

# Task-6 Deploy the Wordpress application on Kubernetes and AWS using terraform .

**AWS 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.

**KUBERNETES :-**

Kubernetes is open source software that allows you to deploy and manage containerized applications at scale.Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem.*Containers are a good way to bundle and run your applications. In a production environment, you need to manage the containers that run the applications and ensure that there is no downtime. For example, if a container goes down, another container needs to start.*

**TASK:-**

Performing 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.

**Let’s start:**

**To access our AWS account :-**

provider “aws” {

region = “us-east-1”

}

**Create the Database-instance with required options..**

resource “aws\_db\_instance” “mysql” {

allocated\_storage = 20

engine = “mysql”

instance\_class = “db.t2.micro”

storage\_type = “gp2”

engine\_version = “5.7.30”

username = “root”

password = “mysql123”

publicly\_accessible = “true”

port = “3306”

vpc\_security\_group\_ids = [“sg-c0b4e9e2”,]

multi\_az = “false”

name = “wordpressdb”

identifier = “wordpress-rds”

}

output “The\_rds\_details\_are” {

description = “ The connection endpint is”

value = “module.aws\_db\_instance.mysql.endpoint”

}

**Now, let's deploy the wordpress container in top of Kubernetes..**

**For this,we first register our minikube as provider for terraform..**

provider “kubernetes” {

config\_context\_cluster = “minikube”

}

resource “null\_resource” “minikubestart” {

provisioner “local-exec” {

command = “minikube start”

}

}

**Now,create the deployment of wordpress as follows..using the terraform code.**.

resource “kubernetes\_deployment” “wordpress” {

metadata {

name = “wordpress-deployment”

labels = {

name = “wordpress”

}

}

spec {

replicas = 2

selector {

match\_labels = {

name = “wordpress”

}

}

template {

metadata {

labels = {

name = “wordpress”

}

}

spec {

container {

image = “wordpress”

name = “wordpress”

}

}

}

}

}

#########################

resource “kubernetes\_service” “wordpress-service” {

metadata {

name = “wp-service”

}

spec {

selector = {

name = “${kubernetes\_deployment.wordpress.metadata.0.labels.name}”

}

port {

port = 80

target\_port = 80

protocol = “TCP”

}

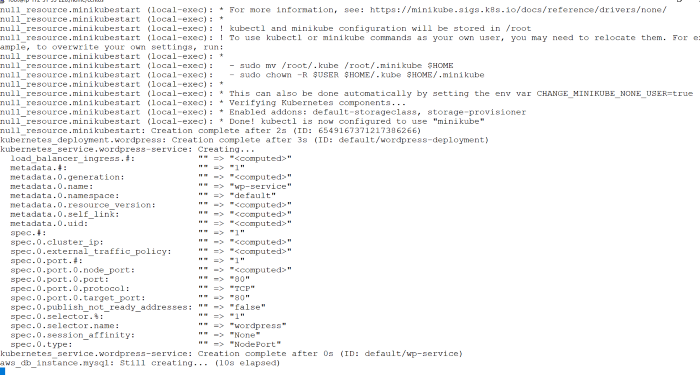
type = “NodePort”

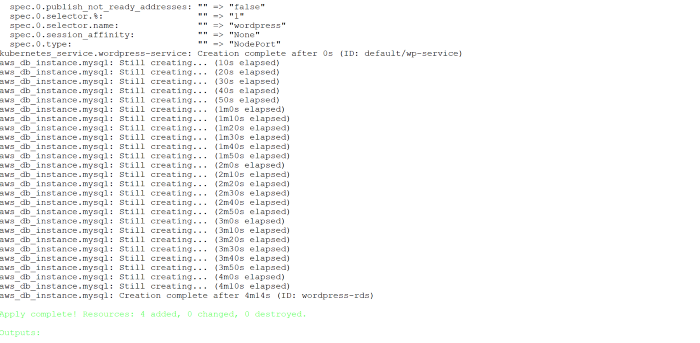
}

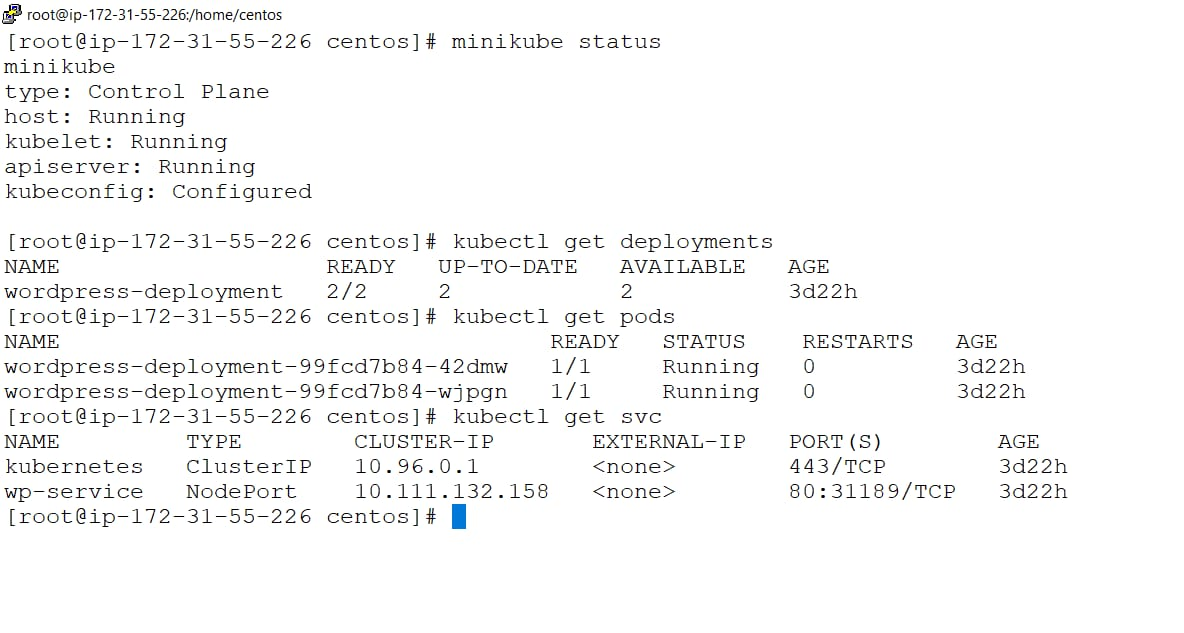
depends\_on = [“kubernetes\_deployment.wordpress”]

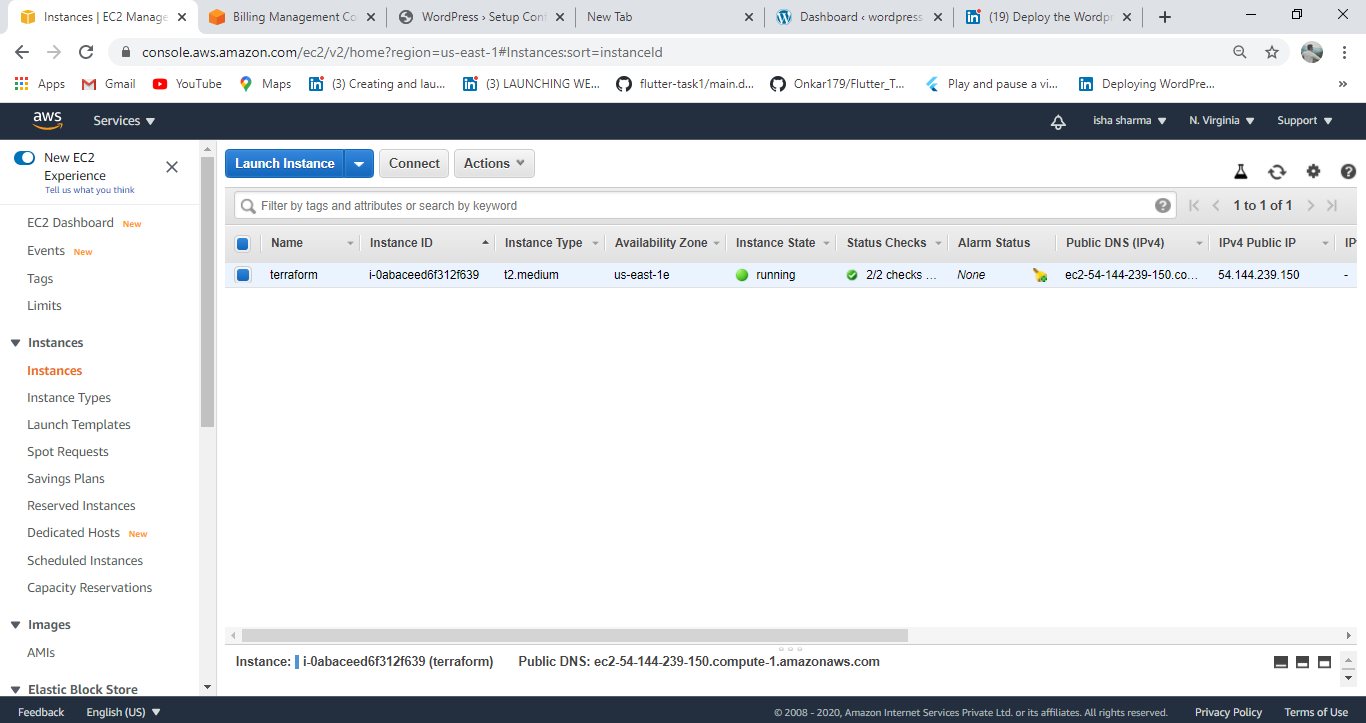
}

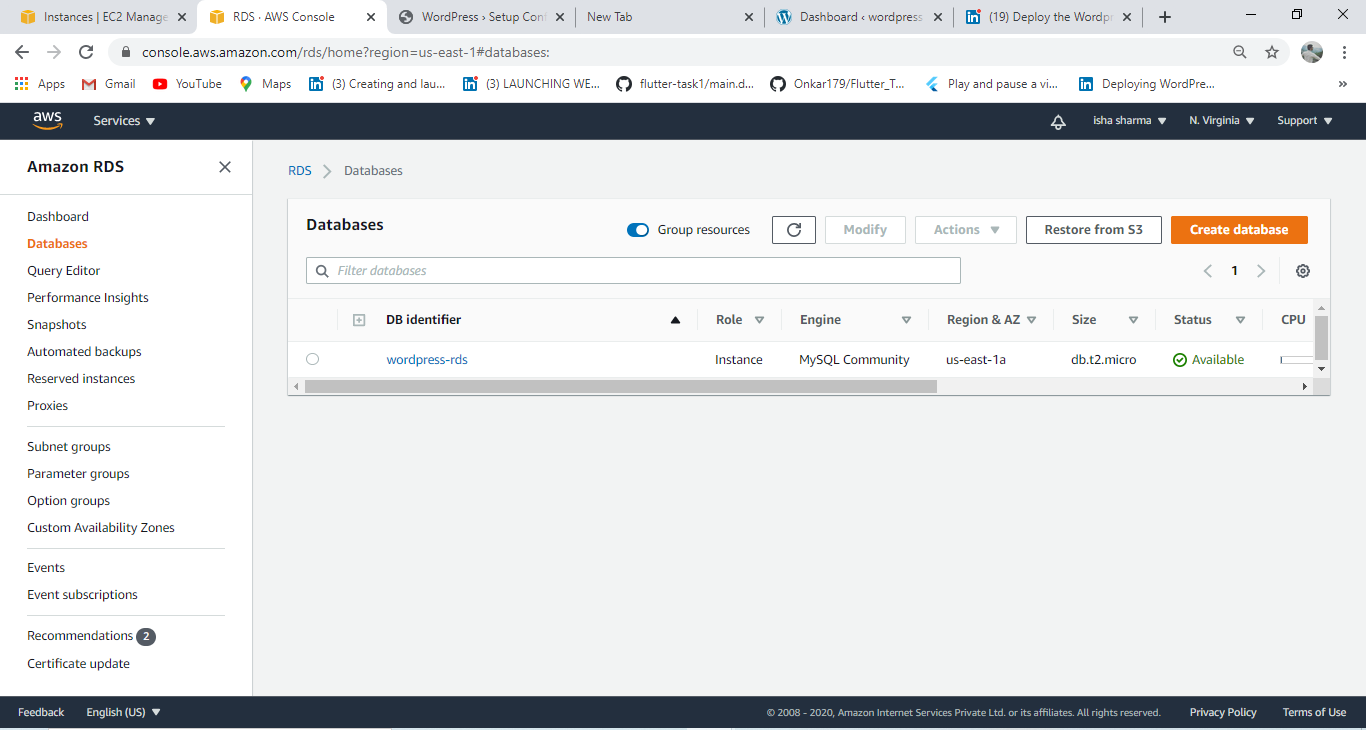
**terraform apply:**

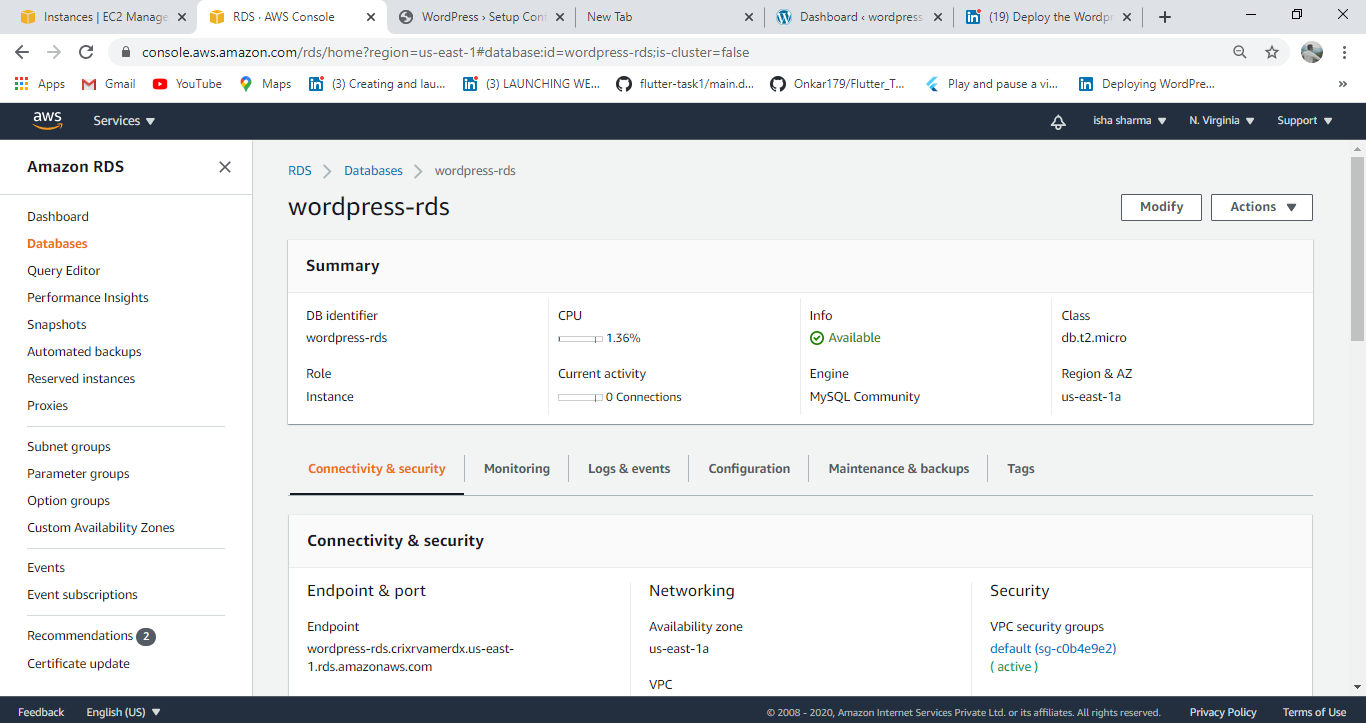
****

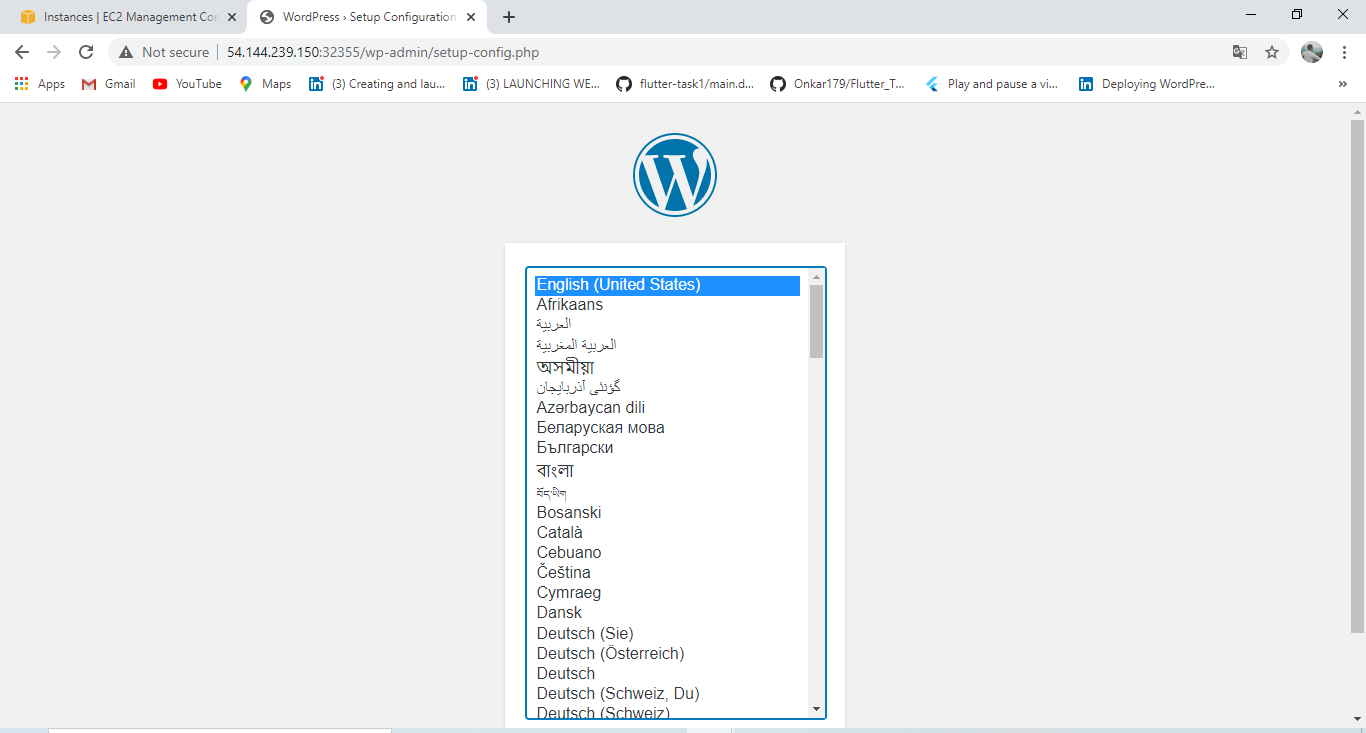
****

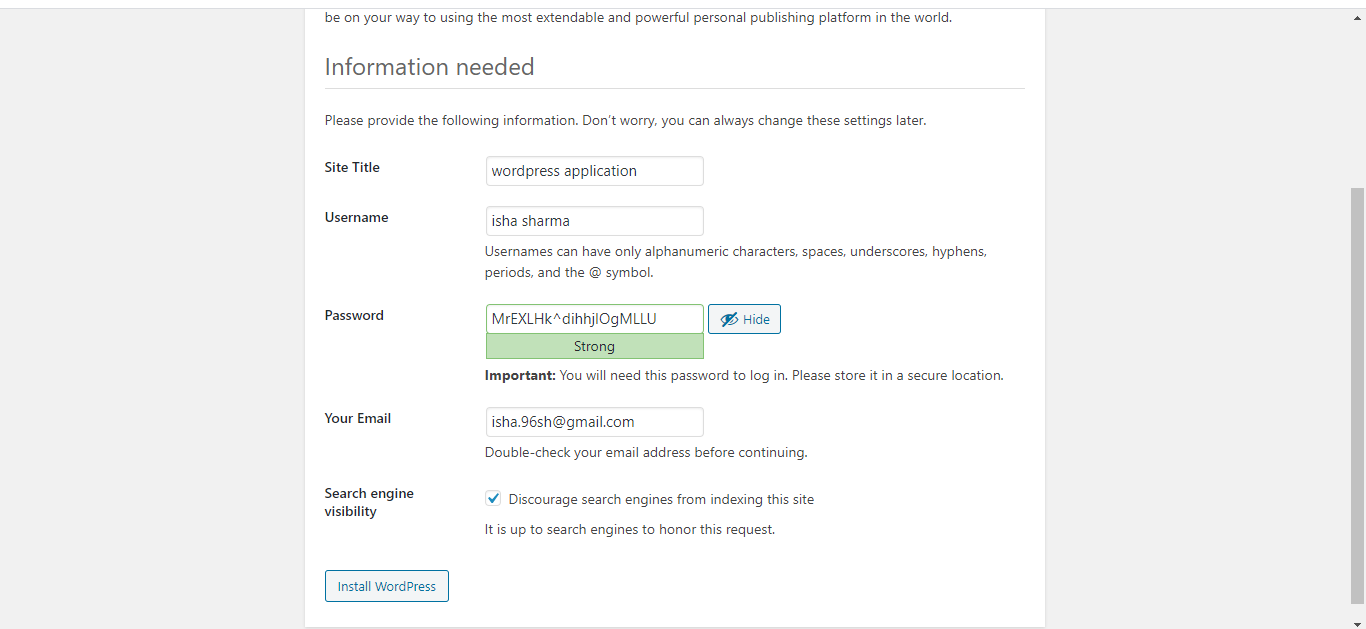
****

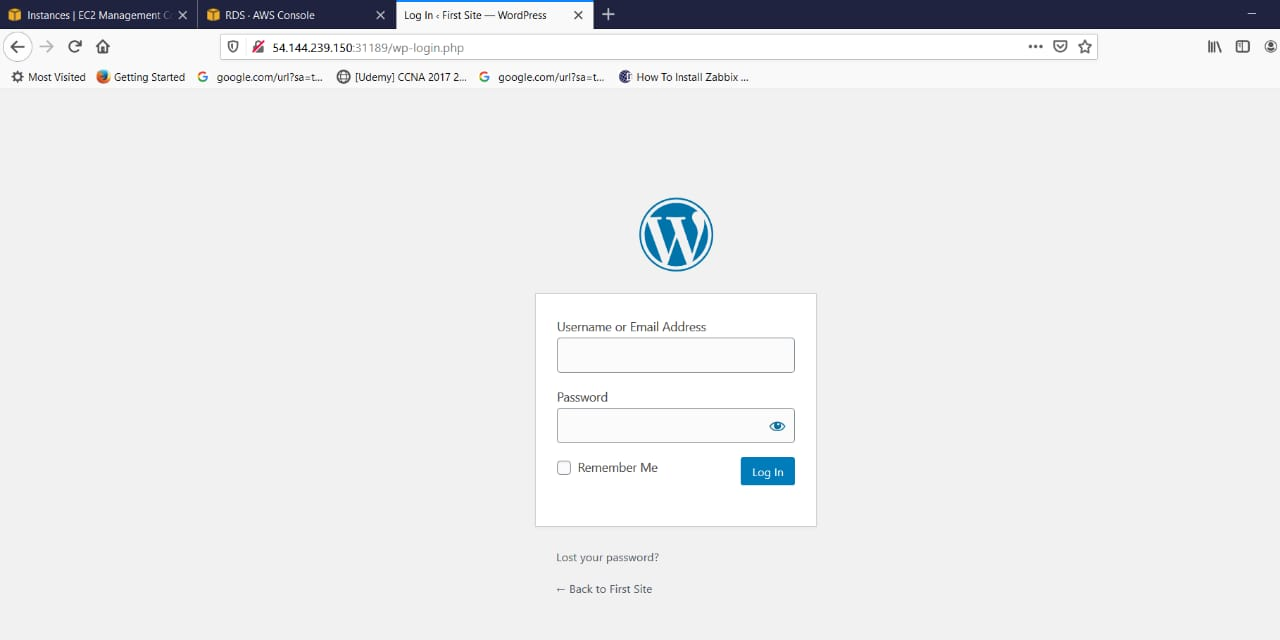
****

****

****

****

****

****

**Thank You...**