Launch VPC Wizard with Public and Private Subnet for WordPress with Bostion Host on AWS using Terraform

This task is almost same with last task with an additional feature to be added that is NAT Gateway to provide the internet access to instances running in the private subnet. I am using Terraform CLI for launch this whole setup on the top of AWS.

Task Overview:

Performing the following steps:

- Write an Infrastructure as code using terraform, which automatically create a VPC.
- In that VPC we have to create 2 subnets:

1. public subnet [Accessible for Public World!]
2. private subnet [Restricted for Public World!]

- Create a public-facing internet gateway to connect our VPC/Network to the internet world and attach this gateway to our VPC.
- Create a routing table for Internet gateway so that instance can connect to the outside world, update, and associate it with the public subnet.
- Create a NAT gateway to connect our VPC/Network to the internet world and attach this gateway to our VPC in the public network
- Update the routing table of the private subnet, so that to access the internet it uses the Nat gateway created in the public subnet
- Launch an ec2 instance that 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 the instance for further login into it.
- Launch an ec2 instance that has MYSQL setup already with security group allowing port 3306 in a private subnet so that our WordPress VM can connect with the same. Also, attach the key with the same.

Note:

- WordPress instance must be part of the public subnet so that our client can connect our site.
- MySQL instance must be part of a private subnet so that the outside world can't connect to it.
- Don't forget to add auto IP assign and auto DNS name assignment option to be enabled.

Task Description:

I create a file having .tf extension with all the code. I'm discussing the code part by part here.

1. Creating a VPC:

For creating VPC, Code is:

```
resource "aws_vpc" "wpvpc" {
  cidr_block = "10.7.0.0/16"
  enable_dns_hostnames = true
  tags = {
    Name = "main"
  }
}
```

This create one VPC with enabling DNS hostname.

2. Create one Public Subnet:

For creating a Public Subnet, you need to create one Internet gateway and also one routing table.

```
resource "aws_subnet" "alpha-1a" {
  vpc_id            = "${aws_vpc.wpvpc.id}"
  availability_zone = "ap-south-1a"
  cidr_block        = "10.7.1.0/24"
  map_public_ip_on_launch = true
  tags = {
    Name = "main-1a"
  }
}


resource "aws_internet_gateway" "gw" {
  vpc_id = "${aws_vpc.wpvpc.id}"
  tags = {
    Name = "main-1a"
  }}
resource "aws_route_table" "rt" {
  vpc_id = "${aws_vpc.wpvpc.id}"
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_internet_gateway.gw.id}"
  }
  tags = {
    Name = "main-1a"
  }
}
```

Now your Public Subnet is launched and one routing Table also. But for attaching the routing table to Subnet, we need to write some more code...

```
resource "aws_route_table_association" "ra" {
  subnet_id      = aws_subnet.alpha-1a.id
  route_table_id = aws_route_table.rt.id
}
```

Now my Public Subnet is created for connecting with the Internet.

3. Create one Public Subnet:

This part is also same as the previous step, yet we do not want to connect with the outer world as we make it private so don't create any Internet Gateway.

```
resource "aws_subnet" "alpha-1b" {
  vpc_id            = "${aws_vpc.myvpc.id}"
  availability_zone = "ap-south-1b"
  cidr_block        = "10.7.2.0/24"
  tags = {
    Name = "main-1b"
  }
}
```

Now as our need, one more thing we want that our private subnet can connect to internet for some important update but No one can connect to this for security reason. For this, we have one concept of **Source Network Address Translation** in networking. AWS has one subservice inside VPC called **NAT Gateway** for this.

Code for launch NAT gateway:

```
resource "aws_eip" "lb" {
  vpc      = true
}


resource "aws_nat_gateway" "gw" {
  allocation_id = "${aws_eip.lb.id}"
  subnet_id     = "${aws_subnet.alpha-1a.id}"
  depends_on    = [ "aws_internet_gateway.gw" ]
}
```

You need one EIP too for creating NAT Gateway. Make sure you make it in your Public Subnet either you do not connect to Internet.

Now for connecting this to our private subnet, we need to create one routing Table and associate it with our Private Subnet.

```
resource "aws_route_table" "nat-table" {
  vpc_id = "${aws_vpc.myvpc.id}"
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_nat_gateway.gw.id}"
  }
  tags = {
    Name = "main-1b"
  }
}
resource "aws_route_table_association" "nat-b" {
  subnet_id      = aws_subnet.alpha-1b.id
```

```
  route_table_id = aws_route_table.nat-table.id
}
```

Now my Private Subnet is launched and also, we can access Internet from this.

4. Launch one EC2 instance for WordPress with allowing http port on security Group:

For doing SSH, we need to provide one Key to it, for creating a Key...

```
resource "tls_private_key" "key_create" {
  algorithm = "RSA"
}


resource "aws_key_pair" "taskkey" {
  key_name   = "taskkey"
  public_key = "${tls_private_key.key_create.public_key_openssh}"
}
output "key-pair" {
  value = tls_private_key.key_create.private_key_pem
}
resource "local_file" "save_key" {
    content     = tls_private_key.key_create.private_key_pem
    filename = "taskkey.pem"
}
```

Now next step is creating one Security Group allowing SSH, Https and Http port...

```
resource "aws_security_group" "allow_http_wordpress" {
  name        = "allow_http_wordpress"
  description = "Allow HTTP inbound traffic"
  vpc_id      = "${aws_vpc.wpvpc.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 = "wpsgroup"
  }
}
```

For launching your wordpress instance, code is...

```
variable "enter_ur_key_name" {
               type = string
        default = "taskkey"
}
resource "aws_instance" "wordpress" {
  ami           = "ami-0979674e4a8c6ea0c"
  instance_type = "t2.micro"
  key_name      = var.enter_ur_key_name
  availability_zone = "ap-south-1a"
  subnet_id     = "${aws_subnet.alpha-1a.id}"
  security_groups = [ "${aws_security_group.allow_http_wordpress.id}" ]
  tags = {
    Name = "Wordpress"
  }
}
```

Now my WordPress is launched.

Now let me explain a little more *why we need one NAT Gateway*?

As MySQL instance is a part of our private subnet so that no one from outside world connect/hack our database. But we also need to do update the software. For this usecase, we need a NAT Gateway.

So for going inside the instance, we need to attach a key and allow SSH to the security group but it is not good to provide permission to everyone. Here concept of **Bastion Host** comes up... Using this OS, you can do SSH only to go inside the MySQL instance. Now come to the next step...

5. Launch one EC2 instance to launch Bastion Host:

We launched this instance in the public Subnet and create one security group for allowing SSH.

```
resource "aws_security_group" "bostion-sg" {
  name        = "bostion-sg"
  description = "SSH to bostion-host"
  vpc_id      = "${aws_vpc.wpvpc.id}"


  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"
  }
}


resource "aws_instance" "bostion-host" {
  ami           = "ami-00b494a3f139ba61f"
  instance_type = "t2.micro"
  key_name      = var.enter_ur_key_name
  availability_zone = "ap-south-1a"
  subnet_id     = "${aws_subnet.alpha-1a.id}"
  vpc_security_group_ids = [ "${aws_security_group.bostion-sg.id}" ]
  tags = {
    Name = "bostion-host"
  }
}
```

6. Launch one EC2 instance for MySQL Database:

For this, we need to provide a key for login inside the instance but also we make one security group to restrict the access through SSH by giving security group as source we create for Bostion Host and also allow MYSQL port.

```
resource "aws_security_group" "mysql-sg" {
  name        = "for-mysql"
  description = "MYSQL-setup"
  vpc_id      = "${aws_vpc.wpvpc.id}"


  ingress {
    description = "MYSQL 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"
    security_groups = [ "${aws_security_group.bostion-sg.id}" ]
  }
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }


  tags = {
    Name = "sgroup"
  }
}

resource "aws_instance" "mysql" {
  ami           = "ami-76166b19"
  instance_type = "t2.micro"
  key_name      = var.enter_ur_key_name
  availability_zone = "ap-south-1b"
  subnet_id     = "${aws_subnet.alpha-1b.id}"
  security_groups = [ "${aws_security_group.mysql-sg.id}" ]
  tags = {
    Name = "MYSQL"
  }
}
```

Now, MySQL database is launched.

Now my setup is completed.

For run the file, open your command prompt, and run these two commands:

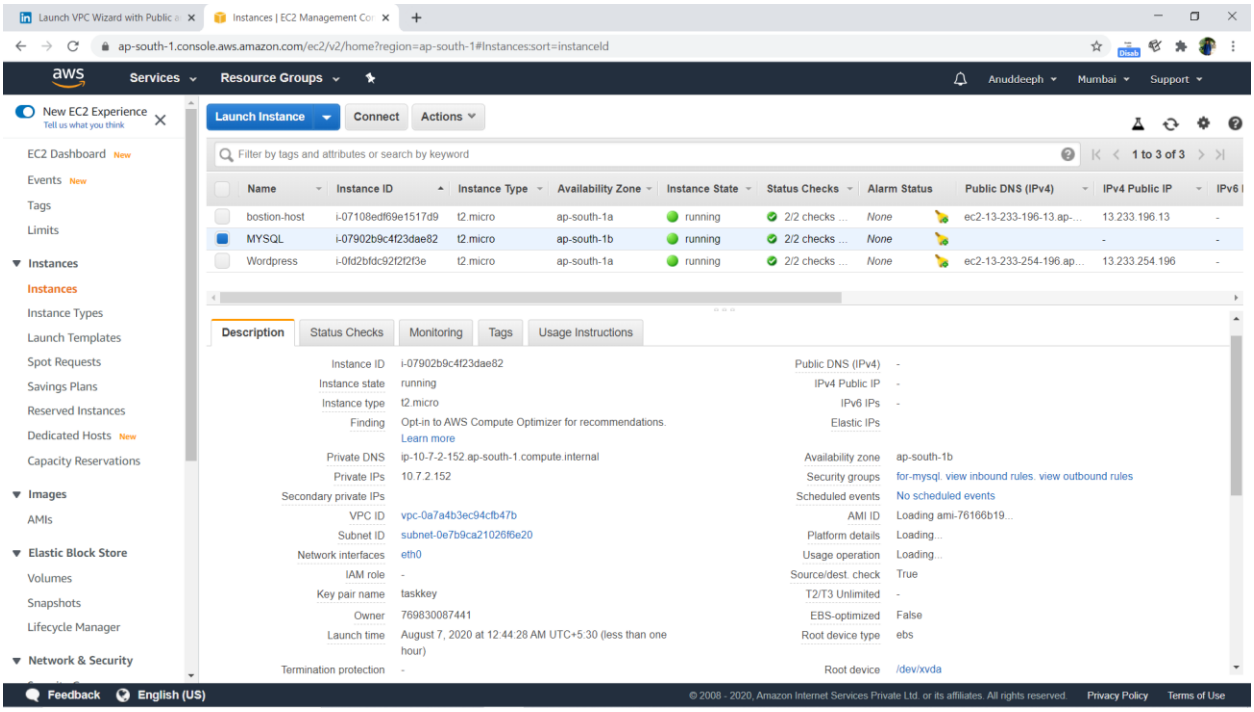
- terraform init
- terraform apply

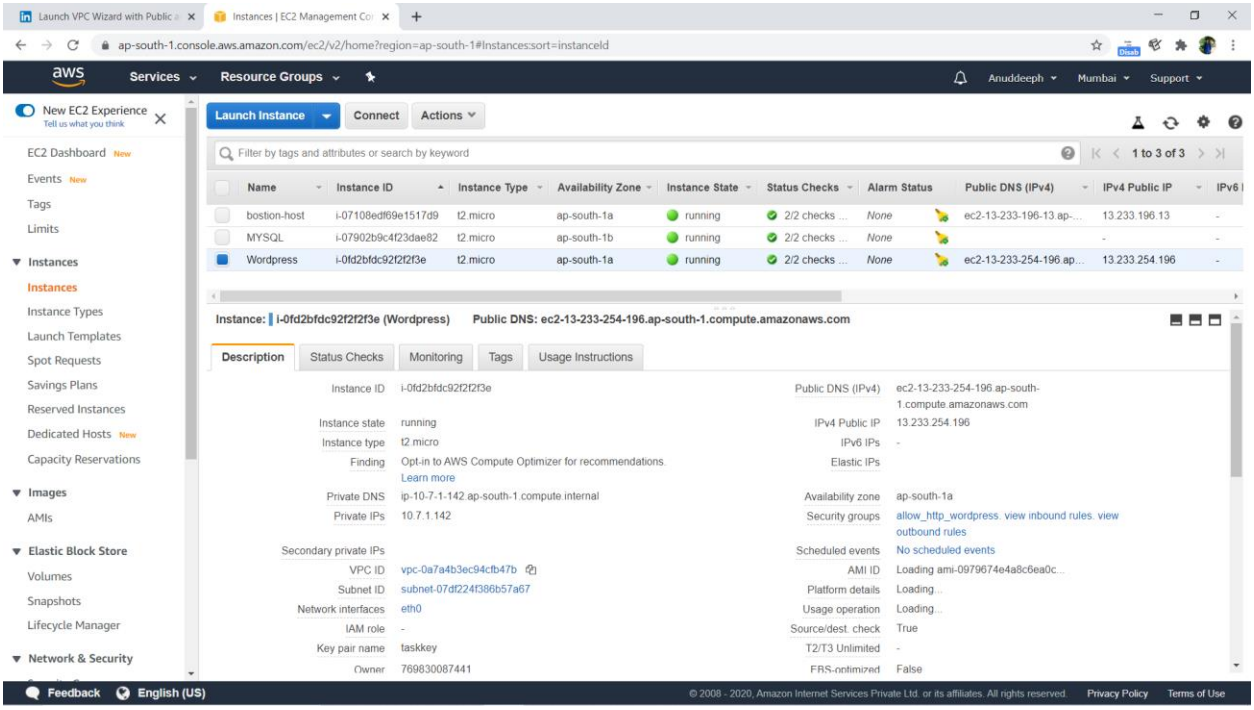
Command Prompt

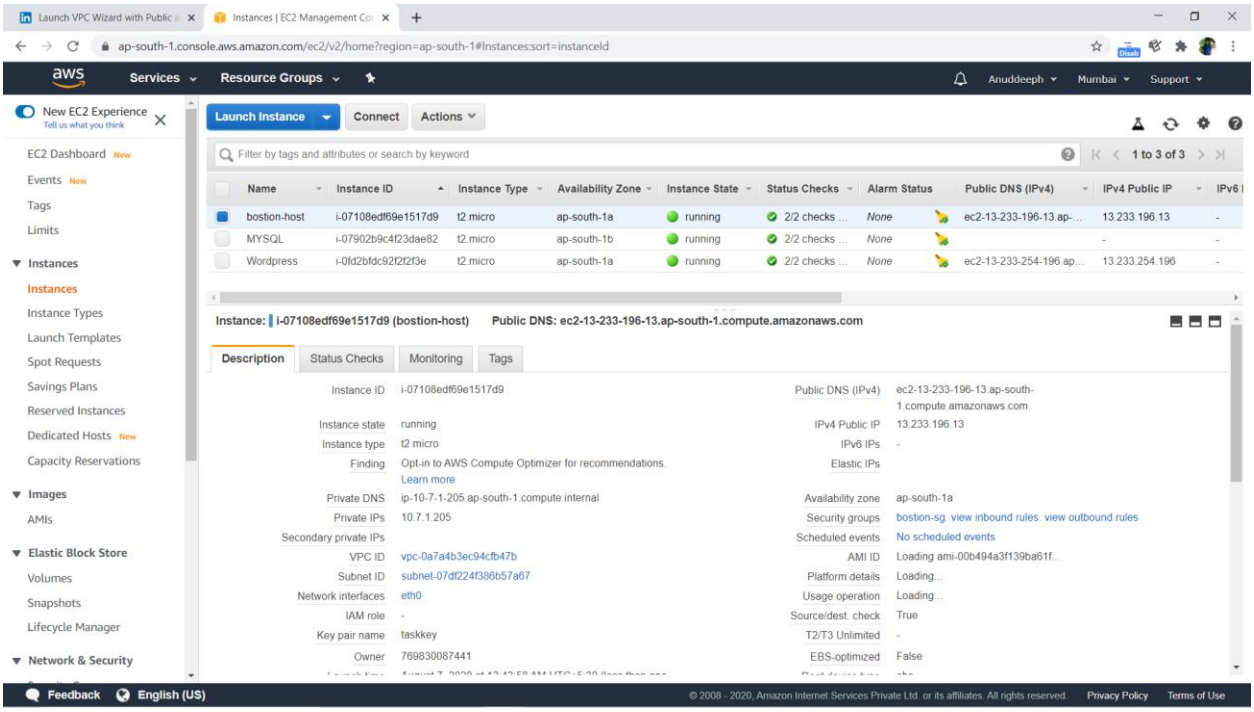
```
C:\Users\Anuddeeph Nalla\Desktop\hybrid\tera\aws-task4>terraform apply

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_eip.lb will be created
  + resource "aws_eip" "lb" {
      + allocation_id     = (known after apply)
      + association_id    = (known after apply)
      + customer_owned_ip = (known after apply)
      + domain            = (known after apply)
      + id                = (known after apply)
      + instance          = (known after apply)
      + network_interface = (known after apply)
      + private_dns       = (known after apply)
      + private_ip        = (known after apply)
      + public_dns        = (known after apply)
      + public_ip         = (known after apply)
      + public_ipv4_pool  = (known after apply)
      + vpc               = true
    }

  # aws_instance.bostion-host will be created
  + resource "aws_instance" "bostion-host" {
      + ami                          = "ami-00b494a3f139ba61f"
      + 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                     = "taskkey"
      + 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)
      + secondary_private_ips        = (known after apply)
      + security_groups              = (known after apply)
```

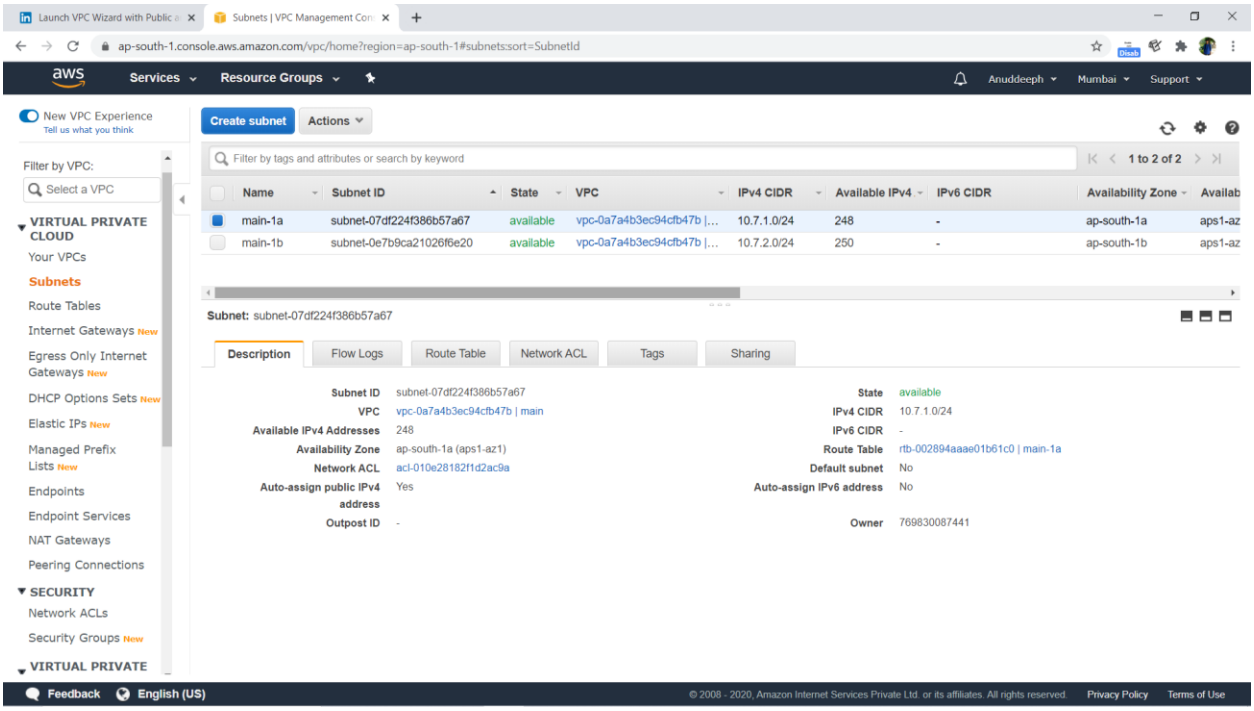
Command Prompt

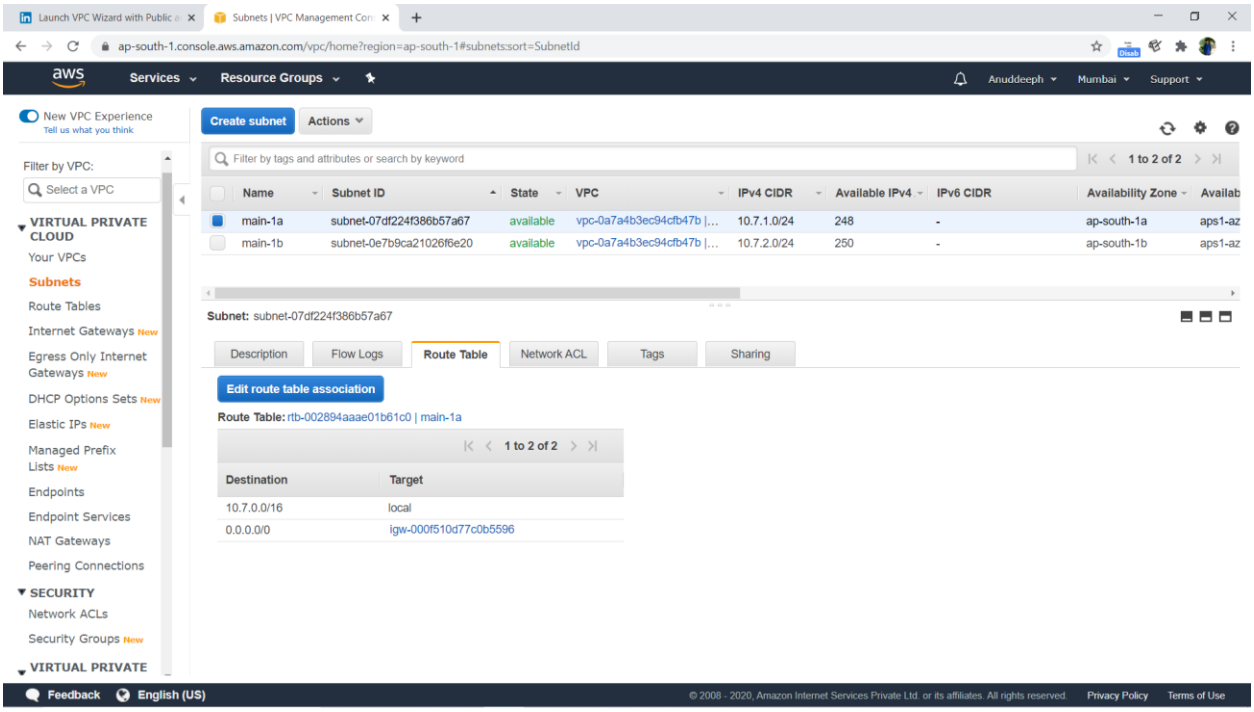
```
aws_security_group.bostion-sg: Creation complete after 29s [id=sg-087d95cec55c8db9c]
aws_instance.bostion-host: Creating...
aws_security_group.allow_http_wordpress: Still creating... [30s elapsed]
aws_security_group.allow_http_wordpress: Creation complete after 30s [id=sg-07160933ea5d5ca2e]
aws_nat_gateway.gw: Still creating... [10s elapsed]
aws_route_table.rt: Still creating... [10s elapsed]
aws_security_group.mysql-sg: Creating...
aws_instance.wordpress: Creating...
aws_route_table.rt: Creation complete after 13s [id=rtb-002894aaae01b61c0]
aws_route_table_association.rta: Creating...
aws_route_table_association.rta: Creation complete after 3s [id=rtbassoc-01dbd6b91bcc9cc78]
aws_instance.bostion-host: Still creating... [10s elapsed]
aws_nat_gateway.gw: Still creating... [20s elapsed]
aws_security_group.mysql-sg: Still creating... [10s elapsed]
aws_instance.wordpress: Still creating... [10s elapsed]
aws_instance.bostion-host: Still creating... [20s elapsed]
aws_nat_gateway.gw: Still creating... [30s elapsed]
aws_security_group.mysql-sg: Still creating... [20s elapsed]
aws_instance.wordpress: Still creating... [20s elapsed]
aws_security_group.mysql-sg: Creation complete after 25s [id=sg-01a07e732d979eaf3]
aws_instance.mysql: Creating...
aws_instance.bostion-host: Still creating... [30s elapsed]
aws_nat_gateway.gw: Still creating... [40s elapsed]
aws_instance.wordpress: Still creating... [30s elapsed]
aws_instance.mysql: Still creating... [10s elapsed]
aws_instance.bostion-host: Still creating... [40s elapsed]
aws_nat_gateway.gw: Still creating... [50s elapsed]
aws_instance.wordpress: Still creating... [40s elapsed]
aws_instance.bostion-host: Creation complete after 42s [id=i-07108edf69e1517d9]
aws_instance.wordpress: Creation complete after 41s [id=i-0fd2bfdc92f2f2f3e]
aws_instance.mysql: Still creating... [20s elapsed]
aws_nat_gateway.gw: Still creating... [1m0s elapsed]
aws_instance.mysql: Still creating... [30s elapsed]
aws_instance.mysql: Creation complete after 33s [id=i-07902b9c4f23dae82]
aws_nat_gateway.gw: Still creating... [1m10s elapsed]
aws_nat_gateway.gw: Still creating... [1m20s elapsed]
aws_nat_gateway.gw: Still creating... [1m30s elapsed]
aws_nat_gateway.gw: Creation complete after 1m34s [id=nat-020df89a5bf8e9b32]
aws_route_table.nat-table: Creating...
aws_route_table.nat-table: Still creating... [10s elapsed]
aws_route_table.nat-table: Creation complete after 13s [id=rtb-0761c32b9c6d81409]
aws_route_table_association.nat-b: Creating...
aws_route_table_association.nat-b: Still creating... [10s elapsed]
aws_route_table_association.nat-b: Still creating... [20s elapsed]
aws_route_table_association.nat-b: Creation complete after 28s [id=rtbassoc-0bafdce5b5082d5da]

Apply complete! Resources: 19 added, 0 changed, 0 destroyed.

Outputs:

myoutaz1 = ap-south-1a
myoutip1 = 13.233.254.196
```

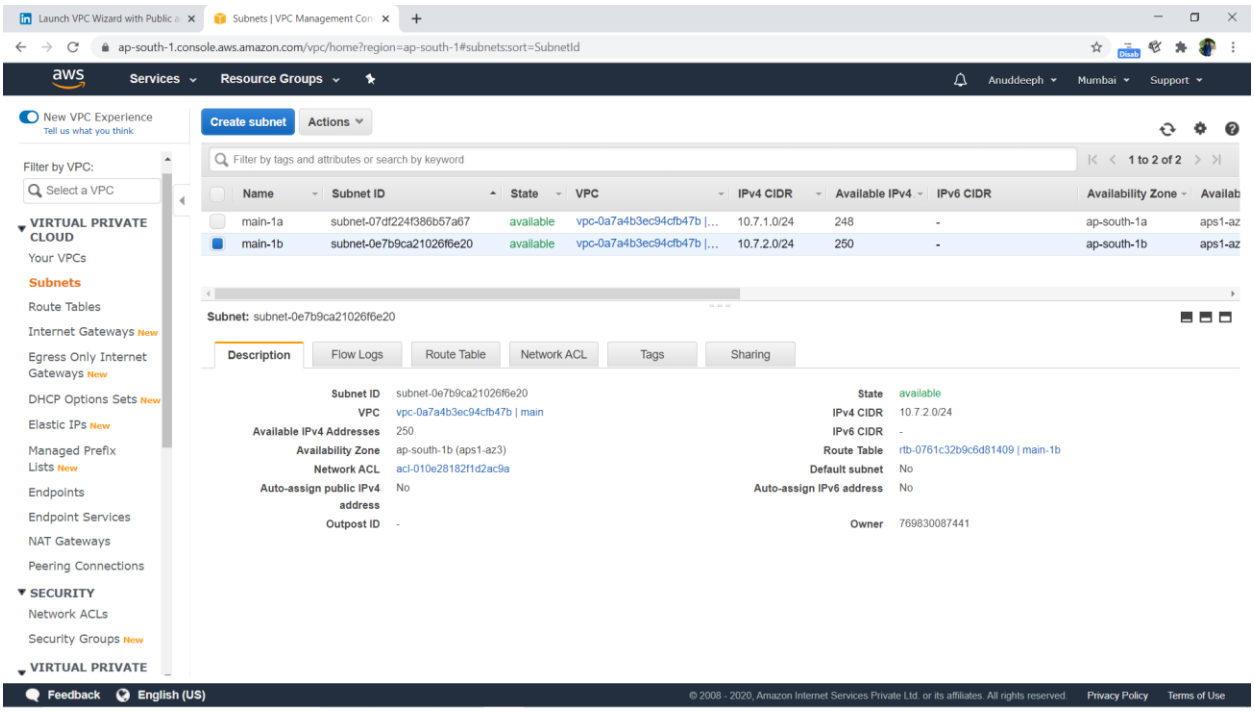

Launch VPC Wizard with Public a
Instances | EC2 Management Co
ap-south-1.console.aws.amazon.com/ec2/v2/home?region=ap-south-1#Instances:sort=instanceId
aws
Services
Resource Groups
Anuddeeph
Mumbai
Support
New EC2 Experience
Tell us what you think
EC2 Dashboard New
Events New
Tags
Limits
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts New
Capacity Reservations
Images
AMIs
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Launch Instance
Connect
Actions
Filter by tags and attributes or search by keyword
1 to 3 of 3
Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS (IPv4)
IPv4 Public IP
IPv6
bostion-host i-07108edf69e1517d9 t2.micro ap-south-1a running 2/2 checks ... None ec2-13-233-196-13.ap-... 13.233.196.13
MYSQL i-07902b9c4f23dae82 t2.micro ap-south-1b running 2/2 checks ... None
Wordpress i-0fd2bfdc92f2f2f3e t2.micro ap-south-1a running 2/2 checks ... None ec2-13-233-254-196.ap... 13.233.254.196
Description
Status Checks
Monitoring
Tags
Usage Instructions
Instance ID i-07902b9c4f23dae82
Instance state running
Instance type t2.micro
Finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
Private DNS ip-10-7-2-152.ap-south-1.compute.internal
Private IPs 10.7.2.152
Secondary private IPs
VPC ID vpc-0a7a4b3ec94cfb47b
Subnet ID subnet-0e7b9ca21026f6e20
Network interfaces eth0
IAM role -
Key pair name taskkey
Owner 769830087441
Launch time August 7, 2020 at 12:44:28 AM UTC+5:30 (less than one hour)
Termination protection -
Public DNS (IPv4) -
IPv4 Public IP -
IPv6 IPs -
Elastic IPs
Availability zone ap-south-1b
Security groups for-mysql. view inbound rules. view outbound rules
Scheduled events No scheduled events
AMI ID Loading ami-76166b19...
Platform details Loading...
Usage operation Loading...
Source/dest. check True
T2/T3 Unlimited -
EBS-optimized False
Root device type ebs
Root device /dev/xvda
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

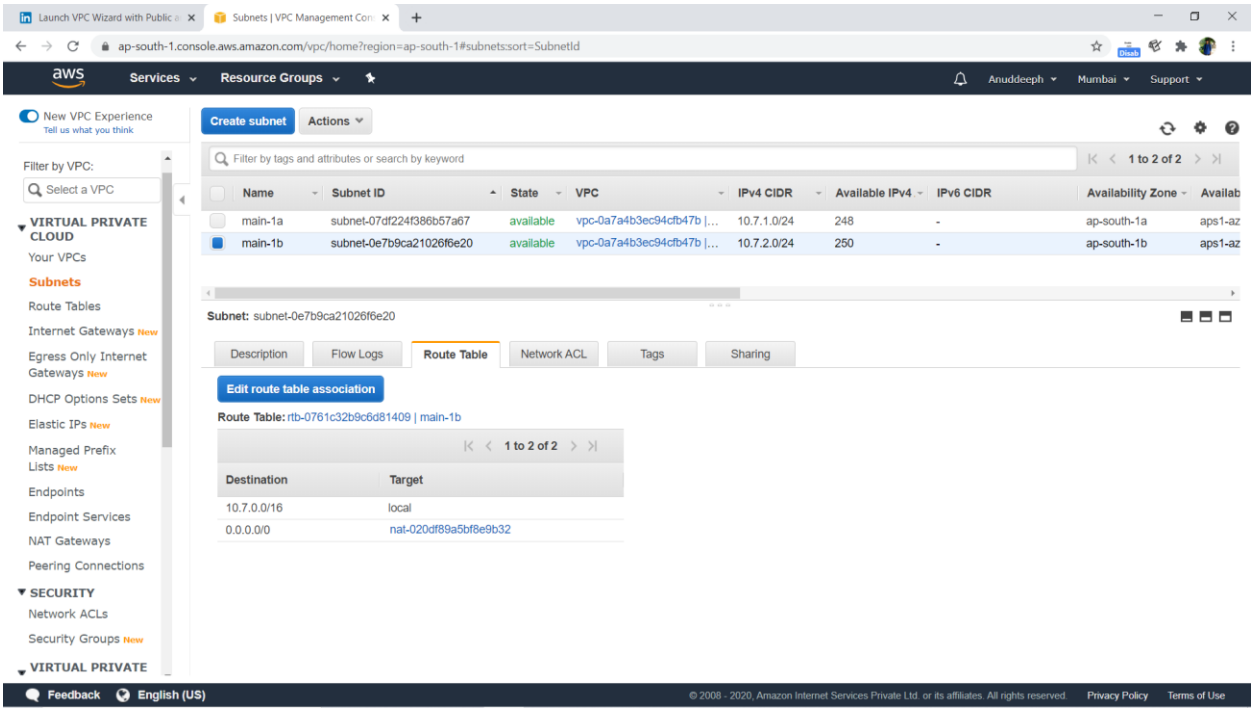

Launch VPC Wizard with Public a
Instances | EC2 Management Co
ap-south-1.console.aws.amazon.com/ec2/v2/home?region=ap-south-1#Instances:sort=instanceId
aws
Services
Resource Groups
Anuddeeph
Mumbai
Support
New EC2 Experience
Tell us what you think
EC2 Dashboard New
Events New
Tags
Limits
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts New
Capacity Reservations
Images
AMIs
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Launch Instance
Connect
Actions
Filter by tags and attributes or search by keyword
1 to 3 of 3
Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS (IPv4)
IPv4 Public IP
IPv6
bostion-host i-07108edf69e1517d9 t2.micro ap-south-1a running 2/2 checks ... None ec2-13-233-196-13.ap-... 13.233.196.13
MYSQL i-07902b9c4f23dae82 t2.micro ap-south-1b running 2/2 checks ... None
Wordpress i-0fd2bfdc92f2f2f3e t2.micro ap-south-1a running 2/2 checks ... None ec2-13-233-254-196.ap... 13.233.254.196
Instance: i-0fd2bfdc92f2f2f3e (Wordpress) Public DNS: ec2-13-233-254-196.ap-south-1.compute.amazonaws.com
Description
Status Checks
Monitoring
Tags
Usage Instructions
Instance ID i-0fd2bfdc92f2f2f3e
Instance state running
Instance type t2.micro
Finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
Private DNS ip-10-7-1-142.ap-south-1.compute.internal
Private IPs 10.7.1.142
Secondary private IPs
VPC ID vpc-0a7a4b3ec94cfb47b
Subnet ID subnet-07df224f386b57a67
Network interfaces eth0
IAM role -
Key pair name taskkey
Owner 769830087441
Public DNS (IPv4) ec2-13-233-254-196.ap-south-1.compute.amazonaws.com
IPv4 Public IP 13.233.254.196
IPv6 IPs -
Elastic IPs
Availability zone ap-south-1a
Security groups allow_http_wordpress. view inbound rules. view outbound rules
Scheduled events No scheduled events
AMI ID Loading ami-0979674e4a8c6ea0c...
Platform details Loading...
Usage operation Loading...
Source/dest. check True
T2/T3 Unlimited -
EBS-optimized False
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


Launch VPC Wizard with Public
Instances | EC2 Management Co
ap-south-1.console.aws.amazon.com/ec2/v2/home?region=ap-south-1#Instances:sort=instanceId
Services
Resource Groups
Anuddeeph
Mumbai
Support
New EC2 Experience
Tell us what you think
Launch Instance
Connect
Actions
EC2 Dashboard New
Events New
Tags
Limits
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts New
Capacity Reservations
Images
AMIs
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Filter by tags and attributes or search by keyword
1 to 3 of 3
Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS (IPv4)
IPv4 Public IP
bostion-host
i-07108edf69e1517d9
t2.micro
ap-south-1a
running
2/2 checks ...
None
ec2-13-233-196-13.ap-...
13.233.196.13
MYSQL
i-07902b9c4f23dae82
t2.micro
ap-south-1b
running
2/2 checks ...
None
Wordpress
i-0fd2bfdc92f2f2f3e
t2.micro
ap-south-1a
running
2/2 checks ...
None
ec2-13-233-254-196.ap...
13.233.254.196
Instance: i-07108edf69e1517d9 (bostion-host)
Public DNS: ec2-13-233-196-13.ap-south-1.compute.amazonaws.com
Description
Status Checks
Monitoring
Tags
Instance ID
i-07108edf69e1517d9
Instance state
running
Instance type
t2.micro
Finding
Opt-in to AWS Compute Optimizer for recommendations.
Learn more
Private DNS
ip-10-7-1-205.ap-south-1.compute.internal
Private IPs
10.7.1.205
Secondary private IPs
VPC ID
vpc-0a7a4b3ec94cfb47b
Subnet ID
subnet-07df224f386b57a67
Network interfaces
eth0
IAM role
Key pair name
taskkey
Owner
769830087441
Public DNS (IPv4)
ec2-13-233-196-13.ap-south-1.compute.amazonaws.com
IPv4 Public IP
13.233.196.13
IPv6 IPs
Elastic IPs
Availability zone
ap-south-1a
Security groups
bostion-sg. view inbound rules. view outbound rules
Scheduled events
No scheduled events
AMI ID
Loading ami-00b494a3f139ba61f...
Platform details
Loading...
Usage operation
Loading...
Source/dest. check
True
T2/T3 Unlimited
EBS-optimized
False
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


Launch VPC Wizard with Public
Subnets | VPC Management Con
ap-south-1.console.aws.amazon.com/vpc/home?region=ap-south-1#subnets:sort=SubnetId
Services
Resource Groups
Anuddeeph
Mumbai
Support
New VPC Experience
Tell us what you think
Create subnet
Actions
Filter by VPC:
Select a VPC
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
Endpoints
Endpoint Services
NAT Gateways
Peering Connections
SECURITY
Network ACLs
Security Groups New
VIRTUAL PRIVATE
Filter by tags and attributes or search by keyword
1 to 2 of 2
Name
Subnet ID
State
VPC
IPv4 CIDR
Available IPv4
IPv6 CIDR
Availability Zone
main-1a
subnet-07df224f386b57a67
available
vpc-0a7a4b3ec94cfb47b |...
10.7.1.0/24
248
-
ap-south-1a
aps1-az
main-1b
subnet-0e7b9ca21026f6e20
available
vpc-0a7a4b3ec94cfb47b |...
10.7.2.0/24
250
-
ap-south-1b
aps1-az
Subnet: subnet-07df224f386b57a67
Description
Flow Logs
Route Table
Network ACL
Tags
Sharing
Subnet ID
subnet-07df224f386b57a67
VPC
vpc-0a7a4b3ec94cfb47b | main
Available IPv4 Addresses
248
Availability Zone
ap-south-1a (aps1-az1)
Network ACL
acl-010e28182f1d2ac9a
Auto-assign public IPv4 address
Yes
Outpost ID
-
State
available
IPv4 CIDR
10.7.1.0/24
IPv6 CIDR
-
Route Table
rtb-002894aaae01b61c0 | main-1a
Default subnet
No
Auto-assign IPv6 address
No
Owner
769830087441
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


Launch VPC Wizard with Public
Subnets | VPC Management Con
ap-south-1.console.aws.amazon.com/vpc/home?region=ap-south-1#subnets:sort=SubnetId
Services
Resource Groups
Anuddeeph
Mumbai
Support
New VPC Experience
Tell us what you think
Create subnet
Actions
Filter by tags and attributes or search by keyword
1 to 2 of 2
Name
Subnet ID
State
VPC
IPv4 CIDR
Available IPv4
IPv6 CIDR
Availability Zone
main-1a
subnet-07df224f386b57a67
available
vpc-0a7a4b3ec94cfb47b |...
10.7.1.0/24
248
-
ap-south-1a
main-1b
subnet-0e7b9ca21026f6e20
available
vpc-0a7a4b3ec94cfb47b |...
10.7.2.0/24
250
-
ap-south-1b
Subnet: subnet-07df224f386b57a67
Description
Flow Logs
Route Table
Network ACL
Tags
Sharing
Edit route table association
Route Table: rtb-002894aaae01b61c0 | main-1a
1 to 2 of 2
Destination
Target
10.7.0.0/16
local
0.0.0.0/0
igw-000f510d77c0b5596
Filter by VPC:
Select a VPC
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
Endpoints
Endpoint Services
NAT Gateways
Peering Connections
SECURITY
Network ACLs
Security Groups New
VIRTUAL PRIVATE
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


Launch VPC Wizard with Public
Subnets | VPC Management Con
ap-south-1.console.aws.amazon.com/vpc/home?region=ap-south-1#subnets:sort=SubnetId
Services
Resource Groups
Anuddeeph
Mumbai
Support
New VPC Experience
Tell us what you think
Create subnet
Actions
Filter by tags and attributes or search by keyword
1 to 2 of 2
Name
Subnet ID
State
VPC
IPv4 CIDR
Available IPv4
IPv6 CIDR
Availability Zone
main-1a
subnet-07df224f386b57a67
available
vpc-0a7a4b3ec94cfb47b |...
10.7.1.0/24
248
-
ap-south-1a
main-1b
subnet-0e7b9ca21026f6e20
available
vpc-0a7a4b3ec94cfb47b |...
10.7.2.0/24
250
-
ap-south-1b
Subnet: subnet-0e7b9ca21026f6e20
Description
Flow Logs
Route Table
Network ACL
Tags
Sharing
Subnet ID
subnet-0e7b9ca21026f6e20
VPC
vpc-0a7a4b3ec94cfb47b | main
Available IPv4 Addresses
250
Availability Zone
ap-south-1b (aps1-az3)
Network ACL
acl-010e28182f1d2ac9a
Auto-assign public IPv4 address
No
Outpost ID
-
State
available
IPv4 CIDR
10.7.2.0/24
IPv6 CIDR
-
Route Table
rtb-0761c32b9c6d81409 | main-1b
Default subnet
No
Auto-assign IPv6 address
No
Owner
769830087441
Filter by VPC:
Select a VPC
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
Endpoints
Endpoint Services
NAT Gateways
Peering Connections
SECURITY
Network ACLs
Security Groups New
VIRTUAL PRIVATE
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


Launch VPC Wizard with Public
Subnets | VPC Management Con
ap-south-1.console.aws.amazon.com/vpc/home?region=ap-south-1#subnets:sort=SubnetId
aws
Services
Resource Groups
Anuddeeph
Mumbai
Support
New VPC Experience
Tell us what you think
Filter by VPC:
Select a VPC
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
Endpoints
Endpoint Services
NAT Gateways
Peering Connections
SECURITY
Network ACLs
Security Groups New
VIRTUAL PRIVATE
Create subnet
Actions
Filter by tags and attributes or search by keyword
1 to 2 of 2
Name
Subnet ID
State
VPC
IPv4 CIDR
Available IPv4
IPv6 CIDR
Availability Zone
Availab
main-1a
subnet-07df224f386b57a67
available
vpc-0a7a4b3ec94cfb47b |...
10.7.1.0/24
248
-
ap-south-1a
aps1-az
main-1b
subnet-0e7b9ca21026f6e20
available
vpc-0a7a4b3ec94cfb47b |...
10.7.2.0/24
250
-
ap-south-1b
aps1-az
Subnet: subnet-0e7b9ca21026f6e20
Description
Flow Logs
Route Table
Network ACL
Tags
Sharing
Edit route table association
Route Table: rtb-0761c32b9c6d81409 | main-1b
1 to 2 of 2
Destination
Target
10.7.0.0/16
local
0.0.0.0/0
nat-020df89a5bf8e9b32
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use