PROFESSIONAL TRAINING REPORT

at

Sathyabama Institute of Science and Technology

(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering

By

CHENNA ANVESH KUMAR (Regd No 40110255)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600119, TAMILNADU

OCT 2022

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119

www.sathyabamauniversity.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **CHENNA ANVESH KUMAR (40110255**) who carried out the project entitled **"Working with RDS**" under my supervision from Aug 2022 to Oct 2022.

Internal Guide

Dr.V. Ulagamuthalvi M.E., Ph. D,

Head of the Department

Dr.L. Lakshmanan M.E., Ph. D,

Submitted for Viva voce Examination held on _____

Internal Examiner **External Examiner**

DECLARATION

I **CHENNA ANVESH KUMAR** hereby declare that the Project Reportentitled **"Working with RDS"** done by me under the guidance of **Dr.V. Ulagamuthalvi** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

SIGNATURE OF THECANDIDATE

DATE:

PLACE:

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D.**, **Dean**, School of Computing,

Dr.S. Vigneshwari M.E., Ph.D., and Dr.L. Lakshmanan M.E., Ph.D., Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.V.Ulagamuthalvi** for his valuable guidance, suggestions andconstantencouragementpavedwayforthesuccessfulcompletionofmyprojectwork.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **DepartmentofComputerScienceandEngineering**whowerehelpfulinmanywaysfor the completion of the project


STAR
CLOUD
COMPUTING
STAR
CERTIFICATION
Award of
Completion
CHENNA ANVESH KUMAR
Has Successfully Completed
Star Cloud Computing
Star Authorized Delivery Partner
Advantage Pro
26-10-2022
DATE
RSCZQCHAAQ
CERTIFICATE NUMBER
DONALD WILLIAM
DIRECTOR
GLOBAL CERTIFICATIONS
You can verify authenticity of this certificate by visiting
http://www.starcertification.org/Verification/Participation
STAR
CERTIFICATION

ABSTRACT

A relational database is a collection of data items with pre-defined relationships between them. These items are organized as a set of tables with columns and rows. Tables are used to hold information about the objects to be represented in the database. Amazon Aurora is a MySQL and PostgreSQL compatible relational database engine that combines the speed and availability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases. Amazon Relational database Service (Amazon RDS) is a collection of managed services that makes it simple to set up, operate, and scale databases in the cloud. Amazon RDS allows you to encrypt your databases using keys you manage through AWS Key Management Service (KMS). On a database instance running with Amazon RDS encryption, data stored at rest in the underlying storage is encrypted, as are its automated backups, read replicas, and snapshots. Amazon RDS DB snapshots and automated backups are stored in S3. You can use the AWS Management Console, the Modify DBInstance API, or the modify-db-instance command to manage the period of time your automated backups are retained by modifying the Retention period parameter

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

ABBREVATION	EXPANSION
Amazon RDS	Amazon Relational Database Service
EC2	Elastic Cloud Computing
AWS	Amazon Web Services
API	Application Programming Interface
MFA	Multi Factor Authentication
RDS	Remote Desktop Services
KMS	Key Management Services

CHAPTER 1

INTRODUCTION

1.1 Cloud Computing:

Cloud computing is a computing technique in which all the data is stored in the cloud. All the SDKs and databases are stored in the cloud only. A Cloud is primarily a platform that allow execution in various forms across multiple resources and potentially across enterprise boundaries. A Cloud is an infrastructure that enables execution of codes (applications, services, etc.) in a managed and elastic fashion, here the meaning of managed is reliability according to pre-defined quality parameters which are automatically ensured and elastic here means that the resources are used according to the current actual requirements. Cloud fits the client-server model, and so far, the typical cloud client is the same as the typical enterprise client (i.e., single desktop or laptop computer), some observers have a tendency to stop at this level of analysis. But of course, the real action in the cloud happens on the server side and that's where things get interesting.

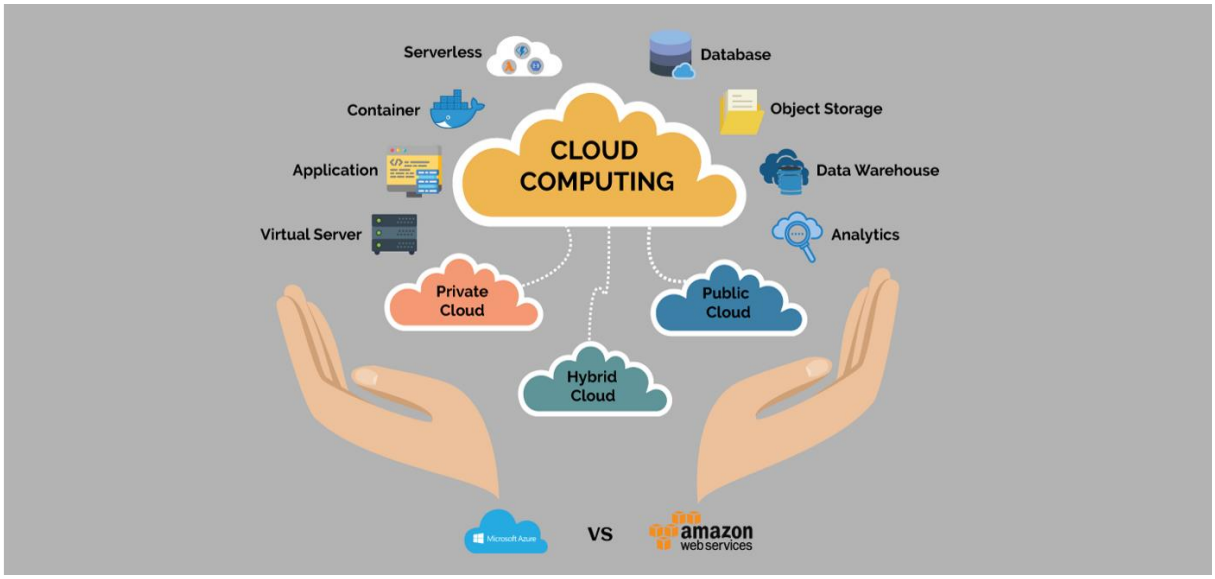


Fig 1.1 Cloud Computing

1.2 Amazon RDS:

- Amazon RDS allows you to encrypt your databases using keys you manage through AWS Key Management Service (KMS). On a database instance running with Amazon RDS encryption, data stored at rest in the underlying storage is encrypted, as are its automated backups, read replicas, and snapshot.

- Amazon Relational Database Service (Amazon RDS) is a collection of managed services that makes it simple to set up, operate, and scale databases in the cloud.

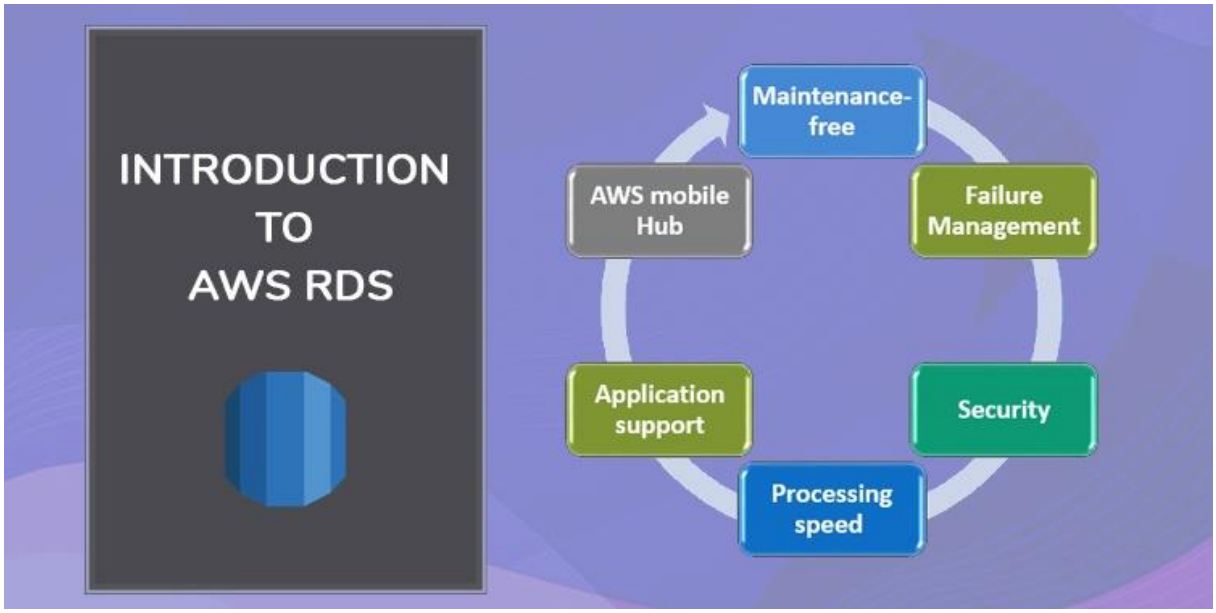


Fig 1.2 Introduction of amazon rds

1.3 Amazon RDS Features

- Amazon RDS is a managed relational database service that provides you six familiar database engines to choose from, including Amazon Aurora, MYSQL, Maria DB, PostgreSQL, Oracle and Microsoft SQL Server. This means that the code, applications, and tools you already use today with your existing databases can be used with Amazon RDS. Amazon RDS handles routine database tasks, such as provisioning, patching, backup, recovery, failure detection, and repair

- Amazon RDS makes it easy to use replication to enhance availability and reliability for production workloads. Using the Multi-AZ deployment option, you can run mission-critical workloads with high availability and built-in automated failover from your primary database to a synchronously replicated secondary database. Using Read Replicas, you can scale out beyond the capacity of a single database deployment for read-heavy database workloads.

- As with all Amazon Web Services, there are no up-front investments required and you pay only for the resources you use

i. Lower administrative burden
ii. Performance
iii. Scalability
iv. Availability and durability
v. Security
vi. Manageability
vii. Cost-effectiveness

HOW IT WORKS

- Amazon Relational Database Service (Amazon RDS) is a collection of managed services that makes it simple to set up, operate, and scale databases in the cloud. Choose from seven popular engines — Amazon Aurora with MySQL compatibility, Amazon Aurora with PostgreSQL compatibility, MySQL, MariaDB, PostgreSQL, Oracle, and SQL Server — and deploy on-premises with Amazon RDS on AWS Outposts.

AMAZON RDS

Fig 1.3 Amazon rds features

AMAZON RDS CUSTOM

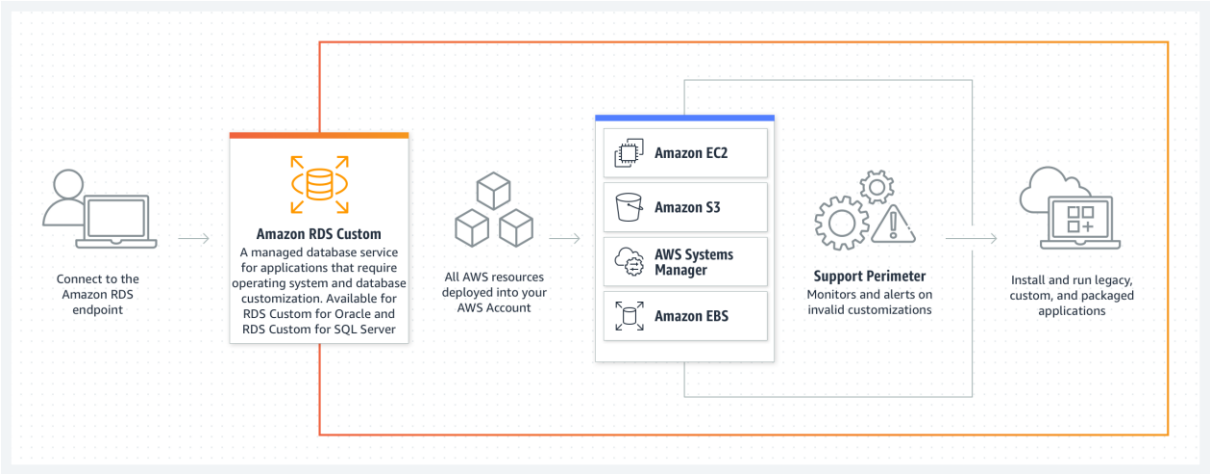
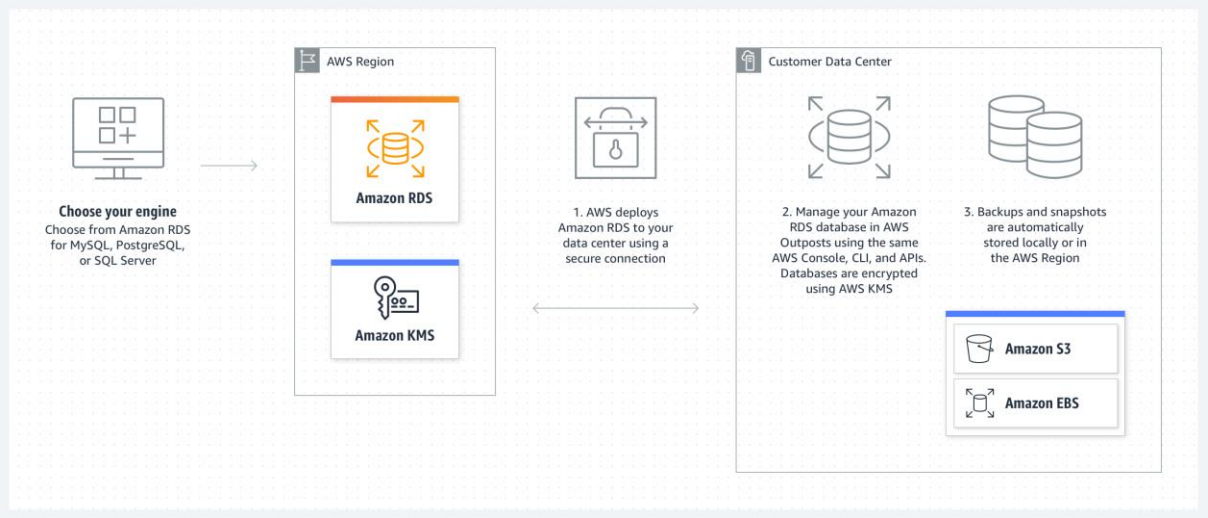


Fig 1.4 Amazon rds resources deployment

AMAZON RDS ON AWS OUTPUTS

- You use the same AWS Management Console, AWS CLI, and RDS API to provision and manage on-premises RDS on Outposts DB instances as you do for RDS DB instances running in the AWS Cloud. RDS on Outposts automates tasks, such as database provisioning, operating system and database patching, backup, and long-term archival in Amazon S3.

- RDS on Outposts supports automated backups of DB instances. Network connectivity between your Outpost and your AWS Region is required to back up and restore DB instances. All DB snapshots and transaction logs from an Outpost are stored in your AWS Region. From your AWS Region, you can restore a DB instance from a DB snapshot to a different Outpost



.

Fig 1.5 Amazon rds engine

CHAPTER-2

2.1 AIM:

- Sign for an AWS account
- Create an administrative user
- Create IAM user access key
- Determine requirements
- Provide access to your DB instance in your VPC by creating a security group
- Creating a MYSQL DB instance and connecting to database

2.2 OBJECTIVE:

- About cloud computing
- Rds in cloud computing
- Features of MySQL
- Versions of MySQL
- Upgrading the MySQL db engine
- Upgrading the MySQL db snapshot

2.3 THE FEATURES OF MYSQL

- RDS for MySQL supports most of the features and capabilities of MySQL. Some features might have limited support or restricted privileges.

  Supported storage engines for RDS for MySQL

  While MySQL supports multiple storage engines with varying capabilities, not all of them are optimized for recovery and data durability. Amazon RDS fully supports the InnoDB storage engine for MySQL DB instances. Amazon RDS features such as Point-In-Time restore and snapshot restore require a recoverable storage engine and are supported for the InnoDB storage engine only

  The Federated Storage Engine is currently not supported by Amazon RDS for MySQL

  For user-created schemas, the MyISAM storage engine does not support reliable

recovery and can result in lost or corrupt data when MySQL is restarted after a recovery, preventing Point-In-Time restore or snapshot restore from working as intended. However, if you still choose to use MyISAM with Amazon RDS, snapshots can be helpful under some conditions.

System tables in the mysql schema can be in MyISAM storage

2.4 VERSIONS OF MYSQL

For MySQL, version numbers are organized as version = X.Y.Z. In Amazon RDS terminology, X.Y denotes the major version, and Z is the minor version number. For Amazon RDS implementations, a version change is considered major if the major version number changes—for example, going from version 5.7 to 8.0. A version change is considered minor if only the minor version number changes—for example, going from version 8.0.27 to 8.0.30.

Supported MySQL versions on Amazon RDS

Amazon RDS currently supports the following versions of MySQL:

You can specify any currently supported MySQL version when creating a new DB instance. You can specify the major version (such as MySQL 5.7), and any supported minor version for the specified major version. If no version is specified, Amazon RDS defaults to a supported version, typically the most recent version. If a major version is specified but a minor version is not, Amazon RDS defaults to a recent release of the major version you have specified. To see a list of supported versions, as well as defaults for newly created DB instances, use the describe-db-engine-versions AWS CLI command.

Deprecated versions for Amazon RDS for MySQL

Amazon RDS for MySQL version 5.1, 5.5, and 5.6 are deprecated.

2.5 UPGRADING THE MYSQL DB ENGINE

When Amazon RDS supports a new version of a database engine, you can upgrade your DB instances to the new version. There are two kinds of upgrades for MySQL DB instances: major version upgrades and minor version upgrades.

Major version upgrades can contain database changes that are not backward-compatible with existing applications. As a result, you must manually perform major version upgrades of your DB instances. You can initiate a major version upgrade by modifying your DB instance. However, before you perform a major version upgrade, we recommend that you follow the instructions in Major version upgrades for MySQL.

In contrast, minor version upgrades include only changes that are backward-compatible with existing applications. You can initiate a minor version upgrade manually by modifying your DB instance. Or you can enable the Auto minor version upgrade option when creating or modifying a DB instance. Doing so means that your DB instance is automatically upgraded after Amazon RDS tests and approves the new version. For information about performing an upgrade, see Upgrading a DB instance engine version.

If your MySQL DB instance is using read replicas, you must upgrade all of the read replicas before upgrading the source instance. If your DB instance is in a multi-AZ deployment, both the primary and standby replicas are upgraded. Your DB instance will not be available until the upgrade is complete.

Topics

- Overview of upgrading
- Major version upgrades for MySQL
- Testing an upgrade
- Upgrading a MySQL DB instance
- Automatic minor version upgrades for MySQL
- Using a read replica to reduce downtime when upgrading a MySQL database

2.5.1 Upgrading a MySQL DB snapshot

With Amazon RDS, you can create a storage volume DB snapshot of your MySQL DB instance. When you create a DB snapshot, the snapshot is based on the engine version used by your Amazon RDS instance. In addition to upgrading the DB engine version of your DB instance, you can also upgrade the engine version for your DB snapshots. For example, you can upgrade DB snapshots created from the MySQL 5.1 engine to DB snapshots for the MySQL 5.5 engine. After restoring a DB snapshot upgraded to a new engine version, you should test that the upgrade was successful. To learn how to test a major version upgrade, see Testing an upgrade. To learn how to restore a DB snapshot, see Restoring from a DB snapshot.

Amazon RDS supports upgrading a MySQL DB snapshot from MySQL 5.1 to MySQL 5.5.

You can upgrade manual DB snapshots, which can be encrypted or not encrypted, from MySQL 5.1 to MySQL 5.5 within the same AWS Region. You can't upgrade automated DB snapshots that are created during the automated backup process.

2.6 RDS IN CLOUD COMPUTING

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

Why do you want to run a relational database in the AWS Cloud?

Because AWS takes over many of the difficult and tedious management tasks of a relational database.

Topics

- Amazon EC2 and on-premises databases
- Amazon RDS and Amazon EC2
- Amazon RDS Custom for Oracle and Microsoft SQL Server
- Amazon RDS on AWS Outputs

2.6.1 Amazon EC2 and on-premises databases

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster.

When you buy an on-premises server, you get CPU, memory, storage, and IOPS, all bundled together. With Amazon EC2, these are split apart so that you can scale them independently. If you need more CPU, less IOPS, or more storage, you can easily allocate them.

For a relational database in an on-premises server, you assume full responsibility for the server, operating system, and software. For a database on an Amazon EC2 instance, AWS manages the layers below the operating system. In this way, Amazon EC2 eliminates some of the burden of managing an on-premises database serve

Amazon EC2 isn't a fully managed service. Thus, when you run a database on Amazon EC2, you're more prone to user errors. For example, when you update the operating system or database software manually, you might accidentally cause application downtime. You might spend hours checking every change to identify and fix an issue.

2.6.2 Amazon RDS and Amazon EC2

Amazon RDS is a managed database service. It's responsible for most management tasks. By eliminating tedious manual tasks, Amazon RDS frees you to focus on your application and your users. We recommend Amazon RDS over Amazon EC2 as your default choice for most database deployments.

Amazon RDS provides the following specific advantages over database deployments that aren't fully managed:

- You can use the database products you are already familiar with: MariaDB, Microsoft SQL Server, MySQL, Oracle, and PostgreSQL.
- Amazon RDS manages backups, software patching, automatic failure detection, and recovery.
- You can turn on automated backups, or manually create your own backup snapshots. You can use these backups to restore a database. The Amazon RDS restore process works reliably and efficiently.
- You can get high availability with a primary instance and a synchronous secondary instance that you can fail over to when problems occur. You can also use read replicas to increase read scaling.
- In addition to the security in your database package, you can help control who can access your RDS databases. To do so, you can use AWS Identity and Access Management (IAM) to define users and permissions. You can also help protect your databases by putting them in a virtual private cloud (VPC).

2.6.3 Amazon RDS Custom for Oracle and Microsoft SQL Server

Amazon RDS Custom is an RDS management type that gives you full access to your database and operating system.

You can use the control capabilities of RDS Custom to access and customize the database environment and operating system for legacy and packaged business applications. Meanwhile, Amazon RDS automates database administration tasks and operations.

In this deployment model, you can install applications and change configuration settings to suit your applications. At the same time, you can offload database administration tasks such as provisioning, scaling, upgrading, and backup to AWS. You can take advantage of the database management benefits of Amazon RDS, with more control and flexibility.

For Oracle Database and Microsoft SQL Server, RDS Custom combines the automation of Amazon RDS with the flexibility of Amazon EC2. For more information on RDS Custom, see Working with Amazon RDS Custom.

With the shared responsibility model of RDS Custom, you get more control than in Amazon RDS, but also more responsibility.

2.6.4 Amazon RDS on AWS Outposts

Amazon RDS on AWS Outposts extends RDS for SQL Server, RDS for MySQL, and RDS for PostgreSQL databases to AWS Outposts environments. AWS Outposts uses the same hardware as in public AWS Regions to bring AWS services, infrastructure, and operation models on-premises. With RDS on Outposts, you can provision managed DB instances close to the business applications that must run on-premises

You use the same AWS Management Console, AWS CLI, and RDS API to provision and manage on-premises RDS on Outposts DB instances as you do for RDS DB instances running in the AWS Cloud. RDS on Outposts automates tasks, such as database provisioning, operating system and database patching, backup, and long-term archival in Amazon S3.

RDS on Outposts supports automated backups of DB instances. Network connectivity between your Outpost and your AWS Region is required to back up and restore DB instances. All DB snapshots and transaction logs from an Outpost are stored in your AWS Region. From your AWS Region, you can restore a DB instance from a DB snapshot to a different Outpost. For more information, see Working with backups.

RDS on Outposts supports automated maintenance and upgrades of DB instances. For more information, see Maintaining a DB instance.

RDS on Outposts uses encryption at rest for DB instances and DB snapshots using your AWS KMS key. For more information about encryption at rest, see Encrypting Amazon RDS resources.

CHAPTER 3

MATERIALS AND METHODS ALGORITHMS

Creating and connecting to a DB instance for Amazon RDS Custom for SQL Server You can create an RDS Custom DB instance, and then connect to it using AWS Systems Manager or Remote Desktop Protocol (RDP)

Before you can create or connect to an RDS Custom for SQL Server DB instance, make sure to complete the tasks in Setting up your environment for Amazon RDS Custom for SQL Server.

You can tag RDS Custom DB instances when you create them, but don't create or modify the AWSRDSCustom tag that's required for RDS Custom automation. For more information, see Tagging RDS Custom for SQL Server resources.

The first time that you create an RDS Custom for SQL Server DB instance, you might receive the following error: The service-linked role is in the process of being created. Try again later. If you do, wait a few minutes and then try again to create the DB instance.

Topics

- Creating an RDS Custom for SQL Server DB instance
- Connecting to RDS Custom DB instance using AWS Systems Manager
- Connecting to your RDS Custom DB instance using RDP
- Deploy WordPress to an Amazon EC2 instance (Amazon Linux or Red Hat Enterprise Linux and Linux, macOS, or Unix)

3.1 CREATING AN RDS CUSTOM FOR SQL SERVER DB INSTANCE

Create an Amazon RDS Custom for SQL Server DB instance using either the AWS Management Console or the AWS CLI. The procedure is similar to the procedure for creating an Amazon RDS DB instance

Console

To create an RDS Custom for SQL Server DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose Databases.
3. Choose Create database.
4. Choose Standard create for the database creation method.
5. For Engine options, choose Microsoft SQL Server for the engine type.
6. For Database management type, choose Amazon RDS Custom.
7. In the Edition section, choose the DB engine edition that you want to use. For RDS Custom for SQL Server, the choices are Enterprise, Standard, and Web.
8. For Database version, keep the SQL Server 2019 default value.
9. For Templates, choose Production.
10. In the Settings section, enter a unique name for the DB instance identifier.
11. To enter your master password, do the following:

a. In the Settings section, open Credential Settings.
b. Clear the Auto generate a password check box.
c. Change the Master username value and enter the same password in Master password and Confirm password.

   By default, the new RDS Custom DB instance uses an automatically generated password for the master user.

12. In the DB instance size section, choose a value for DB instance class.

    For supported classes, see DB instance class support for RDS Custom for SQL Server.

13. Choose Storage settings.
14. For RDS Custom security, do the following:

a. For IAM instance profile, choose the instance profile for your RDS Custom for SQL Server DB instance.

   The IAM instance profile must begin with AWSRDSCustom, for example *AWSRDSCustomInstanceProfileForRdsCustomInstance*.

b. For Encryption, choose Enter a key ARN to list the available AWS KMS keys. Then choose your key from the list.

   An AWS KMS key is required for RDS Custom. For more information, see Make sure that you have a symmetric encryption AWS KMS key.

15. For the remaining sections, specify your preferred RDS Custom DB instance settings. For information about each setting, see Settings for DB instances. The following settings don't appear in the console and aren't supported:

a. Processor features
b. Storage autoscaling
c. Availability & durability
d. Password and Kerberos authentication option in Database authentication (only Password authentication is supported)
e. Database options group in Additional configuration
f. Performance Insights
g. Log exports
h. Enable auto minor version upgrade
i. Deletion protection

   Backup retention period is supported, but you can't choose 0 days.

16. Choose Create database.

    The View credential details button appears on the Databases page.

    To view the master user name and password for the RDS Custom DB instance, choose View credential details.

    To connect to the DB instance as the master user, use the user name and password that appear.

    You can't view the master user password again. If you don't record it, you might have to change it. To change the master user password after the RDS Custom DB instance is available, modify the DB instance. For more information about modifying a DB instance, see Managing an Amazon RDS Custom for SQL Server DB instance

17. Choose Databases to view the list of RDS Custom DB instances.
18. Choose the RDS Custom DB instance that you just created.

    On the RDS console, the details for the new RDS Custom DB instance appear:

- The DB instance has a status of creating until the RDS Custom DB instance is created and ready for use. When the state changes to available, you can connect to the DB instance. Depending on the instance class and storage allocated, it can take several minutes for the new DB instance to be available.
- Role has the value Instance (RDS Custom).
- RDS Custom automation mode has the value Full automation. This setting means that the DB instance provides automatic monitoring and instance recovery.

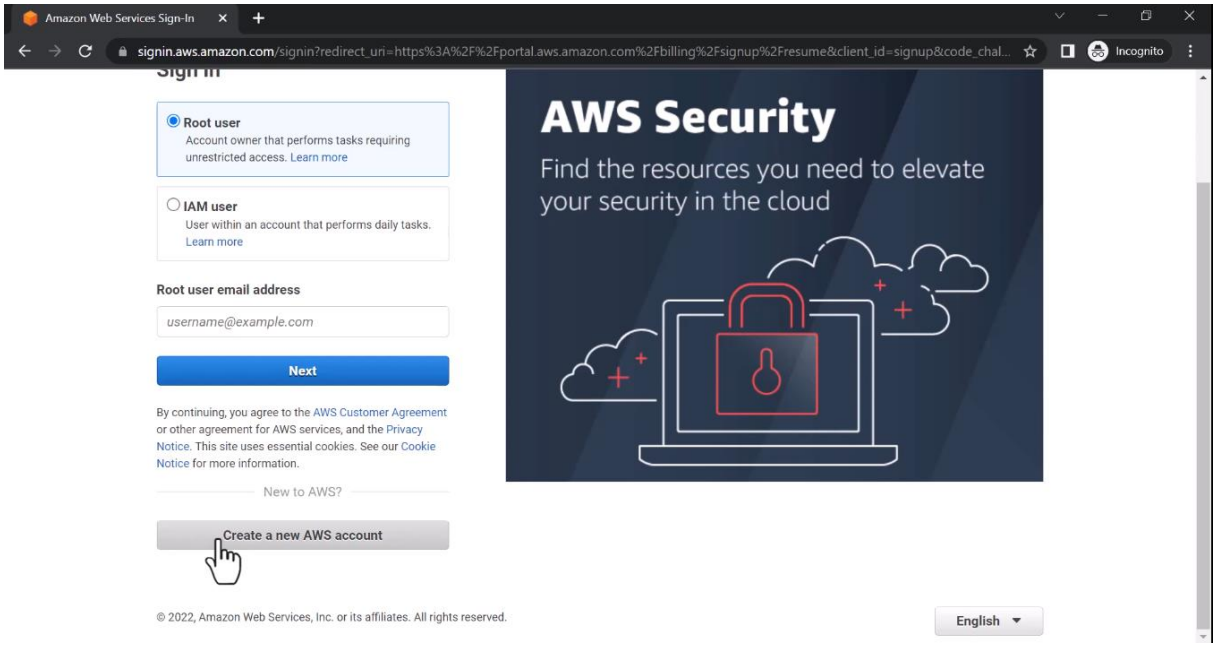


Fig 3.1 Creating a new amazon account

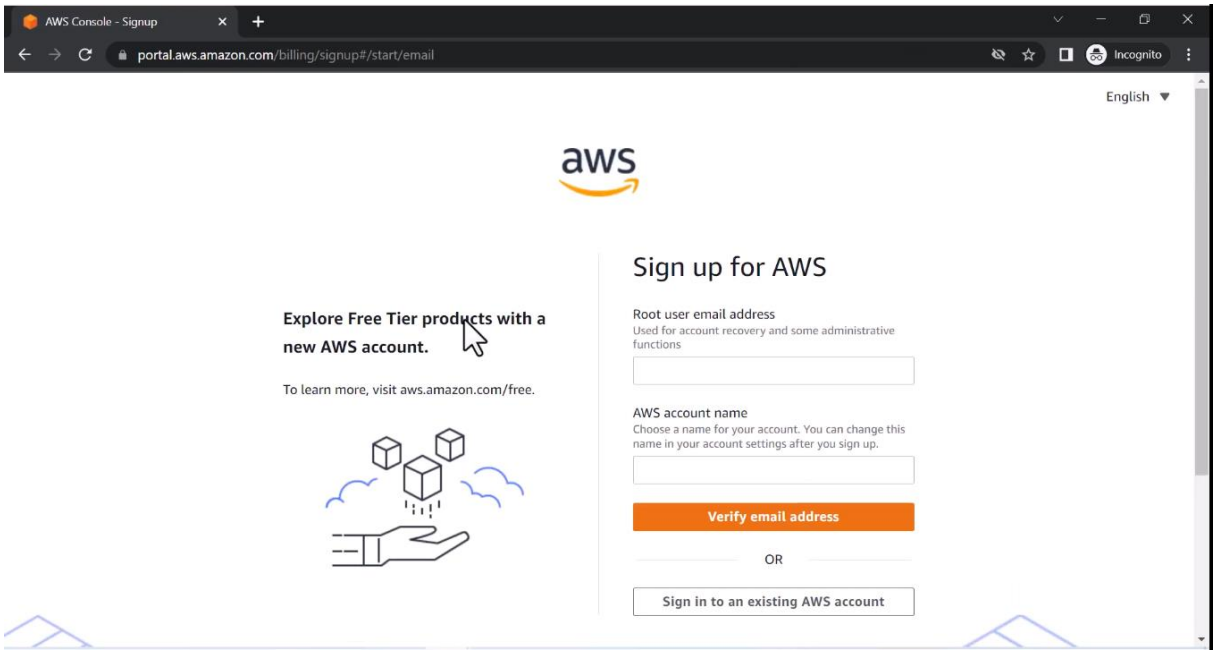


Fig 3.2 Sign up with email and account name

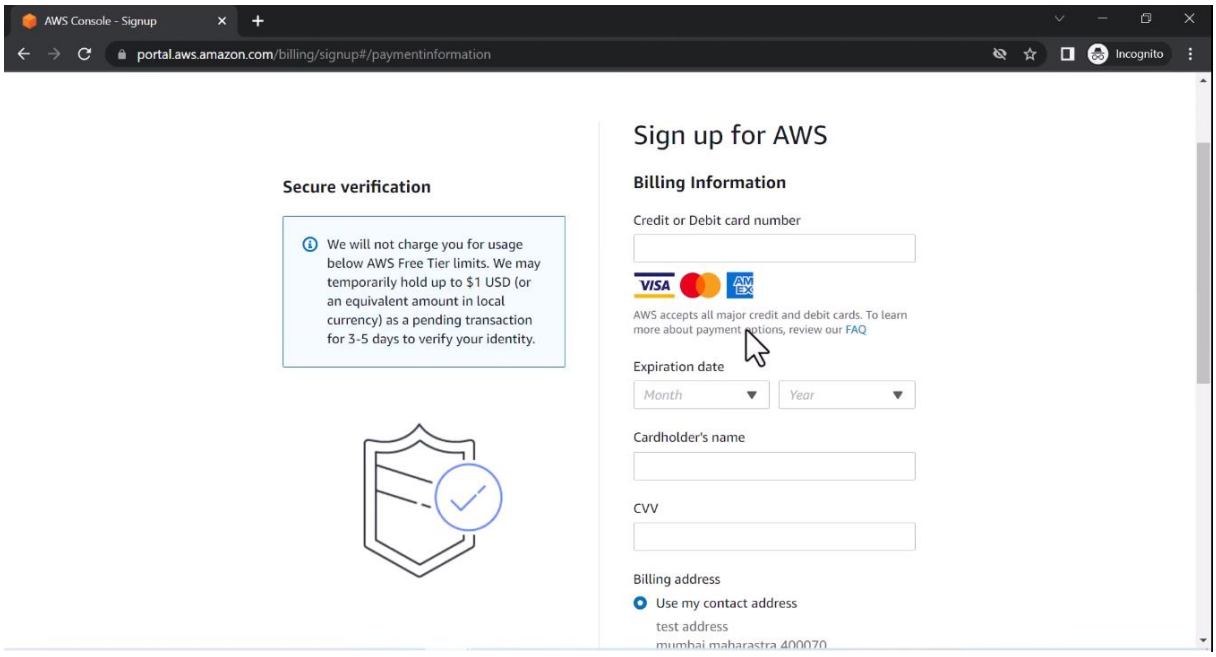


Fig 3.3 Giving the card details

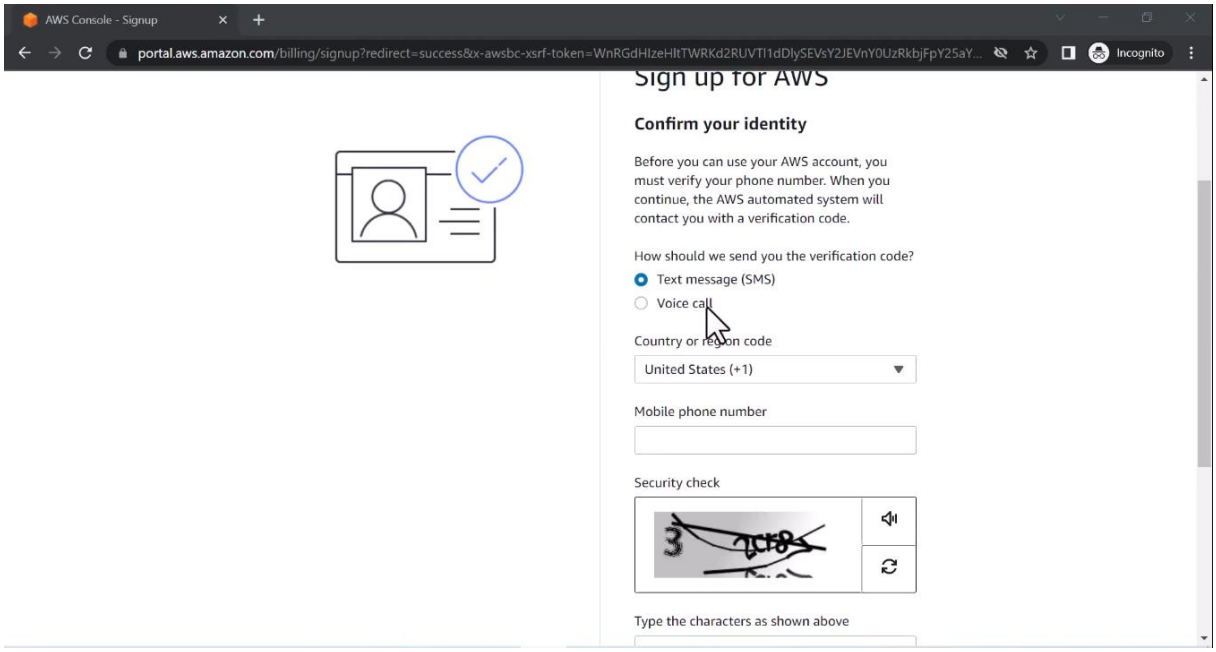


Fig 3.4 Confirm the identity by phone number

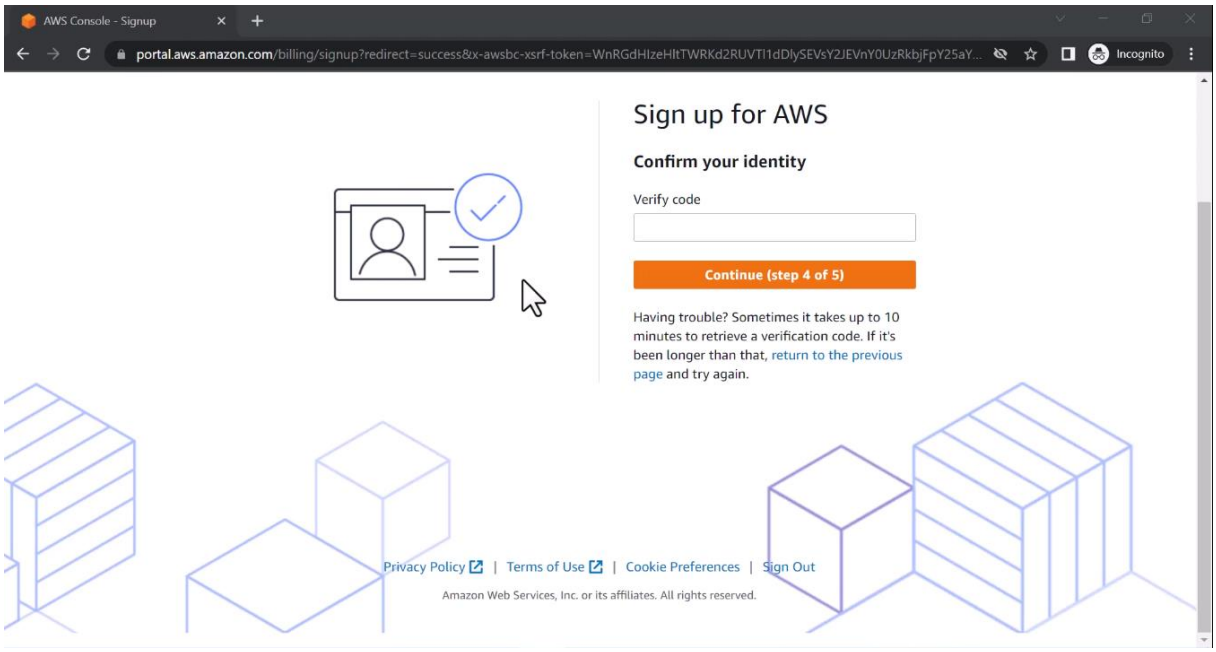


Fig 3.5 Checking whether the giving number is correct or wrong by verification code

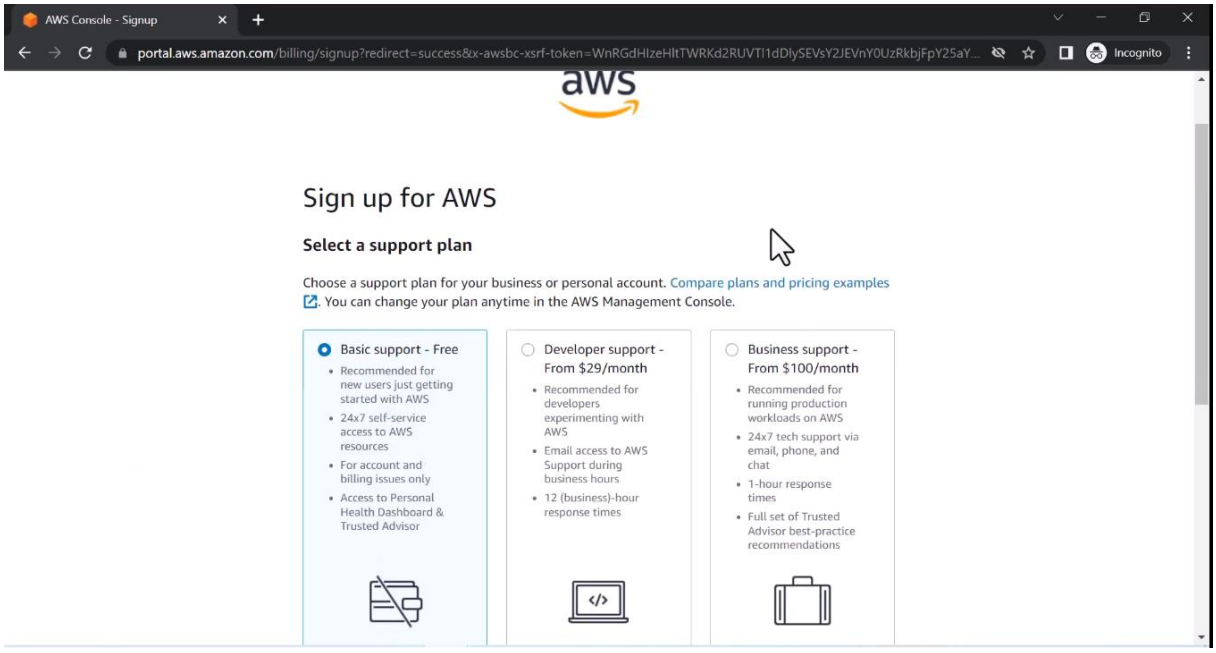


Fig 3.6 Support plan we are using basic support – free (free tier)

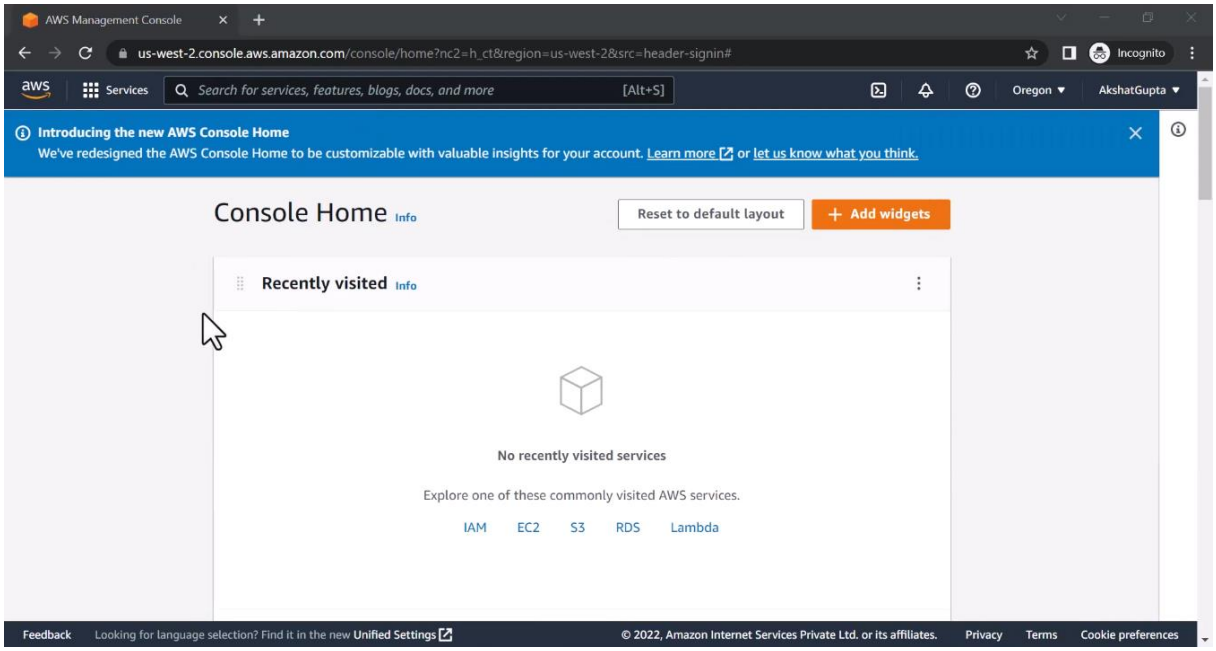


Fig 3.7 After login the main console home in aws account

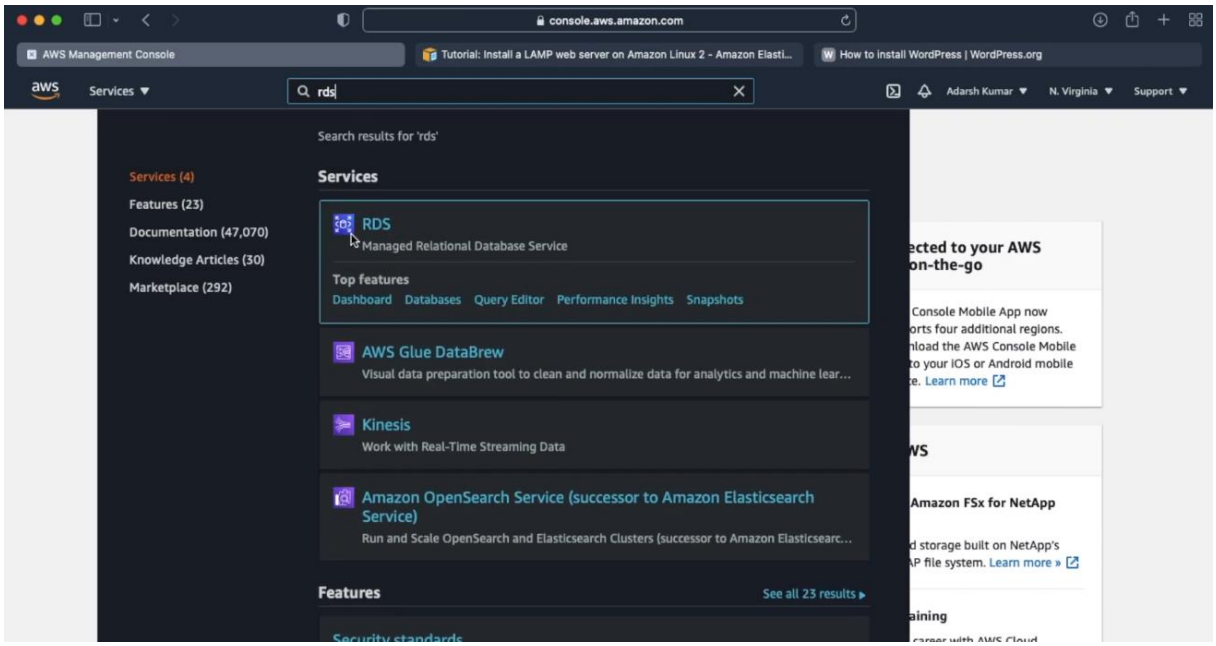


Fig 3.8 In search bar we are typing rds to create a instance

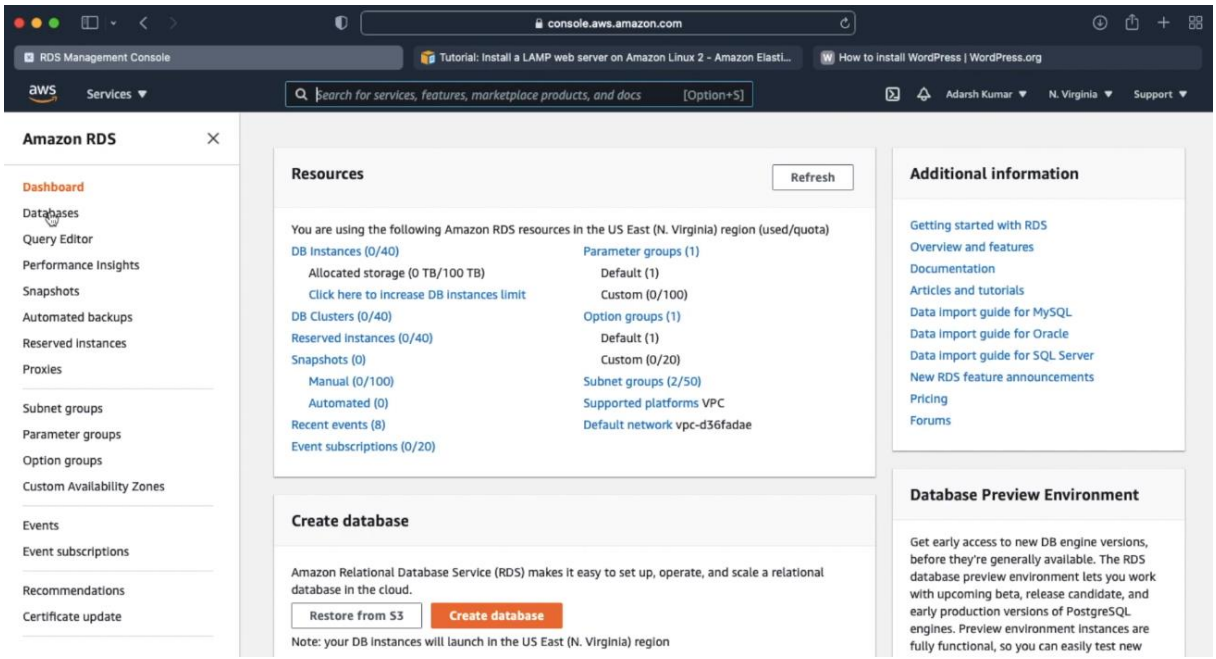


Fig 3.9 In rds we are we are going to click on database

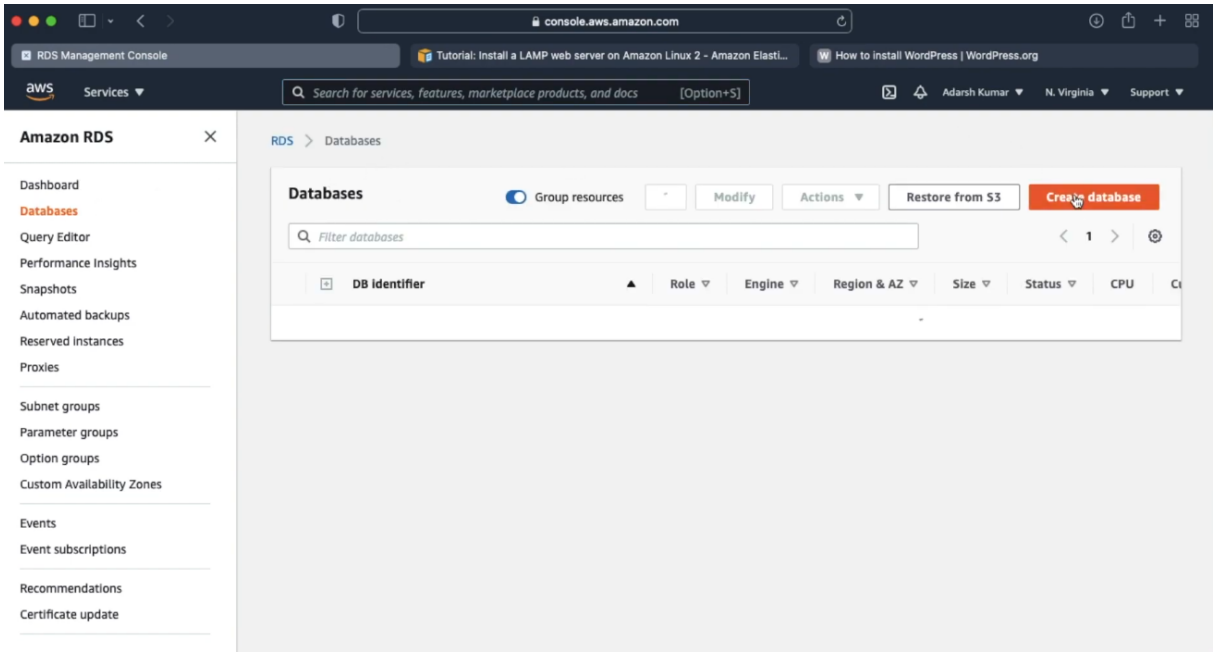


Fig 3.10 Creating a amazon rds database

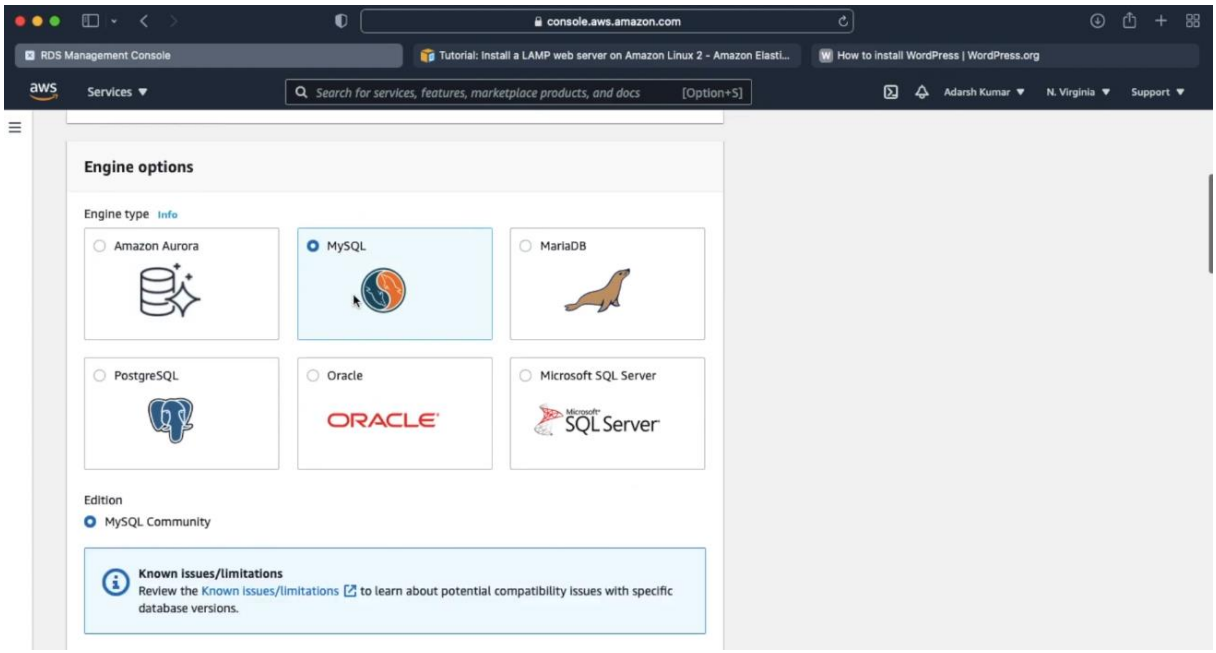


Fig 3.11 In database selecting the engine options (MySQL community)

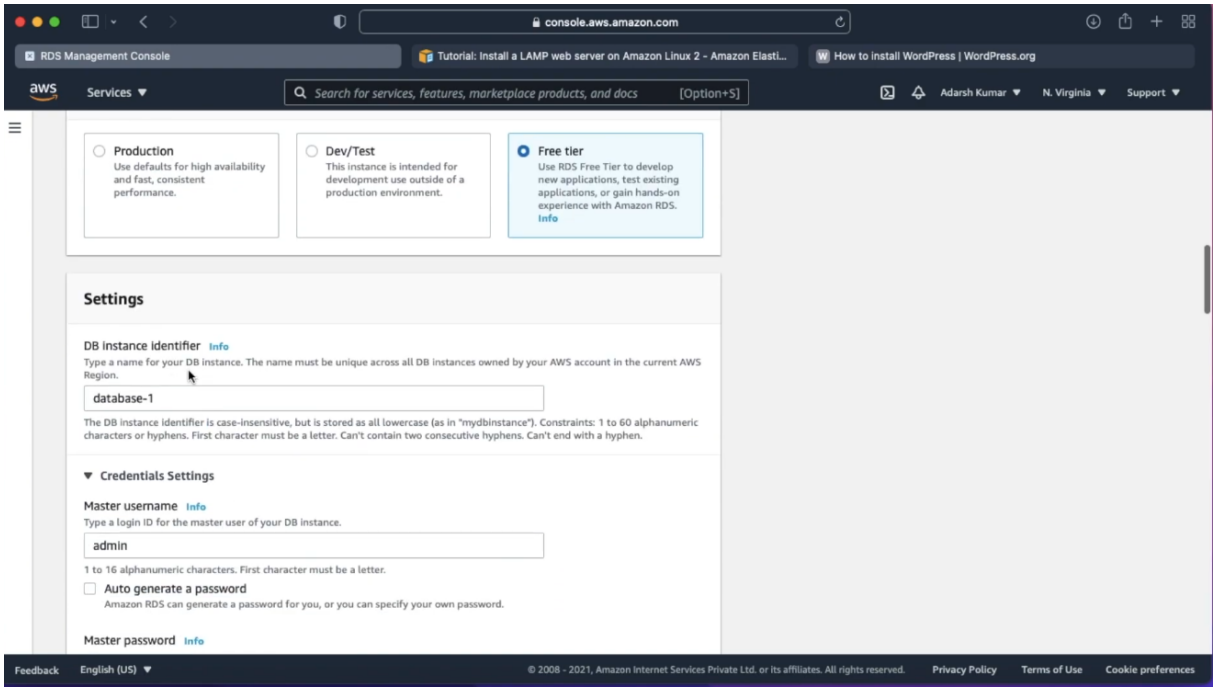


Fig 3.12 The engine is free tier based

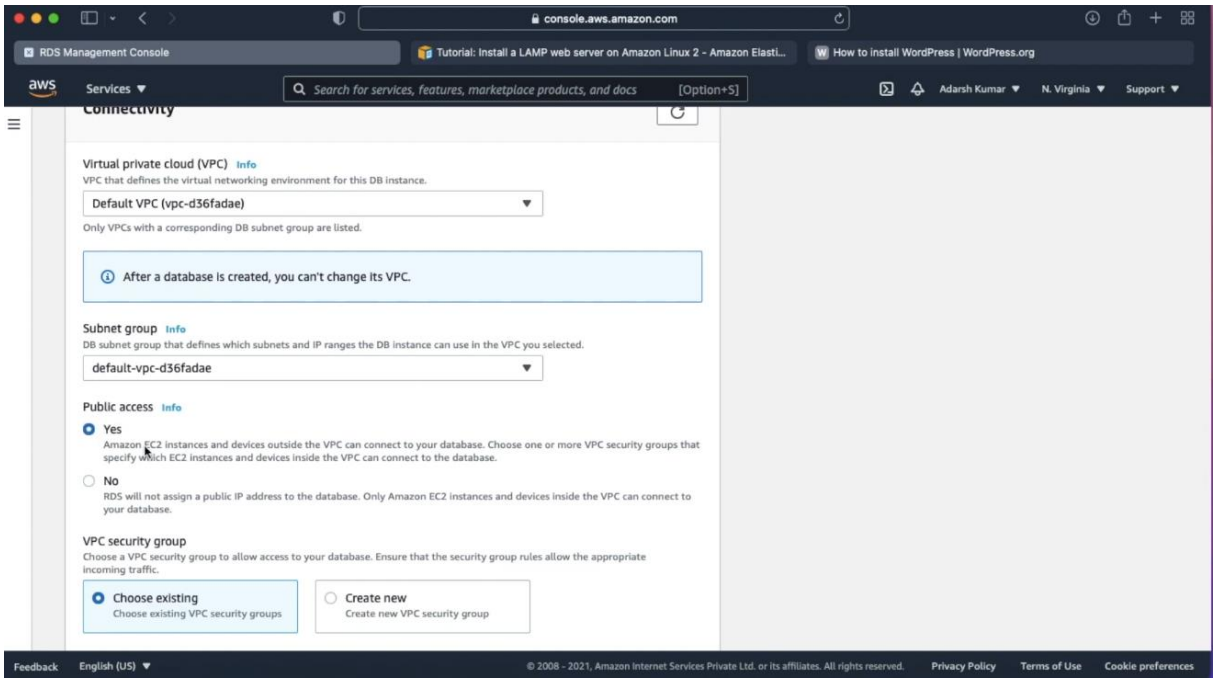


Fig 3.13 Giving access to public

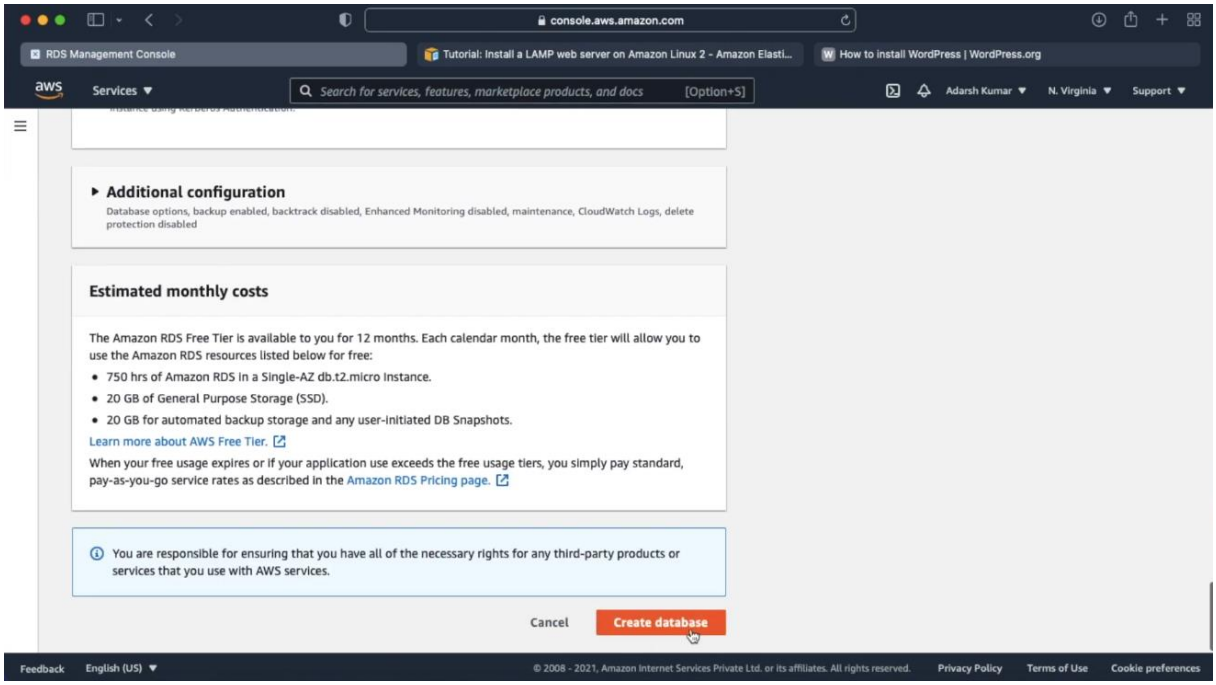


Fig 3.14 Giving access to create database

CODE:

You create an RDS Custom DB instance by using the create-db-instance AWS CLI command.

The following options are required:

- --db-instance-identifier
- --db-instance-class (for a list of supported instance classes, see DB instance class support for RDS Custom for Oracle)
- --engine (custom-sqlserver-ee, custom-sqlserver-se, or custom-sqlserver-web)
- --kms-key-id
- --custom-iam-instance-profile

The following example creates an RDS Custom for SQL Server DB instance named my-custom-instance. The backup retention period is 3 days.

For Linux, macOS, or Unix:

```
aws rds create-db-instance \

    --engine custom-sqlserver-ee \

    --engine-version 15.00.4073.23.v1 \

    --db-instance-identifier my-custom-instance \

    --db-instance-class db.m5.xlarge \

    --allocated-storage 20 \

    --db-subnet-group mydbsubnetgroup \

    --master-username myuser \

    --master-user-password mypassword \

    --backup-retention-period 3 \
```

```
    --no-multi-az \

    --port 8200 \

    --license-model license-included \

    --kms-key-id mykmskey \

    --custom-iam-instance-profile
AWSRDSCustomInstanceProfileForRdsCustomInstance
```

For windows

```
aws rds create-db-instance ^
    --engine custom-sqlserver-ee ^
    --engine-version 15.00.4073.23.v1 ^
    --db-instance-identifier my-custom-instance ^
    --db-instance-class db.m5.xlarge ^
    --allocated-storage 20 ^
    --db-subnet-group mydbsubnetgroup ^
    --master-username myuser ^
    --master-user-password mypassword ^
    --backup-retention-period 3 ^
    --no-multi-az ^
    --port 8200 ^
    --license-model license-included ^
    --kms-key-id mykmskey ^
    --custom-iam-instance-profile
AWSRDSCustomInstanceProfileForRdsCustomInstance
```

Get details about your instance by using the describe-db-instances command.

```
aws rds describe-db-instances --db-instance-identifier my-custom-instance
```

The following partial output shows the engine, parameter groups, and other information.


```
{
    "DBInstances": [
        {
            "PendingModifiedValues": {},
            "Engine": "custom-sqlserver-ee",
            "MultiAZ": false,
            "DBSecurityGroups": [],
            "DBParameterGroups": [
                {
                    "DBParameterGroupName": "default.custom-sqlserver-ee-15",
                    "ParameterApplyStatus": "in-sync"
                }
            ],
            "AutomationMode": "full",
            "DBInstanceIdentifier": "my-custom-instance",
            "TagList": []
        }
    ]
```


3.2 DEPLOY WORDPRESS TO AN AMAZON EC2 INSTANCE (AMAZON LINUX OR RED HAT ENTERPRISE LINUX AND LINUX, MACOS, OR UNIX)

we procedures in these tutorials provide suggestions for the location in which to store files (for example, c:\temp) and the names to give to buckets, subfolders, or files (for example, codedeploydemobucket, HelloWorldApp, and CodeDeployDemo-EC2-Trust.json, respectively), but you are not required to use them. Just be sure to substitute your file locations and names as you perform the

procedures.

you deploy WordPress, an open source blogging tool and content management system based on PHP and MySQL, to a single Amazon EC2 instance running Amazon Linux or Red Hat Enterprise Linux (RHEL).

- To practice deploying to an Amazon EC2 instance running Windows Server instead, see Tutorial: Deploy a "hello, world!" application with CodeDeploy (Windows Server).

- To practice deploying to an on-premises instance instead of an Amazon EC2 instance, see Tutorial: Deploy an application to an on-premises instance with CodeDeploy (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux).

  This steps are presented from the perspective of a local development machine running Linux, macOS, or Unix. Although you can complete most of these steps on a local machine running Windows, you must adapt the steps that cover commands such as chmod and wget, applications such as sed, and directory paths such as /tmp.

Before you start this tutorial, you must complete the prerequisites in Getting started with CodeDeploy. These include configuring your IAM user account, installing or upgrading the AWS CLI, and creating an IAM instance profile and a service role

Topics

- Step 1: Launch and configure an Amazon Linux or Red Hat Enterprise Linux Amazon EC2 instance
- Step 2: Configure your source content to be deployed to the Amazon Linux or Red Hat Enterprise Linux Amazon EC2 instance
- Step 3: Upload your WordPress application to Amazon S3
- Step 4: Deploy your WordPress application
- Step 5: Update and redeploy your WordPress application
- Step 6: Clean up your WordPress application and related resources

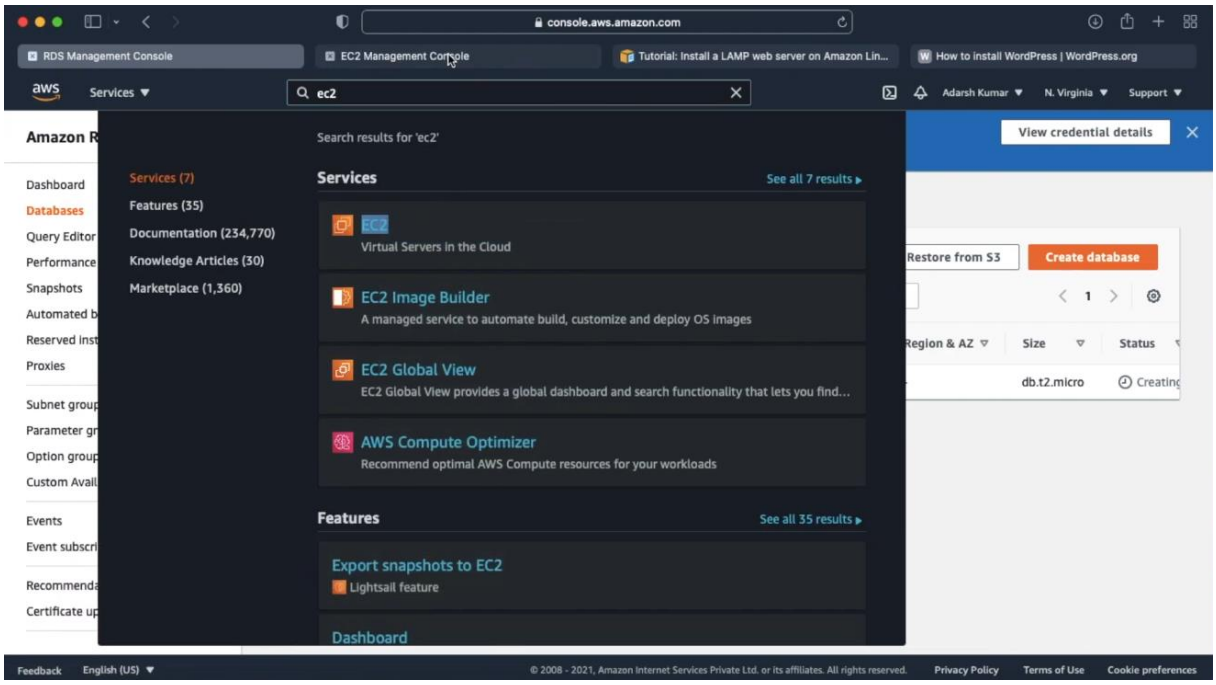


Fig 3.15 In search bar typing EC2 to launch

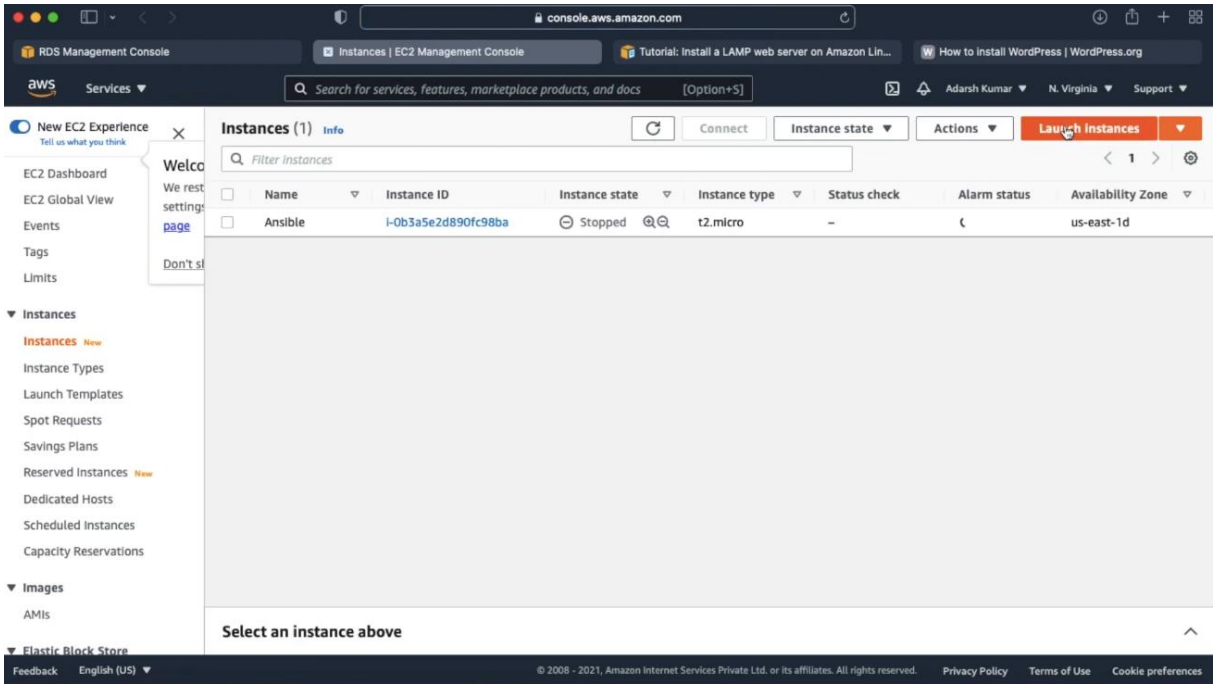


Fig 3.16 Launching the EC2 instance

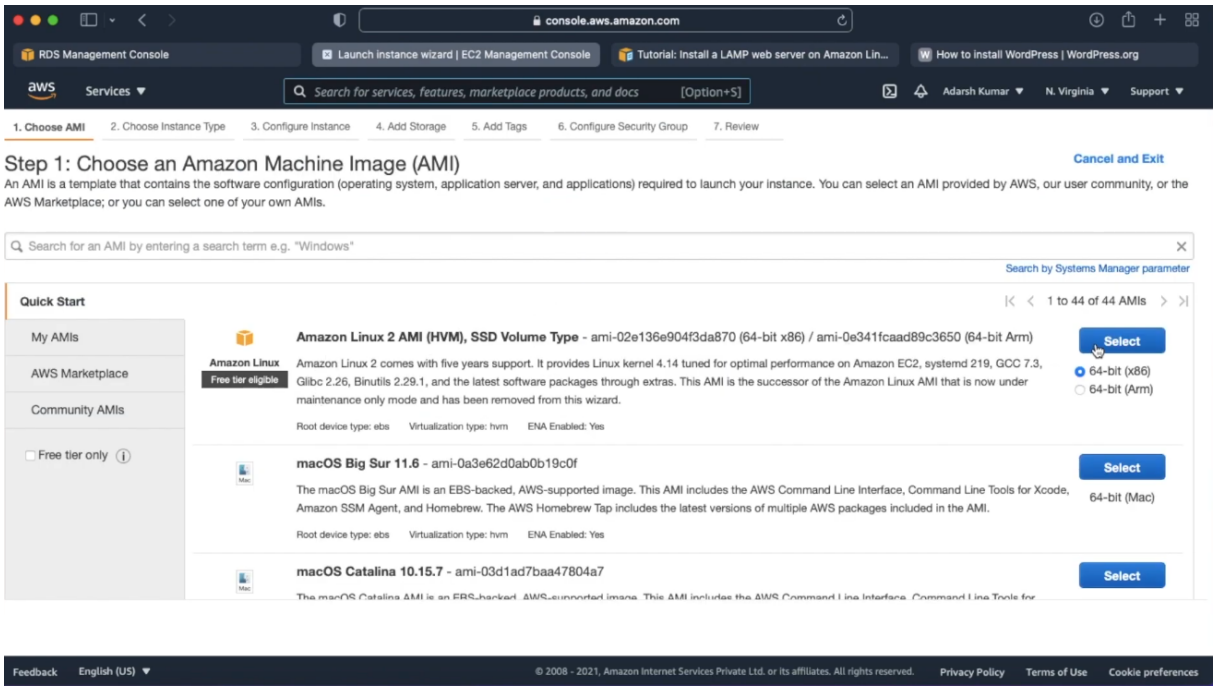


Fig 3.17 Selecting the amazon Linux 2 AMI

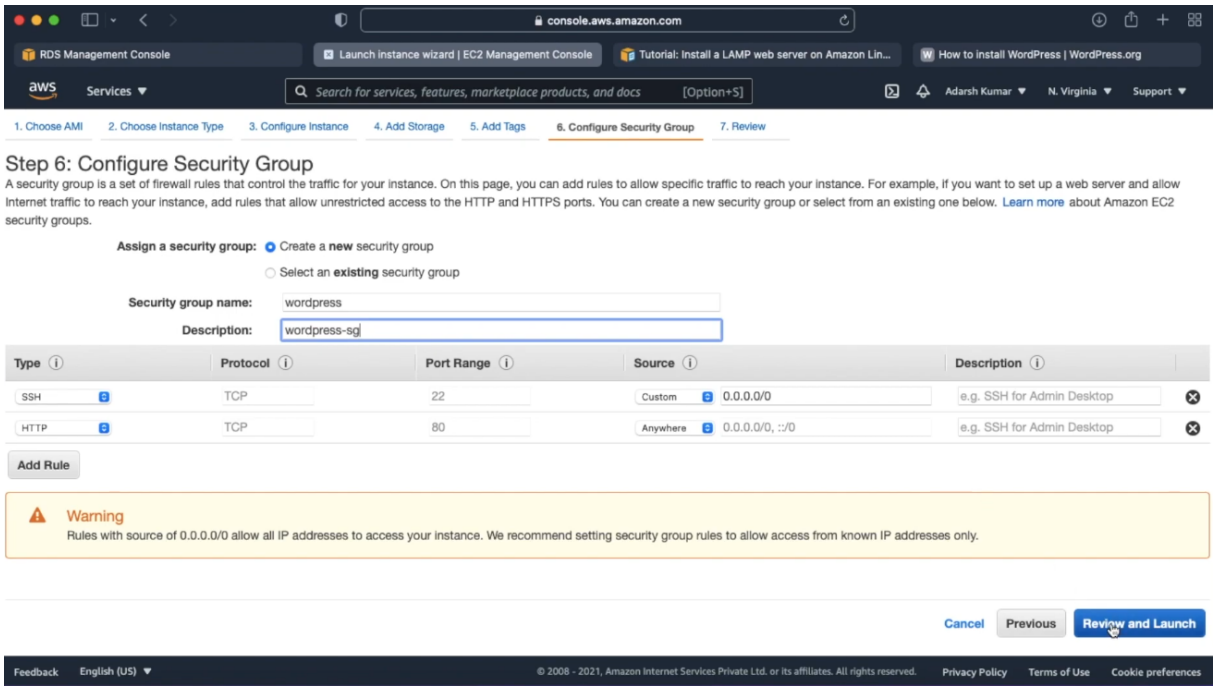


Fig 3.18 Conforming security by user name and description, checking whether the http, ssh are present or not and launching the EC2 instance

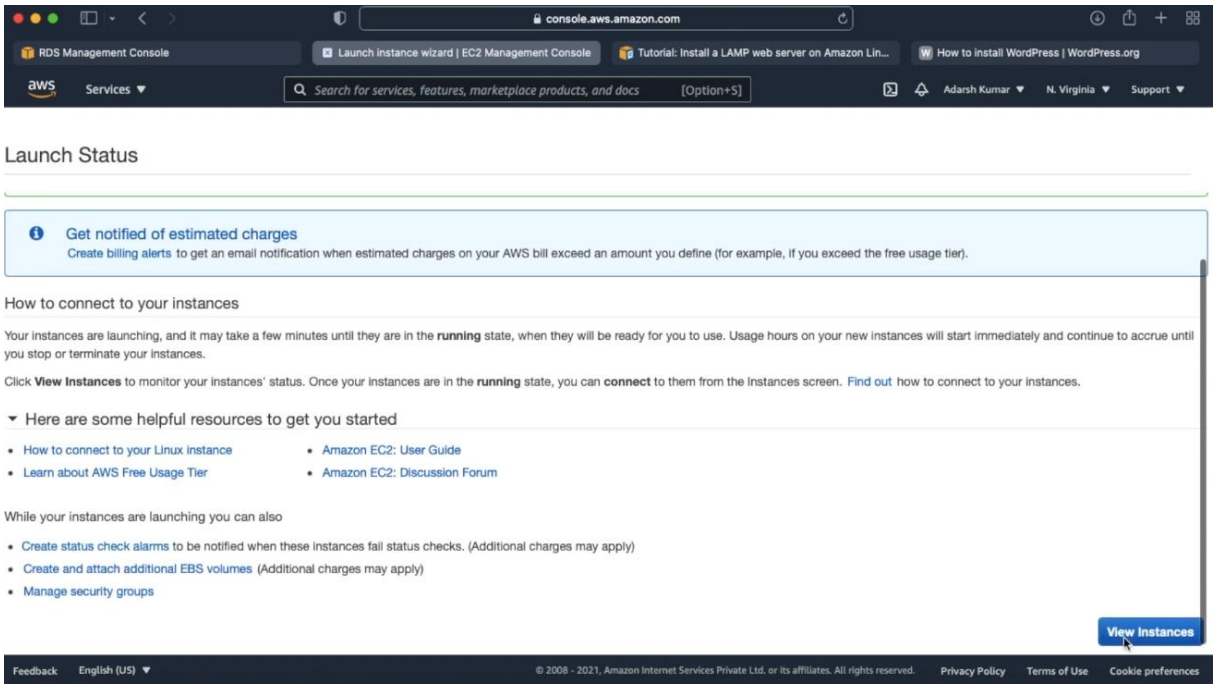


Fig 3.19 Launching status for select the instances

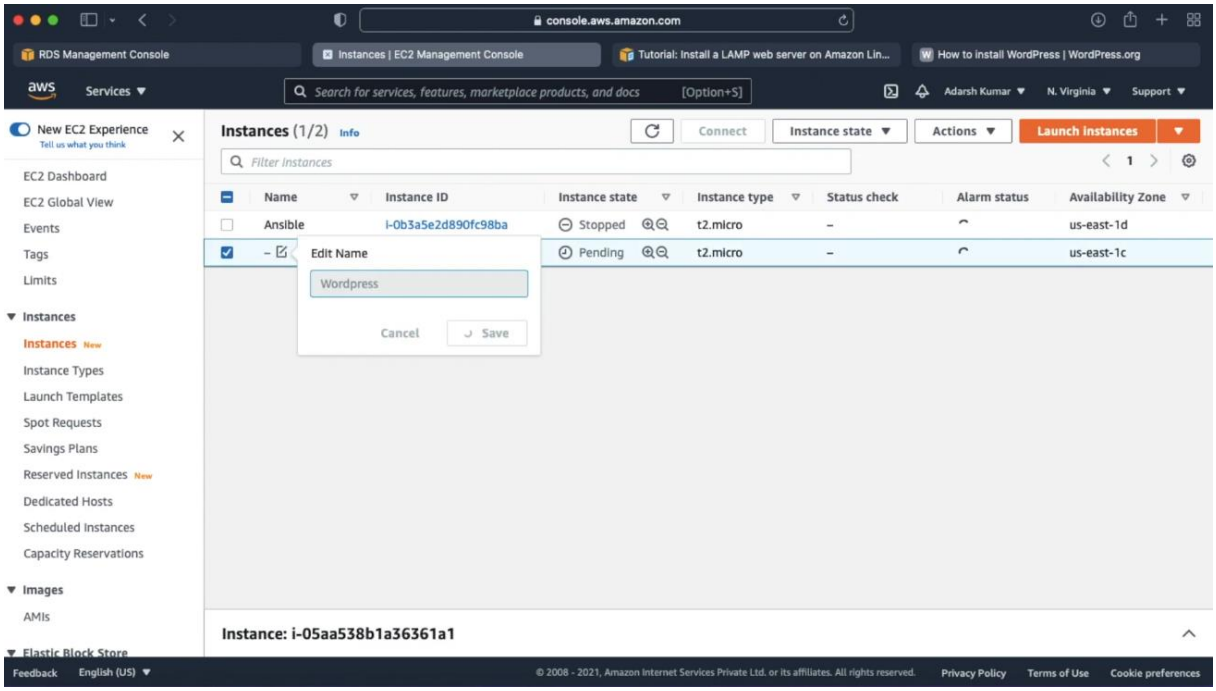


Fig 3.20 Giving name for instance and launching the instances

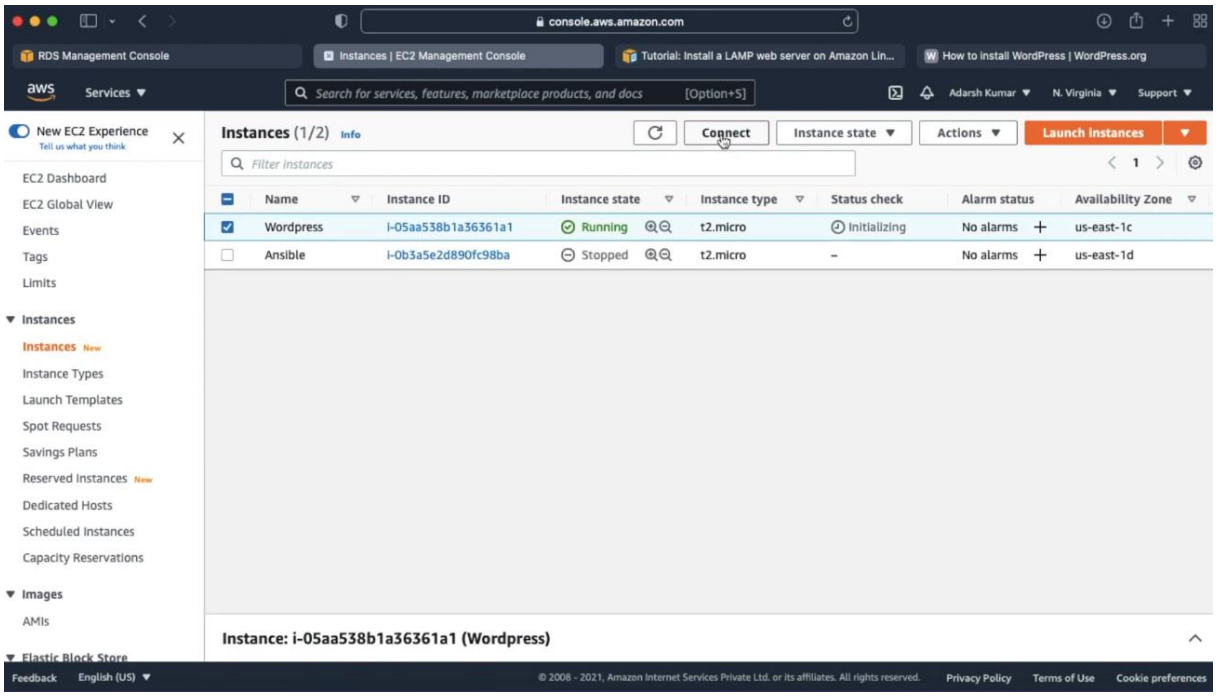


Fig 3.21 Creating the WordPress instance and click on connect instance

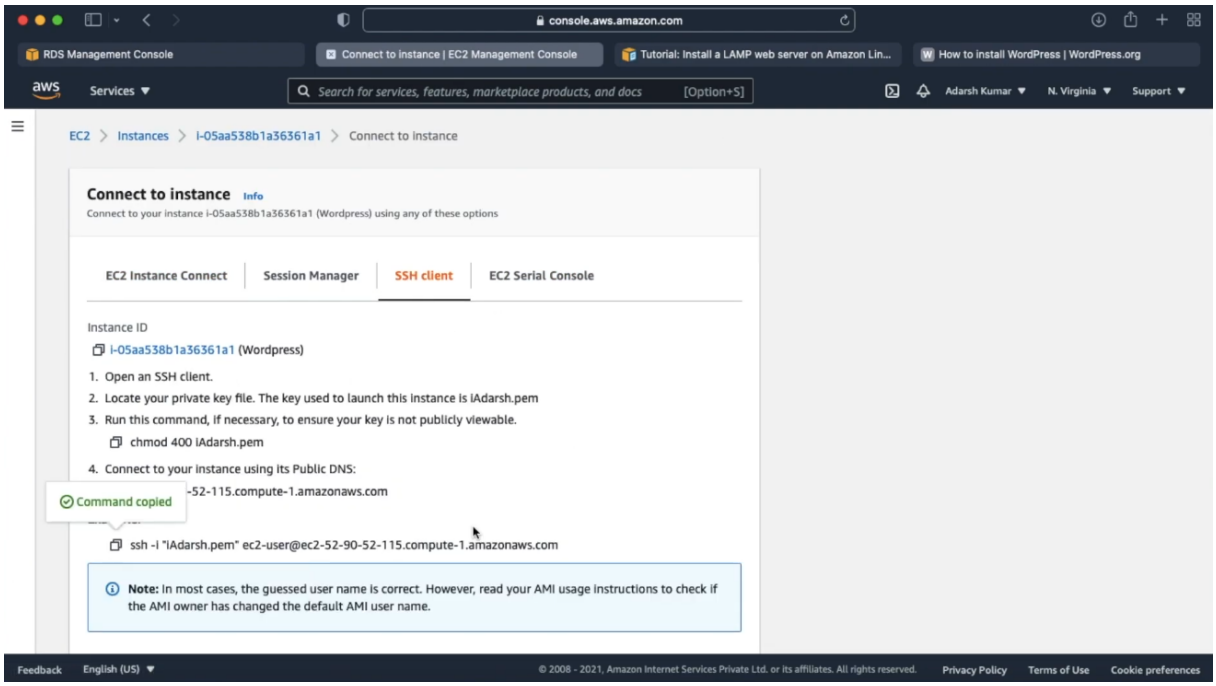


Fig 3.22 In connecting instance copy the address from ssh client

3.2.1 Launch and configure an Amazon Linux or Red Hat Enterprise Linux Amazon EC2 instance

To deploy the WordPress application with CodeDeploy, you'll need an Amazon student intended to learnEC2 instance running Amazon Linux or Red Hat Enterprise Linux (RHEL). The Amazon EC2 instance requires a new inbound security rule that allows HTTP connections. This rule is needed in order to view the WordPress page in a browser after it is successfully deployed.

Follow the instructions in Create an Amazon EC2 instance for CodeDeploy. When you get to the part in those instructions about assigning an Amazon EC2 instance tag to the instance, be sure to specify the tag key of Name and the tag value of CodeDeployDemo. (If you specify a different tag key or tag value, then the instructions in Step 4: Deploy your WordPress application may produce unexpected results.)

After you've followed the instructions to launch the Amazon EC2 instance, return to this page, and continue to the next section. Do not continue on to Create an application with CodeDeploy as the next step.

Connect to your Amazon Linux or RHEL Amazon EC2 instance

After your new Amazon EC2 instance is launched, follow these instructions to practice connecting to it.

1. Use the ssh command (or an SSH-capable terminal emulator like PuTTY) to connect to your Amazon Linux or RHEL Amazon EC2 instance. You will need the public DNS address of the instance and the private key for the key pair you used when you started the Amazon EC2 instance. For more information, see Connect to Your Instance.

   For example, if the public DNS address is ec2-01-234-567-890.compute-1.amazonaws.com, and your Amazon EC2 instance key pair for SSH access is named codedeploydemo.pem, you would type:

   ```
   ssh -i /path/to/codedeploydemo.pem ec2-user@ec2-01-234-567-890.compute-1.amazonaws.com
   ```

Replace */path/to/*codedeploydemo.pem with the path to your .pem file and the example DNS address with the address to your Amazon Linux or RHEL Amazon EC2 instance.

If you receive an error about your key file's permissions being too open, you will need to restrict its permissions to give access only to the current user (you). For example, with the chmod command on Linux, macOS, or Unix, type:

chmod 400 /path/to/codedeploydemo.pem

2. After you are signed in, you will see the AMI banner for the Amazon EC2 instance. For Amazon Linux, it should look like this:

```
__|  __|_  )
_|  (     /   Amazon Linux AMI
___|\___|___|
```

3. You can now sign out of the running Amazon EC2 instance.
   Do not stop or terminate the Amazon EC2 instance. Otherwise, CodeDeploy won't be able to deploy to it.

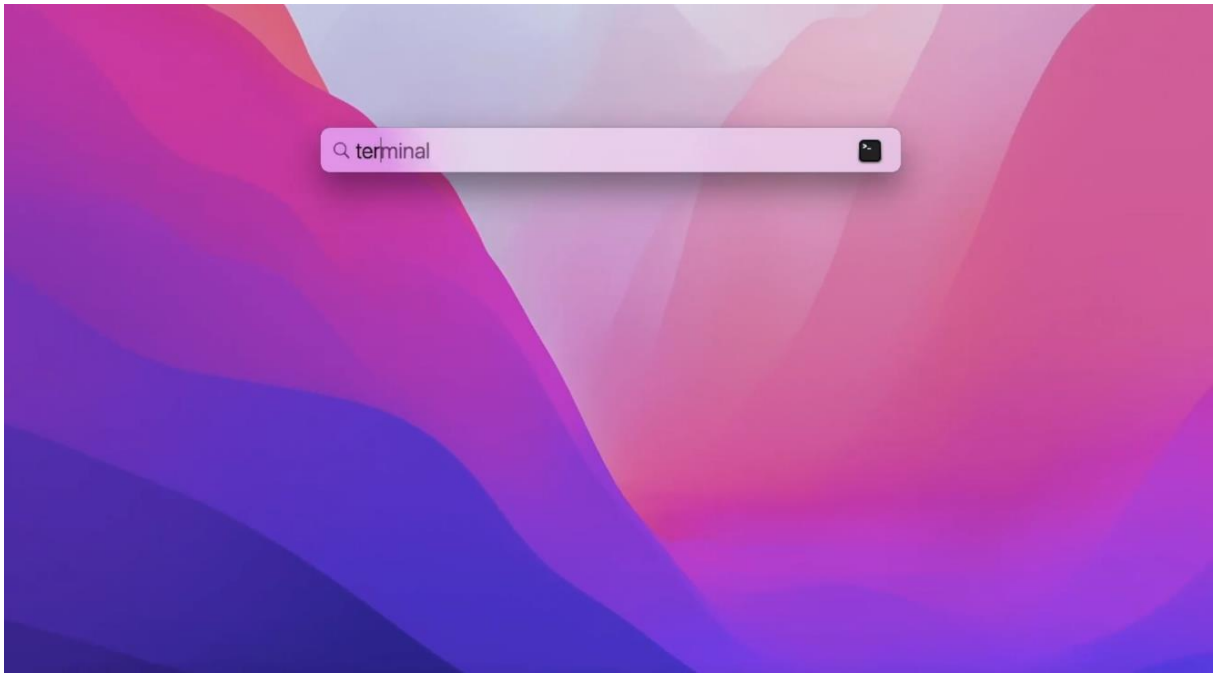


Fig 3.23 In google typing terminal

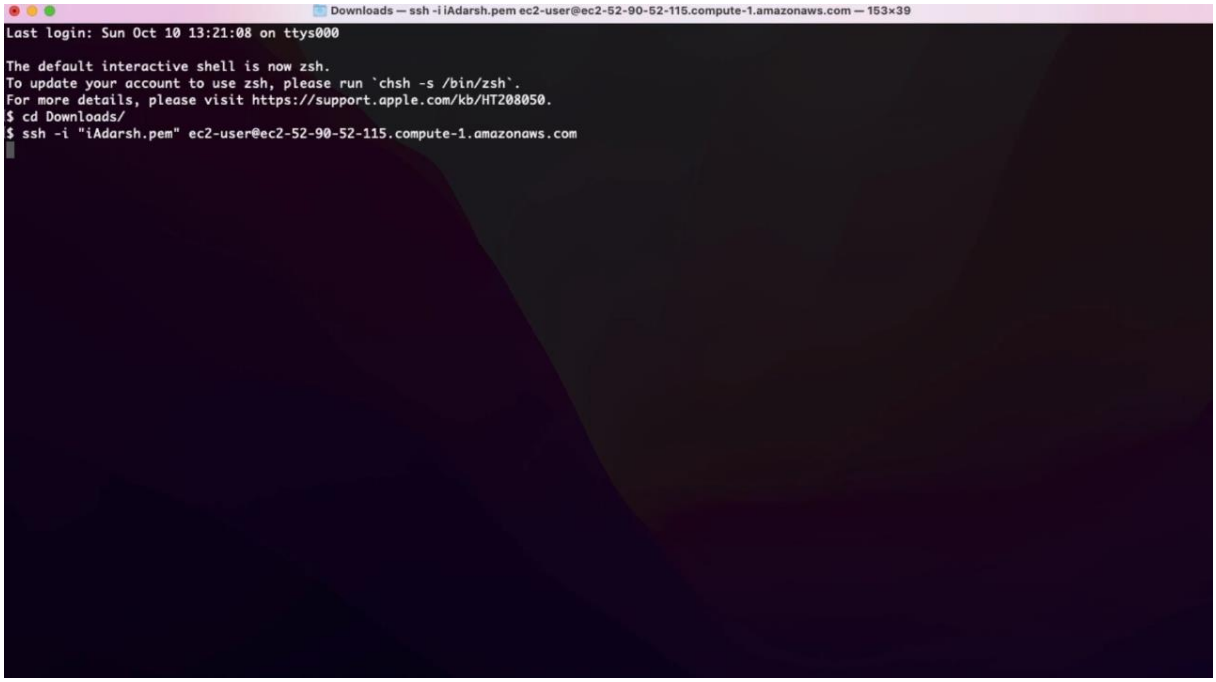


Fig 3.24 First giving directory (downloads),Giving client sight(ssh) then client name(Adarsh) and Ip address

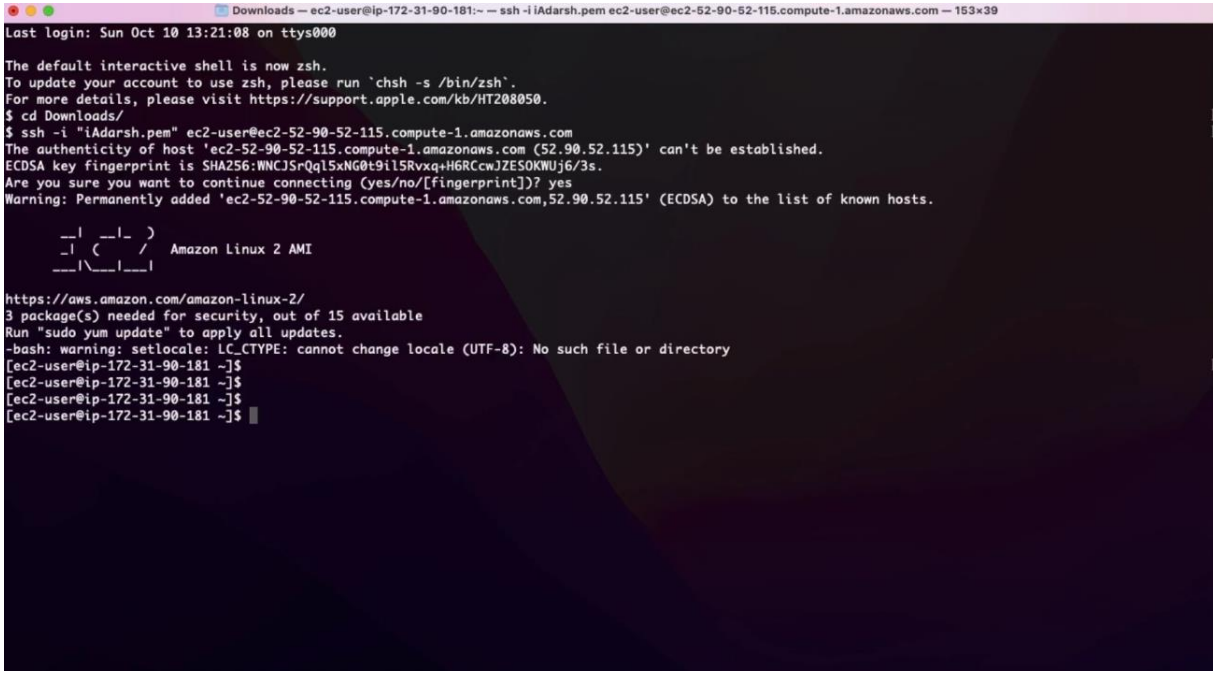


Fig 3.25 Checking weather Connecting or not …. giving reply(yes)

Add an inbound rule that allows HTTP traffic to your Amazon Linux or RHEL Amazon EC2 instance

The next step confirms your Amazon EC2 instance has an open HTTP port so you can see the deployed WordPress application's home page in a browser.

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Choose Instances, and then choose your instance.
3. On the Description tab, under Security groups, choose view inbound rules.

   You should see a list of rules in your security group like the following:

   Security Groups associated with i-1234567890abcdef0

Ports	Protocol	Source	launch-wizard-*N*
22	tcp	0.0.0.0/0	✓

4. Under Security groups, choose the security group for your Amazon EC2 instance. It might be named launch-wizard-*N*. The *N* in the name is a number assigned to your security group when your instance was created.

   Choose the Inbound tab. If the security group for your instance is configured correctly, you should see a rule with the following values:

   - Type: HTTP
   - Protocol: TCP
   - Port Range: 80
   - Source: 0.0.0.0/0
5. If you do not see a rule with these values, use the procedures in Adding Rules to a Security Group to add them to a new security rule.

3.2.2 Configure your source content to be deployed to the Amazon Linux or Red Hat Enterprise Linux Amazon EC2 instance

now it's time to configure your application's source content so you have something to deploy to the instance.

Topics

- Get the source code
- Create scripts to run your application
- Add an application specification file

Get the source code

For this tutorial, you deploy the WordPress content publishing platform from your development machine to the target Amazon EC2 instance. To get the WordPress source code, you can use built-in command-line calls. Or, if you have Git installed on your development machine, you can use that instead.

For these steps, we assume you downloaded a copy of the WordPress source code to the /tmp diresctory on your development machine. (You can choose any directory you like, but remember to substitute your location for /tmp wherever it is specified in these steps.)

Choose one of the following two options to copy the WordPress source files to your development machine. The first option uses built-in command-line calls. The second option uses Git.

Topics

- To get a copy of the WordPress source code (built-in command-line calls)
- To get a copy of the WordPress source code (Git)

To get a copy of the WordPress source code (built-in command-line calls)

- Unpack the master .zip file into the /tmp/WordPress_Temp directory (folder).
- Copy its unzipped contents to the /tmp/WordPress destination folder.
- Delete the temporary /tmp/WordPress_Temp folder and master file.

  Run the commands one at a time:

  unzip master -d /tmp/WordPress_Temp

mkdir -p /tmp/WordPress

cp -paf /tmp/WordPress_Temp/WordPress-master/* /tmp/WordPress

rm -rf /tmp/WordPress_Temp

rm -f master

This leaves you with a clean set of WordPress source code files in the /tmp/WordPress folder.

To get a copy of the WordPress source code (Git)

1. Download and install Git on your development machine.
2. In the /tmp/WordPress folder, call the git init command.
3. Call the git clone command to clone the public WordPress repository, making your own copy of it in the /tmp/WordPress destination folder:
   git clone https://github.com/WordPress/WordPress.git /tmp/WordPress
   This leaves you with a clean set of WordPress source code files in the /tmp/WordPress folder

Create scripts to run your application

Next, create a folder and scripts in the directory. CodeDeploy uses these scripts to set up and deploy your application revision on the target Amazon EC2 instance. You can use any text editor to create the scripts.

1. Create a scripts directory in your copy of the WordPress source code:
   mkdir -p /tmp/WordPress/scripts
2. Create an install_dependencies.sh file in /tmp/WordPress/scripts. Add the following lines to the file. This install_dependencies.sh script installs Apache, MySQL, and PHP. It also adds MySQL support to PHP.
   #!/bin/bash
   sudo amazon-linux-extras install php7.4
   sudo yum install -y httpd mariadb-server php

3. Create a start_server.sh file in /tmp/WordPress/scripts. Add the following lines to the file. This start_server.sh script starts Apache and MySQL.

```
#!/bin/bash
systemctl start mariadb.service
systemctl start httpd.service
systemctl start php-fpm.service
#!/bin/bash
systemctl start mariadb.service
systemctl start httpd.service
systemctl start php-fpm.service
```

4. Create a stop_server.sh file in /tmp/WordPress/scripts. Add the following lines to the file. This stop_server.sh script stops Apache and MySQL

```
#!/bin/bash
isExistApp=pgrep httpd
if [[ -n $isExistApp ]]; then
systemctl stop httpd.service
fi
isExistApp=pgrep mysqld
if [[ -n $isExistApp ]]; then
systemctl stop mariadb.service
fi
isExistApp=pgrep php-fpm
if [[ -n $isExistApp ]]; then
systemctl stop php-fpm.service
fi
```

5. Create a create_test_db.sh file in /tmp/WordPress/scripts. Add the following lines to the file. This create_test_db.sh script uses MySQL to create a test database for WordPress to use.

```
        #!/bin/bash
mysql -uroot <<CREATE_TEST_DB
CREATE DATABASE IF NOT EXISTS test;
CREATE_TEST_DB
```

6. Finally, create a change_permissions.sh script in /tmp/WordPress/scripts. This is used to change the folder permissions in Apache.

   This script updated permissions on the /tmp/WordPress folder so that anyone can write to it. This is required so that WordPress can write to its database during Step 5: Update and redeploy your WordPress application. After the WordPress application is set up, run the following command to update permissions to a more secure setting:

   chmod -R 755 /var/www/html/WordPress

   #!/bin/bash

   chmod -R 777 /var/www/html/WordPress

7. Give all of the scripts executable permissions. On the command line, type:

   chmod +x /tmp/WordPress/scripts/*

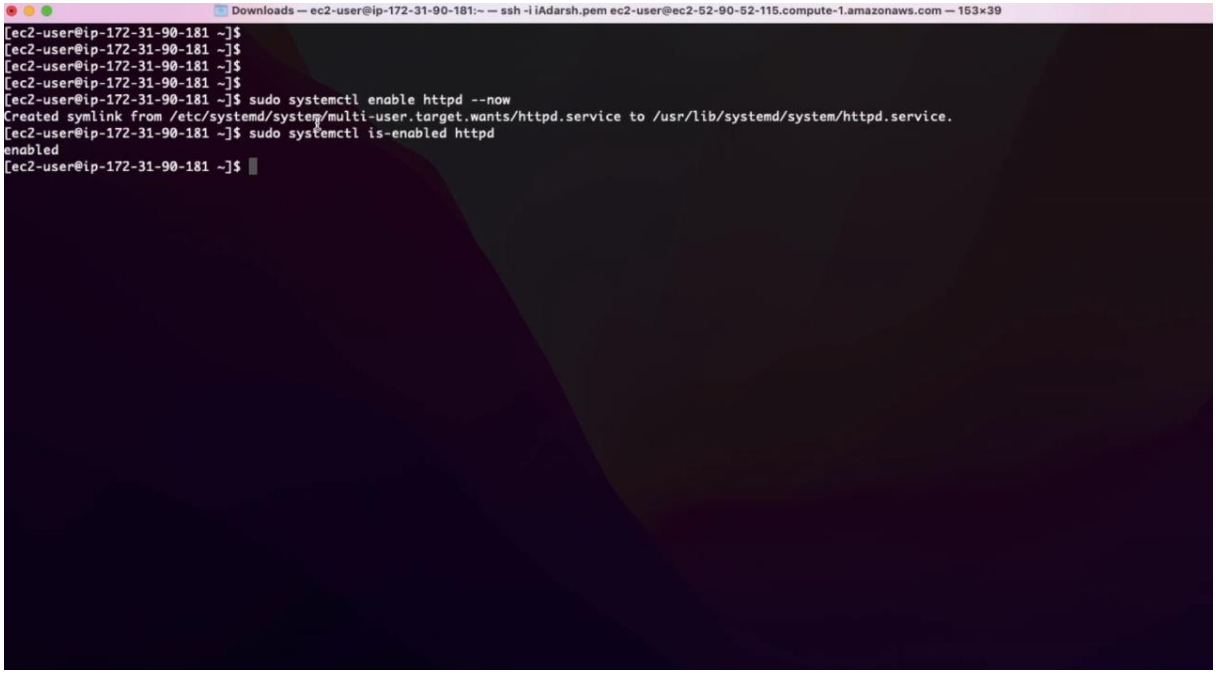


Fig 3.26 It has been enabled httpd

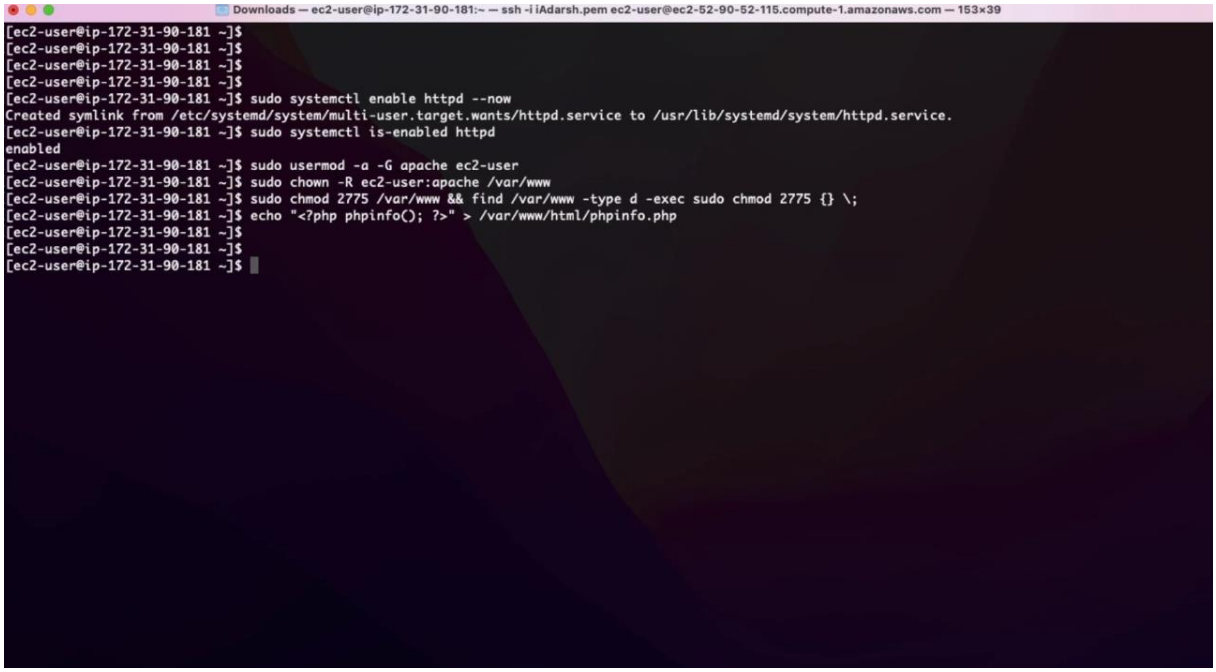


Fig 3.27 User name of apache EC2

Fig 3.28 Coping the php dns address

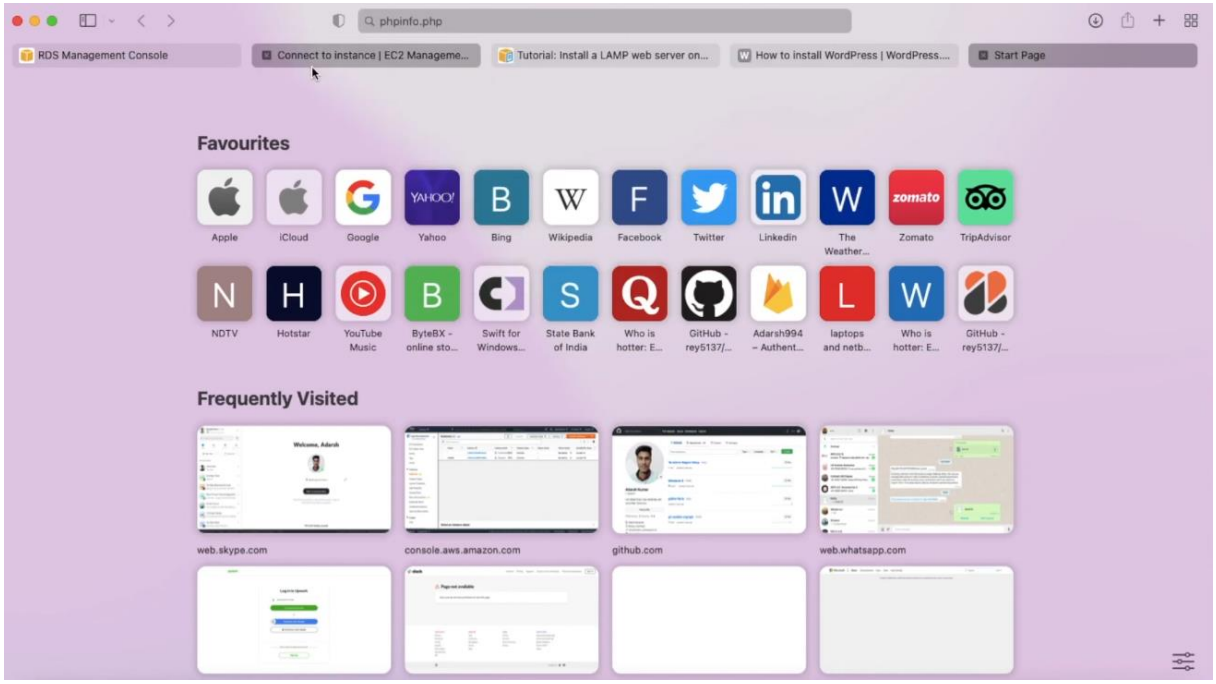


Fig 3.29 Paste the copied dns address in browser

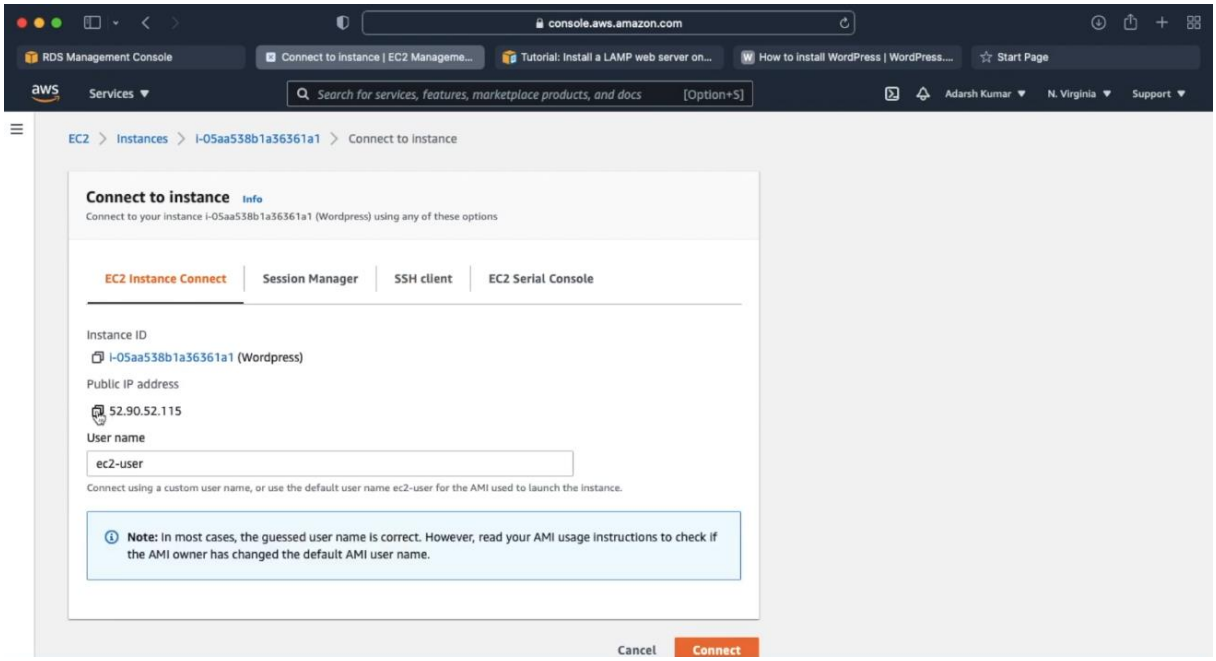


Fig 3.30 Copy the Ip address of EC2 instance

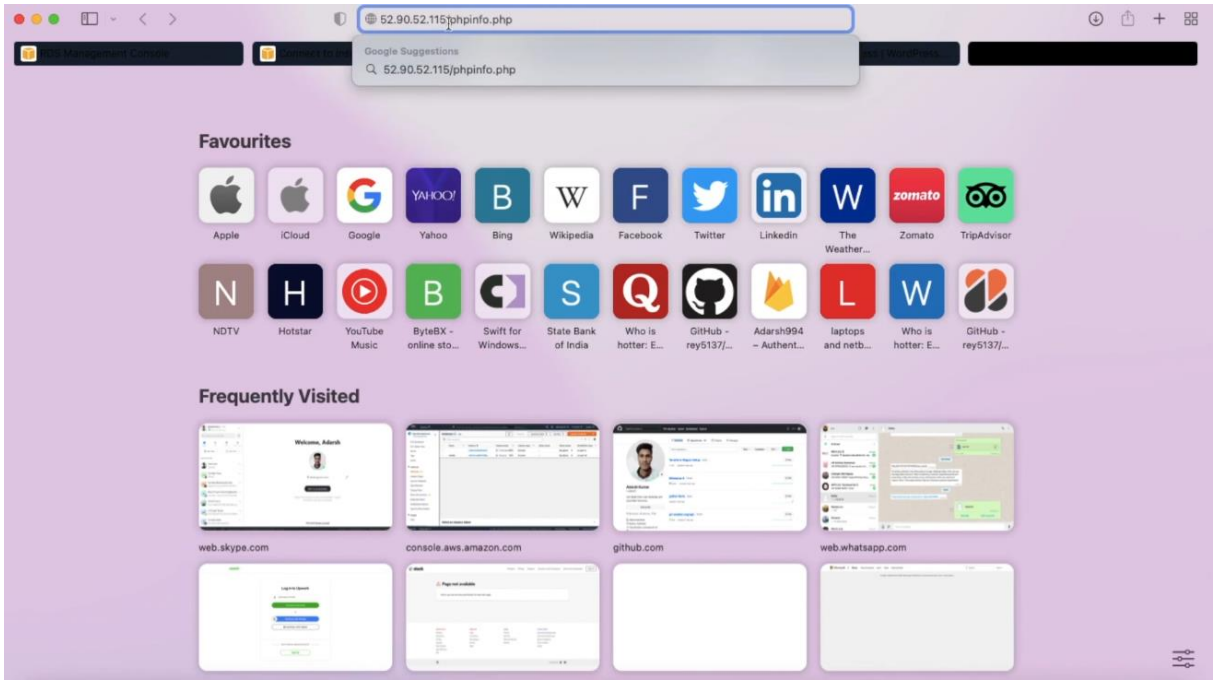


Fig 3.31 paste the EC2 Ip address in the same bar of dns address connect through slash (/)

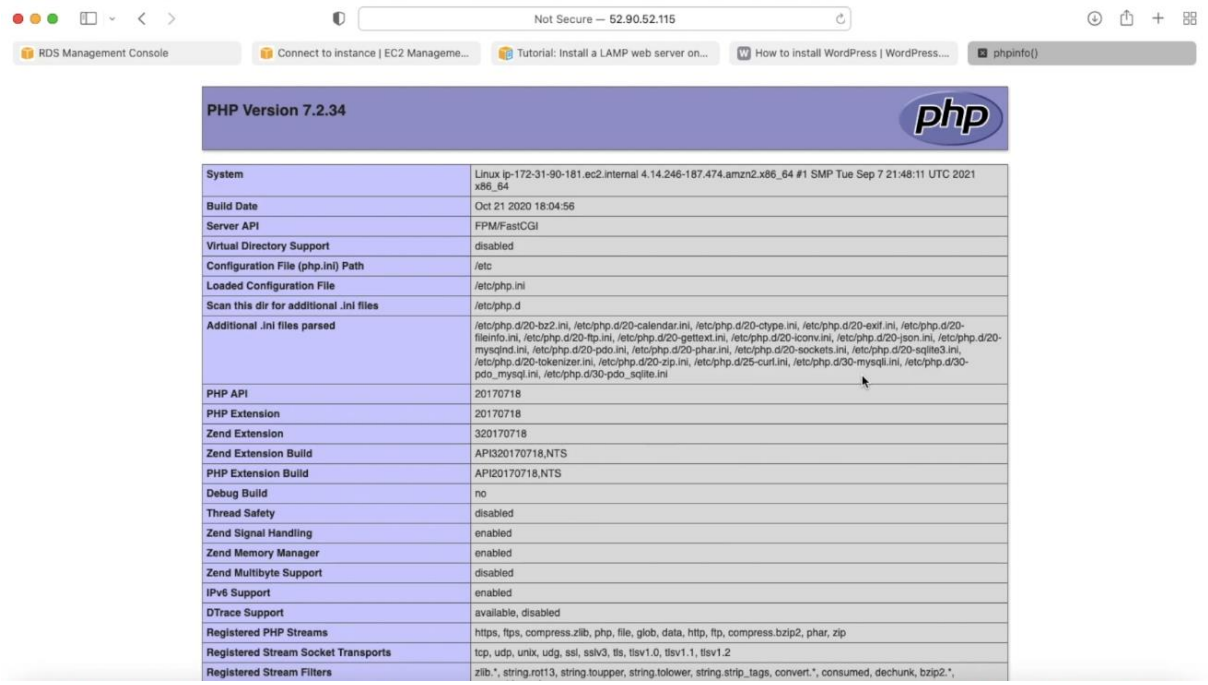


Fig 3.31 The output of php version

Add an application specification file

Next, add an application specification file (AppSpec file), a YAML-formatted file used by CodeDeploy to:

- Map the source files in your application revision to their destinations on the target Amazon EC2 instance.
- Specify custom permissions for deployed files.
- Specify scripts to be run on the target Amazon EC2 instance during the deployment.

  The AppSpec file must be named appspec.yml. It must be placed in the root directory of the application's source code. In this tutorial, the root directory is /tmp/WordPress

  With your text editor, create a file named appspec.yml. Add the following lines to the file:

version: 0.0

os: linux

files:

```
  - source: /
    destination: /var/www/html/WordPress
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  AfterInstall:
    - location: scripts/change_permissions.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
    - location: scripts/create_test_db.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

CodeDeploy uses this AppSpec file to copy all of the files in the /tmp/WordPress folder on the development machine to the /var/www/html/WordPress folder on the target Amazon EC2 instance. During the deployment, CodeDeploy runs the specified scripts as root in the /var/www/html/WordPress/scripts folder on the target Amazon EC2 instance at specified events during the deployment lifecycle, such as BeforeInstall and AfterInstall. If any of these scripts take longer than 300 seconds (5 minutes) to run, CodeDeploy stops the deployment and marks the deployment as failed.

The locations and numbers of spaces between each of the items in this file are important. If the spacing is incorrect, CodeDeploy raises an error that might be difficult to debug. For more information, see AppSpec File spacing.

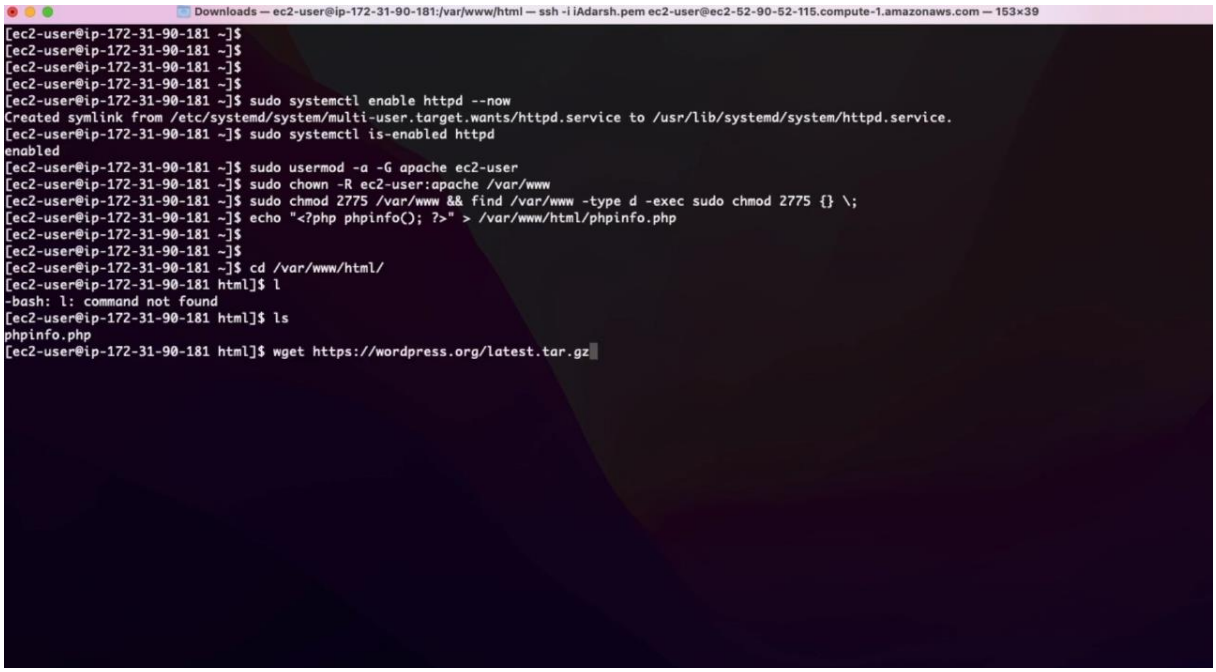


Fig 3.32 Dowlording link of WordPress

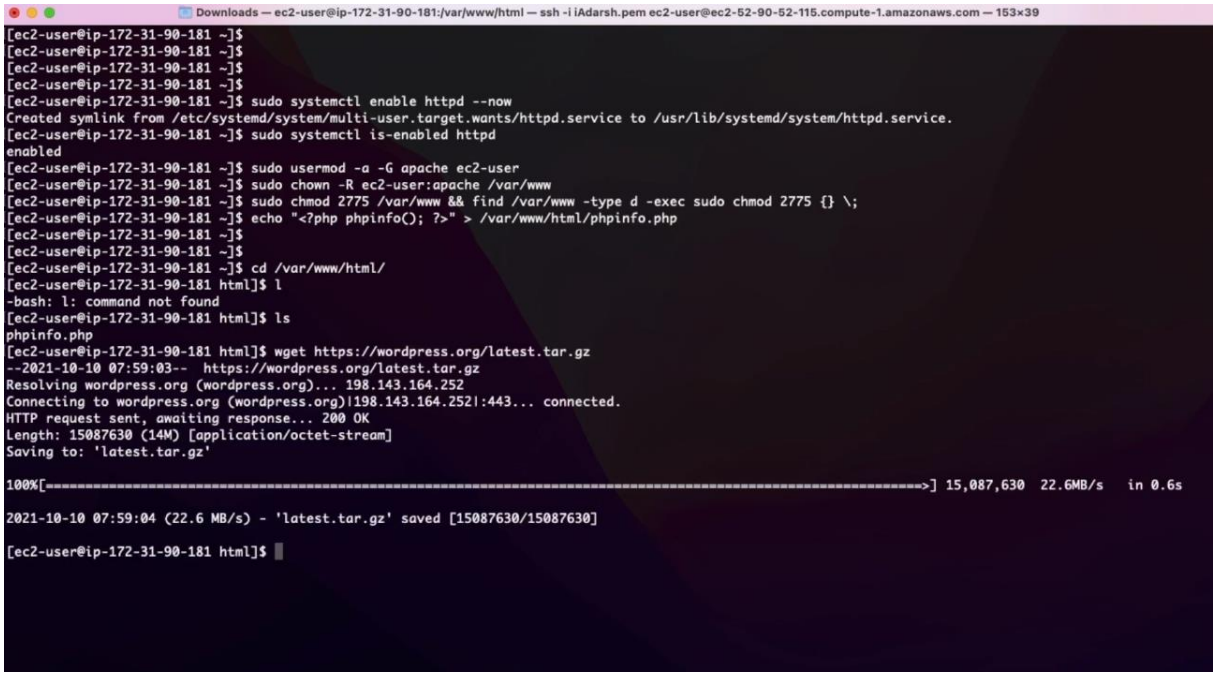


Fig 3.33 Installing the WordPress

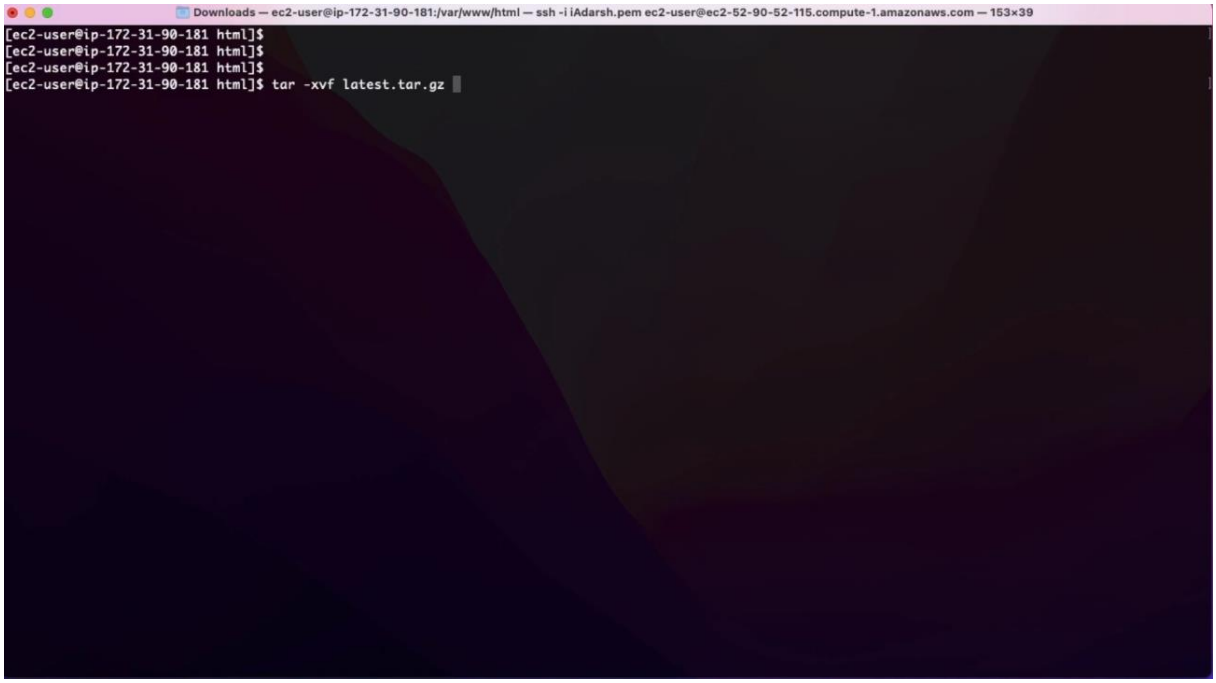


Fig 3.34 Installing weather is latest tar

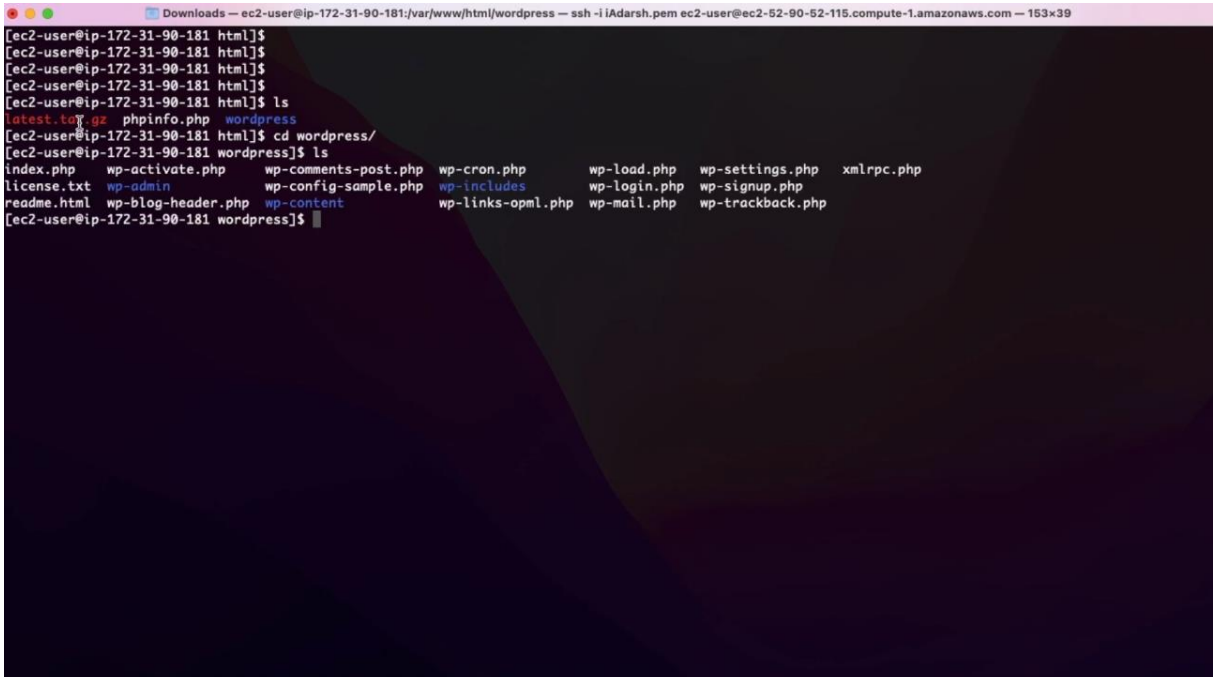


Fig 3.35 Installed the latest tar

3.3 CONNECTING TO AN RDS CUSTOM DB INSTANCE USING AWS SYSTEMS MANAGER

After you create your RDS Custom DB instance, you can connect to it using AWS Systems Manager Session Manager. Session Manager is a Systems Manager capability that you can use to manage Amazon EC2 instances through a browser-based shell or through the AWS CLI.

Console:

To connect to your DB instance using Session Manager

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose Databases, and then choose the RDS Custom DB instance to which you want to connect.
3. Choose Configuration.
4. Note the Resource ID value for your DB instance. For example, the resource ID might be db-ABCDEFGHIJKLMNOPQRS0123456.
5. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
6. In the navigation pane, choose Instances.
7. Look for the name of your EC2 instance, and then choose the instance ID associated with it. For example, the instance ID might be i-abcdefghijklm01234.
8. Choose Connect.
9. Choose Session Manager.
10. Choose Connect.

    A window opens for your session.

CODE:

You can connect to your RDS Custom DB instance using the AWS CLI. This technique requires the Session Manager plugin for the AWS CLI. To learn how to install the plugin, see Install the Session Manager plugin for the AWS CLI.

To find the DB resource ID of your RDS Custom DB instance, use describe-db-instances

```
aws rds describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DbiResourceId]' \
    --output text
```

The following sample output shows the resource ID for your RDS Custom instance. The prefix is db-.

```
db-ABCDEFGHIJKLMNOPQRS0123456
```

To find the EC2 instance ID of your DB instance, use aws ec2 describe-instances. The following example uses db-ABCDEFGHIJKLMNOPQRS0123456 for the resource ID.

```
aws ec2 describe-instances \
    --filters "Name=tag:Name,Values=db-ABCDEFGHIJKLMNOPQRS0123456" \
    --output text \
    --query 'Reservations[*].Instances[*].InstanceId'
```

The following sample output shows the EC2 instance ID.

```
i-abcdefghijklm01234
```

Use the aws ssm start-session command, supplying the EC2 instance ID in the --target parameter.

```
aws ssm start-session --target "i-abcdefghijklm01234"
```

A successful connection looks like the following.

```
Starting session with SessionId: yourid-abcdefghijklm1234
[ssm-user@ip-123-45-67-89 bin]$
```

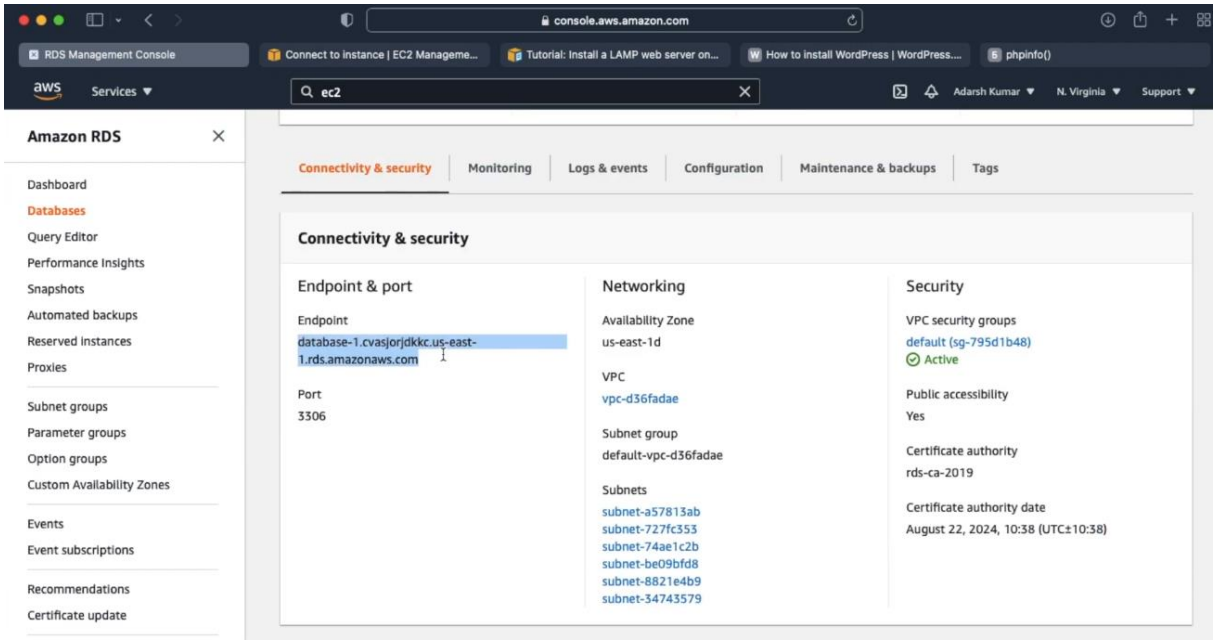


Fig 3.36 Coping the database end point address

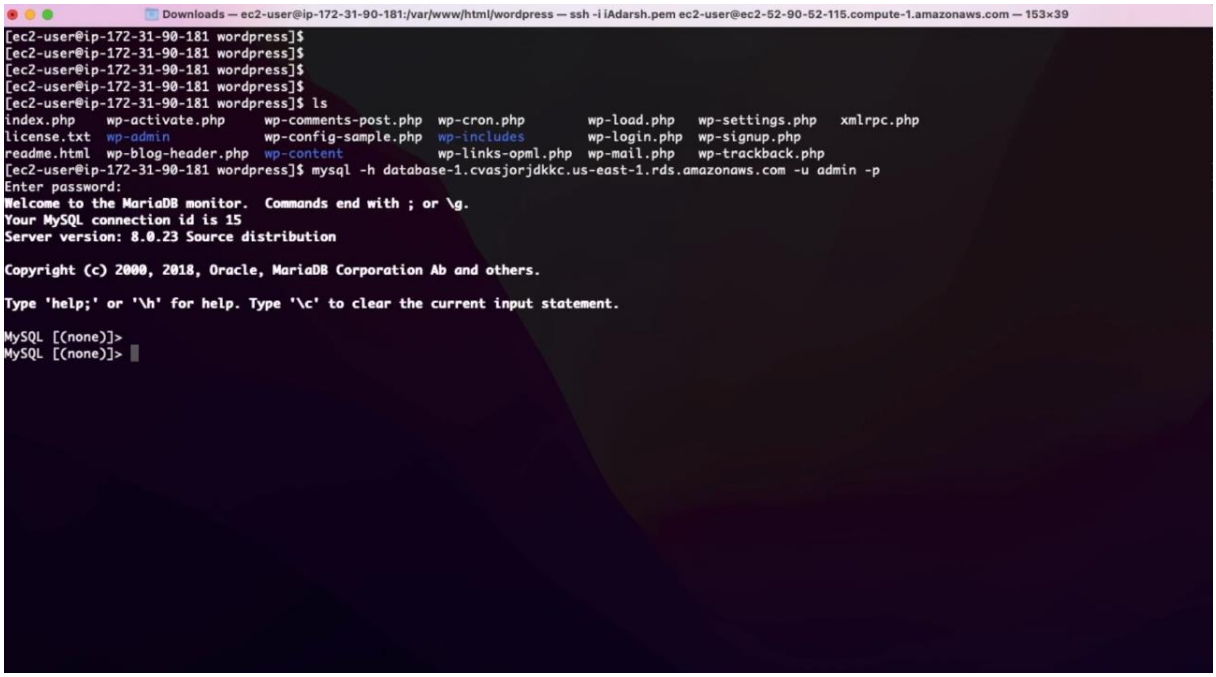


Fig 3.37 Paste the database end point address and give user name and password

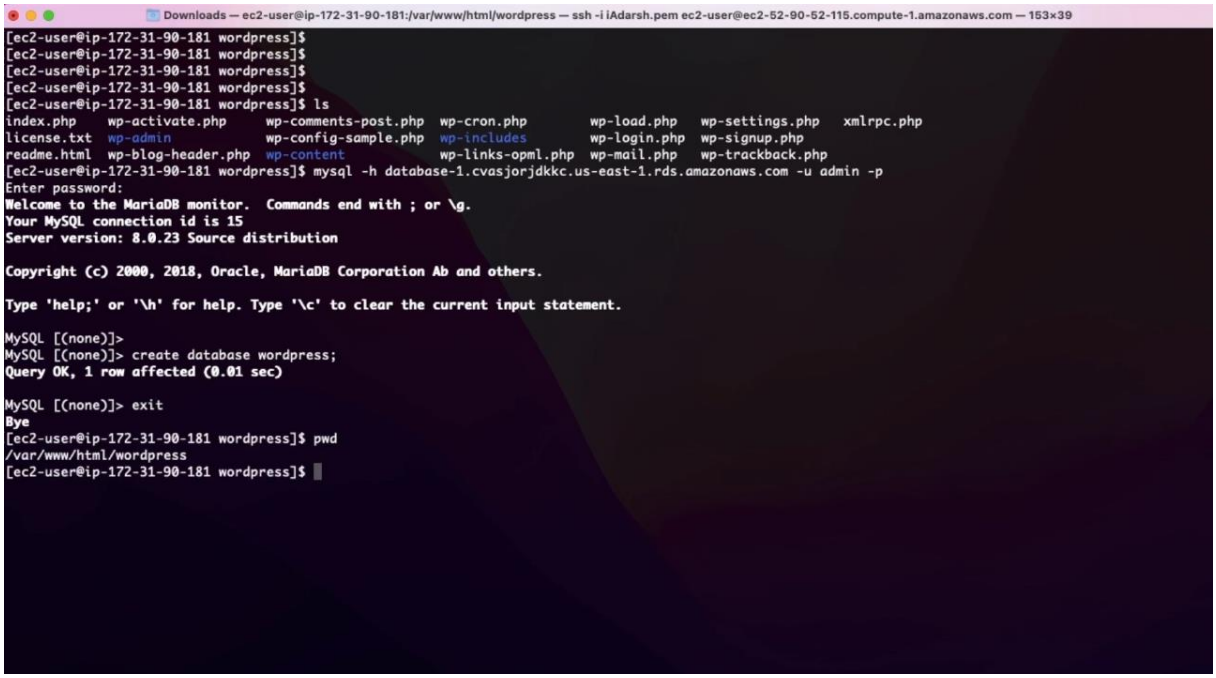


Fig 3.38 Creating a database and give name as WordPress

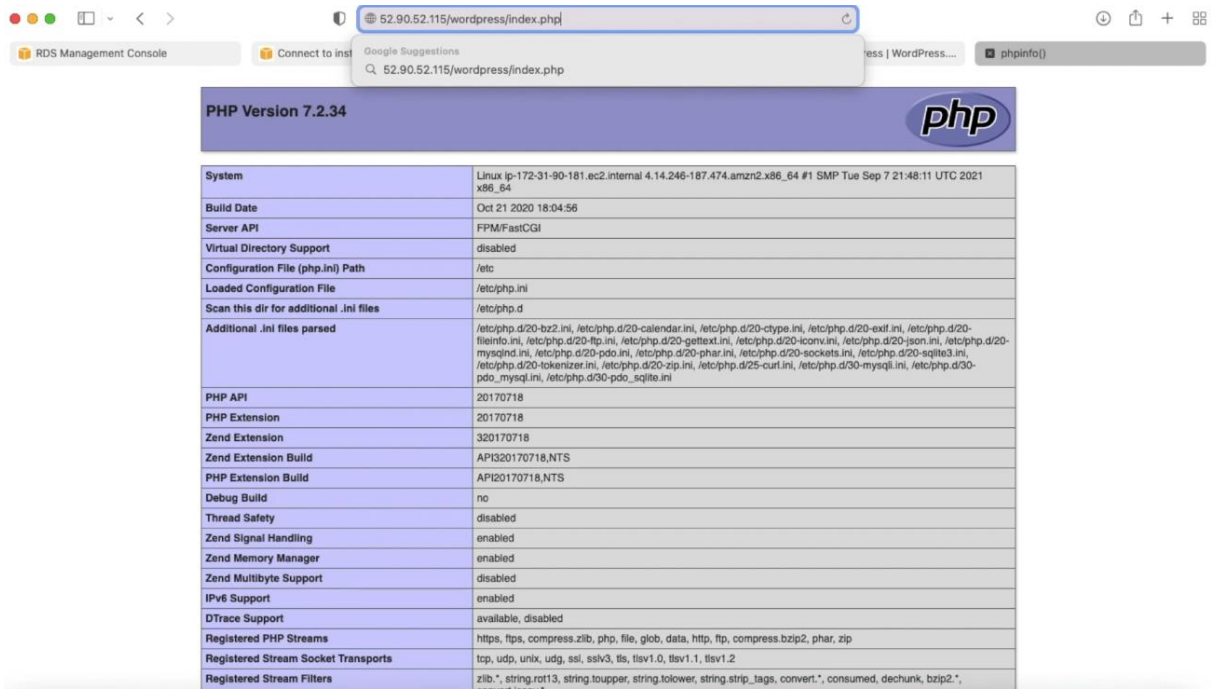


Fig 3.39 After the ip name we need to provide path of index (it is word press)

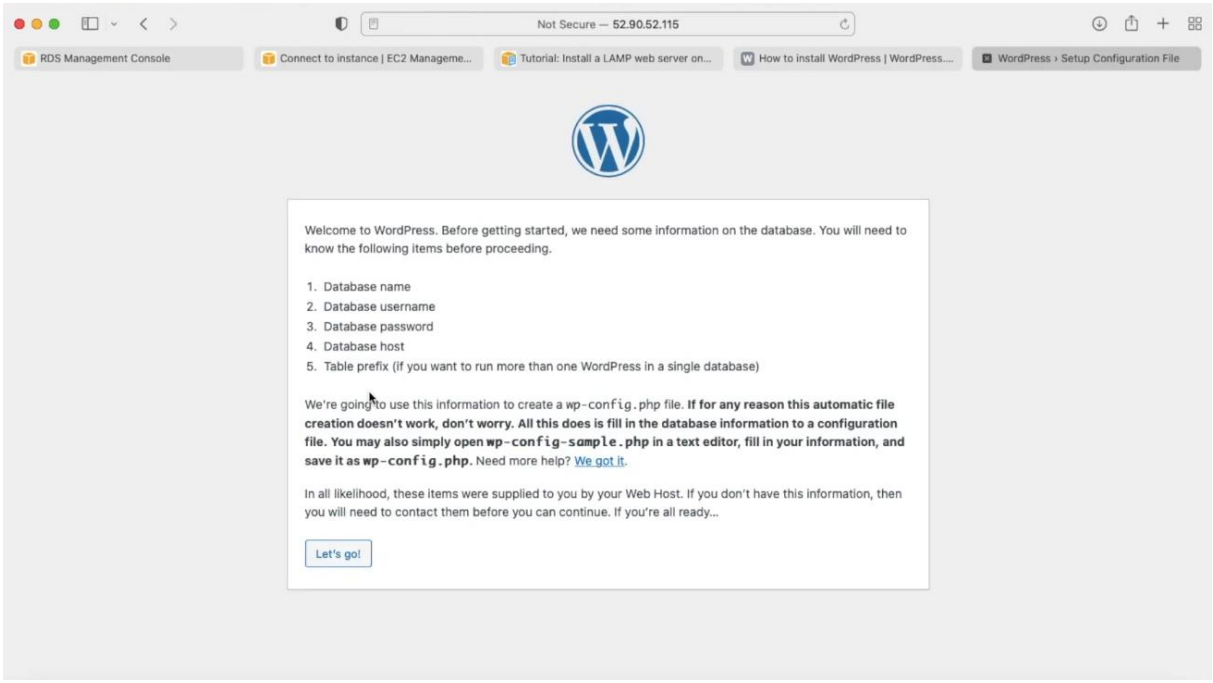


Fig 3.40 The WordPress is working and we need to connect WordPress with rds

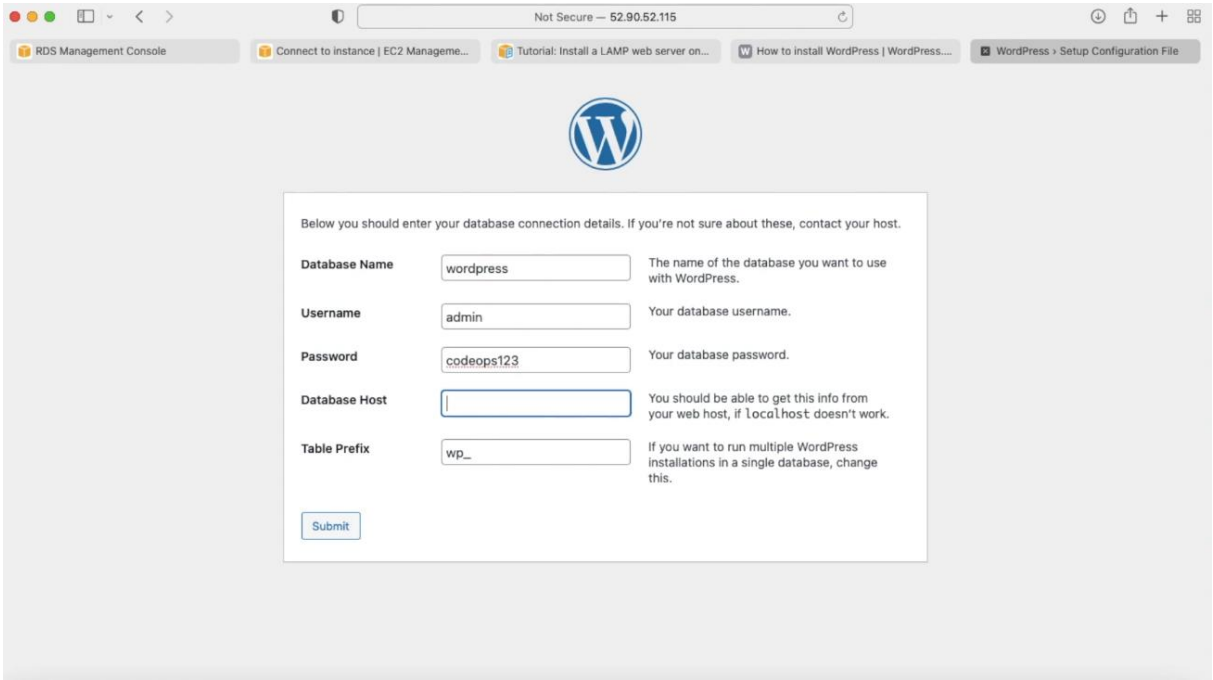


Fig 3.41 Giving details

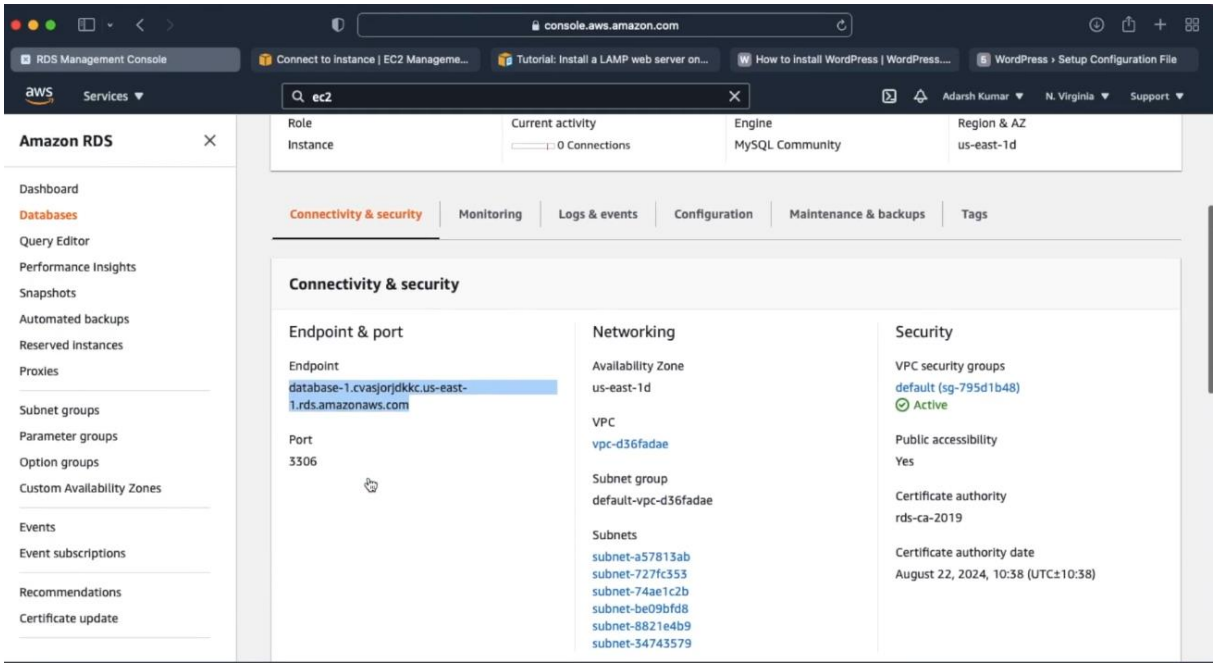


Fig 3.42 In login page database host means we need to give database endpoint

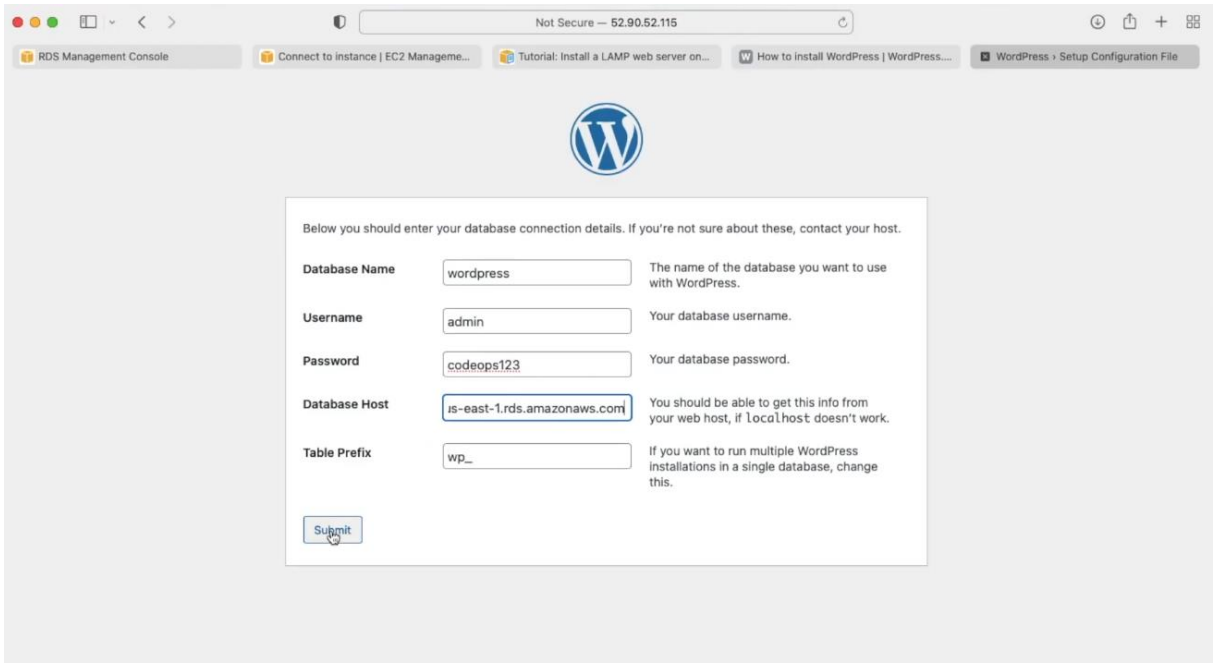


Fig 3.43 Submitting the WordPress details

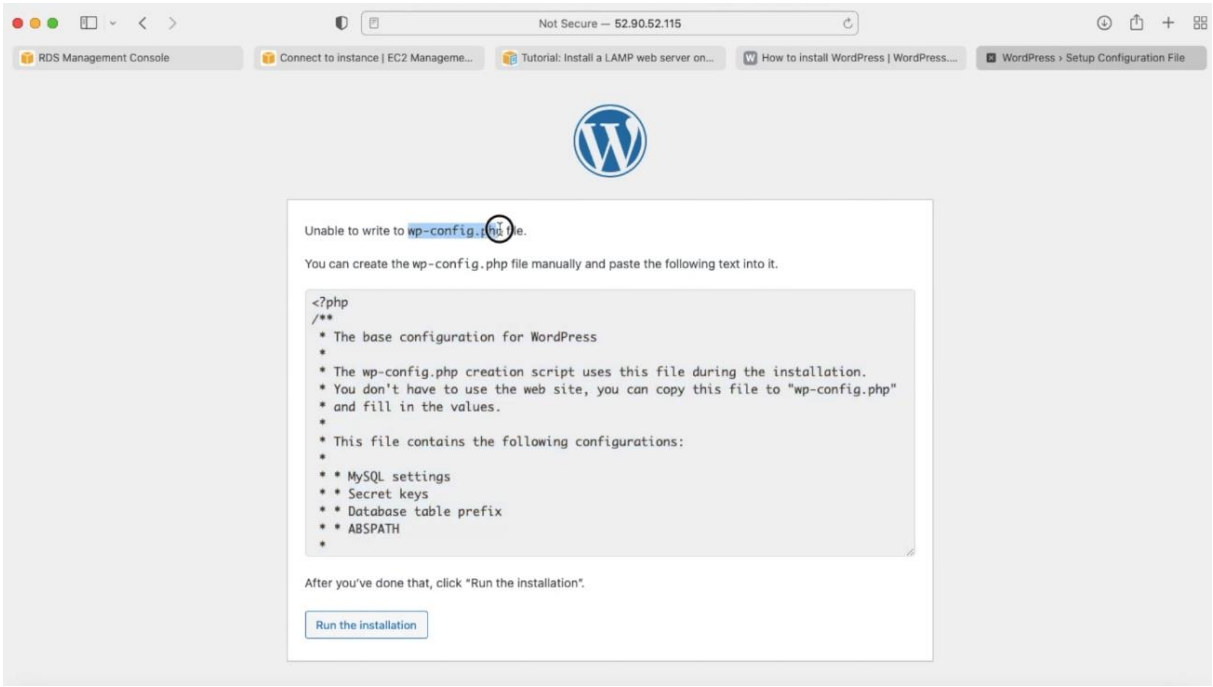


Fig 3.44 The WordPress is connected to rds and we need store the configuration

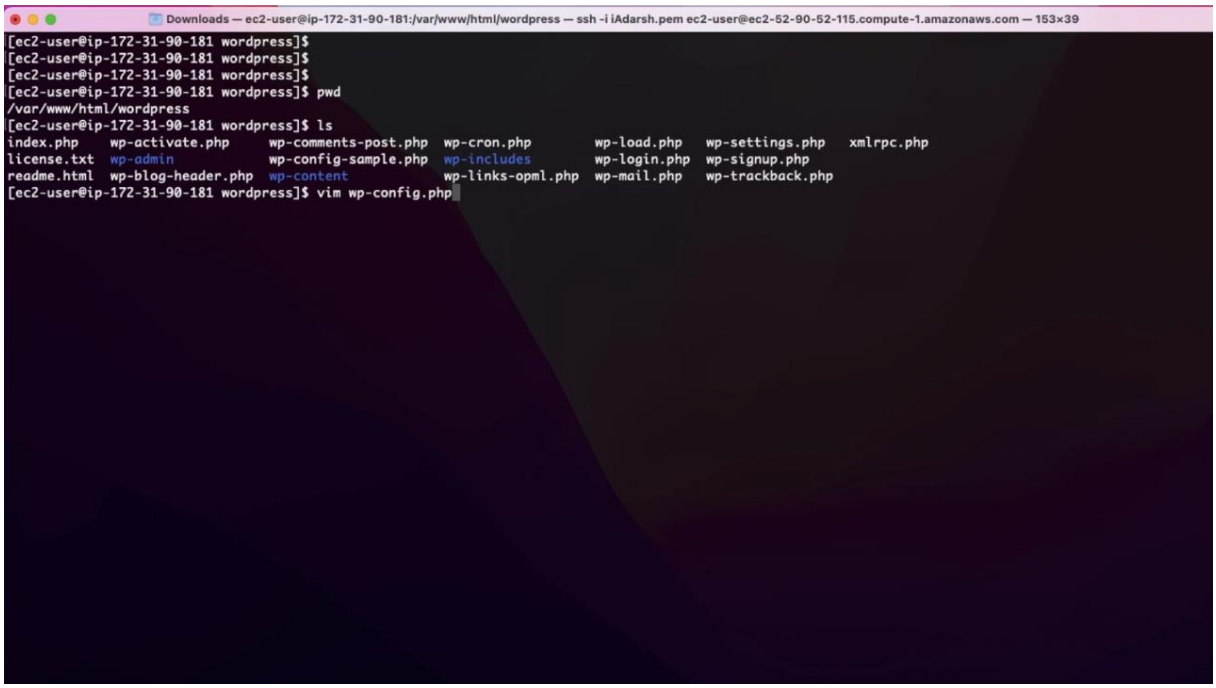


Fig 3.45 Create a file with stored configuration

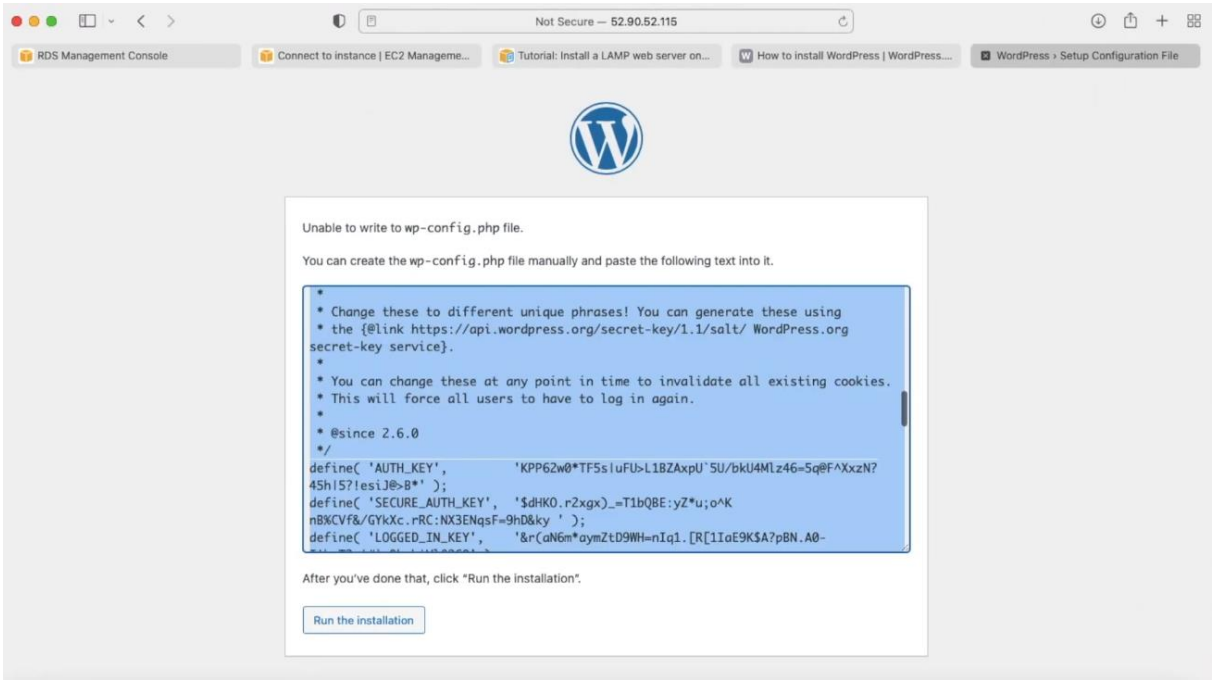


Fig 3.46 Copy the content of the configuration file

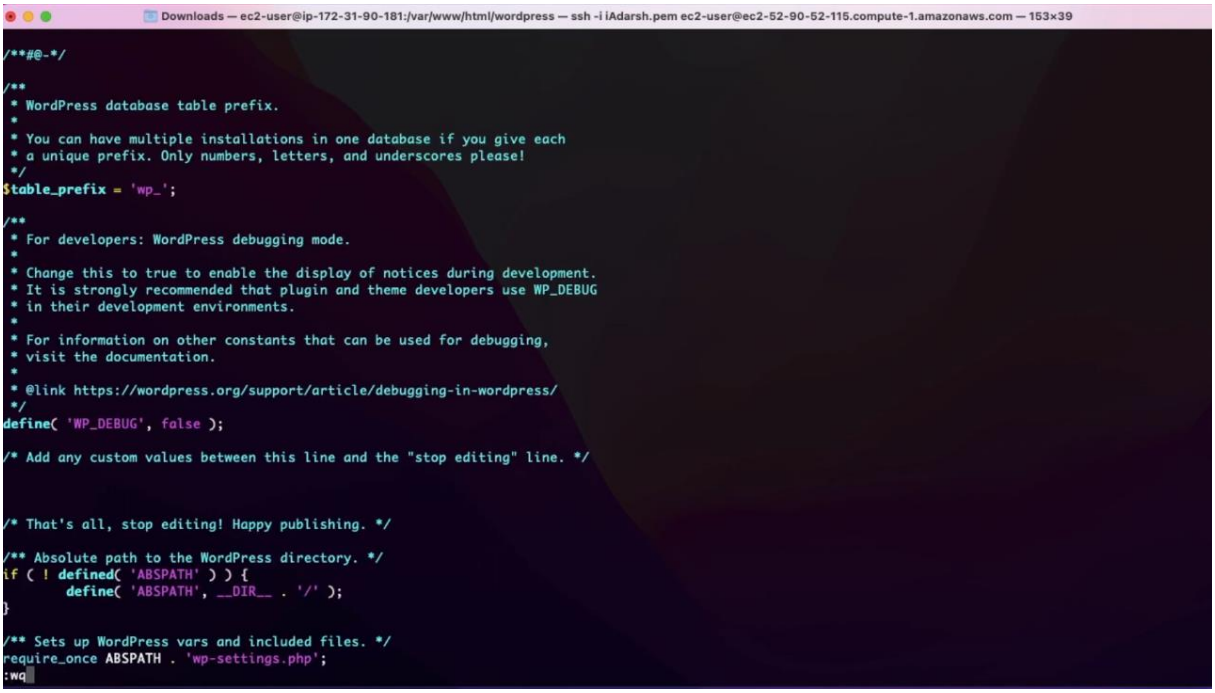


Fig 3.47 Paste the configuration file in the command

Fig 3.48 The word press configuration has been saved in directory path and click on run the installation

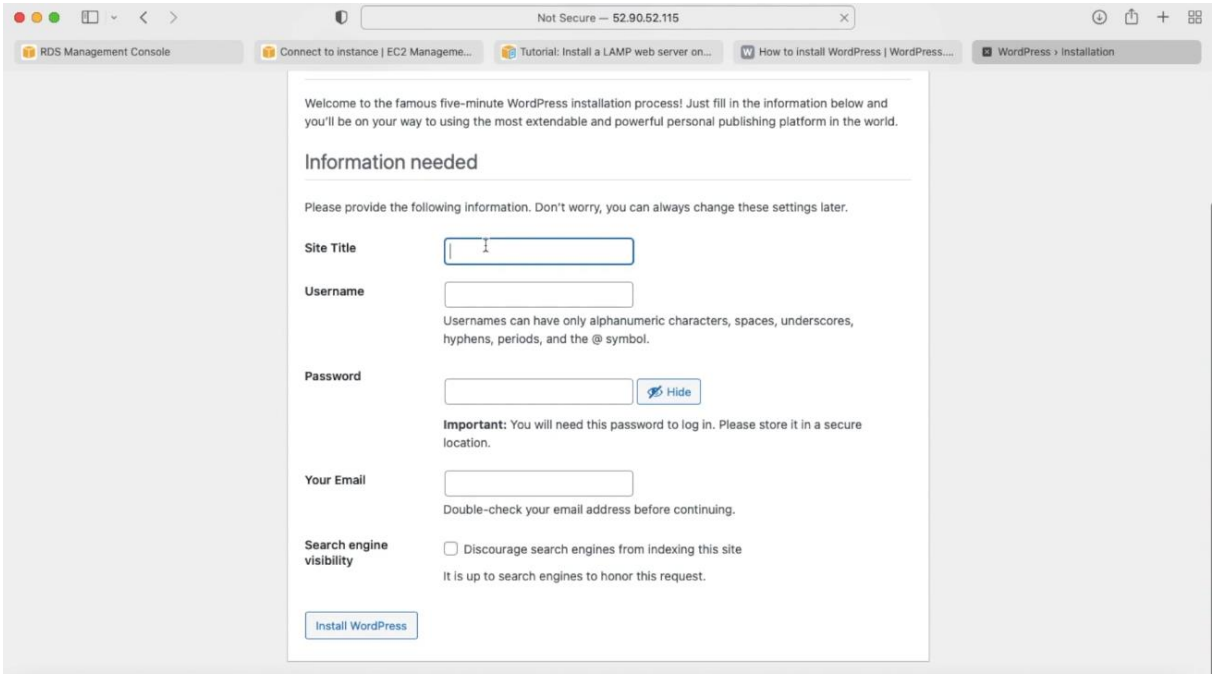


Fig 3.49 Asking for some more details we have to fill it and install the WordPress

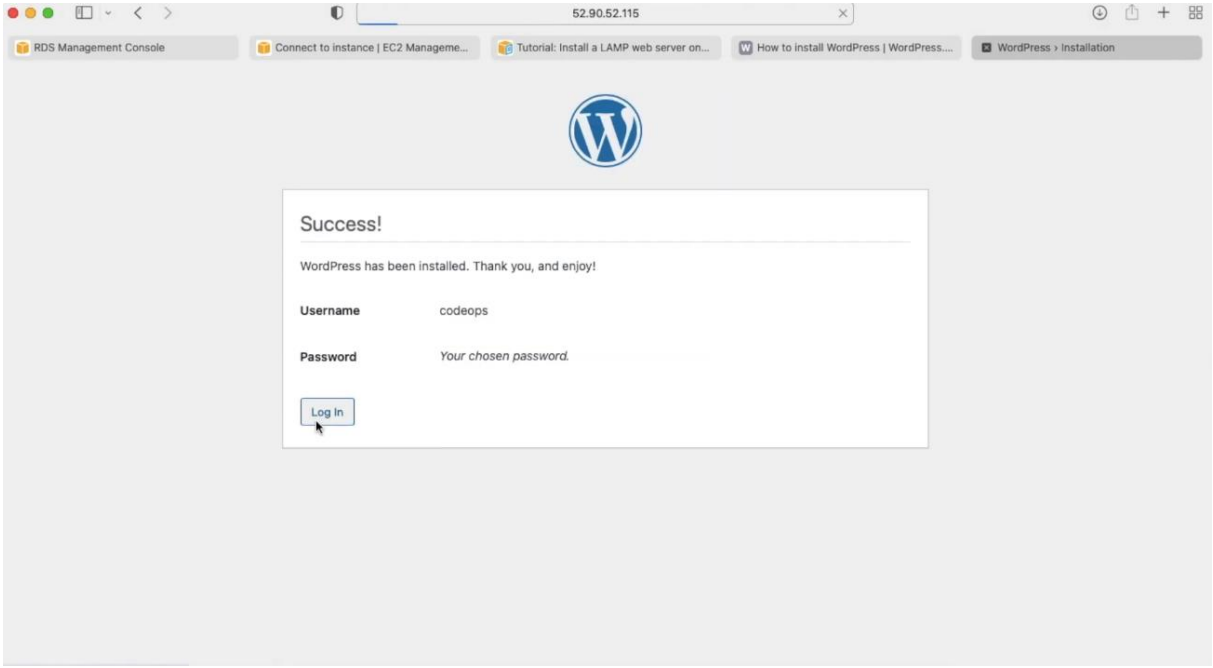


Fig 3.50 It has been succussed and log in through username and password

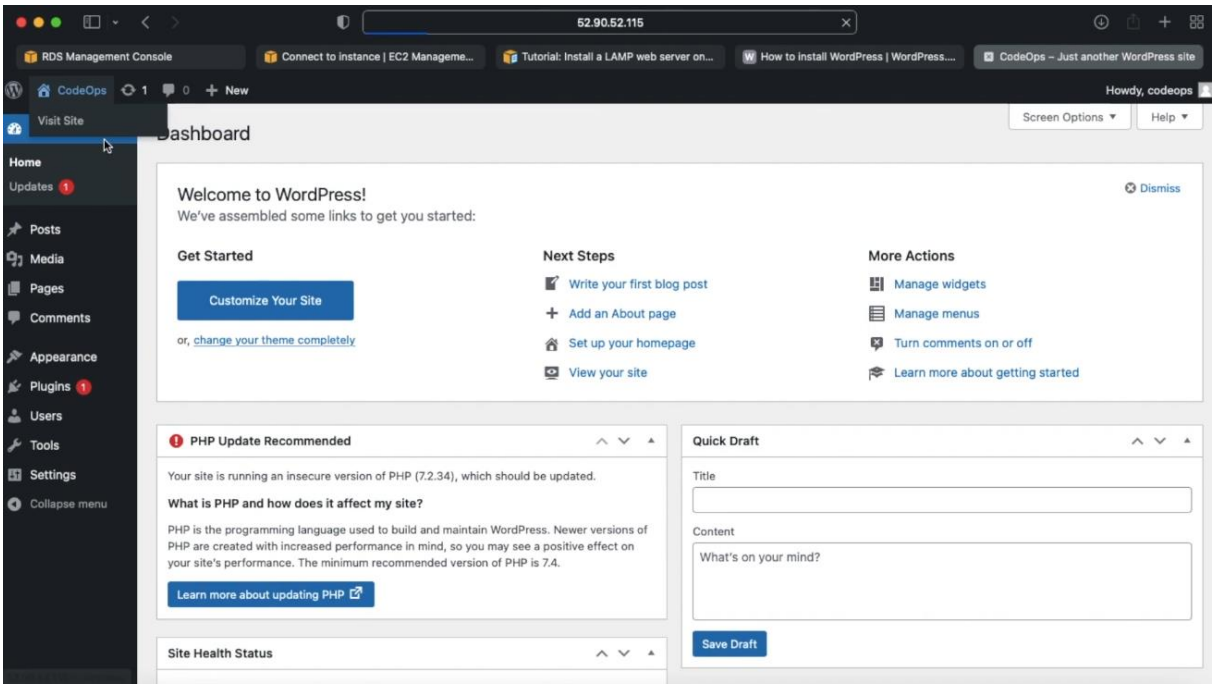


Fig 3.51 WordPress admin panel is opened click on our website

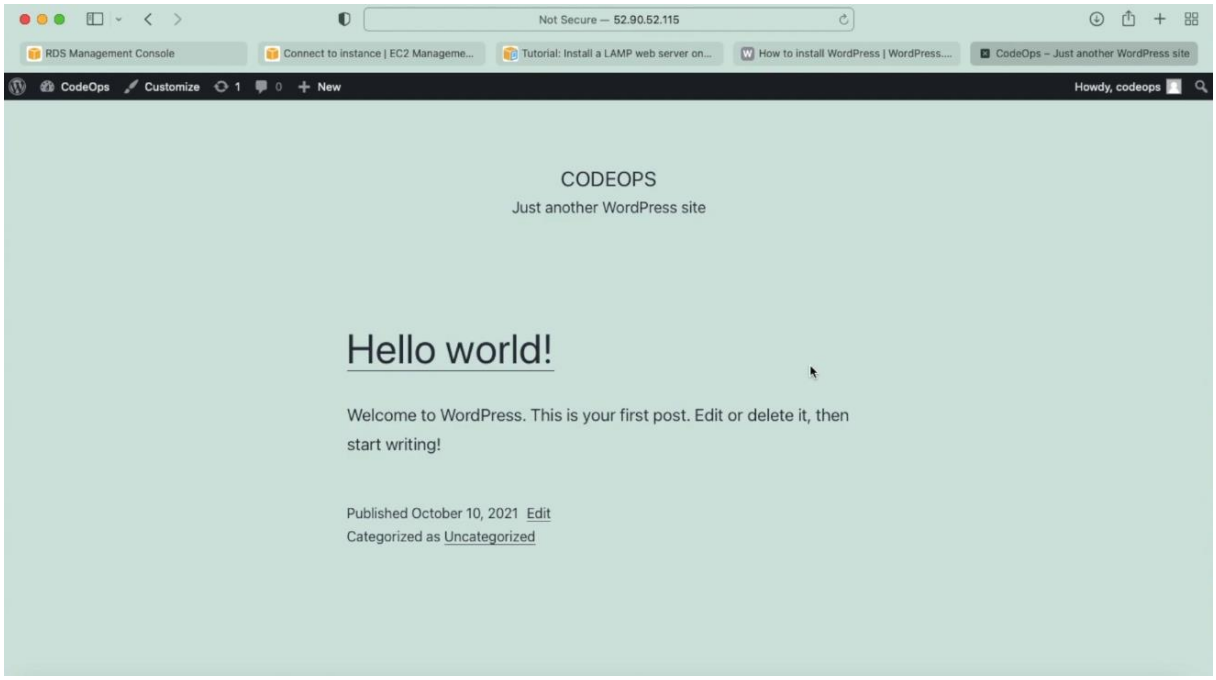


Fig 3.52 The WordPress website by default created by WordPress

CLEAN UP YOUR WORDPRESS APPLICATION AND RELATED RESOURCES

You've now successfully made an update to the WordPress code and redeployed the site. To avoid ongoing charges for resources you created for this tutorial, you should delete:

- Any AWS CloudFormation stacks (or terminate any Amazon EC2 instances, if you created them outside of AWS CloudFormation).
- Any Amazon S3 buckets.
- The WordPress_App application in CodeDeploy.
- The AWS Systems Manager State Manager association for the CodeDeploy agent.

  You can use the AWS CLI, the AWS CloudFormation, Amazon S3, Amazon EC2, and CodeDeploy consoles, or the AWS APIs to perform the cleanup.

Topics

- To clean up resources (CLI)
- To clean up resources (console)

To clean up resources (CLI)

1. If you used our AWS CloudFormation template for this tutorial, call the delete-stack command against the stack named CodeDeployDemoStack. This will terminate all accompanying Amazon EC2 instances and delete all accompanying IAM roles the stack created:

   aws cloud formation delete-stack --stack-name CodeDeployDemoStack

2. To delete the Amazon S3 bucket, call the rm command with the --recursive switch against the bucket named codedeploydemobucket. This will delete the bucket and all objects in the bucket:

   aws s3 rm s3://codedeploydemobucket –recursive

3. To delete the WordPress_App application, call the delete-application command. This will also delete all associated deployment group records and deployment records for the application:

   aws deploy delete-application --application-name WordPress_App

4. To delete the Systems Manager State Manager association, call the delete-association command.

   aws ssm delete-association --association-id association-id

   You can get the *association-id* by calling the describe-association command.

   aws ssm describe-association --name AWS-ConfigureAWSPackage --targets Key=tag: Name, Values=CodeDeployDemo

If you did not use the AWS CloudFormation stack for this tutorial, call the terminate-instances command to terminate any Amazon EC2 instances you manually created. Supply the ID of the Amazon EC2 instance to terminate:

aws ec2 terminate-instances --instance-ids instance Id

To clean up resources (console)

If you used our AWS CloudFormation template for this tutorial, delete the associated AWS CloudFormation stack.

1. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. In the Filter box, type the AWS CloudFormation stack name you created earlier (for example, CodeDeployDemoStack).
3. Select the box beside stack name. In the Actions menu, choose Delete Stack.

   AWS CloudFormation deletes the stack, terminates all accompanying Amazon EC2 instances, and deletes all accompanying IAM roles.

To terminate Amazon EC2 instances you created outside of an AWS CloudFormation stack:

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the INSTANCES list, choose Instances.
3. In the search box, type the name of the Amazon EC2 instance you want to terminate (for example, CodeDeployDemo), and then press Enter.
4. Choose the Amazon EC2 instance name.
5. In the Actions menu, point to Instance State, and then choose Terminate. When prompted, choose Yes, Terminate.

   Repeat these steps for each instance.

To delete the WordPress App application from Code Deploy:

1. Sign in to the AWS Management Console and open the Code Deploy console athttps://console.aws.amazon.com/codedeploy
2. In the navigation pane, expand Deploy, then choose Applications.
3. In the list of applications, choose WordPress_App.
4. On the Application details page, choose Delete application.
5. When prompted, enter the name of the application to confirm you want to delete it, and then choose Delete.

To delete the Systems Manager State Manager association:

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager.
2. In the navigation pane, choose State Manager.
3. Choose the association you created and choose Delete

RESULT SCREENSHOTS

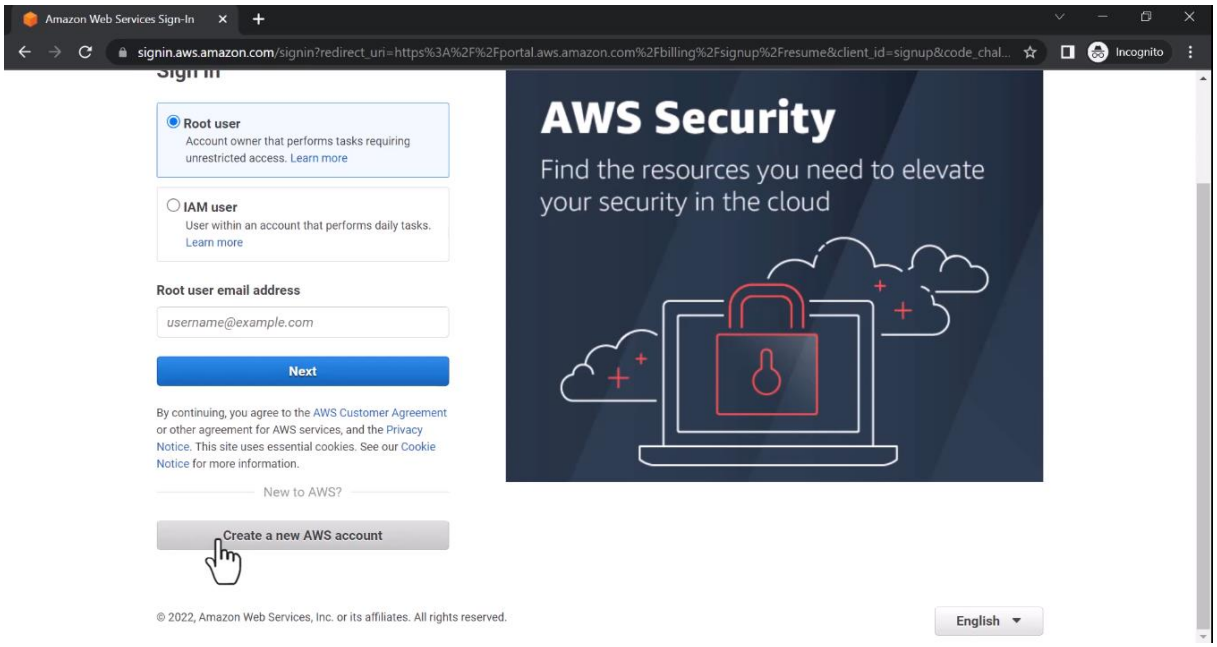


Fig 5.1 Creating a new amazon account

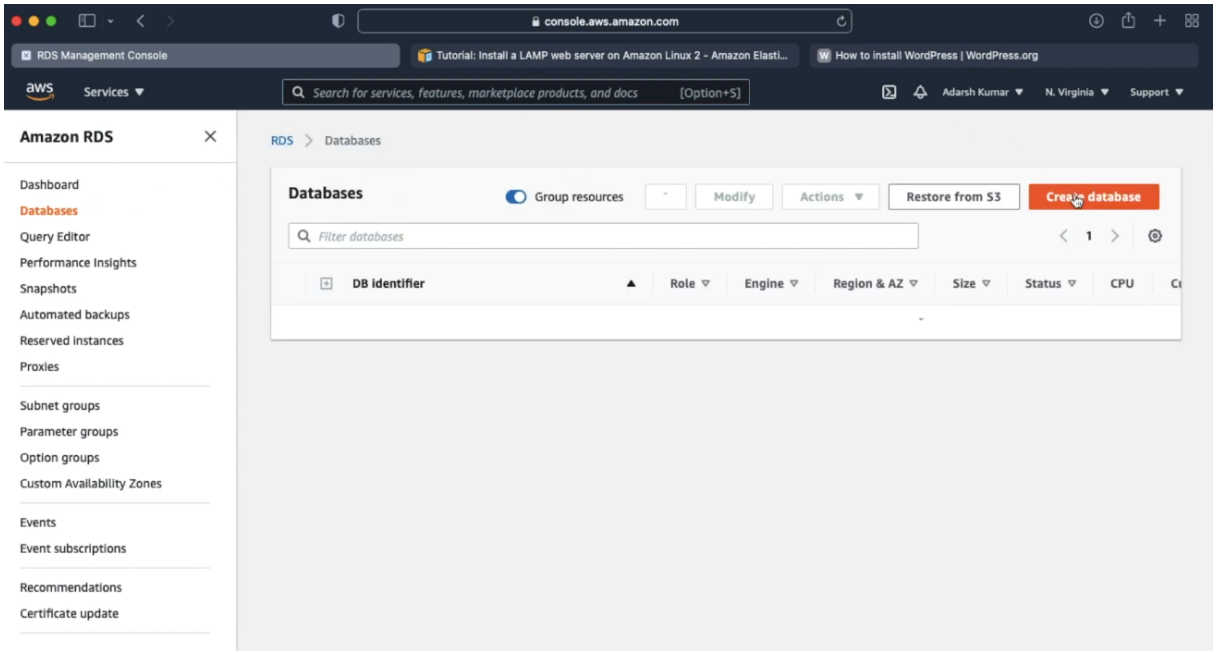


Fig 5.2 Creating a amazon rds database

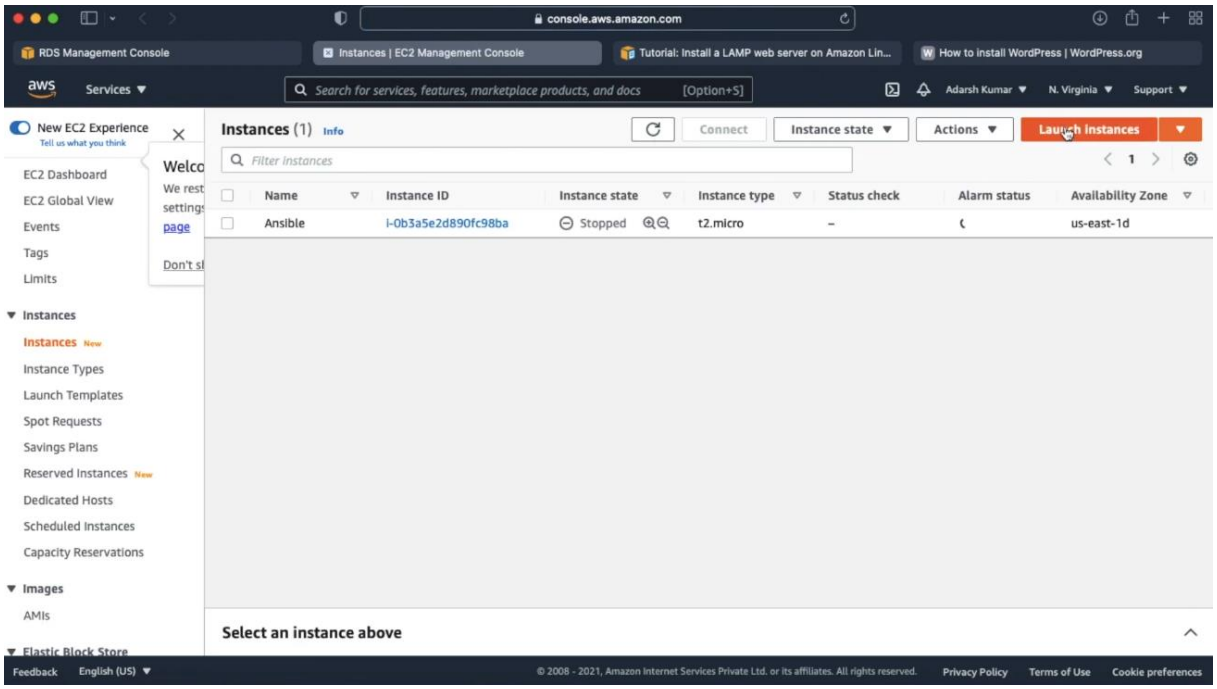


Fig 5.3 Launching the EC2 instance

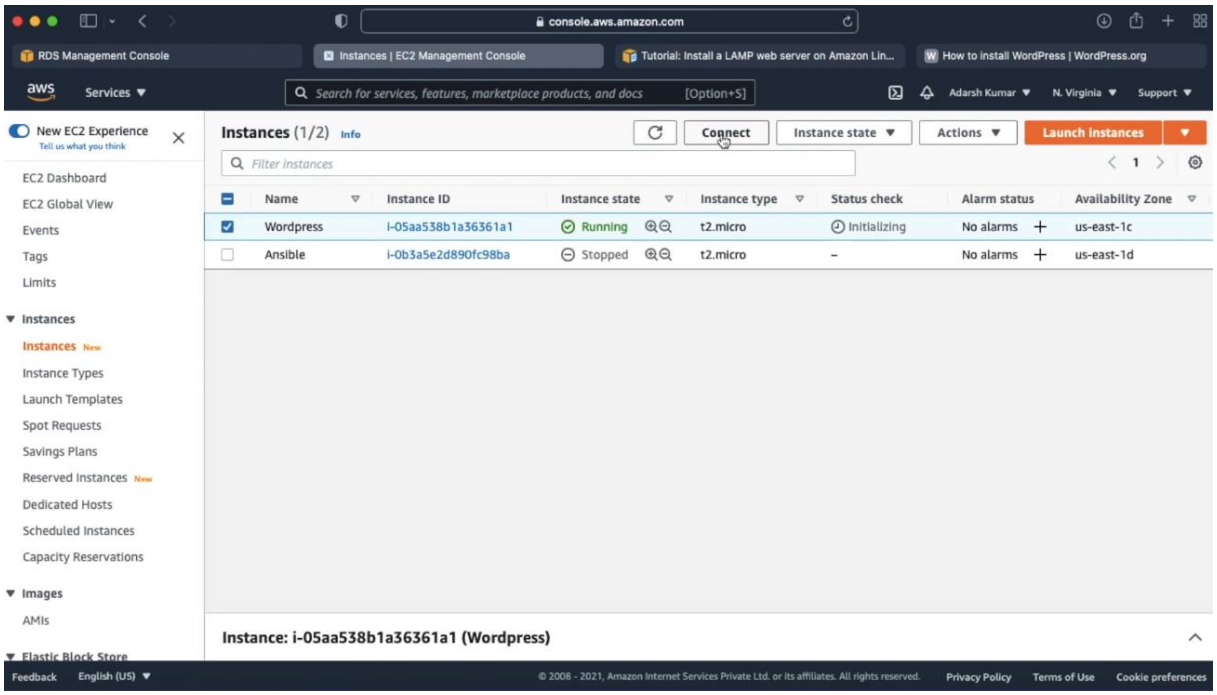


Fig 5.4 Creating the WordPress instance and click on connect instance

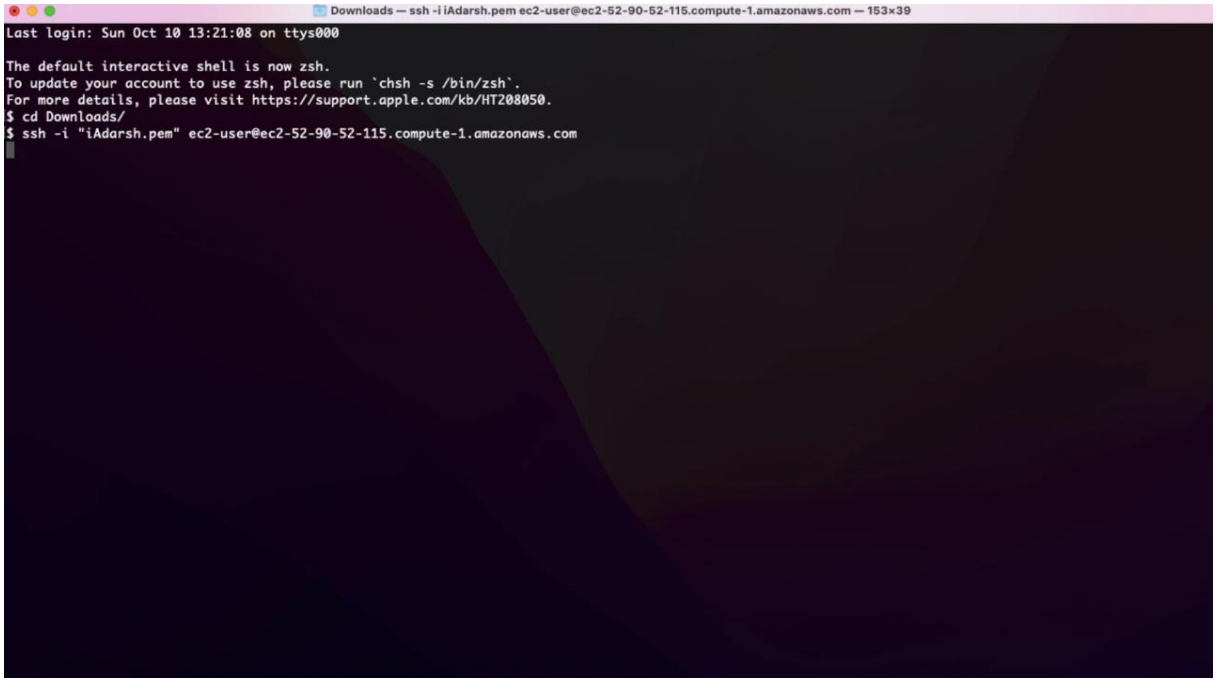


Fig 5.5 First giving directory (downloads), Giving client sight(ssh) then client name(Adarsh) and Ip address

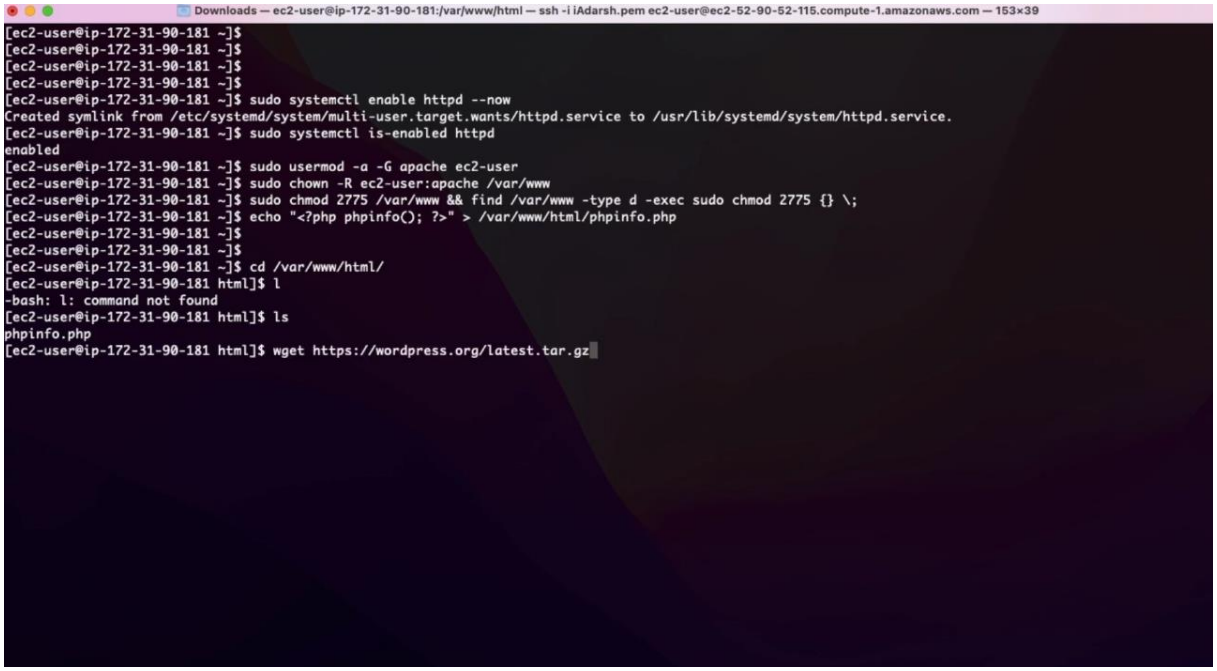


Fig 5.6 Dowlording link of WordPress

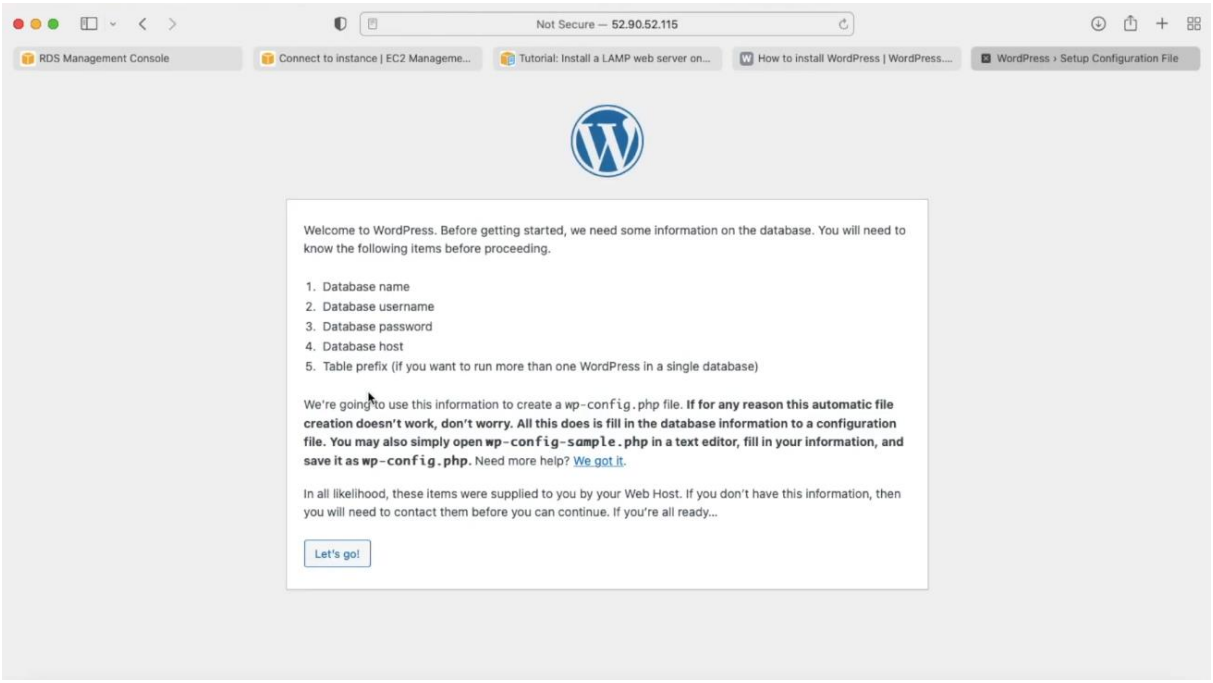


Fig 5.7 The WordPress is working and we need to connect WordPress with rds

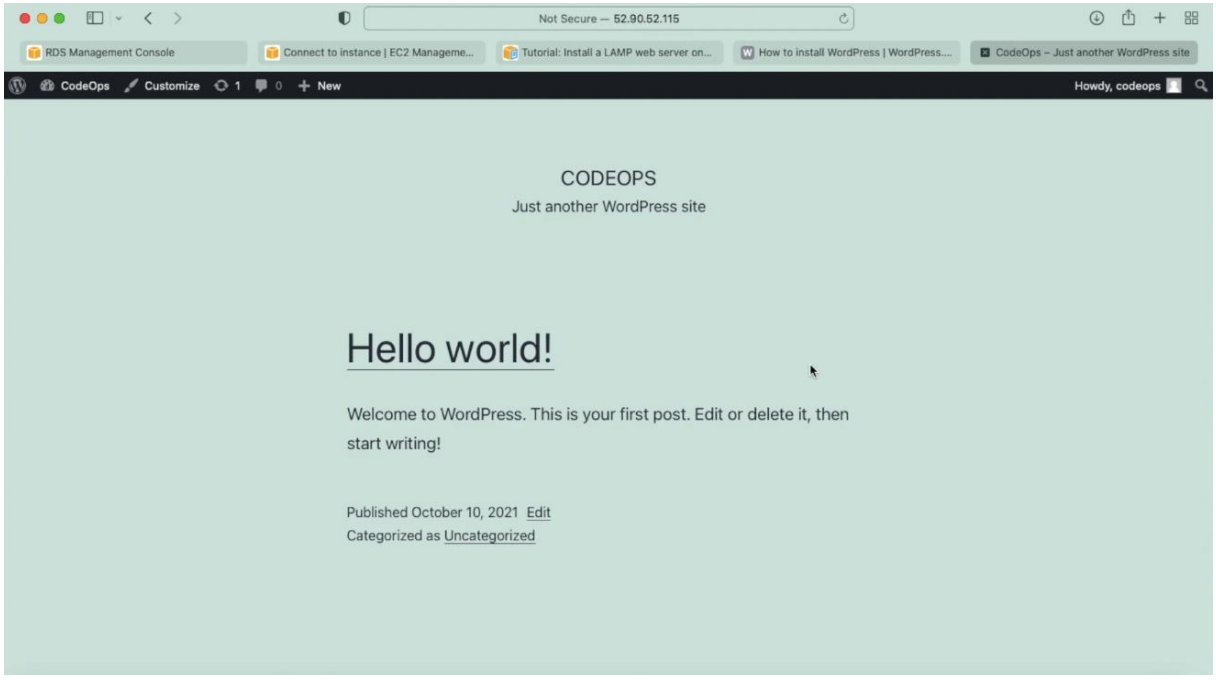


Fig 5.8 The WordPress website by default created by WordPress

CHAPTER 4

SUMMARY AND CONCLUSION

SUMMARY

A relational database is a collection of data items with pre-defined relationships between them. These items are organized as a set of tables with columns and rows. Tables are used to hold information about the objects to be represented in the database. Each column in a table holds a certain kind of data and a field stores the actual value of an attribute. The rows in the table represent a collection of related values of one object or entity. Each row in a table could be marked with a unique identifier called a primary key, and rows among multiple tables can be made related using foreign keys. This data can be accessed in many different ways without reorganizing the database tables themselves.

CONCLUSION

Amazon Relational Database is widely accessible ,scalable and also it offers broad range of security .It is a feature rich and make sure AWS service and it makes easy for startup companies to maintain their own relational database entire process run-through on AWS RDS, we can hence deduce that Amazon RDS is a powerful tool that provides cost-efficient, re-sizable capacity in a standard relational database and manages common database administration tasks with high availability, security, and compatibility of databases.

REFERENCES

External References

1. https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/custom-creating-sqlserver.html
2. https://docs.aws.amazon.com/pdfs/AmazonRDS/latest/UserGuide/rds-ug.pdf#MySQL.Concepts.FeatureSupport
3. https://aws.amazon.com/rds/#
4. https://docs.aws.amazon.com/codedeploy/latest/userguide/tutorials-wordpress-launch-instance.html

star certification material

1. https://drive.google.com/drive/folders/1qkyllkmnTNUgVz_5toPi_9wxE96qHraw
2. https://elearning.starcertification.org
3. https://www.starcertification.org/Certifications/Certificate/cloud
4.https://cloudacademy.com/blog/prerequisites-to-learn-cloud-computing-s3bucket/s