**Deploying Wordpress application on Kubernetes with AWS RDS using terraform**

**Task-6**

**Deploy the Wordpress application on Kubernetes and AWS using terraform including the following steps;**

**1. Write an Infrastructure as code using terraform, which automatically deploy the Wordpress application**

**2. On AWS, use RDS service for the relational database for Wordpress application.**

**3. Deploy the Wordpress as a container either on top of Minikube or EKS or Fargate service on AWS**

**4. The Wordpress application should be accessible from the public world if deployed on AWS or through workstation if deployed on Minikube.**

**GIT Link:**

**Amazon Relational Database Service (RDS)**

**Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching and backups. It frees you to focus on your applications so you can give them the fast performance, high availability, security and compatibility they need.Amazon RDS is available on several database instance types - optimized for memory, performance or I/O - and provides you with six familiar database engines to choose from, including Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database, and SQL Server. You can use the AWS Database Migration Service to easily migrate or replicate your existing databases to Amazon RDS.**



**Provider Setup**

**The easiest way to configure the provider is by creating/generating a config in a default location (~/.kube/config). That allows you to leave the provider block completely empty.**

provider "kubernetes" {}

resource "kubernetes\_service" "example" {

metadata {

name = "wp-service"

labels = {

app = "wordpress"

}

}

spec {

selector = {

app = "wordpress"

tier = "frontend"

}

port {

node\_port = 30000

port = 80

target\_port = 80

}

type = "NodePort"

}

}

resource "kubernetes\_persistent\_volume\_claim" "pvc" {

metadata {

name = "wp-pvc"

labels = {

app = "wordpress"

tier = "frontend"

}

}

spec {

access\_modes = ["ReadWriteMany"]

resources {

requests = {

storage = "5Gi"

}

}

}

}

resource "kubernetes\_deployment" "wp-dep" {

metadata {

name = "wp-dep"

labels = {

app = "wordpress"

tier = "frontend"

}

}

spec {

replicas = 2

selector {

match\_labels = {

app = "wordpress"

tier = "frontend"

}

}

template {

metadata {

labels = {

app = "wordpress"

tier = "frontend"

}

}

spec {

volume {

name = "wordpress-persistent-storage"

persistent\_volume\_claim {

claim\_name = kubernetes\_persistent\_volume\_claim.pvc.metadata.0.name

}

}

container {

image = "wordpress"

name = "wordpress-container"

port {

container\_port = 80

}

volume\_mount {

name = "wordpress-persistent-storage"

mount\_path = "/var/www/html"

}

}

}

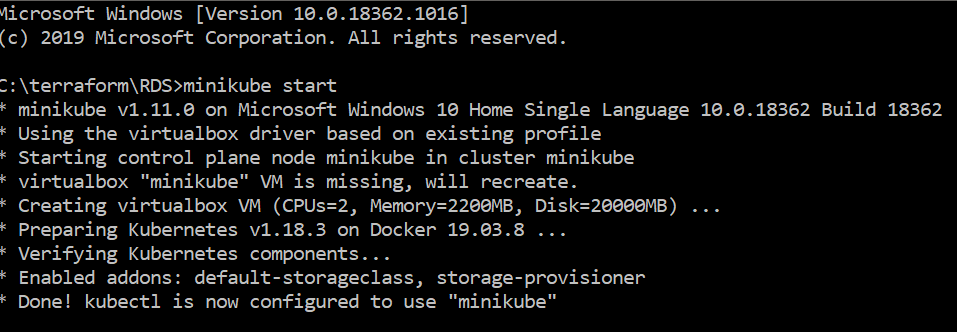
}

}

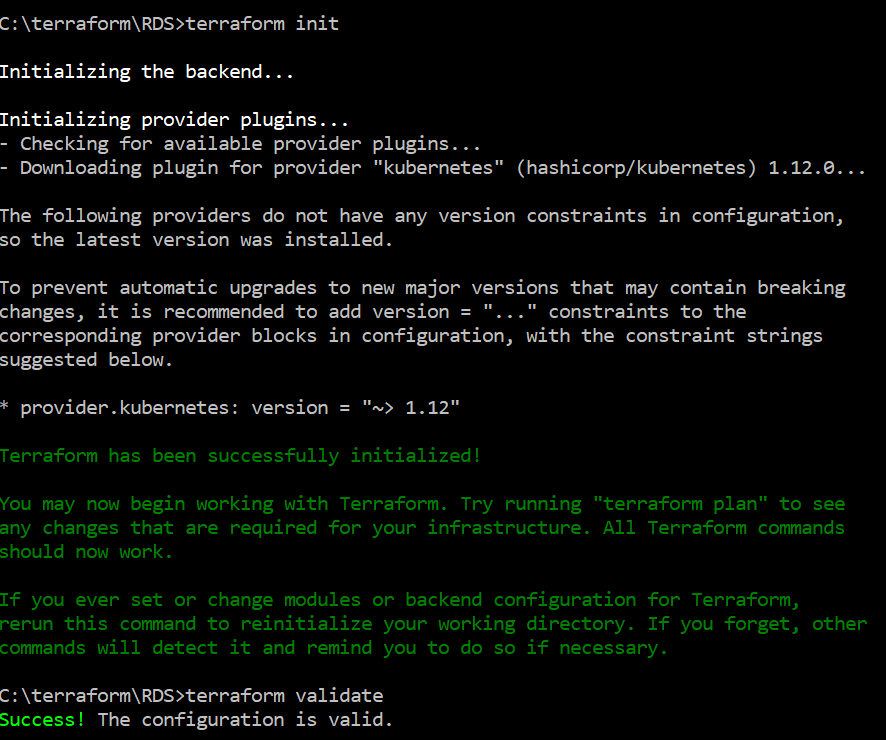
}

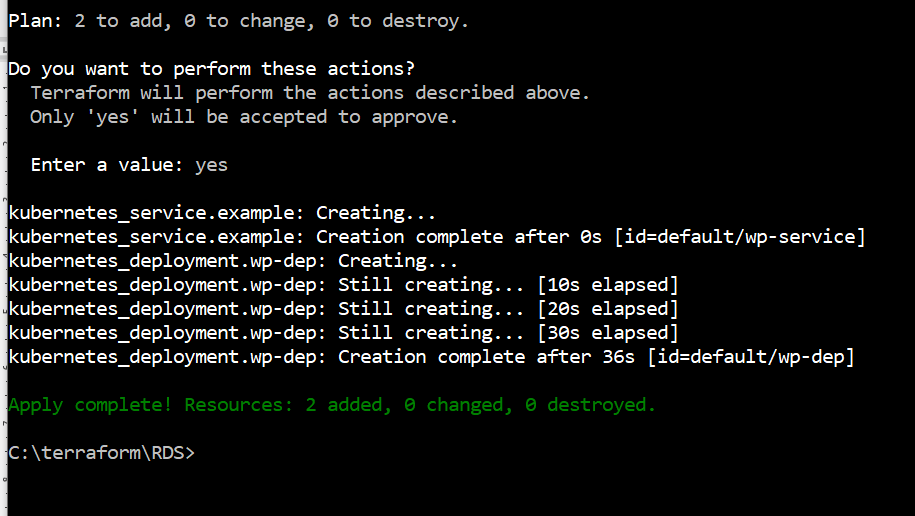
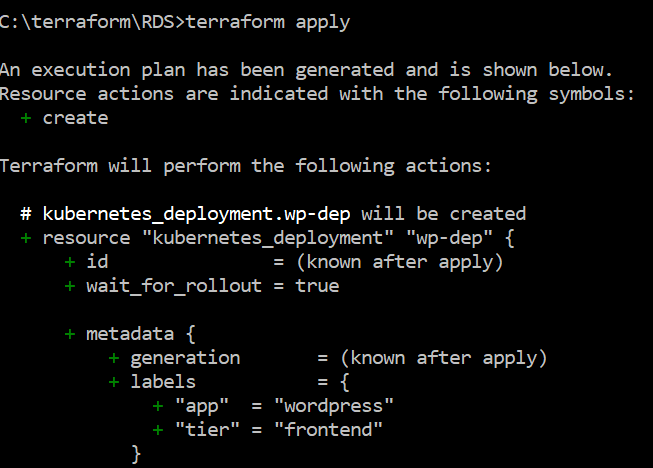
**The above file will deploy the wordpress application. so finally use terraform apply command to run the .tf file.**

**Start the minikube service using the command minikube start**

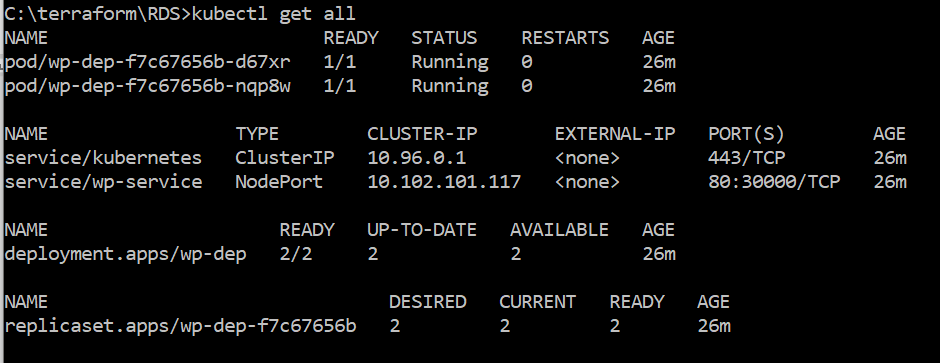


**Now runt the .tf file of the provider service i.e. Kubernetes**

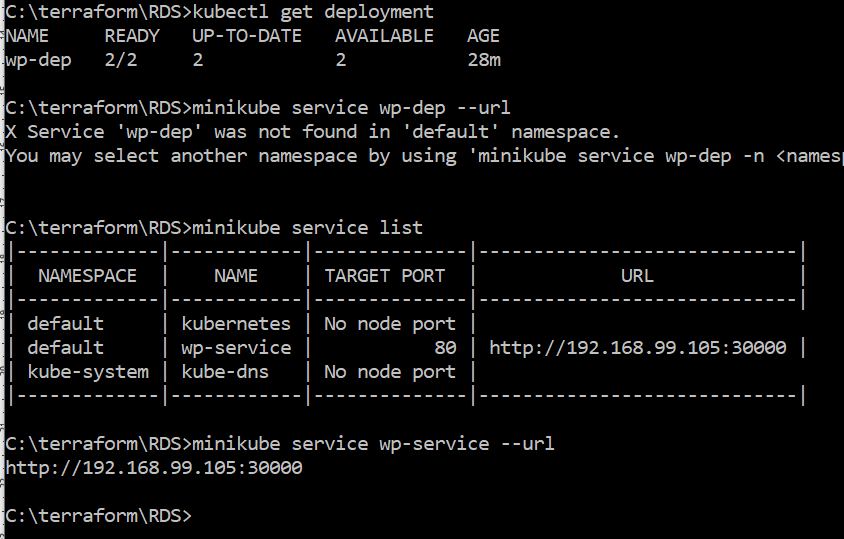




**We can see that terraform file is executed properly without any error now will check if the pods is running by kubectl get pods**



**We can check the deployment the minikube service url by the command “kubectl get deployment” for the wordpress service deployment and “minikube service list” to list all the running services on minikube and to get the url use the command “minikube service <name> --url.**



**Deploying RDS**

**We can see that the conatiner is runnig properly and exposed to port 30000 so if any user from outside world want to access the website he can use my minikube ip:30000 to access the website . Now we will deploy the RDS database for our wordpress application in aws and collect all the information like database name password and username to login to wordpress.**

provider "aws" {

region = "ap-south-1"

profile = "prisha"

}

resource "aws\_db\_instance" "default" {

allocated\_storage = 20

storage\_type = "gp2"

engine = "mysql"

engine\_version = "5.7"

instance\_class = "db.t2.micro"

name = "RDS"

username = "payal"

password = "payal123"

parameter\_group\_name = "default.mysql5.7"

publically accessible = "yes"

skip\_final\_snapshot = true

tags = {

Name = "pika"

}

}

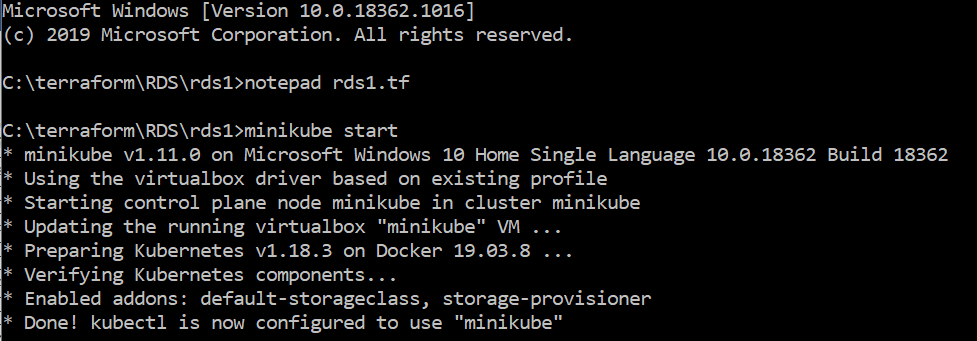
output "dns" {

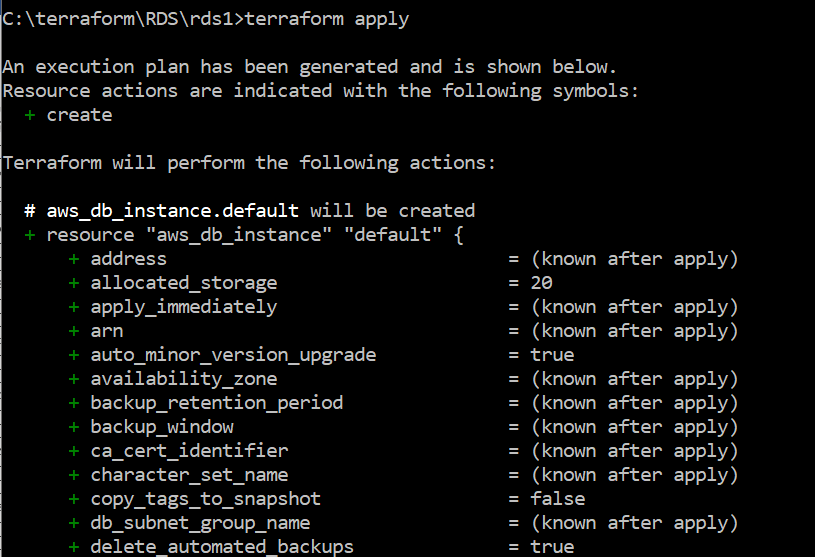
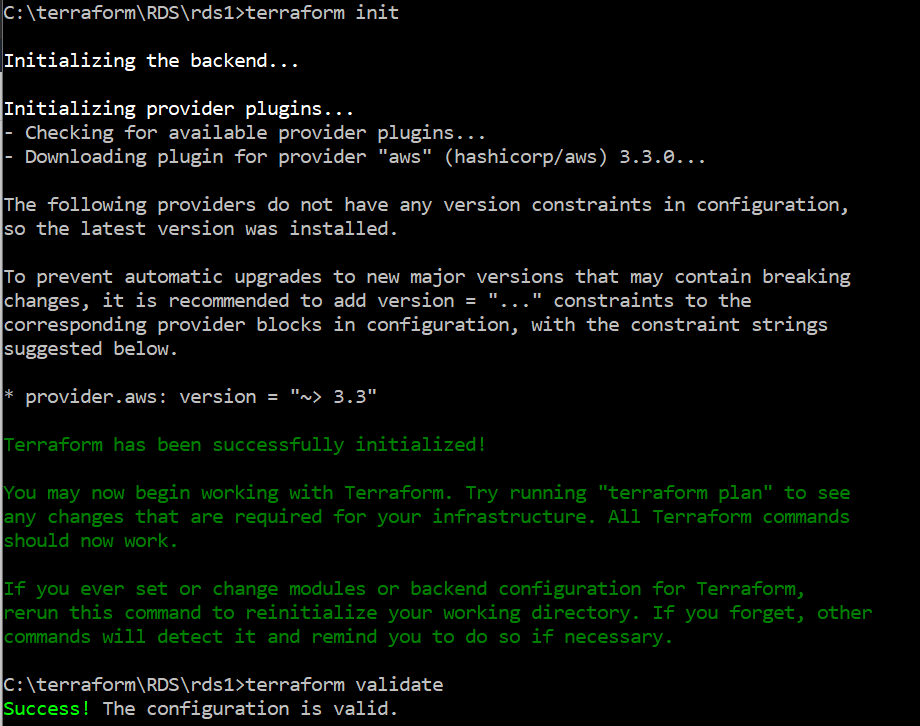
value = aws\_db\_instance.default.address

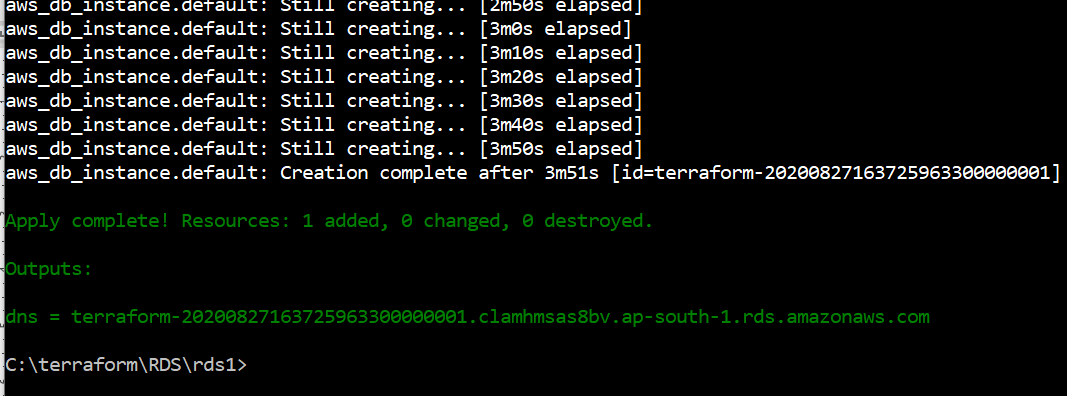
}

**The above terraform code for RDS will create RDS in AWS I have provided the database name username and password to login by wordpress.**

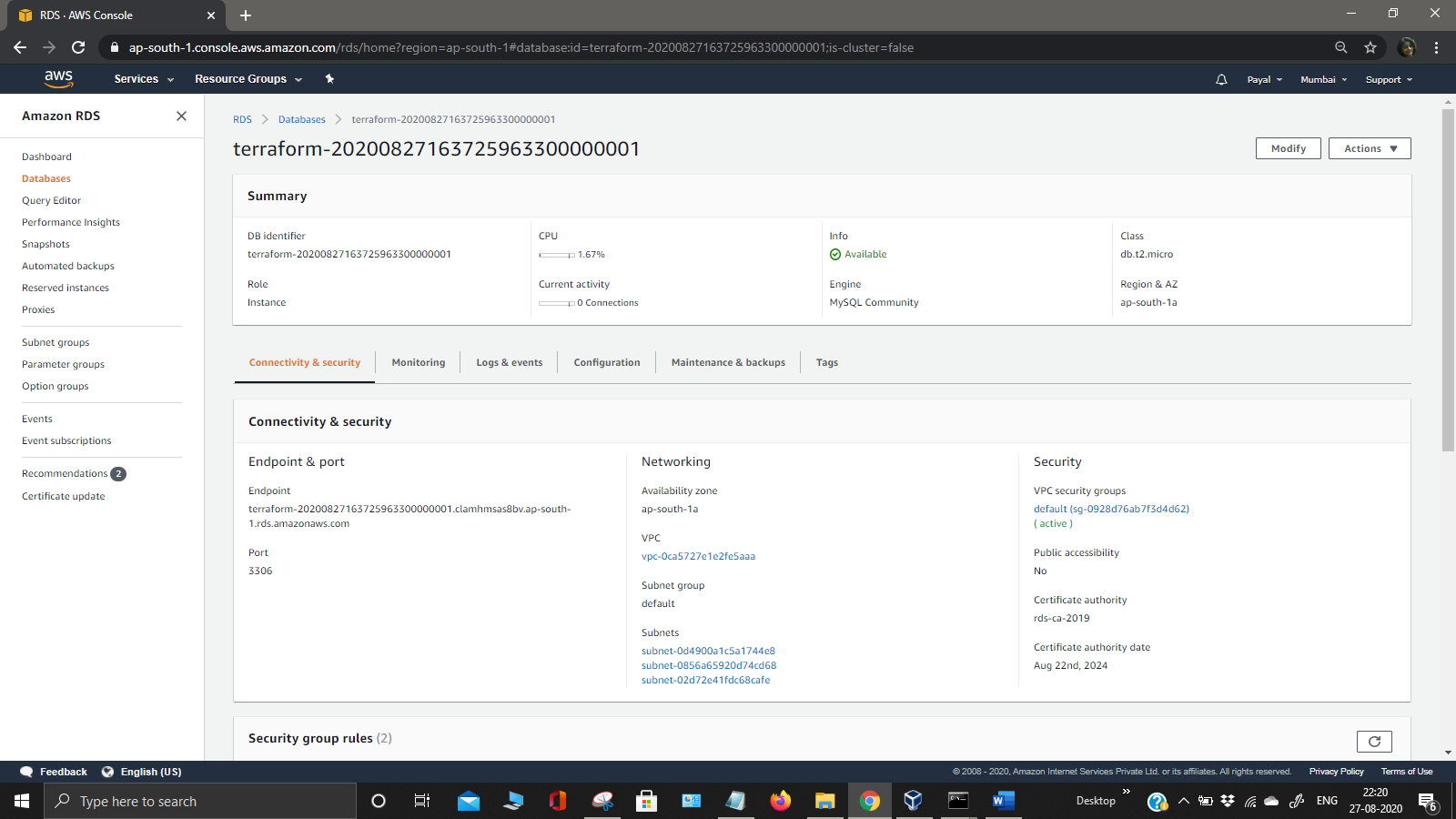
**Starting the minikube service.**







**Now we can see that the terraform code has executed properly so we can check the RDS data base in AWS**



**Connecting RDS to Word Press**

**Finally we can connect wordpress application to our RDS database to providing the username password databse name and database host. Now go to google crome and type the minikube ip:30000 to launch the wordpress application**

