***Hosting a Static Website on AWS***

***Summary:***

The project aims to host a WebApp on amazon AWS involving the use of two services viz Amazon EC2 and Amazon S3.

The Static website that I have used for this assignment is named as “Beach Wave Conditions”.

The page displays two columns with 24 rows: each row displaying the hour of the day and the height of the waves (in metres) near that beach at a specific hour as shown in the figure. The displayed data is imported from a csv file.

A picture containing treemap chart

Description automatically generated

***Objectives:***

1. Launch an EC2 Instance in a Region.
2. Create an amazon S3 Bucket to store the .zip file of the Website containing all .html, .css, .js files.
3. Launch an Apache Server using PuTTY and host the WebApp.

***Background:***

# Amazon EC2: It is a Secure and resizable compute capacity for virtually any workload. Amazon Elastic Compute Cloud (Amazon EC2) offers the broadest and deepest compute platform, with over 500 instances and choice of the latest processor, storage, networking, operating system, and purchase model to help you best match the needs of your workload. We are the first major cloud provider that supports Intel, AMD, and Arm processors, the only cloud with on-demand EC2 Mac instances, and the only cloud with 400 Gbps Ethernet networking. We offer the best price performance for machine learning training, as well as the lowest cost per inference instances in the cloud. More SAP, high performance computing (HPC), ML, and Windows workloads run on AWS than any other cloud.

**Amazon S3:** Amazon Simple Storage Service (Amazon S3) is an object storage service offering industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can store and protect any amount of data for virtually any use case, such as data lakes, cloud-native applications, and mobile apps. With cost-effective storage classes and easy-to-use management features, you can optimize costs, organize data, and configure fine-tuned access controls to meet specific business, organizational, and compliance requirements.

**PuTTY:** PuTTY is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source) [terminal emulator](https://en.wikipedia.org/wiki/Terminal_emulator), [serial console](https://en.wikipedia.org/wiki/Serial_console) and network file transfer application. It supports several [network protocols](https://en.wikipedia.org/wiki/Network_protocol), including [SCP](https://en.wikipedia.org/wiki/Secure_copy), [SSH](https://en.wikipedia.org/wiki/Secure_Shell), [Telnet](https://en.wikipedia.org/wiki/Telnet), [rlogin](https://en.wikipedia.org/wiki/Rlogin), and raw socket connection. It can also connect to a [serial port](https://en.wikipedia.org/wiki/Serial_port). It is the world's most popular free SSH client. It supports [SSH](https://www.ssh.com/ssh/protocol), [telnet](https://www.ssh.com/ssh/telnet), and raw socket connections with good terminal emulation. It supports [public key authentication](https://www.ssh.com/ssh/public-key-authentication) and Kerberos single-sign-on. It also includes command-line [SFTP](https://www.ssh.com/ssh/sftp) and [SCP](https://www.ssh.com/ssh/scp) implementations.

***Web Application Hosting on AWS***

Web Apps have always been required to be hosted on servers and are assigned an IP that allows the users anywhere in the world to access the webapps using that IP. PuTTY allows the integration of the instance IP with the S3 Bucket Objects and launches an Apache Server to host the webapp.

Traditional on-premises web architectures require complex solutions and accurate reserved capacity forecast in order to ensure reliability. Dense peak traffic periods and wild swings in traffic patterns result in low utilization rates of expensive hardware. This yields high operating costs to maintain idle hardware, and an inefficient use of capital for underused hardware.

Amazon Web Services (AWS) provides a reliable, scalable, secure, and highly performing infrastructure for the most demanding web applications. This infrastructure matches IT costs with customer traffic patterns in near-real time.

This whitepaper is meant for IT Managers and System Architects who want to understand how to run traditional web architectures in the cloud to achieve elasticity, scalability, and reliability.

***Static website:*** A static website contains [Web pages](https://techterms.com/definition/webpage) with fixed content. Each page is coded in [HTML](https://techterms.com/definition/html) and displays the same information to every visitor. Static sites are the most basic type of website and are the easiest to create. Unlike [dynamic websites](https://techterms.com/definition/dynamicwebsite), they do not require any Web programming or [database](https://techterms.com/definition/database) design. A static site can be built by simply creating a few HTML pages and publishing them to a Web server.

Since static Web pages contain fixed code, the content of each page does not change unless it is manually updated by the [webmaster](https://techterms.com/definition/webmaster). This works well for small websites, but it can make large sites with hundreds or thousands of pages difficult to maintain. Therefore, larger websites typically use dynamic pages, which can be updated by simply modifying a database record. Static sites that contain a lot of pages are often designed using [templates](https://techterms.com/definition/template). This makes it possible to update several pages at once, and also helps provide a consistent layout throughout the site.

***Lab Files Used:***

URL of the S3 Bucket: <https://bucketwizard.s3.amazonaws.com/labcode.zip>

Drive Link to the zip file of the code: <https://amityedu96491-my.sharepoint.com/:u:/g/personal/viransh_bhardwaj_s_amity_edu/EZUZTO5C_9dNqnF3k6O39MwBTJGprN4OaB1xR7RXIikOIw?e=D5cB0j>

Ipv4 address of the EC2 instance: 3.87.173.113

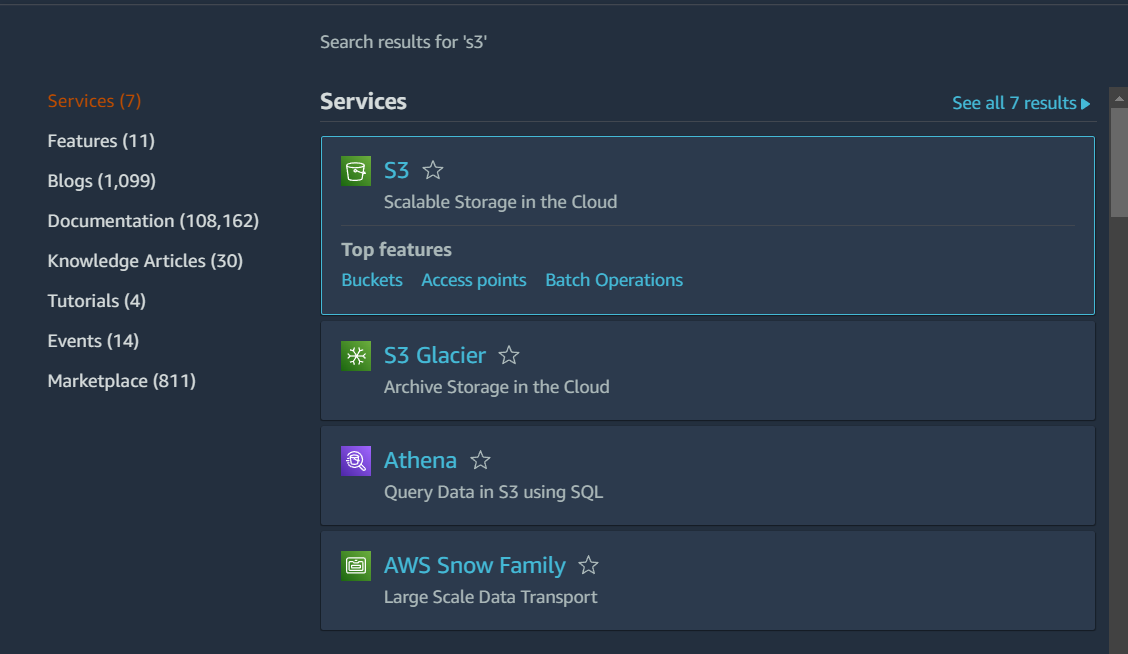
Instance Name: ***webappwizard***

Bucket Name: ***bucketwizard***

Availability Zone: ***us-east-1d***

***Creating an S3 Bucket:***

1. Login to AWS management Console.
2. In the search bar type S3.
3. Select the first option from the results.

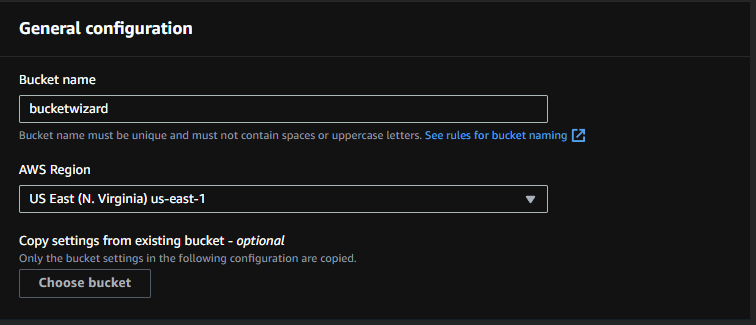


1. In the window click on Create Bucket.

Graphical user interface, text

Description automatically generated

1. Write the name of the bucket you want. Make sure that the name of the bucket is unique.



1. Select ACLs Enabled in the Object ownership. Go with the default Bucket Owner Preferred.

A screenshot of a computer

Description automatically generated with medium confidence

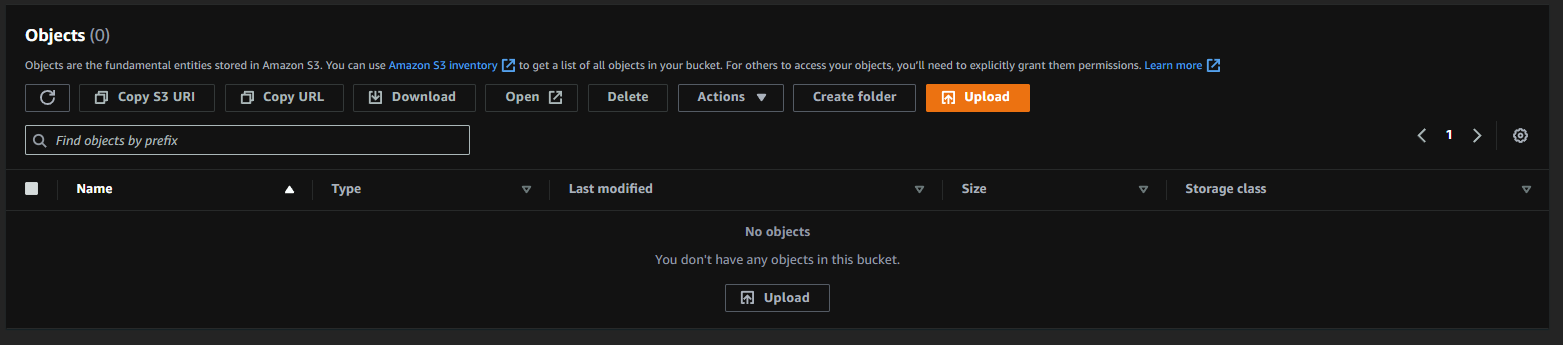
1. Uncheck Block public access checkbox. And Check “I Acknowledge” option under Block Public Access Settings.

Text

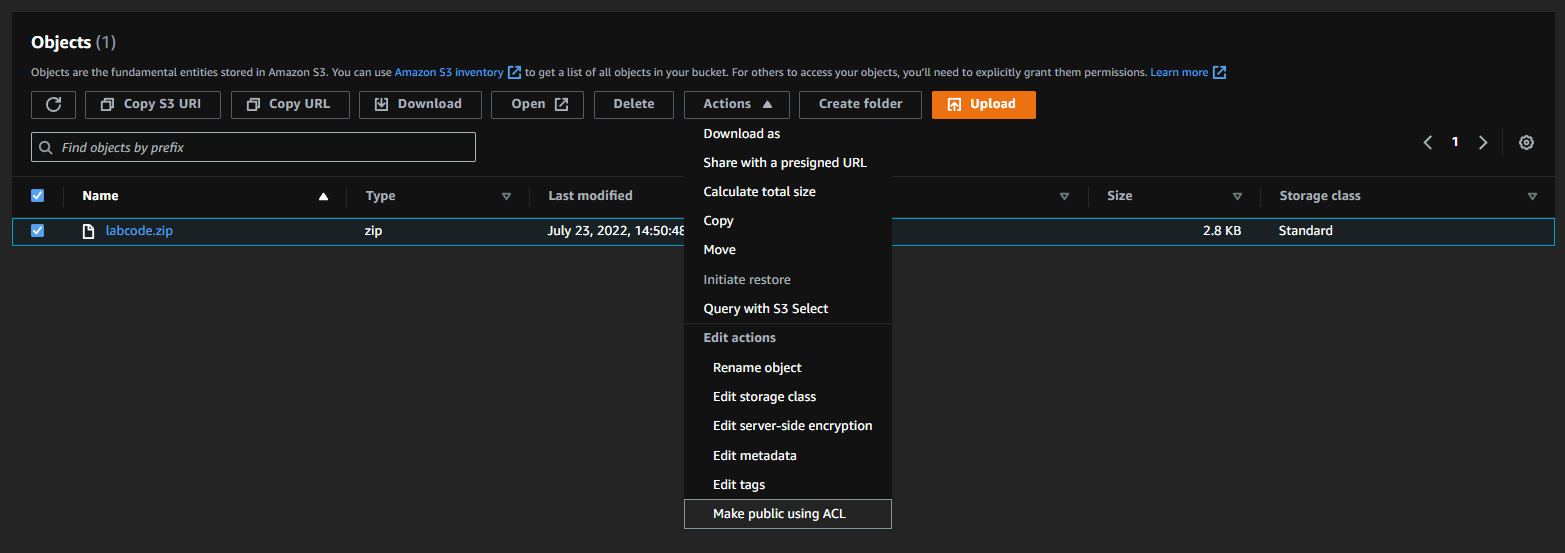
Description automatically generated

1. Scroll Down and Create the bucket.A screenshot of a computer

   Description automatically generated with medium confidence
2. Upload the zip file of your static webapp into the bucket. Make sure you do not upload the file in any other format.



1. Under the Bucket, select the object, chose actions, and make object public using ACL.



1. Review public access of the bucket.



1. Your Bucket has been created successfully.

***Launching an instance:***

1. In the search bar of the AWS Management Console, search for EC2 and chose the first option.

Graphical user interface, application

Description automatically generated

1. Click on Launch Instance:



1. Type The name of the instance and chose appropriate AMI. Here I have chosed amazon Linux.

A screenshot of a computer

Description automatically generated

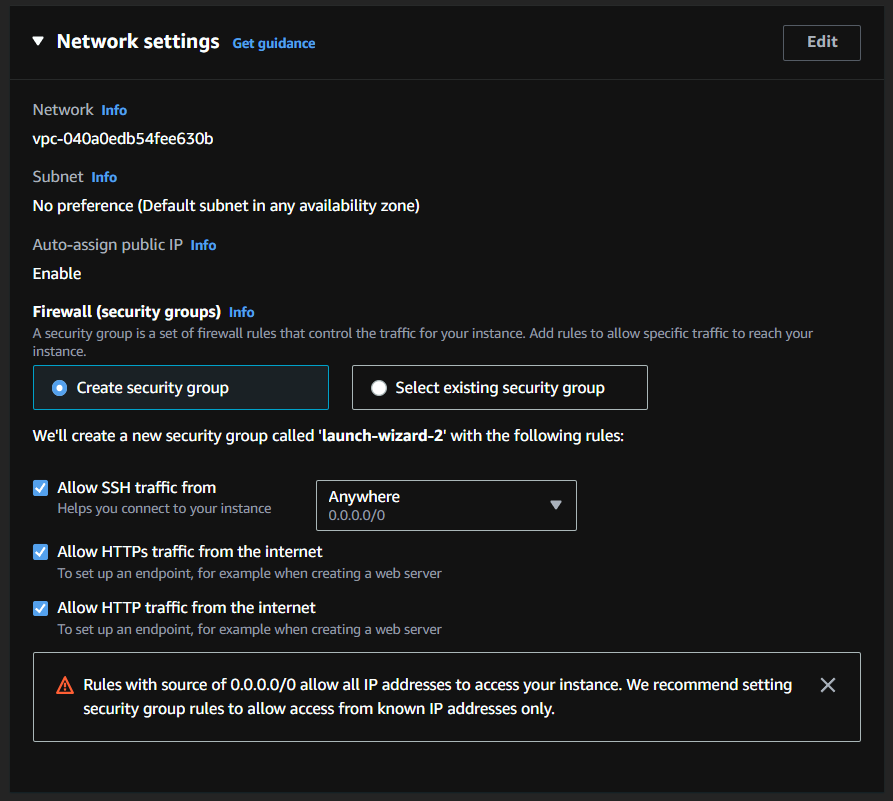
1. Choose the instance type:Graphical user interface, application

   Description automatically generated
2. Create a new Key Pair (or use an existing one) and save it as .pem at a known location on your PC.

Graphical user interface, text

Description automatically generated

1. Under network settings, either create a security group or chose an existing security group and check all the boxes as shown.



1. Launch The instance.

A screenshot of a computer

Description automatically generated with medium confidence

1. View all instances and in the console copy the Public IPV4 address

Text

Description automatically generated

1. Your instance has been launched successfully.

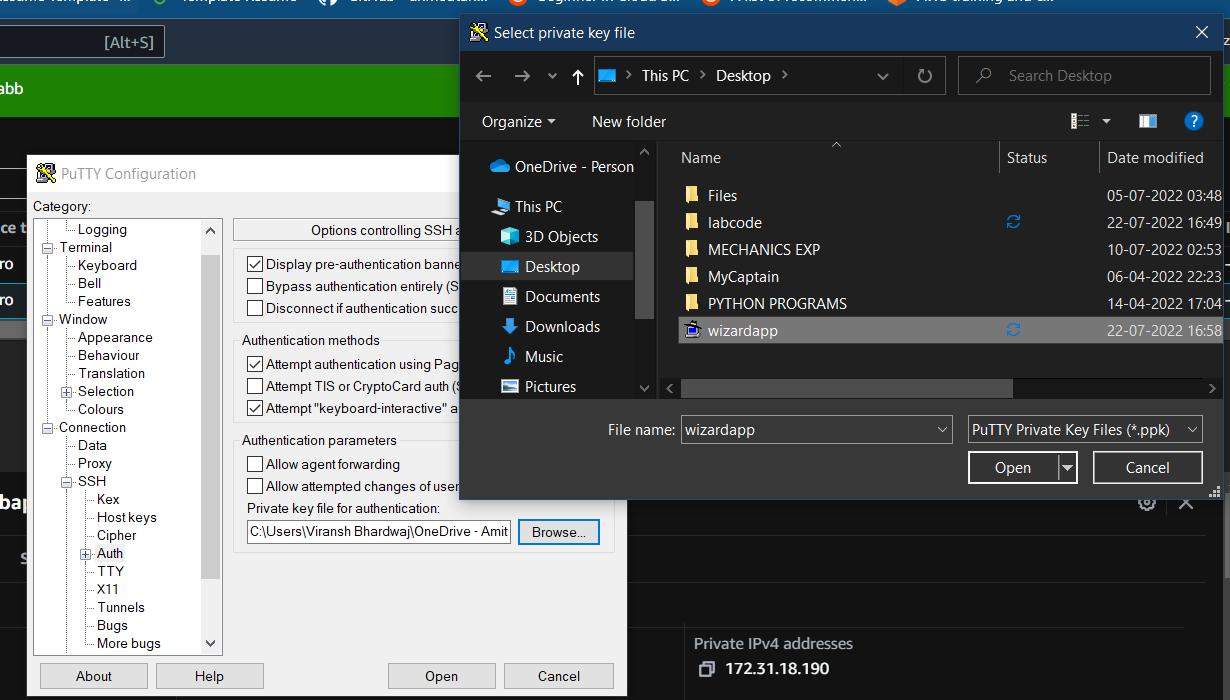
***Launching an Apache server using PuTTY:***

1. Open PuTTYgen from Start Menu and load the key pair file of the instance that you saved.

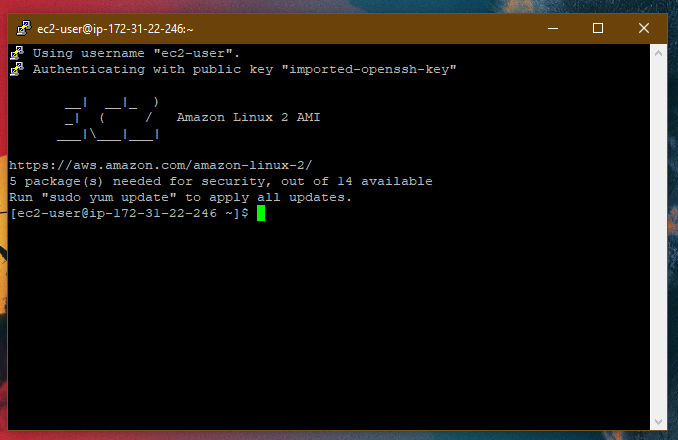
Graphical user interface

Description automatically generated

1. Save the private key at a known location.
2. Now Open PuTTY configuration console from start.
3. Type “ec2-user@<instance ipv4 address>” in the host name.
4. Under “Connection” tab, expand the “SSH” group and select “Auth”.



1. Browse for the key file that you saved from PuTTYgen and open a PuTTY Session.
2. A PuTTY command Prompt will open.



1. In the prompt type in the following commands. Wait for few seconds each command to take effect.
2. “sudo su” command to elevated privileges.



1. To update all installed packages. Wait for the process to complete
2. Install an Apache server using this command.



1. Follow the given commands in sequence:

A screenshot of a computer

Description automatically generated with medium confidence

1. 
2. 

The above commands allow us to check the directory we are in, followed by changing the directory to “/var/www/html”.

1. Return to the AWS Console of S3 and open the object. In the window, copy the URL of the object.

A screenshot of a computer

Description automatically generated with medium confidence

1. In the PuTTY command prompt, write the following command “wget” <space>

“<Copied URL>”



This command will import the zip file form S3 Bucket into the terminal.

1. Type “ls” command to see the imported zip file.

Text

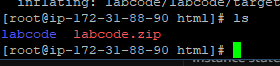
Description automatically generated

1. Extract the contents of the zip file using the given command.

“unzip labcode.zip”

(Note that the name of the file is written correctly)

1. Type “ls” command to see the extracted file.



1. Now use move command to move the contents of the file into the current directory (/var/www/html)

mv labcode/\* .

[Review the dot at the end of the command]

1. Type “ls” to se the contents again.
2. Type “pwd” to check the directory. You should be able to see “/var/www/html”
3. Now write the following command:

start httpd service

1. The Apache server will be launched.
2. Now copy the IPV4 Public Address of the instance and paste in your browser new tab.



1. You will now be able to see your static webpage hosted successfully onto the Apache server on AWS.

A picture containing graphical user interface

Description automatically generated

***Brief:***

Launching an instance

Creating an S3 Bucket.

Uploading the zip file of the webpage

Launching a PuTTY session. And installing an Apache server to host the webpage.

Starting the https service.