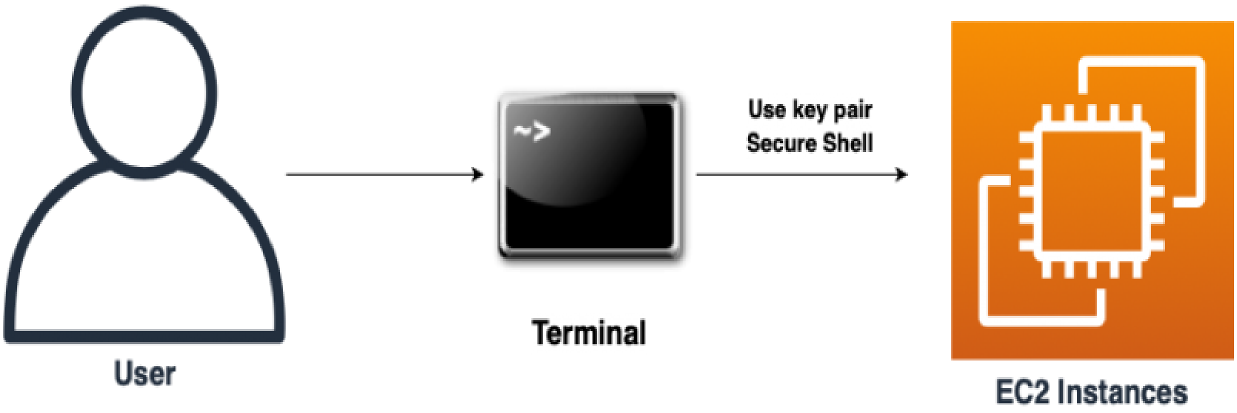
The Mini-Project Report on

“Build a Virtual Server With EC2”



Github Repository Link :

https://github.com/KPranit-2105/AWS-Projects

By :

Pranit P. Kolamkar

Under the Guidance of

Prof. Shikha Dwivedi

Abstract

The project "Build a Virtual Server with EC2" focuses on leveraging Amazon Web Services (AWS) Elastic Compute Cloud (EC2) to create and manage a scalable, on-demand virtual server environment. EC2 provides flexible compute capacity in the cloud, enabling users to launch virtual machines (instances) with varied configurations, depending on their needs. This project explores the fundamental process of setting up a virtual server, including selecting instance types, configuring security groups, managing key pairs for SSH access, and installing necessary software for a functional server environment. Additionally, it covers the steps to monitor and maintain the instance, ensuring optimal performance and cost-efficiency. By the end of this project, users will gain practical experience in cloud infrastructure management, automation, and security best practices while understanding the benefits and challenges of using cloud-based solutions for deploying virtual servers.

Introduction

In today’s rapidly evolving digital landscape, cloud computing has emerged as a cornerstone for businesses and developers seeking scalable, cost-effective, and efficient computing resources. Among the leading cloud service providers, Amazon Web Services (AWS) stands out with its robust offerings, one of the most notable being the Elastic Compute Cloud (EC2). EC2 allows users to provision virtual servers, or instances, that can be tailored to meet specific computational needs, offering flexibility in terms of size, performance, and cost.

The project "Build a Virtual Server with EC2" provides a hands-on approach to understanding and utilizing EC2 for cloud infrastructure management. By following a step-by-step guide, this project aims to walk users through the process of creating a virtual server, configuring it for optimal performance, and understanding the different EC2 instance types and configurations. Additionally, it emphasizes essential practices such as securing the instance, ensuring network connectivity, and monitoring the server to maintain operational efficiency.

Through this project, users will gain a deeper understanding of cloud computing principles, AWS tools, and best practices for creating and managing virtual servers, equipping them with the skills needed to deploy production-level applications or services on the cloud.

Problem Statement :

As businesses and developers increasingly transition to cloud environments for their computing needs, managing scalable and efficient virtual server infrastructure has become a critical challenge. While cloud platforms like Amazon Web Services (AWS) offer powerful solutions, many users struggle with the complexities of provisioning, configuring, and managing virtual servers effectively. Specifically, there is a gap in understanding how to:

1)Select appropriate EC2 instance types and configurations based on specific requirements.

2)Secure and maintain the virtual server environment to ensure data integrity and protection.

3)Optimize performance and minimize costs while running virtual servers on the cloud.

This project aims to address these challenges by providing a comprehensive guide to building and managing a virtual server using AWS EC2. The goal is to empower users with the skills and knowledge needed to deploy, configure, secure, and monitor virtual servers, ensuring both operational efficiency and cost-effectiveness in a cloud-based infrastructure.

Objectives :

The primary objective of the project "Build a Virtual Server with EC2" is to provide users with the knowledge and skills necessary to efficiently create, configure, and manage a virtual server using Amazon Web Services (AWS) EC2. Specifically, the objectives are to:

Set up a Virtual Server: Guide users through the process of launching an EC2 instance, selecting the appropriate instance type, and configuring the server environment to meet specific project needs.

Implement Security Best Practices: Ensure the server is secured by configuring security groups, managing SSH key pairs, and applying essential security measures for safe access and operation.

Optimize Performance and Costs: Demonstrate how to monitor server performance, scale instances as needed, and utilize AWS tools for cost optimization.

Ensure Server Management and Maintenance: Educate users on maintaining and monitoring the virtual server, handling updates, and troubleshooting potential issues to ensure smooth operation over time.

By the end of the project, users will have a comprehensive understanding of AWS EC2, equipped with practical experience in deploying and managing virtual
server environments in the cloud.

.

Scope :

The scope of the project "Build a Virtual Server with EC2" is to provide a structured approach to deploying and managing a virtual server using Amazon Web Services (AWS) EC2. The project will cover the following key areas:

1) Instance Setup and Configuration:

Guide users through the process of selecting and launching EC2 instances with various configurations based on requirements (e.g., instance type, region, operating system).
Provide instructions on configuring basic server settings, such as network interfaces, storage volumes, and security groups.

2) Security and Access Management:

Demonstrate how to configure security groups to control inbound and outbound traffic to the virtual server.
Teach users how to generate and manage SSH key pairs for secure remote access to the instance.

3) Performance Optimization and Cost Management:

Introduce tools for monitoring server performance, including CPU usage, memory, and disk activity.
Offer guidance on selecting the appropriate EC2 instance type for different workloads and scaling resources as necessary.
Discuss methods for reducing operational costs, such as using AWS cost management tools and implementing auto-scaling features.

4) Maintenance and Monitoring:

Provide instructions on setting up logging, system monitoring, and backup mechanisms to ensure the health and availability of the server.
Explore troubleshooting techniques and best practices for resolving common issues related to EC2 instances.

5) Exclusions:

This project will not cover advanced topics such as managing complex cloud architectures, containerization with ECS or EKS, or advanced networking configurations like VPC design and peering.
It will not delve into application-level configurations or software deployment but will focus on the foundational setup and management of EC2 instances.
By the end of this project, users will be equipped to deploy, configure, secure, and manage a basic virtual server on AWS EC2, with a focus on practical skills for real-world cloud infrastructure tasks.

Software Requirements and Specifications :

Hardware Requirements:

-CPU: Minimum 2 cores, ideally 4+ cores for better performance.
-RAM: Minimum 8 GB of RAM, depending on the selected EC2 instance type.
-Storage: Minimum of 20 GB of disk space (elastic block storage or instance storage as required).
-Network: Internet access for communication with AWS services.
3.2 Software Requirements:
-Operating System: Any modern operating system that supports AWS CLI (Windows, macOS, Linux).
-Web Browser: A supported browser for accessing AWS Management Console (Google Chrome, Mozilla Firefox, Safari, or Microsoft Edge).
-AWS Account: An active AWS account to create EC2 instances and use AWS services.
-AWS CLI (Optional): For users comfortable with command-line interfaces, the AWS CLI can be used to manage EC2 instances. The CLI must be installed on the user's local machine.
-SSH Client: For remote access to EC2 instances (PuTTY for Windows, OpenSSH for macOS/Linux).
-Text Editor (Optional): For configuration files and scripting (e.g., Visual Studio Code, Sublime Text, or Vim).

Functional Requirements

The project will involve the following functionalities:

1)EC2 Instance Creation:
-Users should be able to launch EC2 instances from the AWS Management Console.
-The system should allow users to select from a variety of EC2 instance types (e.g., T2, M5, C5).
-The system should allow users to choose the appropriate operating system (e.g., Ubuntu, Amazon Linux 2, Windows Server).
-The system should provide options to configure instance details such as VPC, subnet, storage, and tags.
2)Security Configuration:
-Users should be able to configure security groups to control inbound and outbound traffic for the EC2 instance.
-Users should be able to create and download SSH key pairs for secure remote access to the EC2 instance.

-The system should guide users to ensure best security practices, such as disabling root login, configuring firewalls, and managing user access.

3)Server Performance and Monitoring:

-Users should be able to monitor the EC2 instance performance using AWS CloudWatch (metrics such as CPU usage, memory, and disk space).

-The system should allow users to implement Auto Scaling to automatically adjust resources based on workload demands.

4)Cost Management:

-Users should be able to monitor instance costs using AWS Cost Explorer.

-The system should recommend the most cost-effective instance types based on usage patterns.

5)Maintenance and Troubleshooting:

-The system should include instructions for maintaining the EC2 instance, such as applying software updates and patches.

-The system should provide guidance on troubleshooting common issues such as instance health checks and connectivity problems.

Non-Functional Requirements

1)Performance:

-The EC2 instance should be provisioned with minimal delay, typically within minutes, depending on the instance type and region.Server performance monitoring should provide real-time metrics with minimal lag.

2)Security:

-All communications with EC2 instances should be encrypted using SSH for remote access.

-AWS resources should be secured using IAM (Identity and Access Management) roles, ensuring appropriate access controls for different users.

-Data in transit (e.g., between the EC2 instance and other AWS services) should be encrypted using HTTPS.

3)Usability:

-The process of launching and managing the EC2 instance should be easy to follow, with detailed step-by-step instructions provided to users.

-Clear instructions for configuring security and monitoring server performance should be accessible to users with varying levels of expertise.

4)Scalability:

-The system should allow users to scale the virtual server infrastructure (both horizontally and vertically) as the project grows in complexity.

-Auto Scaling should be supported to ensure that resources are allocated efficiently based on workload demands.

5)Reliability:

-The system should ensure high availability for the EC2 instances through proper configurations (e.g., load balancing, multi-Availability Zone deployment).

-Backup solutions should be in place to ensure data integrity and disaster recovery.

System Interfaces

1)User Interface:

-The AWS Management Console will serve as the primary interface for creating and managing EC2 instances.

-The user interface will include dashboards for monitoring performance, security settings, and cost management.

2)AWS CLI (Optional):

-The AWS CLI can be used for command-line-based interactions with AWS EC2.

-Users can run commands to launch instances, configure security, and monitor performance.

3)SSH Interface:

-SSH will be used for secure remote access to EC2 instances. Users will interact with the EC2 instance's operating system through an SSH client.

4)Limitations and Constraints

-The project will focus primarily on basic EC2 instance setup and management. More advanced topics like VPC design, advanced networking, or containerization are out of scope.

-This project does not cover on-premises server management or deployment outside of AWS.

-AWS service costs are not included in the scope, and users will need to manage their billing through their AWS accounts.

Project Plan :

Software Development Life-cycle :

---

1. Planning and Requirement Analysis

Objectives:

Define the purpose of the EC2 server.

Identify stakeholders and their requirements.

Analyze the costs associated with the EC2 instance types and other AWS services.

Deliverables:

Project Scope Document.

Requirements list (e.g., operating system, storage, security, instance type, scalability needs).

Budget estimation and approval.

---

2. System Design

Tasks:

Choose the EC2 instance type (e.g., t2.micro, t3.large) based on application and workload.

Plan for networking: Select the appropriate VPC (Virtual Private Cloud), subnets, and security groups.

Design storage: Choose between EBS (Elastic Block Store) or Instance Store.

Plan for scalability using Auto Scaling Groups and Elastic Load Balancer.

Define security measures: Use IAM roles, key pairs, and security groups.

Deliverables:

Architecture Diagram.

Design Document (including instance configuration, network setup, and security policies).

---

3. Implementation (Coding and Configuration)

Tasks:

-Set up the environment:

Configure AWS CLI (Command Line Interface) for deployment.

Set up IAM roles and permissions.

-Provision the EC2 Instance:

Use the AWS Management Console or Terraform/CloudFormation (for Infrastructure as Code).

Configure the instance (OS, user data scripts, etc.).

-Networking and Security:

Configure VPC, subnets, and route tables.

Set up security groups and firewalls.

-Application Deployment:

Install necessary software (e.g., web server, database, etc.).

Test connectivity to the instance via SSH or a web interface.

Deliverables:

Configured EC2 instance.

IAM role and security group configurations.

Deployment scripts (if applicable).

---

4. Testing

Tasks:

-Functional Testing:

Verify instance accessibility via SSH.

Test application installation and performance.

-Security Testing:

Ensure only authorized users can access the instance.

Test for security group rule enforcement.

-Performance Testing:

Verify instance performance under expected workloads.

Test scalability with Auto Scaling.

-Deliverables:

Test Plan Document.

Test Results and Bug Reports.

---

5. Deployment

Tasks:

-Deploy the EC2 instance in the production environment.

-Configure monitoring with CloudWatch to track performance and availability.

-Enable automated backups (e.g., EBS snapshots).

Deliverables:

Deployed EC2 instance in production.

Monitoring and backup configuration.

---

6. Maintenance

Tasks:

-Regularly update the instance (OS patches, software updates).

-Monitor usage and scale up/down as needed.

-Periodically review and optimize costs.

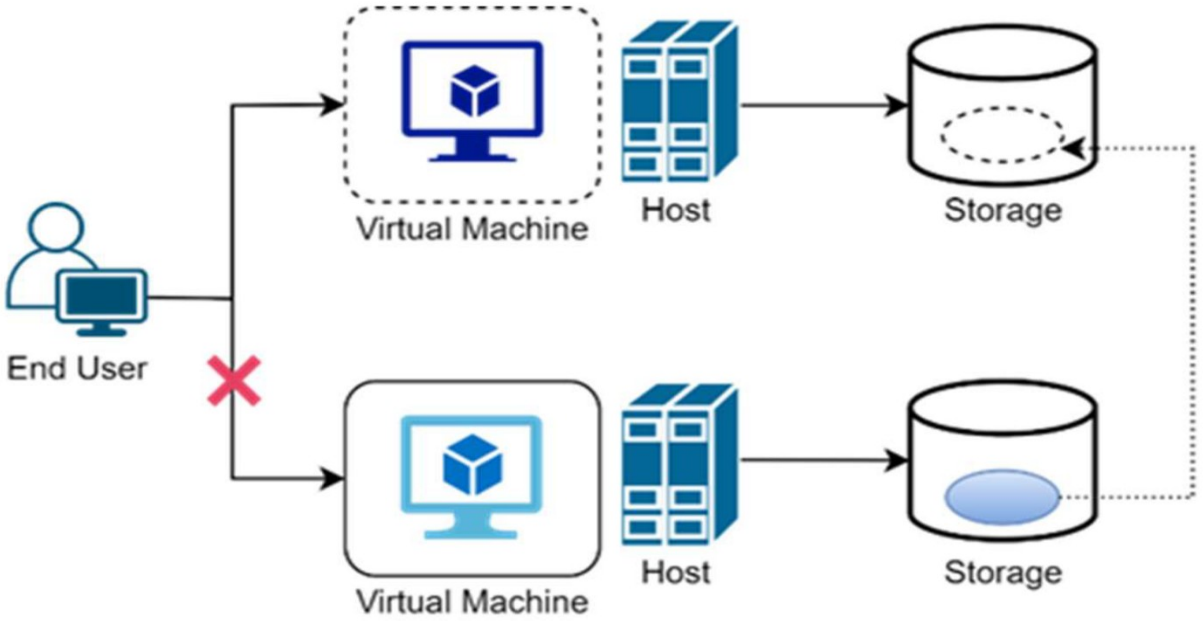
-Perform routine security audits and compliance checks.

Deliverables:

Maintenance Logs.

Cost Optimization Reports.

Architecture Diagram :



The architecture for building a virtual server with Amazon EC2 involves key layers. Users access the server via SSH or HTTP/HTTPS. The networking layer is built within a VPC, with public subnets for internet-facing resources and private subnets for secure components. Connectivity is managed through an Internet Gateway or NAT Gateway, while security groups control traffic.

The compute layer hosts the EC2 instance, configured with the required OS and instance type. For high availability, an Elastic Load Balancer distributes traffic across multiple instances. Storage is provided by EBS for persistent data and S3 for backups and logs.

Security is enforced using IAM roles, key pairs, and security group rules. Monitoring is handled by CloudWatch for performance metrics and CloudTrail for audit logs. Scalability is achieved through Auto Scaling Groups and load balancing, ensuring the server is reliable, secure, and adaptable to workload changes.

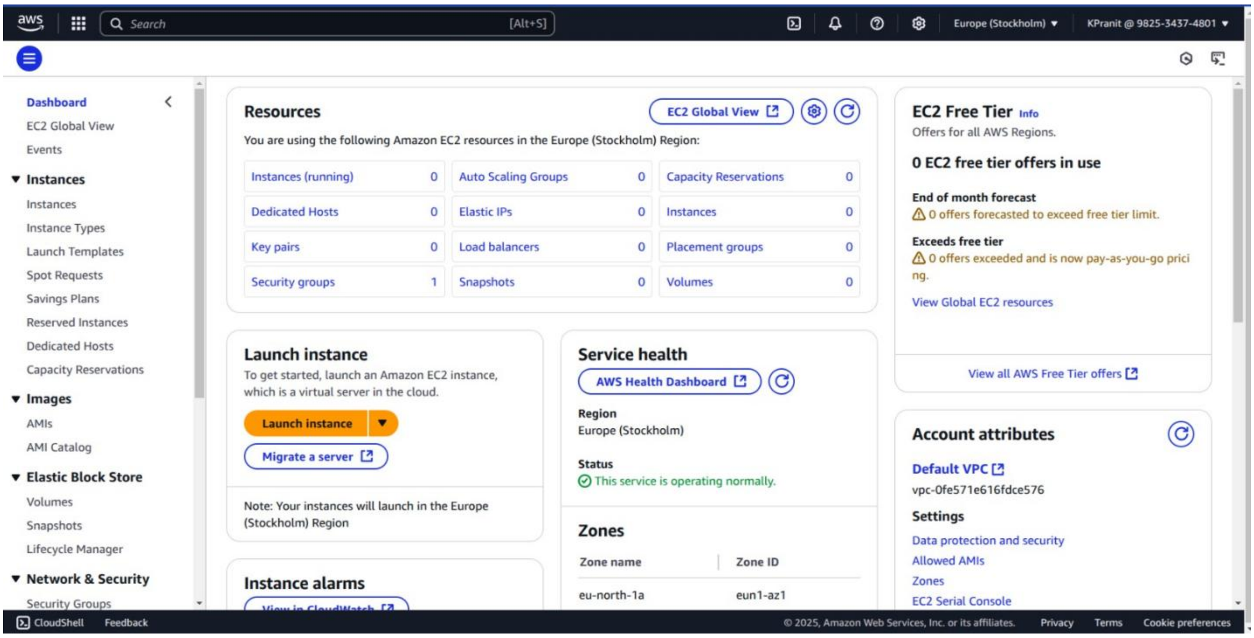
Result :

The result of implementing the architecture for building a virtual server with Amazon EC2 is a scalable, secure, and reliable infrastructure. The server is accessible to authorized users via SSH or HTTP/HTTPS, with controlled traffic through security groups. Resources are organized within a Virtual Private Cloud (VPC), ensuring isolation and efficient networking.

The EC2 instance operates with the required configuration and storage, supported by Elastic Block Store (EBS) for persistent data and Amazon S3 for backups. The system is monitored using CloudWatch for performance metrics and CloudTrail for audit logs. If configured, the Elastic Load Balancer and Auto Scaling ensure high availability and automatic scaling to handle workload changes effectively. This setup provides a robust environment for hosting applications or workloads while maintaining cost efficiency and security.

Screenshots :

Step 01 : Go to the Amazon EC2 and launch the instance.



Resources

You are using the following Amazon EC2 resources in the Europe (Stockholm) Region:

Instances (running)	0	Auto Scaling Groups	0	Capacity Reservations	0
Dedicated Hosts	0	Elastic IPs	0	Instances	0
Key pairs	0	Load balancers	0	Placement groups	0
Security groups	1	Snapshots	0	Volumes	0

EC2 Free Tier Info

Offers for all AWS Regions.

0 EC2 free tier offers in use

End of month forecast

⚠ 0 offers forecasted to exceed free tier limit.

Exceeds free tier

⚠ 0 offers exceeded and is now pay-as-you-go pricing.

View Global EC2 resources

View all AWS Free Tier offers

Launch instance

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

Launch instance ▾

Migrate a server

Note: Your instances will launch in the Europe (Stockholm) Region

Service health

AWS Health Dashboard

Region: Europe (Stockholm)

Status: ✓ This service is operating normally.

Zones

Zone name	Zone ID
eu-north-1a	eun1-az1

Instance alarms

View in CloudWatch

Account attributes

Default VPC

vpc-0fe571e616fdce576

Settings

Data protection and security

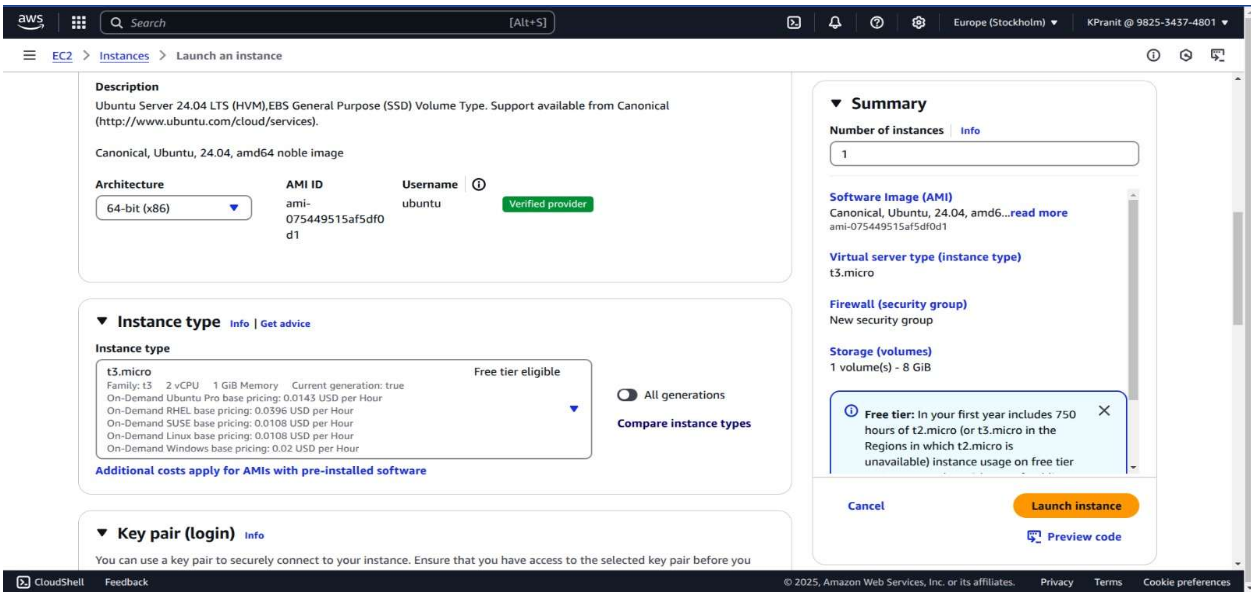
Allowed AMIs

Zones

EC2 Serial Console

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

Step 02 : Go to the Launch section in the instance.


aws
Search [Alt+S]
Europe (Stockholm) ▾
KPranit @ 9825-3437-4801 ▾
EC2 > Instances > Launch an instance
CloudShell
Feedback

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	Username	
64-bit (x86) ▾	ami-075449515af5df0d1	ubuntu	Verified provider

▾ Instance type Info | Get advice

Instance type

t3.micro Free tier eligible

Family: t3 2 vCPU 1 GiB Memory Current generation: true

On-Demand Ubuntu Pro base pricing: 0.0143 USD per Hour

On-Demand RHEL base pricing: 0.0396 USD per Hour

On-Demand SUSE base pricing: 0.0108 USD per Hour

On-Demand Linux base pricing: 0.0108 USD per Hour

On-Demand Windows base pricing: 0.02 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

▾ Key pair (login) Info

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

▾ Summary

Number of instances Info: 1

Software Image (AMI): Canonical, Ubuntu, 24.04, amd6...read more ami-075449515af5df0d1

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 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier

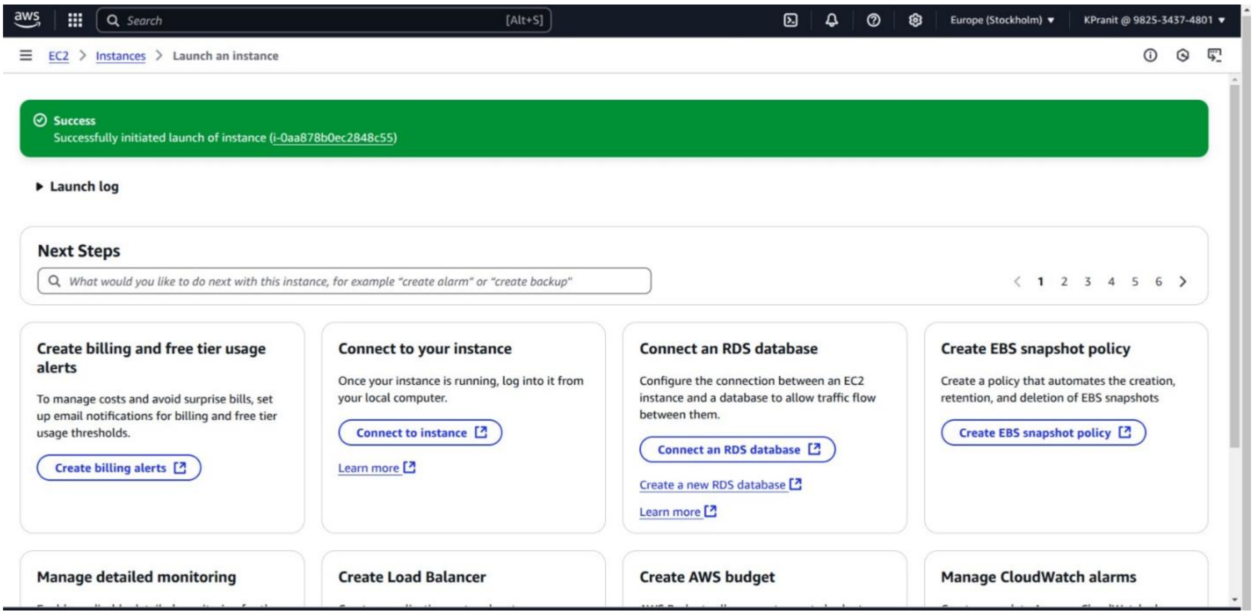
Cancel

Launch instance

Preview code

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

Step 03 : EC2 instance successfully created.


aws
Search [Alt+S]
Europe (Stockholm) ▾
KPranit @ 9825-3437-4801 ▾
EC2 > Instances > Launch an instance

Success
Successfully initiated launch of instance (i-0aa878b0ec2848c55)

▸ Launch log

Next Steps

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 automates the creation, retention, and deletion of EBS snapshots

Create EBS snapshot policy

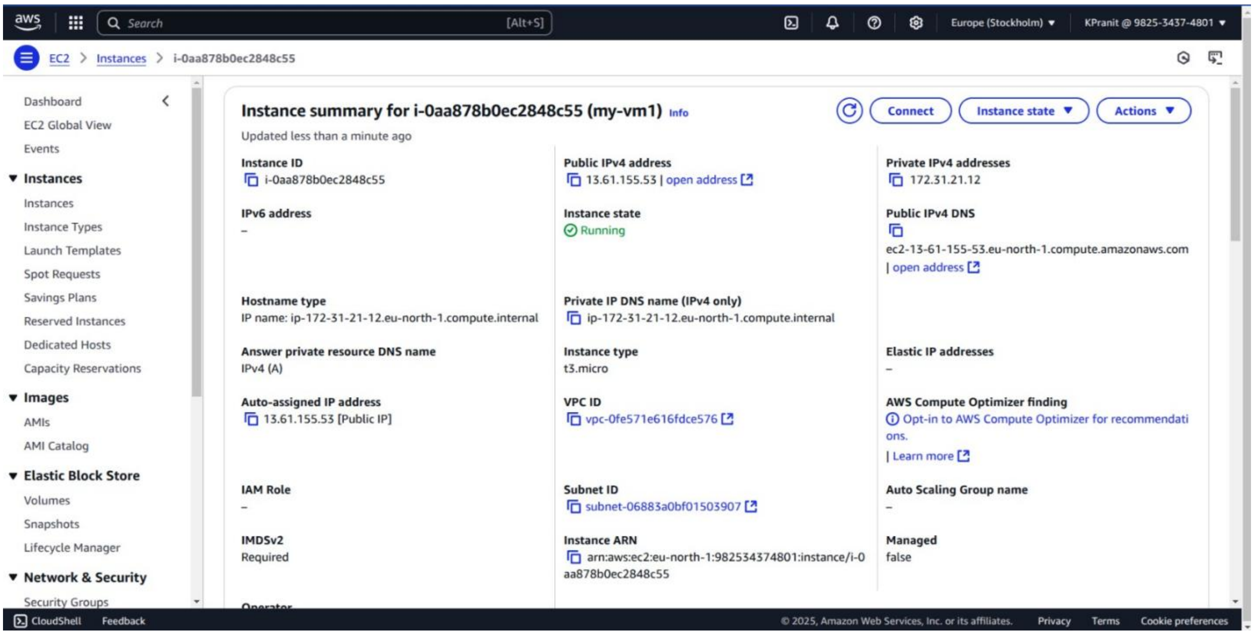
Manage detailed monitoring

Create Load Balancer

Create AWS budget

Manage CloudWatch alarms

Step 04 : Open the Terminal and copy the link in it.

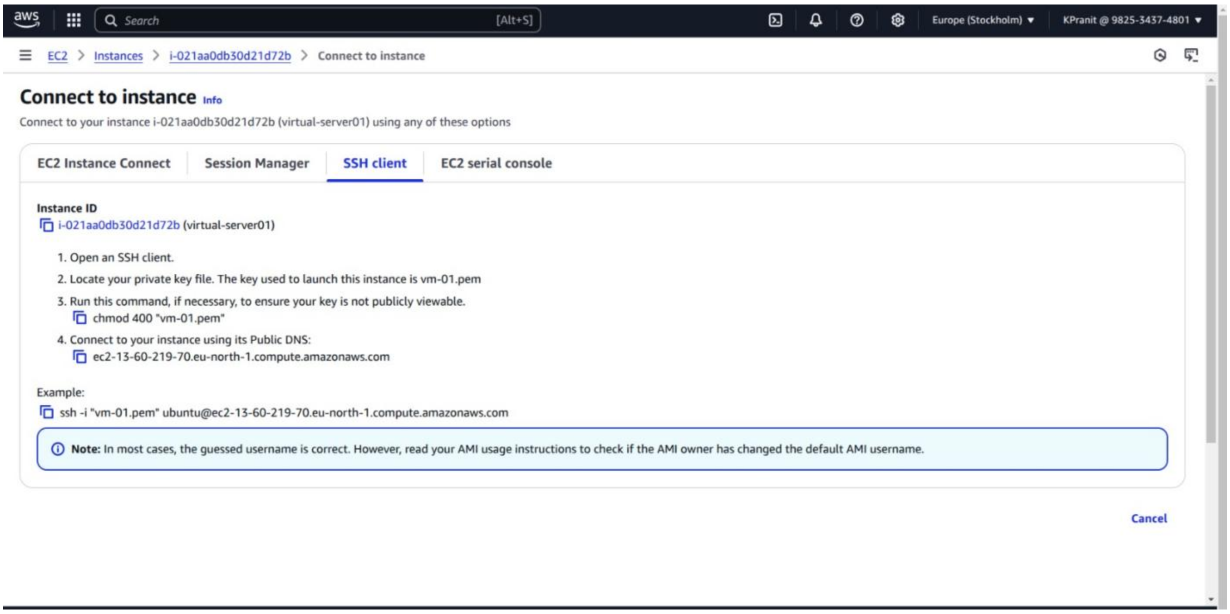


Instance summary for i-0aa878b0ec2848c55 (my-vm1) Info

Updated less than a minute ago

Instance ID i-0aa878b0ec2848c55	**Public IPv4 address** 13.61.155.53 | open address	**Private IPv4 addresses** 172.31.21.12
IPv6 address –	**Instance state** ✓ Running	**Public IPv4 DNS** ec2-13-61-155-53.eu-north-1.compute.amazonaws.com | open address
Hostname type IP name: ip-172-31-21-12.eu-north-1.compute.internal	**Private IP DNS name (IPv4 only)** ip-172-31-21-12.eu-north-1.compute.internal	
Answer private resource DNS name IPv4 (A)	**Instance type** t3.micro	**Elastic IP addresses** –
Auto-assigned IP address 13.61.155.53 [Public IP]	**VPC ID** vpc-0fe571e616fdce576	**AWS Compute Optimizer finding** Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role –	**Subnet ID** subnet-06883a0bf01503907	**Auto Scaling Group name** –
IMDSv2 Required	**Instance ARN** arn:aws:ec2:eu-north-1:982534374801:instance/i-0aa878b0ec2848c55	**Managed** false
Operator		

Step 05 : Connect to the instance screenshot.



aws
Search
[Alt+S]
Europe (Stockholm) ▾
KPranit @ 9825-3437-4801 ▾
EC2 > Instances > i-021aa0db30d21d72b > Connect to instance

Connect to instance

Connect to your instance i-021aa0db30d21d72b (virtual-server01) using any of these options

Instance ID

i-021aa0db30d21d72b (virtual-server01)

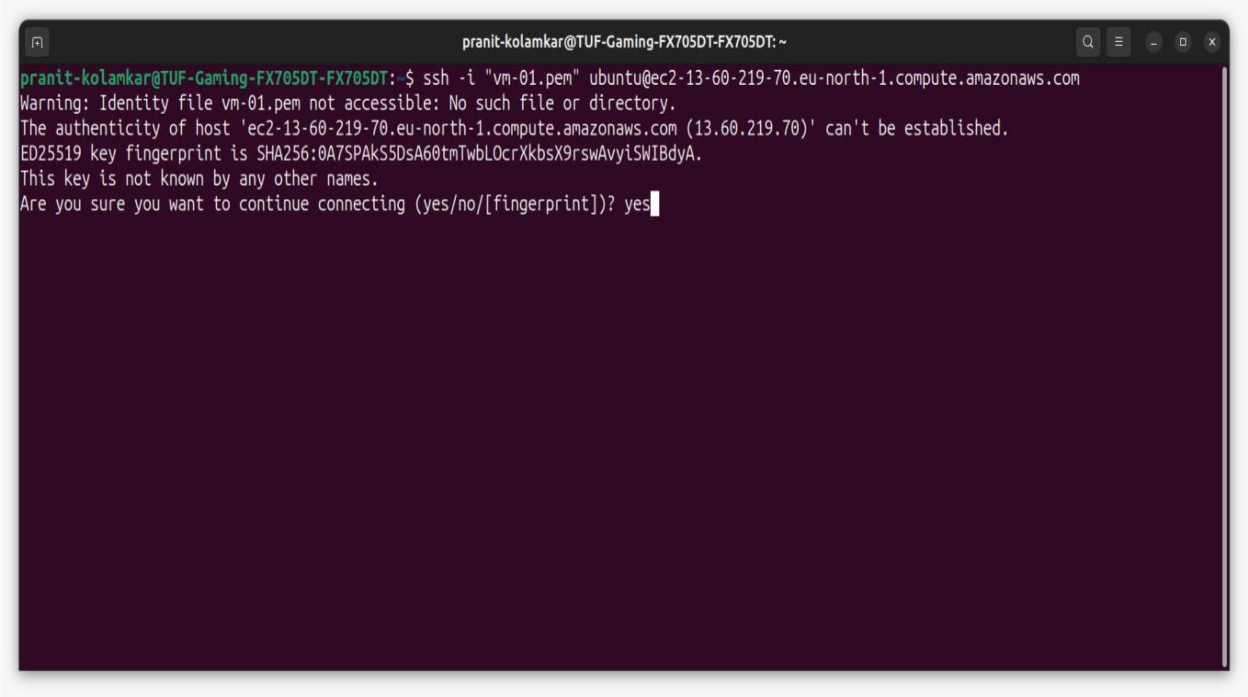
1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is vm-01.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 "vm-01.pem"
4. Connect to your instance using its Public DNS:
   ec2-13-60-219-70.eu-north-1.compute.amazonaws.com

Example:

```
ssh -i "vm-01.pem" ubuntu@ec2-13-60-219-70.eu-north-1.compute.amazonaws.com
```

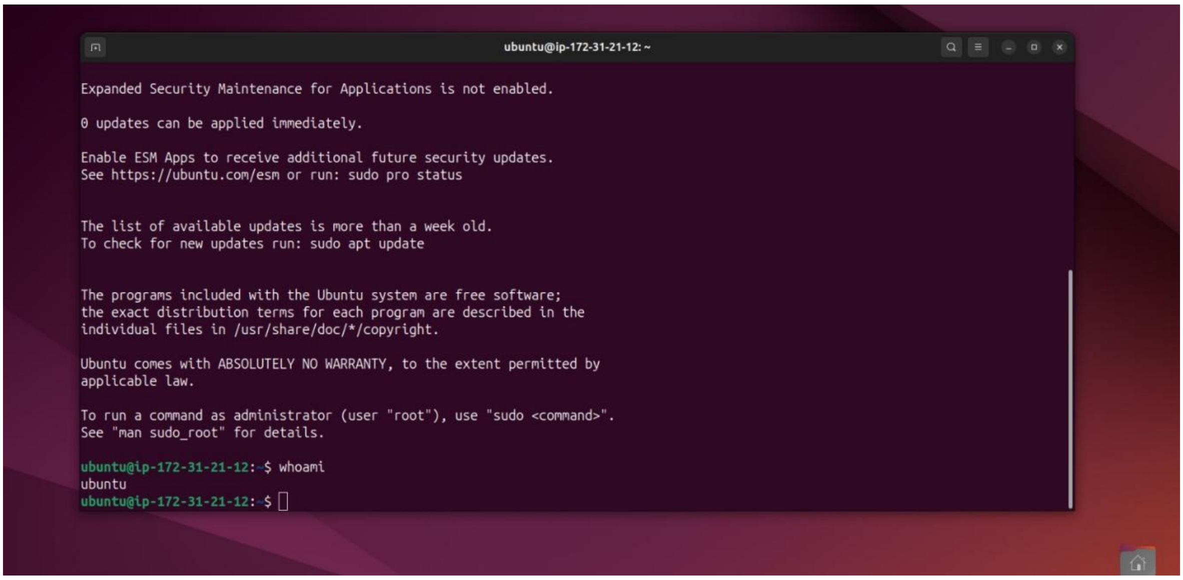
ⓘ **Note:** In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Step 06 : Provide the system password and finally we connect the instance to our local system.



```
pranit-kolamkar@TUF-Gaming-FX705DT-FX705DT:~$ ssh -i "vm-01.pem" ubuntu@ec2-13-60-219-70.eu-north-1.compute.amazonaws.com
Warning: Identity file vm-01.pem not accessible: No such file or directory.
The authenticity of host 'ec2-13-60-219-70.eu-north-1.compute.amazonaws.com (13.60.219.70)' can't be established.
ED25519 key fingerprint is SHA256:0A7SPAkS5DsA60tmTwbLOcrXkbsX9rswAvyiSWIBdyA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

Step 07 :By providing the user-password the instance is successfully connected to local machine.



```
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-172-31-21-12:~$ whoami
ubuntu
ubuntu@ip-172-31-21-12:~$ 
```

Conclusion :

The "Build the Virtual Server on EC2" project successfully demonstrated the setup and configuration of an EC2 instance on AWS. Key tasks included selecting the appropriate AMI, configuring instance types, setting up security groups, and establishing secure SSH access. This project provided hands-on experience with AWS EC2, highlighting scalability, cost optimization, and security. It serves as a solid foundation for implementing advanced cloud solutions.