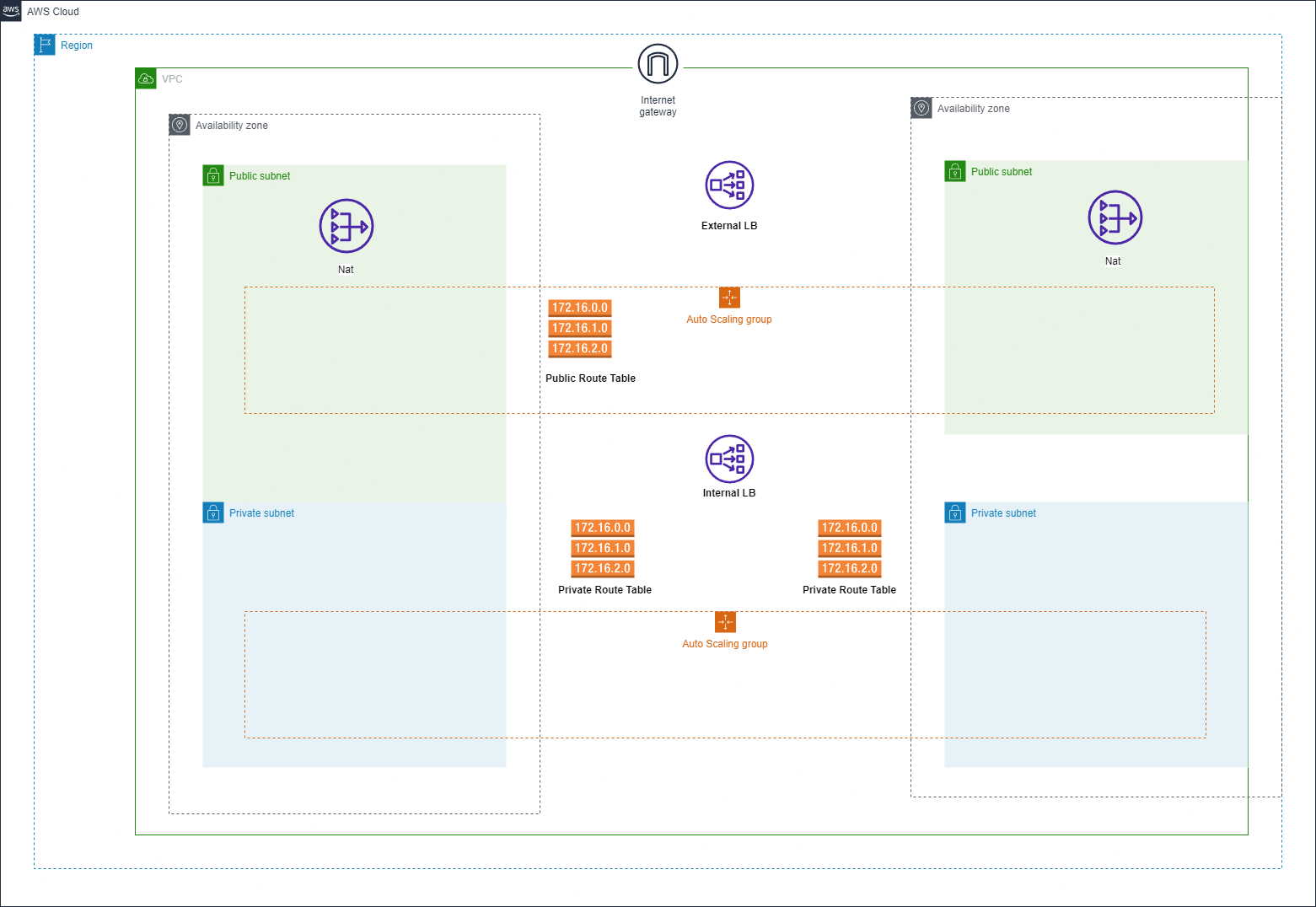
1. **Network Infrastrcture with Terraform**

Infrastructure Network Terraform

* 1. Create Virtual private Cloud (VPC)
  2. Create Four Subnet
     1. Two Public Subnet
     2. Two Private Subnet
  3. Create Router Table
  4. Create Route Table Association
  5. Create Internet Gateway
  6. Create Nat Gateway



1. **Setting up** **a EKS**
   1. Create Security Group
   2. Create Elastic Kubernetes Service (EKS Cluster)
   3. Set up a new IAM role with EKS permission
   4. Create Worker Nodes
   5. Set up a new IAM role with Worker Node
   6. Create Elastic Container Registry (ECR)
2. **Setting up a bastion host**
   1. Create Security Group
   2. Create Bastion Host [Jump Box]
   3. Private key: You should now be able to access the instance using the private key from the same key pair as used to create the instance

A highly available architecture that spans two Availability Zones. \*

A virtual private cloud (VPC) configured with public and private subnets according to AWS best practices, to provide you with your own virtual network on AWS. \*

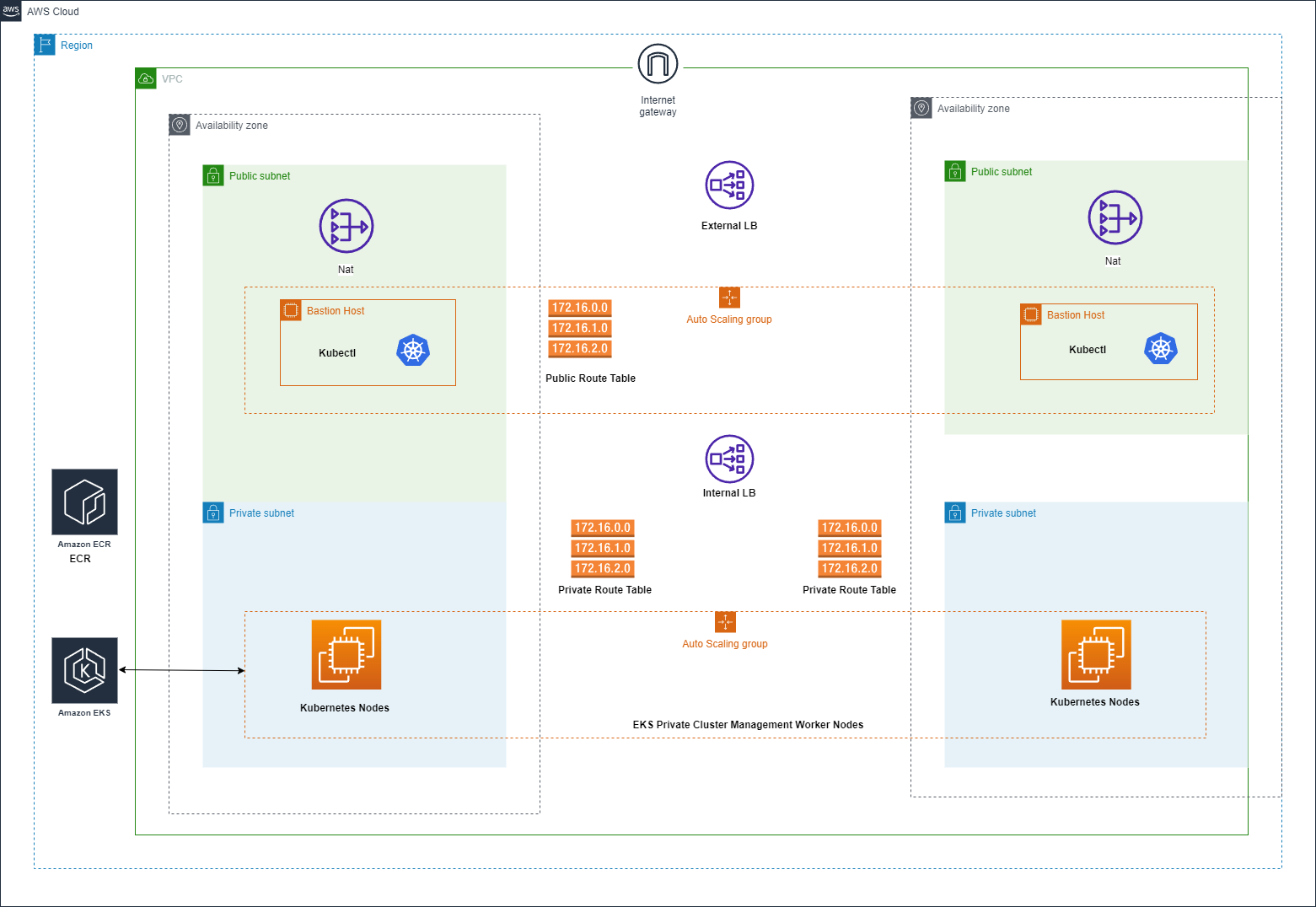
In the public subnets, NAT gateways are managed to allow outbound internet access for resources in the private subnets. \*

In one public subnet, a Linux bastion host in an Auto Scaling group to allow inbound Secure Shell (SSH) access to Amazon Elastic Compute Cloud (Amazon EC2) instances in private subnets. The bastion host is also configured with the Kubernetes kubectl command line interface for managing the Kubernetes cluster.

An Amazon EKS cluster, which provides the Kubernetes control plane.

In the private subnets, a group of Kubernetes nodes.

* **Network Infrastrcture Architecture Explained with Diagram**



## 

## **4. Accessing Private EKS Cluster From Bastion Host (EC2)?**

Before you start You will need to make sure you have the following components installed on bastion host (ec2) and set up before accessing Private EKS Cluster

1. AWS CLI – you need to link your account to the AWS CLI.
   1. AWS Configure
2. Kubectl – used for communicating with the cluster API server.
   1. **Install Kubernetes packages**
   2. Next, we will install the packages that we need to set up a Kubernetes control plane on this host. These packages are described in the following list:
   3. kubectl**:** The Kubernetes command-line client, which will allow us to interact with the Kubernetes API server.
3. Install Docker

So finally, Accessing Private EKS Cluster From Bastion Host (EC2)

` **aws sts get-caller-identity**`

` **aws eks update-kubeconfig --region *region-code* --name *my-cluster***`

**5. Creating Docker Image and pushing the image to Amazon ECR repository**

Retrieve an authentication token and authenticate your Docker client to your registry. Use the AWS CLI:

**`aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws/o0k2y0g4`**

Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions. You can skip this step if your image is already built:

**`docker build -t devops`.**

After the build completes, tag your image so you can push the image to this repository:

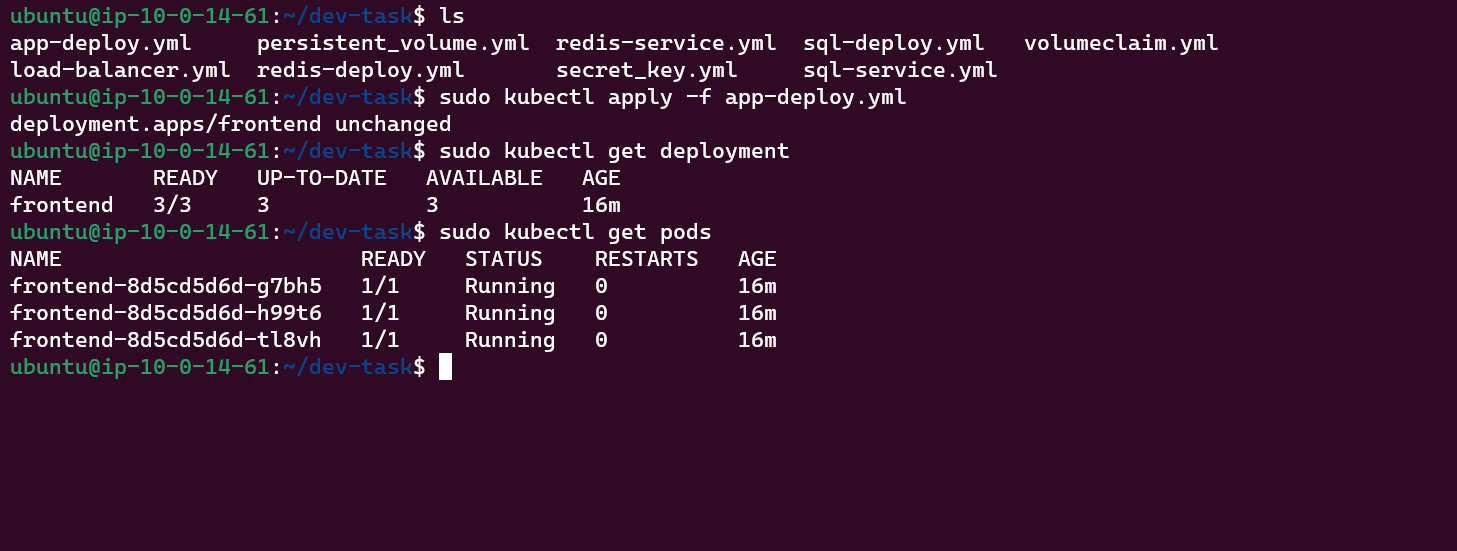
**`docker tag devops:latest public.ecr.aws/o0k2y0g4/devops:latest`**

Run the following command to push this image to your newly created AWS repository:

**`docker push public.ecr.aws/o0k2y0g4/devops:latest`**

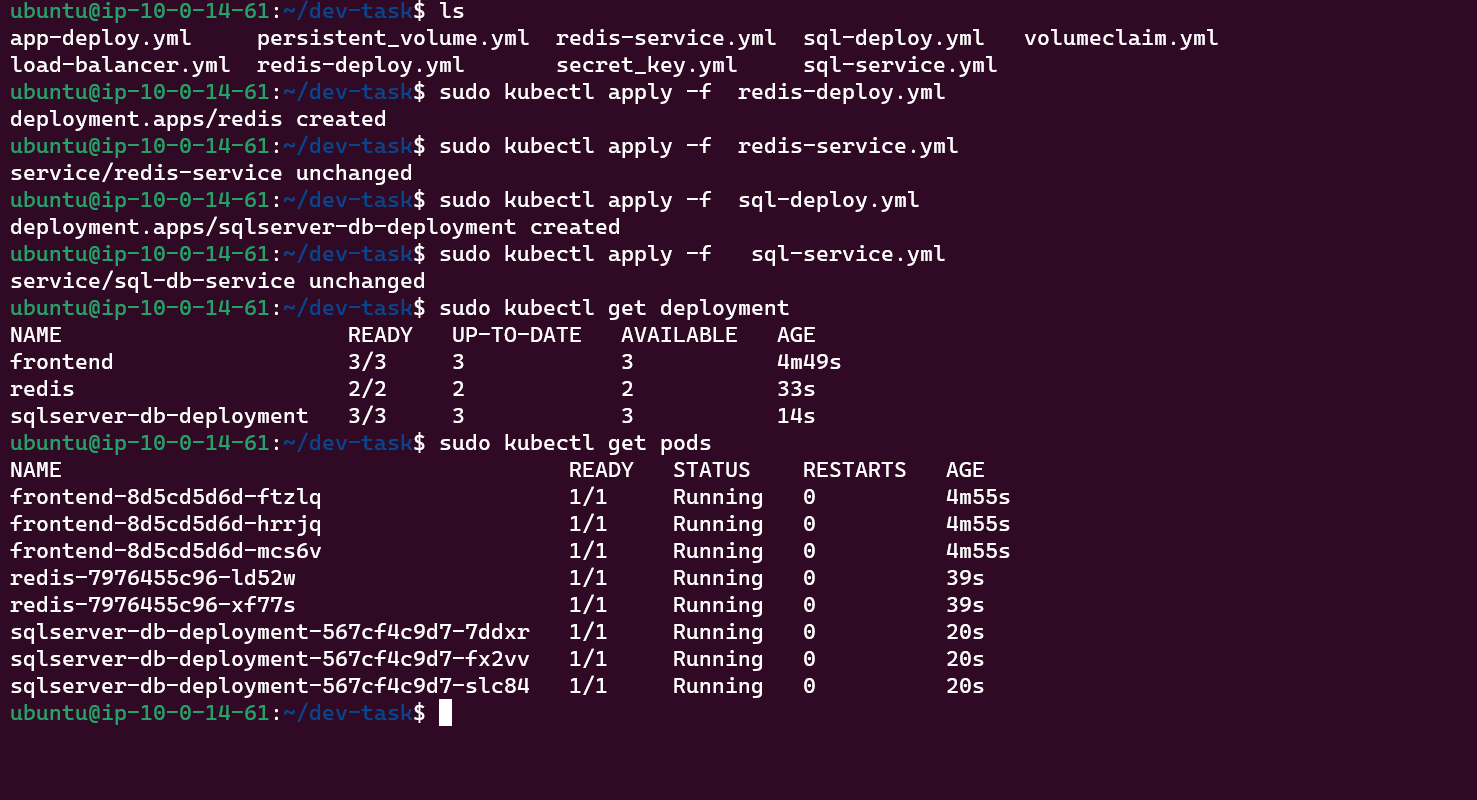
# **Deploy Your Application on Kubernetes**

## Deploy your application with the deployments

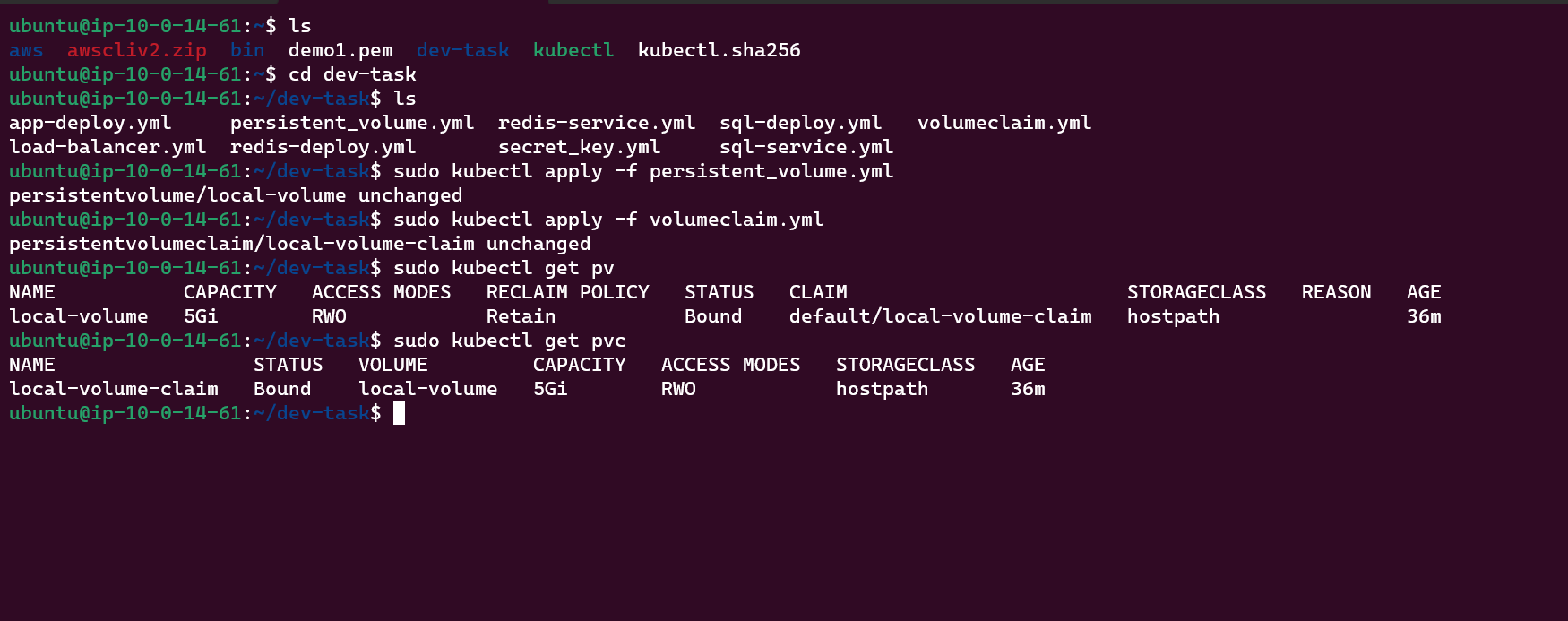


# **Deploy the Database**

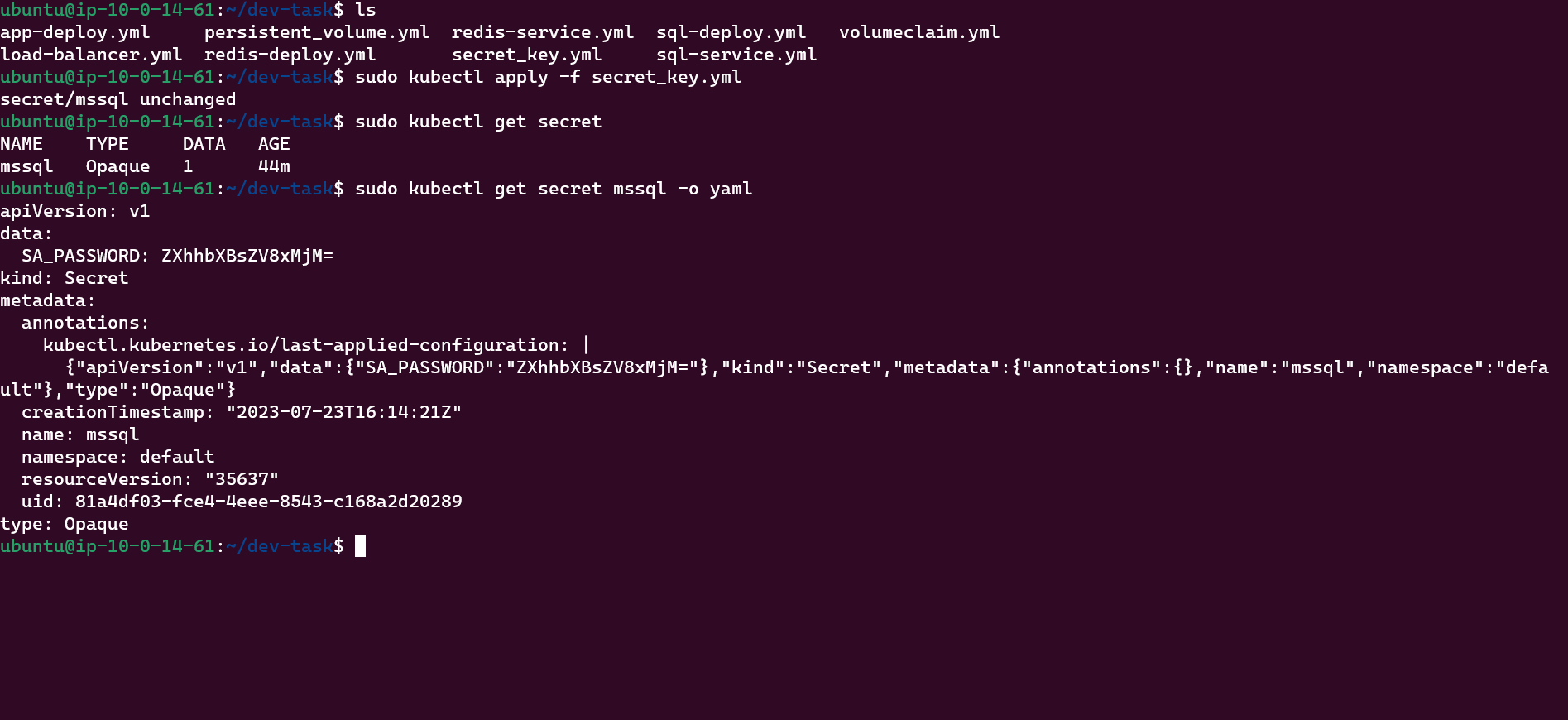
## Create a deployment for SQL Server and Redis



* **Kubernetes Volume and Secret Database Information**
  + Database is not persistent so,
  + There are many of type that make persistent data one of them:
    - Host Path Persistent Volume



Another problem that we can solve now is the storage of sensitive data in a Kubernetes cluster, like the password to access our database. To do this, Kubernetes provides a tool known as a secret, which we already glimpsed when we talked about the token to access the Kubernetes Dashboard.



* + **Create an External Load Balancer**

you have the option of automatically creating a cloud load balancer. This provides

an externally accessible IP address that sends traffic to the correct port on your cluster

nodes, *provided your cluster runs in a supported environment and is configured with the correct cloud load balancer provider package.*

