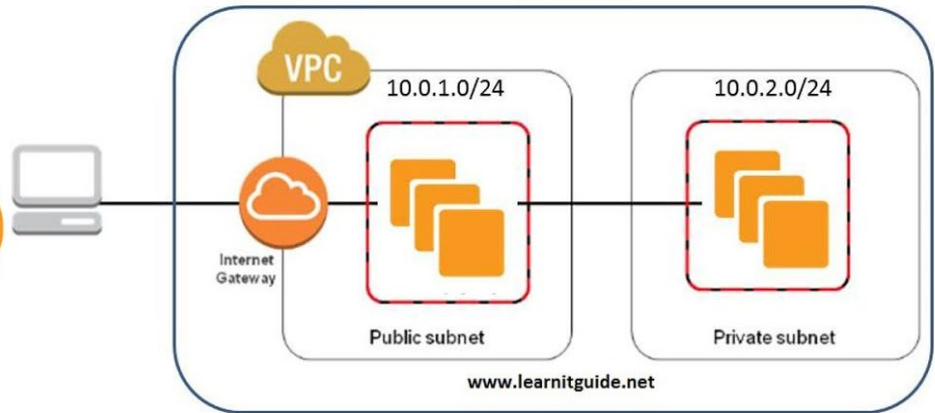


AWS VPC

10.0.0.0/16



HOW TO CREATE NEW VPC SUBNETS, INTERNET GATEWAY



Terraform



Amazon Machine Image (AMI)



Amazon EC2

Task 3 :- We have to create a web portal for our company with all the security

Command prompt :-

Command Prompt

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

C:\Users\Asus>cd desktop

C:\Users\Asus\Desktop>terraformcodefiles
'terraformcodefiles' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Asus\Desktop>cd terraformcodefiles

C:\Users\Asus\Desktop\terraformcodefiles>notepad task3.tf
```

Command Prompt

```
1: provider "aws" {
  A default (non-aliased) provider configuration for "aws" was already given at
  ec2.tf:1,1-15. If multiple configurations are required, set the "alias"
  argument for alternative configurations.

C:\Users\Asus\Desktop\terraformcodefiles>terraform init

Initializing the backend...

Initializing provider plugins...

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

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

* provider.aws: version = "~> 2.70"

Warning: Interpolation-only expressions are deprecated

  on task3.tf line 15, in resource "aws_subnet" "public":
  15:   vpc_id      = "${aws_vpc.myvpc.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 6 more similar warnings elsewhere)

Terraform has been successfully initialized!

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

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

*task3.tf - Notepad

```
File Edit Format View Help
provider "aws" {
  region  = "ap-south-1"
  profile = "Pratik"
}
resource "aws_vpc" "myvpc" {
  cidr_block      = "192.168.0.0/16"
  instance_tenancy = "default"
  enable_dns_hostnames = true

  tags = {
    Name = "pratik_vpc"
  }
}
resource "aws_subnet" "public" {
  vpc_id      = "${aws_vpc.myvpc.id}"
  cidr_block  = "192.168.0.0/24"
  availability_zone = "ap-south-1a"

  tags = {
    Name = "pratikpublicsubnet"
  }
}
resource "aws_subnet" "private" {
  vpc_id      = "${aws_vpc.myvpc.id}"
  cidr_block  = "192.168.1.0/24"
  availability_zone = "ap-south-1b"

  tags = {
    Name = "pratikprivatesubnet"
  }
}
resource "aws_internet_gateway" "gw" {
  vpc_id = "${aws_vpc.myvpc.id}"

  tags = {
    Name = "pratikIG"
  }
}
```

```
Command Prompt

C:\Users\Asus\Desktop\terraformcodefiles>terraform init

Initializing the backend...

Initializing provider plugins...

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

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

* provider.aws: version = "~> 2.70"

Warning: Interpolation-only expressions are deprecated

  on task3.tf line 15, in resource "aws_subnet" "public":
  15:   vpc_id      = "${aws_vpc.myvpc.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 6 more similar warnings elsewhere)

Terraform has been successfully initialized!

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

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

```
Warning: Interpolation-only expressions are deprecated

  on task.tf line 15, in resource "aws_subnet" "public":
  15:   vpc_id      = "${aws_vpc.myvpc.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 6 more similar warnings elsewhere)

Apply complete! Resources: 10 added, 0 changed, 0 destroyed.
```

1.login in aws and create a vpc

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

resource "aws_vpc" "myvpc" {
  cidr_block     = "192.168.0.0/16"
  instance_tenancy = "default"
  enable_dns_hostnames = true

  tags = {
```

```
Name = "pratik_vpc"
```

```
}
```

```
}
```

The screenshot shows the AWS Management Console interface. At the top, there's a navigation bar with the AWS logo, 'Services', 'Resource Groups', and user information (Pratik, Mumbai, Support). Below this is a 'New VPC Experience' banner. The left sidebar contains a 'VIRTUAL PRIVATE CLOUD' section with 'Your VPCs' highlighted, and a 'SECURITY' section with 'Network ACLs'. The main content area shows a table of VPCs with columns: Name, VPC ID, State, IPv4 CIDR, IPv6 CIDR, DHCP options set, Main Route table, and Main Network ACL. The 'pratik_vpc' is the only entry. Below the table, the details for 'VPC: vpc-0a79d675648ea2d6b' are shown, including tabs for Description, CIDR Blocks, Flow Logs, and Tags. The 'Description' tab is active, displaying a list of attributes: VPC ID, State, IPv4 CIDR, IPv6 Pool, Network ACL, DHCP options set, and Owner.

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP options set	Main Route table	Main Network ACL
pratik_vpc	vpc-0a79d675648ea2d6b	available	192.168.0.0/16	-	dopt-0490a668700b92dcb	rtb-05b1e4d44ef116204	acl-07a43e1d72f1b66e3

Attribute	Value
VPC ID	vpc-0a79d675648ea2d6b
State	available
IPv4 CIDR	192.168.0.0/16
IPv6 Pool	-
Network ACL	acl-07a43e1d72f1b66e3
DHCP options set	dopt-0490a668700b92dcb
Owner	810445783252

2. creating two subnet 1 and has auto-launch ip

```
resource "aws_subnet" "public" {  
  vpc_id    = "${aws_vpc.myvpc.id}"  
  cidr_block = "192.168.0.0/24"  
  availability_zone = "ap-south-1a"
```

```
  tags = {  
    Name = "pratikpublicsubnet"  
  }  
}
```

```
resource "aws_subnet" "private" {  
  vpc_id    = "${aws_vpc.myvpc.id}"  
  cidr_block = "192.168.1.0/24"  
  availability_zone = "ap-south-1b"
```

```
  tags = {  
    Name = "pratikprivatesubnet"  
  }  
}
```


aws Services Resource Groups

New VPC Experience
Tell us what you think

VPC Dashboard **New**

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

Create subnet Actions

Filter by tags and attributes or search by keyword

Name	Subnet ID	State	VPC	IPv4 CIDR	Available IPv4	IPv6 CIDR
pratikprivat...	subnet-03a4c91adbdc9faae	available	vpc-0a79d675648ea2d6b ...	192.168.1.0/24	250	-
pratikpublic...	subnet-0af2a448d8b9e4da5	available	vpc-0a79d675648ea2d6b ...	192.168.0.0/24	250	-

VPC: vpc-0a79d675648ea2d6b | pratik_vpc

Available IPv4 Addresses: 250

Availability Zone: ap-south-1a (aps1-az1)

Network ACL: acl-07a43e1d72f1b66e3

Auto-assign public IPv4 address: Yes

IPv4 CIDR: 192.168.0.0/24

IPv6 CIDR: -

Route Table: rtb-0a8164f5a6b26c707 | IGroutetable

Default subnet: No

Auto-assign IPv6 address: No

aws Services Resource Groups

New VPC Experience
Tell us what you think

VPC Dashboard **New**

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**

Create subnet Actions

Filter by tags and attributes or search by keyword

Name	Subnet ID	State	VPC	IPv4 CIDR	Available IPv4	IPv6 CIDR
pratikprivat...	subnet-03a4c91adbdc9faae	available	vpc-0a79d675648ea2d6b ...	192.168.1.0/24	250	-
pratikpublic...	subnet-0af2a448d8b9e4da5	available	vpc-0a79d675648ea2d6b ...	192.168.0.0/24	250	-

Description Flow Logs Route Table Network ACL Tags Sharing

Subnet ID: subnet-03a4c91adbdc9faae

VPC: vpc-0a79d675648ea2d6b | pratik_vpc

Available IPv4 Addresses: 250

Availability Zone: ap-south-1b (aps1-az3)

State: available

IPv4 CIDR: 192.168.1.0/24

IPv6 CIDR: -

Route Table: rtb-05b1e4d44ef116204

Feedback English (US)

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

3. creating an internet gateway for a subnet id in south-1a

```
resource "aws_internet_gateway" "gw" {
  vpc_id = "${aws_vpc.myvpc.id}"

  tags = {
    Name = "pratikIG"
  }
}
```

aws Services Resource Groups VPC > ... > igw-0137bfbddd3fa50fd

New VPC Experience
Tell us what you think

VPC Dashboard **New**

Filter by VPC:
Select a VPC

VIRTUAL PRIVATE CLOUD

- Your VPCs
- Subnets
- Route Tables
- Internet Gateways** **New**

Internet gateway igw-0137bfbddd3fa50fd successfully attached to vpc-0a79d675648ea2d6b

igw-0137bfbddd3fa50fd / pratikIG

Actions

Details Info

Internet gateway ID igw-0137bfbddd3fa50fd	State Attached
--	-------------------

Feedback English (US) Privacy Policy Terms of Use

aws Services Resource Groups VPC > Internet gateways

New VPC Experience
Tell us what you think

VPC Dashboard **New**

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

Internet gateways (1/1) Info

Filter internet gateways

<input checked="" type="checkbox"/>	Name	Internet gateway ID	State	VPC ID
<input checked="" type="checkbox"/>	pratikIG	igw-0137bfbddd3fa50fd	Attached	vpc-0a79d675648ea2d6b

0137bfbddd3fa50fd | pratik_vpc

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

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

```
resource "aws_route_table" "forig" {
  vpc_id = "${aws_vpc.myvpc.id}"

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

  tags = {
    Name = "IGroutetable"
  }
}
```

5. Associating route table with subnet

```
resource "aws_route_table_association" "asstopublic" {
  subnet_id    = aws_subnet.public.id
  route_table_id = aws_route_table.forig.id
}
```

6. Creating the security group with ingress(ssh,http and icmpv4 protocol) - mywebserver_sg

```
resource "aws_security_group" "webserver" {
  name      = "for_wordpress"
  description = "Allow http,ssh"
  vpc_id    = "${aws_vpc.myvpc.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"]
  }
```

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

```
  tags = {
    Name = "mywebserver_sg"
  }
```

Security Groups (1/3) [Info](#)

Filter security groups

	Name	Security group ID	Security group name	VPC ID	Description
<input type="checkbox"/>	-	sg-01ccdb25a34428c28	default	vpc-0a79d675648ea2d6b	default VPC security group
<input checked="" type="checkbox"/>	mywebserver_sg	sg-0722d0c2b42ab7ae1	for_wordpress	vpc-0a79d675648ea2d6b	Allow http,ssh
<input type="checkbox"/>	mydatabase_sg	sg-0f1ea0acaebc9dec6	for_MYSQL	vpc-0a79d675648ea2d6b	Allow ssh and MYSQL

Protocol	Port Range	Source	Destination	Priority
HTTP	TCP 80	:::0	-	100
SSH	TCP 22	0.0.0.0/0	-	100
SSH	TCP 22	:::0	-	100

The screenshot displays the AWS Management Console interface for Security Groups. The left-hand navigation pane includes categories like Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, Elastic Block Store, and Network & Security. The 'Security Groups' link under Network & Security is highlighted. The main panel shows 'Security Groups (1/5)' with a search bar and a table of existing groups. The table has columns for Name, Security group ID, Security group name, and VPC ID. The group 'mywebserver_sg' with ID 'sg-0722d0c2b42ab7ae1' and name 'for_wordpress' in VPC 'vpc-0a79d675648ea2d6b' is selected. Below the table, an ingress rule is shown with Type 'HTTP', Protocol 'TCP', Port range '80', and Source '0.0.0.0/0'.

7. Creating a subnet group with MYSQL protocol and value of security_id(myweb) - mydatabase_sg

```
resource "aws_security_group" "database" {
  name      = "for_MYSQL"
  description = "Allow ssh and MYSQL"
  vpc_id    = "${aws_vpc.myvpc.id}"

  ingress {
    description = "MYSQL"
    security_groups = [aws_security_group.webserver.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 = "mydatabase_sg"
  }
}
```

The screenshot displays the AWS Management Console interface for Security Groups. The left-hand navigation pane includes categories like Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, Elastic Block Store, and Network & Security. The 'Security Groups' link under Network & Security is highlighted. The main panel shows the 'Security Groups (1/5)' page with a search bar and a table of existing security groups. The table has columns for Name, Security group ID, Security group name, and VPC ID. The group 'mydatabase_sg' with ID 'sg-0f1ea0acaebc9dec6' and name 'for_MYSQL' is selected. Below the table, a rule is configured for 'MySQL/Aurora' using 'TCP' on port '3306' with the source set to 'sg-0f1ea0acaebc9dec6 (for_MYSQL)'.

8. Launching the instance

```
resource "aws_instance" "wordpress" {
  ami          = "ami-00b494a3f139ba61f"
  instance_type = "t2.micro"
  associate_public_ip_address = true
  subnet_id = aws_subnet.public.id
  vpc_security_group_ids = [aws_security_group.webserver.id]
  key_name = "mykey111"
```

```
tags = {
  Name = "wordpress"
}
```

```
}

resource "aws_instance" "mysql" {
  ami          = "ami-0019ac6129392a0f2"
  instance_type = "t2.micro"
  subnet_id = aws_subnet.private.id
  vpc_security_group_ids = [aws_security_group.database.id]
  key_name = "mykey111"
```

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

```
}
```



Warning: Interpolation-only expressions are deprecated

```
on task.tf line 15, in resource "aws_subnet" "public":
15:   vpc_id      = "${aws_vpc.myvpc.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 6 more similar warnings elsewhere)

Destroy complete! Resources: 10 destroyed.