Cloud Vault

Storage as a Service solution

Project report

Submitted by

Shudhanshu Singh

Univ. Roll No. 2011763

Under the Guidance of

Dr. D. R. Gangodkar
(Professor)

In partial fulfilment for the

Award of the degree of

Bachelor of technology

In Computer Science and Engineering

Department of Computer Science and Engineering

Graphic Era (deemed to be University)

Dehradun, Uttarakhand - 248002

June-July 2019

GraphicEra
(Deemed to be University)
Accredited by NAAC with Grade A

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the project report entitled **"Cloud Vault"** is submitted in the Department of Computer Science and Engineering of the Graphic Era (deemed to be University), Dehradun is an authentic record of my own work carried out during a period from **5th June 2019 to 20th July 2019** under the supervision of **Dr. D.R. Gangodkar**, Professor, Department of Computer Science and Engineering of the Graphic Era (deemed to be University), Dehradun.

The matter presented in this report has not been submitted by me for the award of any other degree of this or any other Institute/University.

Shudhanshu Singh
Univ. Roll No. 2011763

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Signature Head of Department

(Dr. D.R. Gangodkar)
Supervisor

Abstract

The main objective of this project was to learn using AWS PHP SDKs' for S3 and Cognito by implementing them, to do so I've built a web-based application which is a storage solution for the users of that application by allocating them a private and scalable object-based storage by using amazons S3.

This application guarantees 99.99% availability and 99.999999999% durability of users' data. It's a scalable, highly available, fault tolerant as it deployed on AWS leveraging various services such as ec2, elastic load balancers and auto-scaling groups of AWS.

The specialty about this solution is that users can upload up to 5GB of object file at once and there is no limit to storage per user.

The knowledge gained during building this application would be helpful in associating amazon S3 and Cognito with web applications and would also be helpful in architecting scalable and highly available infrastructure on AWS.

Acknowledgement

I have taken efforts in this project. However, it would not have been possible without the kind support of many individuals and organizations. I would like to extend my sincere thanks to all of them.

Before I introduce my project, I would like to give my deepest gratitude and special thanks to Prof. (Dr.) D. R. Gangodkar who despite being extraordinarily busy with his duties, took time out to hear, guide and keep me on the correct path and allowing me to carry out the project at their esteemed organization. I choose this moment to acknowledge his contribution to this project.

I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, to attain desired career objectives.

Index

List of Figures

Chapter 1

Introduction

1.1 What is Cloud Vault?

Cloud Vault is storage as a service solution for users by providing them unlimited storage for storing and accessing their data remotely and also providing them the capability of downloading their data from anywhere and from any device over any network. It guarantees the availability of 99.99% of users' data along with the durability of 99.999999999%.

It's a fully cloud-based application without any physical infrastructure as it's fully deployed on AWS. It's a highly available, fault-tolerant and fully scalable solution for its users being cost-efficient and reliable also.

1.2 Traditional Storage (On-Premise)

It is the one which we were using and are using to store the data as part of disaster management. In traditional storage we store the data in the computer or servers connected with our computer using LAN/WAN. The data informations are stored in disks which can be reformatted or reprovisioned as and when needed. The number of disks can be increased in order to scale up the storage limit.

In traditional storage type, when storage limit requirements are increased, secondary backup devices and even third party websites are used to store the excess data. Moreover data is stored in multiple office locations to avoid complete loss of data in case of disaster or malfunctioning of equipments.

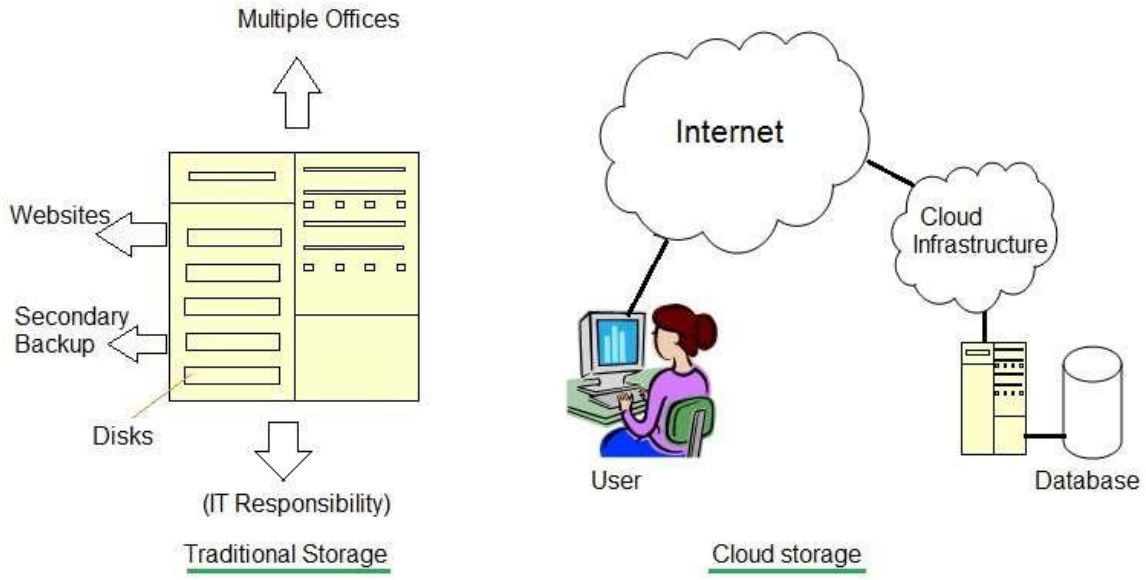

Figure 1.1: On-Premise and Off-Premise storage [1]

1.2.1 On-Premise Storage vs Cloud Storage

On-Premise Storage

- Storage resources are procured, owned and managed by the enterprise.
- The enterprise is responsible for securing the storage resources and data.
- Storage resources remain dedicated to the company.
- The investment is considered CapEx, which is a typically a high cost.

Cloud Storage

- Storage resources are owned and managed by a third party.
- Storage resources may be purchased on a pre-paid or pay-as-you-go basis.
- Storage resources may be shared in a multi-tenant environment.
- Software is kept up-to-date as part of an active subscription.
- IT does not have to install software updates and patches.
- The investment is considered OpEx, which is a lower monthly cost.

Chapter 2

AWS Architecture and Application Architecture

2.1 AWS Architecture

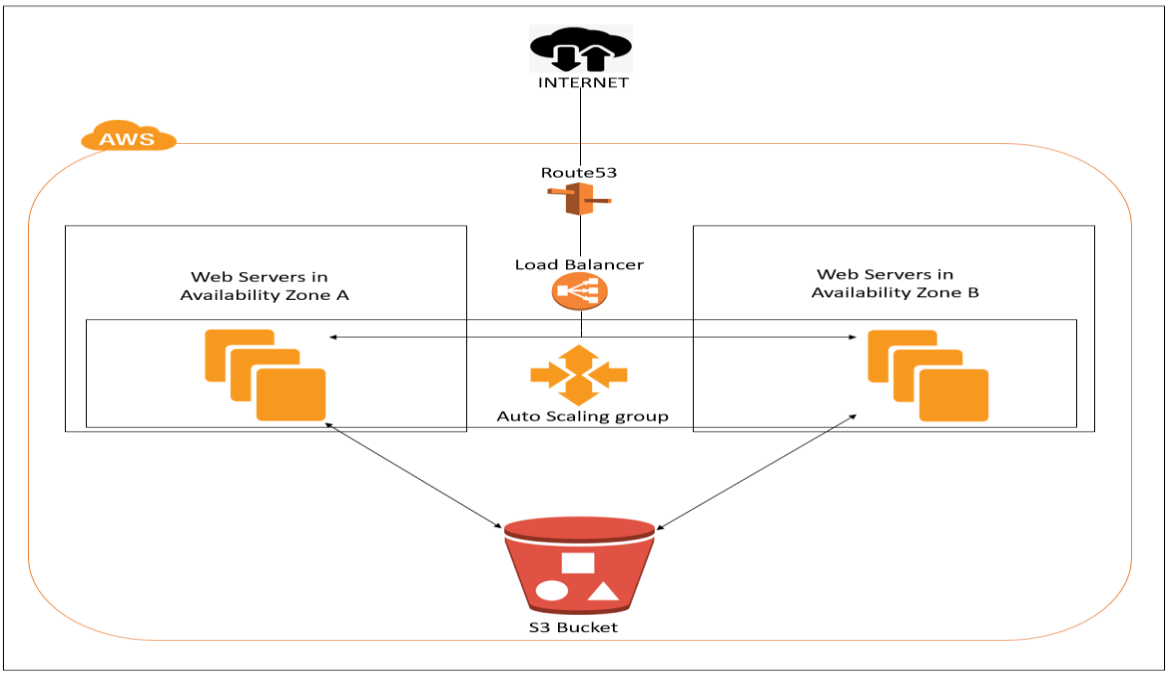

Figure 2.1: AWS Architecture

2.1.1 Architecture Key Services

Route 53

Amazon Route 53 (**Route 53**) [2] is a scalable and highly available Domain Name System (DNS) service. Released on December 5, 2010, it is part of Amazon.com's cloud computing platform, Amazon Web Services (AWS). The name is a reference to TCP or UDP port 53, where DNS server requests are addressed. In addition to being able to route users to various AWS services, including EC2 instances, Route 53 also enables AWS customers to route users to non-AWS infrastructure and to monitor the health of their application and its endpoints. Route 53's servers are distributed throughout the world. Amazon Route 53 supports full, end-to-end DNS resolution over IPv6. Recursive DNS resolvers on IPv6 networks can use either IPv4 or IPv6 transport to send DNS queries to Amazon Route 53,

Amazon Route 53 effectively connects user requests to infrastructure running in AWS – such as Amazon EC2 instances, Elastic Load Balancing load balancers, or Amazon S3 buckets – and can also be used to route users to infrastructure outside of AWS. You can use Amazon Route

53 to configure DNS health checks to route traffic to healthy endpoints or to independently monitor the health of our application and its endpoints. Amazon Route 53 Traffic Flow makes it easy for you to manage traffic globally through a variety of routing types, including Latency Based Routing, Geo DNS, Geo proximity, and Weighted Round Robin—all of which can be combined with DNS Failover in order to enable a variety of low-latency, fault-tolerant architectures. Using Amazon Route 53 Traffic Flow's simple visual editor, you can easily manage how our end-users are routed to our application's endpoints—whether in a single AWS region or distributed around the globe. Amazon Route 53 also offers Domain Name Registration – you can purchase and manage domain names such as example.com and Amazon Route 53 will automatically configure DNS settings for our domains.

Application Load Balancer

A load balancer serves as the single point of contact for clients. Clients send requests to the load balancer, and the load balancer sends them to targets, such as EC2 instances, in two or more Availability Zones. To configure your load balancer, you create target groups, and then register targets with your target groups. You also create listeners to check for connection requests from clients, and listener rules to route requests from clients to the targets in one or more target groups.

Application Load Balancer [3] operates at the request level (layer 7), routing traffic to targets – EC2 instances, containers, IP addresses and Lambda functions based on the content of the request. Ideal for advanced load balancing of HTTP and HTTPS traffic, Application Load Balancer provides advanced request routing targeted at delivery of modern application architectures, including micro services and container-based applications. Application Load Balancer simplifies and improves the security of your application, by ensuring that the latest SSL/TLS ciphers and protocols are used at all times.

Auto Scaling Group

An Auto Scaling Group [4] contains a collection of Amazon EC2 instances that are treated as a logical grouping for the purposes of automatic scaling and management. An Auto Scaling group also enables you to use Amazon EC2 Auto Scaling features such as health check replacements and scaling policies. Both maintaining the number of instances in an Auto Scaling group and automatic scaling are the core functionality of the Amazon EC2 Auto Scaling service.

The size of an Auto Scaling group depends on the number of instances you set as the desired capacity. You can adjust its size to meet demand, either manually or by using automatic scaling.

An Auto Scaling group starts by launching enough instances to meet its desired capacity. It maintains this number of instances by performing periodic health checks on the instances in

the group. The Auto Scaling group continues to maintain a fixed number of instances even if an instance becomes unhealthy. If an instance becomes unhealthy, the group terminates the unhealthy instance and launches another instance to replace it.

Amazon EC2 (Elastic Compute Cloud)

Amazon Elastic Compute Cloud (Amazon EC2) [5] is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of our computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as our computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate them from common failure scenarios.

Amazon S3

Amazon Simple Storage Service (Amazon S3) [6] is an object storage service that offers industry-leading scalability, data availability, security, and performance. This means customers of all sizes and industries can use it to store and protect any amount of data for a range of use cases, such as websites, mobile applications, backup and restore, archive, enterprise applications, IOT devices, and big data analytics. Amazon S3 provides easy-to-use management features so you can organize your data and configure finely-tuned access controls to meet your specific business, organizational, and compliance requirements. Amazon S3 is designed for 99.999999999% (11 9's) of durability, and stores data for millions of applications for companies all around the world.

2.1.2 Architecture characteristics

- **Highly available**
  This application architecture is using amazons' elastic load balancer service to provide its users a highly available application by routing the traffic to multiple instances located in two availability zones.
- **Fault tolerant**
  The ec2 instances on which our web application is deployed sits behind an auto scaling group which keeps fix number of ec2 instances always up and in case of high traffic it also scale out instances in order to keep the application highly available.

- **Data availability and durability**
  For data storage this application is using amazon S3 which is an object storage service that offers industry-leading scalability, data availability, security, and performance and also designed for durability of 99.999999999% and 99.99% availability of objects across multiple Availability Zones.

2.2 Application Architecture

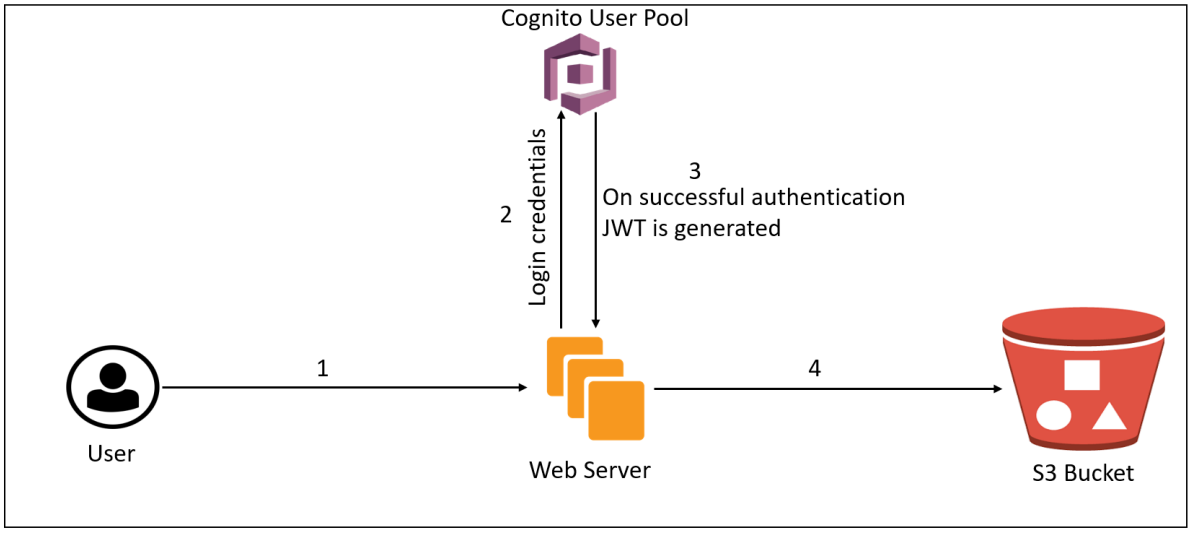

Figure 2.2: Application Architecture

2.2.1 Data flow in application architecture

1: User to Web Server- In this, the web application is accessed by the user via the internet.

2: Web Server to Cognito- In this user get signed up on web application and after that he/she tries to log in his/her account with login credentials.

3: Cognito to Web Server- If successful authentication takes place then only the Cognito generates a JWT token.

4: Web Server to S3 bucket- After successful authentication of user credentials, JWT token is used to set a successful session to his/her account.

2.2.2 Cognito User Pool

A user pool [7] is a user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito. Your users can also sign in through social identity providers like Facebook or Amazon, and through SAML identity providers. Whether your users sign in directly or through a third party, all members of the user pool have a directory profile that you can access through an SDK.

User pools provide:

- Sign-up and sign-in services.
- A built-in, customizable web UI to sign in users.
- Social sign-in with Facebook, Google, and Login with Amazon, as well as sign-in with SAML identity providers from your user pool.
- User directory management and user profiles.
- Security features such as multi-factor authentication (MFA), checks for compromised credentials, account takeover protection, and phone and email verification.
- Customized workflows and user migration through AWS Lambda triggers.

After successfully authenticating a user, Amazon Cognito issues JSON web tokens (JWT) that you can use to secure and authorize access to your own APIs, or exchange for AWS credentials.

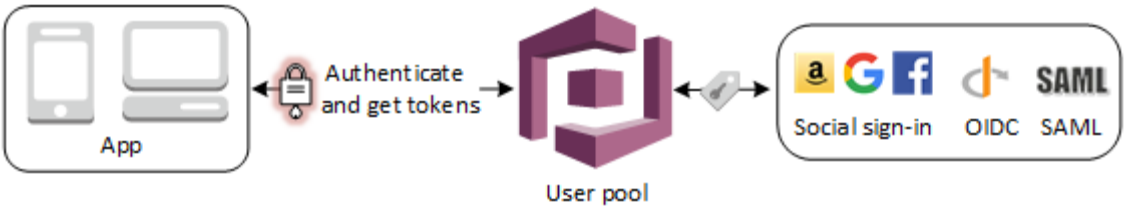

Figure 2.3: Cognito User Pool [7]

Amazon Cognito provides token handling through the Amazon Cognito User Pools Identity SDKs for JavaScript, Android, and iOS.

The two main components of Amazon Cognito are user pools and identity pools. Identity pools provide AWS credentials to grant your users access to other AWS services. To enable users in your user pool to access AWS resources, you can configure an identity pool to exchange user pool tokens for AWS credentials.

Steps for configuring Cognito User pool

Step1: Login to your aws console and go to Cognito service and select the option for creating a user pool

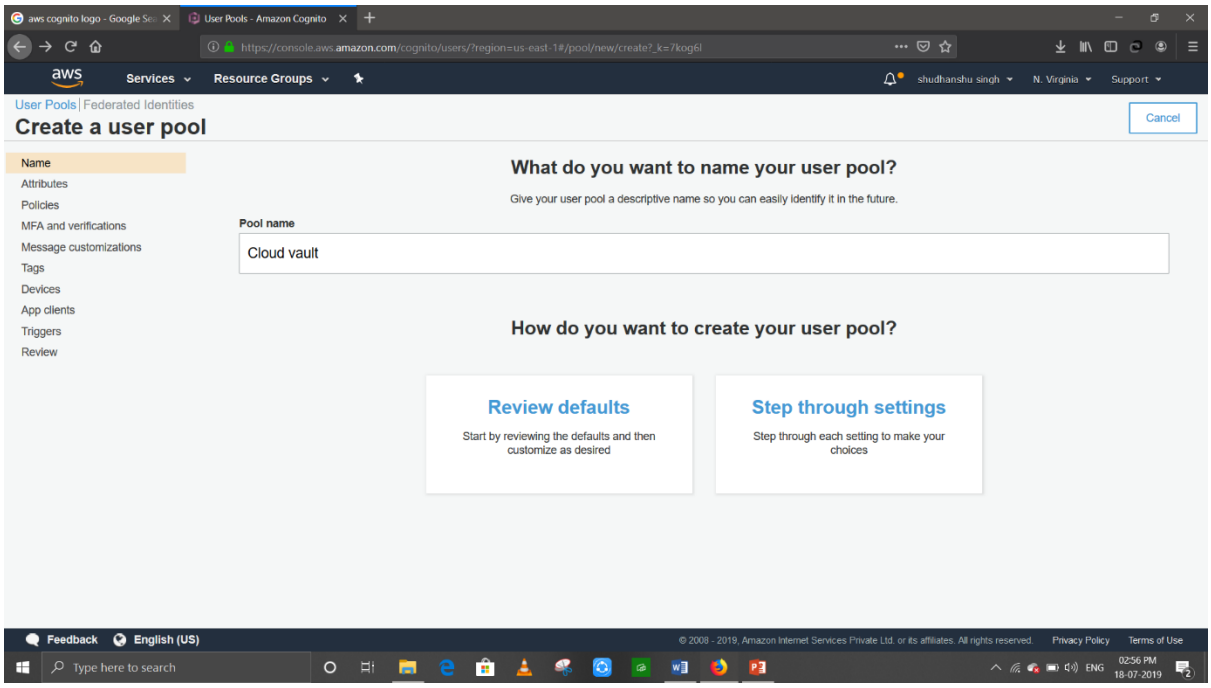

Figure 2.3.1: Entering a Pool name

Step2: Select the attributes you want to include in your sign in and sign up

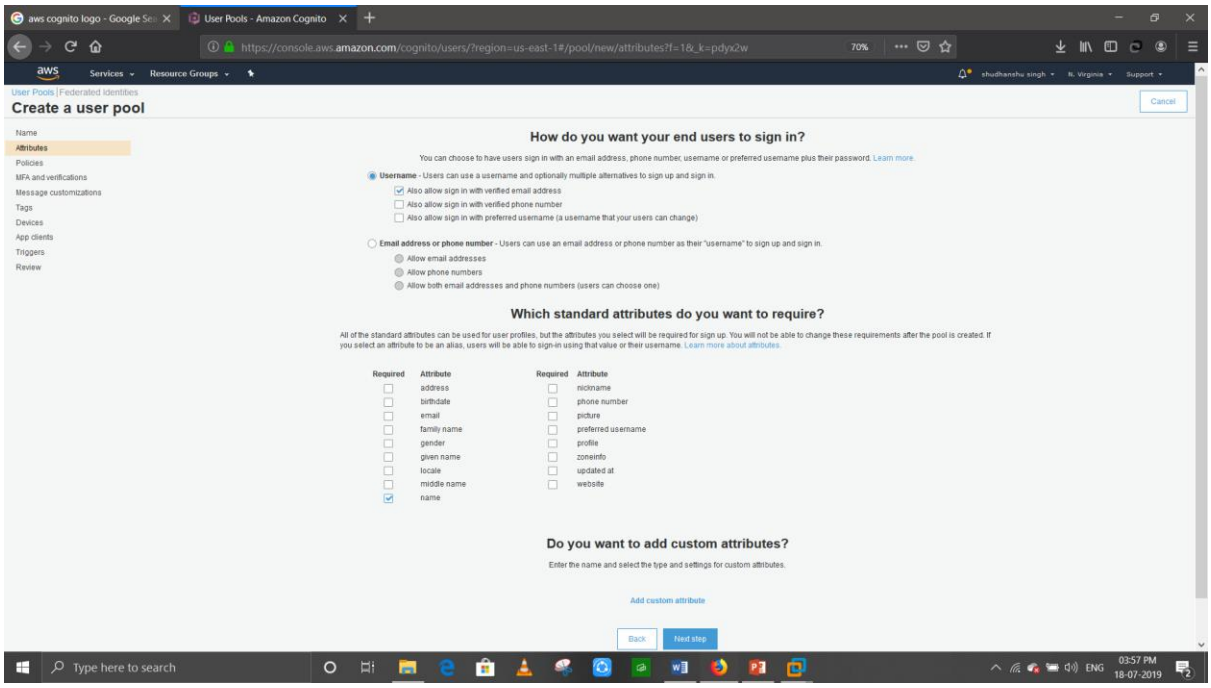

Figure 2.3.2: Selecting sign up attributes

Step3: Select password policies for your sign up

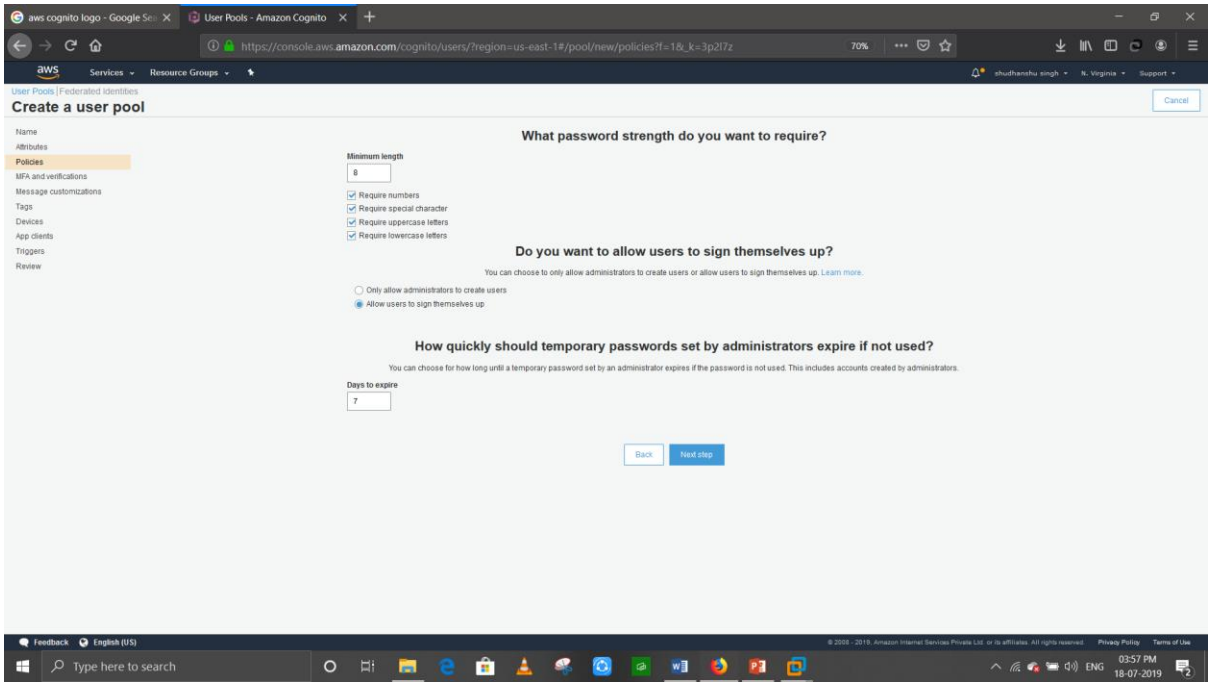

Figure 2.3.3: Cognito user pool password policies

Step4: Leave out all next steps as default and go to review and select create pool option

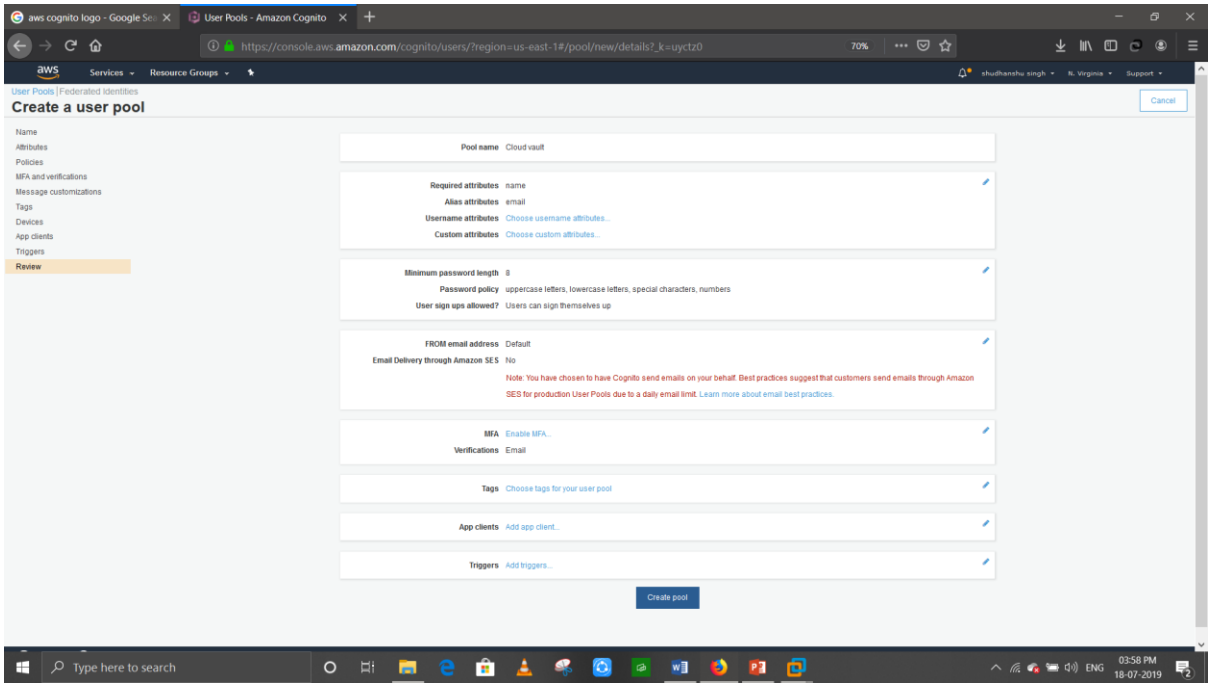

Figure 2.3.4: Creating a user pool

Step5: Add an app clients to enable sign-in API for server-based authentication

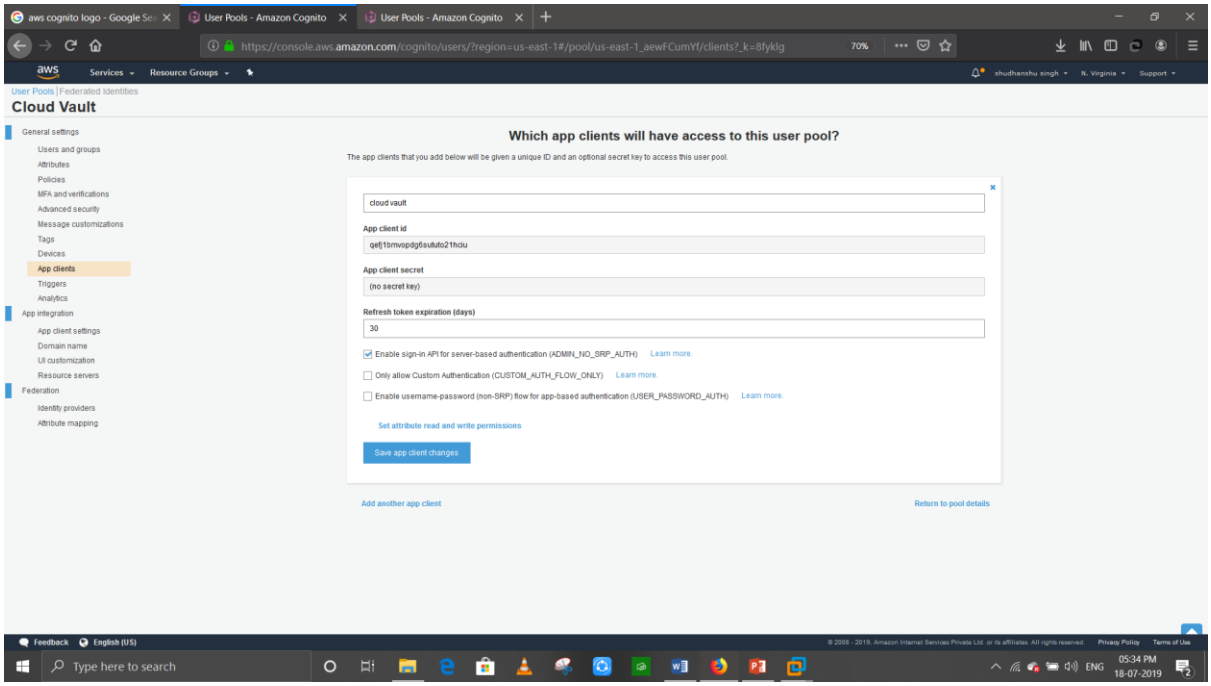

Figure 2.3.5: Adding app clients

Step6: Configure app client settings to configure identity providers and 0Auth 2.0 for your app clients

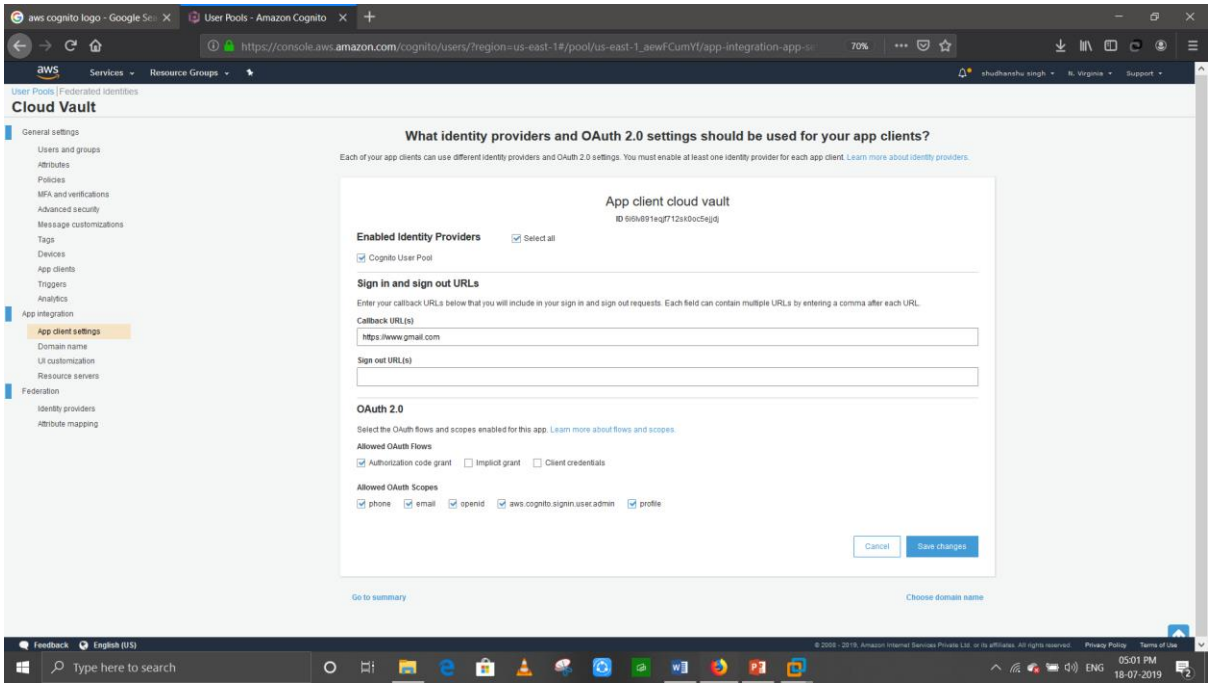

Figure 2.3.6: Configure app clients settings

Chapter 3

Software and Hardware Requirements

This section describes hardware and software requirements that are required to fully deploy this application on AWS.

In terms of hardware requirements, you only require a desktop or laptop with an internet connection with a web browser installed in it.

Whereas in terms of software requirements you require:

1- AWS account: Amazon Web Services (AWS) is a cloud computing department of Amazon that provides on-demand cloud computing platforms to individuals, companies, and governments, on a metered pay-as-you-go basis. In aggregate, these cloud computing web services provide a set of primitive abstract technical infrastructure and distributed computing building blocks and tools.

2- XAMPP: It stands for Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) and Perl (P). It is a simple, lightweight Apache distribution that makes it extremely easy for developers to create a local web server for testing and deployment purposes. Everything needed to set up a web server – server application (Apache), database (MariaDB), and scripting language (PHP) – is included in an extractable file. XAMPP is also cross-platform, which means it works equally well on Linux, Mac and Windows.

3- PHP: The term PHP is an acronym for PHP: Hypertext Preprocessor. PHP is a server-side scripting language designed specifically for web development. PHP can be easily embedded in HTML files and HTML codes can also be written in a PHP file.

4- HTML: It stands for Hyper Text Markup Language. It is used to design web pages using markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. Markup language is used to define the text document within tag which defines the structure of web pages.IT is a markup language which is used by the browser to manipulate text, images and other content to display it in required format.

5- Composer: It is a tool for dependency management in PHP. It allows you to declare the libraries your project depends on and it will manage (install/update) them for you.

6- AWS PHP SDK: The AWS SDK for PHP enables PHP developers to use Amazon Web Services from their PHP code, and build robust applications and software using services like Amazon S3, Amazon DynamoDB, Amazon Glacier, Amazon Cognito, etc.

Chapter 4

Application Design

4.1 User Management [8]

In this application, user management is handled by using PHP SDK for amazon Cognito. We don't have to prepare any database regarding user sign in or sign up credentials, it's all handled by Cognito itself.

4.1.1 User registration

Let's take a look at how to allow users to sign up. The flow that we need to implement is a relatively simple, three-step process:

1. Allow the user to sign up by providing a username, email and password and any other attributes if desired.
2. If no errors are thrown during the signup (such as an already taken username or a password that does not adhere to the requirements), an e-mail is sent containing a confirmation code. If a phone number is known and SMS is properly configured with AWS' Simple Notification Service (SNS), sending a text message is an option as well.
3. Registration is confirmed by providing the username and confirmation code found in the e-mail together.

The signup API call looks as follows:

```
$client->signUp([
  'ClientId' => CLIENT_ID,
  'Username' => $username,
  'Password' => $password,
  'UserAttributes' => [
    [
      'Name' => 'name',
      'Value' => $username
    ],
    [
      'Name' => 'email',
      'Value' => $email
    ]
  ],
]);
```

By default, a simple e-mail (the template can be changed) is sent that looks as follows:

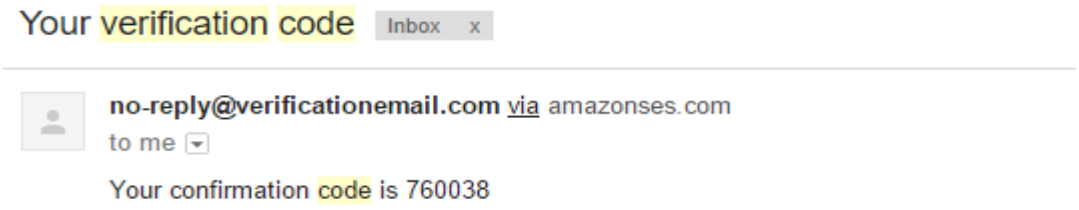

Figure 4.1: Verification e-mail template [8]

Next, given the confirmation code, a user can be confirmed with the following API call:

```
$client->confirmSignUp([
  'ClientId' => CLIENT_ID,
  'Username' => $username,
  'ConfirmationCode' => $code,
]);
```

4.1.2 User Login

Users who have successfully registered and confirmed themselves are able to log in by specifying their username and password (for users who attempts to login but are not yet confirmed, a UserNotConfirmedException will be returned by the API). Two different API calls exist for user login: AdminInitiateAuth and InitiateAuth. With only the first being implemented in the PHP SDK, let's start with that one. The API call that we execute looks as follows:

```
$result = $client->adminInitiateAuth([
  'AuthFlow' => 'ADMIN_NO_SRP_AUTH',
  'ClientId' => CLIENT_ID,
  'UserPoolId' => USERPOOL_ID,
  'AuthParameters' => [
    'USERNAME' => $username,
    'PASSWORD' => $password,
  ],
]);

$accessToken = $result->get('AuthenticationResult')['AccessToken'];
```

4.1.3 Resetting a forgotten password

Another fun piece of functionality is to reset a forgotten password. The general flow for this is as follows;

1. The user requests to reset a forgotten password. This is done by sending a verification code to a known contact method: e-mail or SMS.

2. The user specifies a new password and includes the received verification code in the request. This way we know that this is the same person who requested the reset.

Two API calls are required for this. First to request a reset:

```
$client->forgotPassword([
  'ClientId' => CLIENT_ID,
  'Username' => $username
]);
```

After the user specifies the confirmation code and password, we issue the following API call:

```
$client->confirmForgotPassword([
  'ClientId' => CLIENT_ID,
  'ConfirmationCode' => $code,
  'Password' => $password,
  'Username' => $username
]);
```

4.2 Uploading new Objects [9]

For uploading objects to s3 bucket following API call is used

```
$result = $client->putObject(array(
    'Bucket'     => $bucket,
    'Key'        => 'data_from_file.txt',
    'SourceFile' => $pathToFile,
    'Metadata'   => array(
        'Foo' => 'abc',
        'Baz' => '123'
    )
));
```

4.3 Listing of Objects in S3 bucket [9]

Listing objects is a lot easier in the new SDK thanks to *iterators*. You can list all of the objects in a bucket using the ListObjectsIterator.

```
$iterator = $client->getIterator('ListObjects', array(
    'Bucket' => $bucket
));

foreach ($iterator as $object) {
    echo $object['Key'] . "\n";
}
```

4.4 Downloading Objects from S3 bucket [10]

In this application, I've used Pre-Signed URL for downloading objects. We can create pre-signed URLs for any Amazon S3 operation using the getCommand method for creating a command

object, and then calling the createPresignedRequest() method with the command. When ultimately sending the request, be sure to use the same method and the same headers as the returned request. Sample API used in creating pre-signed url is as following:

```
$s3Client = new Aws\S3\S3Client([
    'profile' => 'default',
    'region' => 'us-east-2',
    'version' => '2006-03-01',
]);

//Creating a presigned URL
$cmd = $s3Client->getCommand('GetObject', [
    'Bucket' => 'my-bucket',
    'Key' => 'testKey'
]);

$request = $s3Client->createPresignedRequest($cmd, '+20 minutes');

// Get the actual presigned-url
$presignedUrl = (string)$request->getUri();
```

Chapter 5

Application Implementation and Testing

5.1 Sign Up module

This module consists of 2 web pages. The 1st page is a signup page which consists of 4 input fields that are a username, full name, email and password which user have to fill in order to create an account and 2nd page is a verification page for your registered e-mail which consists of 2 input fields that are username and verification code.

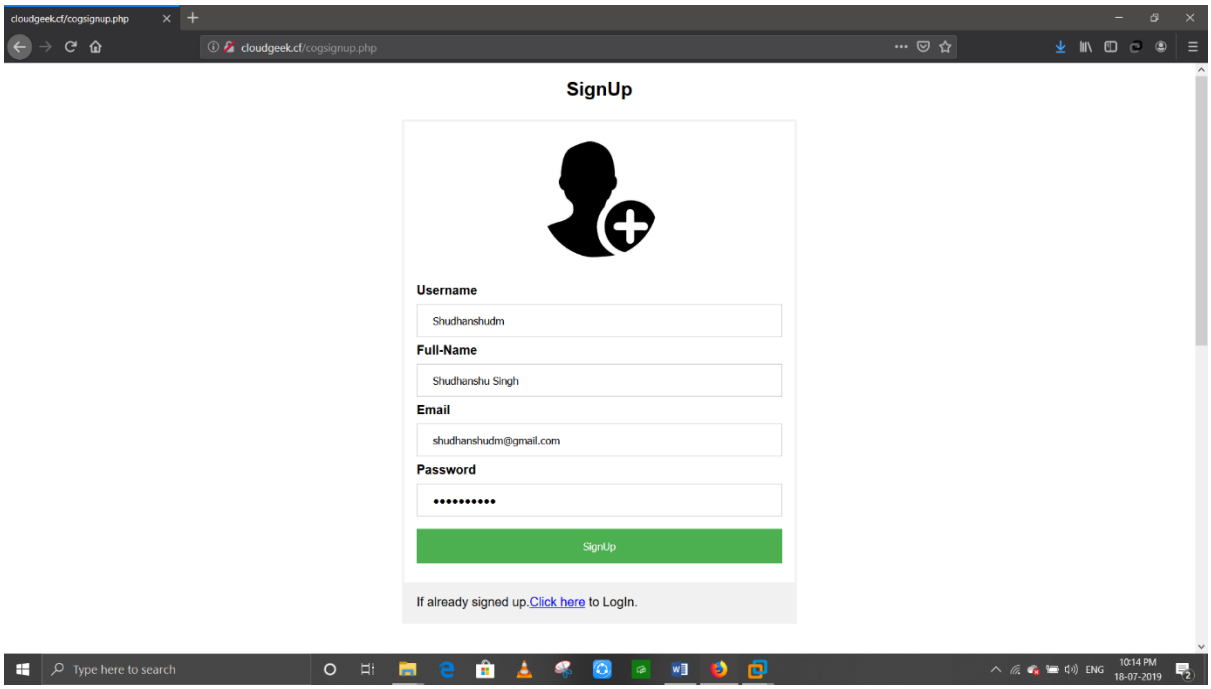

Figure 5.1.1: Sign up page

After this, a mail with verification code would be sent to users registered e-mail

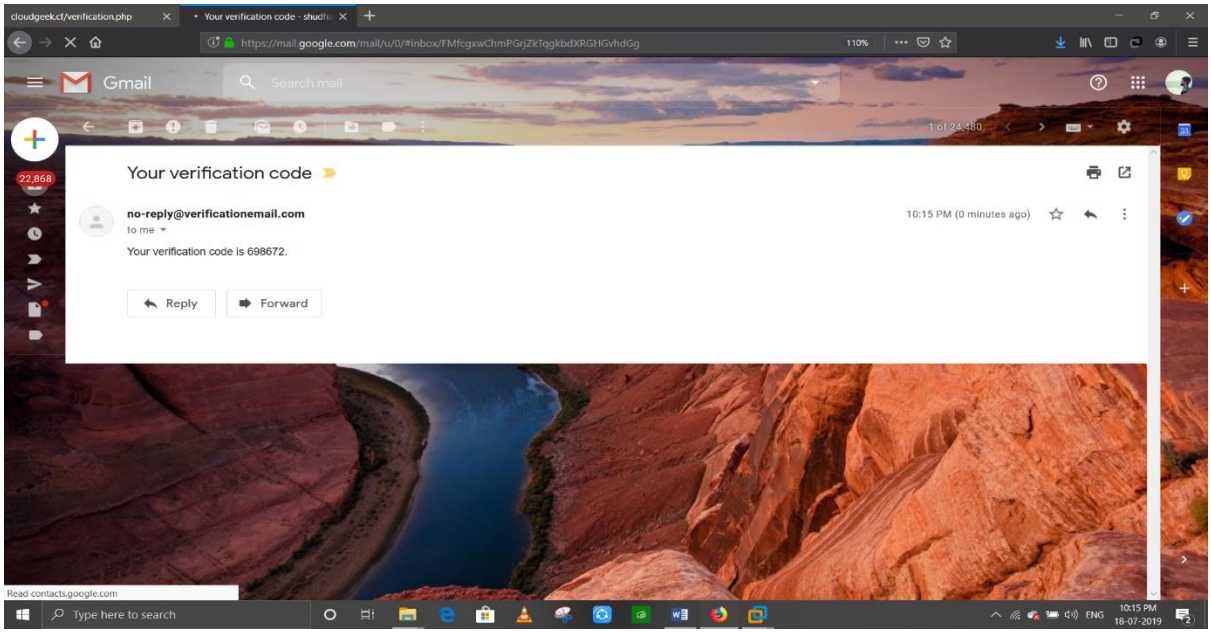

Figure 5.1.2: Verification code mail

After successful verification of e-mail user could log in to his/her account.

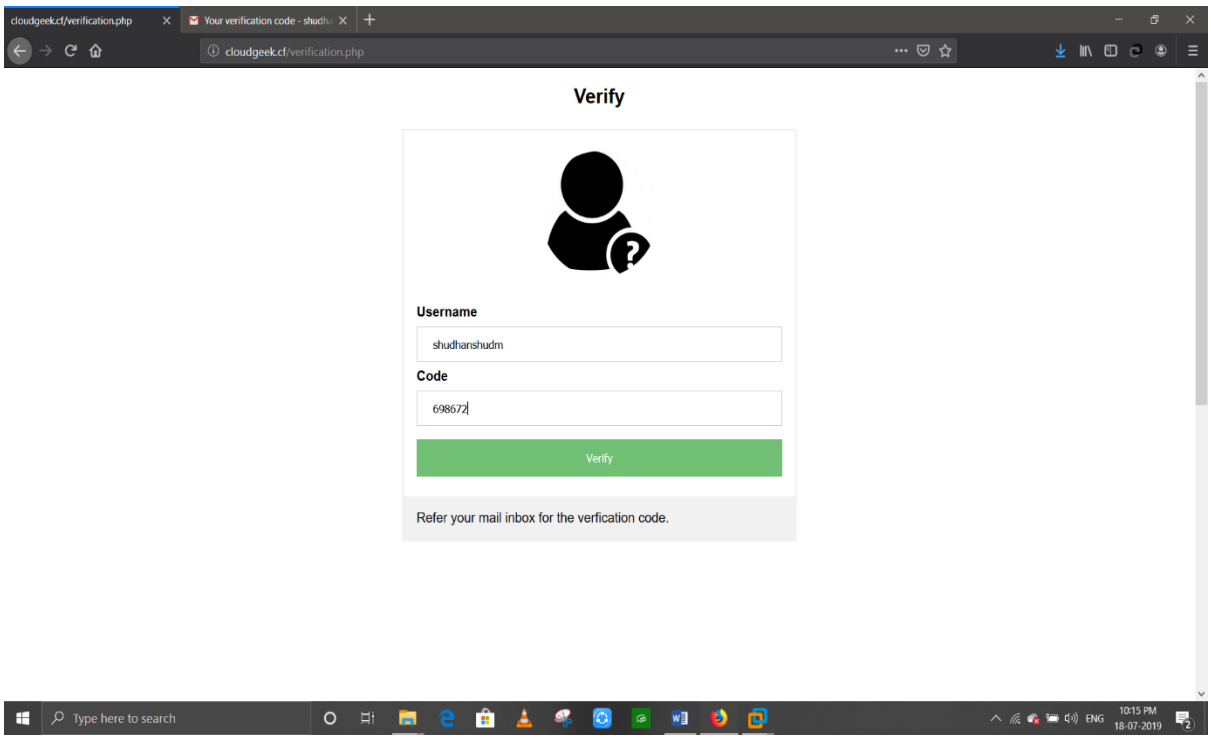

Figure 5.1.3: Successful verification of the account

After successful verification of the user, its username gets available in cognito user pool with CONFIRMED account status.

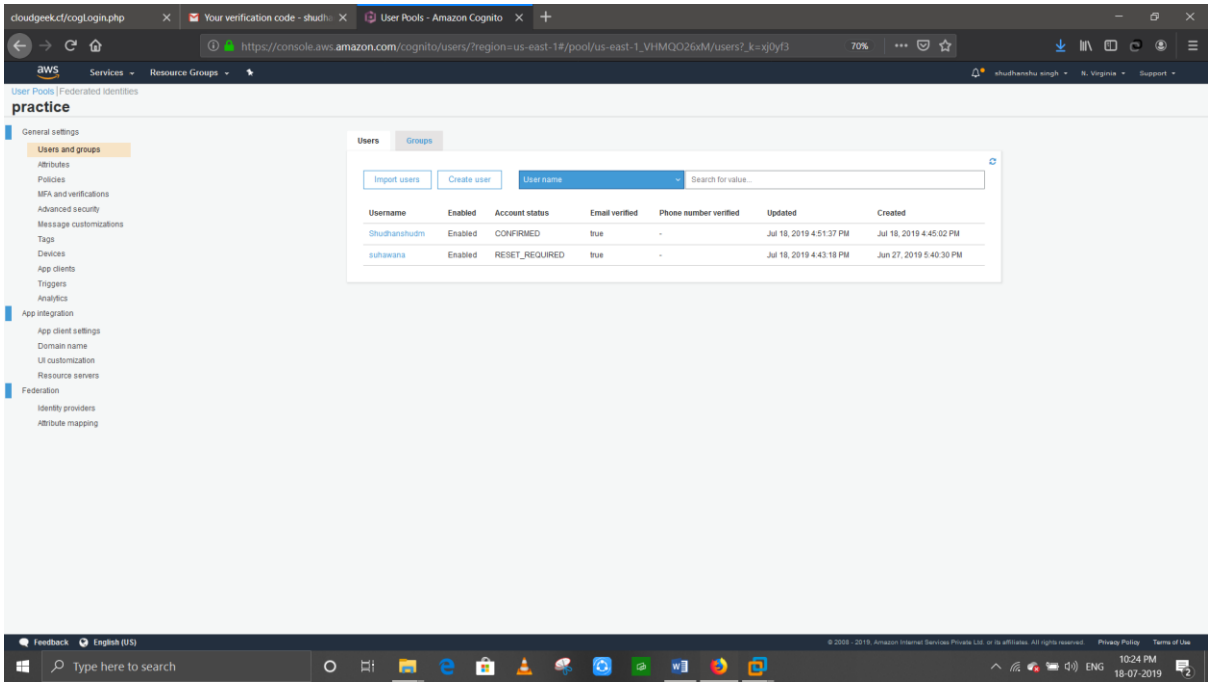

Figure 5.1.4: Entry of the user in user pool

5.2 Login module

This module consists of two web pages 1st one is a login page which has two input fields that are username and password and the 2nd one is a final page where a user could upload, download and delete uploaded objects.

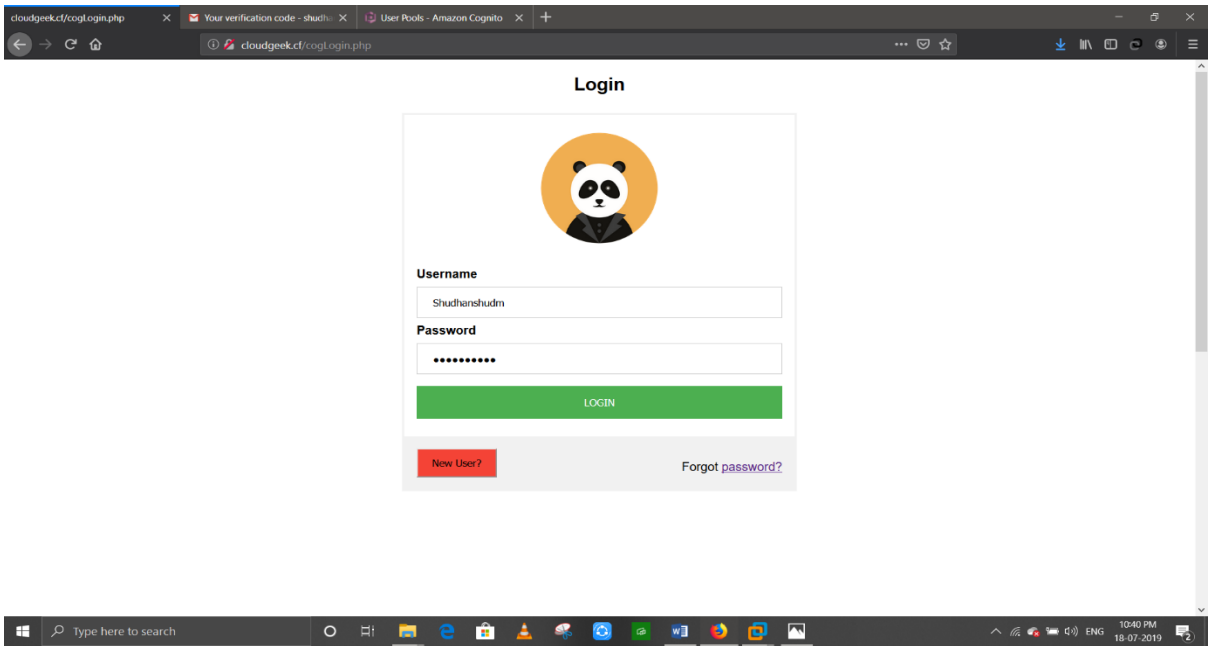

Figure 5.2.1: Login page

After a successful login user can access application final page which would be a page without any object and now on this page user can upload new objects.

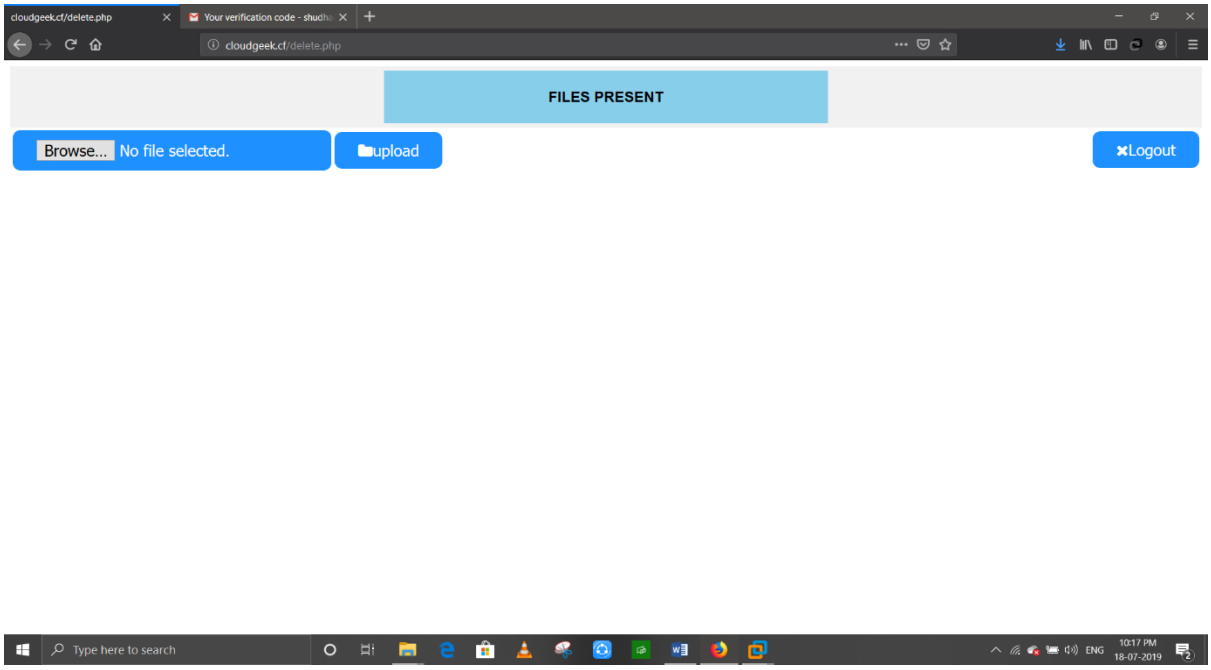

Figure 5.2.2: Final page of the application

After uploading new objects user would be able to see an option for downloading and deleting those objects.

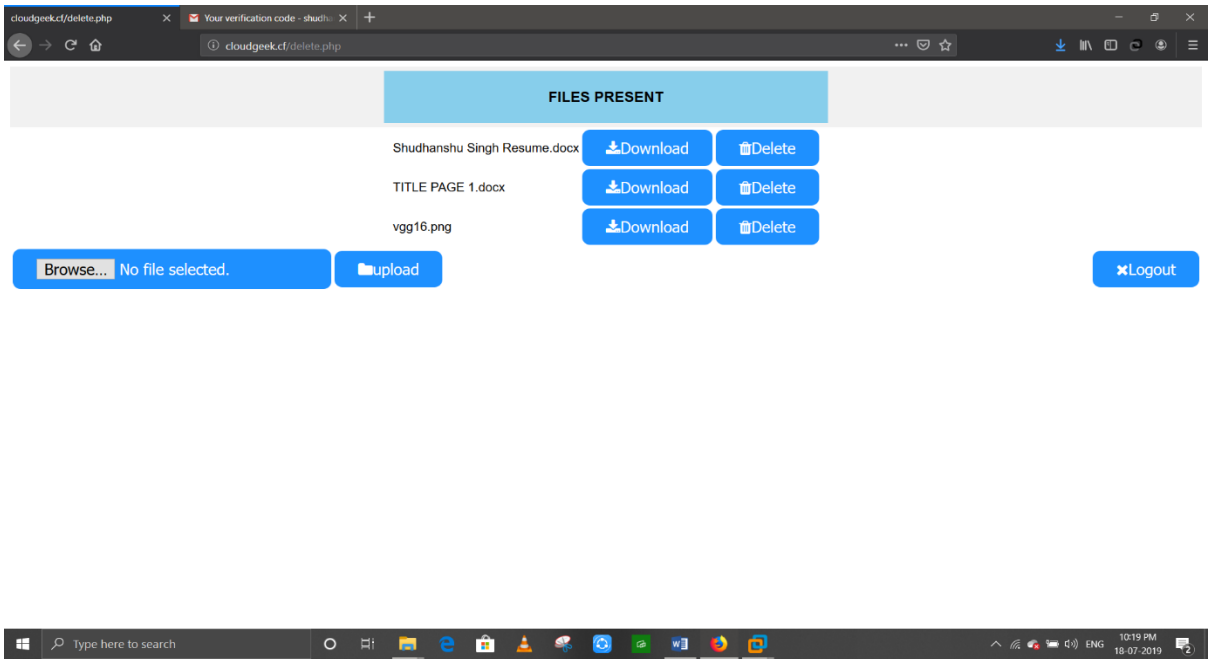

Figure 5.2.3: Listing of objects

5.3 Forgot password module

This module consists of two web pages 1st one is for entering username and the 2nd one is for verifying the new password.

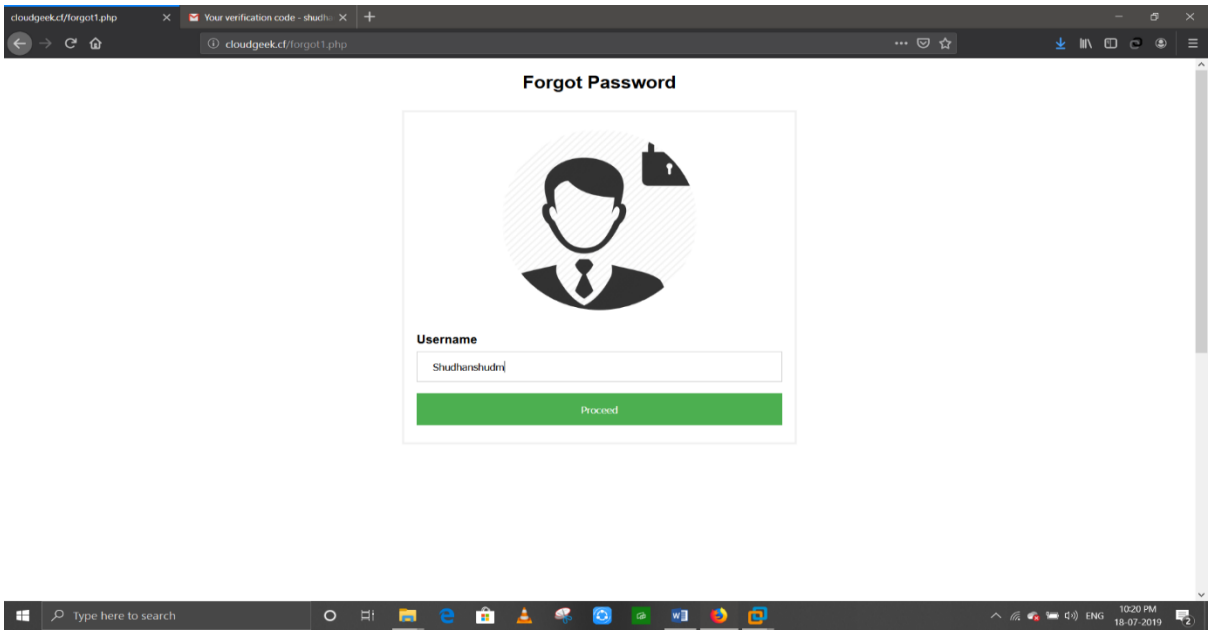

Figure 5.3.1: Forgot password

After entering a valid username user would be redirected to the new password verification page.

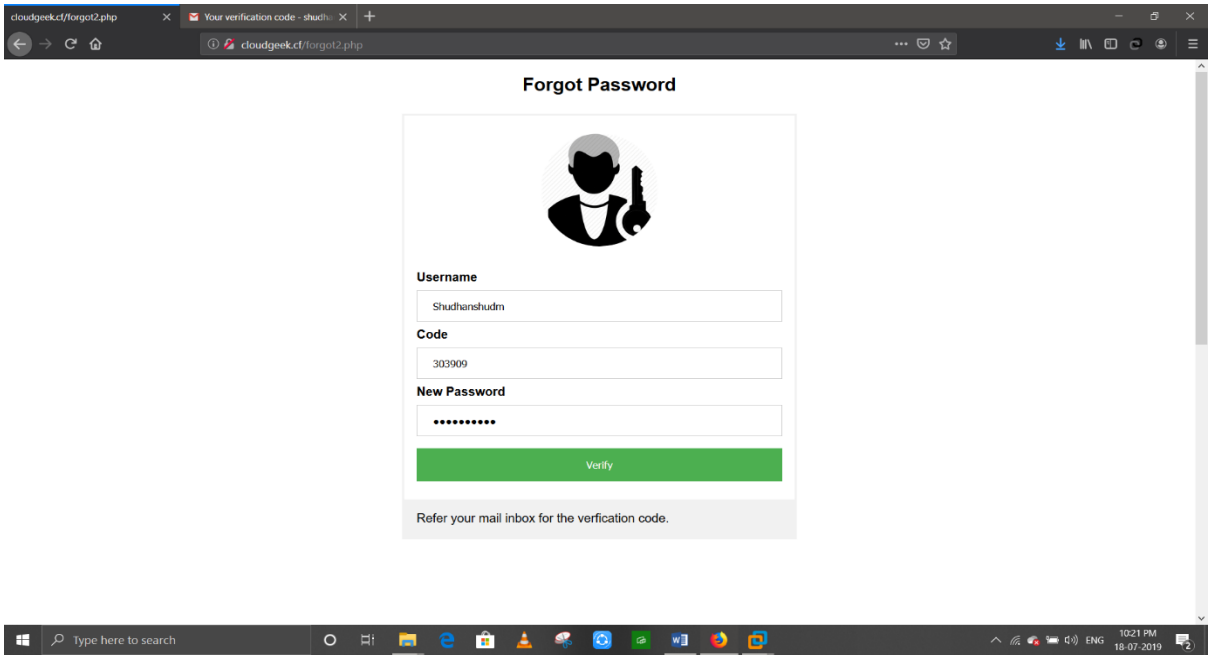

Figure 5.3.2: Verification of new the password

Chapter 6

Conclusion

Through this project, I tried to provide storage as a service solution. What I used for building this solution are various AWS services. I used Amazon S3 for providing unlimited storage to users and for managing those users I used amazon Cognito. The specialty about this solution is that users can upload up to 5GB of object file at once and there is no limit to storage per user and another specialty about this is 99.99% availability and 99.999999999% durability of users' data.

For providing high availability and fault tolerance to this application, I've architected AWS infrastructure for this application by using services like elastic load balancer and auto-scaling groups.

What I have to work further on this solution is as follows:

- Increasing per upload limit of 5GB to 5TB by providing multi-part upload capability to users by using this application.
- Providing capability of archiving data to users.
- Adding data pricing to this solution by which a user can compare its expenses between this solution and its traditional storage solution.

References

Website

[1] Traditional storage vs cloud storage | difference between traditional storage and cloud storage [Online], Available: https://www.rfwireless-world.com/Tutorials/traditional-Storage vs-cloud-storage.html, Accessed on: 20-07-19

[2] Amazon Route 53 [Online], Available: https://aws.amazon.com/route53, Accessed on: 20-07-19

[3] Elastic Load Balancing [Online], Available: https://aws.amazon.com/elasticloadbalancing/, Accessed on: 20-07-19

[4] Amazon EC2 Auto Scaling [Online], Available: https://docs.aws.amazon.com/autoscaling /ec2/userguide/AutoScalingGroup.html Accessed on: 20-07-19

[5] Amazon EC2 [Online], Available: https: //aws.amazon.com/ec2/, Accessed on: 20-07-19

[6] Amazon S3 [Online], Available: https: //aws.amazon.com/s3/, Accessed on: 20-07-19

[7] Amazon Cognito [Online], Available: https://docs.aws.amazon.com/cognito/latest/what-is-amazon-cognito.html, Accessed on: 20-07-19

[8] Getting started with AWS Cognito [Online], Available: https://sanderknape.com/2017/02/getting-started-with-aws-cognito/, Accessed on: 20-07-19

[9] AWS SDK for PHP [Online], Available:https://docs.aws.amazon.com/aws-sdkphp/v2/guide /service-s3.html

[10] Amazon S3 Pre-Signed URL with AWS SDK for PHP Version 3 [Online], Available:https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/s3-presigned-url.html