

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:

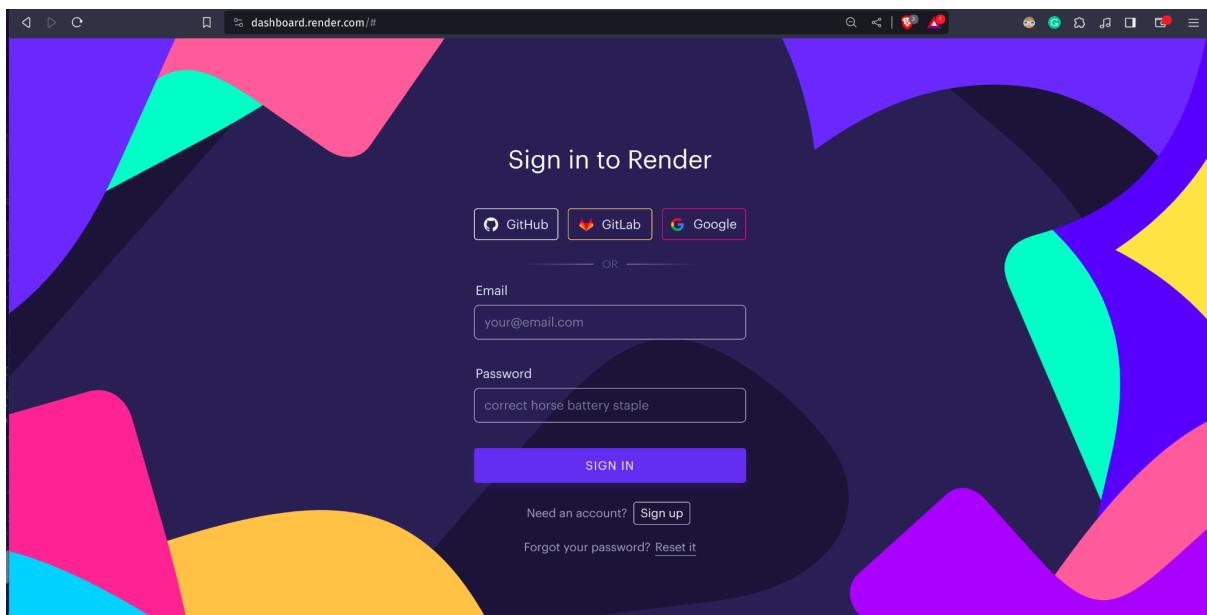
This environment is specifically for user acceptance testing, allowing stakeholders to validate that the application meets their expectations before deployment to production.

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:

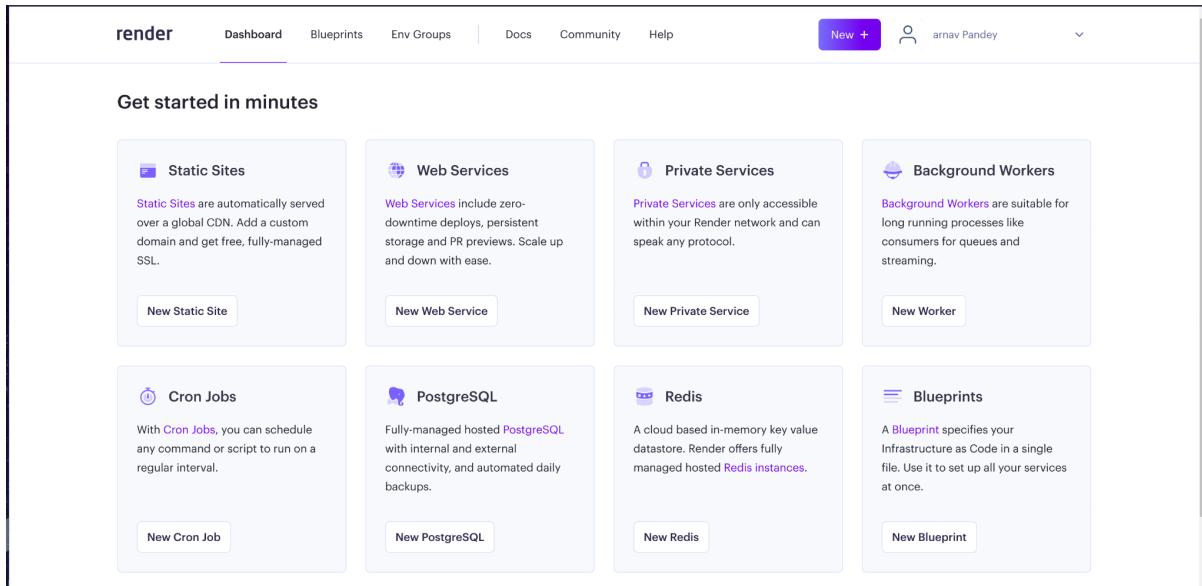
Create an Account:

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:

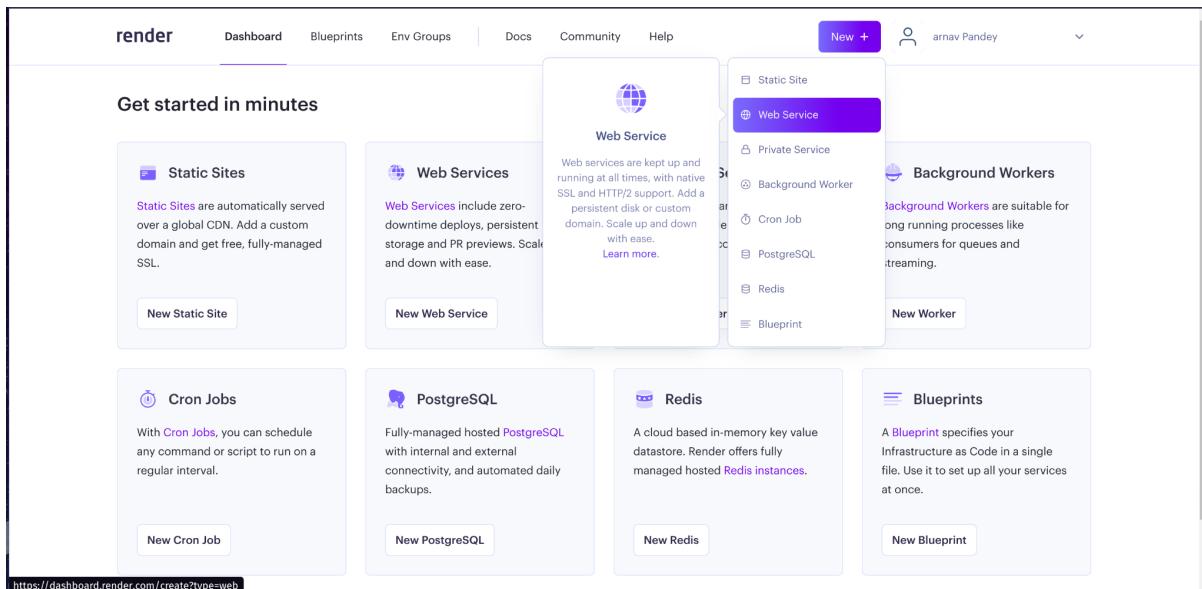


The screenshot shows the Render dashboard with a navigation bar at the top. The 'Blueprints' tab is selected. A user profile for 'arnav Pandey' is visible. Below the navigation, a 'Get started in minutes' section displays eight service categories with icons and brief descriptions, each with a 'New [Service Type]' button.

- Static Sites**: Automatically served over a global CDN. Add a custom domain and get free, fully-managed SSL. (New Static Site)
- Web Services**: Include zero-downtime deploys, persistent storage and PR previews. Scale up and down with ease. (New Web Service)
- Private Services**: Only accessible within your Render network and can speak any protocol. (New Private Service)
- Background Workers**: Suitable for long running processes like consumers for queues and streaming. (New Worker)
- Cron Jobs**: You can schedule any command or script to run on a regular interval. (New Cron Job)
- PostgreSQL**: Fully-managed hosted PostgreSQL with internal and external connectivity, and automated daily backups. (New PostgreSQL)
- Redis**: A cloud based in-memory key value datastore. Render offers fully managed hosted Redis instances. (New Redis)
- Blueprints**: A Blueprint specifies your Infrastructure as Code in a single file. Use it to set up all your services at once. (New Blueprint)

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

Create a New Service:



This screenshot is similar to the previous one but with a specific service highlighted: 'Web Service'. The 'Web Service' card is expanded, showing more detailed information and a 'Learn more.' link. The other service cards are partially visible on the right.

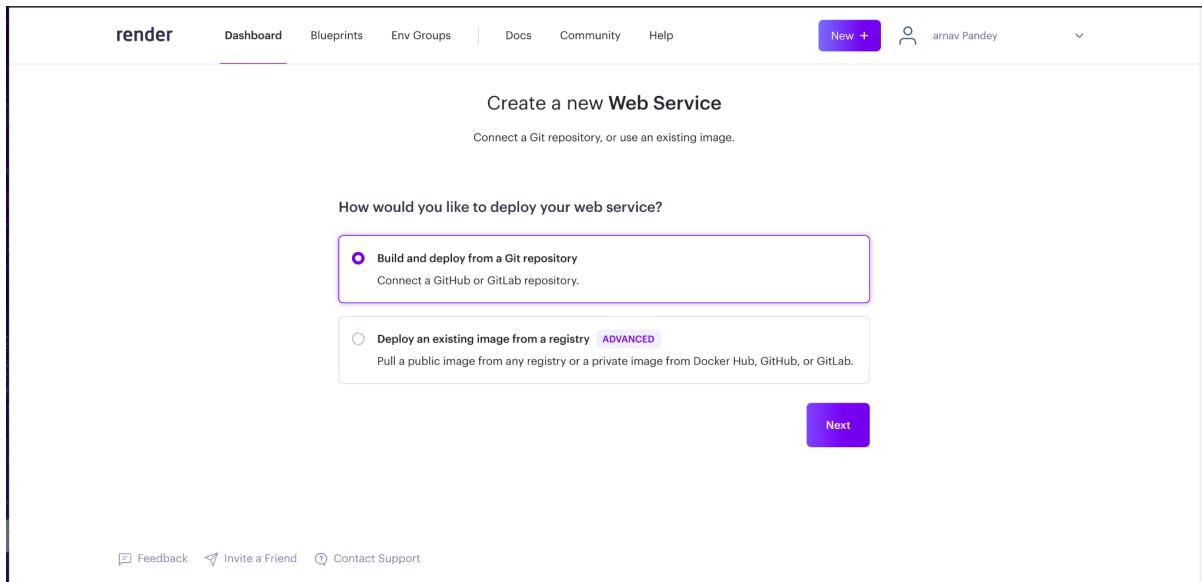
Web Service details:

- Web services are kept up and running at all times, with native SSL and HTTP/2 support. Add a persistent disk or custom domain. Scale up and down with ease.
- [Learn more.](#)

<https://dashboard.render.com/create?type=web>

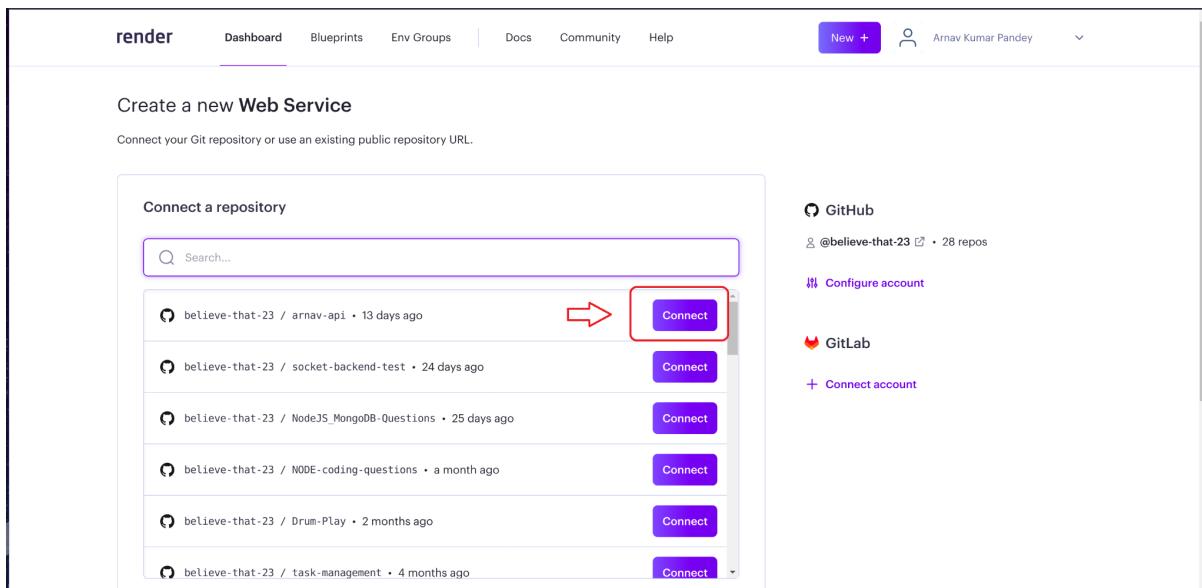
To deploy an application, you need to create a service on Render. This could be a web service, database, or other types of services depending on your application's requirements.

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



The screenshot shows the 'Create a new Web Service' interface. At the top, there are navigation links: render, Dashboard, Blueprints, Env Groups, Docs, Community, Help, and a New + button. On the right, it shows the user arnav Pandey. Below the header, the title 'Create a new Web Service' is displayed, followed by the instruction 'Connect a Git repository, or use an existing image.' A question 'How would you like to deploy your web service?' has two options: 'Build and deploy from a Git repository' (selected) and 'Deploy an existing image from a registry'. The 'Next' button is at the bottom right of the main form area.

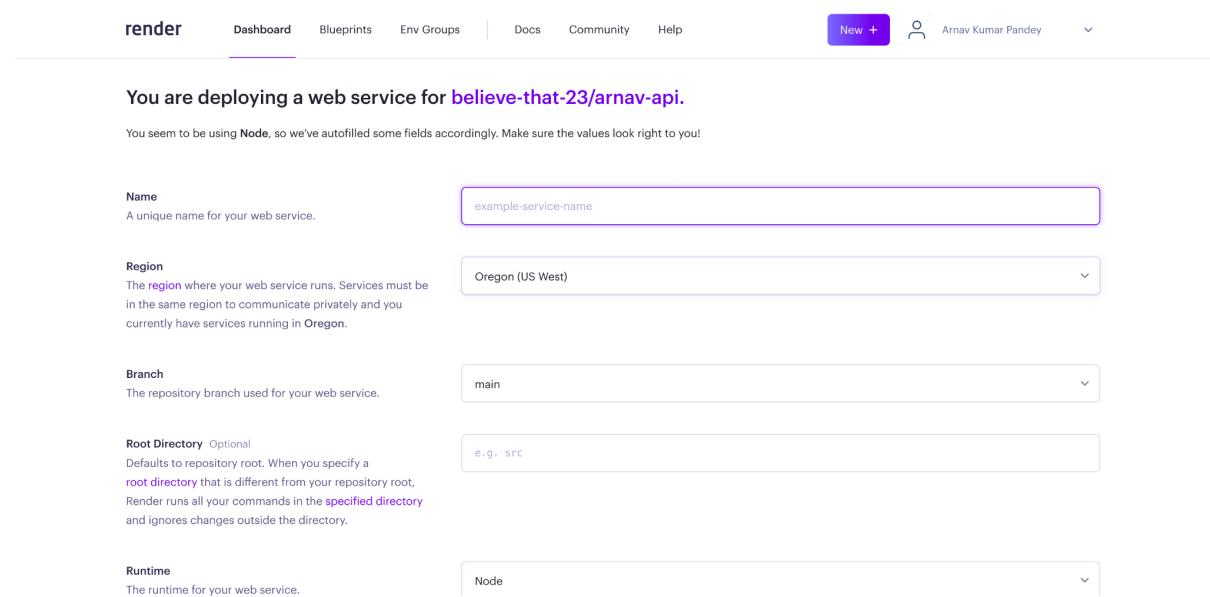
Now one can connect their GitHub account with the same to deploy the project.



The screenshot shows the 'Create a new Web Service' interface again. The 'Connect a repository' section is highlighted. It shows a search bar and a list of repositories belonging to the user '@believe-that-23'. One repository, 'believe-that-23 / arnav-api', has a red arrow pointing to its 'Connect' button, which is also highlighted with a red box. To the right of the repository list, there are sections for GitHub ('@believe-that-23 • 28 repos') and GitLab ('+ Connect account').

Select the repository that you want to deploy.

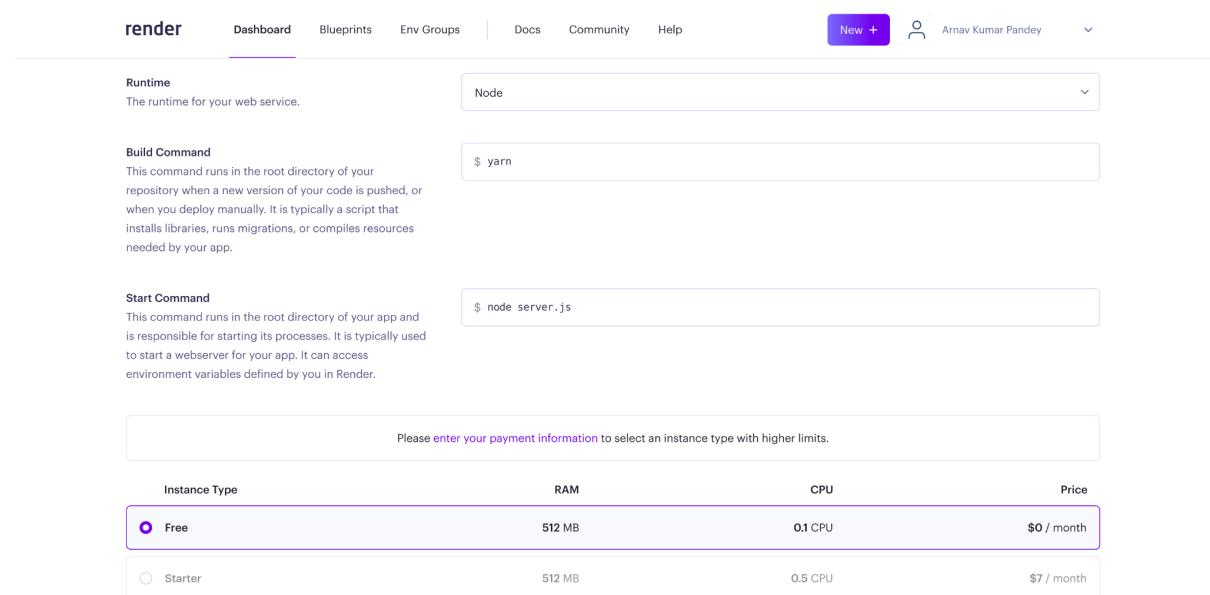
Configure the Service:



You are deploying a web service for **believe-that-23/arnav-api**. You seem to be using **Node**, so we've autofilled some fields accordingly. Make sure the values look right to you!

Name A unique name for your web service.	example-service-name
Region The region where your web service runs. Services must be in the same region to communicate privately and you currently have services running in Oregon .	Oregon (US West)
Branch The repository branch used for your web service.	main
Root Directory <small>Optional</small> Defaults to repository root. When you specify a root directory that is different from your repository root, Render runs all your commands in the specified directory and ignores changes outside the directory.	e.g. <code>src</code>
Runtime The runtime for your web service.	Node

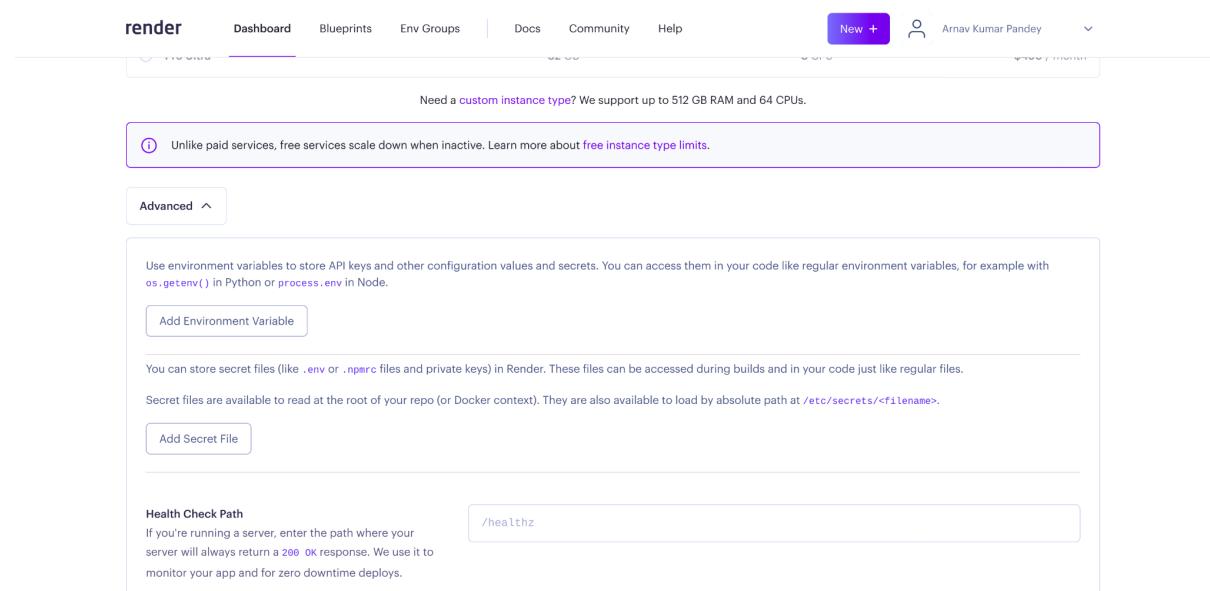
Configure the service settings, including the name, region, and branch, and set the build command to npm or yarn.



Runtime The runtime for your web service.	Node		
Build Command This command runs in the root directory of your repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app.	<code>\$ yarn</code>		
Start Command This command runs in the root directory of your app and is responsible for starting its processes. It is typically used to start a webserver for your app. It can access environment variables defined by you in Render.	<code>\$ node server.js</code>		
Please enter your payment information to select an instance type with higher limits.			
Instance Type	RAM	CPU	Price
<input checked="" type="radio"/> Free	512 MB	0.1 CPU	\$0 / month
<input type="radio"/> Starter	512 MB	0.5 CPU	\$7 / month

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

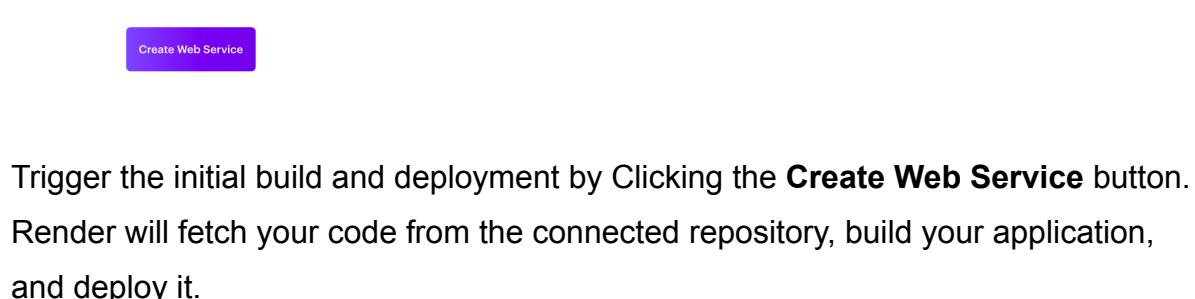
Environment Variables:



The screenshot shows the Render platform's environment variables configuration page. At the top, there are navigation links for Dashboard, Blueprints, Env Groups, Docs, Community, Help, and a user profile for Arnav Kumar Pandey. A purple 'New +' button is visible. Below the header, a message states: "Need a [custom instance type](#)? We support up to 512 GB RAM and 64 CPUs." A note below it says: "Unlike paid services, free services scale down when inactive. Learn more about [free instance type limits](#)." An 'Advanced' dropdown menu is open. A section titled "Use environment variables to store API keys and other configuration values and secrets. You can access them in your code like regular environment variables, for example with `os.getenv()` in Python or `process.env` in Node." contains a "Add Environment Variable" button. Another section for secret files is present with an "Add Secret File" button. A "Health Check Path" input field is set to "/healthz".

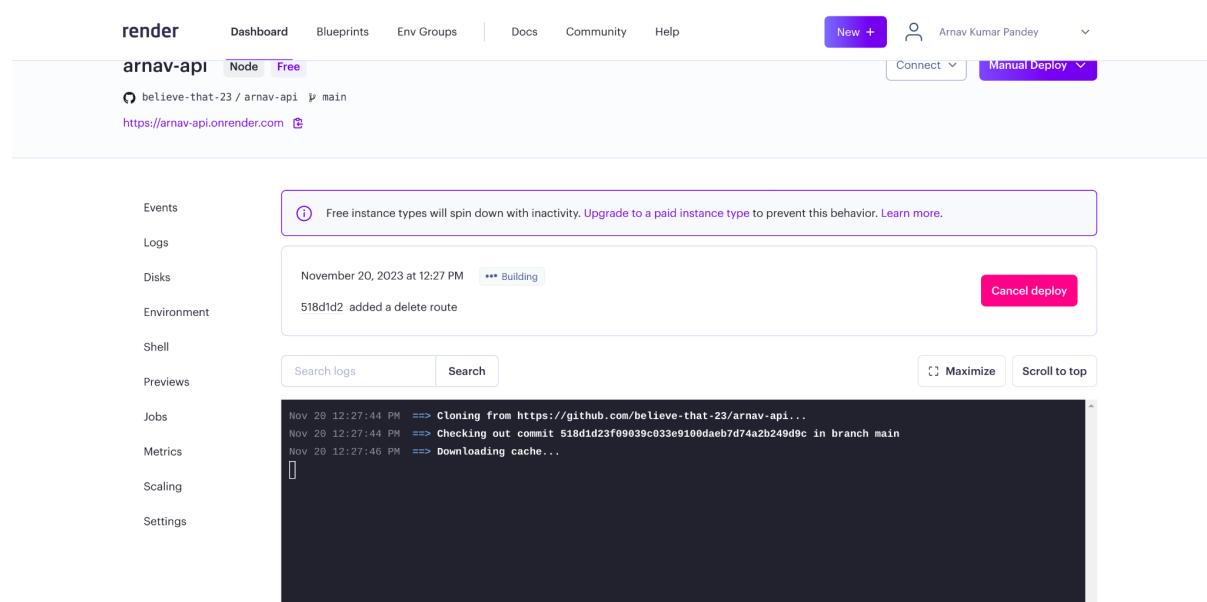
Manage environment variables carefully. Render allows you to set environment variables for your services. This is where you can store sensitive information like API keys or database credentials. You can also directly upload the env file as **Secret File**

Build and Deploy:

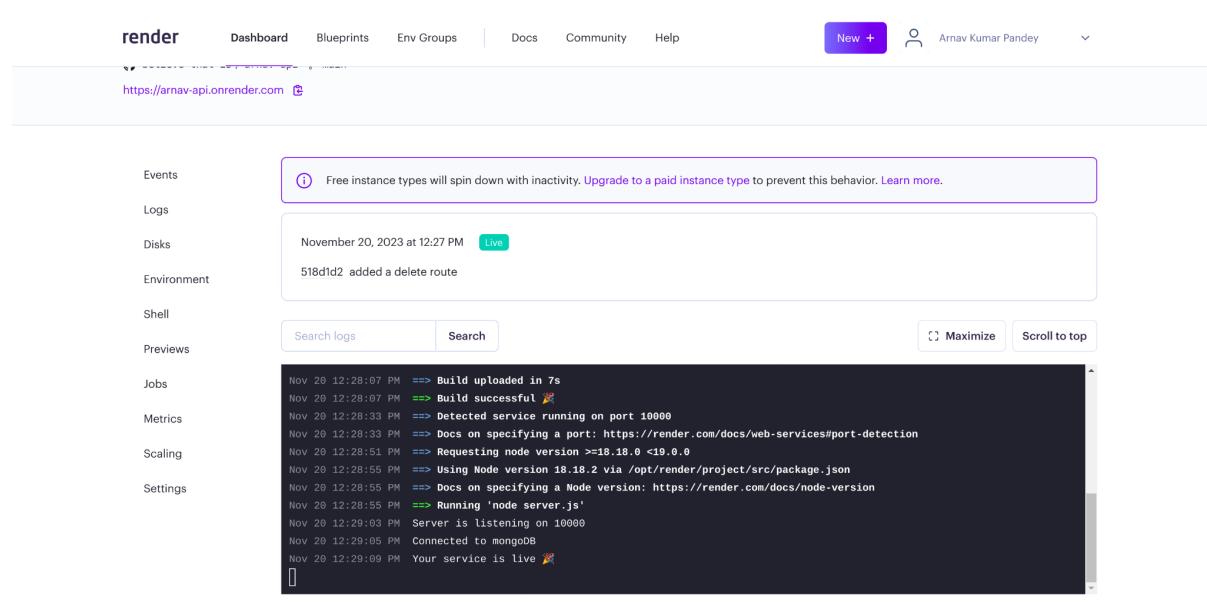


The screenshot shows the Render platform's main dashboard. A prominent purple "Create Web Service" button is located at the top left. The rest of the page is mostly blank, indicating the initial state before any deployment actions are taken.

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:

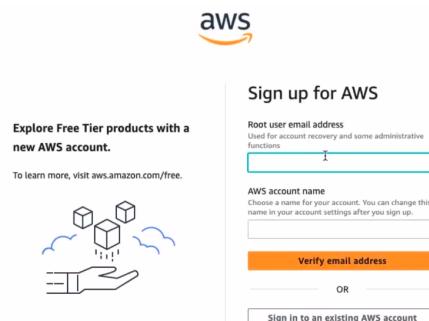
- **Service Variety:** AWS provides an extensive range of services covering almost every aspect of cloud computing, offering more options than render.com, which specializes in certain services.
- **Global Infrastructure:** AWS has a larger global infrastructure with data centers in more regions worldwide. 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:

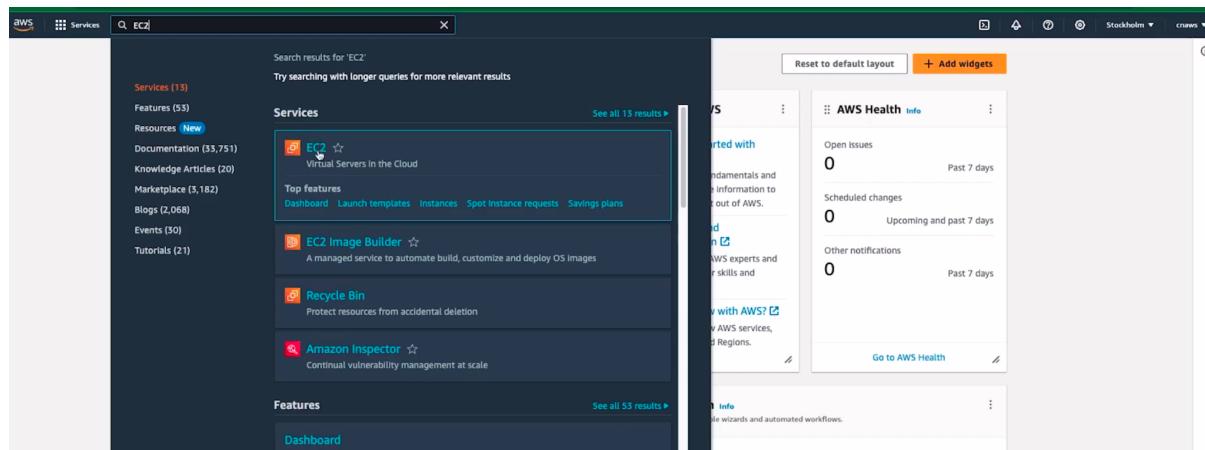
Create an AWS Account:

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.

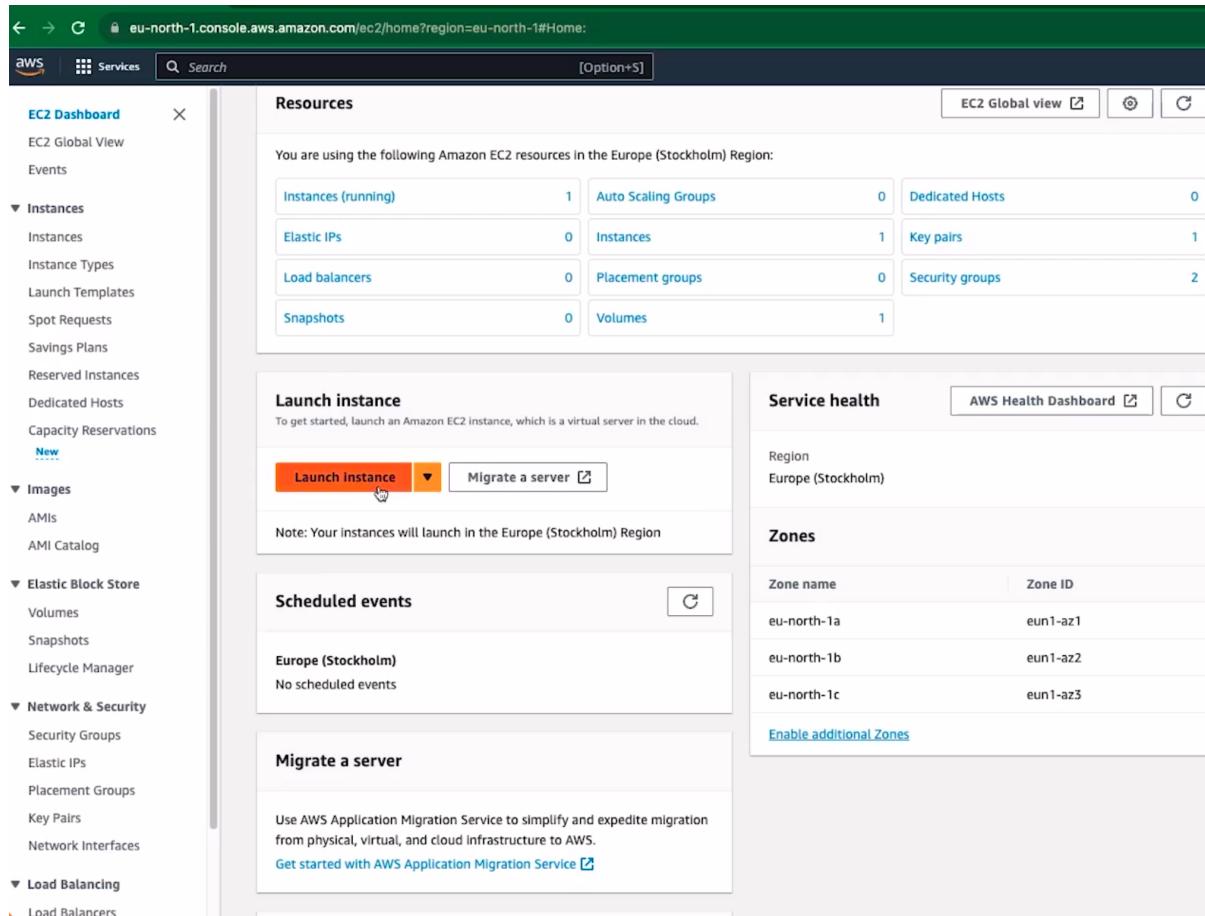


Log in to the AWS Management Console.

Launch an EC2 Instance:

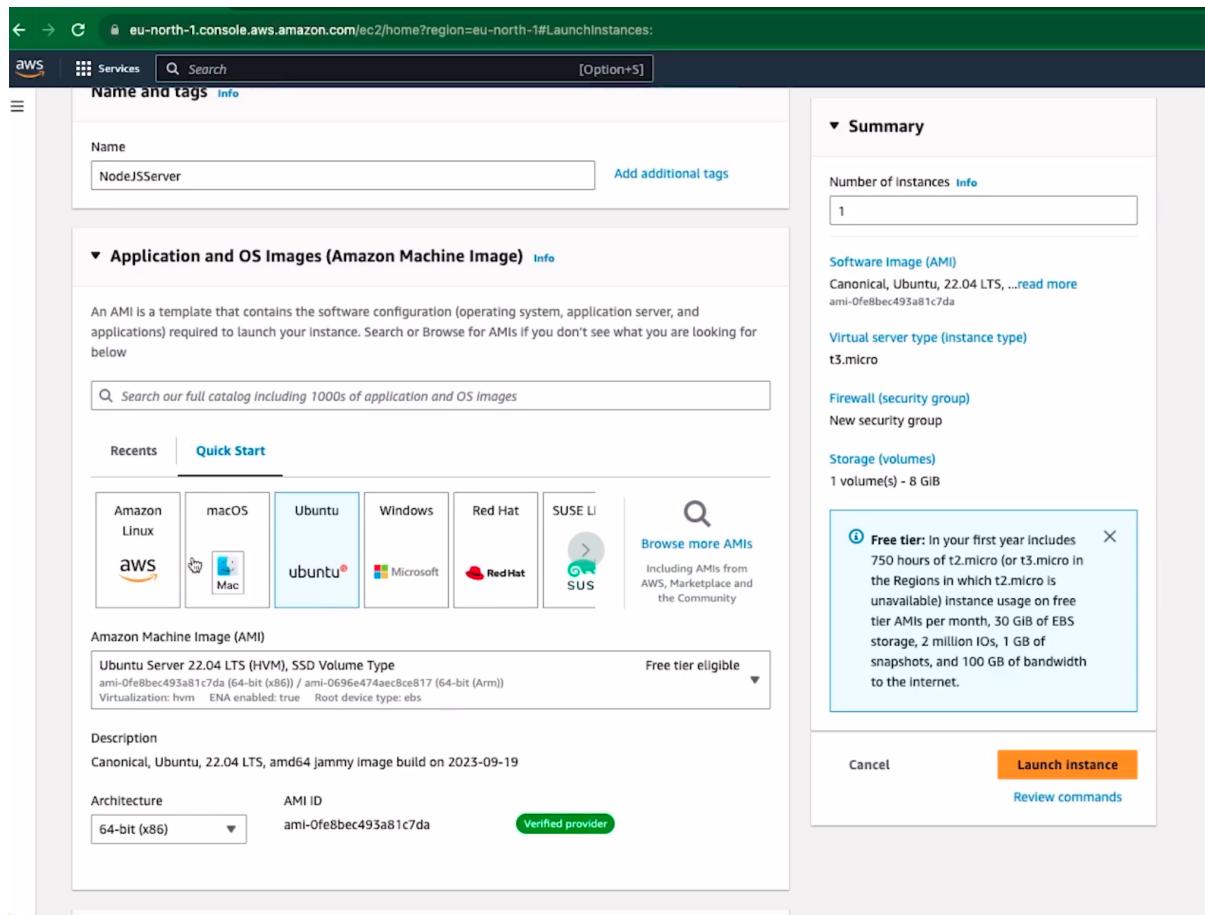


In the AWS Management Console, navigate to the EC2 dashboard.



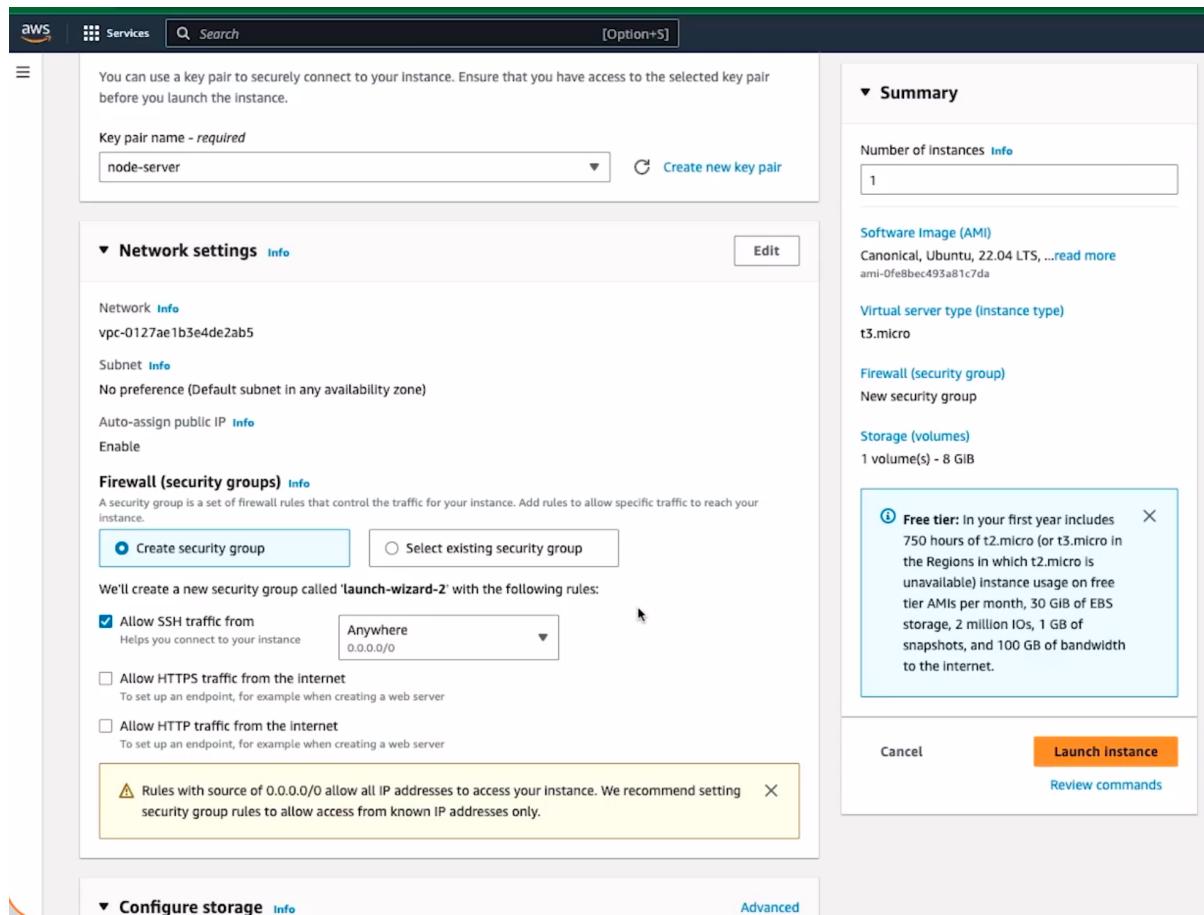
Zone name	Zone ID
eu-north-1a	eun1-az1
eu-north-1b	eun1-az2
eu-north-1c	eun1-az3

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



The screenshot shows the AWS CloudFormation console with a stack named "NodeJSServer" in progress. The "Template" tab is active, displaying the CloudFormation template. The "Outputs" tab shows the output "MyWebAppURL" with the value "https://d13yacurqjgxfc.cloudfront.net/". The "Next Step" button is highlighted in orange at the bottom right.

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



The screenshot shows the AWS Launch Wizard interface for creating a new Amazon EC2 instance. The process is divided into several steps:

- Key pair name - required:** A dropdown menu shows "node-server". To its right is a "Create new key pair" button.
- Network settings:** Shows a Network (Info) section with "vpc-0127ae1b3e4de2ab5" and a Subnet (Info) section indicating "No preference (Default subnet in any availability zone)".
- Firewall (security groups):** A note says "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." It includes two options: "Create security group" (selected) and "Select existing security group". Below this, it says "We'll create a new security group called 'launch-wizard-2' with the following rules:" and lists three checkboxes:
 - Allow SSH traffic from Anywhere (0.0.0.0/0)
 - Allow HTTPS traffic from the internet
 - Allow HTTP traffic from the internet
 A warning message in a box states: "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." There is a close button "X" next to the message.
- Configure storage:** An "Advanced" link is visible at the bottom right of this section.
- Summary:** Shows "Number of instances" (1), "Software Image (AMI)" (Canonical, Ubuntu, 22.04 LTS), "Virtual server type (Instance type)" (t3.micro), "Firewall (security group)" (New security group), and "Storage (volumes)" (1 volume(s) - 8 GiB).
- Free tier information:** A tooltip box provides details: "Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GiB of bandwidth to the internet." It includes a close button "X".
- Launch Instance:** A large orange button at the bottom right.
- Review commands:** A link located just below the "Launch Instance" button.

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:

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*
 [Create new key pair](#)

▼ Network settings [Info](#) [Edit](#)

Network [Info](#)
vpc-0127ae1b3e4de2ab5

Subnet [Info](#)
No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)
Enable

Firewall (security groups) [Info](#)
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-2' with the following rules:

Allow SSH traffic from Anywhere
Helps you connect to your instance
0.0.0.0/0

Allow HTTPS traffic from the Internet
To set up an endpoint, for example when creating a web server

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 [Info](#)
1

Software Image (AMI)
Canonical, Ubuntu, 22.04 LTS, ...[read more](#)
ami-0fe8bec493a81c7da

Virtual server type (instance type)
t3.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

ⓘ Free tier: In your first year includes [X](#)
750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GiB of bandwidth to the internet.

[Cancel](#) [Launch instance](#) [Review commands](#)

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.

EC2 > Instances > Launch an Instance

Success
Successfully initiated launch of instance (i-0724126c596a59a2)

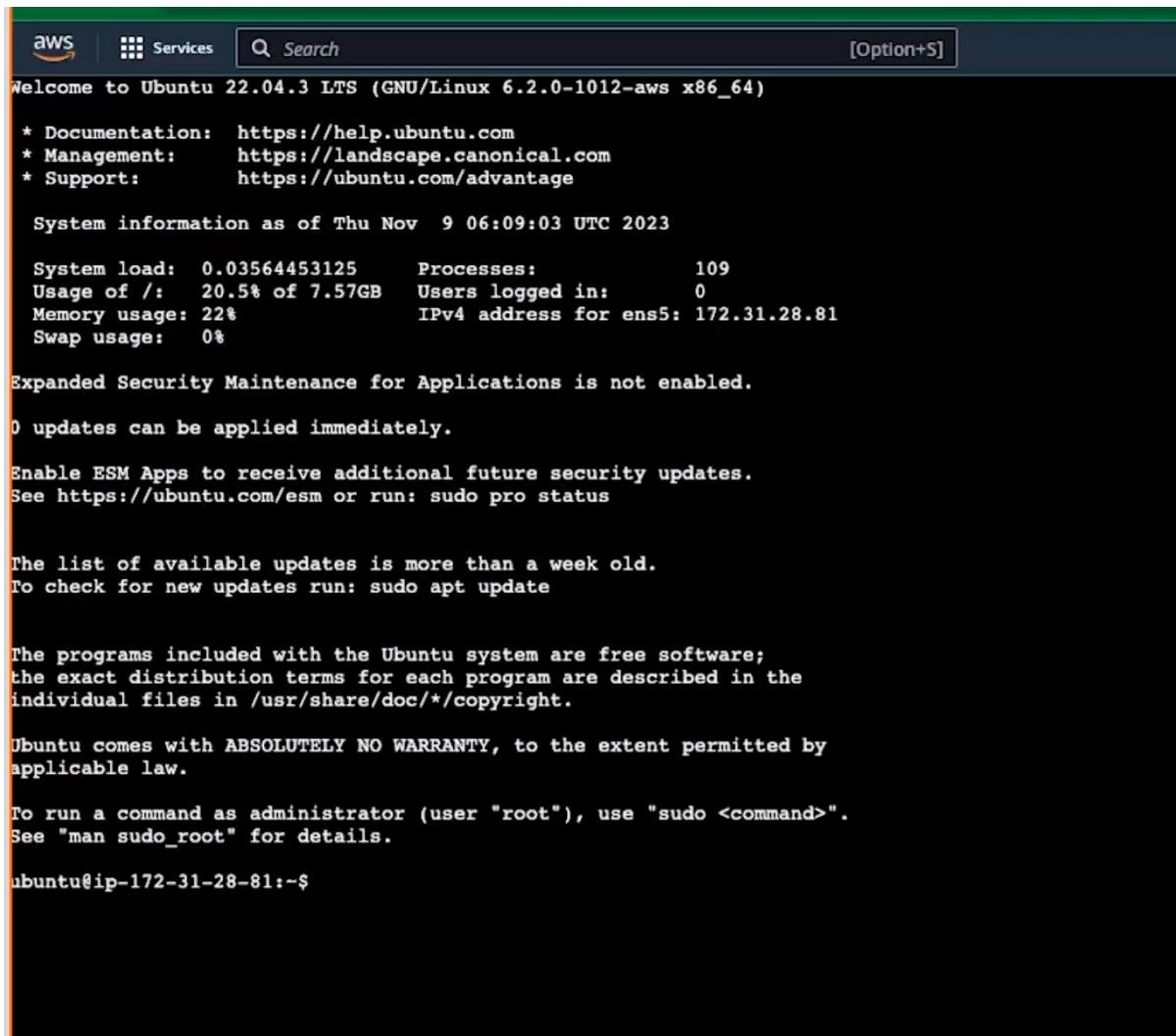
[Launch log](#)

Next Steps

[Q. What would you like to do next with this instance, for example "create alarm" or "create backup"](#)

Create billing and free tier usage alerts To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds. Create billing alerts	Connect to your instance Once your instance is running, log into it from your local computer. Connect to instance Learn more	Connect an RDS database Configure the connection between an EC2 instance and a database to allow traffic flow between them. Connect an RDS database Create a new RDS database Learn more	Create EBS snapshot policy Create a policy that automatically deletes EBS snapshots. Create EBS snapshot policy
Manage detailed monitoring Enable or disable detailed monitoring for the instance. If you enable detailed monitoring, the Amazon EC2 console displays monitoring graphs with a 1-minute period. Manage detailed monitoring	Create Load Balancer Create an application, network gateway or classic Elastic Load Balancer. Create Load Balancer	Create AWS budget AWS Budgets allows you to create budgets, forecast spend, and take action on your costs and usage from a single location. Create AWS budget	Manage CloudWatch alarms Create or update Amazon CloudWatch metrics and alarms. Manage CloudWatch alarms

Now you can launch your instance.



```
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-1012-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 System information as of Thu Nov  9 06:09:03 UTC 2023

 System load: 0.03564453125   Processes:          109
 Usage of /: 20.5% of 7.57GB  Users logged in:    0
 Memory usage: 22%           IPv4 address for ens5: 172.31.28.81
 Swap usage:  0%

 Expanded Security Maintenance for Applications is not enabled.

 0 updates can be applied immediately.

 Enable ESM Apps to receive additional future security updates.
 See https://ubuntu.com/esm or run: sudo pro status

 The list of available updates is more than a week old.
 To check for new updates run: sudo apt update

 The programs included with the Ubuntu system are free software;
 the exact distribution terms for each program are described in the
 individual files in /usr/share/doc/*copyright.

 Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
 applicable law.

 To run a command as administrator (user "root"), use "sudo <command>".
 See "man sudo_root" for details.

ubuntu@ip-172-31-28-81:~$
```

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:

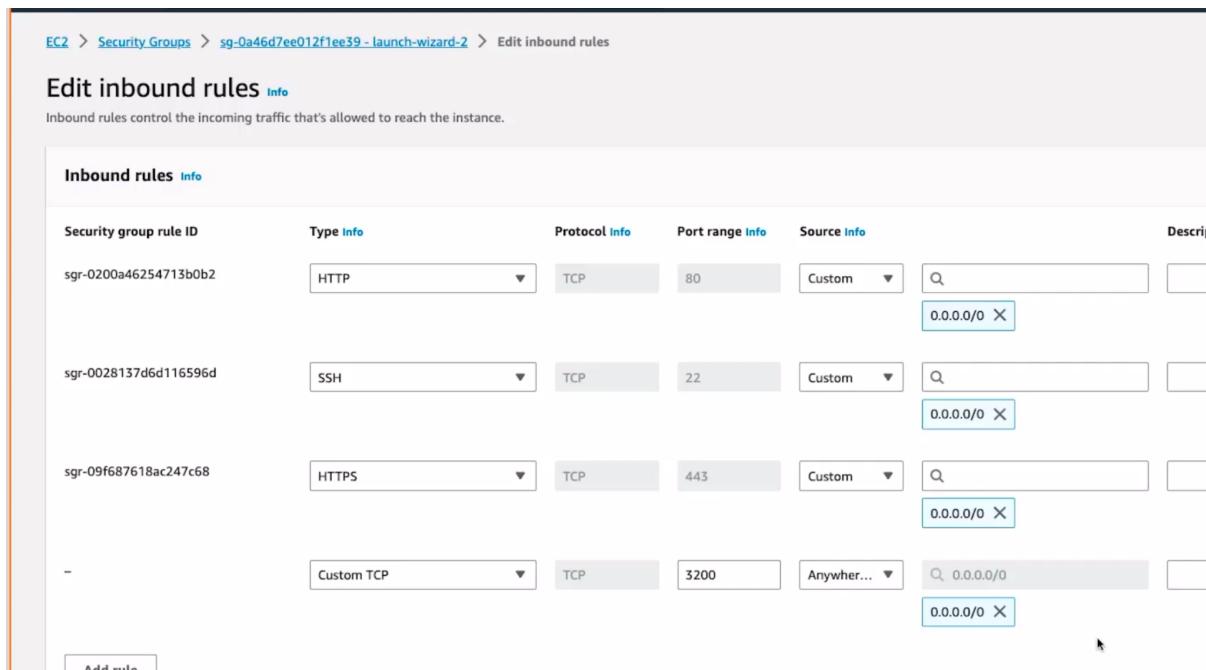
```
scp [options] source destination
```

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 screenshot shows the 'Edit inbound rules' section of the AWS EC2 Security Groups configuration. There are four existing rules listed:

- sgr-0200a46254713b0b2: Type: HTTP, Protocol: TCP, Port range: 80, Source: Custom (0.0.0.0/0)
- sgr-0028137d6d116596d: Type: SSH, Protocol: TCP, Port range: 22, Source: Custom (0.0.0.0/0)
- sgr-09f687618ac247c68: Type: HTTPS, Protocol: TCP, Port range: 443, Source: Custom (0.0.0.0/0)
- : Type: Custom TCP, Protocol: TCP, Port range: 3200, Source: Anywhere (0.0.0.0/0)

A new rule is being added at the bottom:

- Type: Custom TCP, Protocol: TCP, Port range: 3200, Source: Anywhere (0.0.0.0/0)

An 'Add rule' button is visible at the bottom left.

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](#)