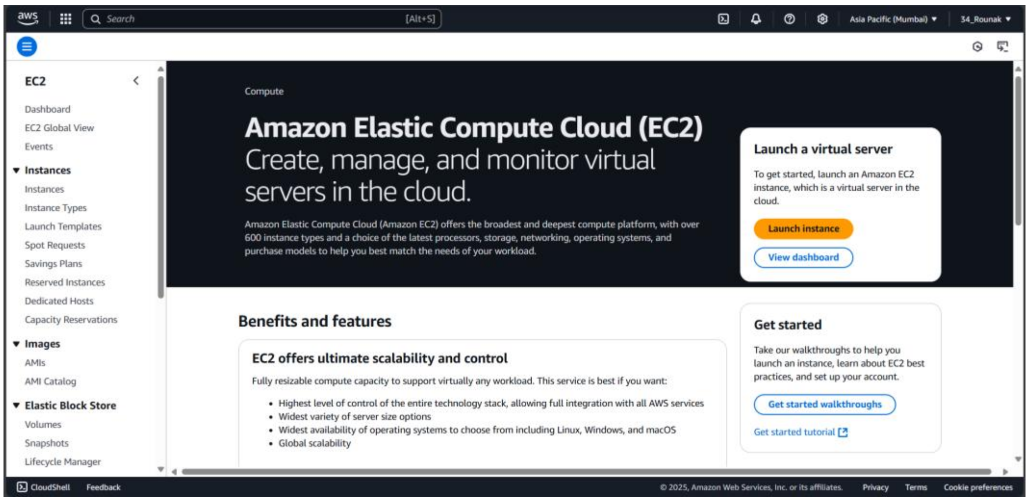
<u>Assignment number:</u> 9

<u>Problem definition:</u> Deploy a project from GitHub to EC2.

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”.


Compute

Amazon Elastic Compute Cloud (EC2)
Create, manage, and monitor virtual servers in the cloud.

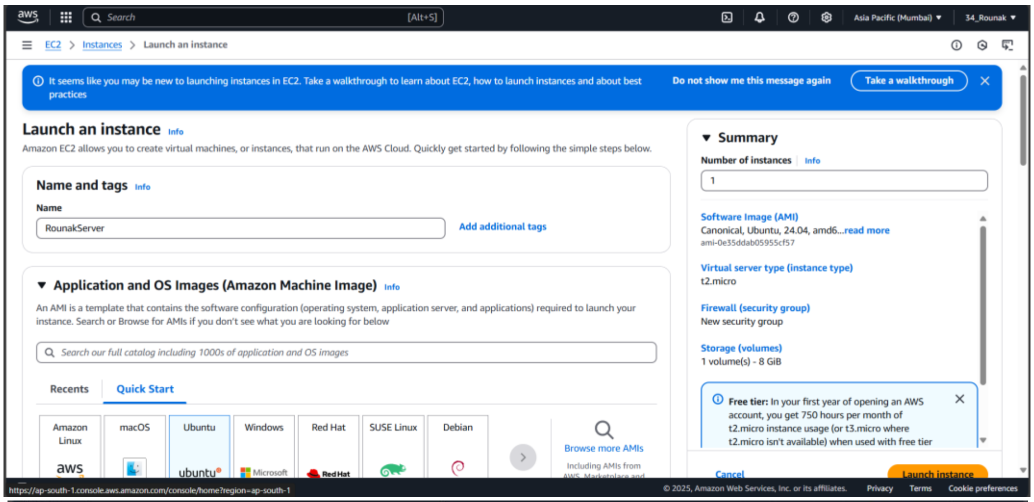
Amazon Elastic Compute Cloud (Amazon EC2) offers the broadest and deepest compute platform, with over 600 instance types and a choice of the latest processors, storage, networking, operating systems, and purchase models to help you best match the needs of your workload.

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

Benefits and features
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

Get started
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.


It seems like you may be new to launching instances in EC2. Take a walkthrough to learn about EC2, how to launch instances and about best practices

Launch an instance
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

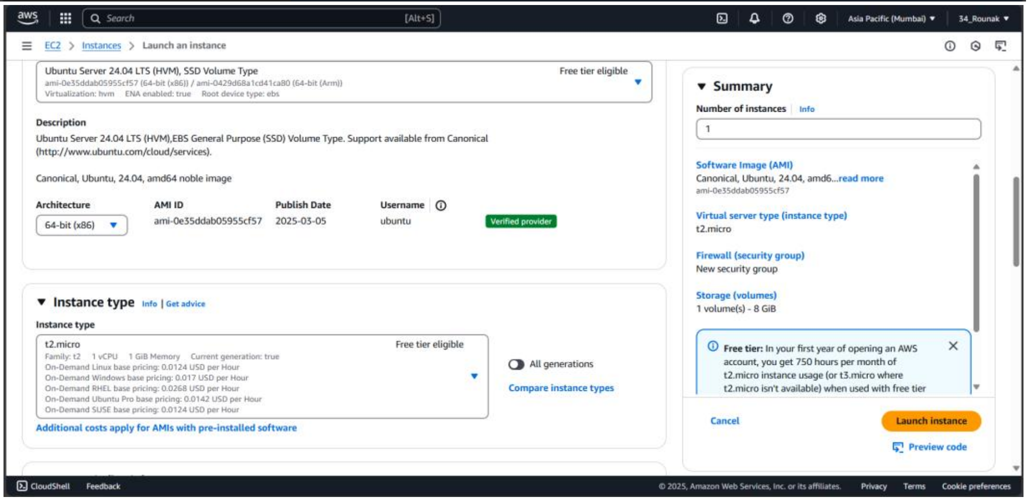
Name and tags
Name: RounakServer

Application and OS Images (Amazon Machine Image)
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Summary
Number of instances: 1
Software Image (AMI): Canonical, Ubuntu, 24.04, amd6...read more
ami-0e35ddab05955cf57
Virtual server type (instance type): t2.micro
Firewall (security group): New security group
Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier

EC2 > Instances > Launch an instance
Do not show me this message again
Take a walkthrough
Add additional tags
Search our full catalog including 1000s of application and OS images
Recents
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE Linux
Debian
Browse more AMIs
Including AMIs from AWS Marketplace and
Cancel
Launch instance
https://ap-south-1.console.aws.amazon.com/console/home?region=ap-south-1


Ubuntu Server 24.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-0e35ddab05955cf57 (64-bit (x86)) / ami-0429d68a1cd41ca80 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description
Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Canonical, Ubuntu, 24.04, amd64 noble image

Architecture	AMI ID	Publish Date	Username
64-bit (x86)	ami-0e35ddab05955cf57	2025-03-05	ubuntu

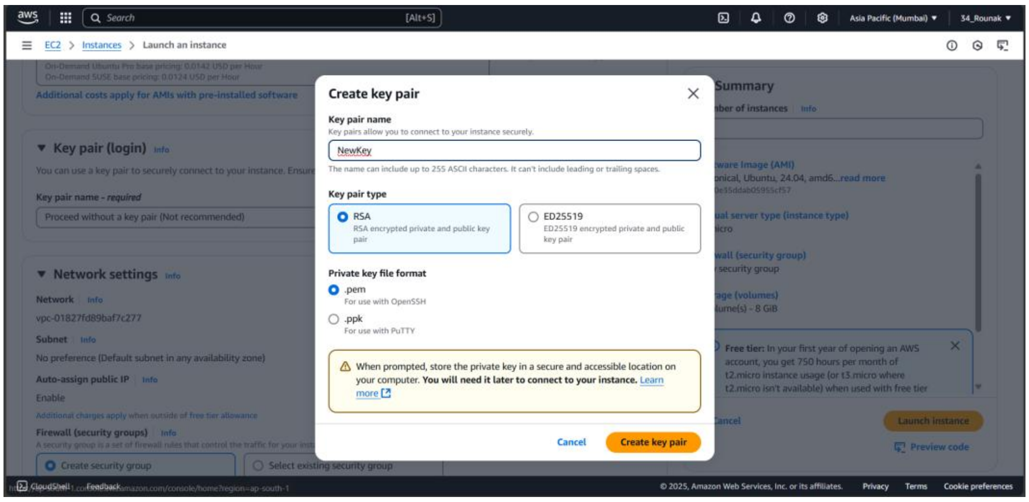
Verified provider

Instance type
t2.micro — Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour
On-Demand Windows base pricing: 0.017 USD per Hour
On-Demand RHEL base pricing: 0.0268 USD per Hour
On-Demand Ubuntu Pro base pricing: 0.0142 USD per Hour
On-Demand SUSE base pricing: 0.0124 USD per Hour
Additional costs apply for AMIs with pre-installed software

Summary
Number of instances: 1
Software Image (AMI): Canonical, Ubuntu, 24.04, amd6...read more
ami-0e35ddab05955cf57
Virtual server type (instance type): t2.micro
Firewall (security group): New security group
Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier

Step 4: In the “Key pair (login)” section, click on **Create new key pair**. Give a name, ensure that the key pair type is set to **RSA** and select **.pem** as the private key file format. Click on **“Create key pair”**.



Create key pair

Key pair name
Key pairs allow you to connect to your instance securely.
NewKey
The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type
[✓ RSA] RSA encrypted private and public key pair
ED25519 ED25519 encrypted private and public key pair

Private key file format
[✓ .pem] For use with OpenSSH
.ppk For use with PuTTY

When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn more

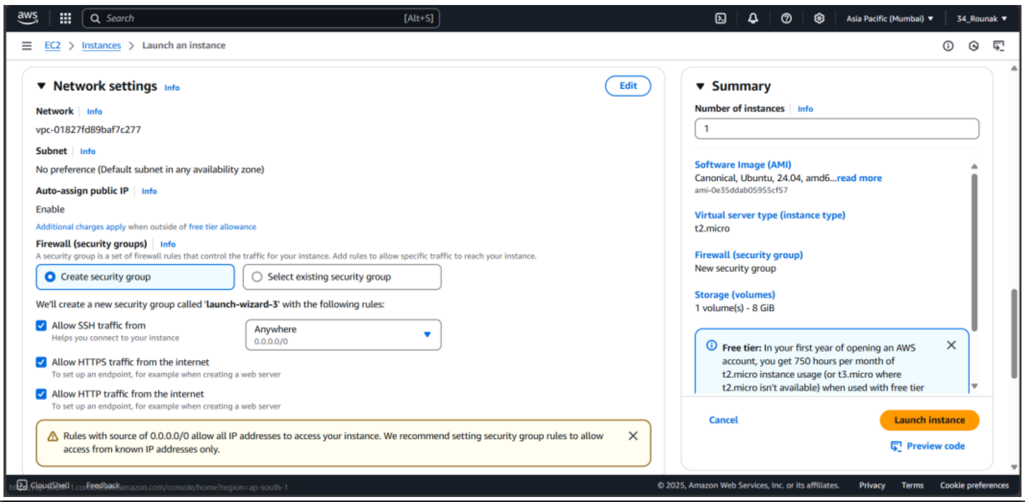
Cancel | Create key pair

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

Step 5: In the “Network Settings” section, tick all the necessary checkboxes to allow essential traffic:

- **SSH (port 22):** For connecting to the instance.
- **HTTP (port 80):** For serving your website.
- **HTTPS (port 443):** For secure browsing.

Click on “Launch instance”.



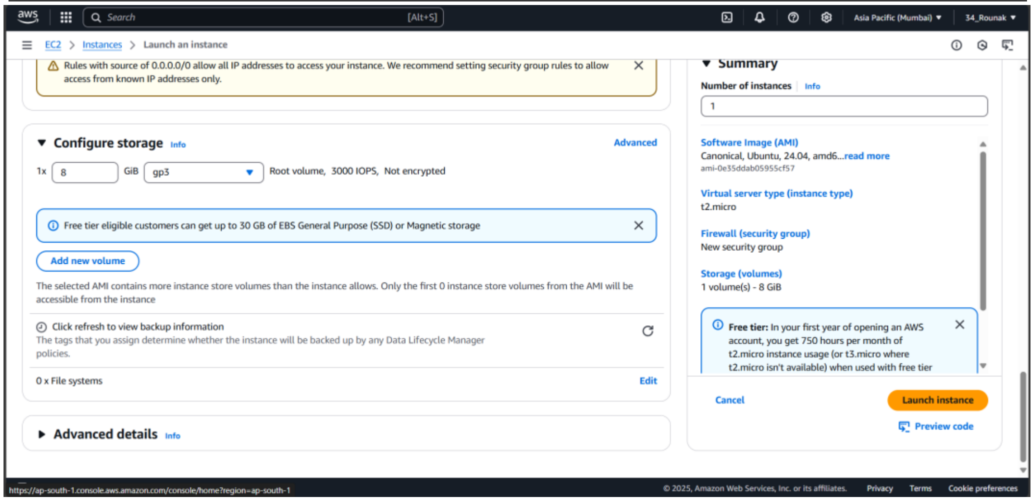
Network settings

Network: vpc-01827fd89baf7c277
Subnet: No preference (Default subnet in any availability zone)
Auto-assign public IP: Enable
Additional charges apply when outside of free tier allowance
Firewall (security groups)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
[✓ Create security group] Select existing security group
We'll create a new security group called '**launch-wizard-3**' with the following rules:
- [x] Allow SSH traffic from — Anywhere 0.0.0.0/0
  Helps you connect to your instance
- [x] Allow HTTPS traffic from the internet
  To set up an endpoint, for example when creating a web server
- [x] Allow HTTP traffic from the internet
  To set up an endpoint, for example when creating a web server

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Summary
Number of instances: 1
Software Image (AMI): Canonical, Ubuntu, 24.04, amd6...read more
ami-0e35ddab05955cf57
Virtual server type (instance type): t2.micro
Firewall (security group): New security group
Storage (volumes): 1 volume(s) - 8 GiB
Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier

Cancel | Launch instance | Preview code



Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Configure storage
1x 8 GiB gp3 Root volume, 3000 IOPS, Not encrypted
Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage
The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance
Click refresh to view backup information
The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.
0 x File systems — Edit

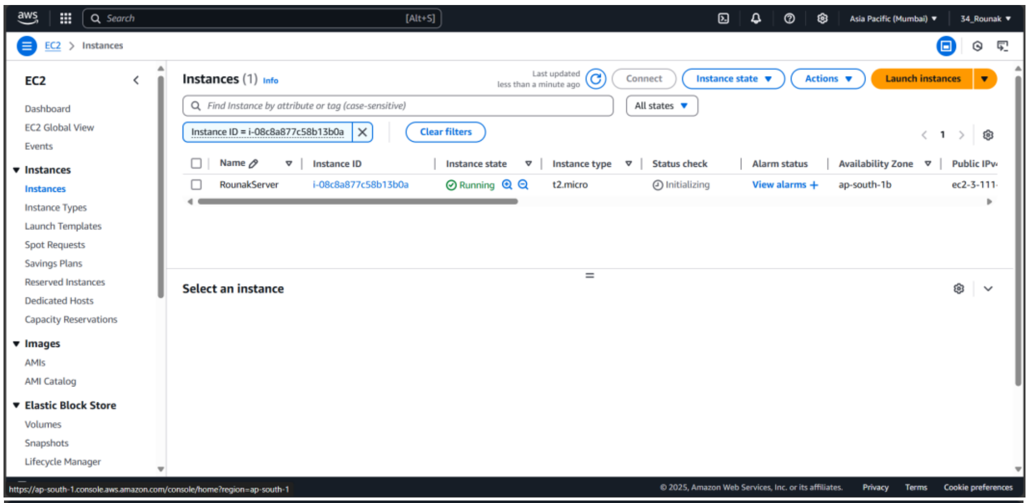
Advanced details

Summary
Number of instances: 1
Software Image (AMI): Canonical, Ubuntu, 24.04, amd6...read more
ami-0e35ddab05955cf57
Virtual server type (instance type): t2.micro
Firewall (security group): New security group
Storage (volumes): 1 volume(s) - 8 GiB
Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier

Cancel | Launch instance | Preview code

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 6: 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.



Instances (1) Info

Last updated less than a minute ago

Find Instance by attribute or tag (case-sensitive)

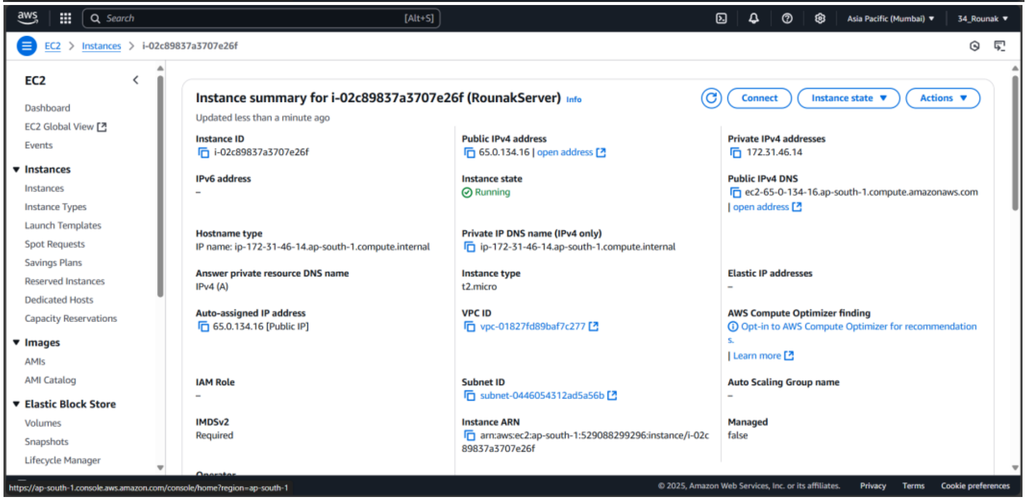
Instance ID = i-08c8a877c58b13b0a

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv
☐	RounakServer	i-08c8a877c58b13b0a	Running	t2.micro	Initializing	View alarms +	ap-south-1b	ec2-3-111

Select an instance

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



Instance summary for i-02c89837a3707e26f (RounakServer) Info

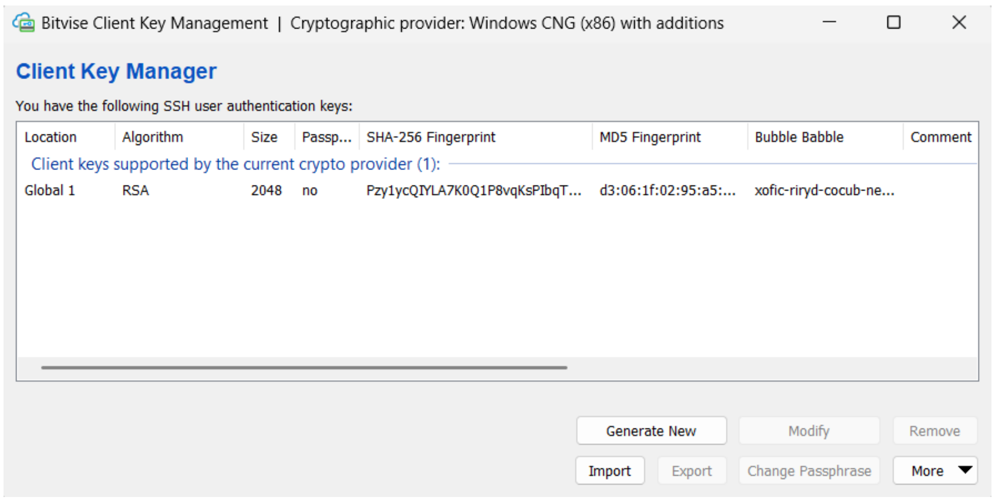
Updated less than a minute ago

Instance ID i-02c89837a3707e26f	**Public IPv4 address** 65.0.134.16 | open address	**Private IPv4 addresses** 172.31.46.14
IPv6 address –	**Instance state** Running	**Public IPv4 DNS** ec2-65-0-134-16.ap-south-1.compute.amazonaws.com | open address
Hostname type IP name: ip-172-31-46-14.ap-south-1.compute.internal	**Private IP DNS name (IPv4 only)** ip-172-31-46-14.ap-south-1.compute.internal	
Answer private resource DNS name IPv4 (A)	**Instance type** t2.micro	**Elastic IP addresses** –
Auto-assigned IP address 65.0.134.16 [Public IP]	**VPC ID** vpc-01827fd89baf7c277	**AWS Compute Optimizer finding** Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role –	**Subnet ID** subnet-0446054312ad5a56b	**Auto Scaling Group name** –
IMDSv2 Required	**Instance ARN** arn:aws:ec2:ap-south-1:529088299296:instance/i-02c89837a3707e26f	**Managed** false

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 7: Open the **Bitvise SSH Client**. In the main window, paste the **Public IPv4 address** into the **Host** field. Click on "**Client Key Manager"**, click "**Import"** (if any key is already present, remove it), select the previously downloaded key file (e.g., NewKey.pem), and click "**Import"** again to add the key. Set **Username** to "ubuntu". From the **Client Key Manager** dropdown, select the imported key. Click "**Log in"** and then "**Accept & Save"** to establish the connection.

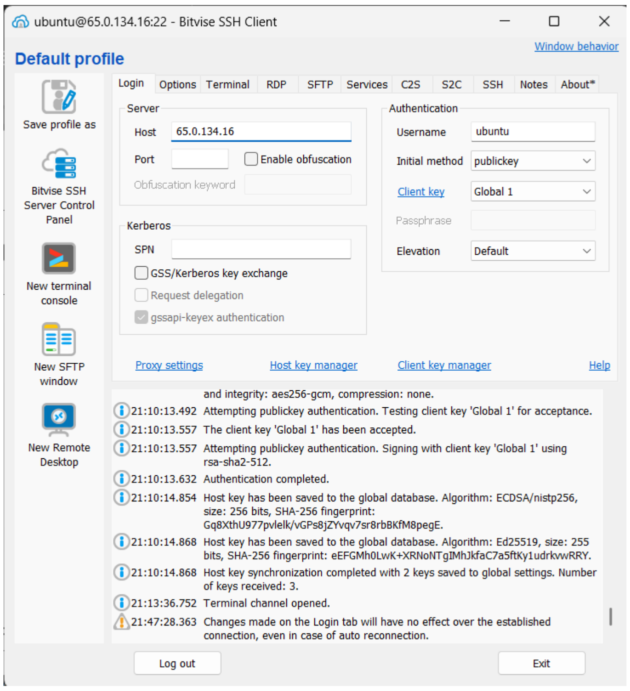


Bitvise Client Key Management | Cryptographic provider: Windows CNG (x86) with additions

Client Key Manager

You have the following SSH user authentication keys:

Location	Algorithm	Size	Passp...	SHA-256 Fingerprint	MD5 Fingerprint	Bubble Babble	Comment
Client keys supported by the current crypto provider (1):							
Global 1	RSA	2048	no	Pzy1ycQIYLA7K0Q1P8vqKsPIbqT...	d3:06:1f:02:95:a5:...	xofic-riryd-cocub-ne...	



ubuntu@65.0.134.16:22 - Bitvise SSH Client

Default profile

Server: Host 65.0.134.16

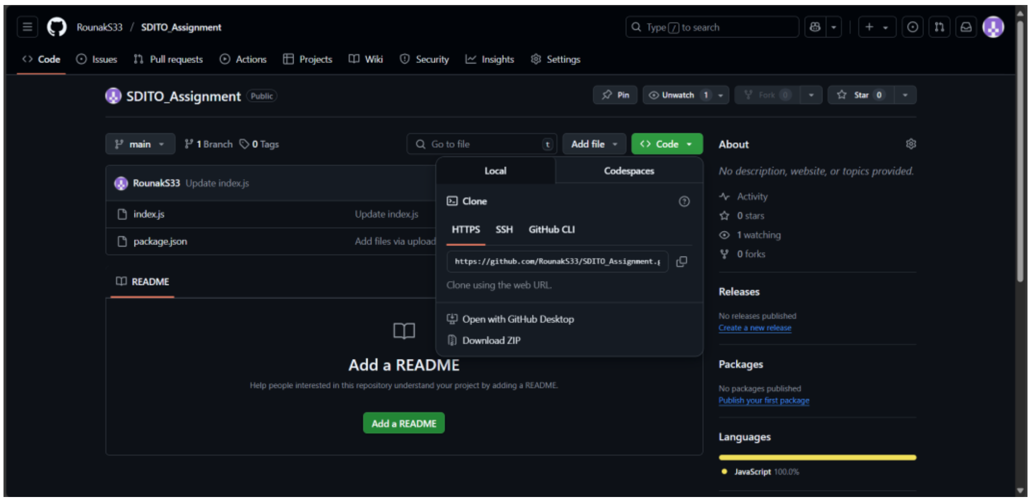
Authentication: Username ubuntu; Initial method publickey; Client key Global 1; Elevation Default

```
                  and integrity: aes256-gcm, compression: none.
21:10:13.492 Attempting publickey authentication. Testing client key 'Global 1' for acceptance.
21:10:13.557 The client key 'Global 1' has been accepted.
21:10:13.557 Attempting publickey authentication. Signing with client key 'Global 1' using
             rsa-sha2-512.
21:10:13.632 Authentication completed.
21:10:14.854 Host key has been saved to the global database. Algorithm: ECDSA/nistp256,
             size: 256 bits, SHA-256 fingerprint:
             Gq8XthU977pvlelk/vGPs8jZYvqv7sr8rbBKfM8pegE.
21:10:14.868 Host key has been saved to the global database. Algorithm: Ed25519, size: 255
             bits, SHA-256 fingerprint: eEFGMh0LwK+XRNoNTgIMhJkfaC7a5ftKy1udrkvwRRY.
21:10:14.868 Host key synchronization completed with 2 keys saved to global settings. Number
             of keys received: 3.
21:13:36.752 Terminal channel opened.
21:47:28.363 Changes made on the Login tab will have no effect over the established
             connection, even in case of auto reconnection.
```

Step 8: 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 9: On your GitHub repository page, click the green "Code" button. Copy the HTTPS URL provided.



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

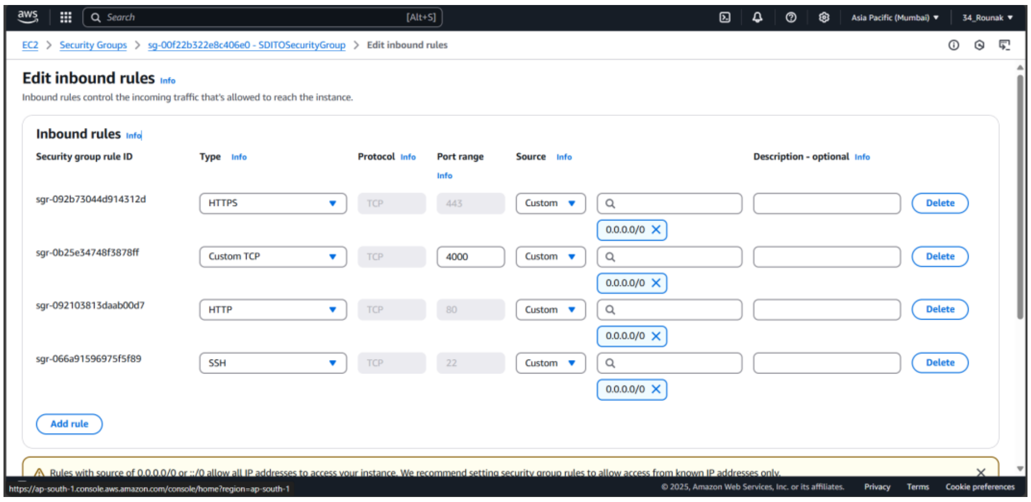
Clone using the web URL.

Step 10: 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 11: Check the port no. specified in the program. It is specified in the app.listen() method as the first parameter. We must add this port number to our EC2 instance security group rule to be able to access the website from anywhere (in our case, it is 4000).

Step 12: Go back to your AWS EC2 Management Console in the browser and navigate to **"Instances"** and select the running EC2 instance you are using. Click on the **"Security"** tab. Click the **security group link**. Go to **"Inbound rules"**. Click **"Add rule"**. Set **Type** to **"Custom TCP"**, **Port range** to **"4000"**, **Source** to **"Anywhere" (0.0.0.0/0)**. Click on **"Save rules"**.


EC2 > Security Groups > sg-00f22b322e8c406e0 - SDITOSecurityGroup > Edit inbound rules

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

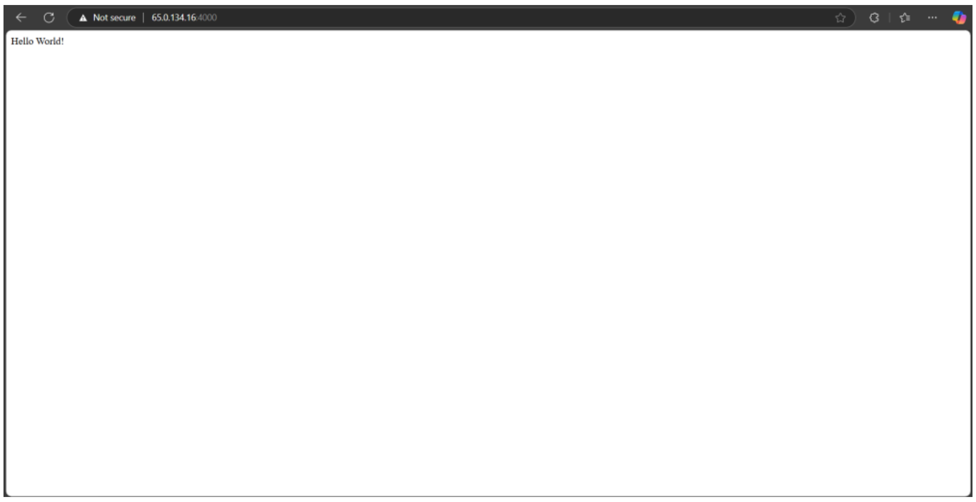
Inbound rules

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-092b73044d914312d	HTTPS	TCP	443	Custom	0.0.0.0/0	
sgr-0b25e34748f3878ff	Custom TCP	TCP	4000	Custom	0.0.0.0/0	
sgr-092103813daab00d7	HTTP	TCP	80	Custom	0.0.0.0/0	
sgr-066a91596975f5f89	SSH	TCP	22	Custom	0.0.0.0/0	

Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Search [Alt+S]
Asia Pacific (Mumbai)
34_Rounak
Info
Delete
Add rule
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 13: Open a new tab in your web browser. In the address bar, enter the **Public IPv4 address** of your EC2 instance, followed by a colon (:) and the port number your application is listening on (in our case, http://65.0.134.16:4000). Press Enter. You should now see your deployed project running in the browser.



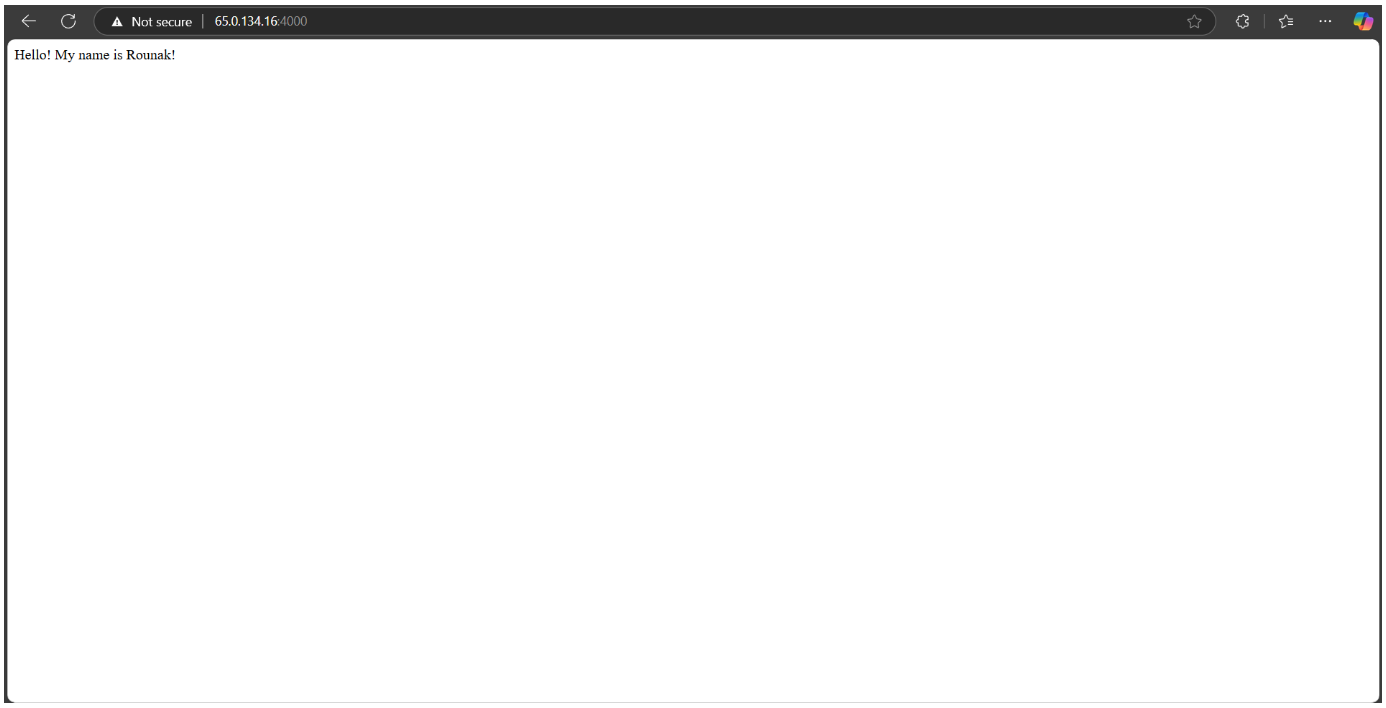
Hello World!

Step 14: To make changes to your project, go to your GitHub repository in the browser. Open the file you want to modify (e.g., index.js). Click the "Edit" (pencil) icon. Make your changes to the code. Scroll down and click the **"Commit changes"** button. Your changes are now in your GitHub repository. However, our changes have not been reflected in our Remote Server. For that we must 'pull' the new updated files into the repository directory in our Remote Server.

Step 15: We must Close our already running Server. For this, go to the Terminal which we have been working with. Then press <CTRL + C> shortcut to close the server. Our server has been stopped.

Step 16: Run the following commands in the terminal:

1. "git pull"
2. Restart your application: "node index.js"



Hello! My name is Rounak!

<u>Conclusion:</u>

In conclusion, this assignment successfully guided us through the process of deploying a web project hosted on GitHub to a running EC2 instance on AWS. We established a connection to the EC2 instance via SSH, installed the necessary runtime environment (Nginx and Node.js), cloned the project repository from GitHub, installed dependencies, and configured the EC2 instance's security group to allow external access to our application on the specified port (4000). Finally, we demonstrated how to update the deployed project by modifying the code in the GitHub repository and pulling those changes to the EC2 instance, followed by a server restart. This exercise provides a foundational understanding of the steps involved in deploying and managing web applications on the AWS cloud using EC2 and GitHub for version control.