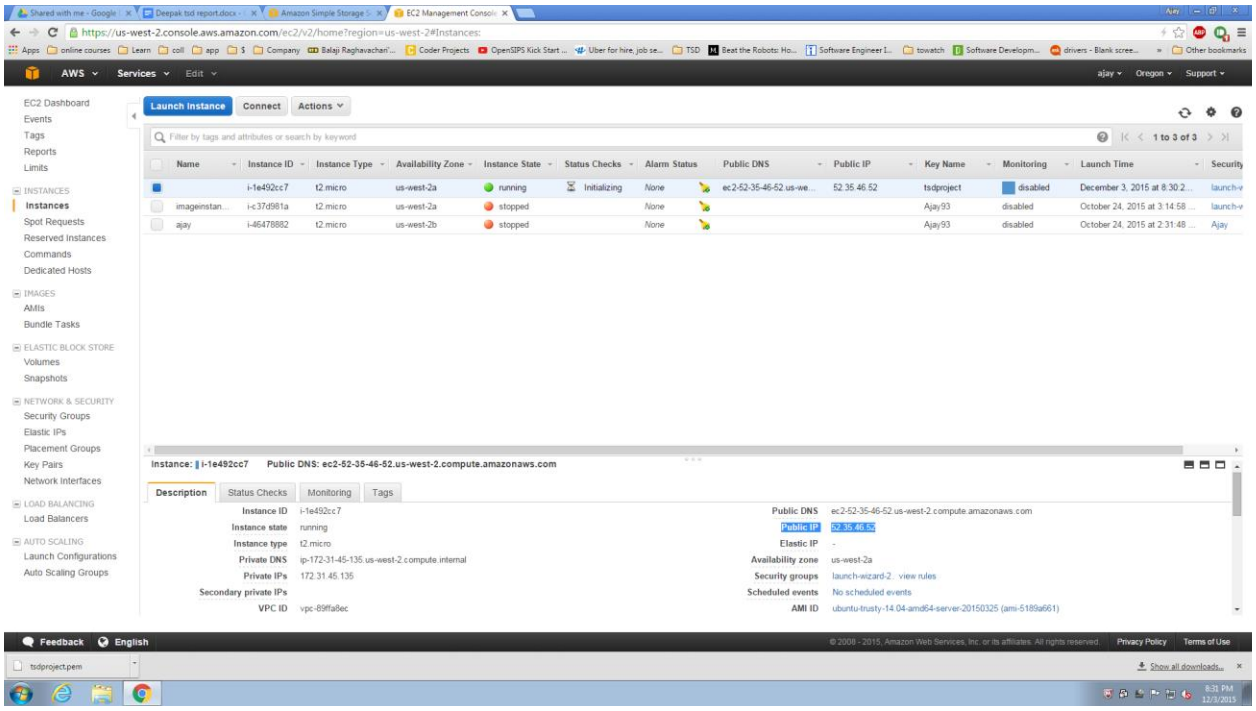
Key pair name

Download Key Pair

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

[Figure 4]

- Figure 5: Select the instance which was created and start it. Note down the IP address of the instance.



	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP	Key Name	Monitoring	Launch Time	Security
		i-1e492cc7	t2.micro	us-west-2a	running	Initializing	None	ec2-52-35-46-52.us-we...	52.35.46.52	tsdproject	disabled	December 3, 2015 at 8:30:2...	launch-v
	imageinstan...	i-c37d981a	t2.micro	us-west-2a	stopped		None			Ajay93	disabled	October 24, 2015 at 3:14:58 ...	launch-v
	ajay	i-46478882	t2.micro	us-west-2b	stopped		None			Ajay93	disabled	October 24, 2015 at 2:31:48 ...	Ajay

Instance: i-1e492cc7 Public DNS: ec2-52-35-46-52.us-west-2.compute.amazonaws.com

Instance ID	i-1e492cc7	Public DNS	ec2-52-35-46-52.us-west-2.compute.amazonaws.com
Instance state	running	Public IP	52.35.46.52
Instance type	t2.micro	Elastic IP	-
Private DNS	ip-172-31-45-135.us-west-2.compute.internal	Availability zone	us-west-2a
Private IPs	172.31.45.135	Security groups	launch-wizard-2. view rules
Secondary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-89ffa8ec	AMI ID	ubuntu-trusty-14.04-amd64-server-20150325 (ami-5189a661)

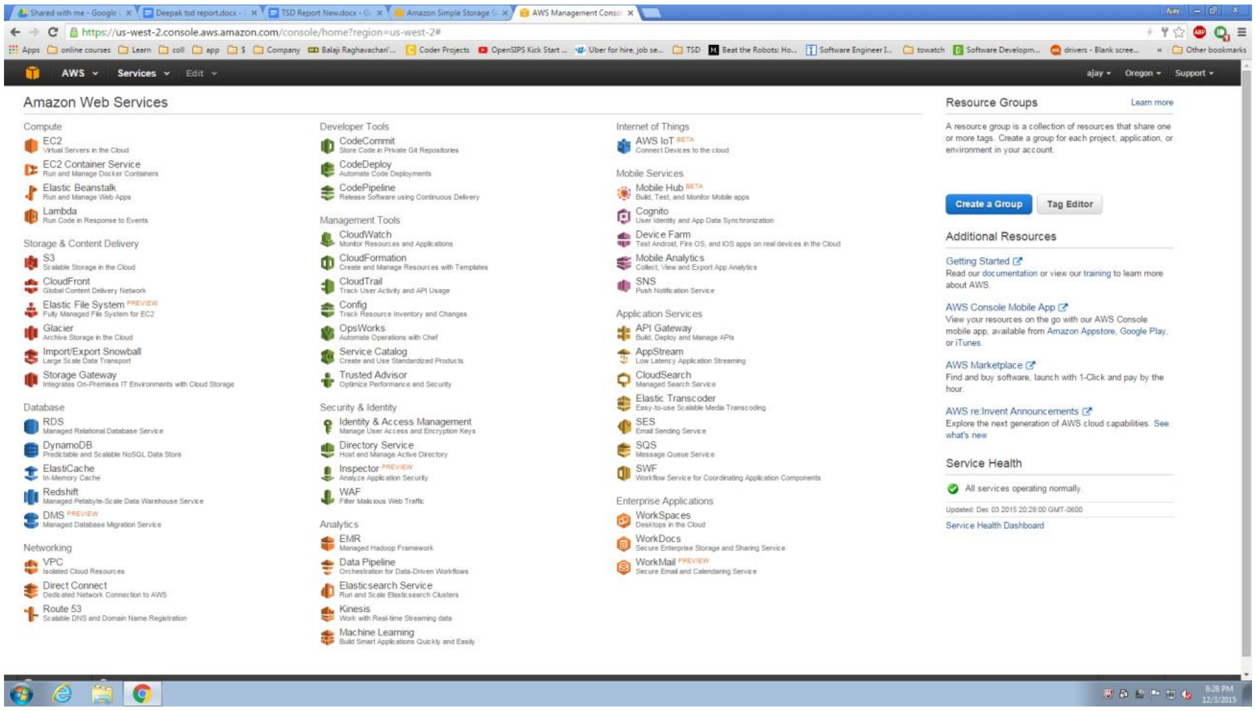
[Figure 5]

- To connect to the AWS Instance, use Putty to SSH. Also, it is required to convert the .pem file to .ppk using Putty Gen. This came be done by following the documentation provided in this link: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html
- The detailed documentation for launching an AWS Instance can be obtained in the following link: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

Setting up an S3 Storage Bucket:

S3 stands for Simple storage service, below shown images are the initial steps followed to create an S3 storage instance.

Figure 6: The above image shows the available services on AWS. Select S3 from the list,



Amazon Web Services

Compute
- EC2 — Virtual Servers in the Cloud
- EC2 Container Service — Run and Manage Docker Containers
- Elastic Beanstalk — Run and Manage Web Apps
- Lambda — Run Code in Response to Events

Storage & Content Delivery
- S3 — Scalable Storage in the Cloud
- CloudFront — Global Content Delivery Network
- Elastic File System PREVIEW — Fully Managed File System for EC2
- Glacier — Archive Storage in the Cloud
- Import/Export Snowball — Large Scale Data Transport
- Storage Gateway — Integrates On-Premises IT Environments with Cloud Storage

Database
- RDS — Managed Relational Database Service
- DynamoDB — Predictable and Scalable NoSQL Data Store
- ElastiCache — In-Memory Cache
- Redshift — Managed Petabyte-Scale Data Warehouse Service
- DMS PREVIEW — Managed Database Migration Service

Networking
- VPC — Isolated Cloud Resources
- Direct Connect — Dedicated Network Connection to AWS
- Route 53 — Scalable DNS and Domain Name Registration

Developer Tools
- CodeCommit — Store Code in Private Git Repositories
- CodeDeploy — Automate Code Deployments
- CodePipeline — Release Software using Continuous Delivery

Management Tools
- CloudWatch — Monitor Resources and Applications
- CloudFormation — Create and Manage Resources with Templates
- CloudTrail — Track User Activity and API Usage
- Config — Track Resource Inventory and Changes
- OpsWorks — Automate Operations with Chef
- Service Catalog — Create and Use Standardized Products
- Trusted Advisor — Optimize Performance and Security

Security & Identity
- Identity & Access Management — Manage User Access and Encryption Keys
- Directory Service — Host and Manage Active Directory
- Inspector PREVIEW — Analyze Application Security
- WAF — Filter Malicious Web Traffic

Analytics
- EMR — Managed Hadoop Framework
- Data Pipeline — Orchestration for Data-Driven Workflows
- Elasticsearch Service — Run and Scale Elasticsearch Clusters
- Kinesis — Work with Real-time Streaming data
- Machine Learning — Build Smart Applications Quickly and Easily

Internet of Things
- AWS IoT BETA — Connect Devices to the cloud

Mobile Services
- Mobile Hub BETA — Build, Test, and Monitor Mobile apps
- Cognito — User Identity and App Data Synchronization
- Device Farm — Test Android, Fire OS, and iOS apps on real devices in the Cloud
- Mobile Analytics — Collect, View and Export App Analytics
- SNS — Push Notification Service

Application Services
- API Gateway — Build, Deploy and Manage APIs
- AppStream — Low Latency Application Streaming
- CloudSearch — Managed Search Service
- Elastic Transcoder — Easy-to-use Scalable Media Transcoding
- SES — Email Sending Service
- SQS — Message Queue Service
- SWF — Workflow Service for Coordinating Application Components

Enterprise Applications
- WorkSpaces — Desktops in the Cloud
- WorkDocs — Secure Enterprise Storage and Sharing Service
- WorkMail PREVIEW — Secure Email and Calendaring Service

Resource Groups — Learn more

A resource group is a collection of resources that share one or more tags. Create a group for each project, application, or environment in your account.

Create a Group | Tag Editor

Additional Resources

Getting Started
Read our documentation or view our training to learn more about AWS.

AWS Console Mobile App
View your resources on the go with our AWS Console mobile app, available from Amazon Appstore, Google Play, or iTunes.

AWS Marketplace
Find and buy software, launch with 1-Click and pay by the hour.

AWS re:Invent Announcements
Explore the next generation of AWS cloud capabilities. See what's new

Service Health

✓ All services operating normally.

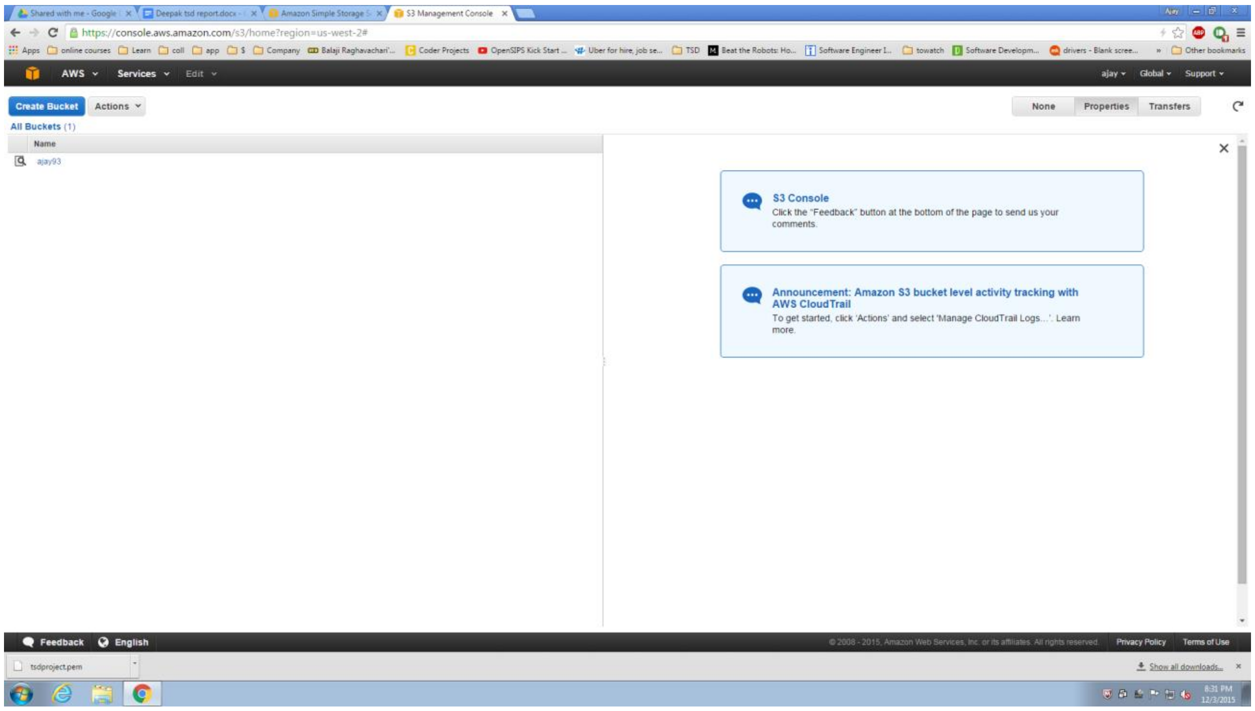
Updated: Dec 03 2015 20:28:00 GMT-0600

Service Health Dashboard

8:28 PM
12/3/2015

[Figure 6]

- Figure 7: Once user enters S3 instance, user is prompted to create a bucket. Bucket is a storage entity in S3. On clicking CREATE BUCKET. Every S3 instance gives a unique URL for the files stored in it.



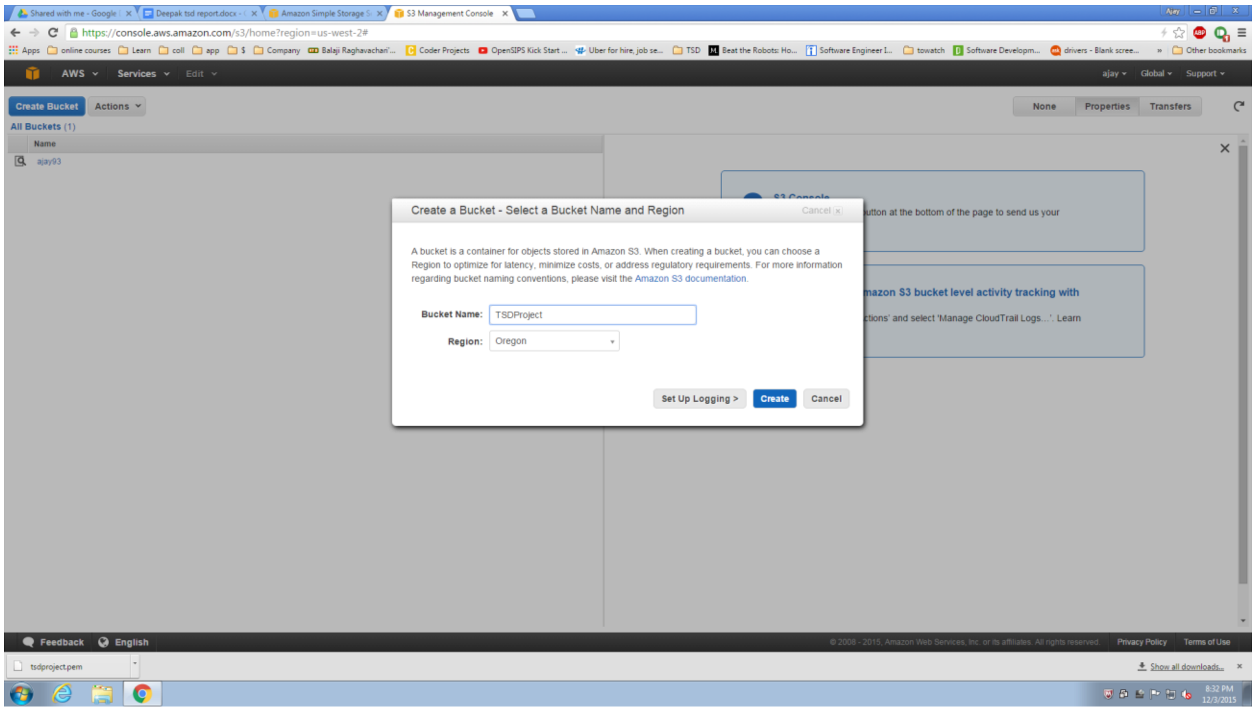
S3 Console
Click the "Feedback" button at the bottom of the page to send us your comments.

Announcement: Amazon S3 bucket level activity tracking with AWS CloudTrail
To get started, click 'Actions' and select 'Manage CloudTrail Logs...'. Learn more.

© 2008 - 2015, Amazon Web Services, Inc. or its affiliates. All rights reserved.

[Figure 7]

- <u>Figure 8/Figure 9:</u> A pop up box comes up asking for the bucket name. It also tells the place where my storage in located on AWS, Oregon in our case. The prompt does not accept uppercase letters for the bucket name.



Create a Bucket - Select a Bucket Name and Region

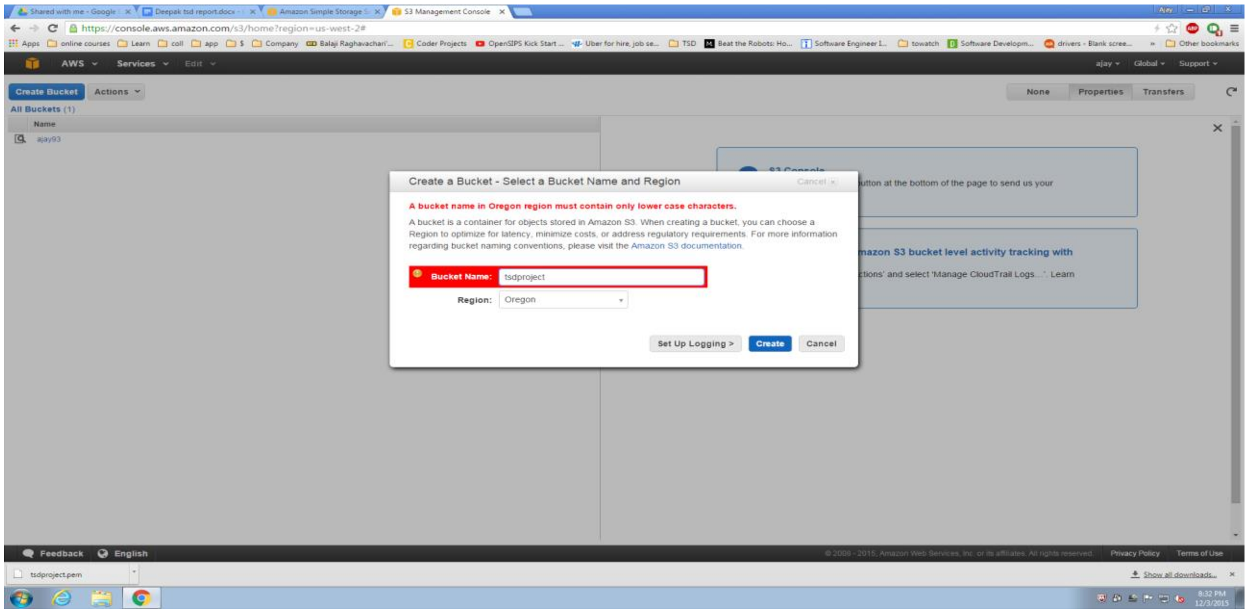
A bucket is a container for objects stored in Amazon S3. When creating a bucket, you can choose a Region to optimize for latency, minimize costs, or address regulatory requirements. For more information regarding bucket naming conventions, please visit the Amazon S3 documentation.

Bucket Name: TSDProject

Region: Oregon

Set Up Logging > | Create | Cancel

[Figure 8]



Create a Bucket - Select a Bucket Name and Region

A bucket name in Oregon region must contain only lower case characters.

A bucket is a container for objects stored in Amazon S3. When creating a bucket, you can choose a Region to optimize for latency, minimize costs, or address regulatory requirements. For more information regarding bucket naming conventions, please visit the Amazon S3 documentation.

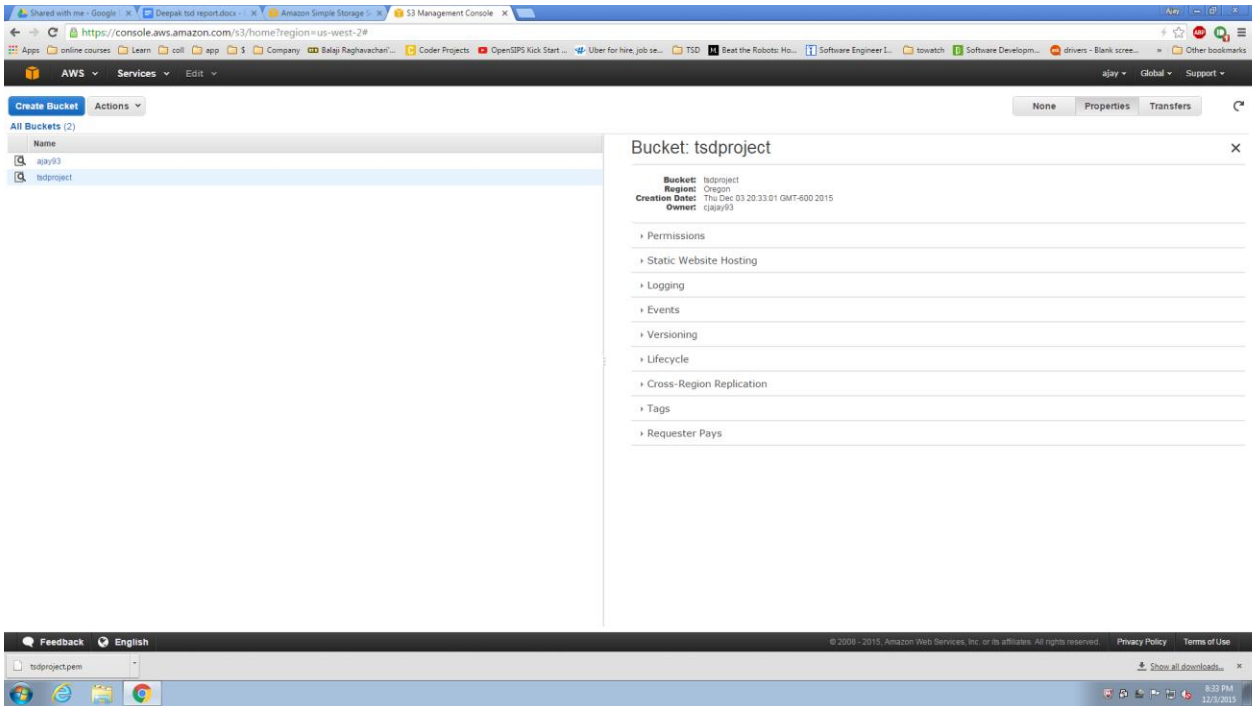
Bucket Name: tsdproject

Region: Oregon

Set Up Logging > | Create | Cancel

[Figure 9]

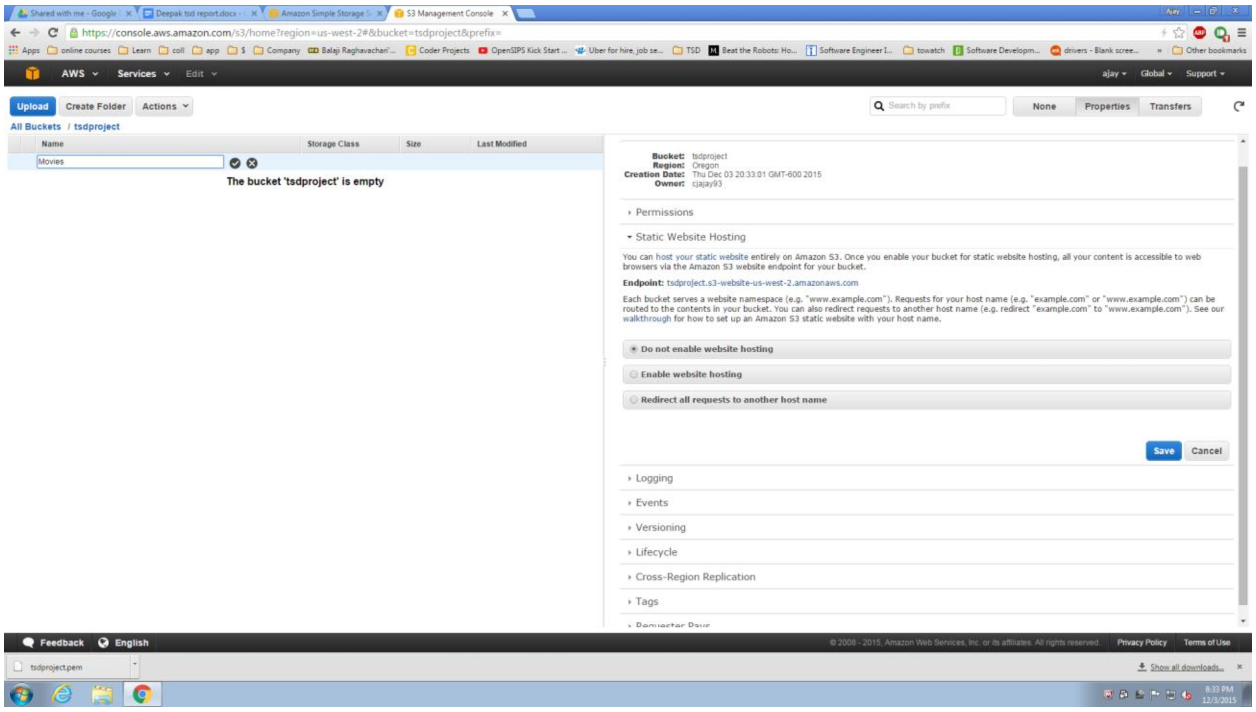
- <u>Figure 10:</u> Once the bucket is created, it is easily accessible. The images shown below are for the creation of multiple folders. The folders created are specific to our project.



Bucket: tsdproject

Bucket: tsdproject
Region: Oregon
Creation Date: Thu Dec 03 20:33:01 GMT-600 2015
Owner: cjajay93

[Figure 10]



The bucket 'tsdproject' is empty

Bucket: tsdproject
Region: Oregon
Creation Date: Thu Dec 03 20:33:01 GMT-600 2015
Owner: cjajay93

Static Website Hosting

You can host your static website entirely on Amazon S3. Once you enable your bucket for static website hosting, all your content is accessible to web browsers via the Amazon S3 website endpoint for your bucket.

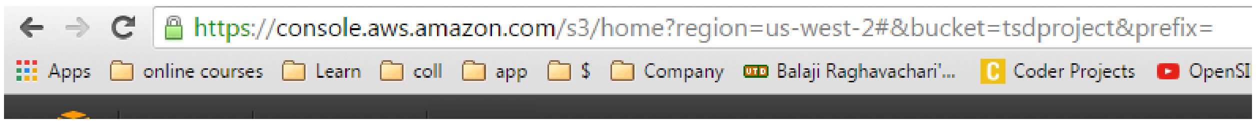
Endpoint: tsdproject.s3-website-us-west-2.amazonaws.com

Each bucket serves a website namespace (e.g. "www.example.com"). Requests for your host name (e.g. "example.com" or "www.example.com") can be routed to the contents in your bucket. You can also redirect requests to another host name (e.g. redirect "example.com" to "www.example.com"). See our walkthrough for how to set up an Amazon S3 static website with your host name.

[✓ Do not enable website hosting] Enable website hosting Redirect all requests to another host name

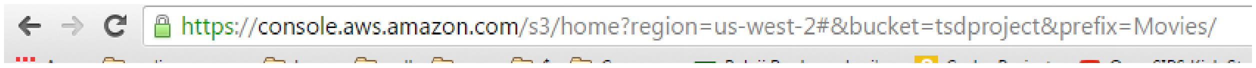
[Figure 11]

- Figure 12: The figure below shows the unique URL for bucket TSDProject.



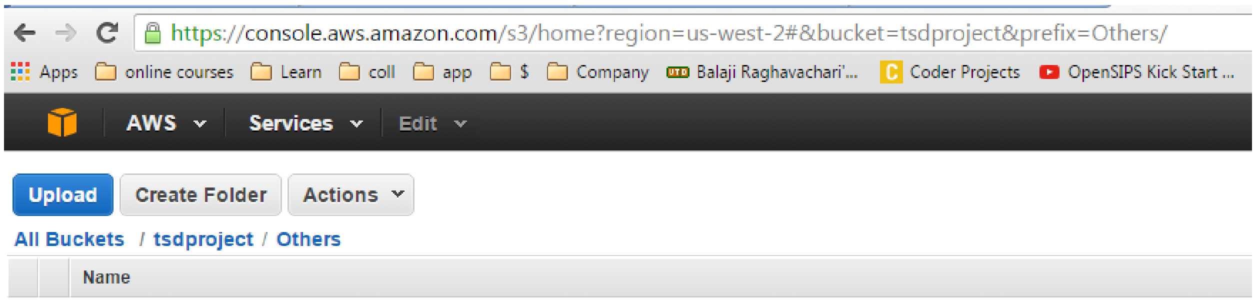
https://console.aws.amazon.com/s3/home?region=us-west-2#&bucket=tsdproject&prefix=

- Figure 13: Figure shown below shows the unique URL for “Movie” folder created inside “tsdproject” bucket.

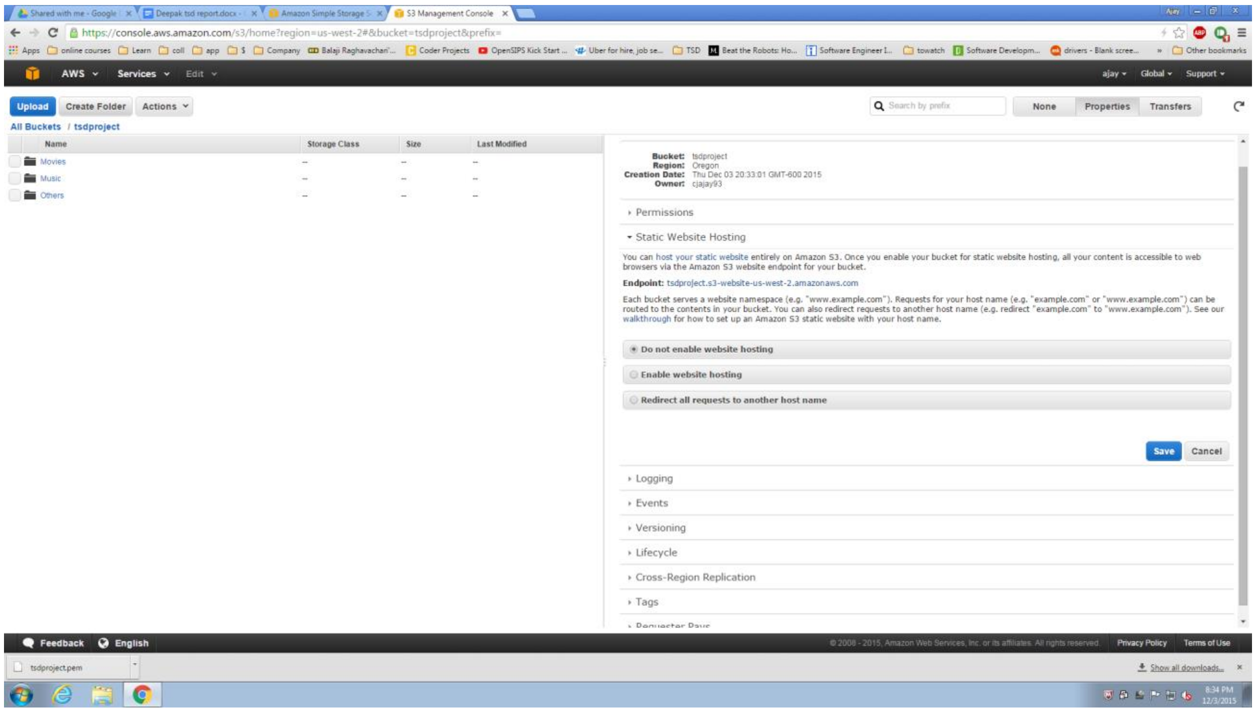


https://console.aws.amazon.com/s3/home?region=us-west-2#&bucket=tsdproject&prefix=Movies/

- Figure 14: The image below shows unique URL for “Others” folder on “tsdproject” Bucket



https://console.aws.amazon.com/s3/home?region=us-west-2#&bucket=tsdproject&prefix=Others/



https://console.aws.amazon.com/s3/home?region=us-west-2#&bucket=tsdproject&prefix=

All Buckets / tsdproject

Name	Storage Class	Size	Last Modified
Movies	--	--	--
Music	--	--	--
Others	--	--	--

Bucket: tsdproject
Region: Oregon
Creation Date: Thu Dec 03 20:33:01 GMT-600 2015
Owner: cjajay93

Permissions

Static Website Hosting

You can host your static website entirely on Amazon S3. Once you enable your bucket for static website hosting, all your content is accessible to web browsers via the Amazon S3 website endpoint for your bucket.

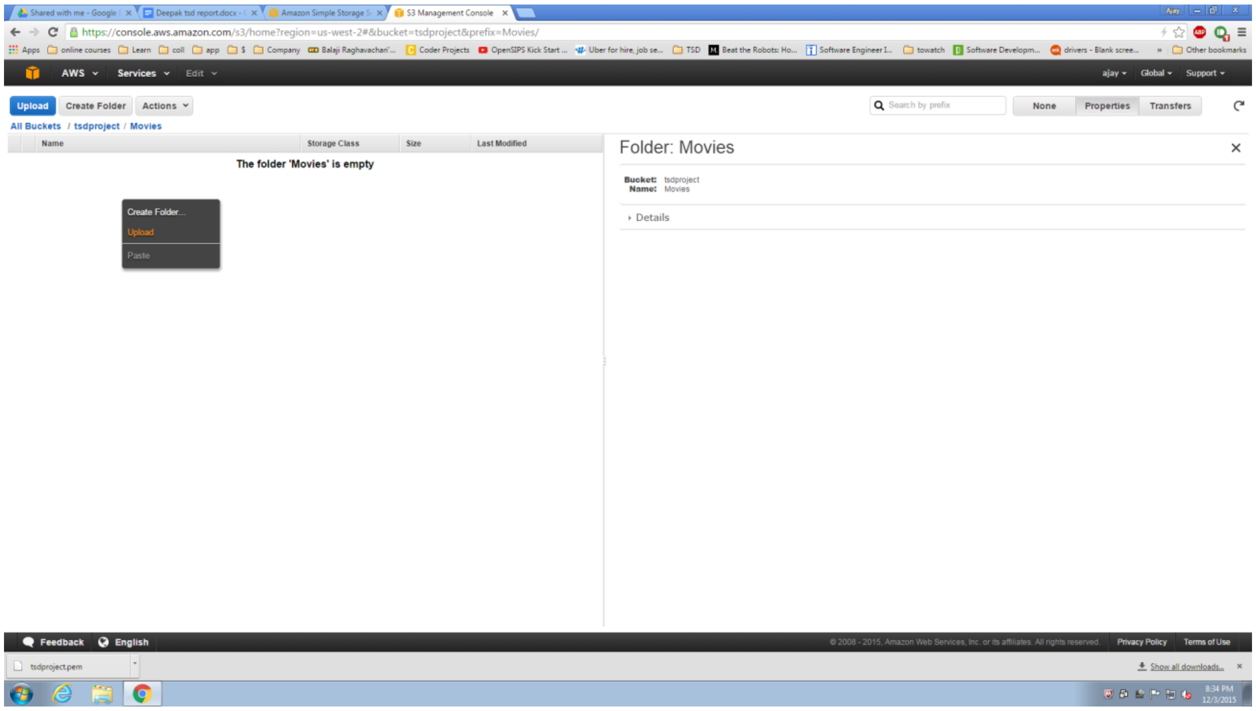
Endpoint: tsdproject.s3-website-us-west-2.amazonaws.com

Each bucket serves a website namespace (e.g. "www.example.com"). Requests for your host name (e.g. "example.com" or "www.example.com") can be routed to the contents in your bucket. You can also redirect requests to another host name (e.g. redirect "example.com" to "www.example.com"). See our walkthrough for how to set up an Amazon S3 static website with your host name.

[✓ Do not enable website hosting]
Enable website hosting
Redirect all requests to another host name

[Figure 15]

- <u>Figure 16:</u> Each folder gives an option to upload as shown below.



https://console.aws.amazon.com/s3/home?region=us-west-2#&bucket=tsdproject&prefix=Movies/

All Buckets / tsdproject / Movies

The folder 'Movies' is empty

Folder: Movies

Bucket: tsdproject
Name: Movies

[Figure 16]

Note:

- It is required to make each folder, object in the bucket public. Or else, they cannot be accessed externally.
- To obtain programmatic access to S3, IAM rules need to be created. Additional details can be found about IAM can be found in the following links:
  - https://console.aws.amazon.com/iam/home#
  - http://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html
- Documentation on AWS SDK for Java can be accessed at: https://aws.amazon.com/sdk-for-java/
- Example codes for AWS Java can be found at: https://github.com/aws/aws-sdk-java/tree/master/src/samples/AmazonS3

References:

- Java SDK for S3: http://javatutorial.net/java-s3-example
- REST API:
  - http://crunchify.com/how-to-build-restful-service-with-java-using-jax-rs-and-jersey/
  - http://www.mkyong.com/webservices/jax-rs/jersey-hello-world-example/
- VLCj: http://capricasoftware.co.uk/#/projects/vlcj/tutorial/first-steps
- OkHTTP: http://square.github.io/okhttp/
- Jetty HTTP/2 Server: https://webtide.com/introduction-to-http2-in-jetty/
- Android: http://hmkcode.com/android-parsing-json-data/