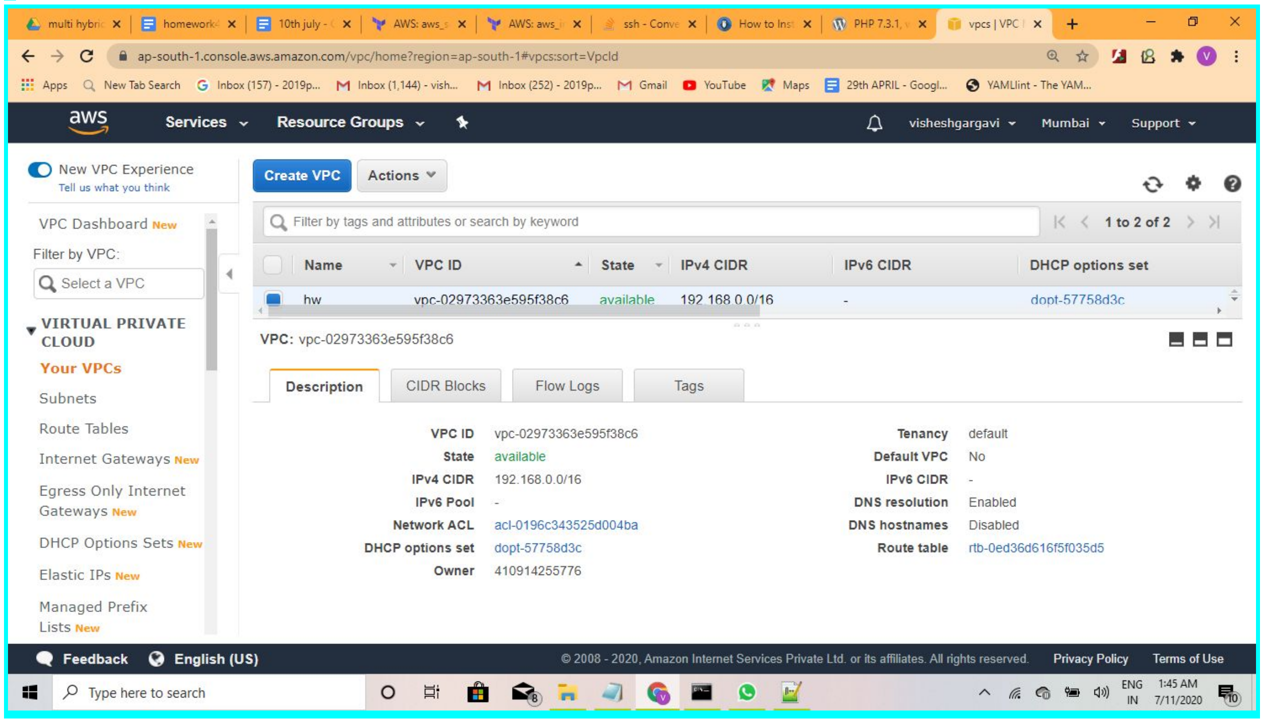
HOMEWORK 4:

> login in aws
> create a vpc

```
provider "aws" {
  region   = "ap-south-1"
  profile  = "myvishesh"
}

resource "aws_vpc" "hw" {
  cidr_block       = "192.168.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "hw"
  }
}
```



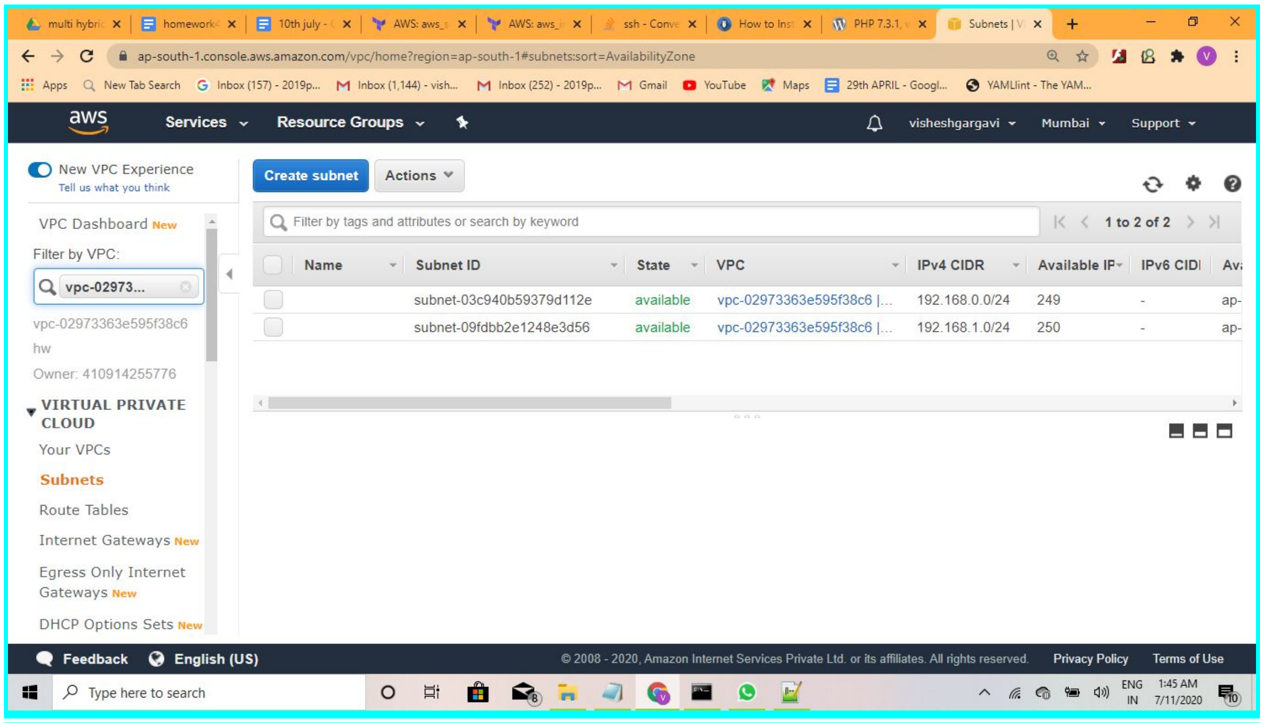
Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP options set
hw	vpc-02973363e595f38c6	available	192.168.0.0/16	-	dopt-57758d3c

VPC: vpc-02973363e595f38c6

VPC ID	vpc-02973363e595f38c6	Tenancy	default
State	available	Default VPC	No
IPv4 CIDR	192.168.0.0/16	IPv6 CIDR	-
IPv6 Pool	-	DNS resolution	Enabled
Network ACL	acl-0196c343525d004ba	DNS hostnames	Disabled
DHCP options set	dopt-57758d3c	Route table	rtb-0ed36d616f5f035d5
Owner	410914255776		

> creating two subnet 1 has auto-launch ip

```
resource "aws_subnet" "hw_subnet-1a" {
  vpc_id     = "${aws_vpc.hw.id}"
  cidr_block = "192.168.0.0/24"
  availability_zone = "ap-south-1a"
  map_public_ip_on_launch = true
}
resource "aws_subnet" "hw_subnet-1b" {
  vpc_id     = "${aws_vpc.hw.id}"
  cidr_block = "192.168.1.0/24"
  availability_zone = "ap-south-1b"
}
```

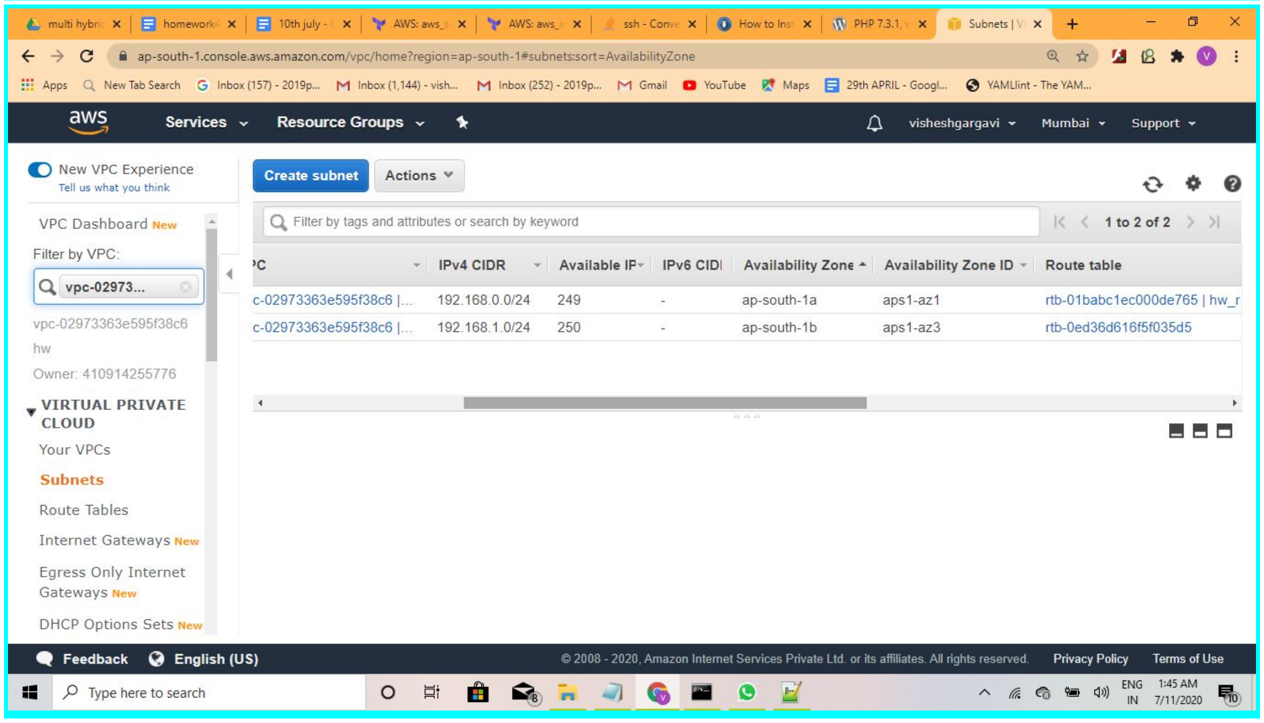


ap-south-1.console.aws.amazon.com/vpc/home?region=ap-south-1#subnets:sort=AvailabilityZone

vpc-02973363e595f38c6
hw
Owner: 410914255776

Name	Subnet ID	State	VPC	IPv4 CIDR	Available IP	IPv6 CIDR	Av
	subnet-03c940b59379d112e	available	vpc-02973363e595f38c6 |...	192.168.0.0/24	249	-	ap-
	subnet-09fdbb2e1248e3d56	available	vpc-02973363e595f38c6 |...	192.168.1.0/24	250	-	ap-

© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use

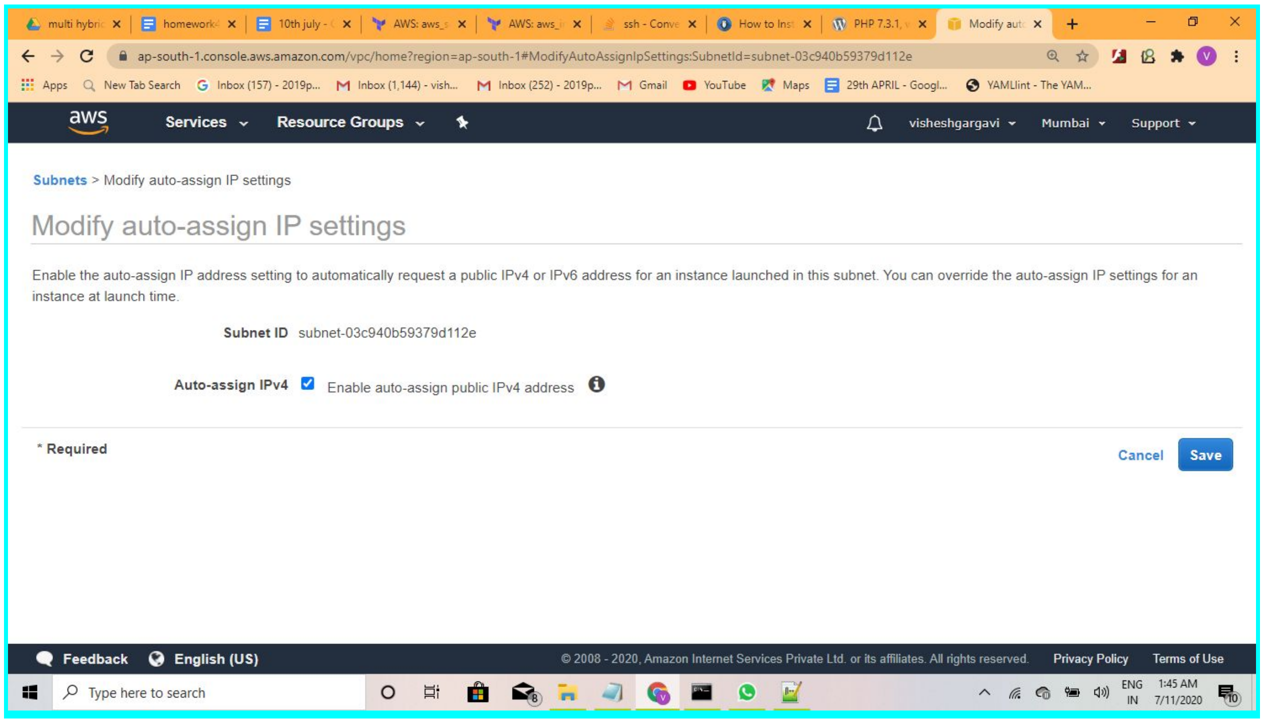


ap-south-1.console.aws.amazon.com/vpc/home?region=ap-south-1#subnets:sort=AvailabilityZone

vpc-02973363e595f38c6
hw
Owner: 410914255776

PC	IPv4 CIDR	Available IP	IPv6 CIDR	Availability Zone	Availability Zone ID	Route table
c-02973363e595f38c6 |...	192.168.0.0/24	249	-	ap-south-1a	aps1-az1	rtb-01babc1ec000de765 | hw_r
c-02973363e595f38c6 |...	192.168.1.0/24	250	-	ap-south-1b	aps1-az3	rtb-0ed36d616f5f035d5

© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use



ap-south-1.console.aws.amazon.com/vpc/home?region=ap-south-1#ModifyAutoAssignIpSettings:SubnetId=subnet-03c940b59379d112e

Subnets > Modify auto-assign IP settings

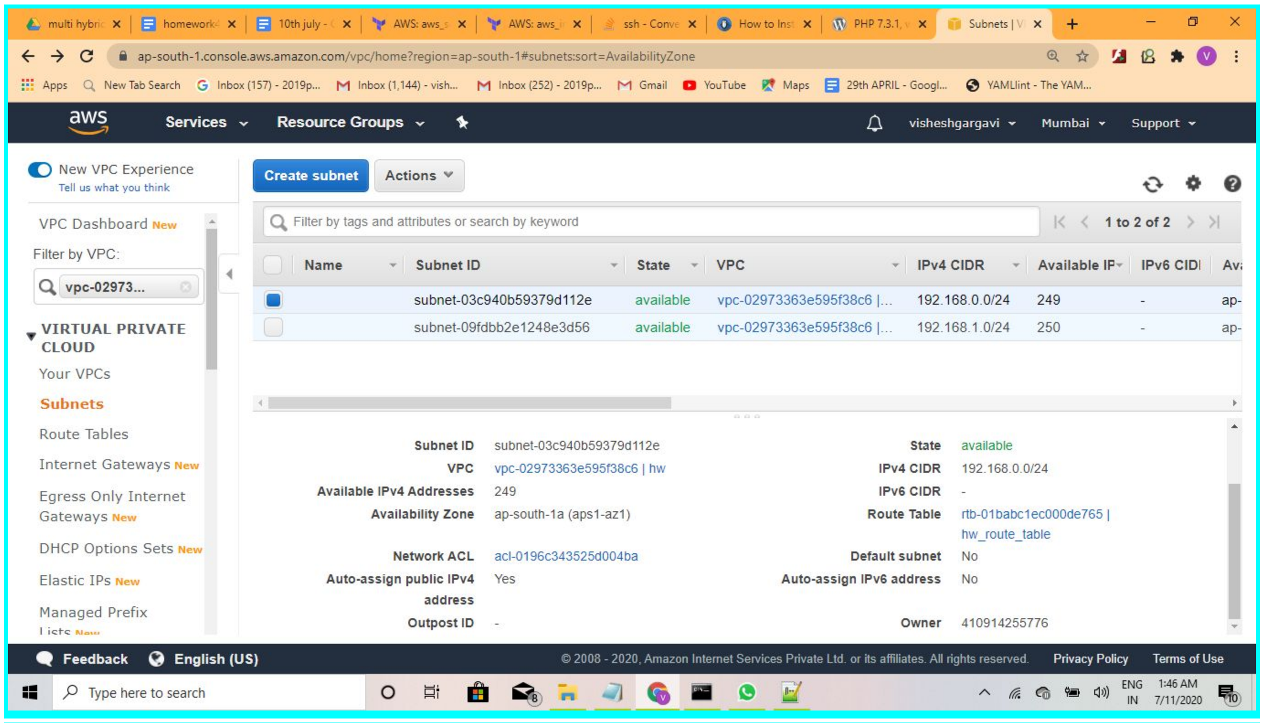
Modify auto-assign IP settings

Enable the auto-assign IP address setting to automatically request a public IPv4 or IPv6 address for an instance launched in this subnet. You can override the auto-assign IP settings for an instance at launch time.

Subnet ID subnet-03c940b59379d112e

Auto-assign IPv4 [✓] Enable auto-assign public IPv4 address

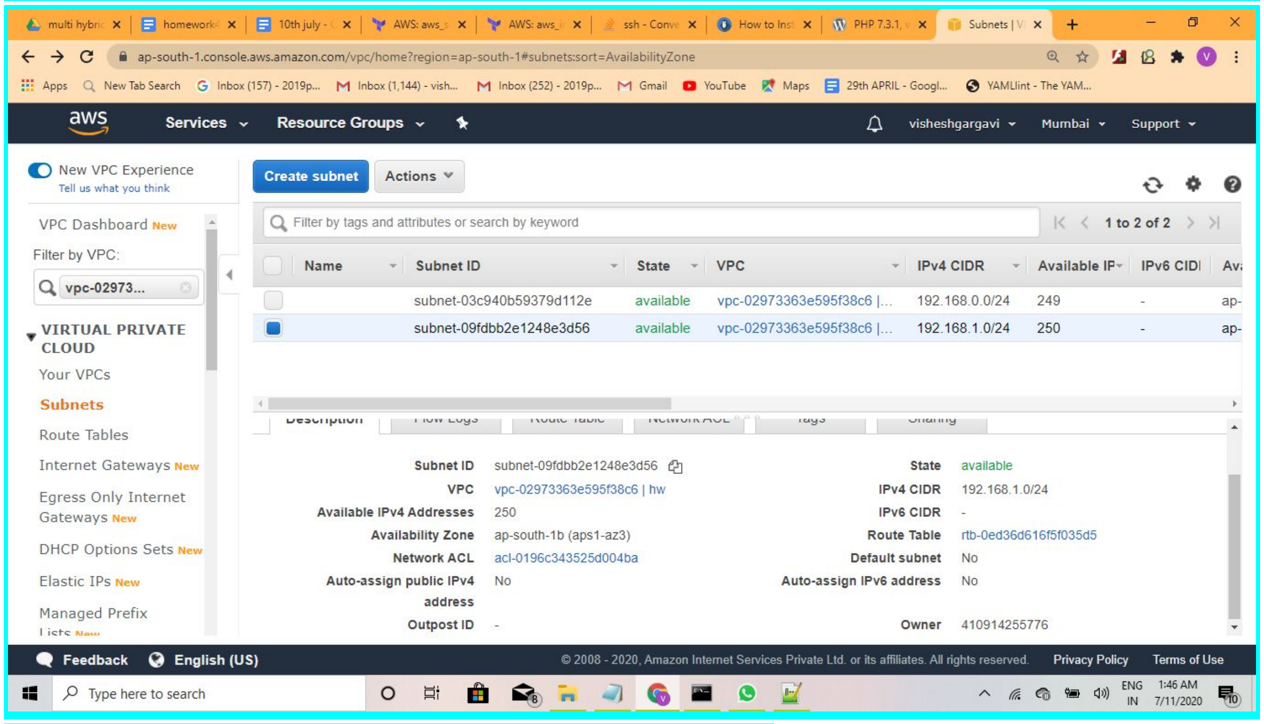
* Required



multi hybrid | homework | 10th july | AWS: aws_ | AWS: aws_ | ssh - Conve | How to Inst | PHP 7.3.1 | Subnets | V
ap-south-1.console.aws.amazon.com/vpc/home?region=ap-south-1#subnets:sort=AvailabilityZone
Apps | New Tab Search | Inbox (157) - 2019p... | Inbox (1,144) - vish... | Inbox (252) - 2019p... | Gmail | YouTube | Maps | 29th APRIL - Googl... | YAMLint - The YAM...
aws | Services | Resource Groups | visheshgargavi | Mumbai | Support
New VPC Experience
Tell us what you think
VPC Dashboard New
Filter by VPC:
vpc-02973...
VIRTUAL PRIVATE CLOUD
Your VPCs
Subnets
Route Tables
Internet Gateways New
Egress Only Internet Gateways New
DHCP Options Sets New
Elastic IPs New
Managed Prefix Lists New
Create subnet | Actions
Filter by tags and attributes or search by keyword
1 to 2 of 2
Feedback | English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use
Type here to search
ENG IN 1:46 AM 7/11/2020

	Name	Subnet ID	State	VPC	IPv4 CIDR	Available IP	IPv6 CIDR	Av
■		subnet-03c940b59379d112e	available	vpc-02973363e595f38c6 |...	192.168.0.0/24	249	-	ap-
		subnet-09fdbb2e1248e3d56	available	vpc-02973363e595f38c6 |...	192.168.1.0/24	250	-	ap-

Subnet ID	subnet-03c940b59379d112e	State	available
VPC	vpc-02973363e595f38c6 | hw	IPv4 CIDR	192.168.0.0/24
Available IPv4 Addresses	249	IPv6 CIDR	-
Availability Zone	ap-south-1a (aps1-az1)	Route Table	rtb-01babc1ec000de765 | hw_route_table
Network ACL	acl-0196c343525d004ba	Default subnet	No
Auto-assign public IPv4 address	Yes	Auto-assign IPv6 address	No
Outpost ID	-	Owner	410914255776



multi hybrid | homework | 10th july | AWS: aws_ | AWS: aws_ | ssh - Conve | How to Inst | PHP 7.3.1 | Subnets | V
ap-south-1.console.aws.amazon.com/vpc/home?region=ap-south-1#subnets:sort=AvailabilityZone
aws | Services | Resource Groups | visheshgargavi | Mumbai | Support
New VPC Experience
Tell us what you think
VPC Dashboard New
Filter by VPC:
vpc-02973...
VIRTUAL PRIVATE CLOUD
Your VPCs
Subnets
Route Tables
Internet Gateways New
Egress Only Internet Gateways New
DHCP Options Sets New
Elastic IPs New
Managed Prefix Lists New
Create subnet | Actions
Filter by tags and attributes or search by keyword
1 to 2 of 2
Description | Flow Logs | Route Table | Network ACL | Tags | Sharing
Feedback | English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use
Type here to search
ENG IN 1:46 AM 7/11/2020

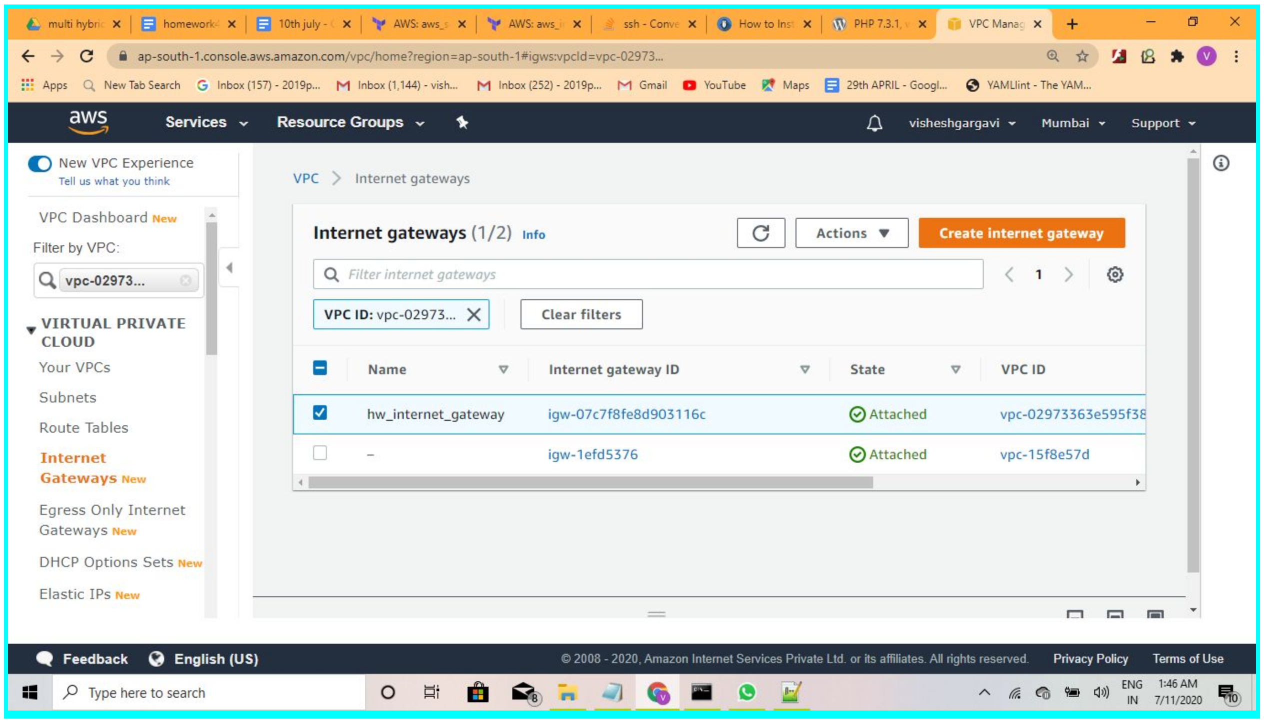
	Name	Subnet ID	State	VPC	IPv4 CIDR	Available IP	IPv6 CIDR	Av
		subnet-03c940b59379d112e	available	vpc-02973363e595f38c6 |...	192.168.0.0/24	249	-	ap-
■		subnet-09fdbb2e1248e3d56	available	vpc-02973363e595f38c6 |...	192.168.1.0/24	250	-	ap-

Subnet ID	subnet-09fdbb2e1248e3d56	State	available
VPC	vpc-02973363e595f38c6 | hw	IPv4 CIDR	192.168.1.0/24
Available IPv4 Addresses	250	IPv6 CIDR	-
Availability Zone	ap-south-1b (aps1-az3)	Route Table	rtb-0ed36d616f5f035d5
Network ACL	acl-0196c343525d004ba	Default subnet	No
Auto-assign public IPv4 address	No	Auto-assign IPv6 address	No
Outpost ID	-	Owner	410914255776

> creating an internet gateway for a subnet id in 1a

```
resource "aws_internet_gateway" "hw_internet_gateway" {
 vpc_id = "${aws_vpc.hw.id}"

 tags = {
   Name = "hw_internet_gateway"
```

```
  }
}
```



VPC > Internet gateways

Internet gateways (1/2) Info

Actions ▼ | Create internet gateway

Filter internet gateways

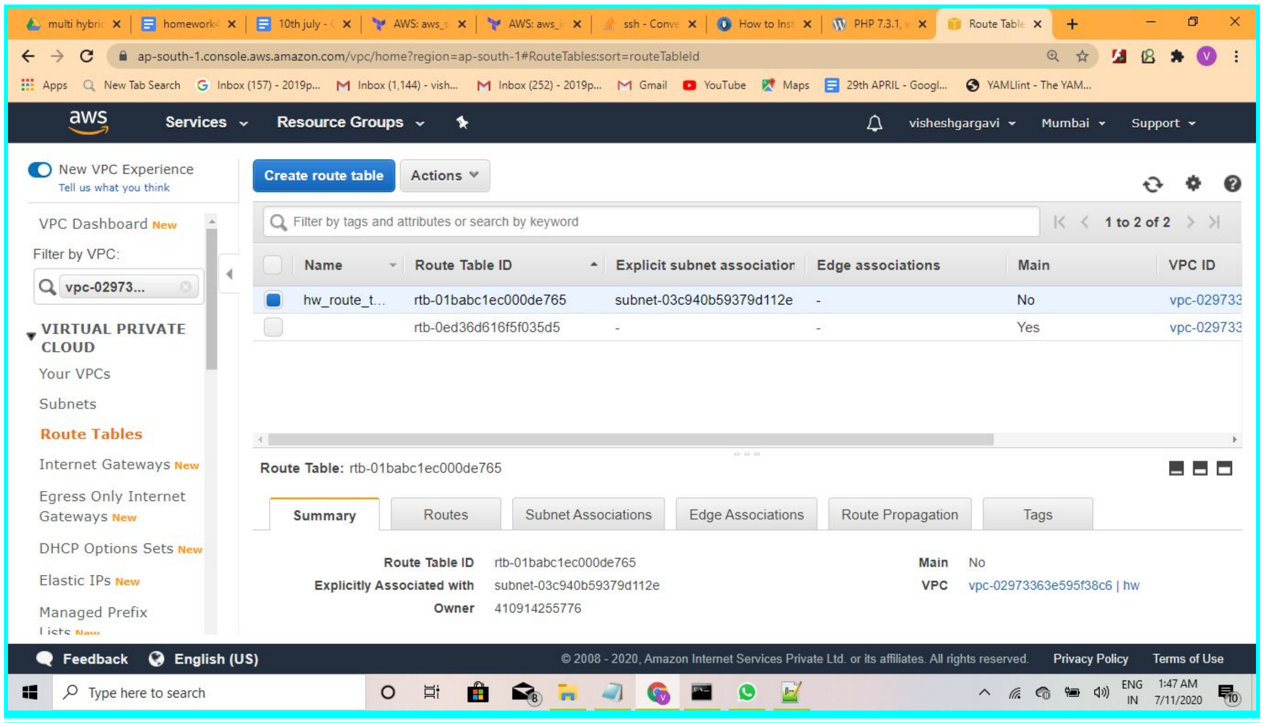
VPC ID: vpc-02973... ✕ | Clear filters

	Name	Internet gateway ID	State	VPC ID
✓	hw_internet_gateway	igw-07c7f8fe8d903116c	Attached	vpc-02973363e595f38
	–	igw-1efd5376	Attached	vpc-15f8e57d

© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

> creating a route-table
> associating route-table with the internet gateway

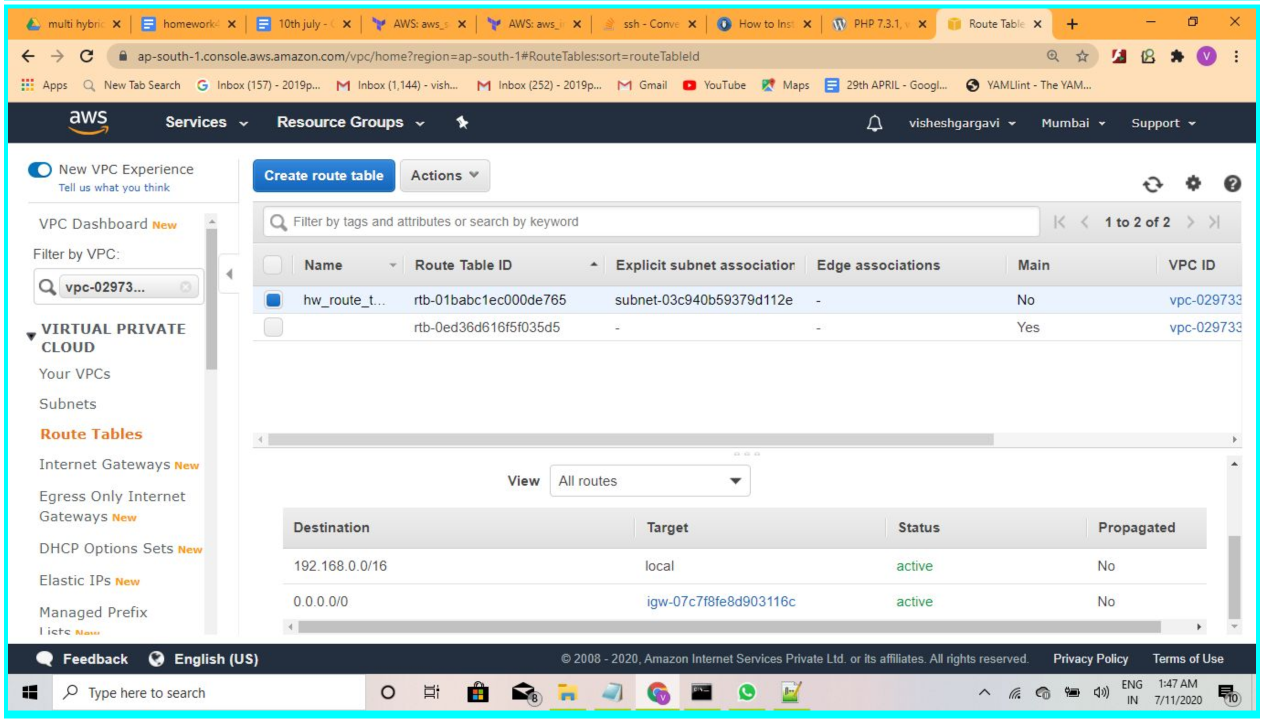
```
resource "aws_route_table" "hw_route_table" {
  vpc_id = "${aws_vpc.hw.id}"

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_internet_gateway.hw_internet_gateway.id}"
  }

  tags = {
    Name = "hw_route_table"
  }
}
```



Route Table: rtb-01babc1ec000de765

Name	Route Table ID	Explicit subnet association	Edge associations	Main	VPC ID
hw_route_t...	rtb-01babc1ec000de765	subnet-03c940b59379d112e	-	No	vpc-029733
	rtb-0ed36d616f5f035d5	-	-	Yes	vpc-029733

Route Table ID rtb-01babc1ec000de765
Explicitly Associated with subnet-03c940b59379d112e
Owner 410914255776
Main No
VPC vpc-02973363e595f38c6 | hw

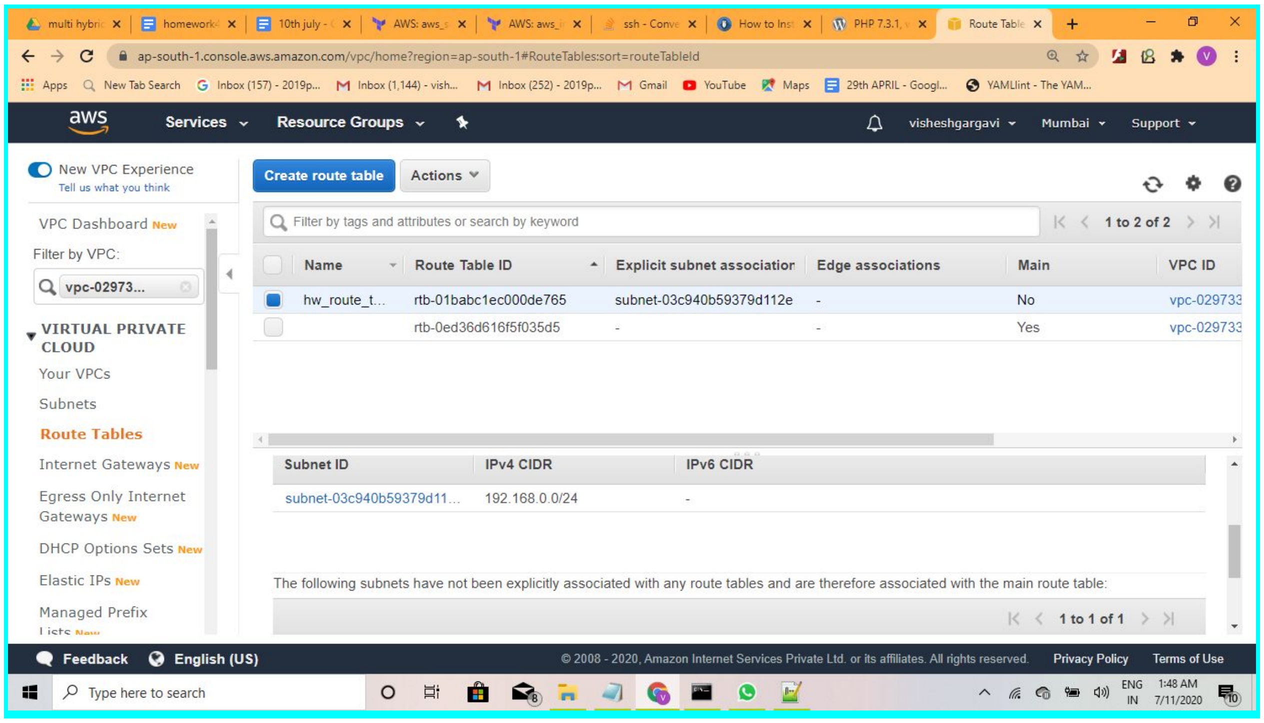


View All routes

Destination	Target	Status	Propagated
192.168.0.0/16	local	active	No
0.0.0.0/0	igw-07c7f8fe8d903116c	active	No

> associating route table with subnet

```
resource "aws_route_table_association" "a" {
  subnet_id      = aws_subnet.hw_subnet-1a.id
  route_table_id = "${aws_route_table.hw_route_table.id}"
}
```



ap-south-1.console.aws.amazon.com/vpc/home?region=ap-south-1#RouteTables:sort=routeTableId

Name	Route Table ID	Explicit subnet association	Edge associations	Main	VPC ID
hw_route_t...	rtb-01babc1ec000de765	subnet-03c940b59379d112e	-	No	vpc-029733
	rtb-0ed36d616f5f035d5	-	-	Yes	vpc-029733

Subnet ID	IPv4 CIDR	IPv6 CIDR
subnet-03c940b59379d11...	192.168.0.0/24	-

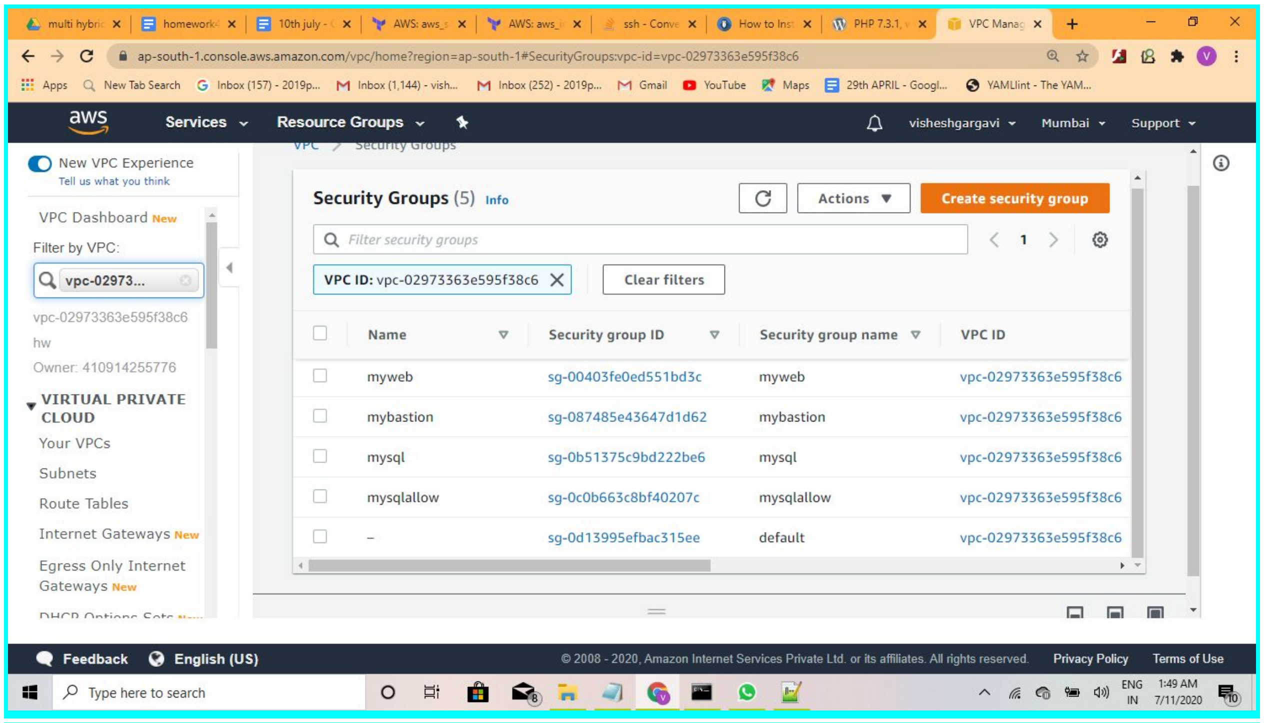
The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:

1 to 1 of 1

© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use

> creating the security group with ingress(ssh,http and icmpv4 protocol)
> myweb

```
resource "aws_security_group" "myweb" {
  name        = "myweb"
  description = "Allow ssh http and icmp"
  vpc_id      = "${aws_vpc.hw.id}"

  ingress {
    description = "http"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
 ingress {
    description = "ssh"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
ingress {
    description = "ICMP-IPv4"
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    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 = "myweb"
  }
}
```



ap-south-1.console.aws.amazon.com/vpc/home?region=ap-south-1#SecurityGroups:vpc-id=vpc-02973363e595f38c6

vpc-02973363e595f38c6
hw
Owner: 410914255776

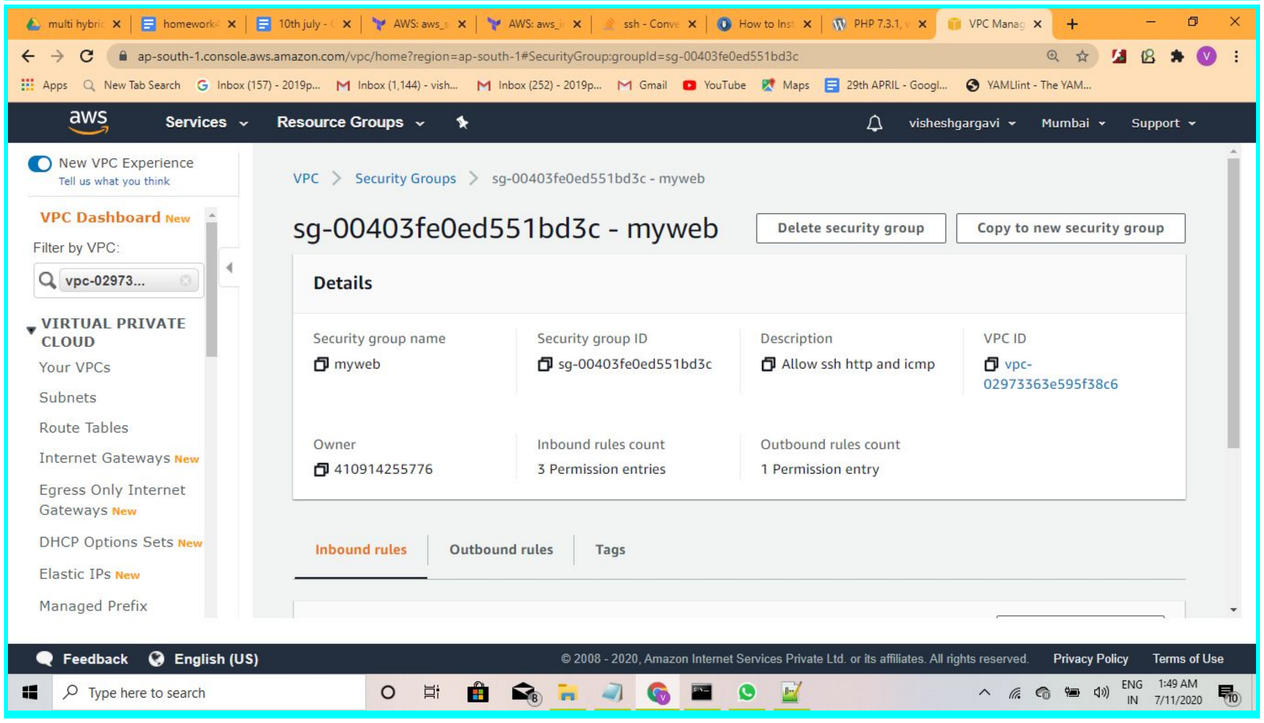
Security Groups (5) Info

VPC ID: vpc-02973363e595f38c6

Name	Security group ID	Security group name	VPC ID
myweb	sg-00403fe0ed551bd3c	myweb	vpc-02973363e595f38c6
mybastion	sg-087485e43647d1d62	mybastion	vpc-02973363e595f38c6
mysql	sg-0b51375c9bd222be6	mysql	vpc-02973363e595f38c6
mysqlallow	sg-0c0b663c8bf40207c	mysqlallow	vpc-02973363e595f38c6
–	sg-0d13995efbac315ee	default	vpc-02973363e595f38c6

© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use

1:49 AM 7/11/2020



ap-south-1.console.aws.amazon.com/vpc/home?region=ap-south-1#SecurityGroup:groupId=sg-00403fe0ed551bd3c

VPC > Security Groups > sg-00403fe0ed551bd3c - myweb

sg-00403fe0ed551bd3c - myweb

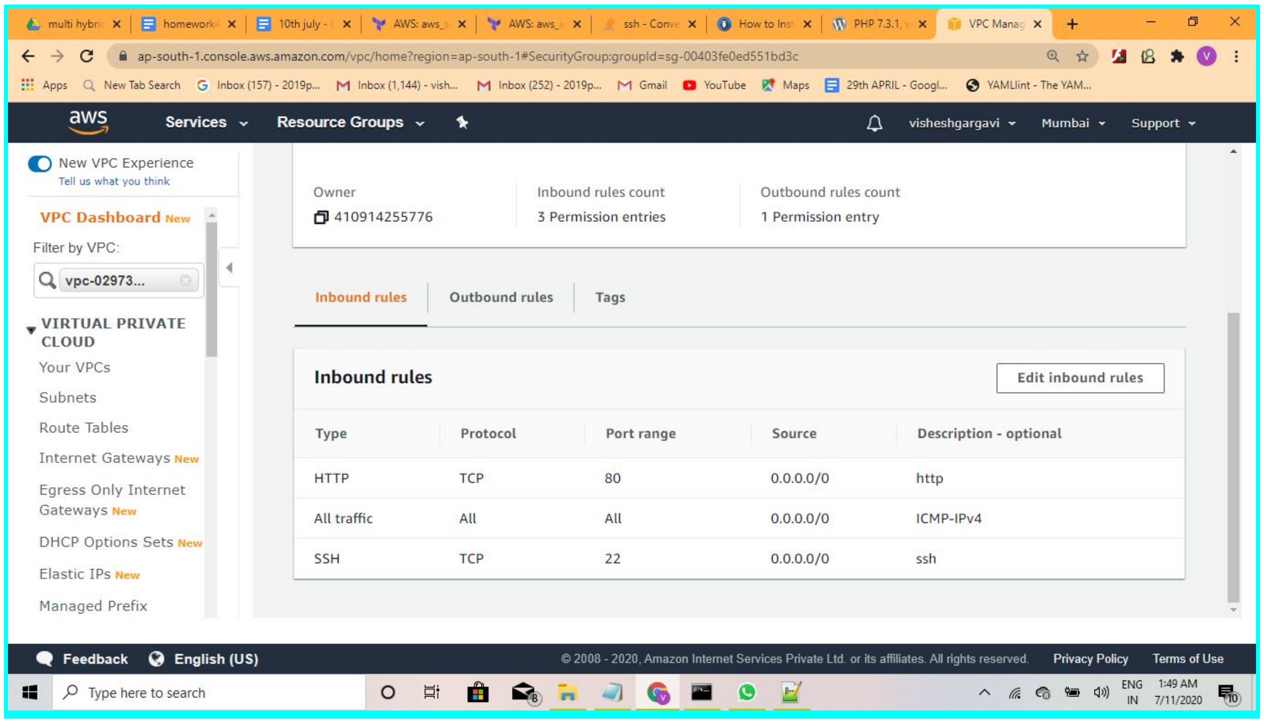
Details

Security group name	Security group ID	Description	VPC ID
myweb	sg-00403fe0ed551bd3c	Allow ssh http and icmp	vpc-02973363e595f38c6

Owner	Inbound rules count	Outbound rules count
410914255776	3 Permission entries	1 Permission entry

© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use

1:49 AM 7/11/2020



Owner	Inbound rules count	Outbound rules count
410914255776	3 Permission entries	1 Permission entry

Inbound rules | Outbound rules | Tags

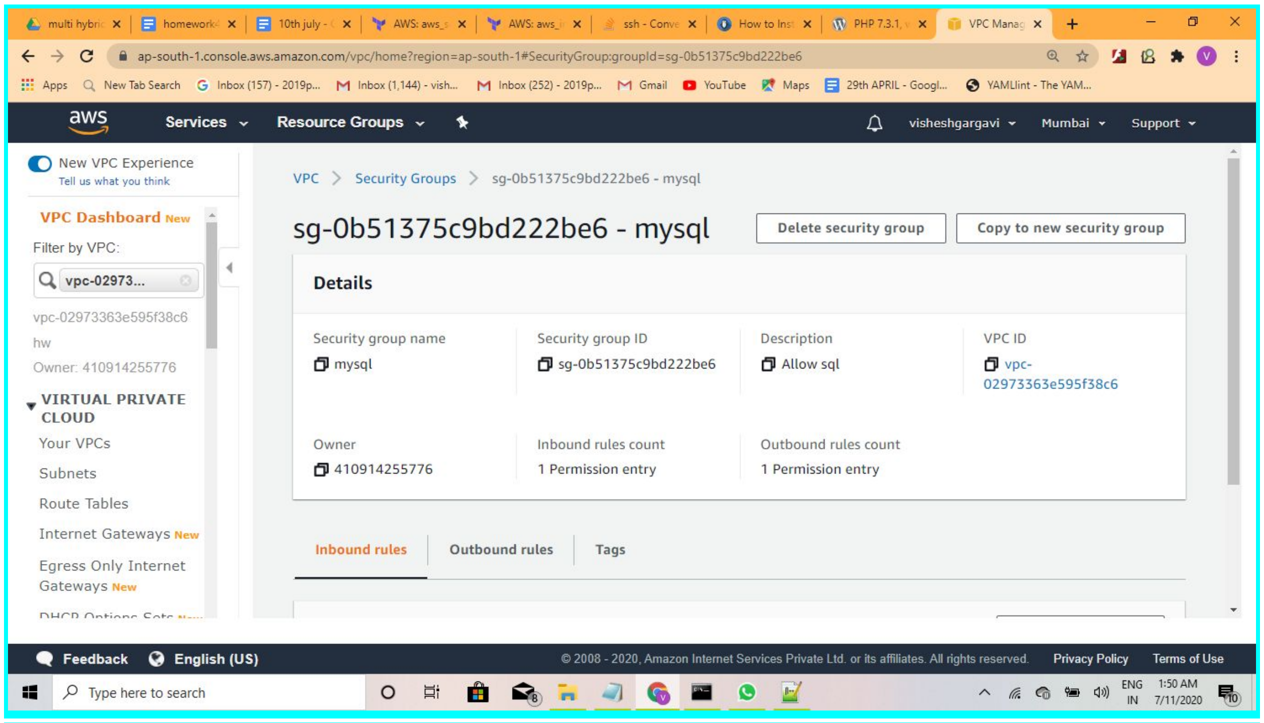
Inbound rules

Edit inbound rules

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	0.0.0.0/0	http
All traffic	All	All	0.0.0.0/0	ICMP-IPv4
SSH	TCP	22	0.0.0.0/0	ssh

> creating a subnet group with MYSQL protocol and value of security_id(myweb) > mysql

```
resource "aws_security_group" "mysql" {
  name        = "mysql"
  description = "Allow sql"
  vpc_id      = "${aws_vpc.hw.id}"

  ingress {
    description = "MYSQL"
    security_groups=[ "${aws_security_group.myweb.id}" ]
    from_port   = 3306
    to_port     = 3306
    protocol    = "tcp"
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
   Name = "mysql"
 }
}
```

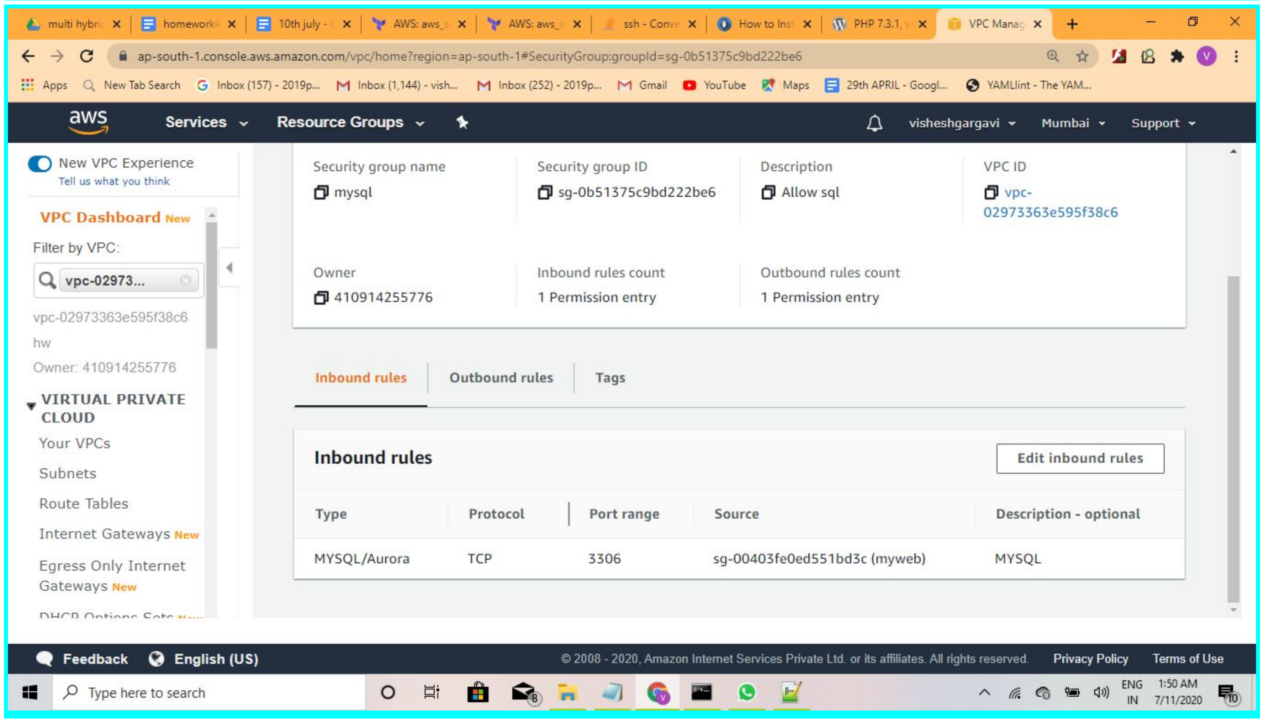


sg-0b51375c9bd222be6 - mysql

Details

Security group name	Security group ID	Description	VPC ID
mysql	sg-0b51375c9bd222be6	Allow sql	vpc-02973363e595f38c6

Owner	Inbound rules count	Outbound rules count
410914255776	1 Permission entry	1 Permission entry

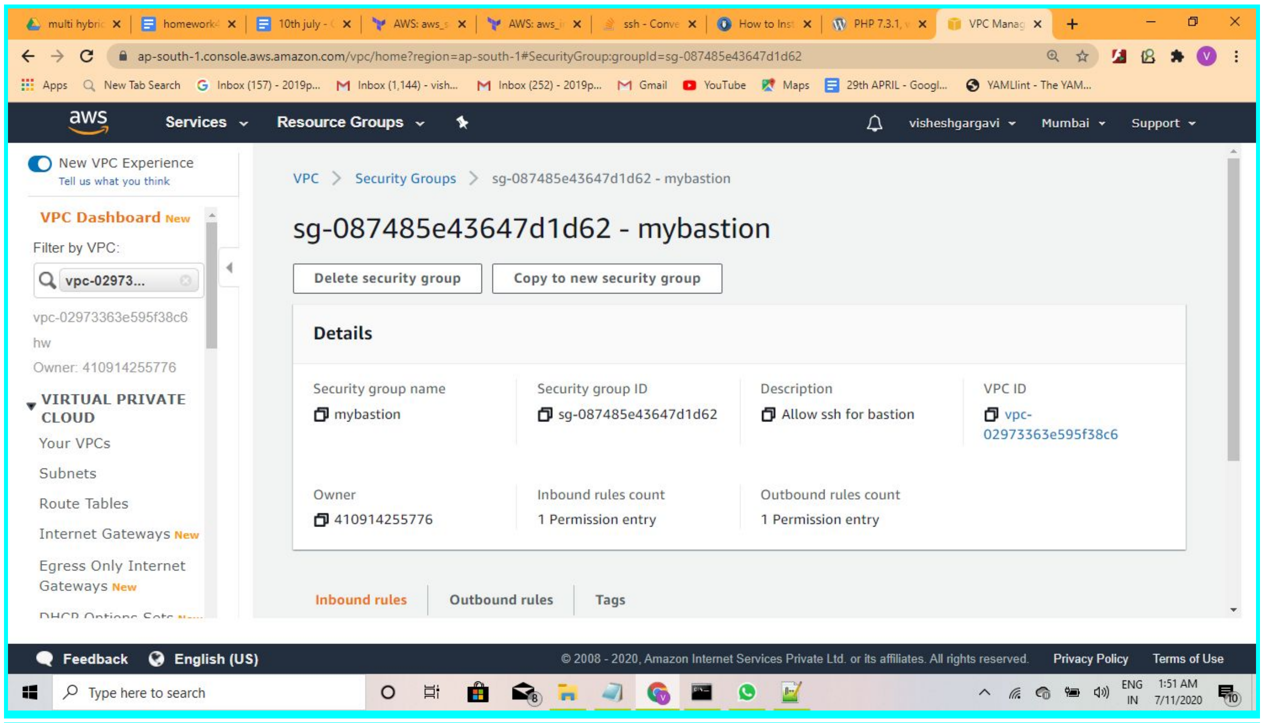


Inbound rules

Type	Protocol	Port range	Source	Description - optional
MYSQL/Aurora	TCP	3306	sg-00403fe0ed551bd3c (myweb)	MYSQL

> creating a security group with ssh protocol
> bastion

```
resource "aws_security_group" "mybastion" {
  name        = "mybastion"
  description = "Allow ssh for bastion"
  vpc_id      = "${aws_vpc.hw.id}"

  ingress {
    description = "ssh"
    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 = "mybastion"
  }
}
```



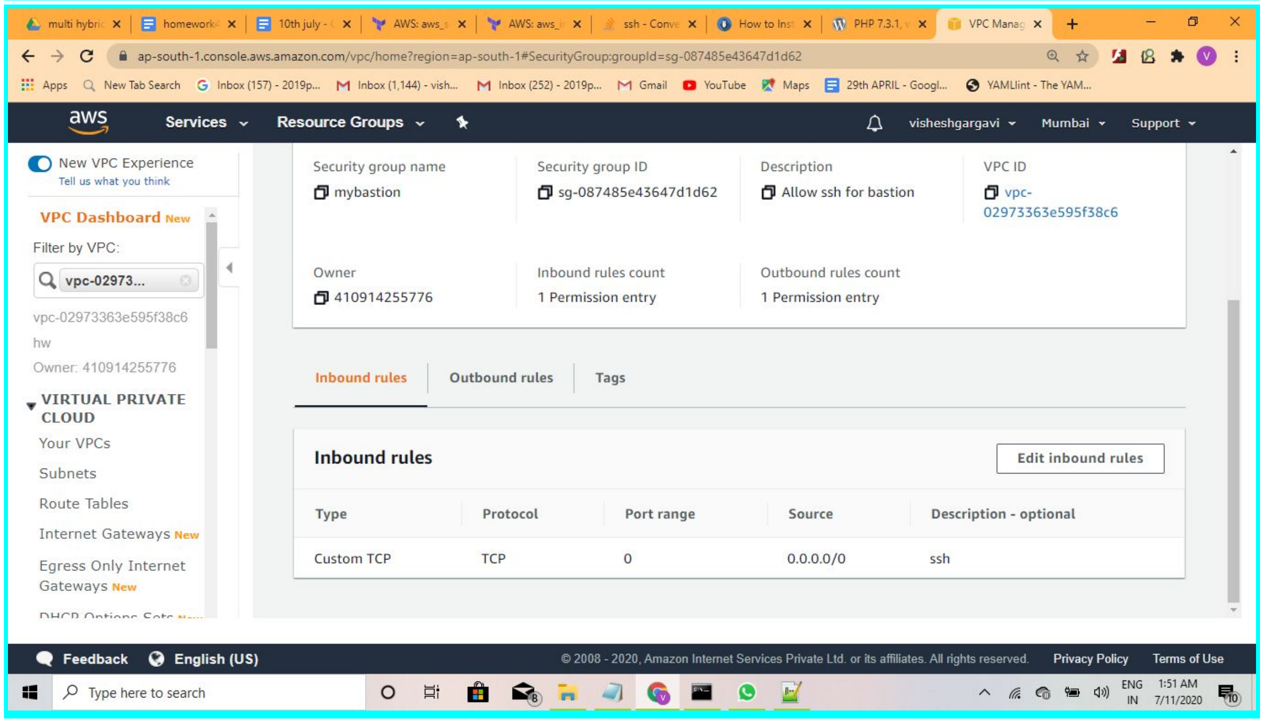
VPC > Security Groups > sg-087485e43647d1d62 - mybastion

sg-087485e43647d1d62 - mybastion

Details

Security group name	Security group ID	Description	VPC ID
mybastion	sg-087485e43647d1d62	Allow ssh for bastion	vpc-02973363e595f38c6

Owner	Inbound rules count	Outbound rules count
410914255776	1 Permission entry	1 Permission entry



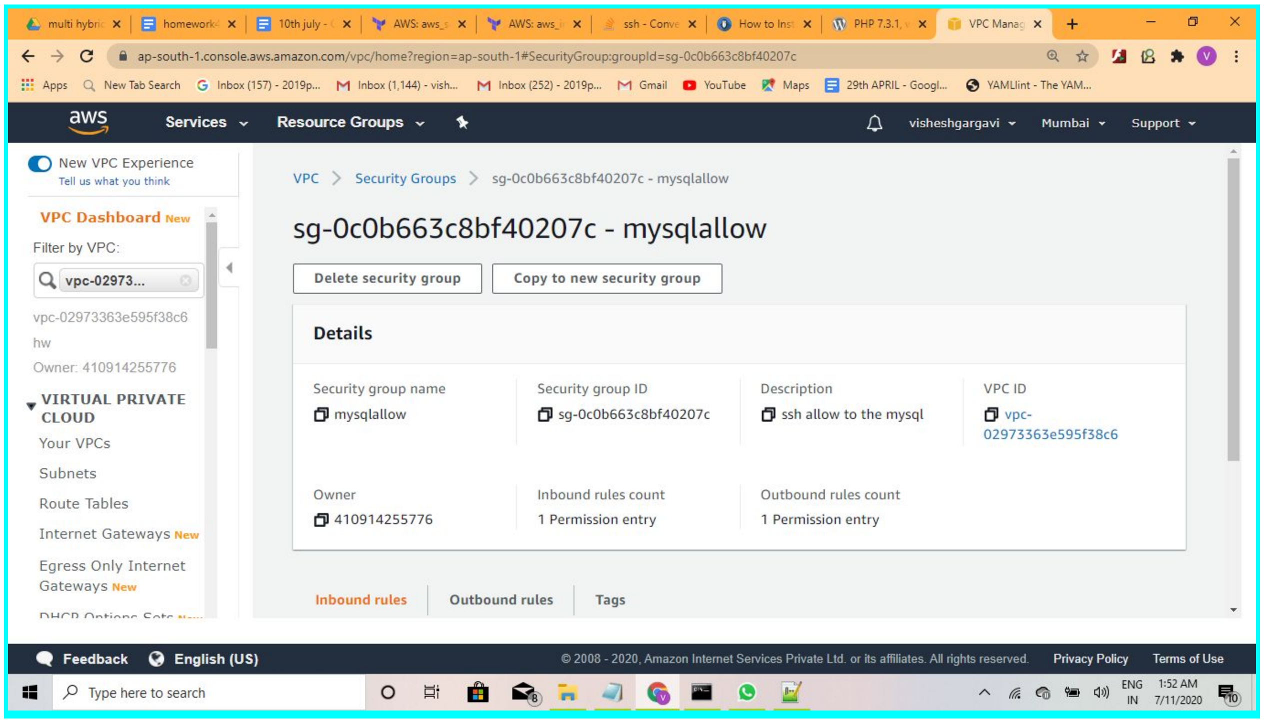
Inbound rules

Type	Protocol	Port range	Source	Description - optional
Custom TCP	TCP	0	0.0.0.0/0	ssh

> creating a subnet group with ssh protocol and value as security_id(mybastion) > mysqlallow

resource "aws_security_group" "mysqlallow" {
name   = "mysqlallow"

```
  description = "ssh allow to the mysql"
  vpc_id      = "${aws_vpc.hw.id}"

  ingress {
    description = "ssh"
    security_groups=[ "${aws_security_group.mybastion.id}" ]
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "mysqlallow"
  }
}
```

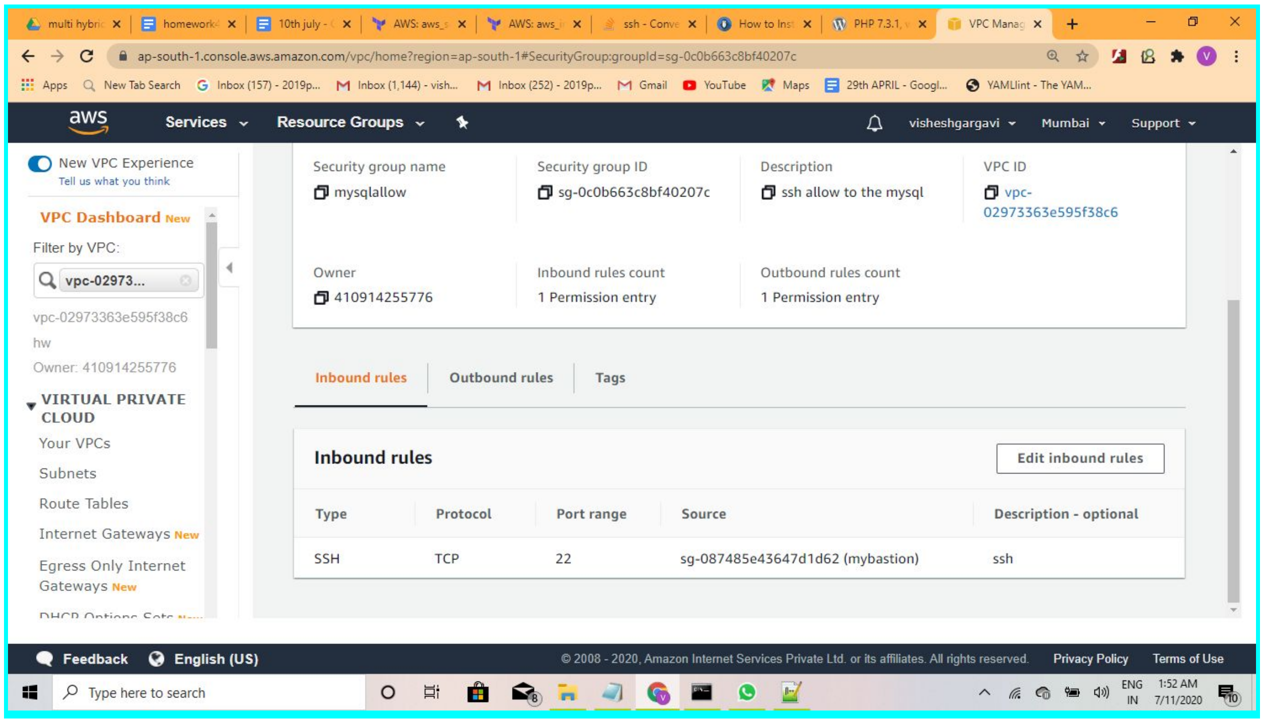

VPC > Security Groups > sg-0c0b663c8bf40207c - mysqlallow

sg-0c0b663c8bf40207c - mysqlallow

Details

Security group name	Security group ID	Description	VPC ID
mysqlallow	sg-0c0b663c8bf40207c	ssh allow to the mysql	vpc-02973363e595f38c6

Owner	Inbound rules count	Outbound rules count
410914255776	1 Permission entry	1 Permission entry

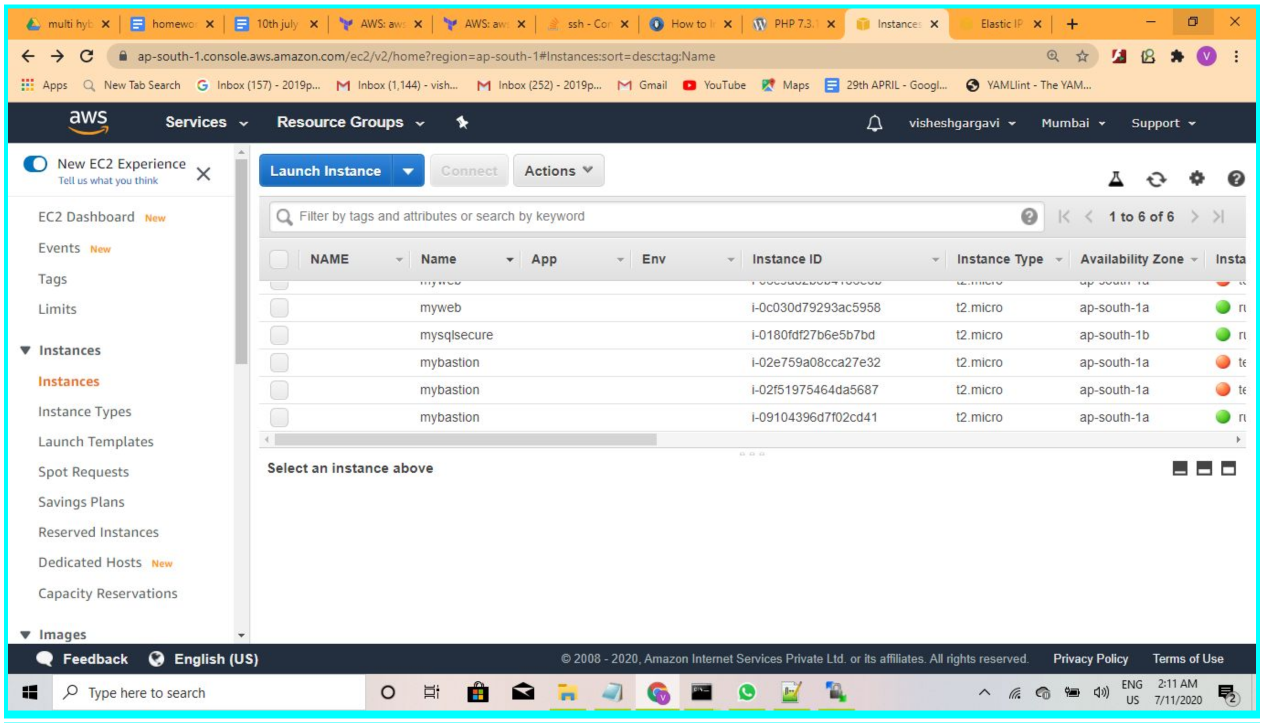

Inbound rules

Type	Protocol	Port range	Source	Description - optional
SSH	TCP	22	sg-087485e43647d1d62 (mybastion)	ssh

> launching the instance with the ami image in the 1a region and attaching the security group myweb
> enabling the public-ip

resource "aws_instance" "myweb" {

```
  ami           = "ami-0b39bdec65a992579"
  instance_type = "t2.micro"
  key_name      = "mykey1111.pem"
  availability_zone = "ap-south-1a"
  subnet_id = "${aws_subnet.hw_subnet-1a.id}"
  security_groups = [ "${aws_security_group.myweb.id}" ]
  tags = {
    Name = "myweb"
  }
}
```



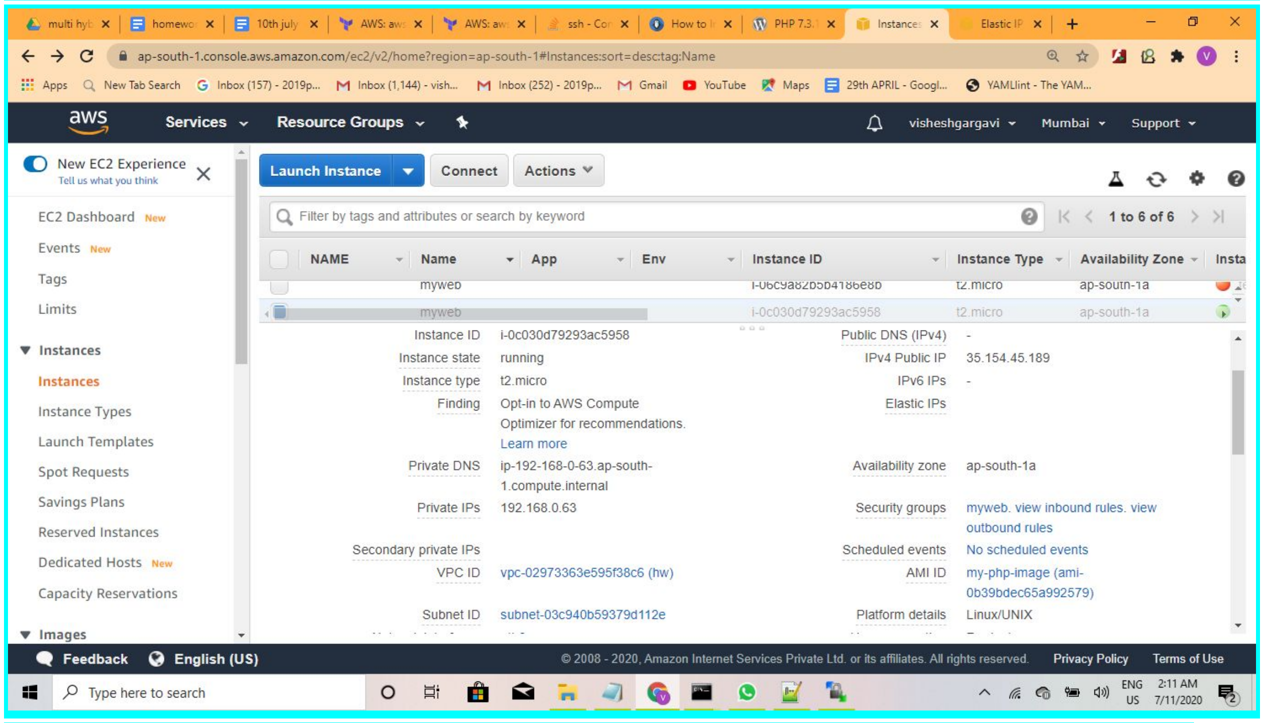
ap-south-1.console.aws.amazon.com/ec2/v2/home?region=ap-south-1#Instances:sort=desc:tag:Name

Filter by tags and attributes or search by keyword — 1 to 6 of 6

NAME	Name	App	Env	Instance ID	Instance Type	Availability Zone	Insta
	myweb			i-0c030d79293ac5958	t2.micro	ap-south-1a	● ru
	mysqlsecure			i-0180fdf27b6e5b7bd	t2.micro	ap-south-1b	● ru
	mybastion			i-02e759a08cca27e32	t2.micro	ap-south-1a	● te
	mybastion			i-02f51975464da5687	t2.micro	ap-south-1a	● te
	mybastion			i-09104396d7f02cd41	t2.micro	ap-south-1a	● ru

Select an instance above

© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use



NAME	Name	App	Env	Instance ID	Instance Type	Availability Zone	Insta
	myweb			i-06c9a82b5b4186e8b	t2.micro	ap-south-1a	●
	myweb			i-0c030d79293ac5958	t2.micro	ap-south-1a	●

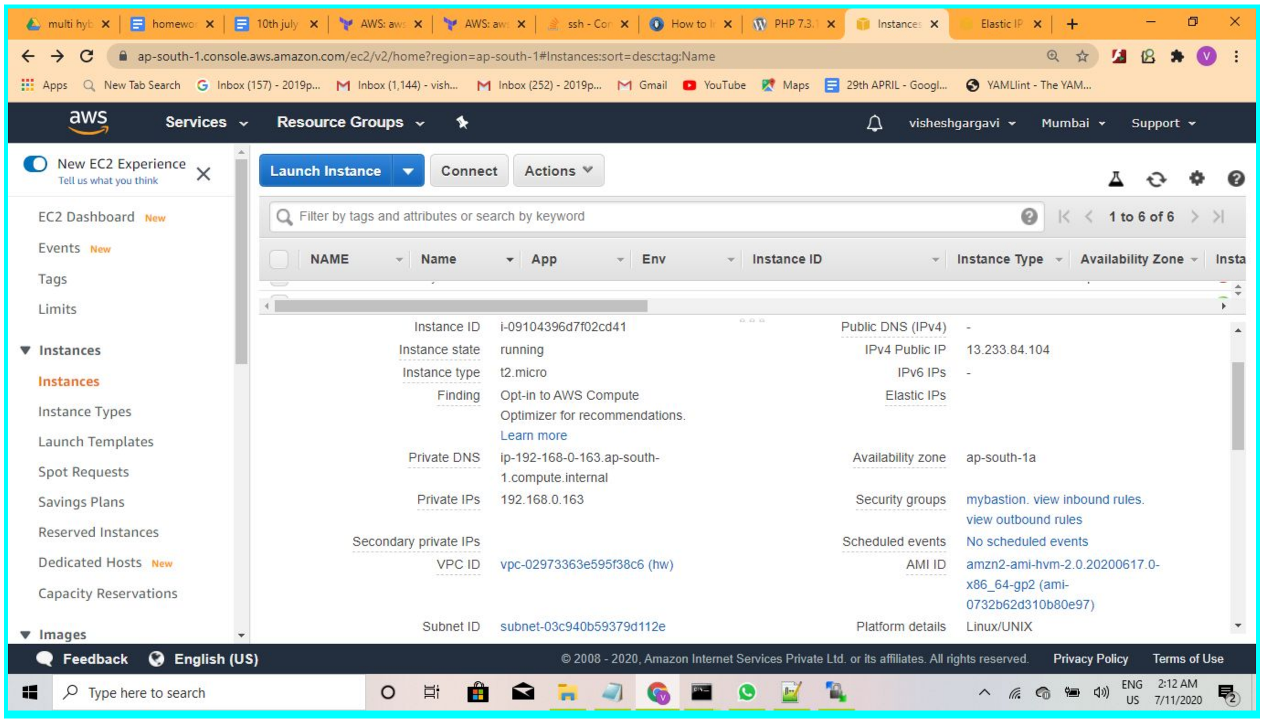
Instance ID	i-0c030d79293ac5958	Public DNS (IPv4)	-
Instance state	running	IPv4 Public IP	35.154.45.189
Instance type	t2.micro	IPv6 IPs	-
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	Elastic IPs	
Private DNS	ip-192-168-0-63.ap-south-1.compute.internal	Availability zone	ap-south-1a
Private IPs	192.168.0.63	Security groups	myweb. view inbound rules. view outbound rules
Secondary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-02973363e595f38c6 (hw)	AMI ID	my-php-image (ami-0b39bdec65a992579)
Subnet ID	subnet-03c940b59379d112e	Platform details	Linux/UNIX

© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use

> launching the instance with the ami image in the region 1b and attaching the security group mysql and mysqlallow

> not enabling the public-ip

```
resource "aws_instance" "mysqlsecure" {
 ami           = "ami-0b39bdec65a992579"
 instance_type = "t2.micro"
 key_name      = "mykey1111.pem"
 availability_zone = "ap-south-1b"
 subnet_id = "${aws_subnet.hw_subnet-1b.id}"
 security_groups = [ "${aws_security_group.mysql.id}" ,
"${aws_security_group.mysqlallow.id}"]

 tags = {
   Name = "mysqlsecure"
 }
}
```



ap-south-1.console.aws.amazon.com/ec2/v2/home?region=ap-south-1#Instances:sort=desc:tag:Name

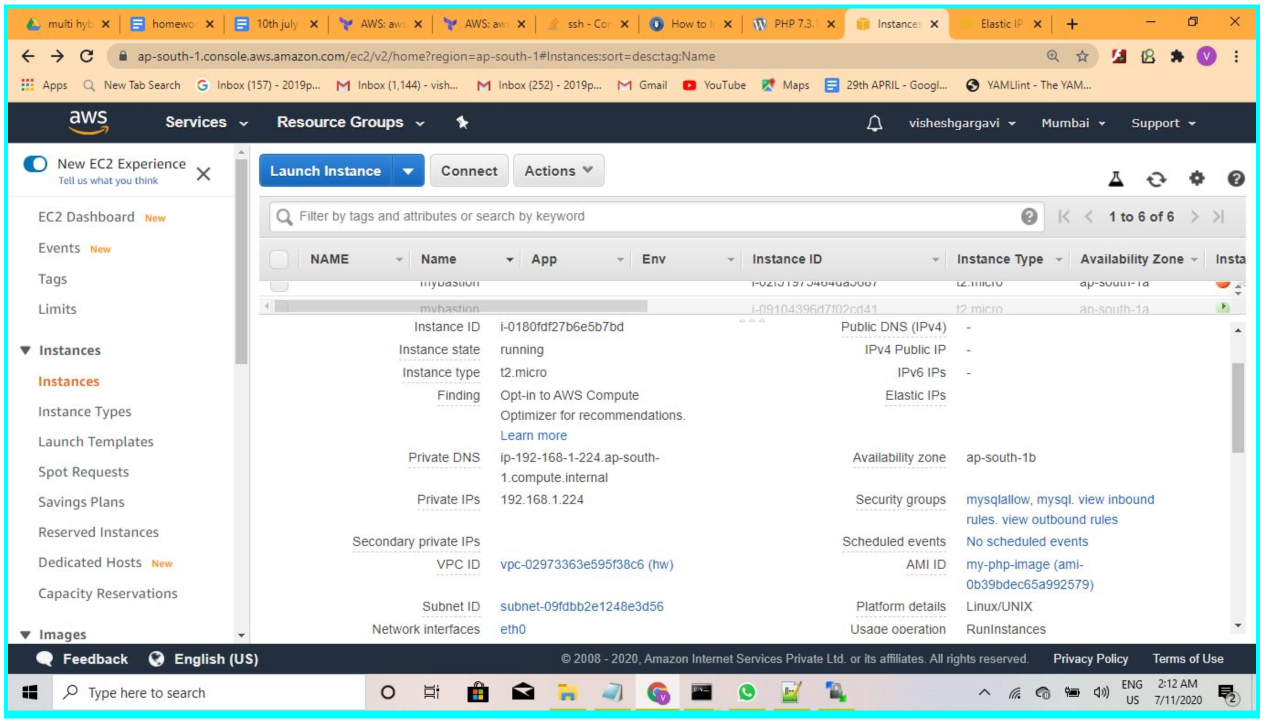
Filter by tags and attributes or search by keyword — 1 to 6 of 6

NAME	Name	App	Env	Instance ID	Instance Type	Availability Zone	Insta

Instance ID	i-09104396d7f02cd41	Public DNS (IPv4)	-
Instance state	running	IPv4 Public IP	13.233.84.104
Instance type	t2.micro	IPv6 IPs	-
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	Elastic IPs	
Private DNS	ip-192-168-0-163.ap-south-1.compute.internal	Availability zone	ap-south-1a
Private IPs	192.168.0.163	Security groups	mybastion. view inbound rules. view outbound rules
Secondary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-02973363e595f38c6 (hw)	AMI ID	amzn2-ami-hvm-2.0.20200617.0-x86_64-gp2 (ami-0732b62d310b80e97)
Subnet ID	subnet-03c940b59379d112e	Platform details	Linux/UNIX

[> launching the instance with the linux image in the region 1a and attaching the security group mybastion](#)
[> enabling the public-ip](#)

```
resource "aws_instance" "mybastion" {
 ami           = "ami-0732b62d310b80e97"
 instance_type = "t2.micro"
 key_name      = "mykey1111.pem"
 availability_zone = "ap-south-1a"
 subnet_id = "${aws_subnet.hw_subnet-1a.id}"
 security_groups = [ "${aws_security_group.mybastion.id}" ]
 tags = {
    Name = "mybastion"
  }
}
```



ap-south-1.console.aws.amazon.com/ec2/v2/home?region=ap-south-1#Instances:sort=desc:tag:Name

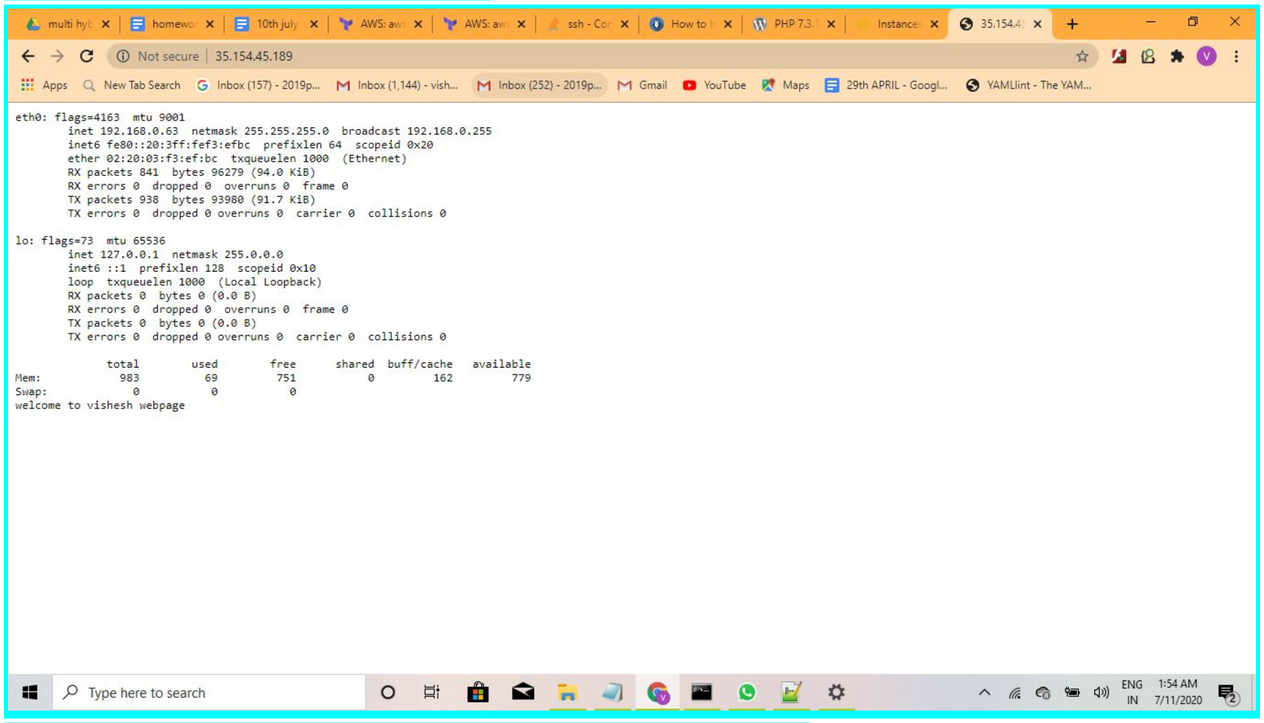
Instance ID	i-0180fdf27b6e5b7bd	Public DNS (IPv4)	-
Instance state	running	IPv4 Public IP	-
Instance type	t2.micro	IPv6 IPs	-
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	Elastic IPs	
Private DNS	ip-192-168-1-224.ap-south-1.compute.internal	Availability zone	ap-south-1b
Private IPs	192.168.1.224	Security groups	mysqlallow, mysql. view inbound rules. view outbound rules
Secondary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-02973363e595f38c6 (hw)	AMI ID	my-php-image (ami-0b39bdec65a992579)
Subnet ID	subnet-09fdbb2e1248e3d56	Platform details	Linux/UNIX
Network interfaces	eth0	Usage operation	RunInstances

© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

>> creating an elastic ip for allowing the NAT CONNECTIVITY
> creating a nat gateway and associating the nat_gateway with the elastic_ip
> associating the nat_gatway with the route table

```
resource "aws_eip" "hw_eip" {
  vpc = true

  instance                  = "${aws_instance.mysql.id}"
  associate_with_private_ip = "10.0.0.12"
  depends_on                = ["aws_internet_gateway.hw_internet_gateway"]
}

resource "aws_nat_gateway" "hw_nat_gateway" {
  allocation_id = "${aws_eip.hw_eip.id}"
  subnet_id     = "${aws_subnet.hw_subnet-1b.id}"

  tags = {
    Name = "hw_nat_gateway"
  }
}

resource "aws_route_table" "hw_route_table2" {
  vpc_id = "${aws_vpc.hw.id}"

  route {
    cidr_block = "0.0.0.0/0"
    nat_gateway_id = "${aws_nat_gateway.hw_nat_gateway.id}"
  }
}
```

> output of the myweb ip(public ip)

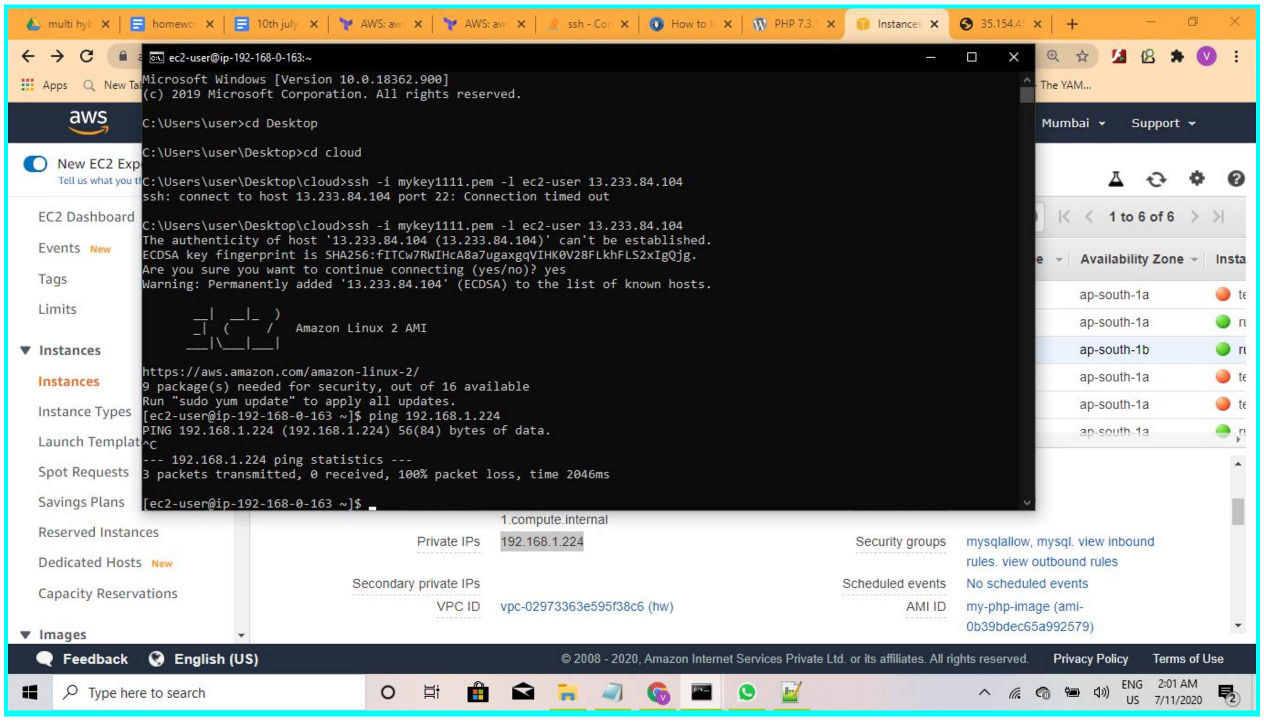


```
eth0: flags=4163  mtu 9001
        inet 192.168.0.63  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::20:3ff:fef3:efbc  prefixlen 64  scopeid 0x20
        ether 02:20:03:f3:ef:bc  txqueuelen 1000  (Ethernet)
        RX packets 841  bytes 96279 (94.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 938  bytes 93980 (91.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

              total        used        free      shared  buff/cache   available
Mem:            983          69         751           0         162         779
Swap:             0           0           0
welcome to vishesh webpage
```

```
C:\Users\user\Desktop\terraform\vpc-inst>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_instance.mybastion will be created
  + resource "aws_instance" "mybastion" {
      + ami                          = "ami-0732b62d310b80e97"
      + arn                          = (known after apply)
```

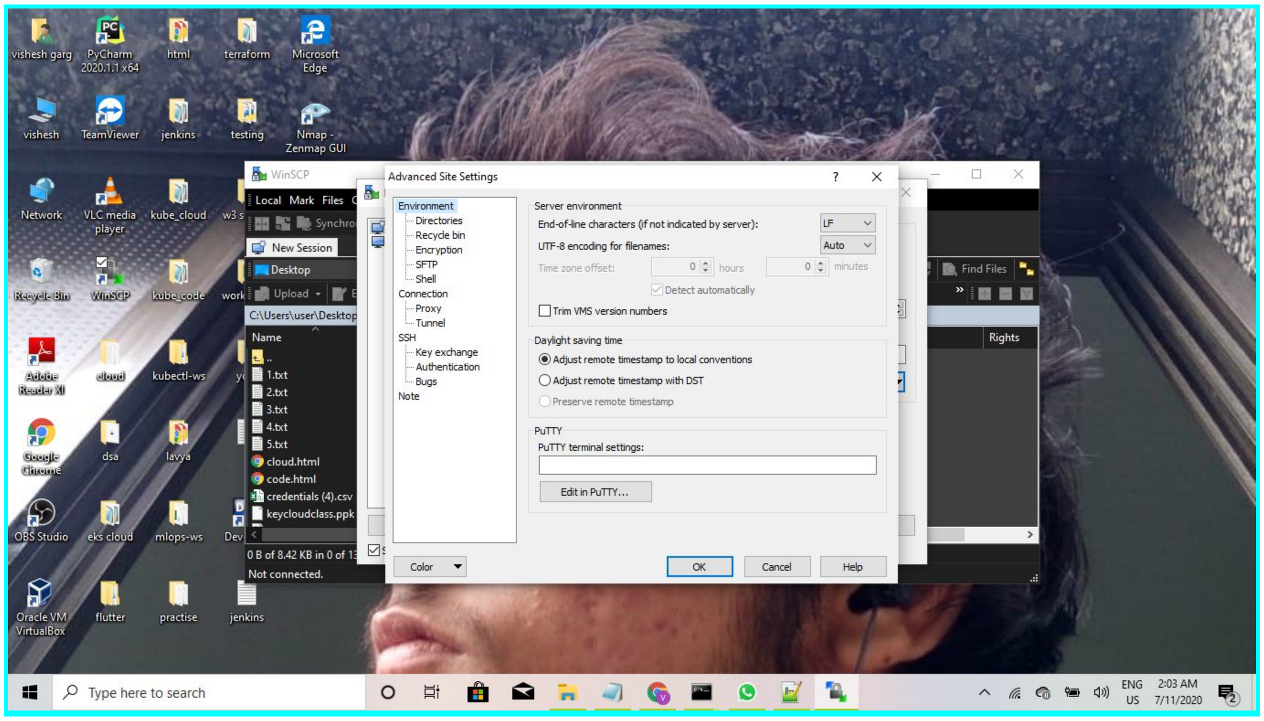
```
    + associate_public_ip_address  = (known after apply)
    + availability_zone            = "ap-south-1a"
    + cpu_core_count               = (known after apply)
    + cpu_threads_per_core         = (known after apply)
    + get_password_data            = false
    + host_id                      = (known after apply)
    + id                           = (known after apply)
    + instance_state               = (known after apply)
    + instance_type                = "t2.micro"
    + ipv6_address_count           = (known after apply)
    + ipv6_addresses               = (known after apply)
    + key_name                     = "mykey1111.pem"
    + network_interface_id         = (known after apply)
    + outpost_arn                  = (known after apply)
    + password_data                = (known after apply)
    + placement_group              = (known after apply)
    + primary_network_interface_id = (known after apply)
    + private_dns                  = (known after apply)
    + private_ip                   = (known after apply)
    + public_dns                   = (known after apply)
    + public_ip                    = (known after apply)
    + security_groups              = (known after apply)
    + source_dest_check            = true
    + subnet_id                    = (known after apply)
    + tags                         = {
        + "Name" = "mybastion"
      }
    + tenancy                      = (known after apply)
    + volume_tags                  = (known after apply)
    + vpc_security_group_ids       = (known after apply)

    + ebs_block_device {
        + delete_on_termination = (known after apply)
        + device_name           = (known after apply)
        + encrypted             = (known after apply)
        + iops                  = (known after apply)
        + kms_key_id            = (known after apply)
        + snapshot_id           = (known after apply)
        + volume_id             = (known after apply)
        + volume_size           = (known after apply)
        + volume_type           = (known after apply)
      }
      + ephemeral_block_device {
          + device_name  = (known after apply)
          + no_device    = (known after apply)
          + virtual_name = (known after apply)
        }

      + metadata_options {
          + http_endpoint               = (known after apply)
          + http_put_response_hop_limit = (known after apply)
          + http_tokens                 = (known after apply)
        }

      + network_interface {
          + delete_on_termination = (known after apply)
          + device_index          = (known after apply)
          + network_interface_id  = (known after apply)
        }

      + root_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }
    }

  # aws_instance.mysqlsecure will be created
  + resource "aws_instance" "mysqlsecure" {
      + ami                          = "ami-0b39bdec65a992579"
      + arn                          = (known after apply)
      + associate_public_ip_address  = (known after apply)
      + availability_zone            = "ap-south-1b"
      + cpu_core_count               = (known after apply)
      + cpu_threads_per_core         = (known after apply)
      + get_password_data            = false
      + host_id                      = (known after apply)
      + id                           = (known after apply)
      + instance_state               = (known after apply)
      + instance_type                = "t2.micro"
      + ipv6_address_count           = (known after apply)
      + ipv6_addresses               = (known after apply)
      + key_name                     = "mykey1111.pem"
      + network_interface_id         = (known after apply)
      + outpost_arn                  = (known after apply)
      + password_data                = (known after apply)
      + placement_group              = (known after apply)
      + primary_network_interface_id = (known after apply)
      + private_dns                  = (known after apply)
      + private_ip                   = (known after apply)
      + public_dns                   = (known after apply)
      + public_ip                    = (known after apply)
      + security_groups              = (known after apply)
      + source_dest_check            = true
      + subnet_id                    = (known after apply)
      + tags                         = {
          + "Name" = "mysqlsecure"
        }
      + tenancy                      = (known after apply)
      + volume_tags                  = (known after apply)
      + vpc_security_group_ids       = (known after apply)

      + ebs_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + snapshot_id           = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }

      + ephemeral_block_device {
          + device_name  = (known after apply)
          + no_device    = (known after apply)
          + virtual_name = (known after apply)
        }

      + metadata_options {
          + http_endpoint               = (known after apply)
          + http_put_response_hop_limit = (known after apply)
          + http_tokens                 = (known after apply)
        }

      + network_interface {
          + delete_on_termination = (known after apply)
          + device_index          = (known after apply)
          + network_interface_id  = (known after apply)
        }

      + root_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }
    }

  # aws_instance.myweb will be created
  + resource "aws_instance" "myweb" {
      + ami                          = "ami-0b39bdec65a992579"
      + arn                          = (known after apply)
      + associate_public_ip_address  = (known after apply)
      + availability_zone            = "ap-south-1a"
      + cpu_core_count               = (known after apply)
      + cpu_threads_per_core         = (known after apply)
      + get_password_data            = false
      + host_id                      = (known after apply)
      + id                           = (known after apply)
      + instance_state               = (known after apply)
      + instance_type                = "t2.micro"
      + ipv6_address_count           = (known after apply)
      + ipv6_addresses               = (known after apply)
      + key_name                     = "mykey1111.pem"
      + network_interface_id         = (known after apply)
      + outpost_arn                  = (known after apply)
      + password_data                = (known after apply)
      + placement_group              = (known after apply)
      + primary_network_interface_id = (known after apply)
      + private_dns                  = (known after apply)
      + private_ip                   = (known after apply)
      + public_dns                   = (known after apply)
      + public_ip                    = (known after apply)
      + security_groups              = (known after apply)
      + source_dest_check            = true
      + subnet_id                    = (known after apply)
      + tags                         = {
          + "Name" = "myweb"
        }
      + tenancy                      = (known after apply)
      + volume_tags                  = (known after apply)
      + vpc_security_group_ids       = (known after apply)

      + ebs_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + snapshot_id           = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }

      + ephemeral_block_device {
          + device_name  = (known after apply)
          + no_device    = (known after apply)
          + virtual_name = (known after apply)
        }

      + metadata_options {
          + http_endpoint               = (known after apply)
          + http_put_response_hop_limit = (known after apply)
          + http_tokens                 = (known after apply)
        }

      + network_interface {
          + delete_on_termination = (known after apply)
          + device_index          = (known after apply)
          + network_interface_id  = (known after apply)
        }
      + root_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }
    }

  # aws_internet_gateway.hw_internet_gateway will be created
  + resource "aws_internet_gateway" "hw_internet_gateway" {
      + arn      = (known after apply)
      + id       = (known after apply)
      + owner_id = (known after apply)
      + tags     = {
          + "Name" = "hw_internet_gateway"
        }
      + vpc_id   = (known after apply)
    }

  # aws_route_table.hw_route_table will be created
  + resource "aws_route_table" "hw_route_table" {
      + id               = (known after apply)
      + owner_id         = (known after apply)
      + propagating_vgws = (known after apply)
      + route            = [
          + {
              + cidr_block                = "0.0.0.0/0"
              + egress_only_gateway_id    = ""
              + gateway_id                = (known after apply)
              + instance_id               = ""
              + ipv6_cidr_block           = ""
              + nat_gateway_id            = ""
              + network_interface_id      = ""
              + transit_gateway_id        = ""
              + vpc_peering_connection_id = ""
            },
        ]
      + tags             = {
          + "Name" = "hw_route_table"
        }
      + vpc_id            = (known after apply)
    }

  # aws_route_table_association.a will be created
  + resource "aws_route_table_association" "a" {
      + id             = (known after apply)
      + route_table_id = (known after apply)
      + subnet_id      = (known after apply)
    }

  # aws_security_group.mybastion will be created
  + resource "aws_security_group" "mybastion" {
      + arn                    = (known after apply)
      + description            = "Allow ssh for bastion"
      + egress                 = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + description      = ""
              + from_port        = 0
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "-1"
              + security_groups  = []
              + self             = false
              + to_port          = 0
            },
        ]
      + id                     = (known after apply)
      + ingress                = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + description      = "ssh"
              + from_port        = 0
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "tcp"
              + security_groups  = []
              + self             = false
              + to_port         = 0
            },
        ]
      + name                   = "mybastion"
      + owner_id               = (known after apply)
      + revoke_rules_on_delete = false
      + tags                   = {
          + "Name" = "mybastion"
        }
      + vpc_id                 = (known after apply)
    }

  # aws_security_group.mysql will be created
  + resource "aws_security_group" "mysql" {
      + arn                    = (known after apply)
      + description            = "Allow sql"
      + egress                 = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + description      = ""
              + from_port        = 0
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "-1"
              + security_groups  = []
              + self             = false
              + to_port          = 0
            },
        ]
      + id                     = (known after apply)
      + ingress                = [
          + {
              + cidr_blocks      = []
              + description      = "MYSQL"
              + from_port        = 3306
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "tcp"
              + security_groups  = (known after apply)
              + self             = false
              + to_port          = 3306
          },
        ]
      + name                   = "mysql"
      + owner_id               = (known after apply)
      + revoke_rules_on_delete = false
      + tags                   = {
          + "Name" = "mysql"
        }
      + vpc_id                 = (known after apply)
    }

  # aws_security_group.mysqlallow will be created
  + resource "aws_security_group" "mysqlallow" {
      + arn                    = (known after apply)
      + description            = "ssh allow to the mysql"
      + egress                 = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + description      = ""
              + from_port        = 0
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "-1"
              + security_groups  = []
              + self             = false
              + to_port          = 0
            },
        ]
      + id                     = (known after apply)
      + ingress                = [
          + {
              + cidr_blocks      = []
              + description      = "ssh"
              + from_port        = 22
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "tcp"
              + security_groups  = (known after apply)
              + self             = false
              + to_port          = 22
            },
      ]
    + name                   = "myweb"
    + owner_id               = (known after apply)
    + revoke_rules_on_delete = false
    + tags                   = {
        + "Name" = "mysql"
      }
    + vpc_id                 = (known after apply)
  }

 # aws_security_group.myweb will be created
 + resource "aws_security_group" "myweb" {
    + arn                    = (known after apply)
    + description            = "Allow ssh http and icmp"
    + egress                 = [
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + description      = ""
            + from_port        = 0
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "-1"
            + security_groups  = []
            + self             = false
            + to_port          = 0
          },
      ]
    + id                     = (known after apply)
    + ingress                = [
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + description      = "ICMP-IPv4"
            + from_port        = 0
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "-1"
            + security_groups  = []
            + self             = false
            + to_port          = 0
          },
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + description      = "http"
            + from_port        = 80
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 80
          },
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + description      = "ssh"
            + from_port        = 22
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 22
          },
      ]
    + name                   = "myweb"
    + owner_id               = (known after apply)
    + revoke_rules_on_delete = false
    + tags                   = {
        + "Name" = "myweb"
      }
    + vpc_id                 = (known after apply)
  }

 # aws_subnet.hw_subnet-1a will be created
 + resource "aws_subnet" "hw_subnet-1a" {
    + arn                             = (known after apply)
    + assign_ipv6_address_on_creation = false
    + availability_zone               = "ap-south-1a"
    + availability_zone_id            = (known after apply)
```

```
      + cidr_block                      = "192.168.0.0/24"
      + id                              = (known after apply)
      + ipv6_cidr_block                 = (known after apply)
      + ipv6_cidr_block_association_id  = (known after apply)
      + map_public_ip_on_launch         = true
      + owner_id                        = (known after apply)
      + vpc_id                          = (known after apply)
    }

  # aws_subnet.hw_subnet-1b will be created
  + resource "aws_subnet" "hw_subnet-1b" {
      + arn                             = (known after apply)
      + assign_ipv6_address_on_creation = false
      + availability_zone               = "ap-south-1b"
      + availability_zone_id            = (known after apply)
      + cidr_block                      = "192.168.1.0/24"
      + id                              = (known after apply)
      + ipv6_cidr_block                 = (known after apply)
      + ipv6_cidr_block_association_id  = (known after apply)
      + map_public_ip_on_launch         = false
      + owner_id                        = (known after apply)
      + vpc_id                          = (known after apply)
    }

  # aws_vpc.hw will be created
  + resource "aws_vpc" "hw" {
      + arn                              = (known after apply)
      + assign_generated_ipv6_cidr_block = false
      + cidr_block                       = "192.168.0.0/16"
      + default_network_acl_id           = (known after apply)
      + default_route_table_id           = (known after apply)
      + default_security_group_id        = (known after apply)
      + dhcp_options_id                  = (known after apply)
      + enable_classiclink               = (known after apply)
      + enable_classiclink_dns_support   = (known after apply)
      + enable_dns_hostnames             = (known after apply)
      + enable_dns_support               = true
      + id                               = (known after apply)
      + instance_tenancy                 = "default"
      + ipv6_association_id              = (known after apply)
      + ipv6_cidr_block                  = (known after apply)
      + main_route_table_id              = (known after apply)
      + owner_id                         = (known after apply)
      + tags                          = {
          + "Name" = "hw"
        }
    }

Plan: 13 to add, 0 to change, 0 to destroy.


Warning: Interpolation-only expressions are deprecated

  on vpc.tf line 16, in resource "aws_subnet" "hw_subnet-1a":
  16:   vpc_id     = "${aws_vpc.hw.id}"

Terraform 0.11 and earlier required all non-constant expressions to be
provided via interpolation syntax, but this pattern is now deprecated. To
silence this warning, remove the "${ sequence from the start and the }"
sequence from the end of this expression, leaving just the inner expression.

Template interpolation syntax is still used to construct strings from
expressions when the template includes multiple interpolation sequences or a
mixture of literal strings and interpolations. This deprecation applies only
to templates that consist entirely of a single interpolation sequence.

(and 12 more similar warnings elsewhere)

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
```



```
Microsoft Windows [Version 10.0.18362.900]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\user>cd Desktop

C:\Users\user\Desktop>cd cloud

C:\Users\user\Desktop\cloud>ssh -i mykey1111.pem -l ec2-user 13.233.84.104
ssh: connect to host 13.233.84.104 port 22: Connection timed out

C:\Users\user\Desktop\cloud>ssh -i mykey1111.pem -l ec2-user 13.233.84.104
The authenticity of host '13.233.84.104 (13.233.84.104)' can't be established.
ECDSA key fingerprint is SHA256:fITCw7RWIHcA8a7ugaxgqVIHK0V28FLkhFLS2xIgQjg.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '13.233.84.104' (ECDSA) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
9 package(s) needed for security, out of 16 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-192-168-0-163 ~]$ ping 192.168.1.224
PING 192.168.1.224 (192.168.1.224) 56(84) bytes of data.
^C
--- 192.168.1.224 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2046ms

[ec2-user@ip-192-168-0-163 ~]$
```

	1.compute.internal		
Private IPs	192.168.1.224	Security groups	mysqlallow, mysql. view inbound rules. view outbound rules
Secondary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-02973363e595f38c6 (hw)	AMI ID	my-php-image (ami-0b39bdec65a992579)


Advanced Site Settings

Environment
- Directories
- Recycle bin
- Encryption
- SFTP
- Shell

Connection
- Proxy
- Tunnel

SSH
- Key exchange
- Authentication
- Bugs

Note

Server environment

End-of-line characters (if not indicated by server): LF

UTF-8 encoding for filenames: Auto

Time zone offset: 0 hours 0 minutes

[✓] Detect automatically

[ ] Trim VMS version numbers

Daylight saving time

(●) Adjust remote timestamp to local conventions

( ) Adjust remote timestamp with DST

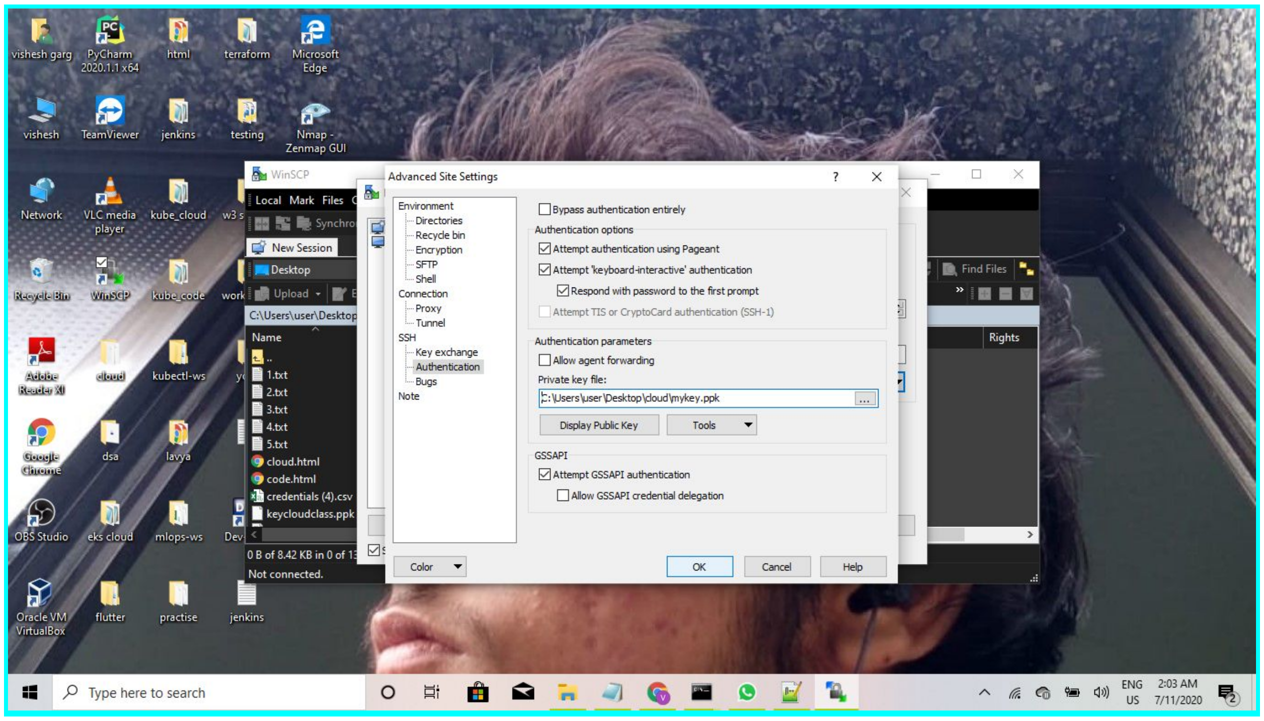
( ) Preserve remote timestamp

PuTTY

PuTTY terminal settings:

Edit in PuTTY...

Color | OK | Cancel | Help


Advanced Site Settings

Environment
- Directories
- Recycle bin
- Encryption
- SFTP
- Shell

Connection
- Proxy
- Tunnel

SSH
- Key exchange
- Authentication
- Bugs

Note

[ ] Bypass authentication entirely

Authentication options

[✓] Attempt authentication using Pageant

[✓] Attempt 'keyboard-interactive' authentication

[✓] Respond with password to the first prompt

[ ] Attempt TIS or CryptoCard authentication (SSH-1)

Authentication parameters

[ ] Allow agent forwarding

Private key file:

C:\Users\user\Desktop\cloud\mykey.ppk

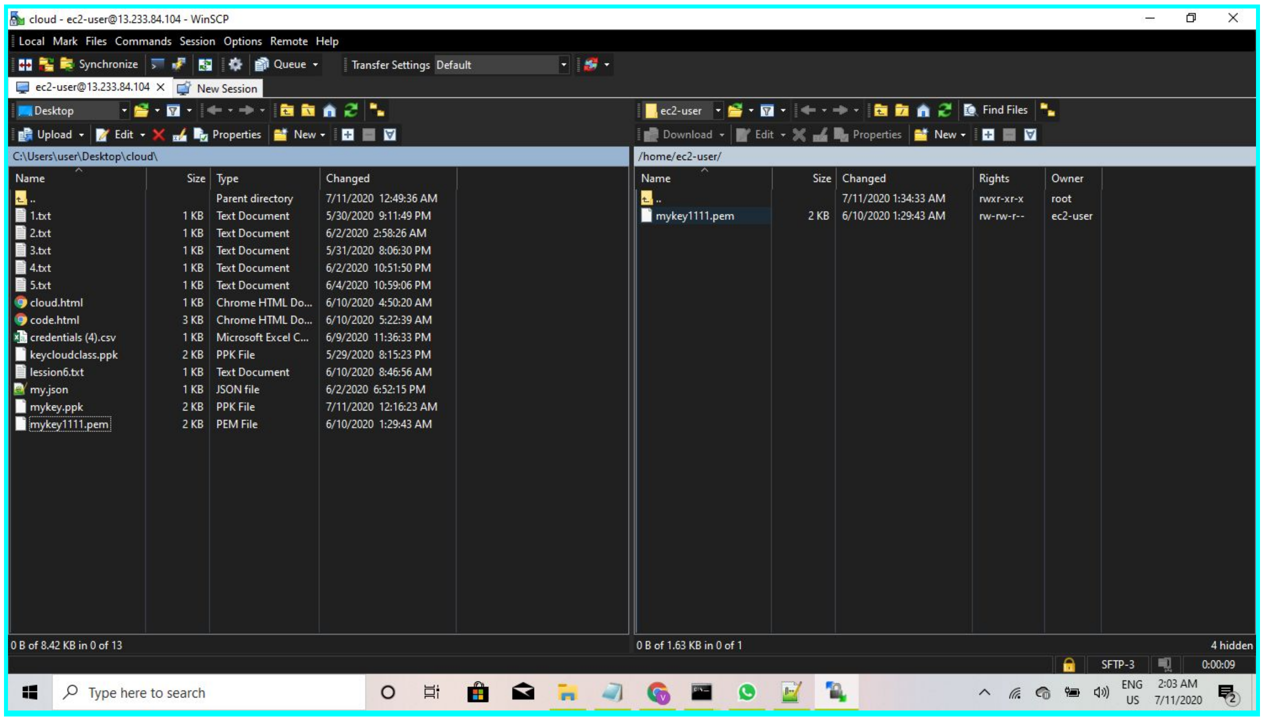
Display Public Key | Tools

GSSAPI

[✓] Attempt GSSAPI authentication

[ ] Allow GSSAPI credential delegation

Color | OK | Cancel | Help


cloud - ec2-user@13.233.84.104 - WinSCP

C:\Users\user\Desktop\cloud\

Name	Size	Type	Changed
..		Parent directory	7/11/2020 12:49:36 AM
1.txt	1 KB	Text Document	5/30/2020 9:11:49 PM
2.txt	1 KB	Text Document	6/2/2020 2:58:26 AM
3.txt	1 KB	Text Document	5/31/2020 8:06:30 PM
4.txt	1 KB	Text Document	6/2/2020 10:51:50 PM
5.txt	1 KB	Text Document	6/4/2020 10:59:06 PM
cloud.html	1 KB	Chrome HTML Do...	6/10/2020 4:50:20 AM
code.html	3 KB	Chrome HTML Do...	6/10/2020 5:22:39 AM
credentials (4).csv	1 KB	Microsoft Excel C...	6/9/2020 11:36:33 PM
keycloudclass.ppk	2 KB	PPK File	5/29/2020 8:15:23 PM
lession6.txt	1 KB	Text Document	6/10/2020 8:46:56 AM
my.json	1 KB	JSON file	6/2/2020 6:52:15 PM
mykey.ppk	2 KB	PPK File	7/11/2020 12:16:23 AM
mykey1111.pem	2 KB	PEM File	6/10/2020 1:29:43 AM

0 B of 8.42 KB in 0 of 13

/home/ec2-user/

Name	Size	Changed	Rights	Owner
..		7/11/2020 1:34:33 AM	rwxr-xr-x	root
mykey1111.pem	2 KB	6/10/2020 1:29:43 AM	rw-rw-r--	ec2-user

0 B of 1.63 KB in 0 of 1 · 4 hidden

SFTP-3 · 0:00:09

```
C:\Users\user>cd Desktop

C:\Users\user\Desktop>cd cloud
C:\Users\user\Desktop\cloud>ssh -i mykey1111.pem -l ec2-user 13.233.84.104
The authenticity of host '13.233.84.104 (13.233.84.104)' can't be established.
ECDSA key fingerprint is SHA256:fITCw7RWIHcA8a7ugaxgqVIHK0V28FLkhFLS2xIgQjg.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '13.233.84.104' (ECDSA) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
9 package(s) needed for security, out of 16 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-192-168-0-163 ~]$ ping 192.168.1.224
PING 192.168.1.224 (192.168.1.224) 56(84) bytes of data.
^C
--- 192.168.1.224 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2046ms

[ec2-user@ip-192-168-0-163 ~]$ ls
mykey1111.pem
[ec2-user@ip-192-168-0-163 ~]$ ls -l
total 4
-rw-rw-r-- 1 ec2-user ec2-user 1670 Jun  9 19:59 mykey1111.pem
[ec2-user@ip-192-168-0-163 ~]$ chmod 400 mykey1111.pem
```

```
[ec2-user@ip-192-168-0-163 ~]$ ssh -i mykey1111.pem -l ec2-user 192.168.1.224
The authenticity of host '192.168.1.224 (192.168.1.224)' can't be established.
ECDSA key fingerprint is SHA256:skHxB6uNTPNMPqlk2wl2qa6P9+H17dsytjISM3S/GyU.
ECDSA key fingerprint is MD5:1f:06:ca:75:29:22:0e:cb:f6:9a:a9:38:16:08:a4:13.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.224' (ECDSA) to the list of known hosts.
Last login: Fri Jul 10 17:06:51 2020 from ec2-13-233-177-1.ap-south-1.compute.amazonaws.com

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-192-168-1-224 ~]$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9001
        inet 192.168.1.224  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::8c4:cdff:fe2f:88cc  prefixlen 64  scopeid 0x20<link>
        ether 0a:c4:cd:2f:88:cc  txqueuelen 1000  (Ethernet)
        RX packets 615  bytes 69049 (67.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 854  bytes 87692 (85.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```