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: https://github.com/payal024/TASK-6-HCC

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

```
Microsoft Windows [Version 10.0.18362.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\terraform\RDS>minikube start
* minikube v1.11.0 on Microsoft Windows 10 Home Single Language 10.0.18362 Build 18362
* Using the virtualbox driver based on existing profile
* Starting control plane node minikube in cluster minikube
* virtualbox "minikube" VM is missing, will recreate.
* Creating virtualbox VM (CPUs=2, Memory=2200MB, Disk=20000MB) ...
* Preparing Kubernetes v1.18.3 on Docker 19.03.8 ...
* Verifying Kubernetes components...
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube"
```

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

```
C:\terraform\RDS>terraform init

Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "kubernetes" (hashicorp/kubernetes) 1.12.0...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.kubernetes: version = "~> 1.12"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\terraform\RDS>terraform validate
Success! The configuration is valid.
```

```
C:\terraform\RDS>terraform apply

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # kubernetes_deployment.wp-dep will be created
  + resource "kubernetes_deployment" "wp-dep" {
      + id               = (known after apply)
      + wait_for_rollout = true

      + metadata {
          + generation       = (known after apply)
          + labels           = {
              + "app"  = "wordpress"
              + "tier" = "frontend"
            }
```

```
Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

kubernetes_service.example: Creating...
kubernetes_service.example: Creation complete after 0s [id=default/wp-service]
kubernetes_deployment.wp-dep: Creating...
kubernetes_deployment.wp-dep: Still creating... [10s elapsed]
kubernetes_deployment.wp-dep: Still creating... [20s elapsed]
kubernetes_deployment.wp-dep: Still creating... [30s elapsed]
kubernetes_deployment.wp-dep: Creation complete after 36s [id=default/wp-dep]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

C:\terraform\RDS>
```

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

```
C:\terraform\RDS>kubectl get all
NAME                          READY   STATUS    RESTARTS   AGE
pod/wp-dep-f7c67656b-d67xr    1/1     Running   0          26m
pod/wp-dep-f7c67656b-nqp8w    1/1     Running   0          26m

NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
service/kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP        26m
service/wp-service   NodePort    10.102.101.117   <none>        80:30000/TCP   26m

NAME                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/wp-dep   2/2     2            2           26m

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/wp-dep-f7c67656b    2         2         2       26m
```

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.

```
C:\terraform\RDS>kubectl get deployment
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
wp-dep   2/2     2            2           28m

C:\terraform\RDS>minikube service wp-dep --url
X Service 'wp-dep' was not found in 'default' namespace.
You may select another namespace by using 'minikube service wp-dep -n <names


C:\terraform\RDS>minikube service list
|-------------|------------|--------------|----------------------------|
|  NAMESPACE  |    NAME    | TARGET PORT  |            URL             |
|-------------|------------|--------------|----------------------------|
| default     | kubernetes | No node port |
| default     | wp-service |           80 | http://192.168.99.105:30000 |
| kube-system | kube-dns   | No node port |
|-------------|------------|--------------|----------------------------|

C:\terraform\RDS>minikube service wp-service --url
http://192.168.99.105:30000

C:\terraform\RDS>
```

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.

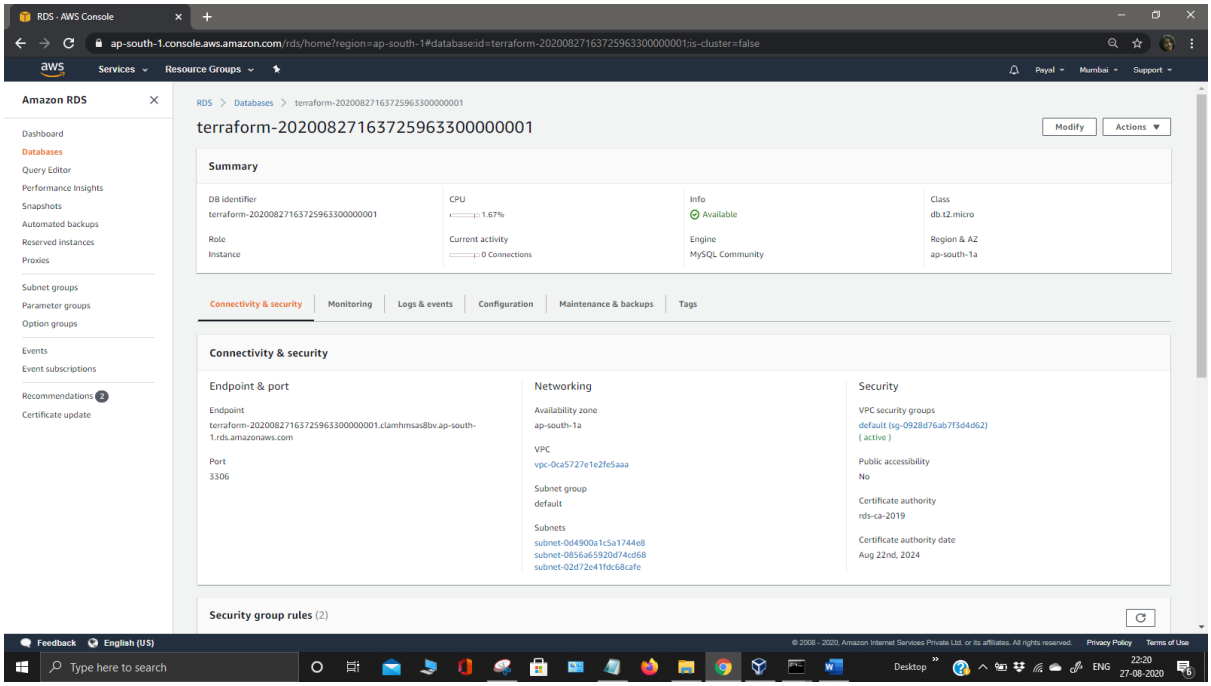
```
Microsoft Windows [Version 10.0.18362.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\terraform\RDS\rds1>notepad rds1.tf

C:\terraform\RDS\rds1>minikube start
* minikube v1.11.0 on Microsoft Windows 10 Home Single Language 10.0.18362 Build 18362
* Using the virtualbox driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Updating the running virtualbox "minikube" VM ...
* Preparing Kubernetes v1.18.3 on Docker 19.03.8 ...
* Verifying Kubernetes components...
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube"
```

```
C:\terraform\RDS\rds1>terraform init

Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "aws" (hashicorp/aws) 3.3.0...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.aws: version = "~> 3.3"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\terraform\RDS\rds1>terraform validate
Success! The configuration is valid.
```

```
C:\terraform\RDS\rds1>terraform apply

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_db_instance.default will be created
  + resource "aws_db_instance" "default" {
      + address                               = (known after apply)
      + allocated_storage                     = 20
      + apply_immediately                     = (known after apply)
      + arn                                   = (known after apply)
      + auto_minor_version_upgrade            = true
      + availability_zone                     = (known after apply)
      + backup_retention_period               = (known after apply)
      + backup_window                         = (known after apply)
      + ca_cert_identifier                    = (known after apply)
      + character_set_name                    = (known after apply)
      + copy_tags_to_snapshot                 = false
      + db_subnet_group_name                  = (known after apply)
      + delete_automated_backups              = true
```

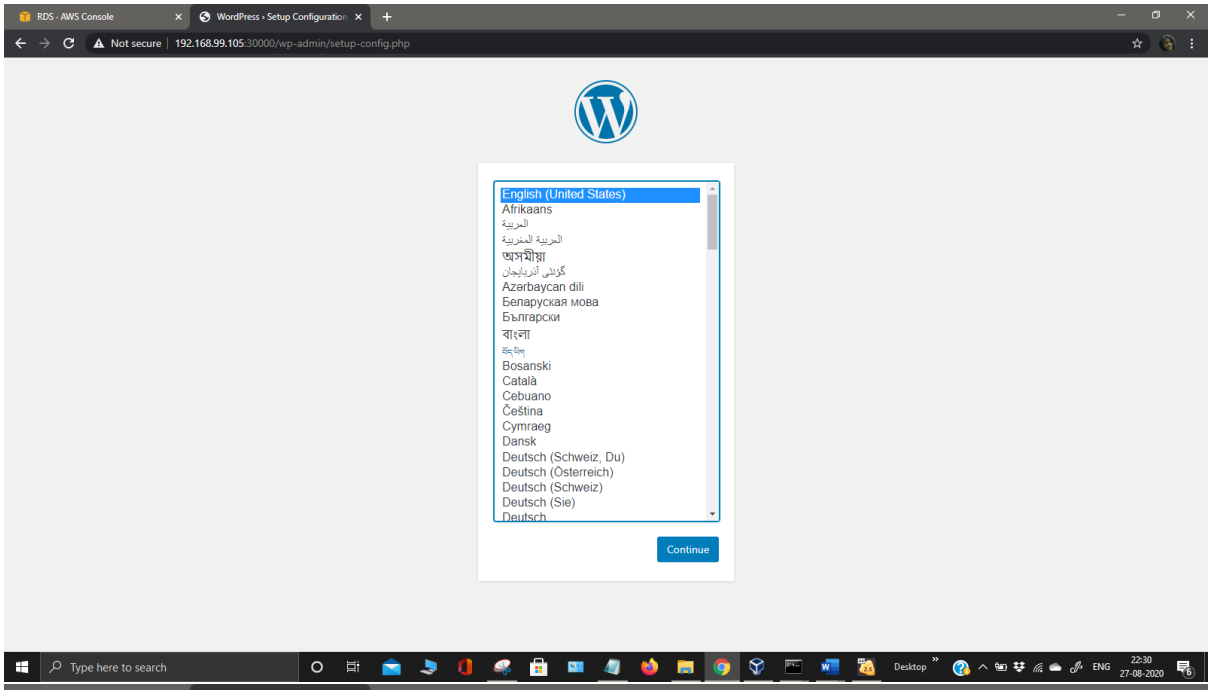
```
aws_db_instance.default: Still creating... [2m50s elapsed]
aws_db_instance.default: Still creating... [3m0s elapsed]
aws_db_instance.default: Still creating... [3m10s elapsed]
aws_db_instance.default: Still creating... [3m20s elapsed]
aws_db_instance.default: Still creating... [3m30s elapsed]
aws_db_instance.default: Still creating... [3m40s elapsed]
aws_db_instance.default: Still creating... [3m50s elapsed]
aws_db_instance.default: Creation complete after 3m51s [id=terraform-20200827163725963300000001]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

dns = terraform-20200827163725963300000001.clamhmsas8bv.ap-south-1.rds.amazonaws.com

C:\terraform\RDS\rds1>
```

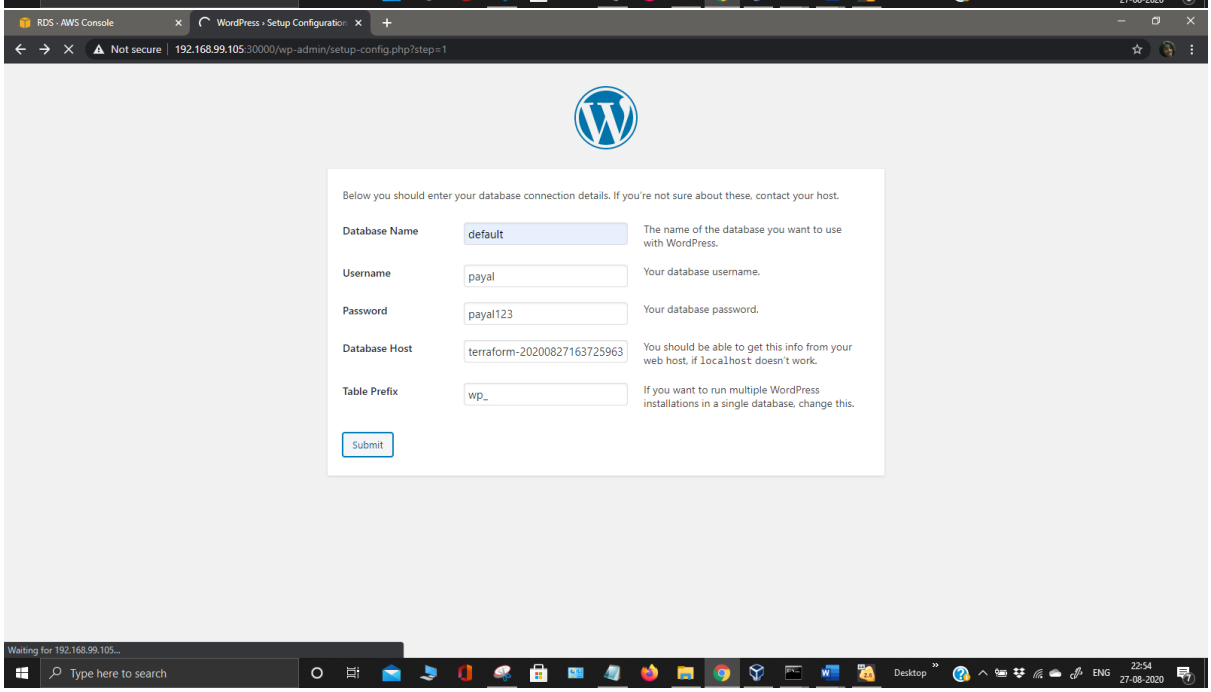
Now we can see that the terraform code has executed properly so we can check the RDS data base in AWS

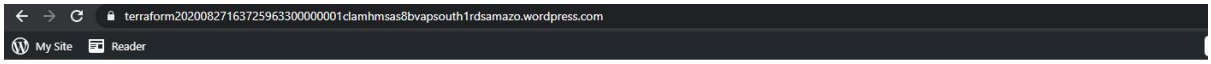


Connecting RDS to Word Press

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


RDS - AWS Console
WordPress › Setup Configuration
Not secure | 192.168.99.105:30000/wp-admin/setup-config.php
English (United States)
Afrikaans
العربية
العربية المغربية
অসমীয়া
گۆڕی آزربایجان
Azərbaycan dili
Беларуская мова
Български
বাংলা
Bosanski
Català
Cebuano
Čeština
Cymraeg
Dansk
Deutsch (Schweiz, Du)
Deutsch (Österreich)
Deutsch (Schweiz)
Deutsch (Sie)
Continue
Type here to search
Desktop
ENG
22:30
27-08-2020


RDS - AWS Console
WordPress › Setup Configuration
Not secure | 192.168.99.105:30000/wp-admin/setup-config.php?step=1
Below you should enter your database connection details. If you're not sure about these, contact your host.
Database Name
default
The name of the database you want to use with WordPress.
Username
payal
Your database username.
Password
payal123
Your database password.
Database Host
terraform-20200827163725963
You should be able to get this info from your web host, if localhost doesn't work.
Table Prefix
wp_
If you want to run multiple WordPress installations in a single database, change this.
Submit
Waiting for 192.168.99.105...
Type here to search
Desktop
ENG
22:54
27-08-2020


terraform20200827163725963300000001clamhmsas8bvapsouth1rdsamazo.wordpress.com
My Site
Reader

Payal's Site