# **Fully secured and automated setup of WordPress using AWS and Terraform**

**Task Overview:**

1) Write a Infrastructure as code using terraform, which automatically create a VPC.

2) In that VPC we have to create 2 subnets:

    a)  public  subnet [ Accessible for Public World! ]

    b)  private subnet [ Restricted for Public World! ]

3) Create a public facing internet gateway for connect our VPC/Network to the internet world and attach this gateway to our VPC.

4) Create a routing table for Internet gateway so that instance can connect to outside world, update and associate it with public subnet.

5) Create two security group for WordPress and MySql instance.

6) Launch an ec2 instance which has Wordpress setup already having the security group allowing port 80 so that our client can connect to our wordpress site also attach the key to instance for further login into it.

7) Launch an ec2 instance which has MySQL setup already with security group allowing  port 3306 in private subnet so that our wordpress vm can connect with the same and also attach the key with the same.

### **Prerequisites for Task:**

1. Install AWS CLI
2. Install Terraform
3. Basic knowledge of VPC, Route Tables and Internet gateway

### **Commands used in task:**

1. Terraform init
2. Terraform validate
3. Terraform apply -auto-approve
4. Terraform destroy -auto-approve

**Step 1** **For using AWS as a provide in terraform, we need to login first...**

**provider "aws"{**

**region = "ap-south-1"**

**profile = “Dkchanuhan”**

**}**

**Step 2 For creating the VPC...**

**resource "aws\_vpc" "dkcvpc"{**

**cidr\_block = "10.5.0.0/16"**

**enable\_dns\_hostnames = true**

**tags = {**

**Name = "main"**

**}**

**}**

**Output**

### **Step 3 In that VPC we have to create 2 subnets:**

### 

**resource "aws\_subnet" "dkcsubnet" {**

**vpc\_id = "${aws\_vpc.dkcvpc.id}"**

**availability\_zone = "ap-south-1a"**

**cidr\_block = "10.5.1.0/24"**

**map\_public\_ip\_on\_launch = true**

**tags = {**

**Name = "main-1a"**

**}**

**}**

**resource "aws\_subnet" “dkcsubnet" {**

**vpc\_id = "${aws\_vpc.dkcvpc.id}"**

**availability\_zone = "ap-south-1a"**

**cidr\_block = "10.5.1.0/24"**

**map\_public\_ip\_on\_launch = true**

**tags = {**

**Name = "main-1a"**

**}**

**}**

**Step 3. Create a public facing internet gateway for connect our VPC/Network to the internet world and attach this gateway to our VPC:**

**resource "aws\_internet\_gateway" "dkinternet" {**

**vpc\_id = "${aws\_vpc.dkcvpc.id}"**

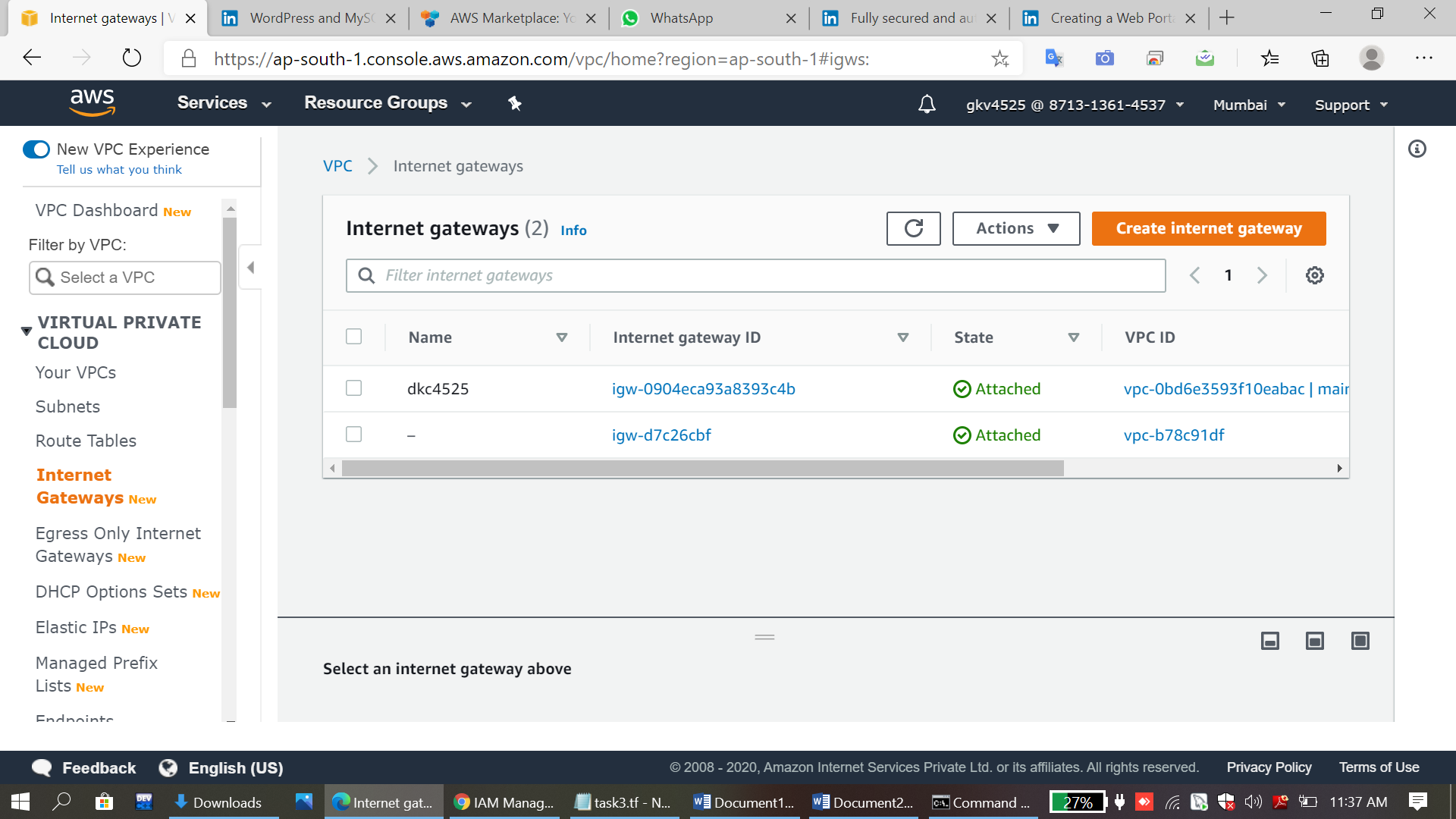
**tags = {**

**Name = "main-1a"**

**}**

**}**

**Output**



**Step 4 Create a routing table for Internet gateway so that instance can connect to outside world, update and associate it with public subnet:**

**resource "aws\_route\_table" "dktable" {**

**vpc\_id = "${aws\_vpc.dkcvpc.id}"**

**route {**

**cidr\_block = "0.0.0.0/0"**

**gateway\_id = "${aws\_internet\_gateway.dkinternet.id}"**

**}**

**tags = {**

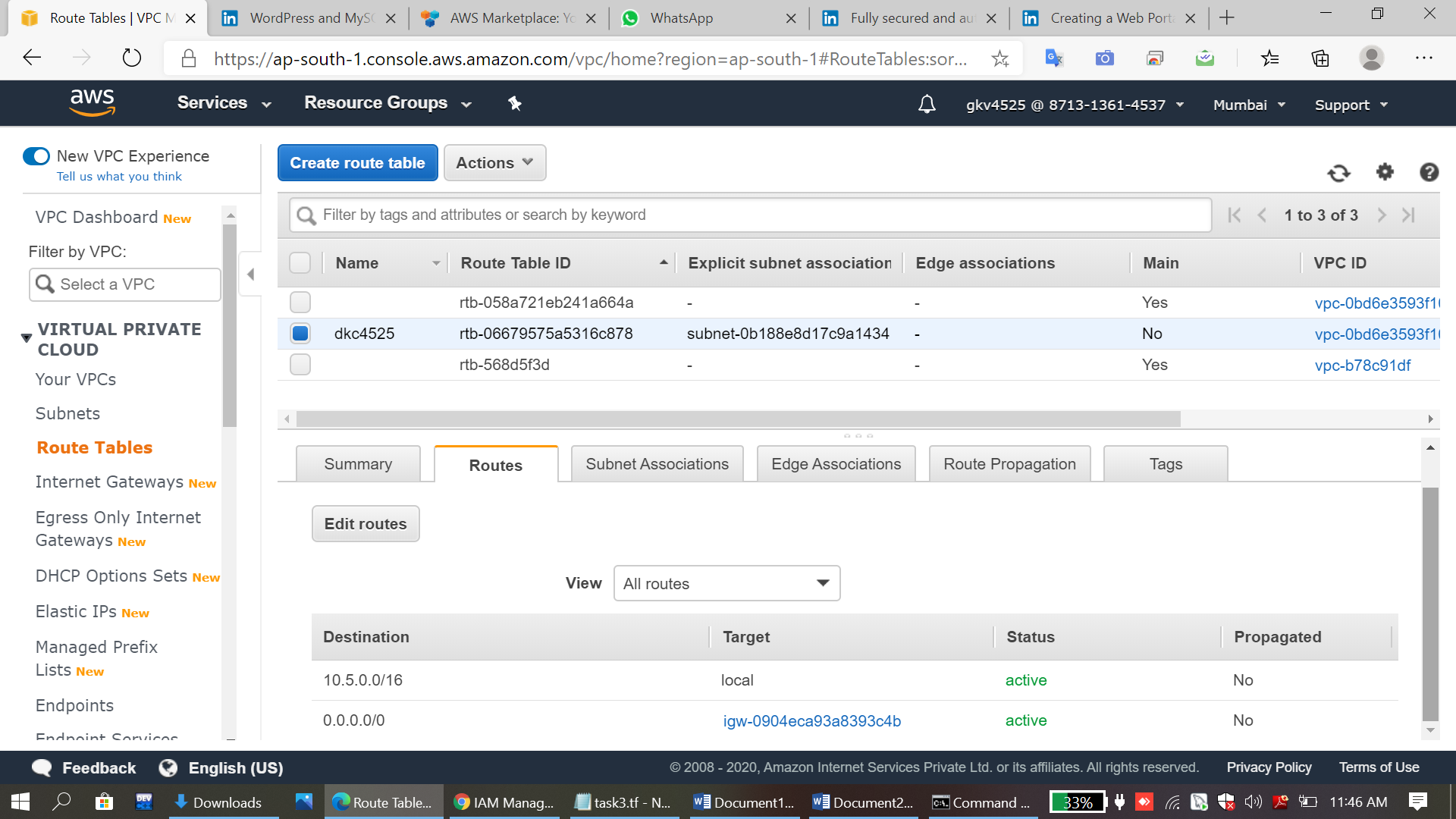
**Name = "dkc4525"**

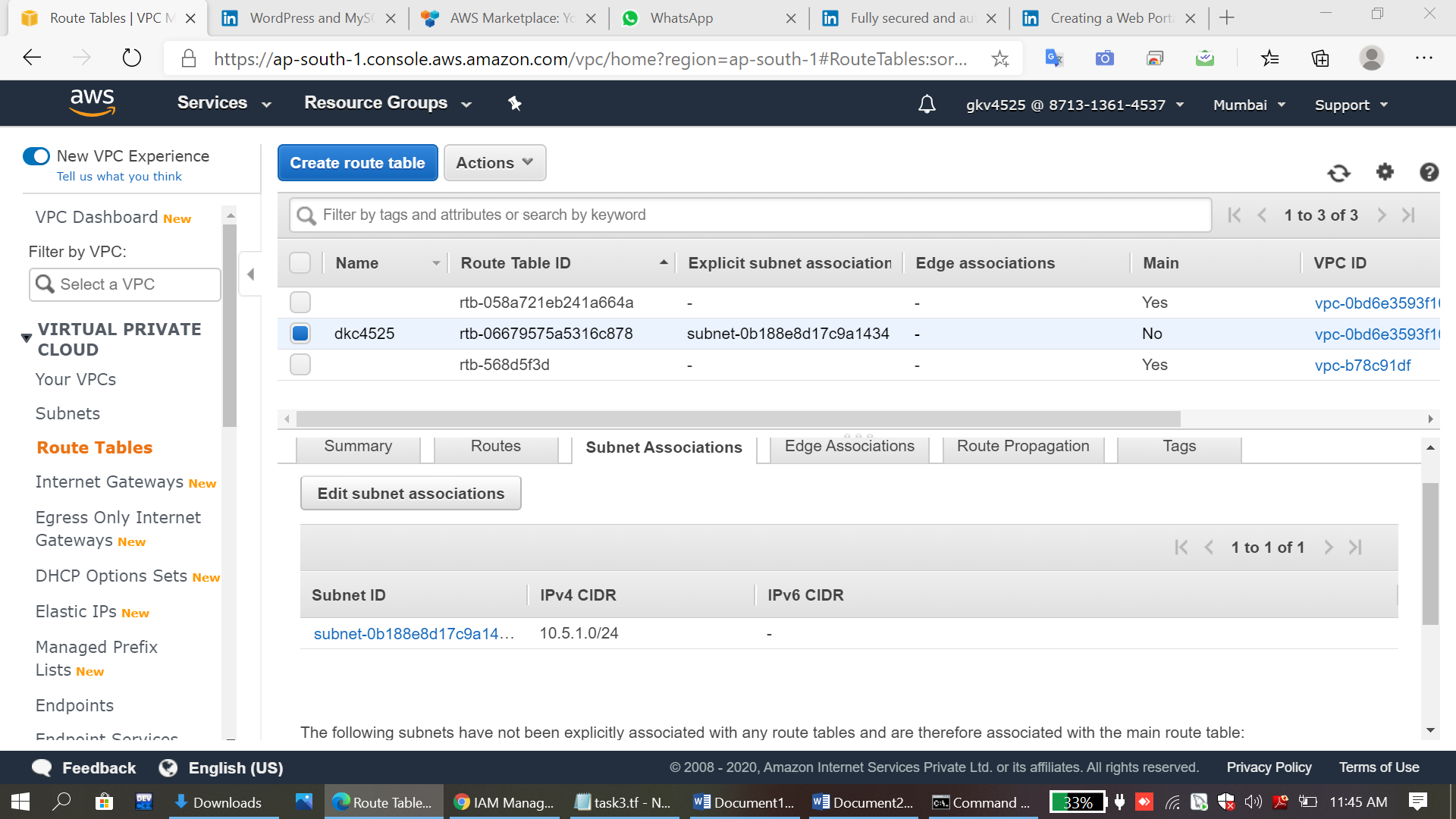
**}**

**}**

**resource "aws\_route\_table\_association" "a" {**

**subnet\_id = aws\_subnet.dkcsubnet.id route\_table\_id = aws\_route\_table.dktable.id**

**}**

**Step 5 Create two security group for WordPress and MySql instance**

**resource "aws\_security\_group" "allow\_http\_wordpress" {**

**name = "allow\_http\_wordpress"**

**description = "Allow HTTP inbound traffic"**

**vpc\_id = "${aws\_vpc.dkcvpc.id}"**

**ingress {**

**description = "Http from VPC"**

**from\_port = 80**

**to\_port = 80**

**protocol = "tcp"**

**cidr\_blocks = [ "0.0.0.0/0" ]**

**}**

**ingress {**

**description = "SSH from VPC"**

**from\_port = 22**

**to\_port = 22**

**protocol = "tcp"**

**cidr\_blocks = [ "0.0.0.0/0" ]**

**}**

**ingress {**

**description = "HTTPS"**

**from\_port = 443**

**to\_port = 443**

**protocol = "tcp"**

**cidr\_blocks = [ "0.0.0.0/0" ]**

**}**

**egress {**

**from\_port = 0**

**to\_port = 0**

**protocol = "-1"**

**cidr\_blocks = ["0.0.0.0/0"]**

**}**

**tags = {**

**Name = "sgroup"**

**}**

**}**

**resource "aws\_security\_group" "dkcmysql-sg" {**

**name = "for-dkcmysql"**

**description = "DKCMYSQL-setup"**

**vpc\_id = "${aws\_vpc.dkcvpc.id}"**

**ingress {**

**description = "DKCMYSQL from VPC"**

**from\_port = 3306**

**to\_port = 3306**

**protocol = "tcp"**

**cidr\_blocks = [ "0.0.0.0/0" ]**

**}**

**ingress {**

**description = "SSH from VPC"**

**from\_port = 22**

**to\_port = 22**

**protocol = "tcp"**

**cidr\_blocks = [ "0.0.0.0/0" ]**

**}**

**egress {**

**from\_port = 0**

**to\_port = 0**

**protocol = "-1"**

**cidr\_blocks = ["0.0.0.0/0"]**

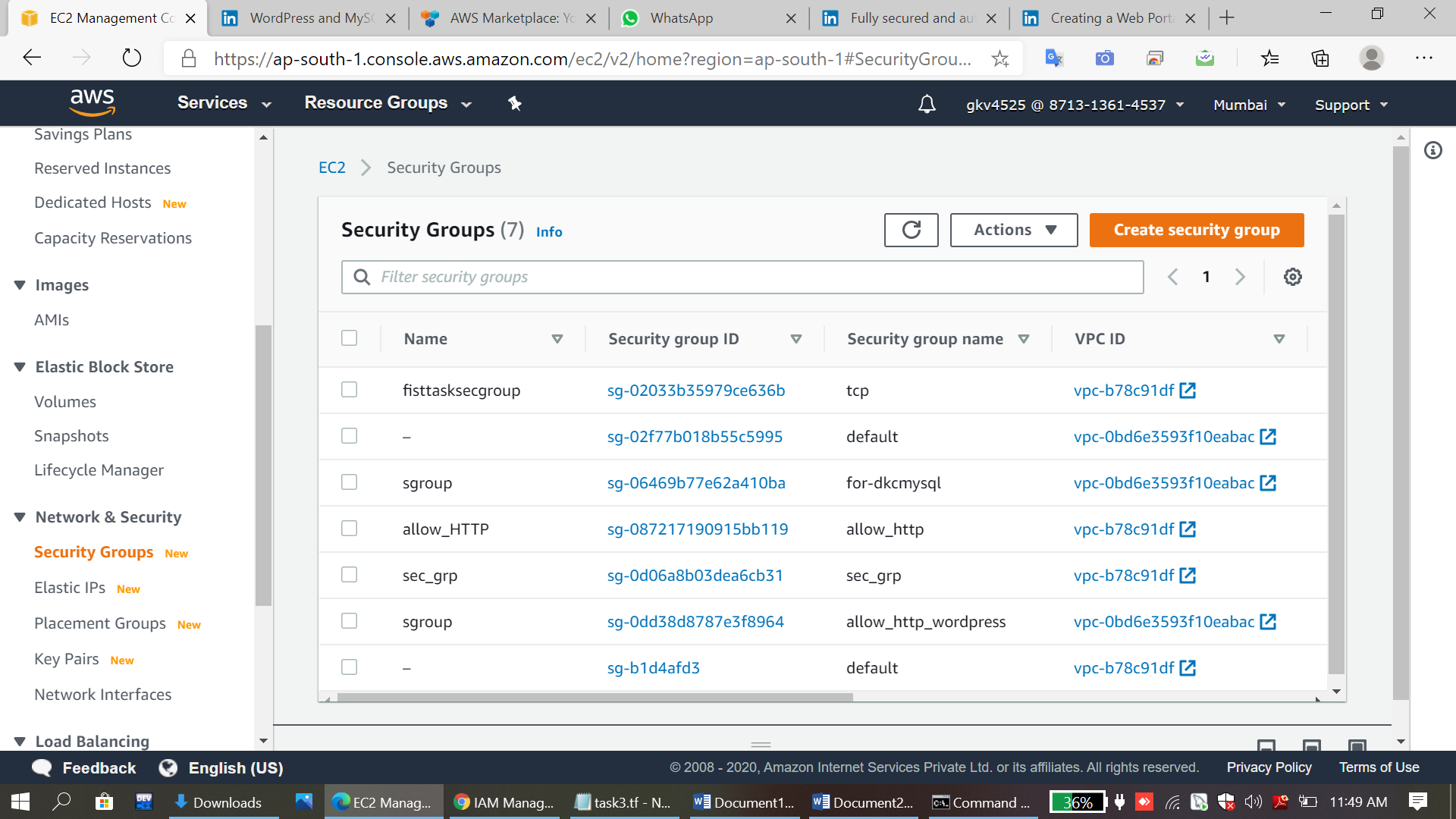
**}**

**tags = {**

**Name = "sgroup"**

**}**

**}**



**Step 6 Launch an ec2 instance which has Wordpress setup already having the security group allowing port 80 so that our client can connect to our wordpress site also attach the key to instance for further login into it:**

**resource "aws\_instance" "wordpress" {**

**ami = "ami-0979674e4a8c6ea0c"**

**instance\_type = "t2.micro"**

**key\_name = "task\_key"**

**availability\_zone = "ap-south-1a"**

**subnet\_id = "${aws\_subnet.dkcsubnet.id}"**

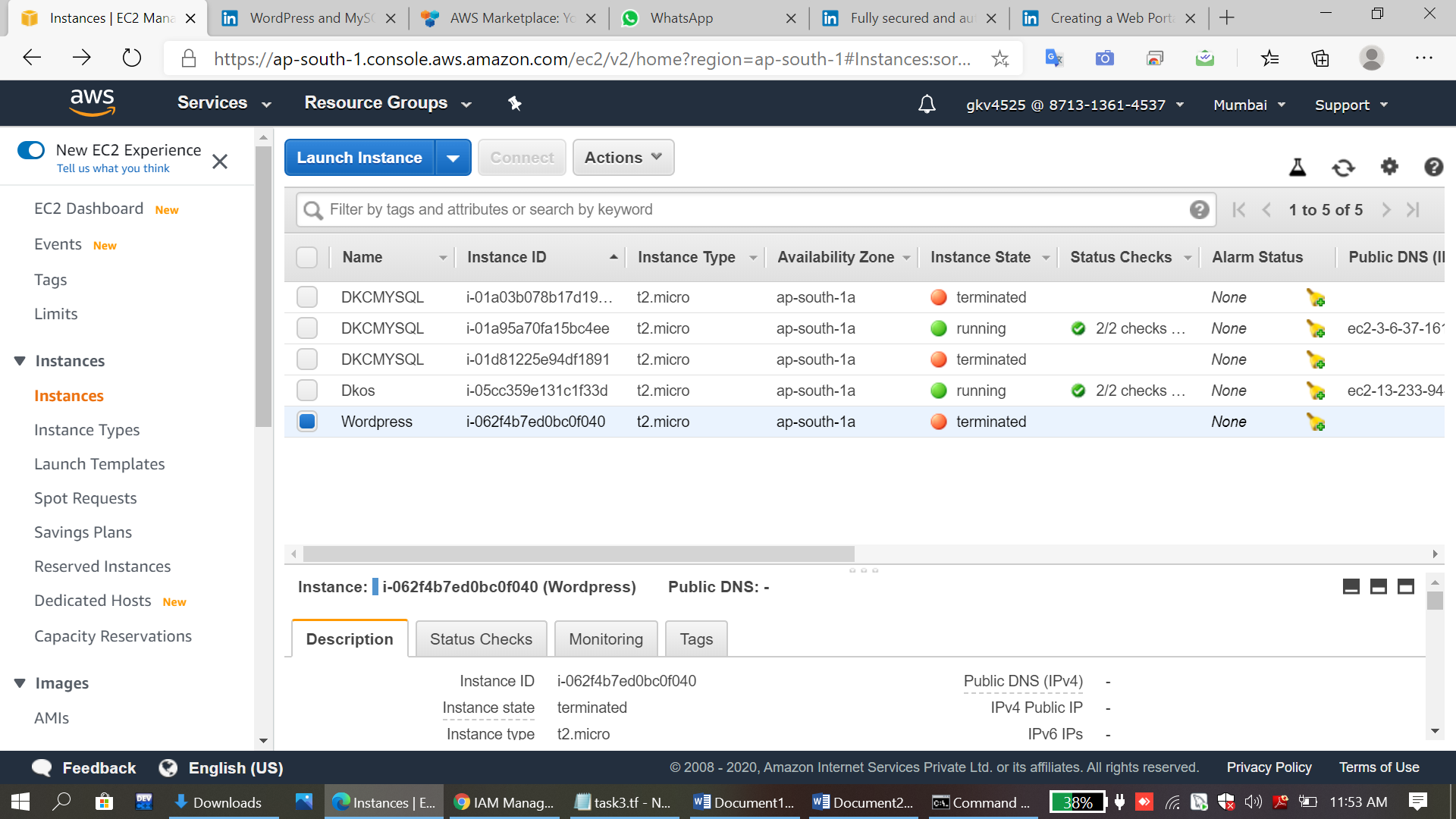
**security\_groups = [ "${aws\_security\_group.allow\_http\_wordpress.id}" ]**

**tags = {**

**Name = "Wordpress"**

**}**

**}**

 **Step 7  Launch an ec2 instance which has MySQL setup already with security group allowing  port 3306 in private subnet so that our wordpress vm can connect with the same and also attach the key with the same**

resource "aws\_instance" "dkcmysql" {

ami = "ami-76166b19"

instance\_type = "t2.micro"

key\_name = "task\_key"

availability\_zone = "ap-south-1a"

subnet\_id = "${aws\_subnet.dkcsubnet.id}"

security\_groups = [ "${aws\_security\_group.dkcmysql-sg.id}" ]

tags = {

Name = "DKCMYSQL"

}

}

