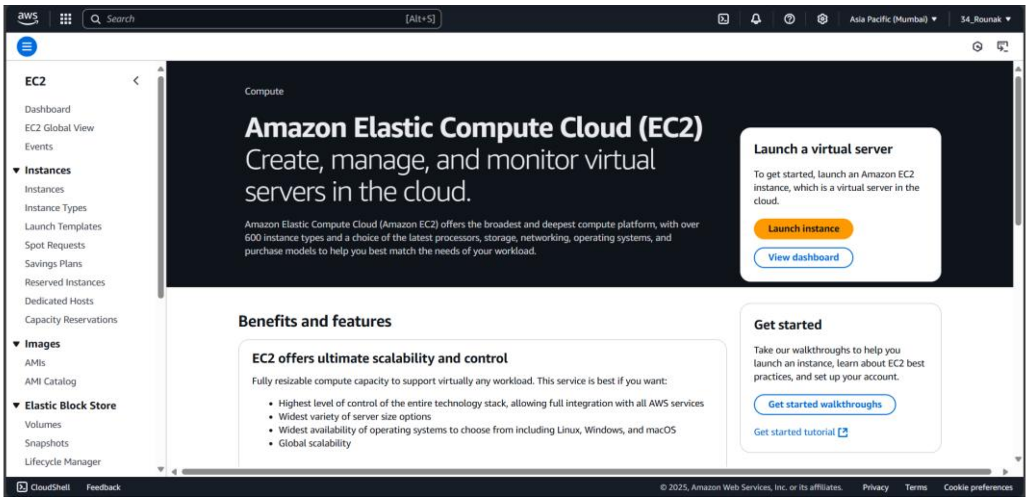
Assignment number: 12

Problem definition: Deploy and run the project in AWS without using the port.

Step 1: Sign in to your AWS account as the root user and into GitHub.

Step 2: Log in to the AWS Management Console, use the search bar to search for "EC2," and click on the EC2 service. Click on "Launch instance".



Amazon Elastic Compute Cloud (EC2)

Create, manage, and monitor virtual servers in the cloud.

Amazon Elastic Compute Cloud (Amazon EC2) offers the broadest and deepest compute platform, with over 600 instance types and a choice of the latest processors, storage, networking, operating systems, and purchase models to help you best match the needs of your workload.

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

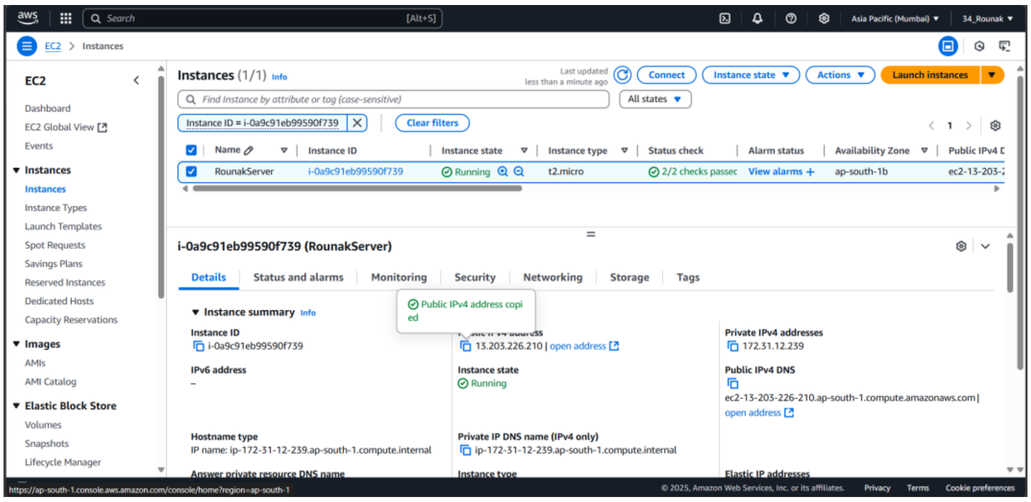
EC2 offers ultimate scalability and control

Fully resizable compute capacity to support virtually any workload. This service is best if you want:

- Highest level of control of the entire technology stack, allowing full integration with all AWS services
- Widest variety of server size options
- Widest availability of operating systems to choose from including Linux, Windows, and macOS
- Global scalability

Take our walkthroughs to help you launch an instance, learn about EC2 best practices, and set up your account.

Step 3: In the **Name and Tags** section, enter a descriptive name for your instance. Under **Application and OS Images**, choose **Ubuntu**. Ensure that the selected instance type meets your requirements. In the **Key pair (login)** section, select your pre-existing key pair. Next, in **Network settings**, under **Security group** choose **Select an existing security group** and attach the group that already defines your inbound/outbound rules. Click on "Launch instance". Once the instance is launched, you should see it listed under **Instances**. Click on the **Instance ID** to view more details. In the EC2 details, click on the **Public IPv4 address** and copy it.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D
RounakServer	i-0a9c91eb99590f739	Running	t2.micro	2/2 checks passec	View alarms +	ap-south-1b	ec2-13-203-2

i-0a9c91eb99590f739 (RounakServer)

Instance ID: i-0a9c91eb99590f739

Public IPv4 address: 13.203.226.210 | open address

Private IPv4 addresses: 172.31.12.239

IPv6 address: –

Instance state: Running

Public IPv4 DNS: ec2-13-203-226-210.ap-south-1.compute.amazonaws.com | open address

Hostname type: IP name: ip-172-31-12-239.ap-south-1.compute.internal

Private IP DNS name (IPv4 only): ip-172-31-12-239.ap-south-1.compute.internal

Answer private resource DNS name

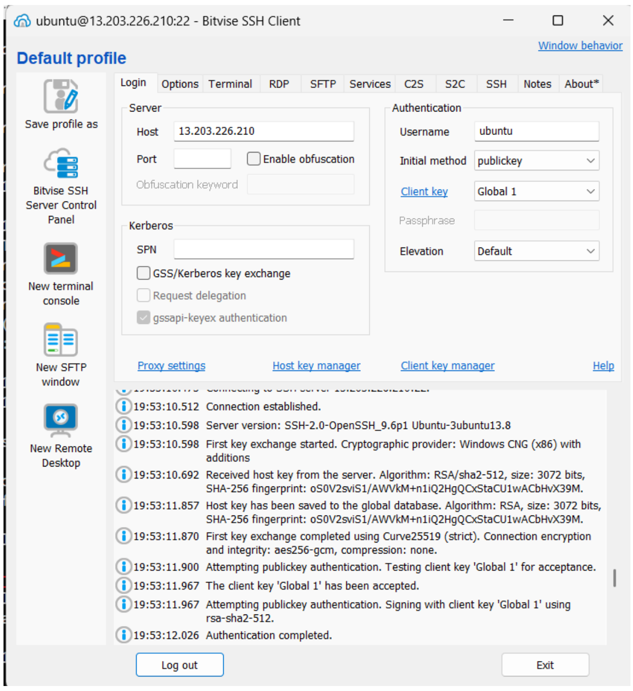
Instance type

Elastic IP addresses

https://ap-south-1.console.aws.amazon.com/console/home?region=ap-south-1

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 4: Open the **Bitvise SSH Client**. In the main window, paste the **Public IPv4 address** into the **Host** field. Set **Username** to “ubuntu”. From the **Client Key Manager** dropdown, select the key. Click **Log in** and then **Accept & Save** to establish the connection.



Server
Host: 13.203.226.210
Port:
Enable obfuscation
Obfuscation keyword:

Kerberos
SPN:
GSS/Kerberos key exchange
Request delegation
gssapi-keyex authentication

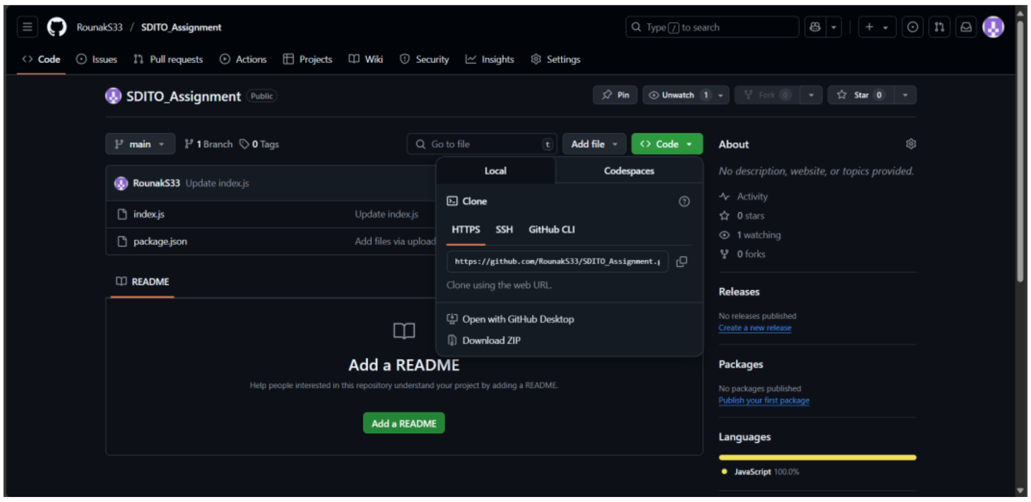
Authentication
Username: ubuntu
Initial method: publickey
Client key: Global 1
Passphrase:
Elevation: Default

```
19:53:10.512 Connection established.
19:53:10.598 Server version: SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.8
19:53:10.598 First key exchange started. Cryptographic provider: Windows CNG (x86) with
             additions
19:53:10.692 Received host key from the server. Algorithm: RSA/sha2-512, size: 3072 bits,
             SHA-256 fingerprint: oS0V2sviS1/AWVkM+n1iQ2HgQCxStaCU1wACbHvX39M.
19:53:11.857 Host key has been saved to the global database. Algorithm: RSA, size: 3072 bits,
             SHA-256 fingerprint: oS0V2sviS1/AWVkM+n1iQ2HgQCxStaCU1wACbHvX39M.
19:53:11.870 First key exchange completed using Curve25519 (strict). Connection encryption
             and integrity: aes256-gcm, compression: none.
19:53:11.900 Attempting publickey authentication. Testing client key 'Global 1' for acceptance.
19:53:11.967 The client key 'Global 1' has been accepted.
19:53:11.967 Attempting publickey authentication. Signing with client key 'Global 1' using
             rsa-sha2-512.
19:53:12.026 Authentication completed.
```

Step 5: In Bitvise SSH Client, click on “**New Terminal Console**”. Run the following commands:

1. Update the package lists and upgrade existing packages on your EC2 instance: “sudo apt-get update” and “sudo apt-get upgrade”
2. Install the Nginx web server: “sudo apt-get install -y nginx”
3. “nginx -v”
4. Download the NodeJS installation files and dependencies: “curl -Sl https://deb.nodesource.com/setup_18.x | sudo -E bash -“ and “sudo apt install -y nodejs’
5. “node -v”

Step 6: On your GitHub repository page, click the green "Code" button. Copy the HTTPS URL provided.



SDITO_Assignment Public

RounakS33 Update index.js

index.js	Update index.js
package.json	Add files via upload

https://github.com/RounakS33/SDITO_Assignment.g

Clone using the web URL.

Add a README

Help people interested in this repository understand your project by adding a README.

No description, website, or topics provided.

No releases published
Create a new release

No packages published
Publish your first package

Step 7: Run the following commands in the terminal:

1. Clone your project: “git clone https://github.com/RounakS33/SDITO_Assignment.git”
2. “cd SDITO_Assignment”
3. “npm install”

4. Start your application: "node index.js"

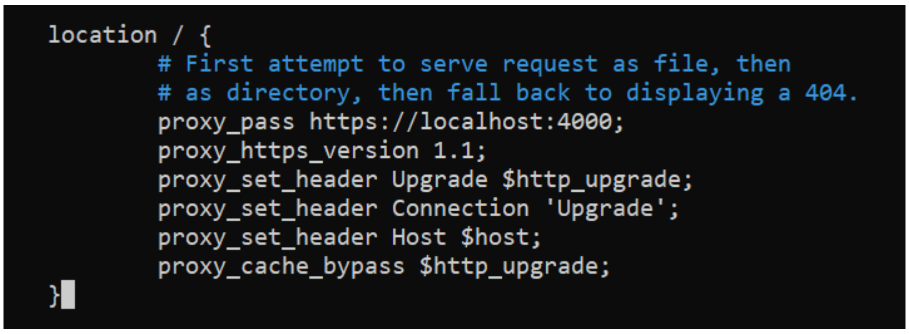
Step 8: Check the port no. specified in the program. It is specified in the app.listen() method as the first parameter (in our case, it is 4000). By default, Nginx listens on port 80 for HTTP traffic. Instead of exposing your Node.js port 4000 directly in the security group, you should open port 80 so external clients talk to Nginx, which then proxies internally to localhost:4000.

Step 9: Run the following commands in the terminal:

1. "cd /etc/nginx/sites-available/"
2. "sudo chmod 777 default"
3. "sudo nano default"

Step 10: Edit your site's server-block file (e.g. /etc/nginx/sites-available/default), find the location / { … } section, delete the try_files lines, and insert the proxy directives shown below.

proxy_pass http://localhost:4000;
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection 'Upgrade';
proxy_set_header Host $host;
proxy_cache_bypass $http_upgrade;



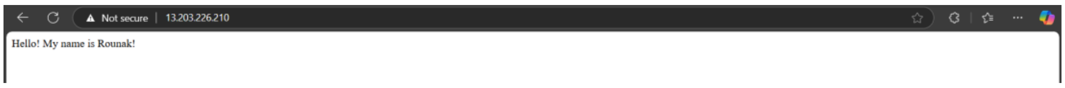
```
location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        proxy_pass https://localhost:4000;
        proxy_https_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'Upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
}
```

Save and exit the editor.

Step 11: Run the following commands in the terminal:

1. "sudo systemctl restart nginx"
2. "cd /home/ubuntu/SDITO_Assignment"
3. Start your application: "node index.js"

Open a new tab in your web browser. In the address bar, enter the **Public IPv4 address** of your EC2 instance. Press Enter. You should now see your deployed project running in the browser.



Not secure | 13.203.226.210

Hello! My name is Rounak!

<u>Conclusion:</u>

In this assignment, we successfully deployed a Node.js application on an AWS EC2 instance, utilizing Nginx as a reverse proxy to serve the app over the standard HTTP port (80). By configuring Nginx to forward incoming requests to the Node.js server running on port 4000, we ensured seamless access without exposing non-standard ports. This setup not only enhances security but also aligns with best practices for web application deployment. The process provided valuable insights into cloud-based deployment, server configuration, and efficient application hosting using AWS services.