

aws ec2 create-key-pair --key-name key-pair-poridhi-poc --query 'KeyMaterial' --output text > key-pair-poridhi-poc.pem



import pulumi

import pulumi\_aws as aws

# Create a VPC

vpc = aws.ec2.Vpc("my-vpc",

cidr\_block="10.0.0.0/16",

enable\_dns\_support=True,

enable\_dns\_hostnames=True

)

# Create an Internet Gateway

internet\_gateway = aws.ec2.InternetGateway("my-igw",

vpc\_id=vpc.id

)

# Create a Public Subnet

public\_subnet = aws.ec2.Subnet("public-subnet",

vpc\_id=vpc.id,

cidr\_block="10.0.1.0/24",

map\_public\_ip\_on\_launch=True

)

# Create a Private Subnet

private\_subnet = aws.ec2.Subnet("private-subnet",

vpc\_id=vpc.id,

cidr\_block="10.0.2.0/24",

map\_public\_ip\_on\_launch=False

)

# Create a Route Table for the Public Subnet

public\_route\_table = aws.ec2.RouteTable("public-route-table",

vpc\_id=vpc.id,

routes=[aws.ec2.RouteTableRouteArgs(

cidr\_block="0.0.0.0/0",

gateway\_id=internet\_gateway.id,

)]

)

# Associate the Public Subnet with the Public Route Table

public\_route\_table\_association = aws.ec2.RouteTableAssociation("public-route-table-association",

subnet\_id=public\_subnet.id,

route\_table\_id=public\_route\_table.id

)

# Create a route table for the private subnet

private\_route\_table = aws.ec2.RouteTable("private-route-table",

vpc\_id=vpc.id,

tags={

"Name": "my-private-route-table"

}

)

# Allocate an Elastic IP for the NAT Gateway

eip = aws.ec2.Eip("nat-eip", vpc=True)

# Create the NAT Gateway

nat\_gateway = aws.ec2.NatGateway("nat-gateway",

subnet\_id=public\_subnet.id,

allocation\_id=eip.id,

tags={

"Name": "my-nat-gateway"

}

)

# Create a route in the route table for the NAT Gateway

private\_route = aws.ec2.Route("nat-route",

route\_table\_id=private\_route\_table.id,

destination\_cidr\_block="0.0.0.0/0",

nat\_gateway\_id=nat\_gateway.id

)

# Associate the route table with the private subnet

private\_route\_table\_association = aws.ec2.RouteTableAssociation("private-route-table-association",

subnet\_id=private\_subnet.id,

route\_table\_id=private\_route\_table.id

)

# Create a Security Group for the Bastion Host

bastion\_sg = aws.ec2.SecurityGroup("bastion-sg",

vpc\_id=vpc.id,

description="Allow SSH from all IPs",

ingress=[

aws.ec2.SecurityGroupIngressArgs(

protocol="tcp",

from\_port=22,

to\_port=22,

cidr\_blocks=["0.0.0.0/0"], # Allow SSH from anywhere

),

],

egress=[

aws.ec2.SecurityGroupEgressArgs(

protocol="-1",

from\_port=0,

to\_port=0,

cidr\_blocks=["0.0.0.0/0"], # Allow all outbound traffic

),

],

)

# Create a Security Group for the MySQL Server

mysql\_sg = aws.ec2.SecurityGroup("mysql-sg",

vpc\_id=vpc.id,

description="Allow SSH from Bastion Host and MySQL from Bastion Host",

ingress=[

# Allow SSH from the Bastion Host

aws.ec2.SecurityGroupIngressArgs(

protocol="tcp",

from\_port=22,

to\_port=22,

security\_groups=[bastion\_sg.id], # Restrict SSH to the Bastion Host

),

# Allow MySQL from the Bastion Host

aws.ec2.SecurityGroupIngressArgs(

protocol="tcp",

from\_port=3306,

to\_port=3306,

security\_groups=[bastion\_sg.id], # Restrict MySQL to the Bastion Host

),

],

egress=[

aws.ec2.SecurityGroupEgressArgs(

protocol="-1",

from\_port=0,

to\_port=0,

cidr\_blocks=["0.0.0.0/0"], # Allow all outbound traffic

),

],

)

# Create the Bastion Host in the Public Subnet

bastion\_host = aws.ec2.Instance("bastion-host",

instance\_type="t2.micro",

ami="ami-0672fd5b9210aa093", # Replace with your desired AMI ID

vpc\_security\_group\_ids=[bastion\_sg.id],

subnet\_id=public\_subnet.id,

key\_name="key-pair-poridhi-poc", # Replace with your key pair name

tags={

"Name": "bastion-host",

}

)

# Create the MySQL Server in the Private Subnet

mysql\_server = aws.ec2.Instance("mysql-server",

instance\_type="t2.micro",

ami="ami-0672fd5b9210aa093", # Replace with your desired AMI ID

vpc\_security\_group\_ids=[mysql\_sg.id],

subnet\_id=private\_subnet.id,

key\_name="key-pair-poridhi-poc", # Replace with your key pair name

tags={

"Name": "mysql-server",

}

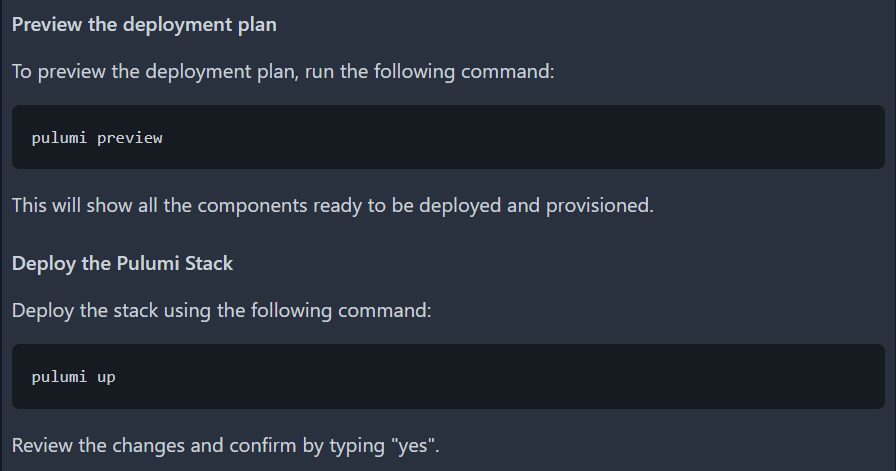
)

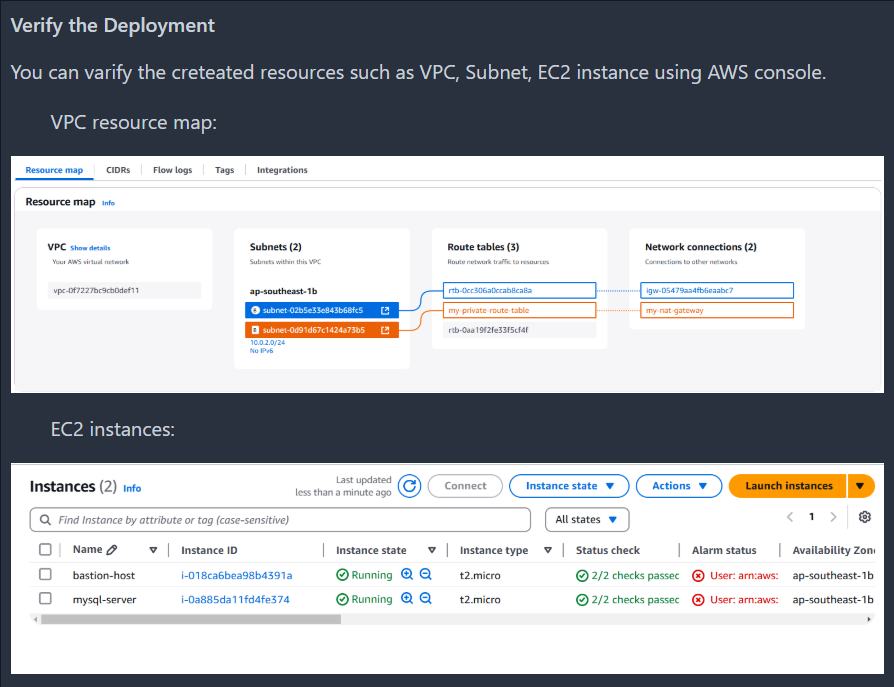
# Output the Public IP of the Bastion Host

pulumi.export("bastion\_host\_public\_ip", bastion\_host.public\_ip)

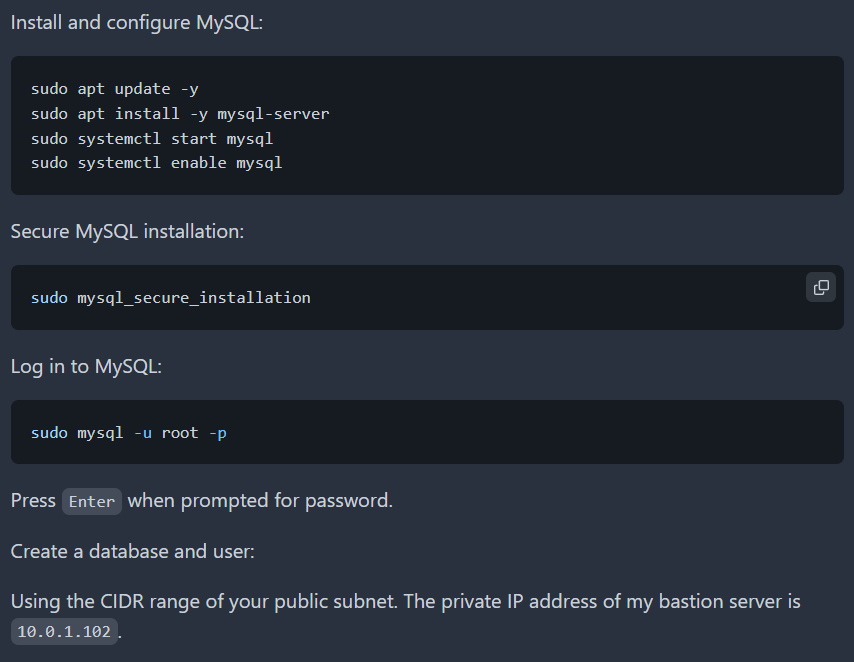
# Output the Private IP of the MySQL Server

pulumi.export("mysql\_server\_private\_ip", mysql\_server.private\_ip)











CREATE DATABASE demo\_db;

CREATE USER 'demo\_user'@'10.0.1.%' IDENTIFIED BY 'password';

GRANT ALL PRIVILEGES ON \*.\* TO 'demo\_user'@'10.0.1.%';

FLUSH PRIVILEGES;

EXIT;

CREATE USER 'demo\_user'@'localhost' IDENTIFIED BY 'password';

GRANT ALL PRIVILEGES ON \*.\* TO 'demo\_user'@'localhost';

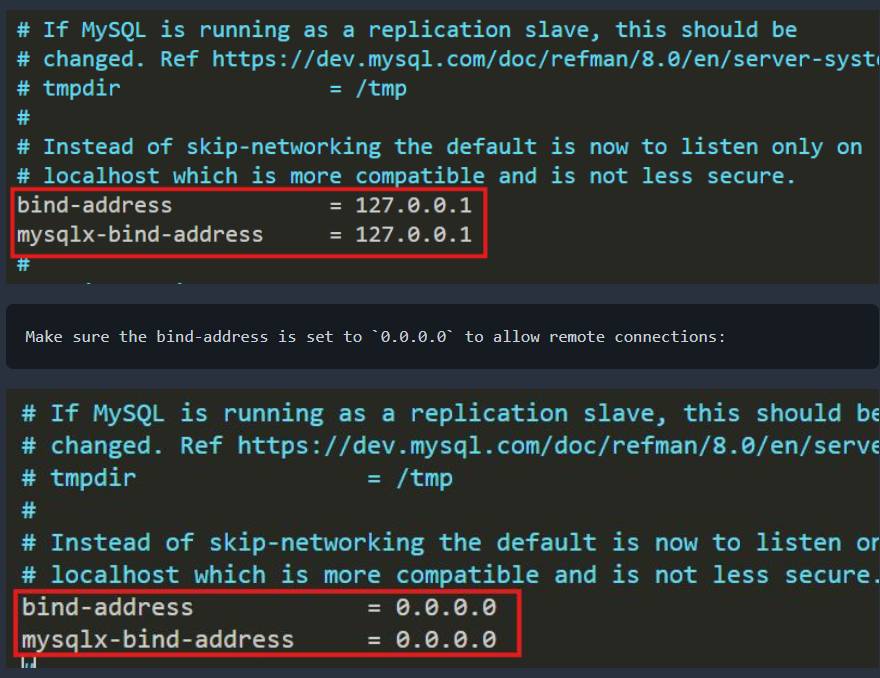
FLUSH PRIVILEGES;

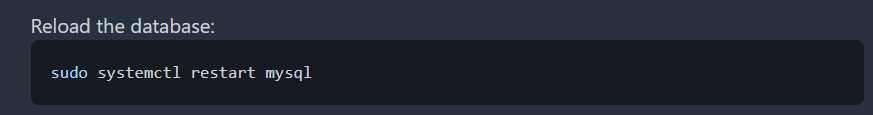
USE demo\_db;

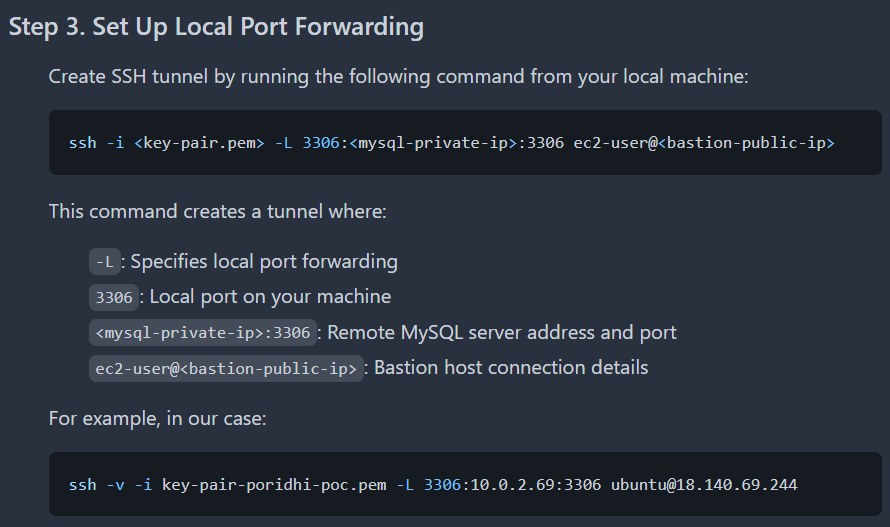
CREATE TABLE test\_table (id INT, name VARCHAR(50));

INSERT INTO test\_table VALUES (1, 'Test Data');

SELECT \* FROM test\_table;



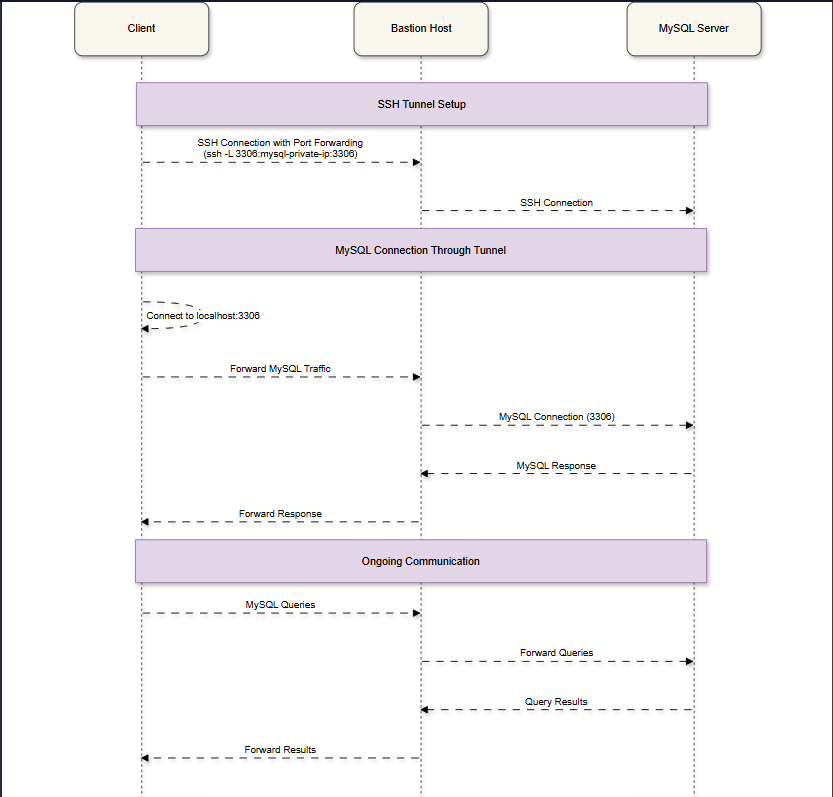
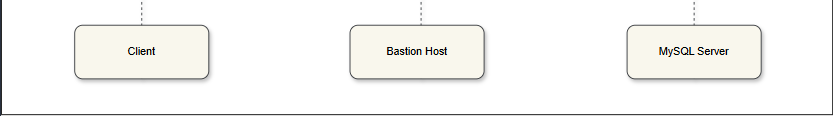




ssh -i <key-pair.pem> -L 3306:<mysql-private-ip>:3306 ec2-user@<bastion-public-ip>

ssh -f -N -i <key-pair.pem> -L 3306:<mysql-private-ip>:3306 ec2-user@<bastion-public-ip> [For running in Background]

How it works





mysql -u demo\_user -p -h 127.0.0.1 -P 3306

