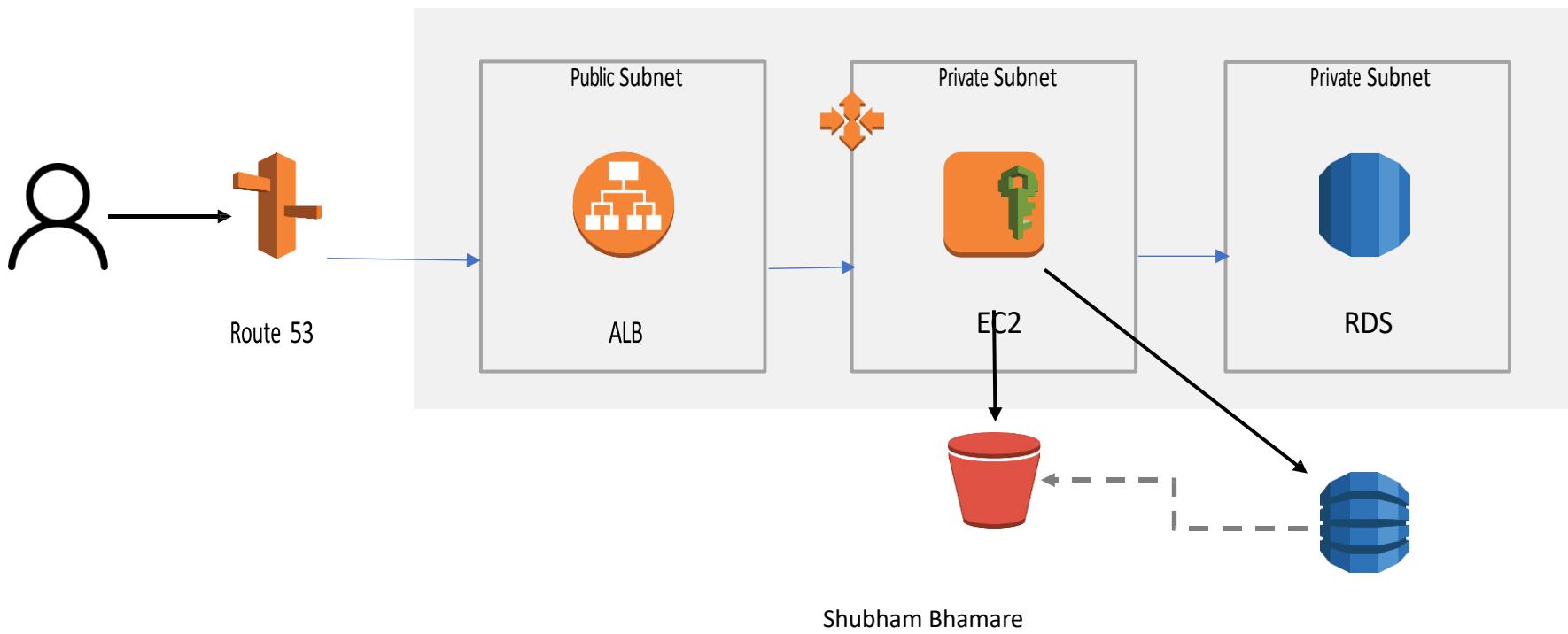


Problem Statement

Employee profile of XYZ company – New employees input their information and upload photos. Photos to be stored into S3 bucket and other information to be stored into RDS. Metadata to be stored into DynamoDB.

Technical Architecture



Shubham Bhamare

AWS Setup

- Setup VPC for Load Balancer, Application EC2 instance and RDS Database - one public and one private subnets.
- Create Load balancer.
- Create RDS DB instance and DynamoDB table.
- S3 Bucket
- Get domain name and map it with Load Balancer
- Create instance profile that has to be attached to the EC2 instances being launched. Instance profile should have permission to access RDS, DynamoDB and S3 bucket.

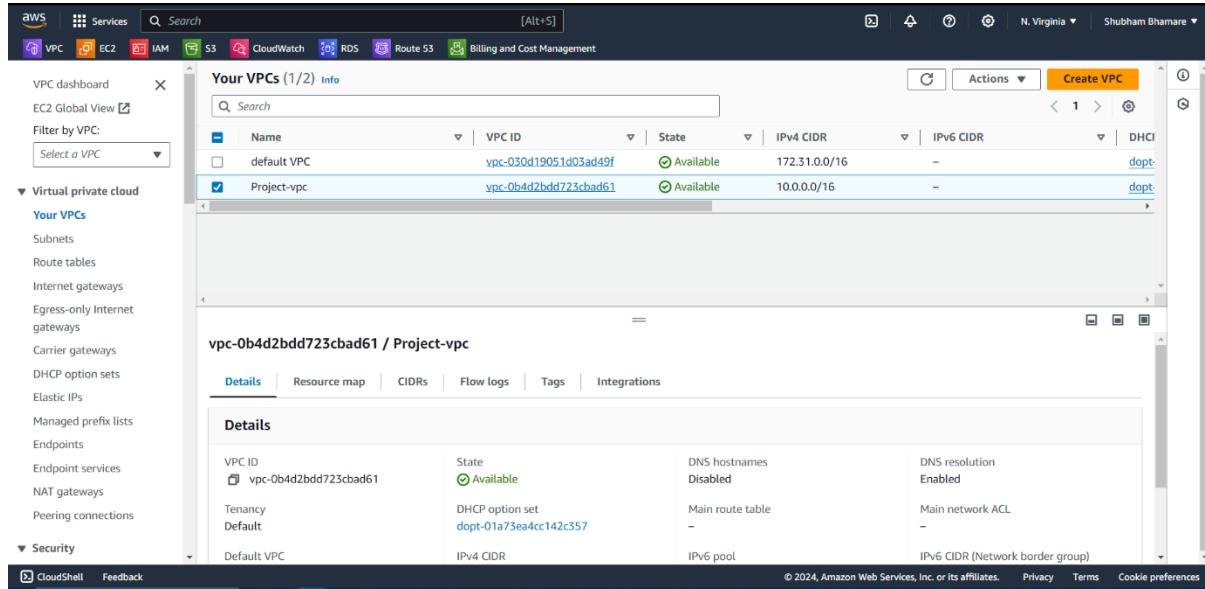
Step 1: Go to VPC Dashboard and click on “Create VPC” to create a new private VPC.

The screenshot shows the AWS VPC Dashboard. On the left, there's a sidebar with options like 'Virtual private cloud' (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only Internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways, Peering connections) and 'Security'. The main area displays 'Your VPCs (1) Info' with a table showing one entry: 'default VPC' (VPC ID: vpc-030d19051d05ad49f, State: Available, IPv4 CIDR: 172.31.0.0/16, IPv6 CIDR: -, DHCP: dopt). At the top right, there's a 'Create VPC' button.

Step 2: Specify the name, CIDR for your VPC.

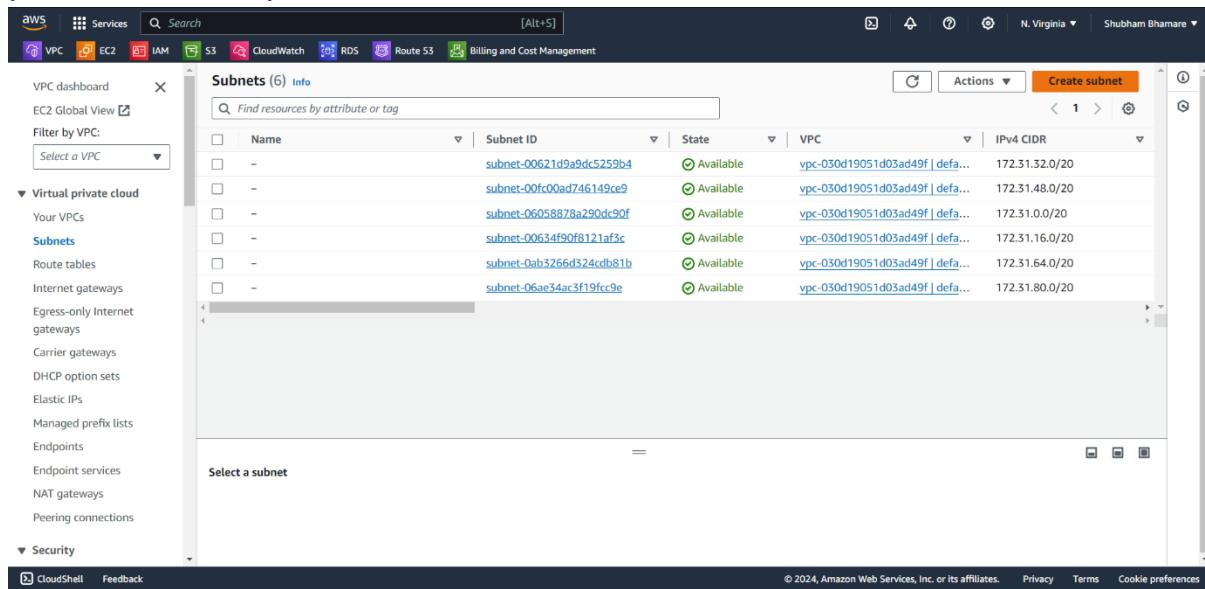
The screenshot shows the 'Create VPC' wizard. Step 1: VPC settings. It asks for 'Resources to create': 'VPC only' (selected) or 'VPC and more'. It has a 'Name tag - optional' field with 'Project-vpc'. Under 'IPv4 CIDR block', it shows '10.0.0.0/16'. Under 'IPv6 CIDR block', it shows 'No IPv6 CIDR block' selected. The bottom of the screen includes standard AWS navigation links: CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

Step 3: VPC is created successfully.



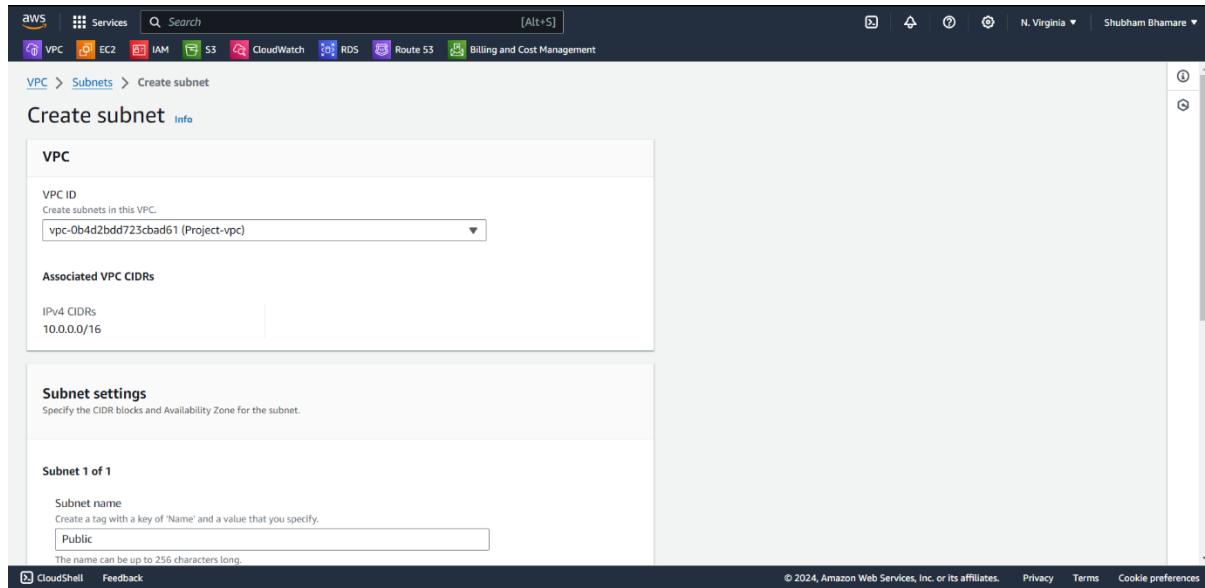
The screenshot shows the AWS VPC dashboard. On the left, there's a sidebar with navigation links for VPC dashboard, EC2 Global View, Filter by VPC, Virtual private cloud (with 'Your VPCs' selected), Subnets, Route tables, Internet gateways, Egress-only Internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways, and Peering connections. Below that is a Security section. At the bottom of the sidebar are CloudShell and Feedback links. The main area has a search bar and a 'Your VPCs (1/2) Info' table. The table has columns for Name, VPC ID, State, IPv4 CIDR, IPv6 CIDR, and DHCP. It shows two rows: 'default VPC' (VPC ID: [vpc-030d19051d03ad49f](#), State: Available, IPv4 CIDR: 172.31.0.0/16) and 'Project-vpc' (VPC ID: [vpc-0b4d2bdd723cbad61](#), State: Available, IPv4 CIDR: 10.0.0.0/16). A modal window titled 'vpc-0b4d2bdd723cbad61 / Project-vpc' is open, showing tabs for Details, Resource map, CIDs, Flow logs, Tags, and Integrations. The 'Details' tab displays information like VPC ID, State, DNS hostnames, and Main network ACL. The 'IPv4 CIDR' field is highlighted in red. At the bottom right of the modal is a note: '© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences'.

Step 4: Now go to subnet and click on “Create Subnet” to create one public and one private subnet.



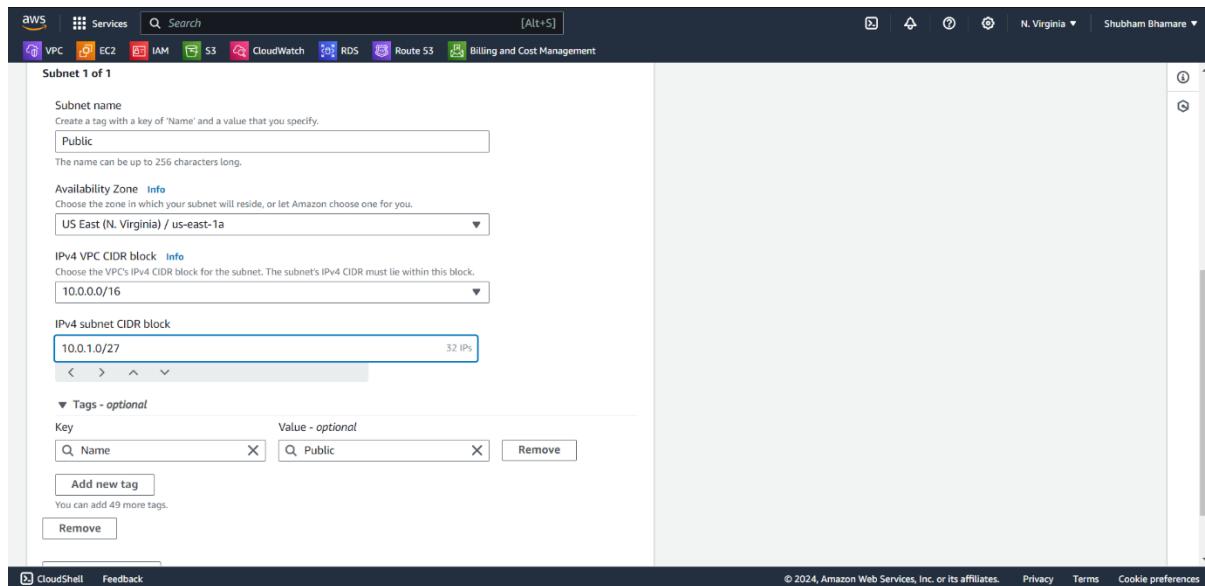
The screenshot shows the AWS Subnets dashboard. The sidebar is identical to the previous VPC dashboard. The main area has a search bar and a 'Subnets (6) Info' table. The table has columns for Name, Subnet ID, State, VPC, and IPv4 CIDR. It lists six subnets: 'subnet-00621d9a9dc5259b4' (State: Available, VPC: [vpc-030d19051d03ad49f](#), IPv4 CIDR: 172.31.32.0/20), 'subnet-00fc00ad746149ce9' (State: Available, VPC: [vpc-030d19051d03ad49f](#), IPv4 CIDR: 172.31.48.0/20), 'subnet-06058878a290dc90f' (State: Available, VPC: [vpc-030d19051d03ad49f](#), IPv4 CIDR: 172.31.0.0/20), 'subnet-00634f90fb121af3c' (State: Available, VPC: [vpc-030d19051d03ad49f](#), IPv4 CIDR: 172.31.16.0/20), 'subnet-0ab3266d524cd81b' (State: Available, VPC: [vpc-030d19051d03ad49f](#), IPv4 CIDR: 172.31.64.0/20), and 'subnet-06ae34ac3f19fcc9e' (State: Available, VPC: [vpc-030d19051d03ad49f](#), IPv4 CIDR: 172.31.80.0/20). A modal window titled 'Select a subnet' is open at the bottom. At the top right of the main area is a 'Create subnet' button. The bottom right of the main area contains the copyright notice: '© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences'.

Step 5: Select your VPC in which you want to create subnet.



The screenshot shows the 'Create subnet' page in the AWS VPC service. At the top, there's a navigation bar with 'Search' and other services like EC2, IAM, S3, CloudWatch, RDS, Route 53, and Billing and Cost Management. Below that, it says 'VPC > Subnets > Create subnet'. The main form has a section titled 'VPC' with a dropdown labeled 'VPC ID' containing 'vpc-0b4d2bdd723cbad61 [Project-vpc]'. Another section 'Associated VPC CIDRs' shows 'IPv4 CIDRs' as '10.0.0.0/16'. A 'Subnet settings' section follows, with a note to 'Specify the CIDR blocks and Availability Zone for the subnet.' It shows 'Subnet 1 of 1' with a 'Subnet name' field containing 'Public'. The bottom of the screen includes standard AWS footer links for CloudShell, Feedback, and legal notices.

Step 6: Mention the name to your subnet , Select AZ, Specify CIDR for your subnet.



This screenshot shows the 'Subnet 1 of 1' configuration page. It includes fields for 'Subnet name' (set to 'Public'), 'Availability Zone' (set to 'US East (N. Virginia) / us-east-1a'), and 'IPv4 VPC CIDR block' (set to '10.0.0.0/16'). Below these, the 'IPv4 subnet CIDR block' is set to '10.0.1.0/27'. Under 'Tags - optional', there are two entries: 'Name' with value 'Public' and another entry with 'Name' and 'Public' both selected. A note indicates 'You can add 49 more tags.' The bottom of the screen features the usual AWS footer.

Step 7: Again add another subnet and now create one private subnet and mention name, AZ, CIDR to your subnet.

The screenshot shows the AWS VPC Subnet creation interface. The 'Subnet name' field is set to 'private'. The 'Availability Zone' dropdown is set to 'US East (N. Virginia) / us-east-1b'. The 'IPv4 VPC CIDR block' dropdown is set to '10.0.0.0/16'. The 'IPv4 subnet CIDR block' dropdown is set to '10.0.2.0/27'. A 'Tags - optional' section shows a single tag 'Name: private'. The interface includes standard AWS navigation and status bars at the top and bottom.

Step 8: Two subnets are created successfully.

The screenshot shows the AWS VPC Subnets list. It displays two subnets: 'Public' (Subnet ID: subnet-03144e87f66a97a26, State: Available, VPC: vpc-0b4d2bdd723cbad61, IPv4 CIDR: 10.0.1.0/27) and 'private' (Subnet ID: subnet-0f451eab54e2401d4, State: Available, VPC: vpc-0b4d2bdd723cbad61, IPv4 CIDR: 10.0.2.0/27). The interface includes a sidebar for VPC management and standard AWS navigation and status bars.

Name	Subnet ID	State	VPC	IPv4 CIDR
Public	subnet-03144e87f66a97a26	Available	vpc-0b4d2bdd723cbad61 Proj...	10.0.1.0/27
private	subnet-0f451eab54e2401d4	Available	vpc-0b4d2bdd723cbad61 Proj...	10.0.2.0/27

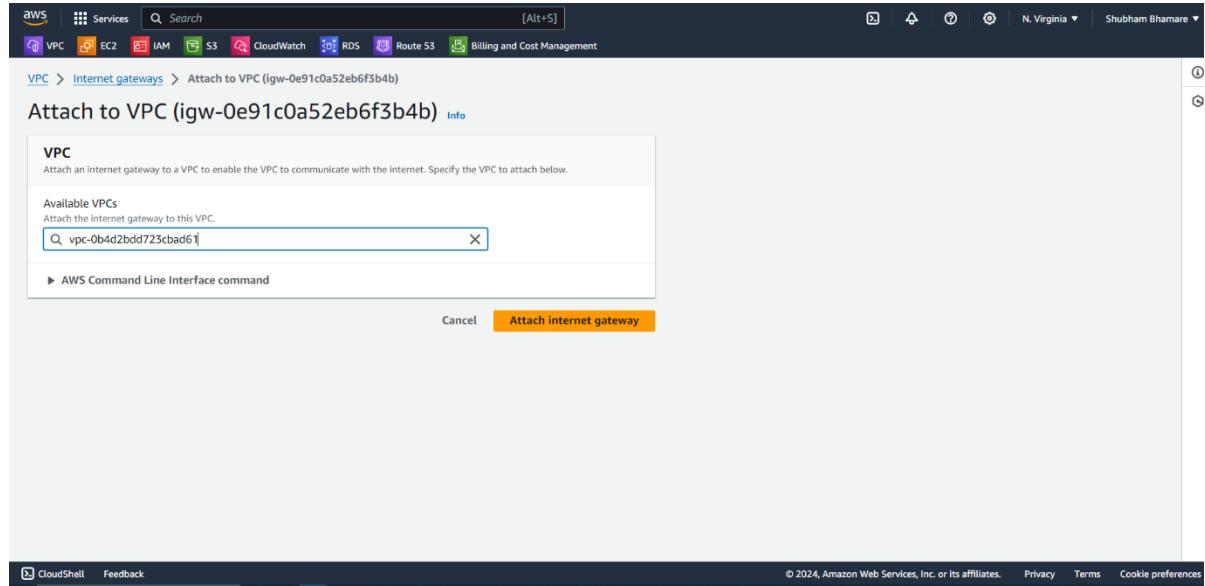
Step 9: Now go to Internet Gateway and create one internet gateway.

The screenshot shows the 'Create internet gateway' page in the AWS VPC service. In the 'Internet gateway settings' section, a 'Name tag' is added with the value 'Project-VPC'. Under 'Tags - optional', a single tag 'Name: Project-VPC' is listed. At the bottom right are 'Cancel' and 'Create internet gateway' buttons.

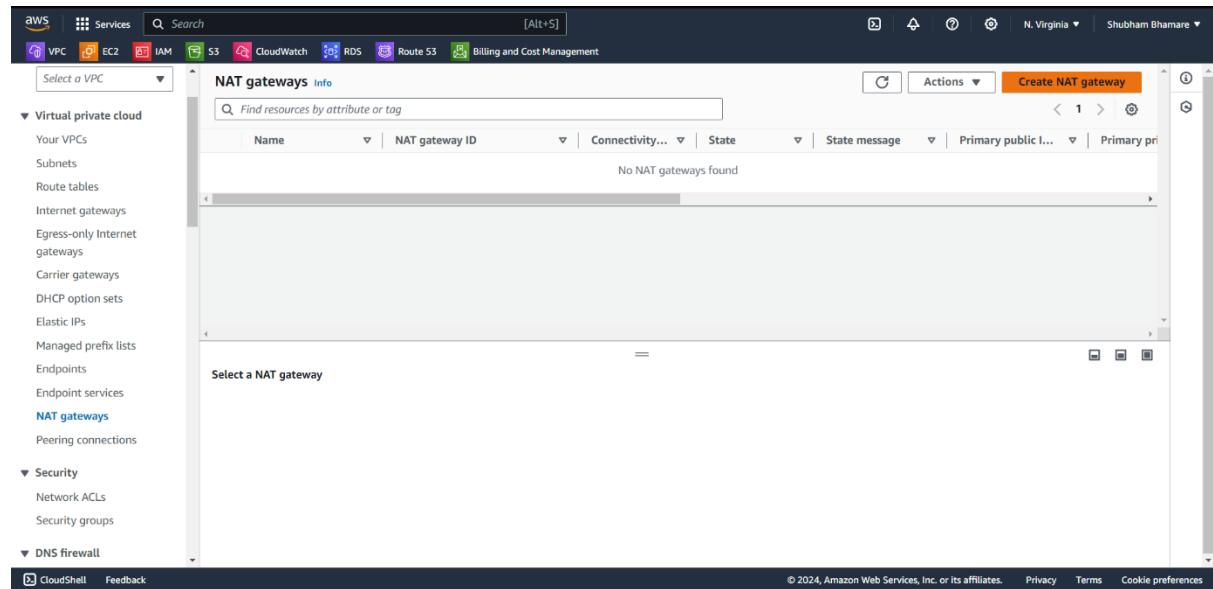
Step 10: Now attach that internet gateway to your VPC.

The screenshot shows the 'Internet Gateways' page in the AWS VPC service. A success message indicates an internet gateway was created: 'igw-0e91c0a52eb6f3b4b - Project-VPC'. Below it, the 'igw-0e91c0a52eb6f3b4b / Project-VPC' details are shown, including its state as 'Detached'. The 'Actions' menu for this gateway includes options like 'Attach to VPC', 'Detach from VPC', 'Manage tags', and 'Delete'. The left sidebar shows the VPC navigation menu.

Step 11: Select your VPC and click on “Attach Internet Gateway”.



Step 12: Now go to NAT gateway and click on “Create NAT gateway”.



Step 13: Specify a name to your NAT gateway , In subnet select the private subnet because NAT gateway is accessing internet through that public subnet, Allocate elastic IP to your NAT gateway.

Create NAT gateway [Info](#)

A highly available, managed Network Address Translation (NAT) service that instances in private subnets can use to connect to services in other VPCs, on-premises networks, or the internet.

NAT gateway settings

Name - *optional*
Create a tag with a key of 'Name' and a value that you specify.

The name can be up to 256 characters long.

Subnet
Select a subnet in which to create the NAT gateway.

Connectivity type
Select a connectivity type for the NAT gateway.
 Public
 Private

Elastic IP allocation ID [Info](#)
Assign an Elastic IP address to the NAT gateway.

[► Additional settings \[Info\]\(#\)](#)

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 14: Now go to Route table and create one public route table.

Create route table [Info](#)

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

Route table settings

Name - *optional*
Create a tag with a key of 'Name' and a value that you specify.

VPC
The VPC to use for this route table.

Tags
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - <i>optional</i>
<input type="text" value="Name"/>	<input type="text" value="Public-RT"/> <input type="button" value="Remove"/>

You can add 49 more tags.

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 15: Now go to edit routes of the public route table and add Internet gateway to your public route table.

The screenshot shows the AWS VPC Route Tables interface. A route table named 'rtb-0e049ac7ab21105df' is being edited. A new route is being added for destination '0.0.0.0/0' with target 'Internet Gateway' (selected from a dropdown) and ID 'igw-0e91c0a52eb6f3b4b'. The route is currently listed as 'local' in the target field. The status is 'Active' and propagation is set to 'No'. There is a 'Remove' button next to the route entry. At the bottom right are 'Cancel', 'Preview', and 'Save changes' buttons. The 'Save changes' button is highlighted in orange.

Step 16: Internet gateway is attached successfully to your public route table.

The screenshot shows the AWS VPC Route Tables interface. A success message at the top states 'Updated routes for rtb-0e049ac7ab21105df / Public-RT successfully'. The main view shows the details of the route table 'rtb-0e049ac7ab21105df / Public-RT'. It lists two routes: one for '0.0.0.0/0' targeting 'igw-0e91c0a52eb6f3b4b' (status Active, propagated No) and another for '10.0.0.16' targeting 'local' (status Active, propagated No). The left sidebar shows navigation links for VPC dashboard, EC2 Global View, Filter by VPC, Virtual private cloud (Your VPCs, Subnets, Route tables), Security, and CloudShell/Feedback.

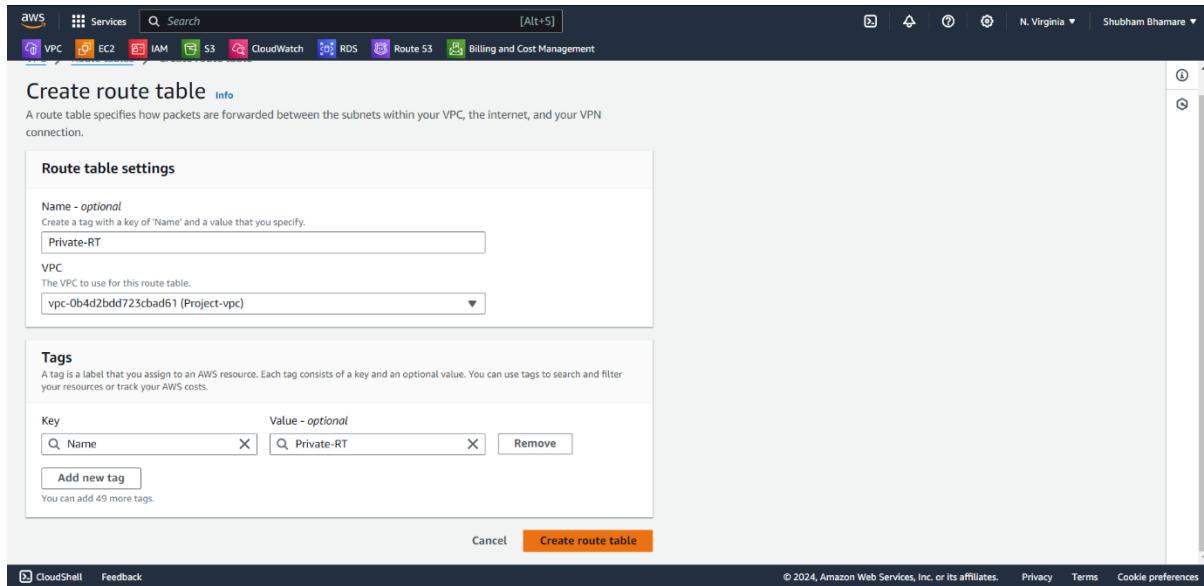
Step 17: Now go to subnet association of public route table and click on “Edit subnet association”.

The screenshot shows the AWS VPC Details page for a public route table. The main header says "Updated routes for rtb-0e049ac7ab21105df / Public-RT successfully". The "Subnet associations" tab is selected. It shows one explicit subnet association: "Public" (subnet-03144e87f66a97a26) with an IPv4 CIDR of 10.0.1.0/27. There are no other subnet associations listed under "Explicit subnet associations" or "Subnets without explicit associations".

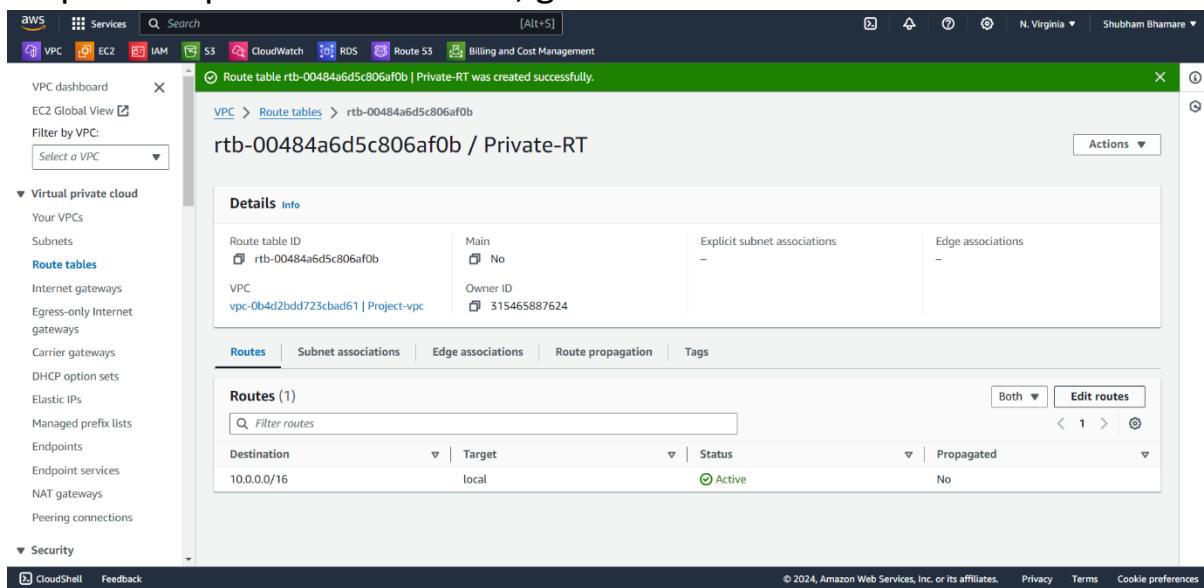
Step 18: Public subnet is associates to your public route table.

The screenshot shows the AWS VPC Details page for the same public route table after a subnet has been associated. The "Subnet associations" tab now lists the "Public" subnet with the specified IPv4 CIDR. The "Explicit subnet associations" section shows the association, and the "Subnets without explicit associations" section is empty.

Step 19: Again create one private route table.



Step 20 : In private route table, go to Routes and click on “Edit routes”.



Step 21: Attach NAT gateway to your private route table.

The screenshot shows the AWS VPC Route Tables interface. A route is being added to a private route table. The destination is 10.0.0.0/16, and the target is a NAT Gateway with ID nat-0bebb093ce52cd9ad. The status is Active, and propagation is disabled. The 'Save changes' button is highlighted.

Step 22: Now go to subnet association of your private route table and click on “Edit subnet association”.

The screenshot shows the AWS VPC Route Tables interface for the route table rtb-00484a6d5c806af0b. The 'Subnet associations' tab is selected. It displays the route table ID, main status, and owner information. Below this, the 'Explicit subnet associations' section is shown, which currently has no associations. The 'Edit subnet associations' button is visible.

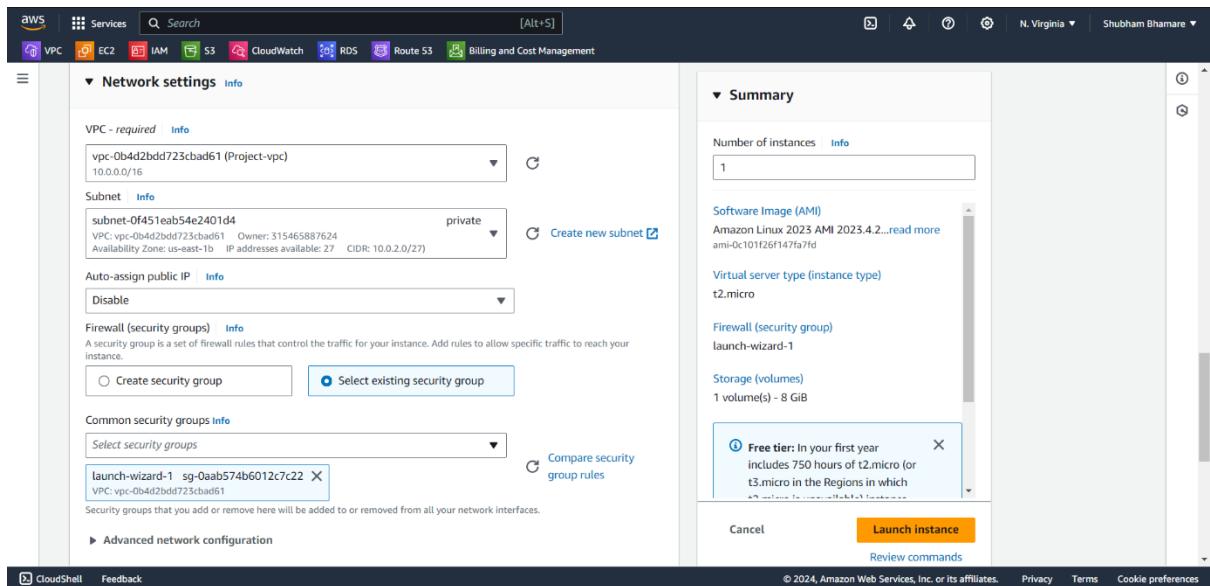
Step 23: Attach private subnet to private route table.

The screenshot shows the 'Edit subnet associations' page for a specific route table. In the 'Available subnets' section, two subnets are listed: 'Public' (subnet-03144e87f66a97a26) and 'private' (subnet-0f451eab54e2401d4). The 'private' subnet is selected and associated with the route table 'rtb-0e049ac7ab21105df / Public-RT'. In the 'Selected subnets' section, the 'private' subnet is listed. At the bottom right, there are 'Cancel' and 'Save associations' buttons.

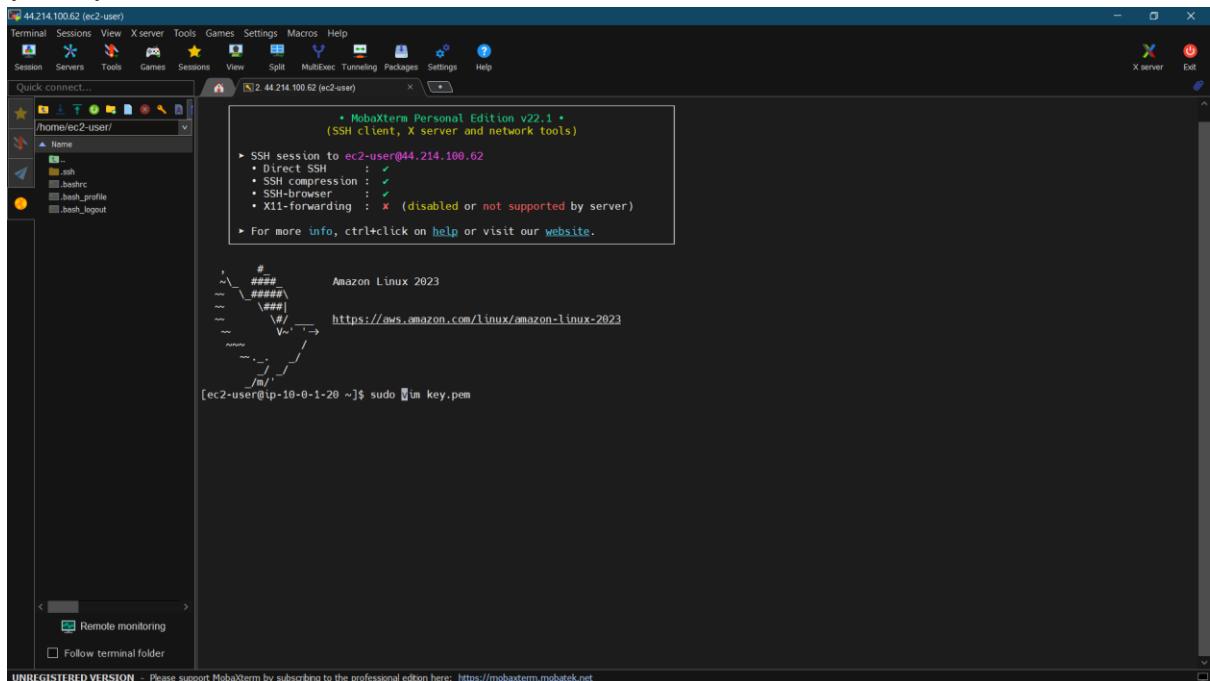
Step 24: Now Go to EC2 service and launch one public instance with public subnet and enable public IP.

The screenshot shows the 'Network settings' step of the EC2 'Launch instance' wizard. It specifies a VPC (Project-vpc) and a public subnet (subnet-03144e87f66a97a26). Under 'Auto-assign public IP', 'Enable' is selected. A note about additional charges applies when outside of free tier allowance. Under 'Firewall (security groups)', it says 'Create security group' is selected. A security group named 'launch-wizard-1' is being created. The 'Description' field contains 'launch-wizard-1 created 2024-03-24T13:12:26.604Z'. On the right, the 'Summary' panel shows 1 instance, the AMI (Amazon Linux 2023 AMI 2025.4.2...), the instance type (t2.micro), and a note about the free tier. At the bottom right are 'Cancel', 'Launch instance', and 'Review commands' buttons.

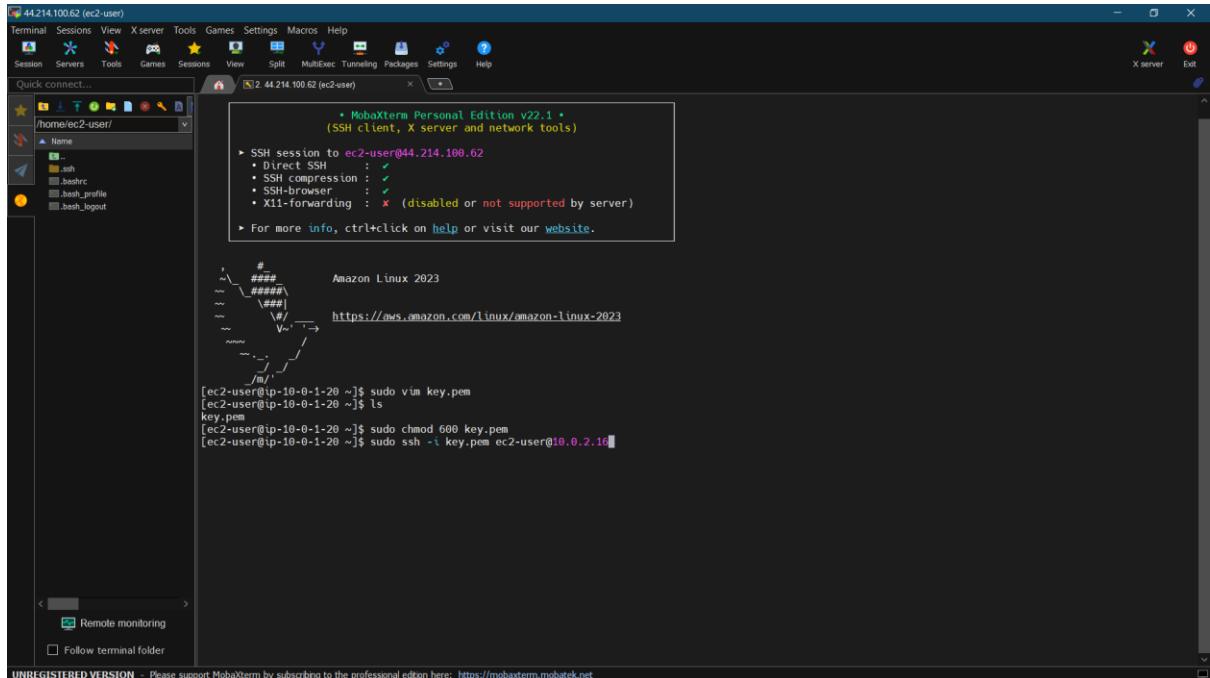
Step 25: Again launch one private instance in your private subnet.



Step 26: Take SSH to your public Instance and create one key file and paste the content of instance key into that file for taking SSH access to your private instance.



Step 27: Key is created successfully. Now change the permission for that key using “chmod 600 <filename>” command. Then run command “sudo ssh -l <keyfile_name> ec2-user@<private_ip_private_Instance>”.



The screenshot shows the MobaXterm interface with a terminal window titled '44.214.100.62 (ec2-user)'. The terminal output shows:

```
* MobaXterm Personal Edition v22.1 *
(SSH client, X server and network tools)

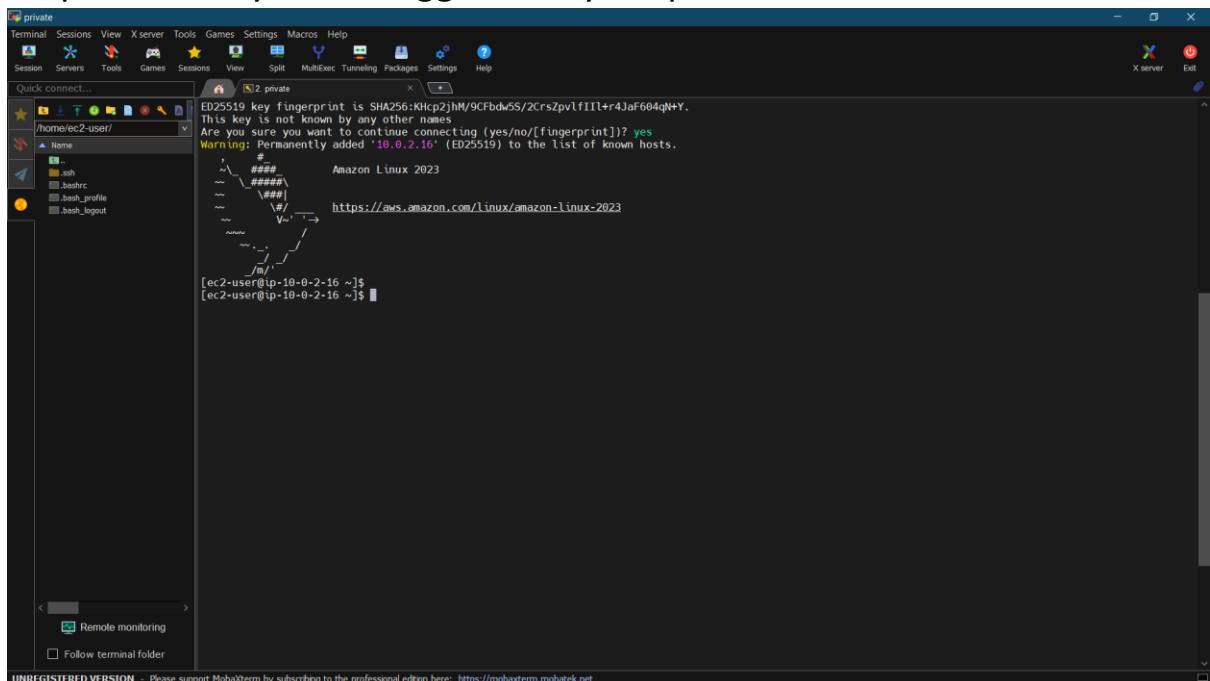
> SSH session to ec2-user@44.214.100.62
  • Direct SSH : ✓
  • SSH compression : ✓
  • SSH-browser : ✓
  • X11-forwarding : ✘ (disabled or not supported by server)

> For more info, ctrl+click on help or visit our website.

, #### Amazon Linux 2023
~~ \####*
~~ \|#|
~~ \|#| https://aws.amazon.com/linux/amazon-linux-2023
~~ \|#| . . .
~~ \|#| / . .
[ec2-user@ip-10-0-1-20 ~]$ sudo vim key.pem
[ec2-user@ip-10-0-1-20 ~]$ ls
key.pem
[ec2-user@ip-10-0-1-20 ~]$ sudo chmod 600 key.pem
[ec2-user@ip-10-0-1-20 ~]$ sudo ssh -l key.pem ec2-user@10.0.2.16
```

At the bottom of the terminal, there is a watermark: UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

Step 28 : Now you are logged into your private instance.



The screenshot shows the MobaXterm interface with a terminal window titled 'private'. The terminal output shows:

```
ED25519 key fingerprint is SHA256:KHcp2jMw/9CFbdwSS/2CrsZpvlfIII+r4JaF604qNHY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.16' (ED25519) to the list of known hosts.

, #### Amazon Linux 2023
~~ \####*
~~ \|#|
~~ \|#| https://aws.amazon.com/linux/amazon-linux-2023
~~ \|#| . . .
~~ \|#| / . .
[ec2-user@ip-10-0-2-16 ~]$ [ec2-user@ip-10-0-2-16 ~]$
```

At the bottom of the terminal, there is a watermark: UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

Step 29: Go to S3 bucket and create one public bucket.

The screenshot shows the 'Create bucket' page in the AWS Management Console. The 'General configuration' section is selected. Under 'Bucket type', the 'General purpose' option is selected, which is described as recommended for most use cases and access patterns. The bucket name is set to 'project-s3-b29'. The 'Bucket settings from existing bucket - optional' section contains a 'Choose bucket' button and a note about copying settings from another bucket. At the bottom, there's a note about the bucket prefix format: 'Format: s3://bucket/prefix'.

Step 30 : Enable ACLs.

The screenshot shows the 'Object Ownership' page in the AWS Management Console. The 'ACLs enabled' option is selected, which is described as recommended for controlling access to objects. A note below says: 'We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.' In the 'Object Ownership' section, the 'Bucket owner preferred' option is selected, which is described as preferred for new objects. A note below says: 'If you want to enforce object ownership for new objects only, your bucket policy must specify that the bucket-owner-full-control canned ACL is required for object uploads. [Learn more](#)'.

Step 31: Untick “block all public access”, and create bucket.

The screenshot shows the 'Block Public Access settings for this bucket' configuration page. At the top, there is a note about public access being granted through various mechanisms like ACLs, bucket policies, and access point policies. Below this, the 'Block all public access' checkbox is selected. A detailed description follows, explaining how turning it on affects different access paths. A warning message in a yellow box states: 'Turning off block all public access might result in this bucket and the objects within becoming public. AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.' At the bottom, a checkbox is checked, acknowledging the warning. The page includes standard AWS navigation elements like CloudShell, Feedback, and a footer with copyright information.

Step 32: Go to RDS and click on “Create database”.

The screenshot shows the 'Amazon RDS > Databases' page. On the left, a sidebar lists various RDS management options: Dashboard, Databases (which is selected), Query Editor, Performance insights, Snapshots, Exports in Amazon S3, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Custom engine versions, Zero-ETL integrations, Events, and Event subscriptions. The main area displays a table titled 'Databases (0)' with columns for DB identifier, Status, Role, Engine, Region & AZ, Size, Recommendations, CPU, and Current activity. A search bar at the top allows filtering by database name. An orange 'Create database' button is prominently displayed at the top right of the table area. The page also includes standard AWS navigation elements like CloudShell, Feedback, and a footer with copyright information.

Step 33: Select the engine for your database.

The screenshot shows the 'Choose a database creation method' step in the AWS RDS wizard. It offers two options: 'Standard create' (selected) and 'Easy create'. The 'Standard create' section includes a note: 'You set all of the configuration options, including ones for availability, security, backups, and maintenance.' On the right, a detailed description of MySQL is provided, highlighting its popularity and various features. Below this, 'Engine options' are listed under 'Engine type': MySQL (selected), Aurora (MySQL Compatible), Aurora (PostgreSQL Compatible), MariaDB, and Oracle. The MySQL option is highlighted with a blue border and a circular icon featuring a globe and a gear.

Step 34: Select a template as per your requirement.

The screenshot shows the 'Templates' step in the AWS RDS wizard. It lists three template options: 'Production', 'Dev/Test', and 'Free Tier' (selected). The 'Free Tier' option is described as using RDS Free Tier to develop new applications, test existing ones, or gain hands-on experience with Amazon RDS. Below this, the 'Availability and durability' section is shown, with 'Deployment options' including 'Multi-AZ DB Cluster', 'Multi-AZ DB instance (not supported for Multi-AZ DB cluster snapshot)', and 'Single DB instance (not supported for Multi-AZ DB cluster snapshot)'. The 'MySQL' details panel on the right is identical to the one in Step 33, detailing MySQL's features and popularity.

Step 35: Specify name for your DB instance.

The screenshot shows the AWS RDS MySQL settings page. In the 'DB instance identifier' field, the value 'project-DB' is entered. To the right of the page, there is a detailed description of MySQL and a bulleted list of its features.

MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TiB.
- Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.

Step 36: Specify username and password for your DB.

The screenshot shows the AWS RDS MySQL settings page. The 'Master username' is set to 'admin'. The 'Self managed' password option is selected. Both the 'Master password' and 'Confirm master password' fields contain the value '*****'. The 'Instance configuration' section at the bottom is visible.

MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TiB.
- Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.

Step 37: You can specify storage as per your requirement.

The screenshot shows the AWS RDS MySQL storage configuration page. On the left, there's a sidebar with 'Storage' selected. Under 'Storage type', 'General Purpose SSD (gp2)' is chosen. The 'Allocated storage' field is set to 20 GiB. A note says: 'After you modify the storage for a DB instance, the status of the DB instance will be in storage-optimization. Your instance will remain available as the storage-optimization operation completes.' On the right, a 'MySQL' details panel lists features like support for up to 64 TiB and automated backups.

Step 38: In connectivity select your VPC.

The screenshot shows the AWS RDS MySQL connectivity configuration page. Under 'Compute resource', 'Don't connect to an EC2 compute resource' is selected. In the 'Virtual private cloud (VPC)' section, 'Project-vpc (vpc-0b4d2bd723cbad61)' is chosen. A note says: 'After a database is created, you can't change its VPC.' On the right, a 'MySQL' details panel lists features like support for up to 64 TiB and automated backups.

Step 39: Restrict public access for your DB.

The screenshot shows the AWS RDS console for creating a new database. In the 'Public access' section, the 'No' option is selected. Under 'VPC security group (firewall)', the 'Choose existing' button is highlighted. The 'MySQL' details pane on the right describes MySQL as the most popular open-source database and lists its features, including support for up to 64 TiB of database size and automated backup.

Step 40: Your database is created successfully.

The screenshot shows the AWS RDS console displaying the 'Databases' table. A single database named 'project-db' is listed, showing it is 'Available' and associated with an 'Instance', 'Engine' (MySQL Community), and 'Region & AZ' (us-east-1b). The table includes columns for DB identifier, Status, Role, Engine, Region & AZ, Size, Recommendations, and CPU.

DB identifier	Status	Role	Engine	Region & AZ	Size	Recommendations	CPU
project-db	Available	Instance	MySQL Community	us-east-1b	db.t3.micro		

Step 41: Now go to DynamoDB and click on “create table”.

The screenshot shows the AWS DynamoDB console homepage. At the top, there's a navigation bar with links for VPC, EC2, IAM, S3, CloudWatch, RDS, Route 53, Billing and Cost Management, and DynamoDB. The main content area features the heading "Amazon DynamoDB: A fast and flexible NoSQL database service for any scale". Below this, a sub-section titled "How it works" contains a video player with the title "What is Amazon DynamoDB?". To the right, there are two boxes: "Get started" (with a "Create table" button) and "Pricing" (with a link to learn more about pricing). On the left, a sidebar menu includes "Dashboard", "Tables", "Explore items", "PartiQL editor", "Backups", "Exports to S3", "Imports from S3", "Integrations", "Reserved capacity", and "Settings". Under "DAX", there are options for "Clusters", "Subnet groups", "Parameter groups", and "Events". The URL in the browser is <https://us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#create-table>.

Step 42: Specify Name for your table, select partition key and datatype for that partition key.

The screenshot shows the "Create table" wizard in the AWS DynamoDB console. The top navigation bar is identical to the previous screenshot. The main form is titled "Create table" and has a "Table details" section. It asks for a "Table name" (set to "metadata_table") and a "Partition key" (set to "empid" of type "Number"). There is also a "Sort key - optional" field (set to "Enter the sort key name" of type "String"). The URL in the browser is https://us-east-1.console.aws.amazon.com/dynamodbv2/tables/CreateTable?region=us-east-1&table_name=metadata_table&partition_key.empid.type=Number&sort_key.name.type=String.

Step 43: Table is created successfully.

The screenshot shows the AWS DynamoDB console. In the top navigation bar, 'DynamoDB' is selected. Below it, the 'Tables' section is active. A table named 'metadata_table' is listed, showing it is Active with a partition key 'empid (N)', no sort key, 0 indexes, Off deletion protection, Provisioned (5) read capacity mode, and Provisioned (5) write capacity mode. The left sidebar includes sections for Dashboard, Tables, Explore items, PartQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. The bottom of the screen shows CloudShell, Feedback, and copyright information for 2024, Amazon Web Services, Inc. or its affiliates.

Step 44: Now go to your private instance and install mariadb using

“sudo yum install mariadb105-server -y”.

The screenshot shows a terminal window titled 'private'. The user runs the command 'sudo yum install mariadb105-server -y'. The terminal output shows the installation of the MariaDB 10.5 server and its dependencies. The transaction summary indicates 22 packages were installed, totaling 20 M download size and 118 M installed size. The repository used is amazonlinux. The terminal interface includes tabs for Sessions, View, Xserver, Tools, Games, Settings, Macros, Help, and various session management buttons.

```
[ec2-user@ip-10-0-2-16 ~]$ sudo yum install mariadb105-server -y
[ec2-user@ip-10-0-2-16 ~]$ sudo yum install mariadb105-server -y
Last metadata expiration check: 0:02:52 ago on Sun Mar 24 14:06:22 2024.
Dependencies resolved.

=====
---> Package           Architecture Version      Repository
--->
Installing:
  mariadb105-server          x86_64   3:10.5.23-1.amzn2023.0.1    amazonlinux
Installing dependencies:
  mariadb-connector-c         x86_64   3.1.13-1.amzn2023.0.3    amazonlinux
  mariadb-connector-c-config  noarch   3:10.5.23-1.amzn2023.0.1    amazonlinux
  mariadb105                  x86_64   3:10.5.23-1.amzn2023.0.1    amazonlinux
  mariadb105-common           x86_64   3:10.5.23-1.amzn2023.0.1    amazonlinux
  mariadb105-ermsg            x86_64   3:10.5.23-1.amzn2023.0.1    amazonlinux
  mysql-selinux               noarch   1.0.4-2.amzn2023.0.3      amazonlinux
  perl-B                      x86_64   1.80-47.amzn2023.0.6     amazonlinux
  perl-DB-MariaDB             x86_64   1.22-1.amzn2023.0.4       amazonlinux
  perl-DBI                     x86_64   1.643-7.amzn2023.0.3     amazonlinux
  perl-Data-Dumper              x86_64   2.174-460.amzn2023.0.2    amazonlinux
  perl-File-Copy                noarch   2.34-477.amzn2023.0.6     amazonlinux
  perl-FileHandle               noarch   2.03-477.amzn2023.0.6     amazonlinux
  perl-Math-BigInt              noarch   1:1.9998.39-2.amzn2023.0.2 amazonlinux
  perl-Math-BigRat              noarch   0.2614-458.amzn2023.0.2    amazonlinux
  perl-Math-Complex              x86_64   1.59-477.amzn2023.0.6     amazonlinux
  perl-Sys-Hostname              x86_64   1.23-477.amzn2023.0.6     amazonlinux
  perl-base                     noarch   2.27-477.amzn2023.0.6     amazonlinux
Installing weak dependencies:
  mariadb105-backup            x86_64   3:10.5.23-1.amzn2023.0.1    amazonlinux
  mariadb105-cracklib-password-check x86_64   3:10.5.23-1.amzn2023.0.1    amazonlinux
  mariadb105-gssapi-server      x86_64   3:10.5.23-1.amzn2023.0.1    amazonlinux
  mariadb105-server-utils       x86_64   3:10.5.23-1.amzn2023.0.1    amazonlinux

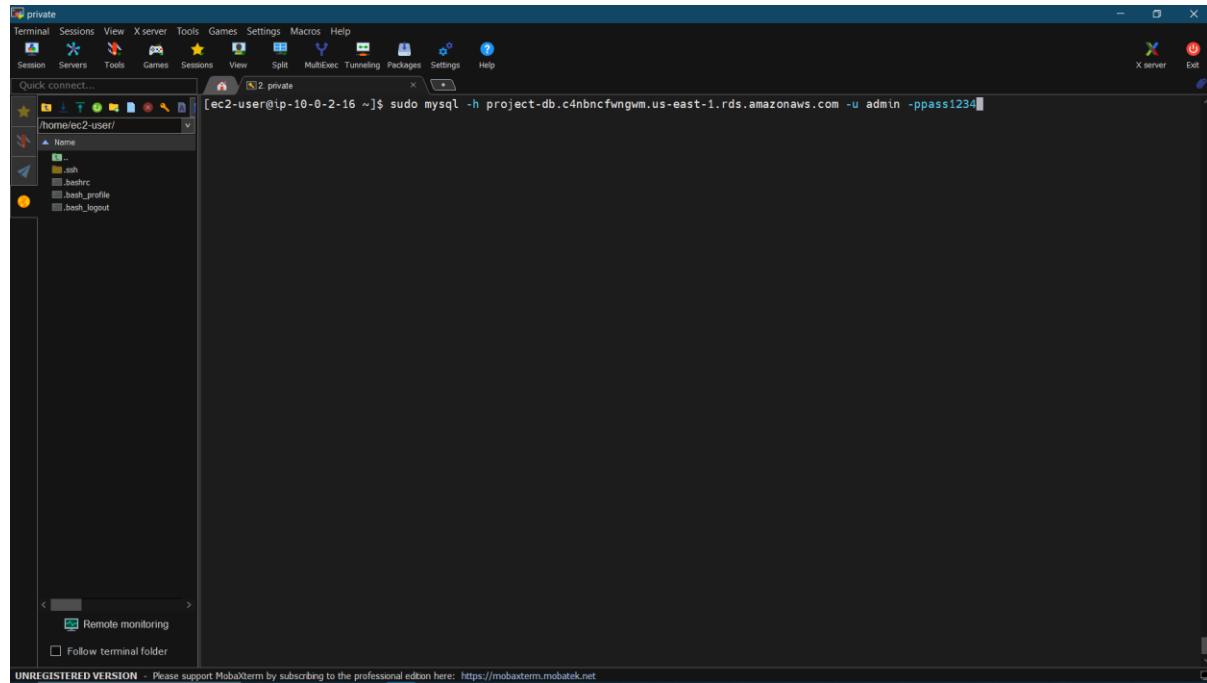
Transaction Summary
=====
Install  22 Packages

Total download size: 20 M
Installed size: 118 M
Downloading Packages:
(1/22): perl-Sys-Hostname-1.23-477.amzn2023.0.6.x86_64.rpm  274 kB/s | 18 kB
(2/22): perl-DBD-MariaDB-1.22-1.amzn2023.0.4.x86_64.rpm  6.1 MB/s | 153 kB
(3/22): perl-DBI-1.643-7.amzn2023.0.3.x86_64.rpm  5.3 MB/s | 700 kB

```

Step 45: Now login into your database using command

“`sudo mysql -h <DBendpoint> -u <username> -p<password>`”.

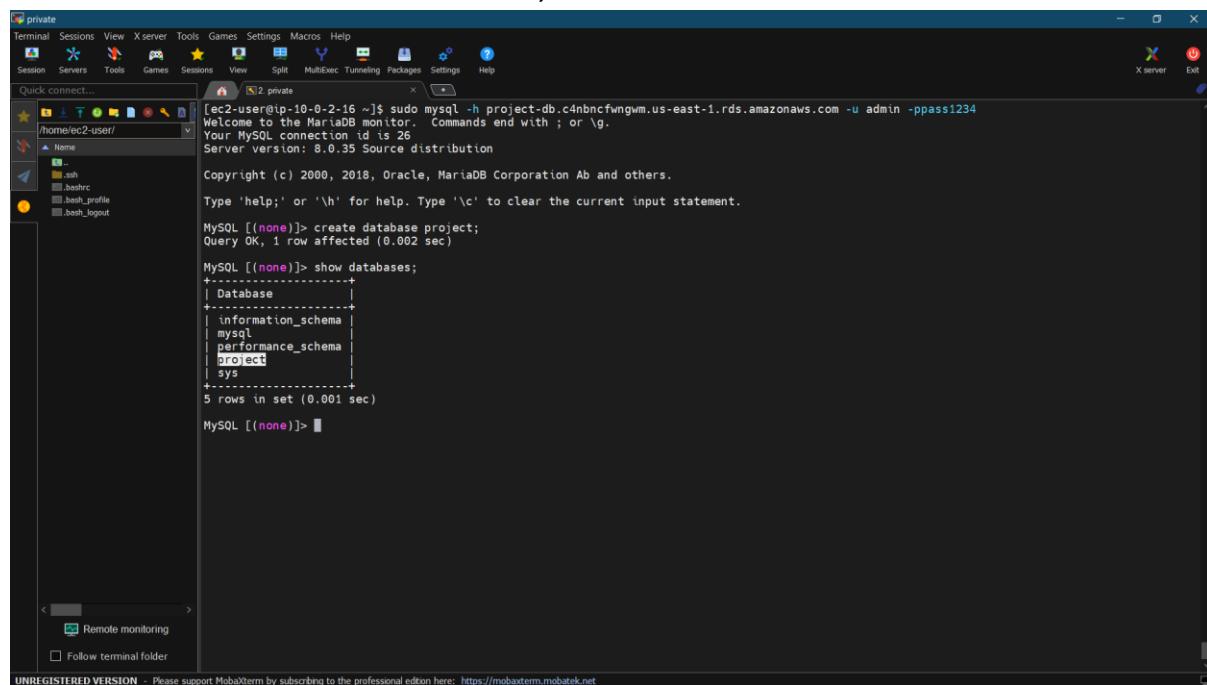


A screenshot of the MobaXterm application window. The title bar says "private". The menu bar includes Terminal, Sessions, View, Xserver, Tools, Games, Settings, Macros, Help. The toolbar has icons for Session, Servers, Tools, Games, Sessions, View, Split, MultiExec, Tunneling, Packages, Settings, Help. The main terminal window shows the command:

```
[ec2-user@ip-10-0-2-16 ~]$ sudo mysql -h project-db.c4nbncfwngwm.us-east-1.rds.amazonaws.com -u admin -ppass1234
```

The file browser sidebar shows a directory tree under "/home/ec2-user/". The bottom status bar says "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

Step 46: Run command “create database <DBname>;” to create DB and run command “show databases;” to list all databases.



A screenshot of the MobaXterm application window. The title bar says "private". The menu bar includes Terminal, Sessions, View, Xserver, Tools, Games, Settings, Macros, Help. The toolbar has icons for Session, Servers, Tools, Games, Sessions, View, Split, MultiExec, Tunneling, Packages, Settings, Help. The main terminal window shows the MySQL monitor output:

```
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 26
Server version: 8.0.35 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

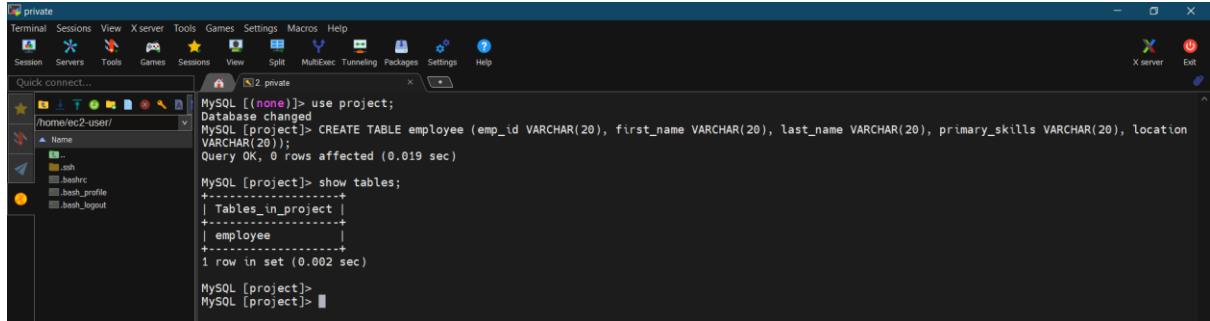
MySQL [(none)]> create database project;
Query OK, 1 row affected (0.002 sec)

MySQL [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| project |
| sys |
+-----+
5 rows in set (0.001 sec)

MySQL [(none)]>
```

The file browser sidebar shows a directory tree under "/home/ec2-user/". The bottom status bar says "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

Step 47: Now run command “use <db-name>;” to enter into that database then run command “CREATE TABLE employee (emp_id VARCHAR(20), first_name VARCHAR(20), last_name VARCHAR (20), primary_skills VARCHAR(20), location VARCHAR(20));” to create table., Now run command “show tables;” to list all tables in your DB.

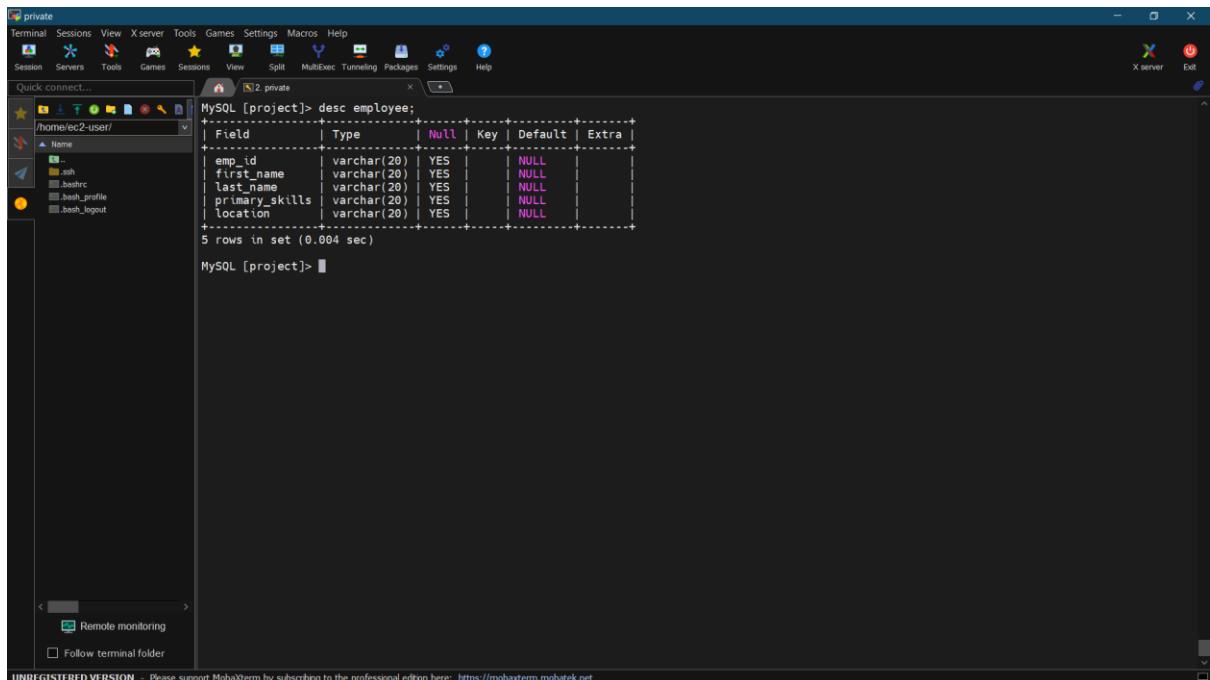


```
private
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
/home/ec2-user/
MySQL [(none)]> use project;
Database changed
MySQL [project]> CREATE TABLE employee (emp_id VARCHAR(20), first_name VARCHAR(20), last_name VARCHAR(20), primary_skills VARCHAR(20), location VARCHAR(20));
Query OK, 0 rows affected (0.019 sec)

MySQL [project]> show tables;
+-----+
| Tables_in_project |
+-----+
| employee |
+-----+
1 row in set (0.002 sec)

MySQL [project]>
MySQL [project]>
```

Step 48: Run command “desc <table_name>;” to describe table.



```
private
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
/home/ec2-user/
MySQL [project]> desc employee;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| emp_id | varchar(20) | YES | NULL | NULL |
| first_name | varchar(20) | YES | NULL | NULL |
| last_name | varchar(20) | YES | NULL | NULL |
| primary_skills | varchar(20) | YES | NULL | NULL |
| location | varchar(20) | YES | NULL | NULL |
+-----+-----+-----+-----+-----+
5 rows in set (0.004 sec)

MySQL [project]>
```

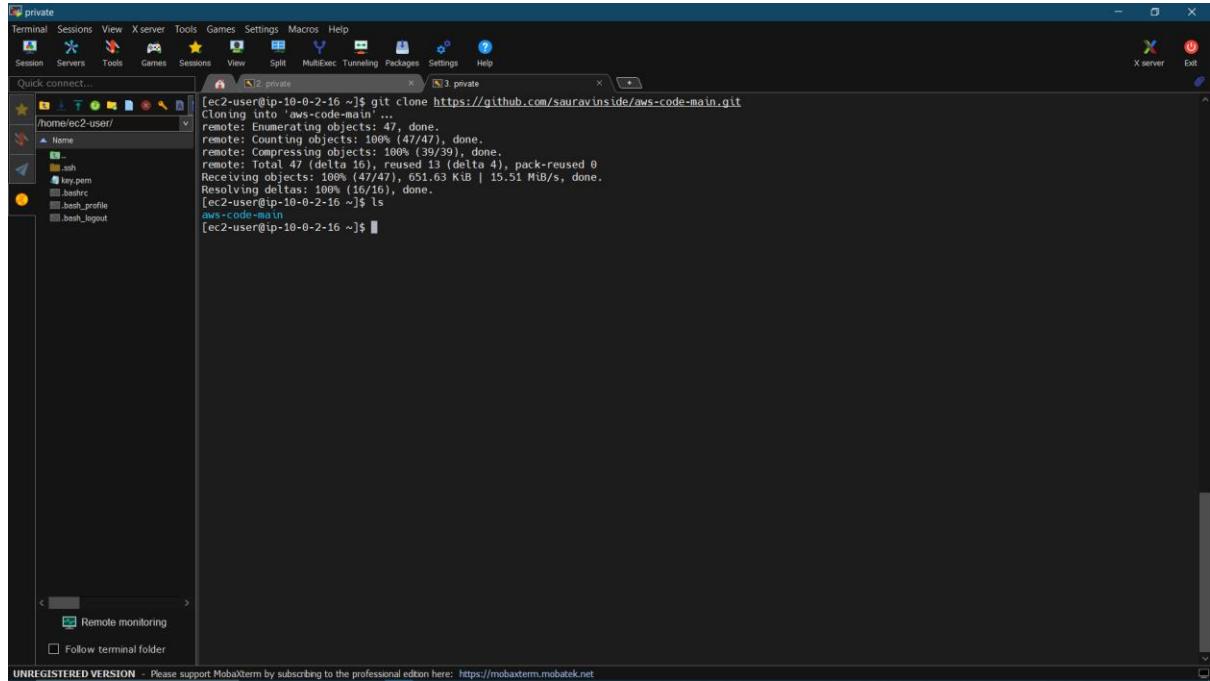
Step 49: Now go to <https://github.com/sauravinside/aws-code-main> then click on “code” and the url.

The screenshot shows the GitHub repository page for 'aws-code-main'. The 'Code' tab is selected, displaying the repository's structure. The repository contains a file named 'EmpApp.py' which has been updated. The 'About' section provides details about the repository, including its purpose as 'AWS python code for application', its URL ('https://github.com/sauravinside/aws-code-main'), and its activity metrics (10 stars, 2 watching, 72 forks). The 'Languages' section indicates that the code is primarily written in Python (55.3%) and HTML (44.7%).

Step 50: On private instance run command “sudo yum install git -y” to install git on your private instance.

```
[ec2-user@ip-10-0-2-16 ~]$ sudo yum install git -y
Last metadata expiration check: 0:19:54 ago on Sun Mar 24 14:06:22 2024.
Dependencies resolved.
=====
Installing:
  git
  git-core
  git-core-doc
  perl-Error
  perl-File-Find
  perl-Git
  perl-TermReadKey
  perl-Lib
=====
Transaction Summary
=====
Install 8 Packages
Total download size: 7.1 M
Installed size: 34 M
Downloading Packages:
(1/8): perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64.rpm      502 kB/s | 36 kB   00:00
(2/8): perl-lib-0.05-477.amzn2023.0.6.x86_64.rpm           640 kB/s | 15 kB   00:00
(3/8): git-2.40.1-1.amzn2023.0.1.noarch.rpm             527 kB/s | 5 kB   00:00
(4/8): git-core-2.40.1-1.amzn2023.0.1.x86_64.rpm          31 MB/s | 4.3 kB   00:00
(5/8): perl-Error-0.17029-5.amzn2023.0.2.noarch.rpm       3.0 MB/s | 41 kB   00:00
(6/8): perl-File-Find-1.37-477.amzn2023.0.6.noarch.rpm    1.5 MB/s | 26 kB   00:00
(7/8): perl-Git-2.40.1-1.amzn2023.0.1.noarch.rpm         2.8 MB/s | 45 kB   00:00
(8/8): git-core-doc-2.40.1-1.amzn2023.0.1.noarch.rpm      23 MB/s | 2.6 MB   00:00
=====
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing:
  Installing : git-core-2.40.1-1.amzn2023.0.1.x86_64
  Installing : git-core-doc-2.40.1-1.amzn2023.0.1.noarch
  Installing : perl-Error-0.17029-5.amzn2023.0.2.noarch
  Installing : perl-File-Find-1.37-477.amzn2023.0.6.noarch
=====
1/1
2/8
3/8
4/8
```

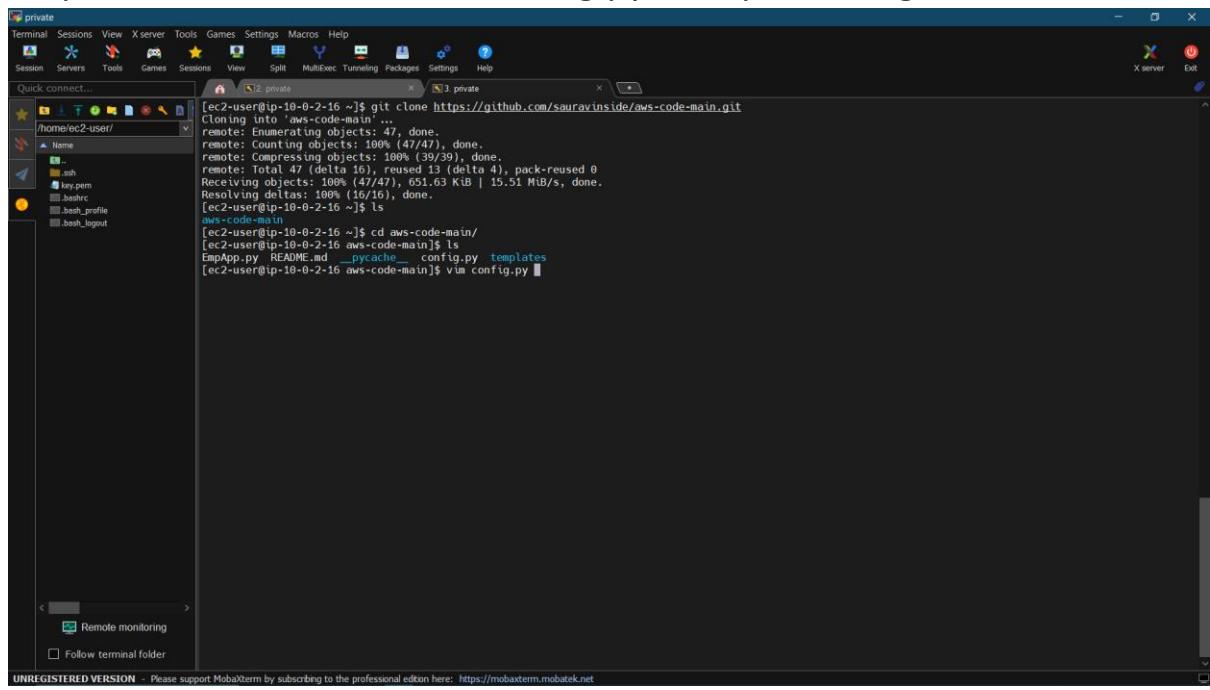
Step 51: Run command “git clone <copied git-URL>” to download the application from git repository.



A screenshot of the MobaXterm terminal window titled "private". The terminal shows the command "git clone https://github.com/sauravinside/aws-code-main.git" being run. The output indicates the cloning process is complete, with 47 objects received and 15.51 MiB/s transfer speed. The terminal window has a dark theme and includes a sidebar with session management and monitoring tools.

```
[ec2-user@ip-10-0-2-16 ~]$ git clone https://github.com/sauravinside/aws-code-main.git
Cloning into 'aws-code-main'...
remote: Enumerating objects: 100% (47/47), done.
remote: Compressing objects: 100% (39/39), done.
remote: Total 47 (delta 16), reused 13 (delta 4), pack-reused 0
Receiving objects: 100% (47/47), 651.63 KiB | 15.51 MiB/s, done.
Resolving deltas: 100% (16/16), done.
[ec2-user@ip-10-0-2-16 ~]$ ls
aws<-code-main
[ec2-user@ip-10-0-2-16 ~]$
```

Step 52: Run command “vim config.py” to open configuration file.



A screenshot of the MobaXterm terminal window titled "private". The terminal shows the command "vim config.py" being run. The user is in the directory "aws-code-main". The terminal window has a dark theme and includes a sidebar with session management and monitoring tools.

```
[ec2-user@ip-10-0-2-16 ~]$ git clone https://github.com/sauravinside/aws-code-main.git
Cloning into 'aws-code-main'...
remote: Enumerating objects: 100% (47/47), done.
remote: Compressing objects: 100% (39/39), done.
remote: Total 47 (delta 16), reused 13 (delta 4), pack-reused 0
Receiving objects: 100% (47/47), 651.63 KiB | 15.51 MiB/s, done.
Resolving deltas: 100% (16/16), done.
[ec2-user@ip-10-0-2-16 ~]$ cd aws-code-main/
[ec2-user@ip-10-0-2-16 aws-code-main]$ ls
EmpApp.py README.md __pycache__ config.py templates
[ec2-user@ip-10-0-2-16 aws-code-main]$ vim config.py
```

Step 53: Now in configuration file change all configuration as per your resources.

In customhost -Enter your RDS endpoint

Customuser- username

Custompass- password of RDS

Customdb- database name in your RDS

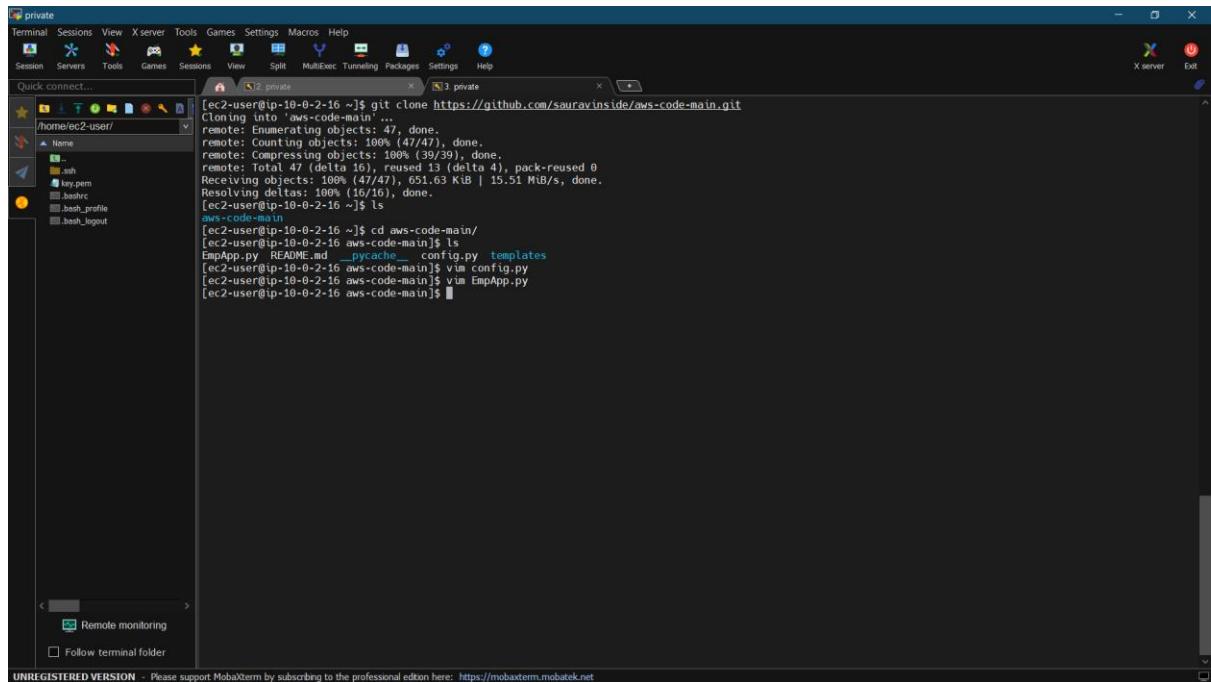
Custombucket – S3 bucket name

Customregion – region in which you bucket is present and save file.



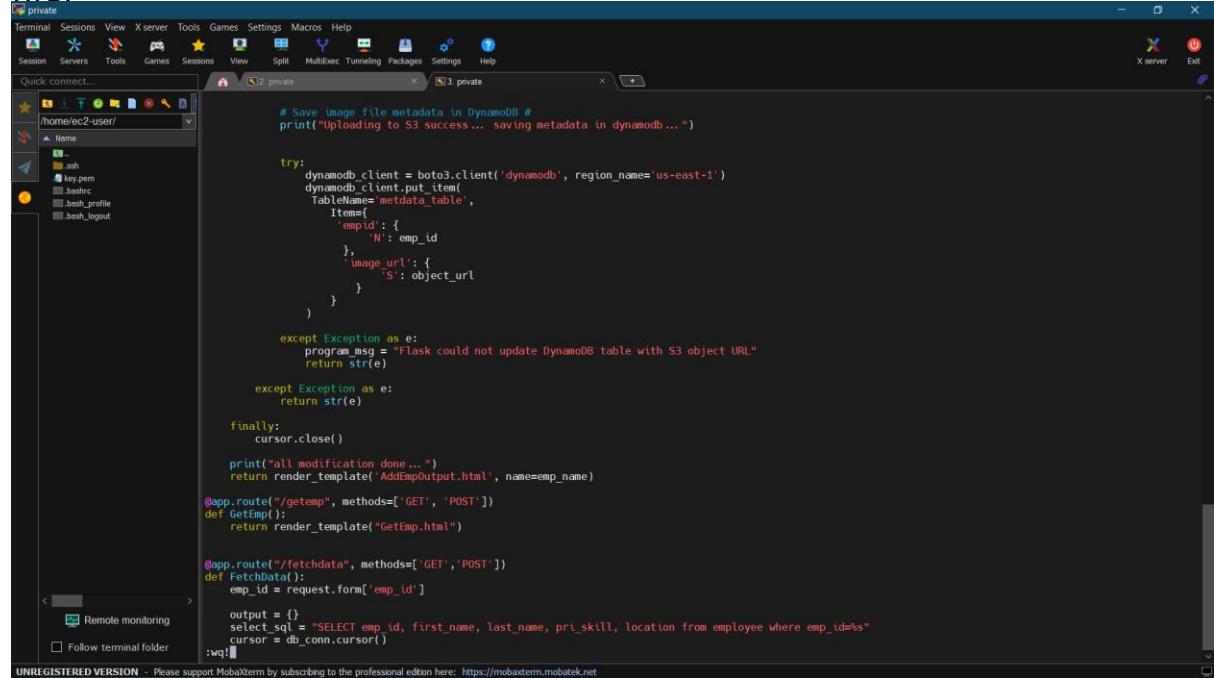
```
private
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
customhost = "project-db.c4bnncfwngw.us-east-1.rds.amazonaws.com"
customuser = "admin"
custompass = "pass1234"
customdb = "project"
custombucket = "project-s3-b29"
customregion = "us-east-1"
~
```

Step 54: Now run command “vim EMpApp.py” to open the DynamoDB configuration file.



```
[ec2-user@ip-10-0-2-16 ~]$ git clone https://github.com/sauravinsde/aws-code-main.git
Cloning into 'aws-code-main'...
remote: Enumerating objects: 47, done.
remote: Counting objects: 100% (47/47), done.
remote: Compressing objects: 100% (39/39), done.
remote: Total 47 (delta 10), reused 13 (delta 4), pack-reused 0
Receiving objects: 100% (47/47) 651.63 KiB | 15.51 MiB/s, done.
Resolving deltas: 100% (16/16), done.
[ec2-user@ip-10-0-2-16 ~]$ ls
aws-code-main
[ec2-user@ip-10-0-2-16 ~]$ cd aws-code-main/
[ec2-user@ip-10-0-2-16 aws-code-main]$ ls
EmpApp.py README.md __pycache__ config.py templates
[ec2-user@ip-10-0-2-16 aws-code-main]$ vim config.py
[ec2-user@ip-10-0-2-16 aws-code-main]$ vim EmpApp.py
[ec2-user@ip-10-0-2-16 aws-code-main]$
```

Step 55: Now in DynamoDB block change the table name to your table name which you have created in DynamoDb console and then save the file.



```
# Save image file metadata in DynamoDB #
print("Uploading to S3 success... saving metadata in dynamodb... ")

try:
    dynamodb_client = boto3.client('dynamodb', region_name='us-east-1')
    dynamodb_client.put_item(
        TableName='metadata_table',
        Item={
            'emp_id': {
                'N': emp_id
            },
            'image_url': {
                'S': object_url
            }
        }
    )
except Exception as e:
    program_msg = "Flask could not update DynamoDB table with S3 object URL"
    return str(e)
except Exception as e:
    return str(e)

finally:
    cursor.close()

print("all modification done ... ")
return render_template('AddEmpOutput.html', name=emp_name)

@app.route('/getemp', methods=['GET', 'POST'])
def GetEmp():
    return render_template("GetEmp.html")

@app.route('/fetchdata', methods=['GET', 'POST'])
def Fetchdata():
    emp_id = request.form['emp_id']

    output = {}
    select_sql = "SELECT emp_id, first_name, last_name, pri_skill, location from employee where emp_id=%s"
    cursor = db_conn.cursor()
    cursor.execute(select_sql, (emp_id,))

    for row in cursor:
        output = {
            'emp_id': row[0],
            'first_name': row[1],
            'last_name': row[2],
            'pri_skill': row[3],
            'location': row[4]
        }

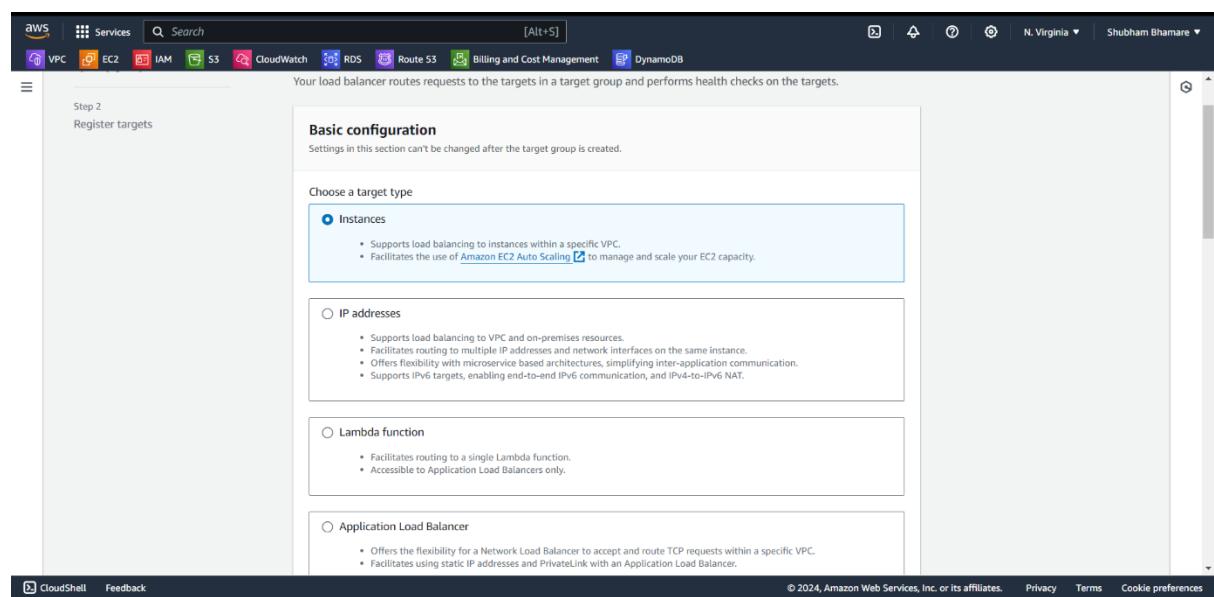
    cursor.close()
    db_conn.close()

    return render_template('AddEmpOutput.html', name=output['first_name'])

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

UNREGISTERED VERSION Please support MoboTerm by subscribing to the professional edition here: <https://moboterm.mobatek.net>

Step 66: Now go to target group and create new target group for your instance.



Step 67: Select your private instance only.

The screenshot shows the 'Register targets' step in the 'Target groups' creation wizard. It displays a table of available instances (1/2) with two entries: 'i-047fcc45d44435040' (Private, Running, launch-wizard-1) and 'i-09f85865a92f14d9d' (Public, Running, launch-wizard-1). A port range '80' is selected for routing traffic. A button 'Include as pending below' is visible.

Step 68: Target group is created successfully.

The screenshot shows the 'Project-TG' target group details page. A success message indicates it was created successfully. The 'Details' section shows the target type as 'Instance', protocol as 'HTTP: 80', version as 'HTTP1', and VPC as 'vpc-0b4d2bdd723cbad61'. The table summarizes target health: 1 total target, 0 healthy, 0 unhealthy, 1 unused, 0 initial, and 0 draining.

Step 69: Go to load balance and create one application load balancer.

The screenshot shows the 'Basic configuration' step of the AWS Load Balancer creation wizard. It includes fields for the load balancer name ('Project -LB'), scheme ('Internet-facing'), IP address type ('IPv4'), and network mapping settings. The VPC dropdown is set to 'Project-vpc'. The 'Mappings' section shows two subnets selected: 'us-east-1a (use1-az1)' and 'us-east-1b (use1-az2)'. A warning message indicates that the selected subnet does not have a route to an internet gateway, which is required for internet traffic to reach the load balancer.

Step 70: In networking select your VPC and both subnets.

The screenshot shows the 'Network mapping' step of the AWS Load Balancer creation wizard. It displays the selected VPC ('Project-vpc') and its subnets ('us-east-1a (use1-az1)' and 'us-east-1b (use1-az2)'). A warning message highlights that the selected subnet does not have a route to an internet gateway, which is required for internet traffic to reach the load balancer.

Step 71: Select the security group (All Traffic) which is also attached to your instances.

The screenshot shows the AWS CloudFront console. In the top navigation bar, the 'VPC' service is selected. Below the navigation bar, there is a search bar and a dropdown menu for selecting up to 5 security groups. A specific security group, 'launch-wizard-1' (sg-0aab574b6012c7c22), is highlighted. The main content area displays the 'Security groups' section with a brief description and a dropdown menu for selecting up to 5 security groups. The selected security group is listed as 'Select up to 5 security groups'. At the bottom of the page, there are links for CloudShell, Feedback, and copyright information.

Step 72: Select your target group.

The screenshot shows the AWS CloudFront console. In the top navigation bar, the 'VPC' service is selected. Below the navigation bar, there is a search bar and a dropdown menu for selecting up to 5 security groups. A specific security group, 'launch-wizard-1' (sg-0aab574b6012c7c22), is highlighted. The main content area displays the 'Listeners and routing' section with a brief description. It shows a 'Listener HTTP:80' configuration. The protocol is set to 'HTTP' and the port is '80'. The 'Default action' dropdown is set to 'Forward to' and has a placeholder 'Select a target group'. Below this, there is a link to 'Create target group'. Further down, there is a section for 'Listener tags - optional' with a button to 'Add listener tag' and a note that you can add up to 50 more tags. At the bottom of the page, there are links for CloudShell, Feedback, and copyright information.

Step 73: Load balancer is created successfully.

The screenshot shows the AWS EC2 Load Balancers console. A green success message at the top states: "Successfully created load balancer: ProjectLB. It might take a few minutes for your load balancer to fully set up and route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks." Below this, the "ProjectLB" load balancer is listed with its details. The "Details" section shows the following information:

Load balancer type	Status	VPC	IP address type
Application	Provisioning	vpc-0b4d2bdd723cbad61	IPv4
Scheme	Hosted zone	Availability Zones	Date created
Internet-facing	Z35SXDOTRQ7X7K	subnet-03144e87f66a97a26 us-east-1a (use1-az1)	March 24, 2024, 20:21 (UTC+05:30)
		subnet-0f451eab54e2401d4 us-east-1b (use1-az2)	
Load balancer ARN	DNS name	Info	
arn:aws:elasticloadbalancing:us-east-1:315465887624:loadbalancer/app/ProjectLB/c9fbf86fc957e21	ProjectLB-1800548799.us-east-1.elb.amazonaws.com (A Record)		

Below the details, there are tabs for "Listeners and rules", "Network mapping", "Resource map - new", "Security", "Monitoring", "Integrations", "Attributes", and "Tags". The bottom of the page includes standard AWS navigation links like CloudShell, Feedback, and copyright information: "© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

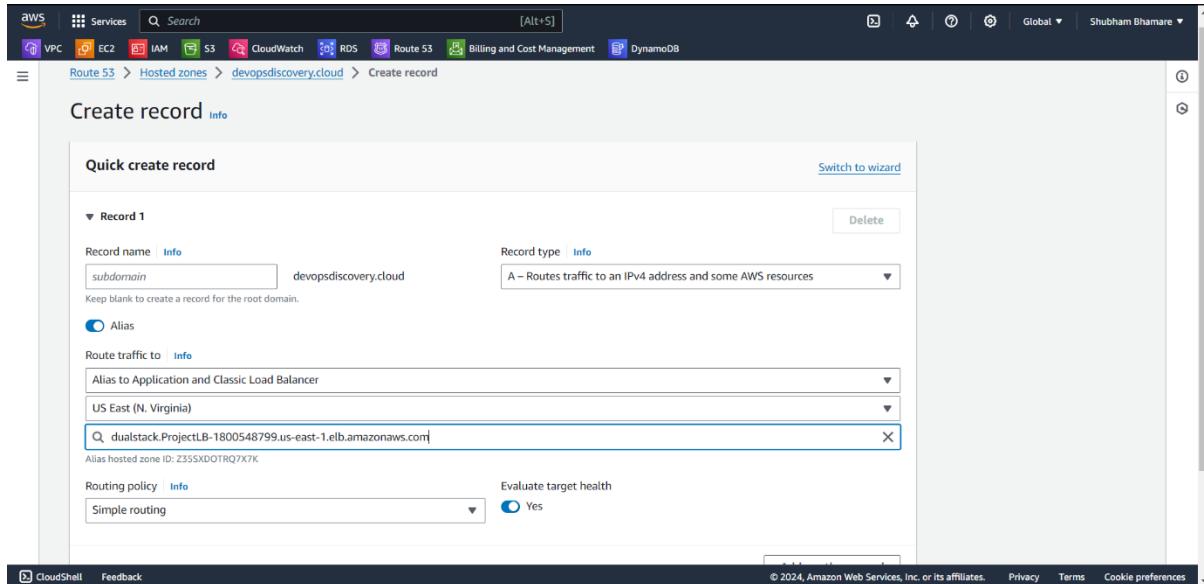
Step 74: Go to Route 53 and create one hosted zone for your Domain. Change name servers for your domain into your domain provider.

The screenshot shows the AWS Route 53 Hosted Zones console. A hosted zone named "devopsdiscovery.cloud" is selected. The "Hosted zone details" section shows the following information:

Records (2)	DNSSEC signing	Hosted zone tags (0)
Records (2) Info Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings. Filter records by property or value Type Routing policy Alias Value/Route traffic to TTL (s...) Health ... ns-1840.awsdns-38.co.uk. ns-74.awsdns-09.com. ns-601.awsdns-11.net. ns-1210.awsdns-23.org. ns-1840.awsdns-38.co.uk. a... 900 - devopsdis... NS Simple - No ns-1840.awsdns-38.co.uk. 172800 - devopsdis... SOA Simple - No ns-1840.awsdns-38.co.uk. a... 900 -		

Below the records, there are tabs for "Records (2)", "DNSSEC signing", and "Hosted zone tags (0)". The bottom of the page includes standard AWS navigation links like CloudShell, Feedback, and copyright information: "© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

Step 75: Create one record with any subdomain , record type – A, enable alias, route traffic to the application load balancer, Select region, and select your Load balancer and create record.



Step 76: Now run command “sudo yum install python3-pip -y” to install python on your private instance.

```

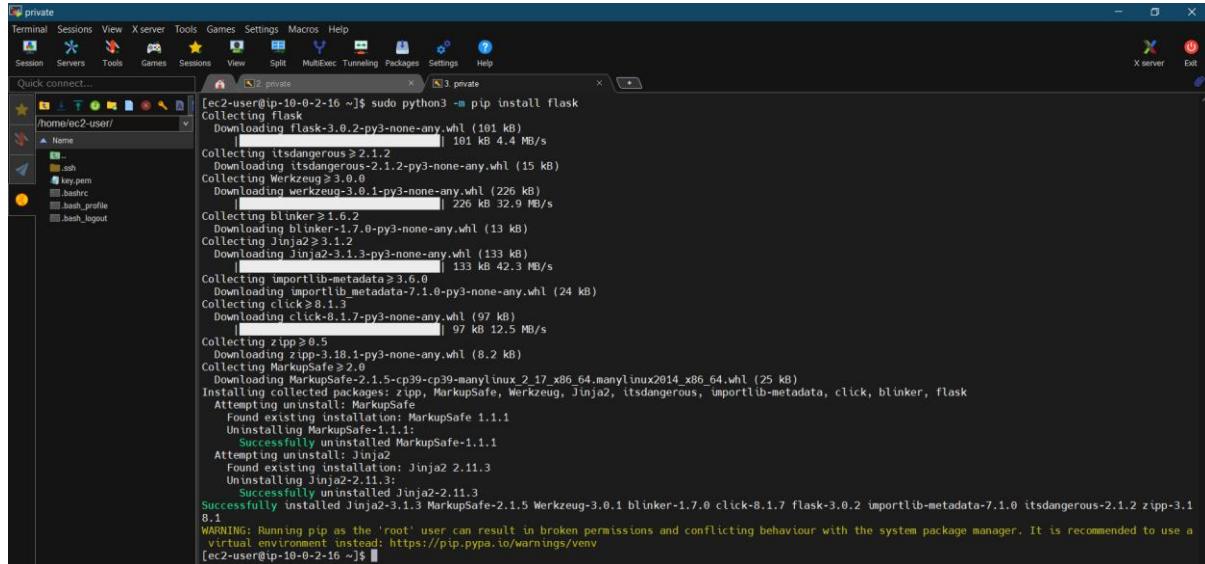
private
Terminal Sessions View Xserver Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Packages Settings Help
Quick connect...
[ec2-user@ip-10-0-2-16 ~]$ sudo yum install python3-pip -y
Last metadata expiration check: 1:23:04 ago on Sun Mar 24 14:06:22 2024.
Dependencies resolved.
=====
Package           Architecture Version      Repository   Size
=====
Installing:
  python3-pip      noarch    21.3.1-2.amzn2023.0.7   amazonlinux 1.8 M
Installing weak dependencies:
  libCRYPT-compat  x86_64    4.4.33-7.amzn2023          amazonlinux 92 k
=====
Transaction Summary
=====
Install 2 Packages

Total download size: 1.9 M
Installed size: 11 M
Downloading Packages:
(1/2): libCRYPT-compat-4.4.33-7.amzn2023.x86_64.rpm          1.4 MB/s | 92 kB  00:00
(2/2): python3-pip-21.3.1-2.amzn2023.0.7.noarch.rpm          16 MB/s | 1.8 MB  00:00
Total                                         10 MB/s | 1.9 MB  00:00
=====
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing   :
  Installing  : libCRYPT-compat-4.4.33-7.amzn2023.x86_64          1/1
  Installing  : python3-pip-21.3.1-2.amzn2023.0.7.noarch          2/2
  Running scriptlet: python3-pip-21.3.1-2.amzn2023.0.7.noarch
  Verifying   : libCRYPT-compat-4.4.33-7.amzn2023.x86_64          2/2
  Verifying   : python3-pip-21.3.1-2.amzn2023.0.7.noarch
=====
Installed:
  libCRYPT-compat-4.4.33-7.amzn2023.x86_64                  python3-pip-21.3.1-2.amzn2023.0.7.noarch
=====
Complete!
[ec2-user@ip-10-0-2-16 ~]$ 

```

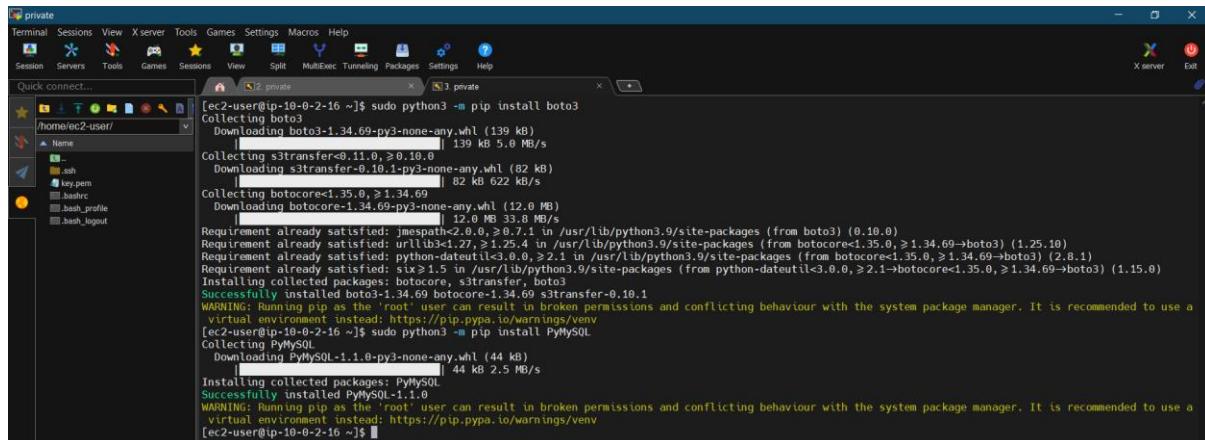
UNREGISTERED VERSION Please support MoboTerm by subscribing to the professional edition here: <http://moboterm.mobostek.net>

Step 77: Now run command “sudo python3 -m pip install flask” to download dependencies required for running python application.



```
[ec2-user@ip-10-0-2-16 ~]$ sudo python3 -m pip install flask
Collecting flask
  Downloading flask-3.0.2-py3-none-any.whl (101 kB)
Collecting itsdangerous >= 2.1.2
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting Werkzeug >= 3.0.0
  Downloading werkzeug-3.0.1-py3-none-any.whl (226 kB)
Collecting blinker >= 1.6.2
  Downloading blinker-1.6.2-py3-none-any.whl (13 kB)
Collecting Jinja2 >= 3.1.2
  Downloading Jinja2-3.1.3-py3-none-any.whl (133 kB)
Collecting importlib-metadata >= 3.6.0
  Downloading importlib_metadata-7.1.0-py3-none-any.whl (24 kB)
Collecting click >= 8.1.3
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
Collecting zipp >= 0.5
  Downloading zipp-3.18.1-py3-none-any.whl (8.2 kB)
Collecting MarkupSafe >= 2.0
  Downloading MarkupSafe-2.1.5-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (25 kB)
Installing collected packages: zipp, MarkupSafe, Werkzeug, Jinja2, itsdangerous, importlib-metadata, click, blinker, flask
  Attempting uninstall: MarkupSafe
    Found existing installation: MarkupSafe 1.1.1
    Uninstalling MarkupSafe-1.1.1:
      Successfully uninstalled MarkupSafe-1.1.1
  Attempting uninstall: Jinja2
    Found existing installation: Jinja2 2.11.3
    Uninstalling Jinja2-2.11.3:
      Successfully uninstalled Jinja2-2.11.3
Successfully installed Jinja2-3.1.3 MarkupSafe-2.1.5 Werkzeug-3.0.1 blinker-1.7.0 flask-3.0.2 importlib-metadata-7.1.0 itsdangerous-2.1.2 zipp-3.1
8.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
[ec2-user@ip-10-0-2-16 ~]$
```

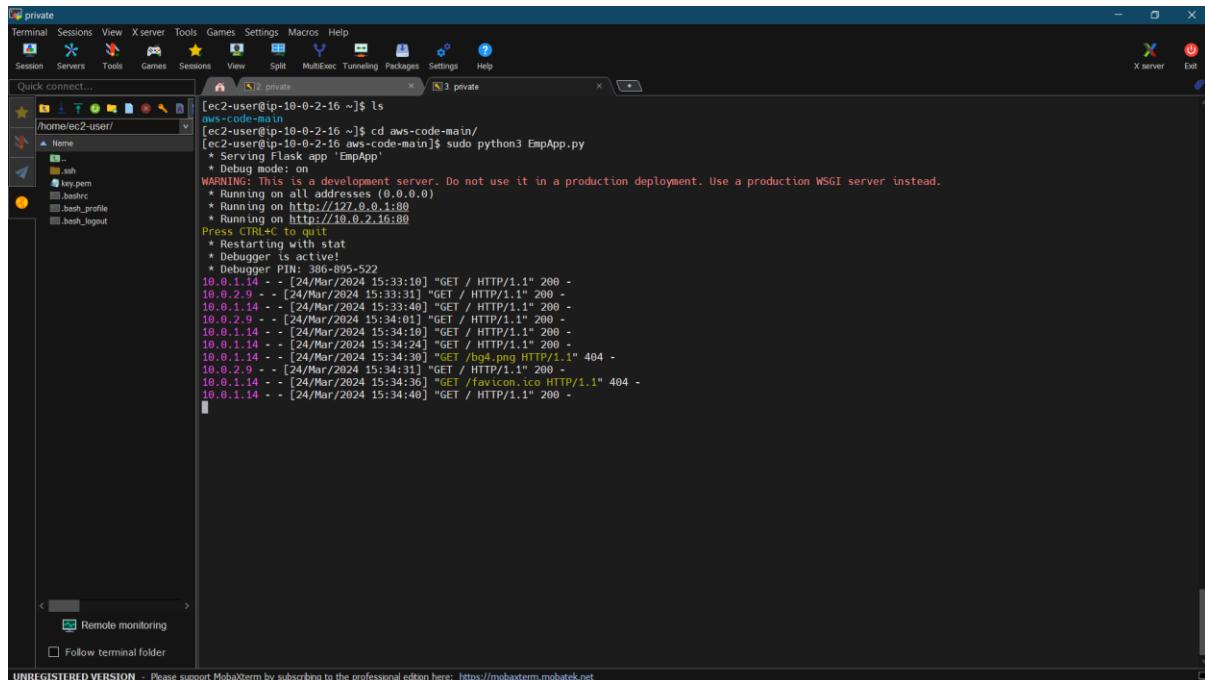
Step 78: “sudo python3 -m pip install PyMySQL”, “sudo python3 -m pip install boto3” to install required dependencies.



```
[ec2-user@ip-10-0-2-16 ~]$ sudo python3 -m pip install boto3
Collecting boto3
  Downloading boto3-1.34.69-py3-none-any.whl (139 kB)
Collecting s3transfer<0.11.0,>=0.10.0
  Downloading s3transfer-0.10.1-py3-none-any.whl (82 kB)
Collecting botocore<1.35.0,>=1.34.69
  Downloading botocore-1.34.69-py3-none-any.whl (12.0 MB)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3.9/site-packages (from boto3) (0.18.0)
Requirement already satisfied: urllib3<1.27,>1.25.4 in /usr/lib/python3.9/site-packages (from botocore<1.35.0,>=1.34.69->boto3) (1.25.10)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore<1.35.0,>=1.34.69->boto3) (2.8.1)
Requirement already satisfied: six<1.5.2,>=1.5.1 in /usr/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.35.0,>=1.34.69->boto3) (1.15.0)
Installing collected packages: botocore, s3transfer, boto3
  Successfully installed boto3-1.34.69 botocore-1.34.69 s3transfer-0.10.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
[ec2-user@ip-10-0-2-16 ~]$ sudo python3 -m pip install PyMySQL
Collecting PyMySQL
  Downloading PyMySQL-1.1.0-py3-none-any.whl (44 kB)
Installing collected packages: PyMySQL
  Successfully installed PyMySQL-1.1.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
[ec2-user@ip-10-0-2-16 ~]$
```

Step 79: Now go to application directory and run command

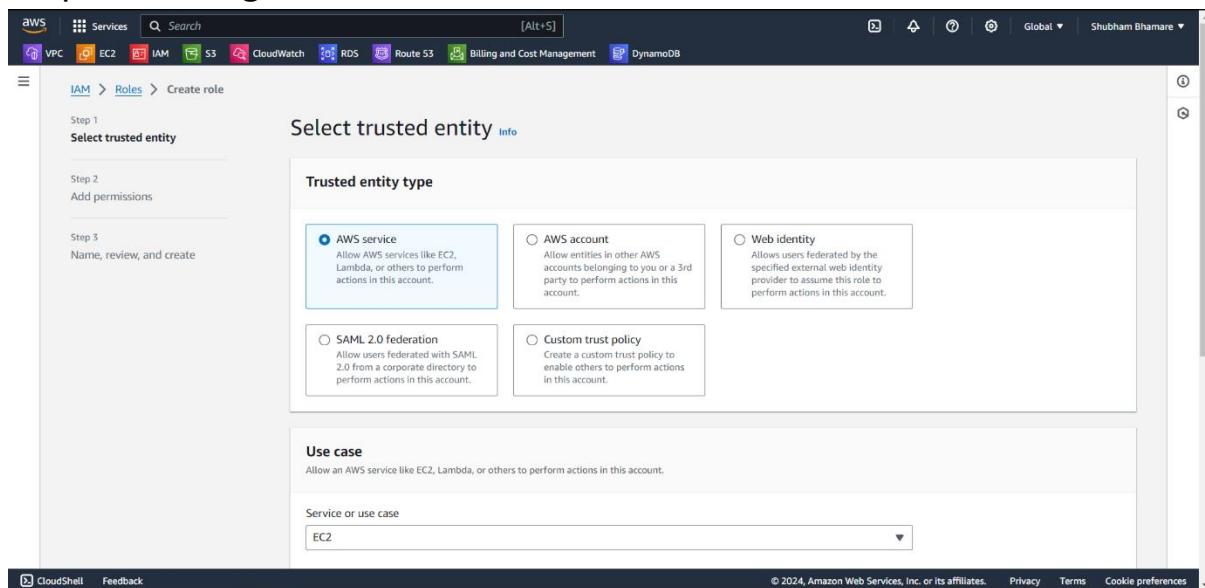
“sudo python3 EmpApp.py” on your private instance to start python application.



The screenshot shows a terminal window in MobaXterm with the title 'private'. The terminal session is running on an EC2 instance with IP 10.0.2.16. The user has navigated to the directory /home/ec2-user/ and run the command 'sudo python3 EmpApp.py'. The output shows the application is serving on port 80. A warning message is displayed: 'WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.' Below this, several log entries from the Flask application are listed, showing requests for various static files like index.html, bg4.png, and favicon.ico.

```
[ec2-user@ip-10-0-2-16 ~]$ ls
aws-code-main
[ec2-user@ip-10-0-2-16 ~]$ cd aws-code-main/
[ec2-user@ip-10-0-2-16 aws-code-main]$ sudo python3 EmpApp.py
* Serving Flask app "EmpApp"
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:80
* Running on http://10.0.2.16:80
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 386-895-522
10.0.1.14 - [24/Mar/2024 15:33:18] "GET / HTTP/1.1" 200 -
10.0.2.9 - [24/Mar/2024 15:33:18] "GET /favicon.ico HTTP/1.1" 200 -
10.0.1.14 - [24/Mar/2024 15:33:48] "GET / HTTP/1.1" 200 -
10.0.2.9 - [24/Mar/2024 15:34:01] "GET / HTTP/1.1" 200 -
10.0.1.14 - [24/Mar/2024 15:34:10] "GET / HTTP/1.1" 200 -
10.0.1.14 - [24/Mar/2024 15:34:24] "GET / HTTP/1.1" 200 -
10.0.1.14 - [24/Mar/2024 15:34:30] "GET /bg4.png HTTP/1.1" 404 -
10.0.2.9 - [24/Mar/2024 15:34:31] "GET / HTTP/1.1" 200 -
10.0.1.14 - [24/Mar/2024 15:34:36] "GET /favicon.ico HTTP/1.1" 404 -
10.0.1.14 - [24/Mar/2024 15:34:40] "GET / HTTP/1.1" 200 -
```

Step 80: Now go to IAM console and create one IAM role for EC2.



Step 81: Create role with DynamoDB,RDS,S3 full access.

The screenshot shows the AWS IAM console with the 'Create New Role' wizard. Step 2: Add permissions. A table lists three AWS managed policies:

Policy name	Type	Attached as
AmazonDynamoDBFullAccess	AWS managed	Permissions policy
AmazonRDSFullAccess	AWS managed	Permissions policy
AmazonS3FullAccess	AWS managed	Permissions policy

Step 82: Provide name to IAM role and create role.

The screenshot shows the AWS IAM console with the 'Create New Role' wizard. Step 3: Name, review, and create. The role details are as follows:

- Role name:** Project
- Description:** Allows EC2 Instances to call AWS services on your behalf.

Step 1: Select trusted entities. A partial JSON trust policy is visible:

```
1 - [{}  
2 - "Version": "2012-10-17",  
3 - "Statement": [  
4 - ]]
```

Step 83: Now attach that role role to private EC2 instance.

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with links like EC2 Dashboard, VPC, IAM, S3, CloudWatch, RDS, Route 53, and Billing and Cost Management. The main area displays a table of instances. One instance is selected: "Private" (Instance ID: i-047fcc45d44435040). The instance is running, t2.micro type, with 2/2 checks passed. A context menu is open over this instance, with the "Modify IAM role" option highlighted. Below the table, a modal window titled "Instance: i-047fcc45d44435040 (Private)" is open, showing details like Instance ID, Public IPv4 address (empty), Private IP4 addresses (10.0.2.16), and Public IPv4 DNS (empty).

Step 84: Update IAM role.

The screenshot shows the "Modify IAM role" dialog box. At the top, it says "EC2 > Instances > i-047fcc45d44435040 > Modify IAM role". The dialog has a heading "Modify IAM role" with a link to "Info". Below that is a sub-instruction "Attach an IAM role to your instance." It shows the "Instance ID" as "i-047fcc45d44435040 (Private)". Under "IAM role", there's a note: "Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance." A dropdown menu is open, showing "Project" as the selected option. There are also "Create new IAM role" and "Cancel" buttons, and a prominent orange "Update IAM role" button at the bottom right.

Step 85: Now copy your domain name with subdomain.

The screenshot shows the AWS Route 53 console. A blue banner at the top says "www.devopsdiscovery.cloud was successfully updated." Below it, a table lists three DNS records:

Type	Name	Value	TTL (seconds)	Routing policy
NS	devopsdiscovery.cloud	ns-1840.awsdns-74.awsdns-601.awsdns-1210.awsdn	-	Simple
SOA	devopsdiscovery.cloud	ns-1840.awsdn	-	Simple
A	www.devopsdiscovery.cloud	dualstack.projectlb-1800548799.us-east-1.elb.amazonaws.com	-	Simple

A tooltip on the right says "Copied" and shows the URL "www.devopsdiscovery.cloud".

Step 86: Hit that URL on browser and fill all information, Upload image and click on “update database”.

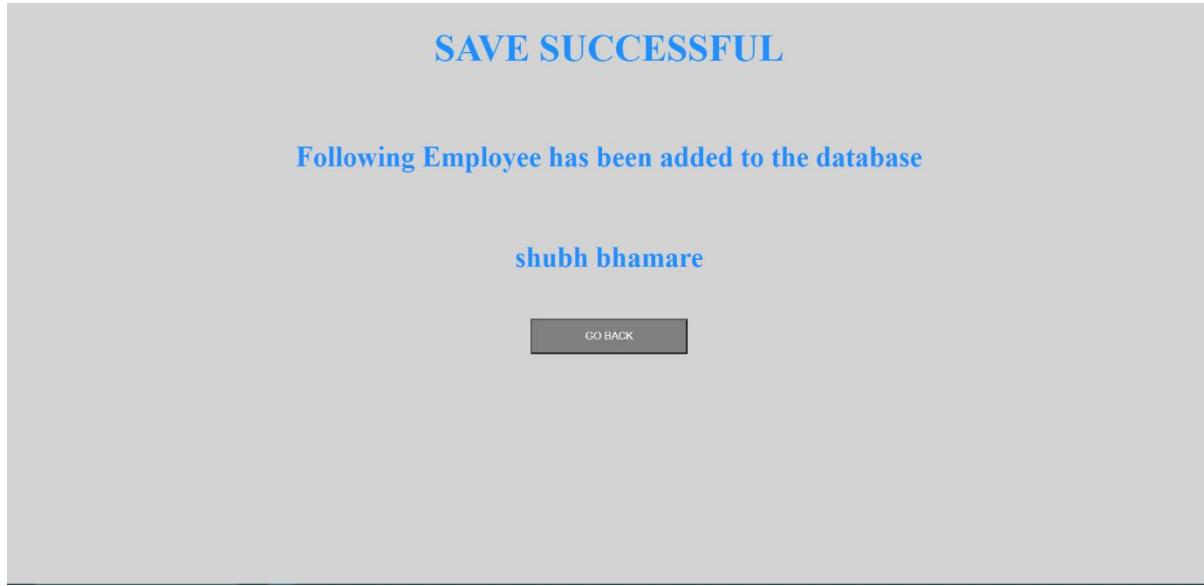
EMPLOYEE DATABASE

GET EMPLOYEE INFORMATION

Employee ID:	<input type="text" value="3"/>
First Name:	<input type="text" value="shubh"/>
Last Name:	<input type="text" value="bhamare"/>
Primary Skills:	<input type="text" value="AWS"/>
Location:	<input type="text" value="Dhule"/>

Image: ShubhamBhamare.png

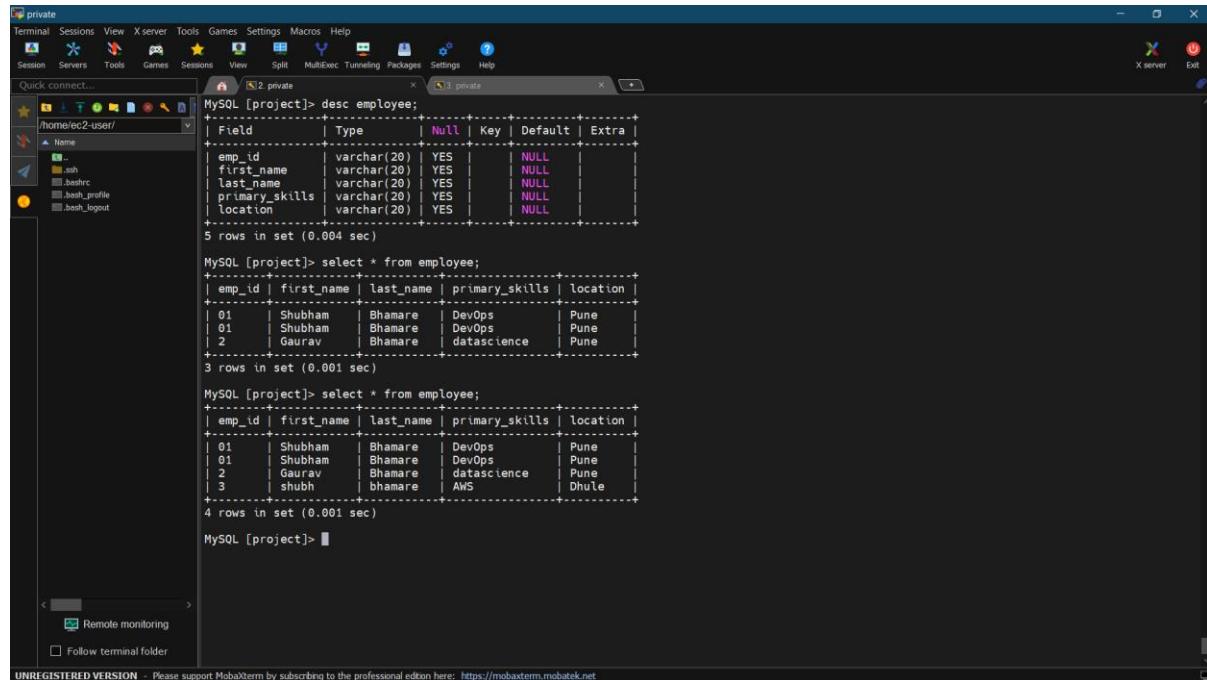
Step 87: Information is added to database successfully.



Step 88: As we can see on private instance that “Data inserted in MySQL RDS , Uploading image to s3, upload successful, saving metadata in DynamoDB, All modification done”.

A screenshot of a MobaXterm terminal window titled "private". The terminal shows a session named "private" with a command-line interface. The log output includes commands like "aws config", "sudo python3 EmpApp.py", and "curl" requests. It also shows messages about data being inserted into MySQL RDS and uploaded to S3, along with metadata being saved in DynamoDB. The terminal interface includes tabs, a file browser, and various toolbars.

Step 89: Run command “select * from <table_name>” to check if the data stored into RDS Database.



The screenshot shows a terminal window titled "private" in MobaXterm. It displays three MySQL commands:

```
MySQL [project]> desc employee;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| emp_id | varchar(20) | YES | NULL |          |          |
| first_name | varchar(20) | YES | NULL |          |          |
| last_name | varchar(20) | YES | NULL |          |          |
| primary_skills | varchar(20) | YES | NULL |          |          |
| location | varchar(20) | YES | NULL |          |          |
+-----+-----+-----+-----+-----+
5 rows in set (0.004 sec)

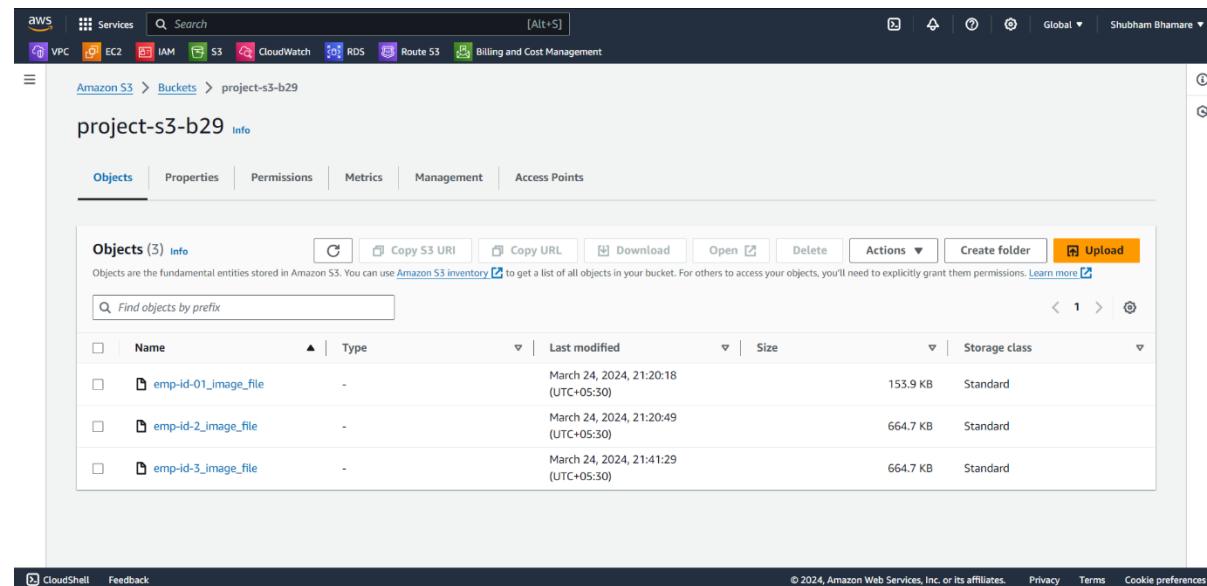
MySQL [project]> select * from employee;
+-----+-----+-----+-----+
| emp_id | first_name | last_name | primary_skills | location |
+-----+-----+-----+-----+
| 01    | Shubham   | Bhamare  | DevOps        | Pune      |
| 01    | Shubham   | Bhamare  | DevOps        | Pune      |
| 2     | Gaurav    | Bhamare  | datascience  | Pune      |
+-----+-----+-----+-----+
3 rows in set (0.001 sec)

MySQL [project]> select * from employee;
+-----+-----+-----+-----+
| emp_id | first_name | last_name | primary_skills | location |
+-----+-----+-----+-----+
| 01    | Shubham   | Bhamare  | DevOps        | Pune      |
| 01    | Shubham   | Bhamare  | DevOps        | Pune      |
| 2     | Gaurav    | Bhamare  | datascience  | Pune      |
| 3     | shubh     | bhamare  | AWS           | Dhule    |
+-----+-----+-----+-----+
4 rows in set (0.001 sec)

MySQL [project]>
```

At the bottom of the terminal, there is a note: "UNREGISTERED VERSION - Please support MobaTerm by subscribing to the professional edition here: <https://mobaterm.mobatek.net>".

Step 90: Check into your S3 bucket whet images is uploaded into your bucket or not.



The screenshot shows the AWS S3 console. The top navigation bar includes "Services", "Search", and links for VPC, EC2, IAM, S3, CloudWatch, RDS, Route 53, and Billing and Cost Management. The user "Shubham Bhamare" is logged in.

The main view shows the "Objects" tab for the bucket "project-s3-b29". The table lists three objects:

Name	Type	Last modified	Size	Storage class
emp-id-01_image_file	-	March 24, 2024, 21:20:18 (UTC+05:30)	153.9 KB	Standard
emp-id-2_image_file	-	March 24, 2024, 21:20:49 (UTC+05:30)	664.7 KB	Standard
emp-id-3_image_file	-	March 24, 2024, 21:41:29 (UTC+05:30)	664.7 KB	Standard

At the bottom of the page, there are links for "CloudShell", "Feedback", "© 2024, Amazon Web Services, Inc. or its affiliates.", "Privacy", "Terms", and "Cookie preferences".