Lecture 19 - Deployment

What is Deployment?

Deployment is the process of making a software application or system available for use in a specific environment. It involves the transfer of code, configuration, and other necessary resources from a development environment to a production environment, ensuring that the application runs efficiently and reliably for end-users.

Need for Deployment

- **Release of New Features:** Deployment allows the integration of new features and improvements developed by the team, making them accessible to users.
- **Bug Fixes and Updates:** Deployments are essential for delivering bug fixes, security patches, and updates to improve the functionality and security of the application.
- **User Accessibility:** Deployment ensures that users have access to the latest version of the application, providing a seamless and up-to-date experience.
- **Performance Optimization:** It allows for the implementation of performance optimizations and enhancements, ensuring the application runs efficiently in a production environment.
- **Meeting Business Objectives:** Deployment is crucial for aligning the software with business objectives and maintaining a competitive edge in the market.

Environments in Deployment

Development Environment:

The development environment is where developers write, test, and debug code. It's an isolated space for experimentation and initial testing of new features.

Testing Environment:

The testing environment is used for more comprehensive testing, including unit tests, integration tests, and system tests. It mimics the production environment to identify and address issues before deployment.

Production Environment:

The production environment is the live environment where the application is accessed by end-users. It requires careful deployment to ensure stability, performance, and security.

Staging Environment:

Some projects have a staging environment that closely mirrors the production environment. It serves as a final testing ground before deploying to the live production environment.

QA (Quality Assurance) Environment:

The QA environment is dedicated to quality assurance activities, providing a controlled space to validate that the application meets specified requirements and standards.

Integration Environment:

The integration environment is where individual components or modules are combined to test their interactions and interoperability.

User Acceptance Testing (UAT) Environment:

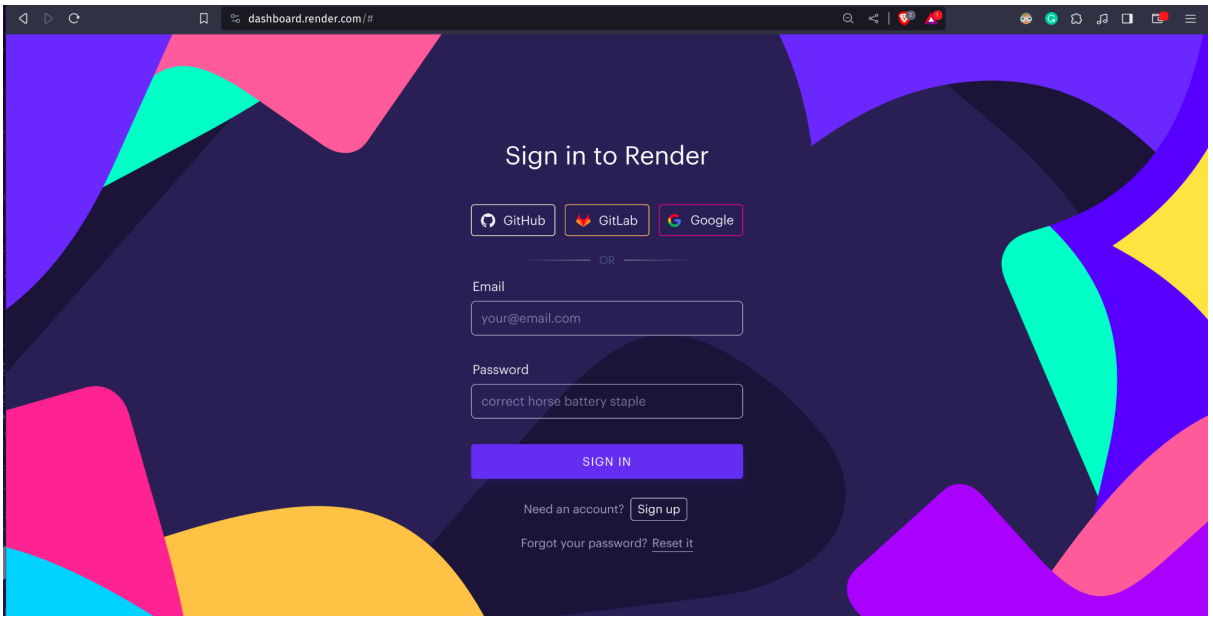
This environment is specifically for user acceptance testing, allowing stakeholders to validate that the application meets their expectations before deployment to production.

Deployment on Render

Render is a cloud platform that provides hosting services for web applications, databases, and other services. Keep in mind that there might have been changes or updates to Render since then, so it's advisable to refer to the official documentation for the most up-to-date information. Here are general notes on deploying on Render:

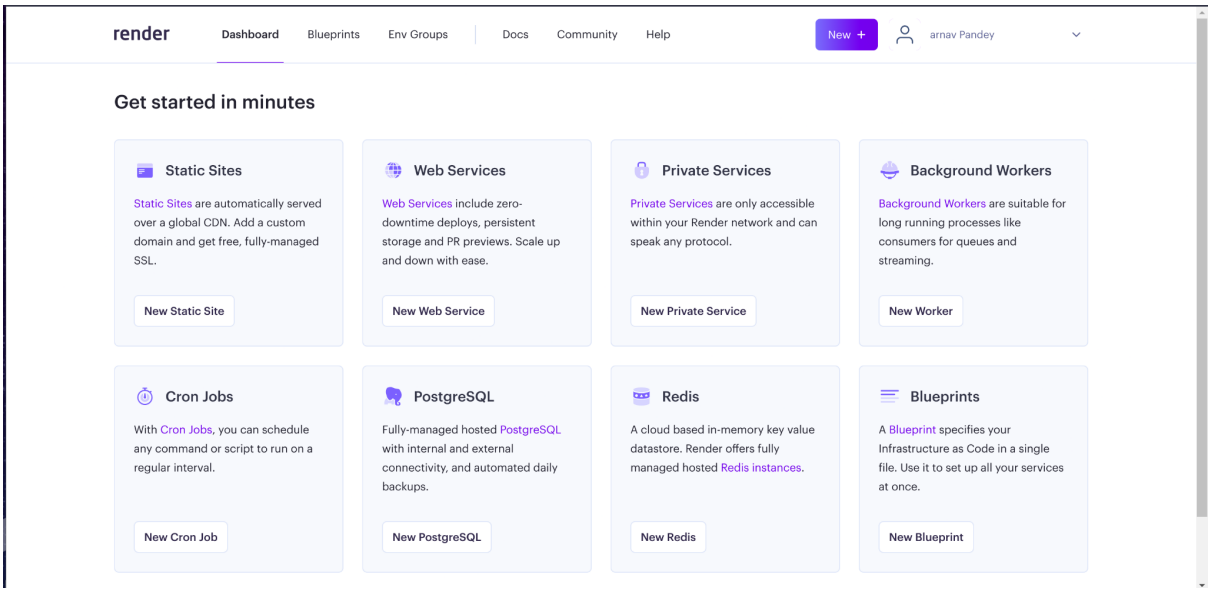
Create an Account:

Start by creating an account on the Render platform if you haven't done so already.



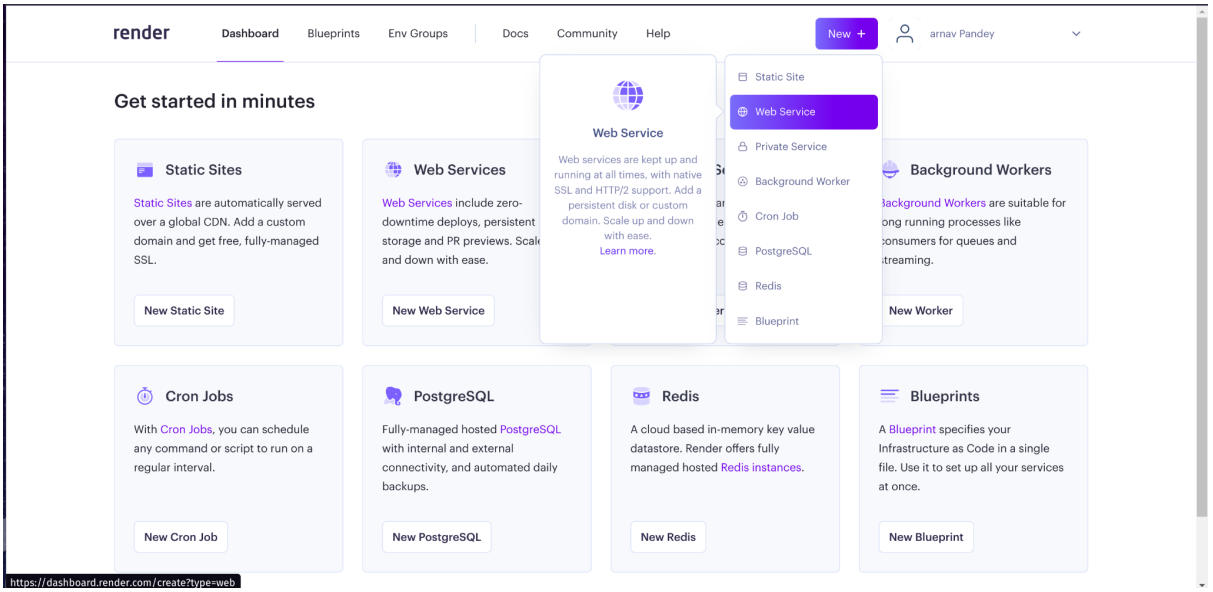
Alternatively, registration is possible through the utilization of Google or GitHub accounts.

Dashboard Overview:



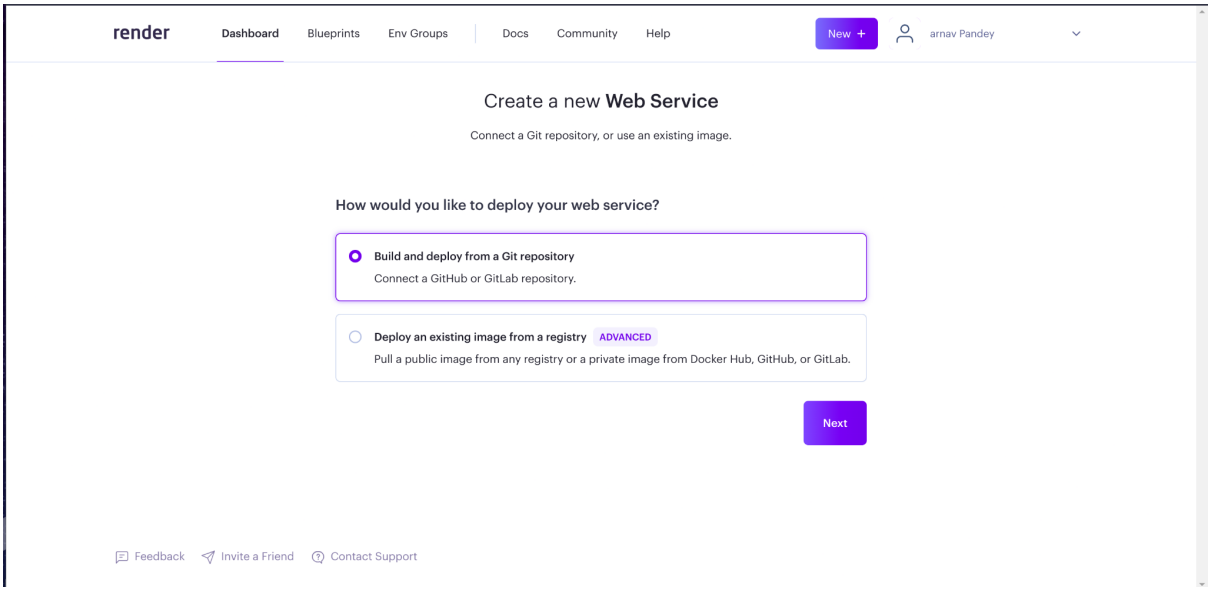
Familiarize yourself with the Render dashboard. This is where you'll manage your services, deployments, and settings.

Create a New Service:

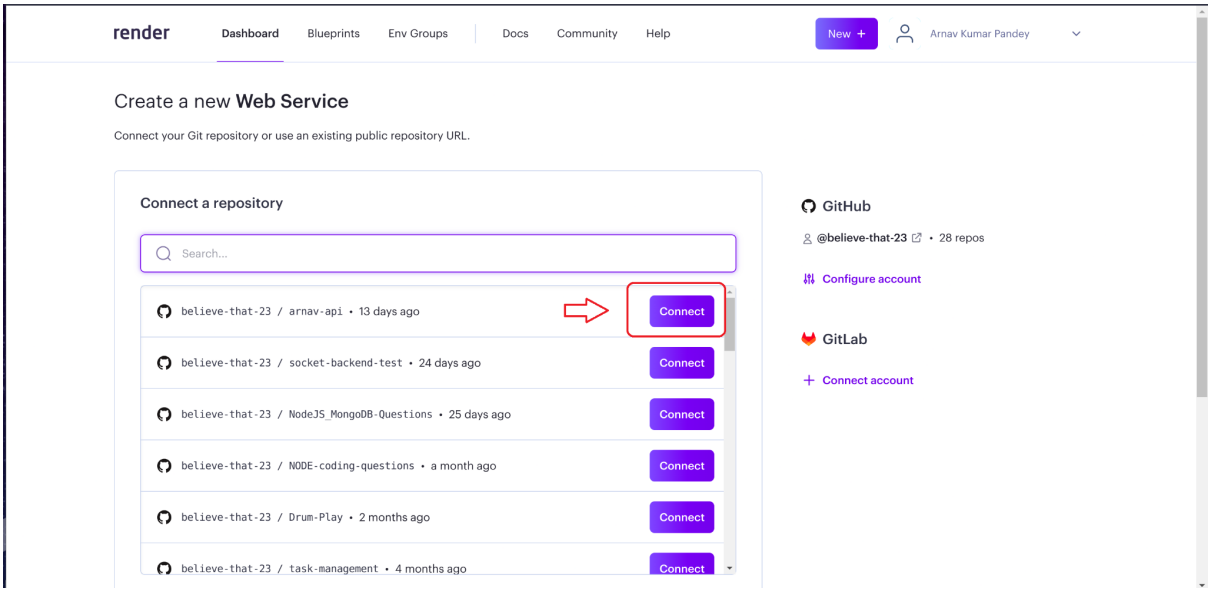


To deploy an application, you need to create a service on Render. This could be a web service, database, or other types of services depending on your application's requirements.

We intend to leverage the Web Services feature on the Render platform for hosting our project.

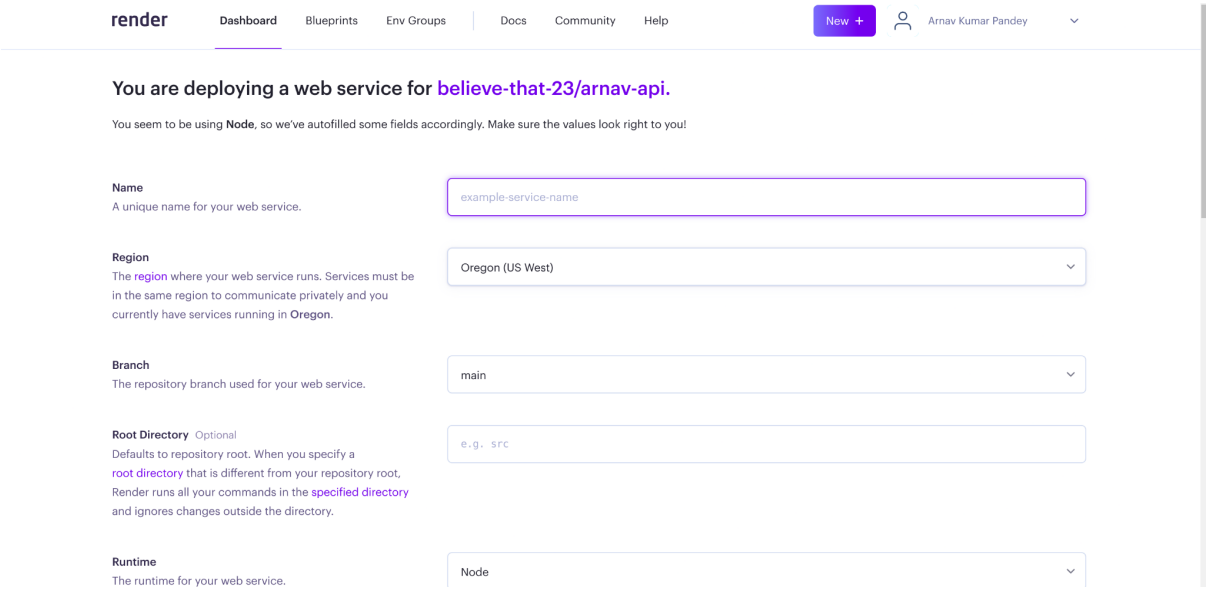


Now one can connect their GitHub account with the same to deploy the project.

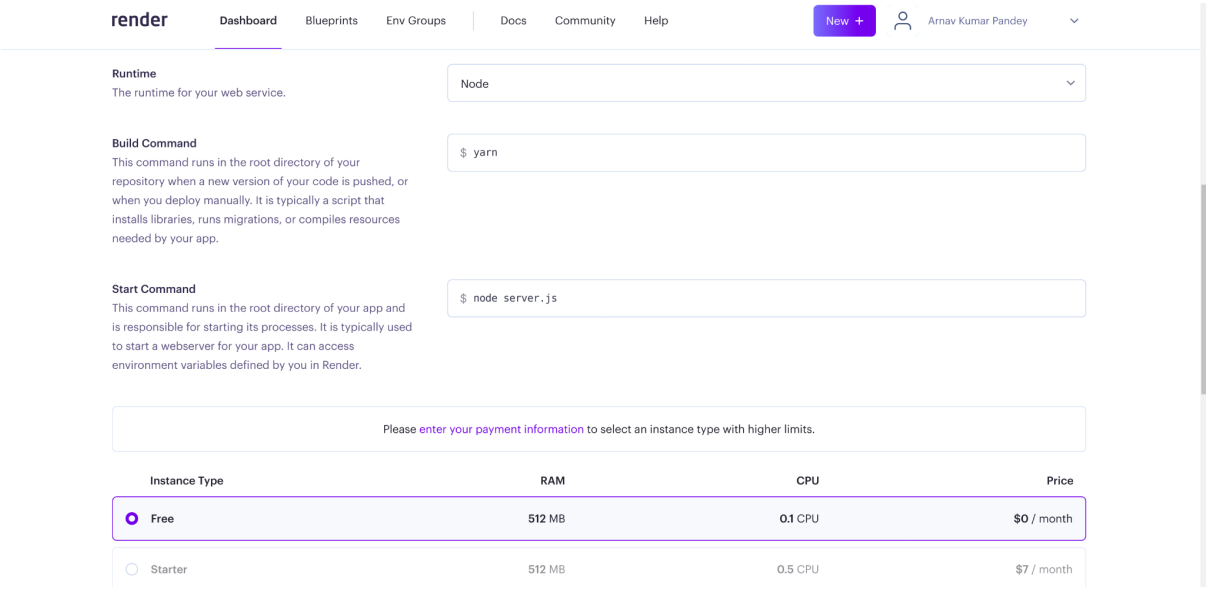


Select the repository that you want to deploy.

Configure the Service:

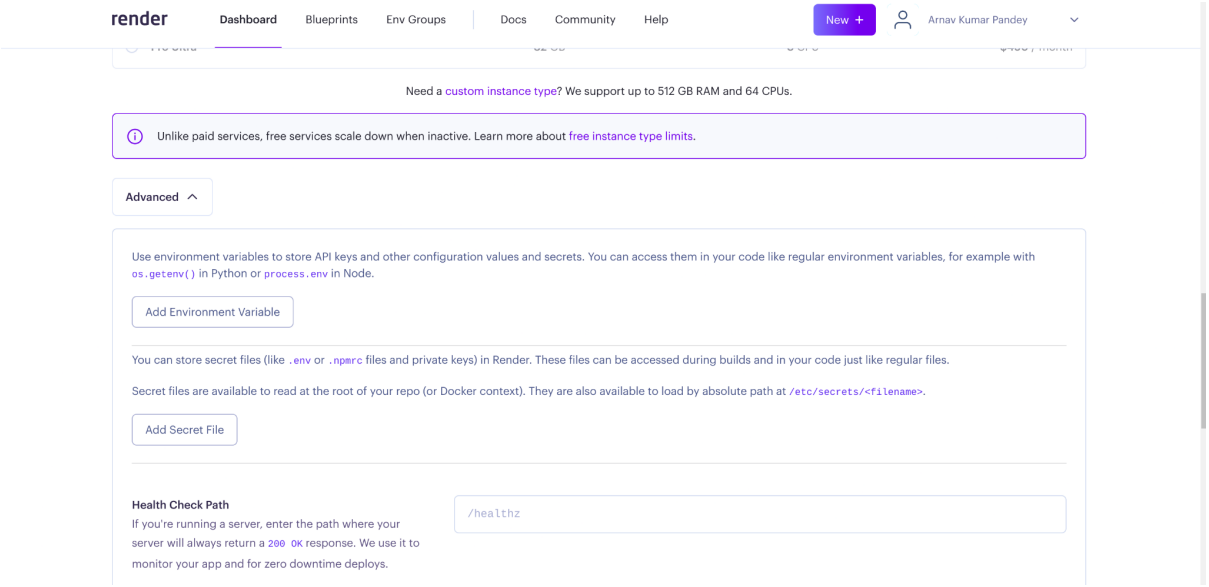


Configure the service settings, including the name, region, and branch, and set the build command to npm or yarn.



NOTE: The Root Directory is the parent directory of our project. Make sure to configure the **Start command** accordingly.

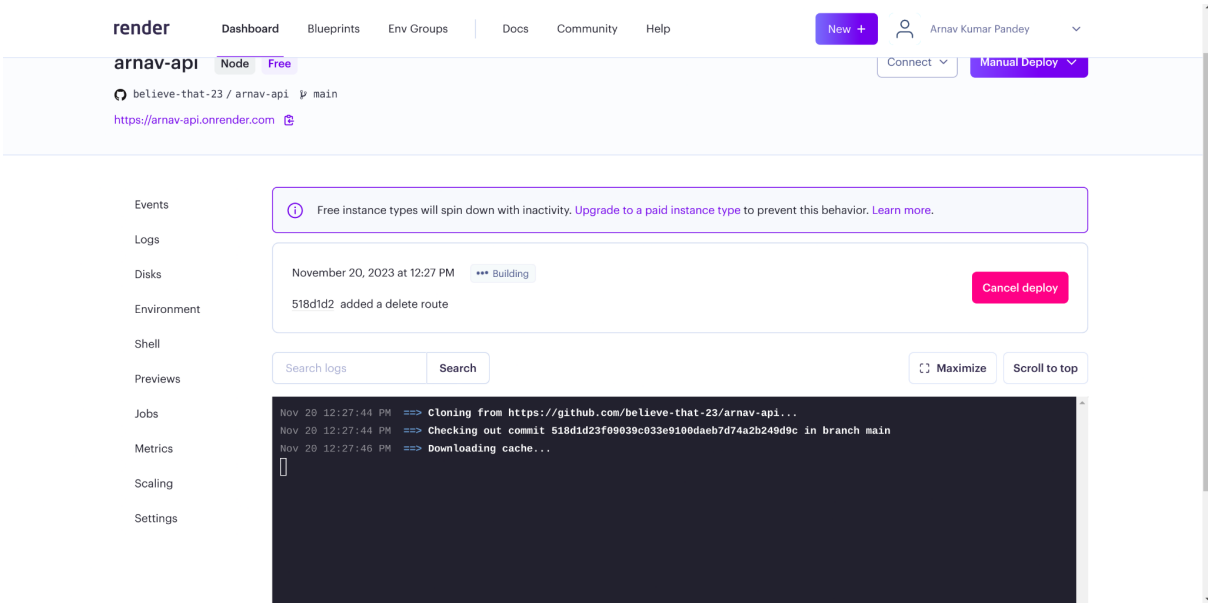
Environment Variables:



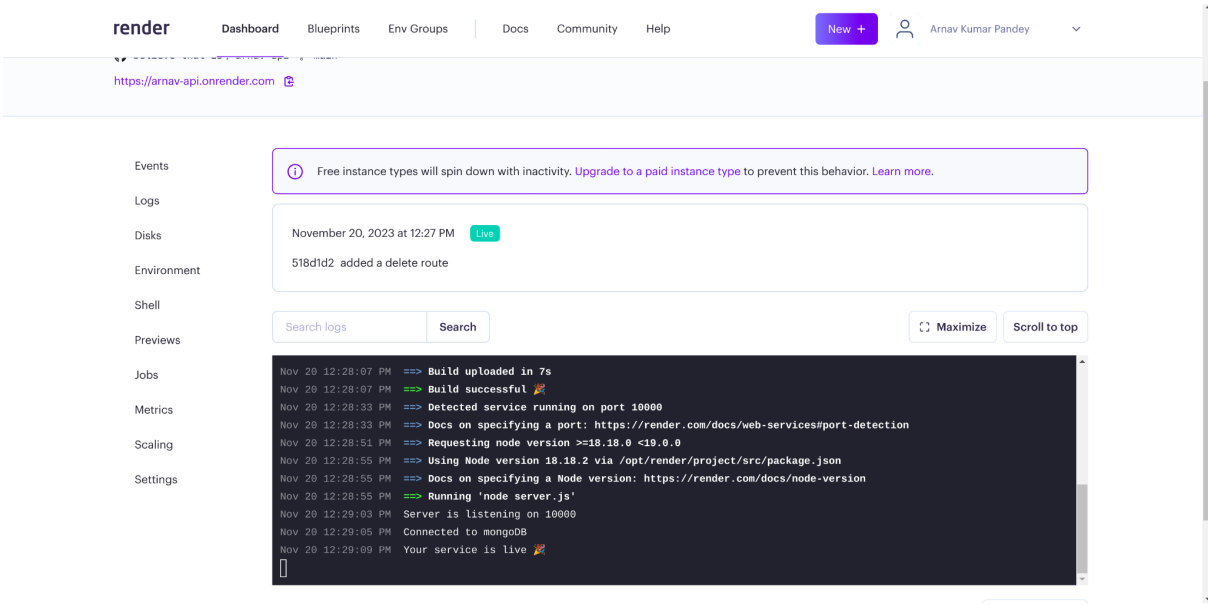
Manage environment variables carefully. Render allows you to set environment variables for your services. This is where you can store sensitive information like API keys or database credentials. You can also directly upload the env file as **Secret File**

Build and Deploy:

Trigger the initial build and deployment by Clicking the **Create Web Service** button. Render will fetch your code from the connected repository, build your application, and deploy it.



The deployment initiation process will commence, and logs will subsequently become visible.



After successful deployment, you should see the above log messages. Now you can access the deployed link.

Cloud with AWS

What is AWS?

Amazon Web Services (AWS) is a comprehensive and widely adopted cloud computing platform provided by Amazon. It offers a vast array of scalable and reliable cloud services, enabling businesses and individuals to build and deploy applications and services with flexibility and cost-effectiveness.

Why Use AWS?

- **Scalability**: AWS provides elastic and scalable computing resources, allowing users to scale up or down based on demand. This ensures optimal performance and cost efficiency.
- **Global Reach**: With a global network of data centers, AWS allows users to deploy applications and services close to their end-users, reducing latency and enhancing user experience.
- **Diverse Service Offerings**: AWS offers a broad range of services, including computing power (EC2), storage (S3), databases (RDS), machine learning (Sagemaker), and more. This diversity caters to the varied needs of different applications.
- **Security and Compliance**: AWS adheres to stringent security measures and compliance standards. It provides tools and features to help users build secure and compliant solutions, including identity and access management (IAM) and encryption services.
- **Cost Management**: AWS follows a pay-as-you-go pricing model, allowing users to pay only for the resources they consume. This flexibility is cost-effective, especially for startups and enterprises with varying workloads.
- **Innovation and Agility**: AWS continually introduces new services and features, enabling users to stay at the forefront of technological innovation. This agility is crucial for adapting to evolving business requirements.

- **Reliability and Availability**: AWS guarantees high reliability and availability through its global infrastructure. Users can design applications for fault tolerance and redundancy across multiple availability zones.
- **Managed Services**: AWS offers managed services that simplify operational tasks. For example, Amazon RDS manages database administration tasks, allowing users to focus on application development.
- **Community and Support**: AWS has a large and active community of users, providing forums, tutorials, and resources for assistance. Additionally, AWS offers various support plans to address technical issues promptly.

Benefits of Deploying on AWS Compared to render.com:

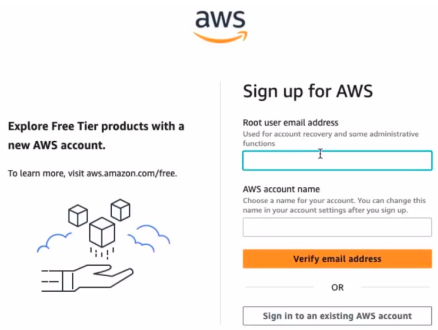
- **Service Variety**: AWS provides an extensive range of services covering almost every aspect of cloud computing, offering more options than render.com, which specializes in certain services.
- **Global Infrastructure**: AWS has a larger global infrastructure with data centers in more regions worldwide. This can be advantageous for applications with a need for geographically distributed resources.
- **Enterprise-Grade Features**: AWS offers enterprise-grade features, making it suitable for large-scale, mission-critical applications. Render.com may be more streamlined and focused on specific use cases.
- **Ecosystem and Integrations**: AWS has a rich ecosystem of partners and integrations, providing users with a vast array of tools and services to enhance their applications. Render.com may have a more focused ecosystem.
- **Industry Adoption**: AWS is one of the most widely adopted cloud platforms globally, making it a preferred choice for enterprises and businesses with complex infrastructure needs.
- **Cost Flexibility**: While render.com is known for its simplicity and transparent pricing, AWS offers more flexibility in pricing models, catering to a broader range of budgeting and usage scenarios.

Deployment on an EC2 platform

Deploying applications on Amazon Elastic Compute Cloud (EC2) involves several steps, and the exact process may vary depending on the type of application you are deploying. Below is a general guide for deploying applications on the EC2 platform:

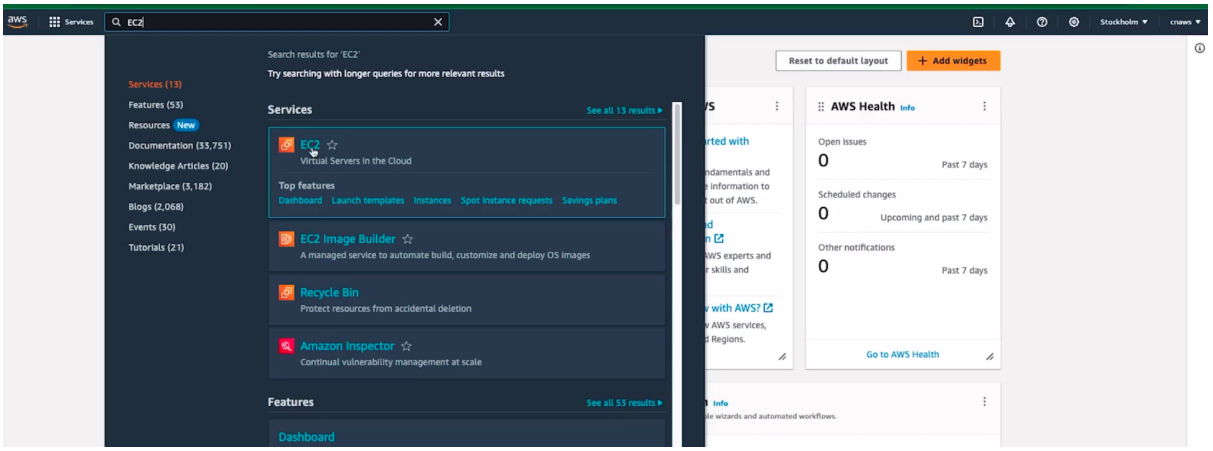
Create an AWS Account:

If you are not currently registered with AWS, please create an AWS account by accessing the AWS platform. Follow the registration process outlined in the instructional lecture video to complete the account setup.

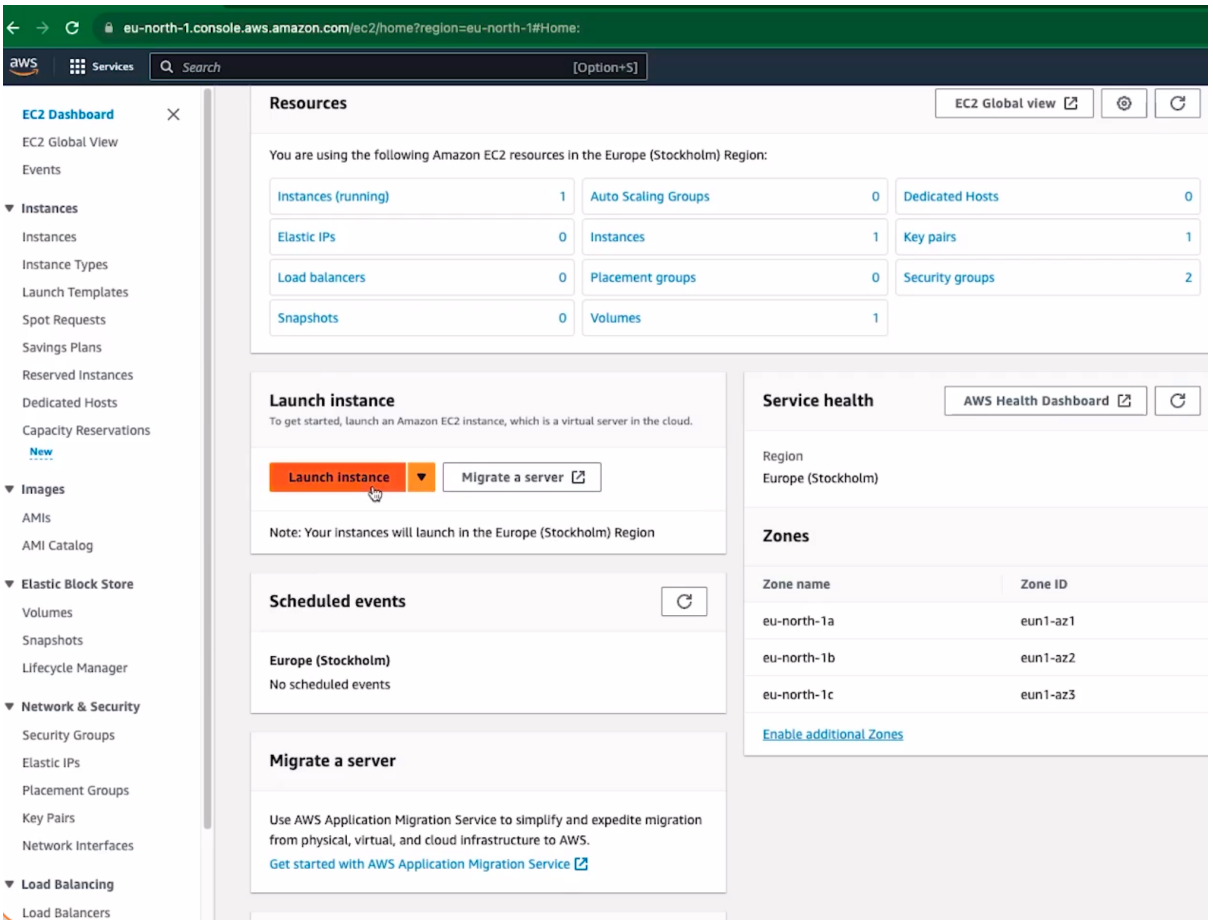


Log in to the AWS Management Console.

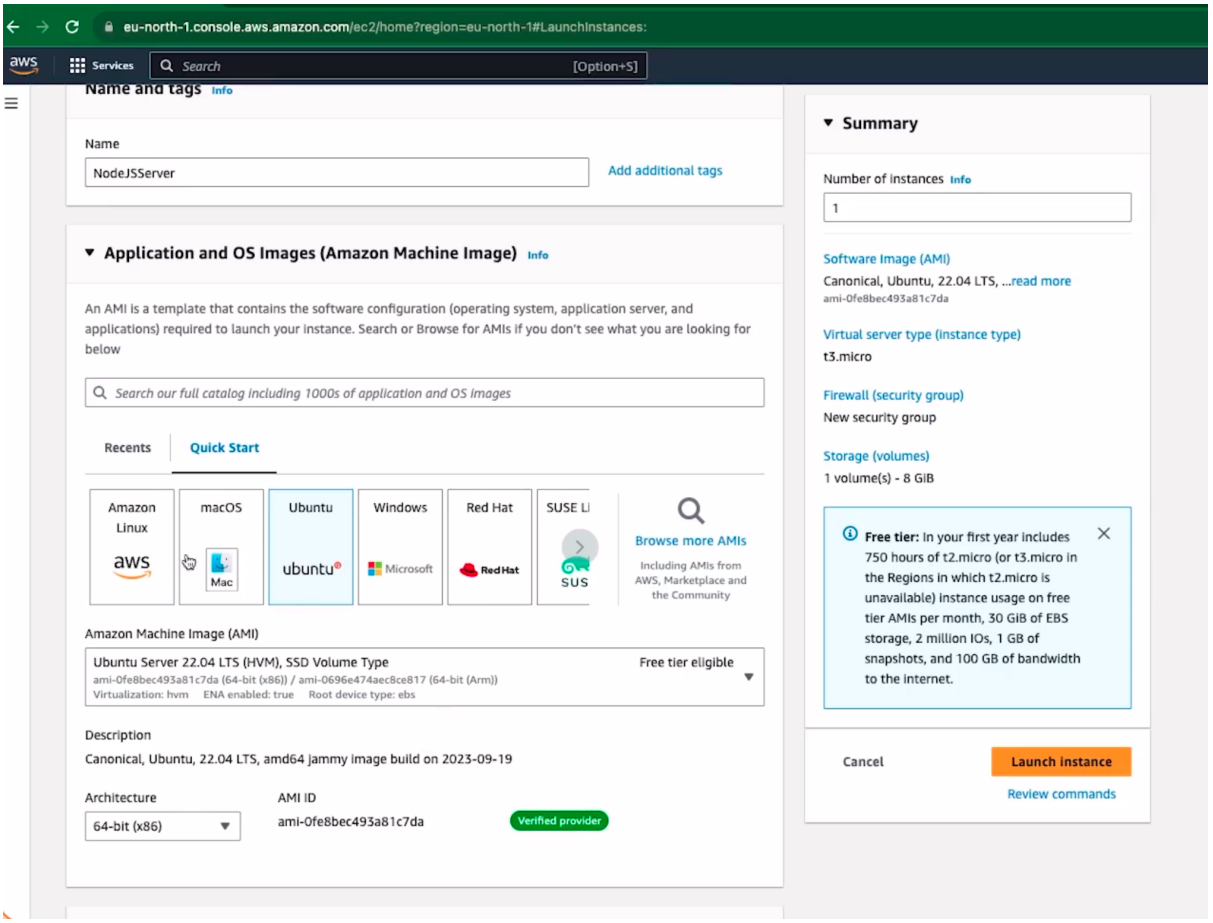
Launch an EC2 Instance:



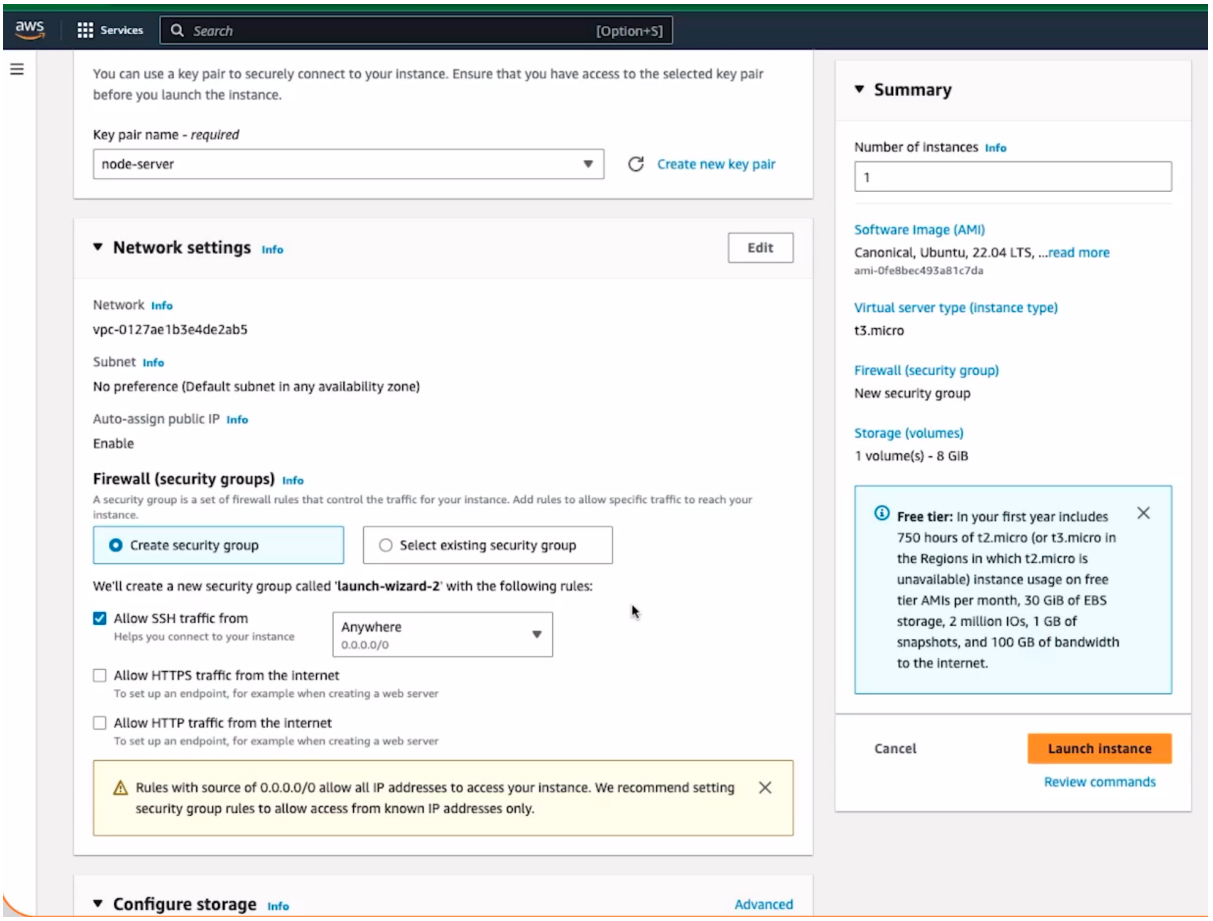
In the AWS Management Console, navigate to the EC2 dashboard.



Click on "Launch Instance" to create a new EC2 instance.



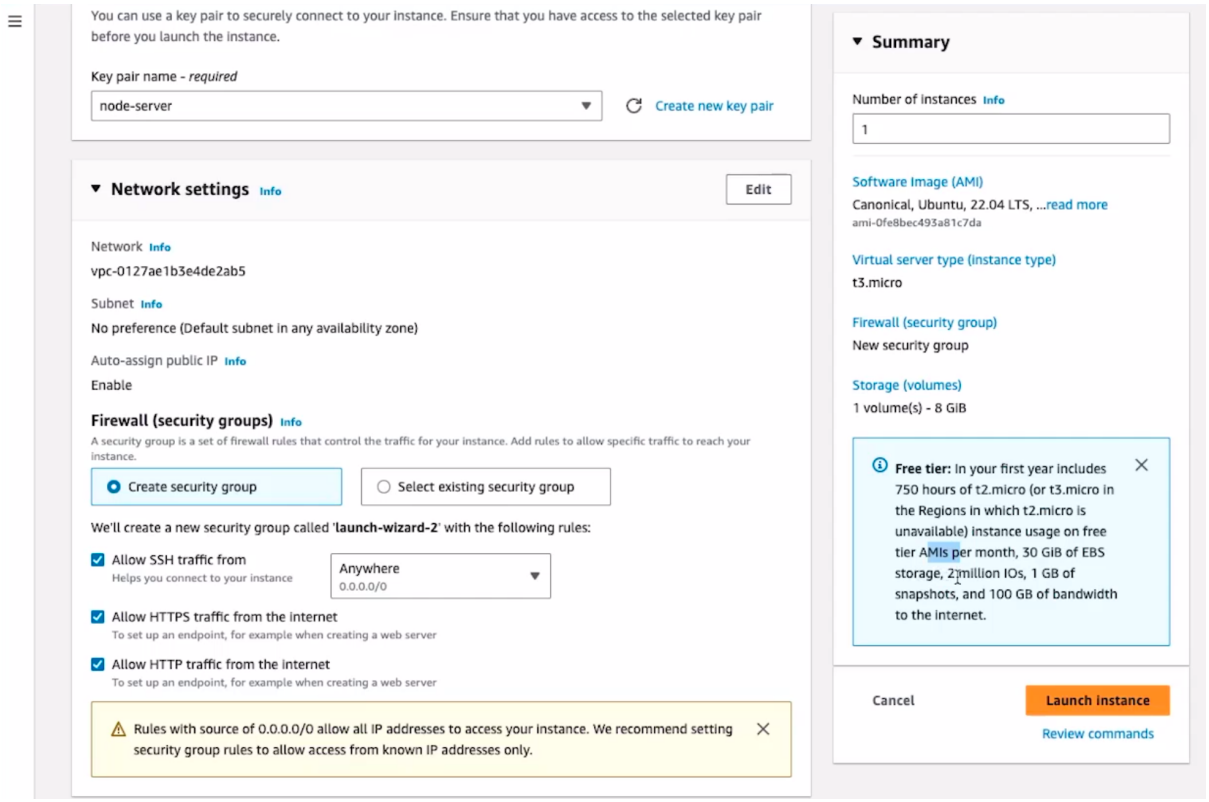
Choose an Amazon Machine Image (AMI) based on your application's requirements.



Select an instance type, configure instance details, add storage, and configure security groups (firewall settings).

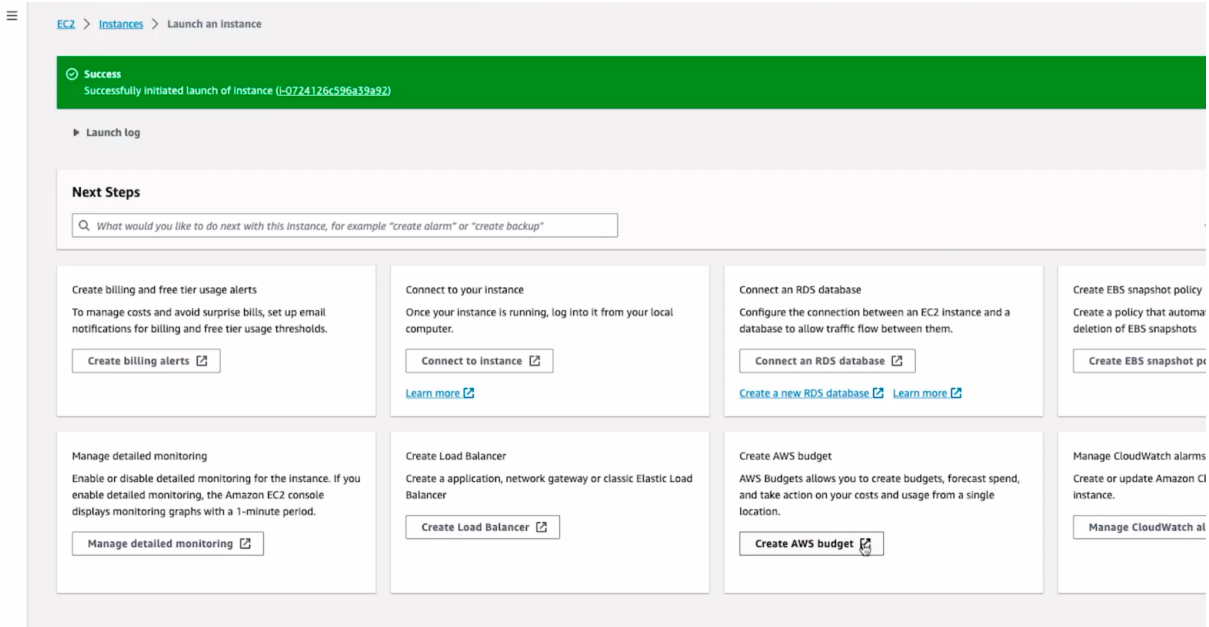
Review your settings and launch the instance.

Connect to the EC2 Instance:

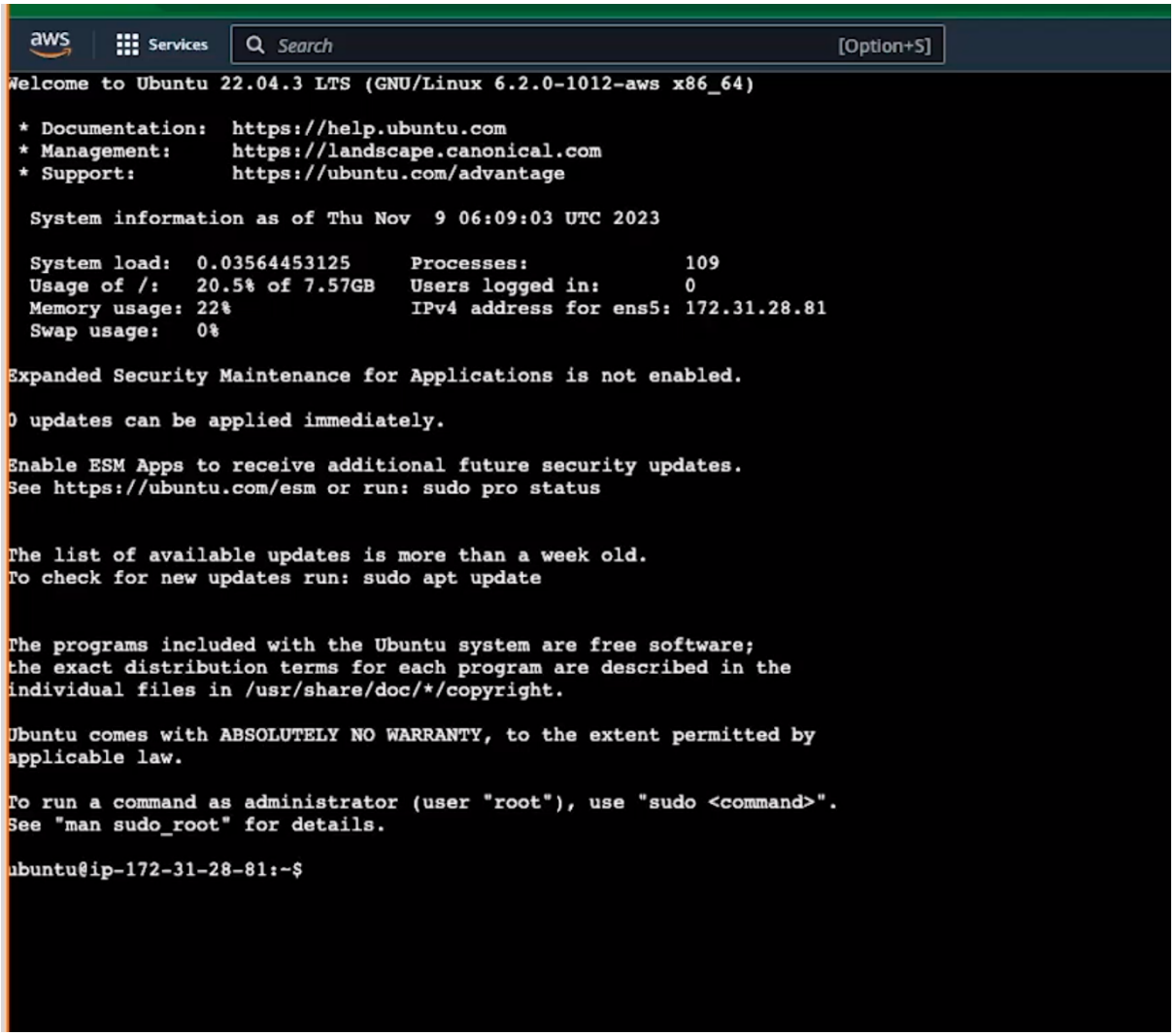


Once the instance is running, connect to it using SSH (Secure Shell) or other connection methods.

Use the private key associated with the key pair you selected when launching the instance.



Now you can launch your instance.



Update the System:

After connecting to the EC2 instance, update the package lists and install the latest updates:

```
sudo apt update
sudo apt upgrade
```

Install Required Software:

Install any software or dependencies required for your application. For example, if you are deploying a web application, you might need to install a web server (e.g., Apache or Nginx), a database (e.g., MySQL or PostgreSQL), etc.

NOTE: Follow the steps shown in the videos to get a better understanding of the above steps

Deploy Your Application:

Upload your application code to the EC2 instance. You can use tools like scp or rsync for this purpose.

scp (Secure Copy): scp is a command-line tool that uses the Secure Shell (SSH) protocol to securely copy files between a local and a remote host.

The basic syntax for scp is:

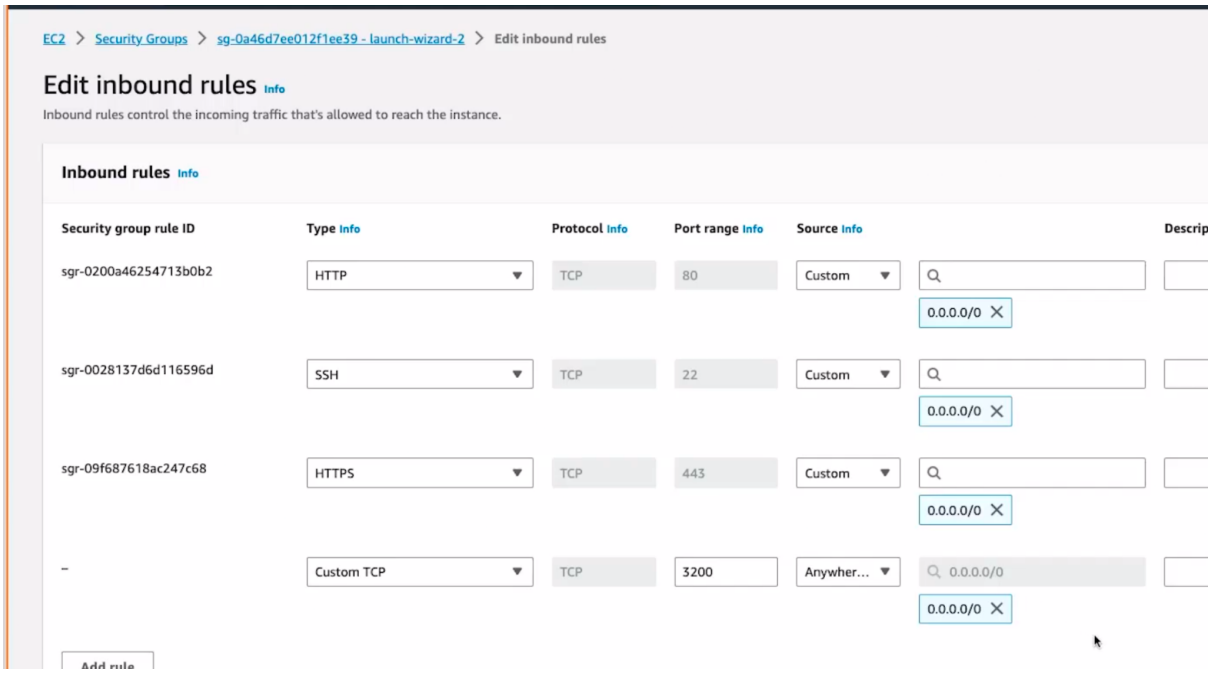
```
scp [options] source destination
```

rsync: rsync is a powerful and versatile file synchronization tool. It not only copies files but also has the ability to synchronize directories, copy files recursively, and efficiently handle incremental updates.

The basic syntax for rsync is:

```
rsync [options] source destination
```

Set up the necessary configurations for your application.



The above settings allow any user to view your deployed app.

Configure Environment Variables:

If your application relies on environment variables, set them up on the EC2 instance. You can use a configuration file or tools like AWS Systems Manager Parameter Store.

Add the export statements for your environment variables. For example

```
export MY_VARIABLE="my_value"
```

Set Up Domain and SSL (If Applicable):

If you have a domain, configure it to point to your EC2 instance's public IP address or assign an Elastic IP to the instance.
If your application requires SSL, consider using services like AWS Certificate Manager or Let's Encrypt.

PM2 (Process Manager 2):

Overview:

PM2 is a production process manager for Node.js applications.
It enables easy management, monitoring, and scaling of Node.js processes.

Key Features:

- Process Management: Automatically restarts crashed applications. Monitors application health.
- Logging: Centralized logging of application output. Supports log rotation.
- Load Balancing: Enables load balancing between multiple instances.
- Zero Downtime Deployment: Supports seamless deployment without downtime.
- Startup Scripts: Can generate startup scripts to ensure persistence after reboots.

Installation:

Install globally via npm: `sudo npm install -g pm2`

Basic Commands:

- **Start an application**: pm2 start app.js.

- **List running processes**: pm2 list.
- **Monitor CPU/Memory usage**: pm2 monit.
- **Stop an application:** pm2 stop app_name.

Use Cases:

Ideal for deploying and managing Node.js applications in production environments. Widely used for applications requiring high availability and reliability.

Summarising it

Let’s summarise what we have learned in this module:

- We discussed deployment and the need for deployment.
- We also observed various deployment environments.
- We observed the deployment of our project on Render.
- We discussed AWS, its need, and how it is preferred over Render.
- We also discussed the steps for deployment on Amazon EC2(Elastic Compute Cloud).

Some Additional Resources:

- How To Secure Apache with Let's Encrypt on Ubuntu 20.04