September 14, 2021

Article | 7 min

Multi-VPC, Single Internet Egress and a Transit Gateway

Focusing on scalability and security as you move to an AWS cloud.

Arun Daniel

When you first get into cloud technologies, you begin with one account and create boundaries in that account. This works for a while and then, sure enough, the floodgates open at some point and you are faced with the Wild West of cloud. As your organization grows its footprint in AWS, you end up relying on hundreds of accounts and Virtual Private Clouds (VPCs) to segment the workloads and/or create segregation within its business divisions. Creating scalable and secure networking architectures lets you further increase that footprint with more governance compared to the Wild West most organizations find themselves in as they mature in the cloud.

Explore the Options

Maintaining Security with Scalability

This article touches the tip of the iceberg when it comes to creating multiple VPCs and using one of those VPCs as your egress (and ingress) to (and from) the internet. In the following example, I will walk through having two consumer VPCs that are connected (separately) to an egress VPC, which will be used to send/receive internet traffic.

Although this is a very basic example, you can use your imagination to think about how this could be a stepping stone towards a more secure option by using the egress VPC to hold firewall appliances, logging for security, gateway load balancers (more on this in another post), as well as a slew of other AWS services that can be used to converge a sprawling cloud campus down to a more manageable security post to allow/deny/log all internet traffic.

Managing in AWS Cloud

Here are the ingredients we need to make this hub-and-spoke architecture work:

VPCs
Outbound VPC:
This will be our bouncer for access into Club Internet.

Consumer VPC1 and Consumer VPC2:
These will be two of many VPCs that can hold private AWS services, which will use the Outbound VPC to get out to the internet.

Subnets (and Route Tables for These Subnets)
The outbound VPC will have a public and a private subnet (in production, always using more than one for everything to obtain resiliency). The public subnet will hold the NAT Gateway (again, always have more than one), while the private subnet will hold the Transit Gateway attachment, which will be used to send/receive traffic to/from the internet to the consumer VPCs (also via their own attachments to the Transit Gateway).

Each Consumer VPC will have a private subnet each (in production environments, always have multiple subnets for resiliency).

Transit Gateway

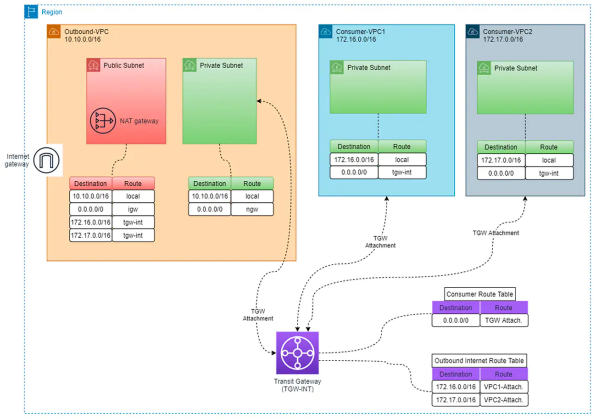
create blackholes between the consumer VPCs so as not to create any “bridge” between the two (for security purposes).

Internet Gateway
This is our door to the internet and will be connected only to the Outbound VPC

NAT Gateway
Although this object will sit in the Public Subnet in our Outbound VPC, we will use this as our door to the internet for all Consumer VPC traffic destined to the Internet.

Here is what the final logical view will look like:

Hub-and-Spoke Architecture Set-Up



The Breakdown (in Terraform)

Outbound-VPC

#EGRESS/OUTBOUND VPC

```
resource "aws_vpc" "vpcoutbound" {

  cidr_block = "10.10.0.0/16"

  tags = {

    Name = "vpc-outbound"

  }
```

```
resource "aws_eip" "nat" {

vpc = true

tags = {

Name = "vpc-outbound-nat" }

}
#NAT Gateway object and attachment of the Elastic IP Address from above

resource "aws_nat_gateway" "ngw" {

allocation_id = aws_eip.nat.id

subnet_id = aws_subnet.vpcoutboundpubsub1.id

depends_on = [aws_internet_gateway.igw]

  tags = {

  Name = "ngw-outbound"

}

}
#Internet Gateway

resource "aws_internet_gateway" "igw" {

vpc_id = aws_vpc.vpcoutbound.id

tags = {

Name = "igw-outbound"

}

}
```

#SUBNETS

```
#Public Subnet 1

resource "aws_subnet" "vpcoutboundpubsub1" {

cidr_block = "10.10.0.0/24"

vpc_id = aws_vpc.vpcoutbound.id

map_public_ip_on_launch = true

availability_zone = data.aws_availability_zones.available.names[0]

tags = {

Name = "vpcoutboundpubsub1"

}

}
#Public Route Table Entry - Internet Bound

resource "aws_route_table" "vpcoutboundroutetablepublic" {
```

```
gateway_id = aws_internet_gateway.igw.id

}

route { #send all VPC1 Consumer traffic through the Transit Gateway

cidr_block = aws_vpc.vpc1consumer.cidr_block

gateway_id = aws_ec2_transit_gateway.tgw.id

}

route { #send all VPC2 Consumer traffic through the Transit Gateway

cidr_block = aws_vpc.vpc2consumer.cidr_block

gateway_id = aws_ec2_transit_gateway.tgw.id

}

}
#Associate Public Route Table to Public Subnet

resource "aws_route_table_association" "vpcoutboundroutetablepublicas1" {

subnet_id = aws_subnet.vpcoutboundpubsub1.id

route_table_id = aws_route_table.vpcoutboundroutetablepublic.id

}

#Private Subnet 1

resource "aws_subnet" "vpcoutboundprisub1" {

cidr_block = "10.10.2.0/24"

vpc_id = aws_vpc.vpcoutbound.id

availability_zone = data.aws_availability_zones.available.names[0]

tags = {

Name = "vpcoutboundprisub1"

}

}

#Private Route Table

resource "aws_route_table" "vpcoutboundroutetableprivate" {

vpc_id = aws_vpc.vpcoutbound.id

route {

cidr_block = "0.0.0.0/0"

gateway_id = aws_nat_gateway.ngw.id

}

tags = {

Name = "vpcoutboundroutetableprivate"

}
```

```
resource "aws_route_table_association" "vpcoutboundroutetableprivateas1" {

subnet_id = aws_subnet.vpcoutboundprisub1.id

route_table_id = aws_route_table.vpcoutboundroutetableprivate.id

}
#TRANSIT GATEWAYS

resource "aws_ec2_transit_gateway" "tgw" {

  default_route_table_association = "disable"

#for security reasons, we dont want to have attached VPCs to use the
default route table

  default_route_table_propagation = "disable"

#for security reasons, we dont want to have attached VPCs to propogate
their networks to the route tables

  auto_accept_shared_attachments = "enable"

tags = {

Name = "tgw"

}

}

#outbound vpc attachment

resource "aws_ec2_transit_gateway_vpc_attachment" "outboundvpcattachment" {

 subnet_ids        = [aws_subnet.vpcoutboundprisub1.id,

                  aws_subnet.vpcoutboundprisub2.id]

 transit_gateway_id = aws_ec2_transit_gateway.tgw.id

 vpc_id           = aws_vpc.vpcoutbound.id

 transit_gateway_default_route_table_association = false

 transit_gateway_default_route_table_propagation = false

 tags = {

   Name = "OutboundAttachment"

 }

}

#tgw outbound route table

resource "aws_ec2_transit_gateway_route_table" "egressroutetable" {

 transit_gateway_id = aws_ec2_transit_gateway.tgw.id

 tags = {

   Name = "OutboundRouteTable"

 }
```

```
 transit_gateway_attachment_id  = aws_ec2_transit_gateway_vpc_attachment.outboundvp
 transit_gateway_route_table_id = aws_ec2_transit_gateway_route_table.egressroutetable.
 }
#route to the consumer 1 via vpc1 attachment
resource "aws_ec2_transit_gateway_route" "egressroutetableRouteVPC1" {
 destination_cidr_block       = aws_vpc.vpc1consumer.cidr_block
 transit_gateway_attachment_id  = aws_ec2_transit_gateway_vpc_attachment.vpc1consum
 transit_gateway_route_table_id = aws_ec2_transit_gateway_route_table.egressroutetable.
}
#route to the consumer 2 via vpc2 attachment
resource "aws_ec2_transit_gateway_route" "egressroutetableRouteVPC2" {
 destination_cidr_block       = aws_vpc.vpc2consumer.cidr_block
 transit_gateway_attachment_id  = aws_ec2_transit_gateway_vpc_attachment.vpc2consum
 transit_gateway_route_table_id = aws_ec2_transit_gateway_route_table.egressroutetable.
}
```

Consumer-VPC1

#CONSUMER VPC1

```
resource "aws_vpc" "vpc1consumer" {
   cidr_block = "172.16.0.0/16"
   tags = {
      Name = "vpc1-consumer"
   }
}
#Private Subnet 1
resource "aws_subnet" "vpc1consumerprisub1" {
cidr_block = "172.16.1.0/24"
vpc_id = aws_vpc.vpc1consumer.id
availability_zone = data.aws_availability_zones.available.names[0]
tags = {
Name = "vpc1consumerprisub1"
}
}
#Private Route Table
resource "aws_route_table" "vpc1consumerroutetableprivate" {
```

```
gateway_id = aws_ec2_transit_gateway.tgw.id

}

tags = {

Name = "vpc1consumerroutetableprivate"

}

}
#Associate Private Route Table to Private Subnet

resource "aws_route_table_association" "vpcoutboundroutetableprivateVPC1" {

subnet_id = aws_subnet.vpc1consumerprisub1.id

route_table_id = aws_route_table.vpc1consumerroutetableprivate.id

}

#Create TGW Attachment for this VPC

resource "aws_ec2_transit_gateway_vpc_attachment" "vpc1consumervpcattachment" {

 subnet_ids        = [aws_subnet.vpc1consumerprisub1.id]

 transit_gateway_id = aws_ec2_transit_gateway.tgw.id

 vpc_id            = aws_vpc.vpc1consumer.id

 transit_gateway_default_route_table_association = false

 transit_gateway_default_route_table_propagation = false

 tags = {

    Name = "VPC1Attachment"

 }

}

#tgw vpc 1 consumer route table association

resource "aws_ec2_transit_gateway_route_table_association" "vpc1association" {

 transit_gateway_attachment_id  = aws_ec2_transit_gateway_vpc_attachment.vpc1consum

 transit_gateway_route_table_id = aws_ec2_transit_gateway_route_table.consumerrouteta

 }

#tgw consumer route table

resource "aws_ec2_transit_gateway_route_table" "consumerroutetable" {

 transit_gateway_id = aws_ec2_transit_gateway.tgw.id

 tags = {

    Name = "ConsumerRouteTable"

 }

}
```

```
 transit_gateway_attachment_id  = aws_ec2_transit_gateway_vpc_attachment.outboundvp
 transit_gateway_route_table_id = aws_ec2_transit_gateway_route_table.consumerrouteta
}
resource "aws_ec2_transit_gateway_route" "blackholevpc1" {
 destination_cidr_block       = aws_vpc.vpc1consumer.cidr_block
 blackhole                    = true
 transit_gateway_route_table_id = aws_ec2_transit_gateway_route_table.consumerrouteta
}
resource "aws_ec2_transit_gateway_route" "blackholevpc2" {
 destination_cidr_block       = aws_vpc.vpc2consumer.cidr_block
 blackhole                    = true
 transit_gateway_route_table_id = aws_ec2_transit_gateway_route_table.consumerrou
tetable.id
```

Consumer-VPC2

#CONSUMER VPC2

```
resource "aws_vpc" "vpc2consumer" {
  cidr_block = "172.17.0.0/16"
  tags = {
    Name = "vpc2-consumer"
  }
}
#Private Subnet 1
resource "aws_subnet" "vpc2consumerprisub1" {
cidr_block = "172.17.1.0/24"
vpc_id = aws_vpc.vpc2consumer.id
availability_zone = data.aws_availability_zones.available.names[0]
tags = {
Name = "vpc2consumerprisub1"
}
}
#Private Route Table
resource "aws_route_table" "vpc2consumerroutetableprivate" {
vpc_id = aws_vpc.vpc2consumer.id
route {
```

```
tags = {
Name = "vpc2consumerroutetableprivate"
}
}
#Associate Private Route Table to Private Subnet
resource "aws_route_table_association" "vpcoutboundroutetableprivateVPC2" {
subnet_id = aws_subnet.vpc2consumerprisub1.id
route_table_id = aws_route_table.vpc2consumerroutetableprivate.id
}
resource "aws_ec2_transit_gateway_vpc_attachment" "vpc2consumervpcattachment" {
 subnet_ids       = [aws_subnet.vpc2consumerprisub1.id]
 transit_gateway_id = aws_ec2_transit_gateway.tgw.id
 vpc_id           = aws_vpc.vpc2consumer.id
 transit_gateway_default_route_table_association = false
 transit_gateway_default_route_table_propagation = false
 tags = {
    Name = "VPC2Attachment"
 }
}
resource "aws_ec2_transit_gateway_route_table_association" "vpc2association" {
 transit_gateway_attachment_id  = aws_ec2_transit_gateway_vpc_attachment.vpc2consum
 transit_gateway_route_table_id = aws_ec2_transit_gateway_route_table.consumerrouteta
}
```

Testing Your Architecture

To test the above hub-and-spoke architecture, you can:

Create a jump host in the Outbound VPC in the public subnet and confirm that the EC2 instance will have a public IP address along with a security group to allow you to SSH or RDP to this jump box.

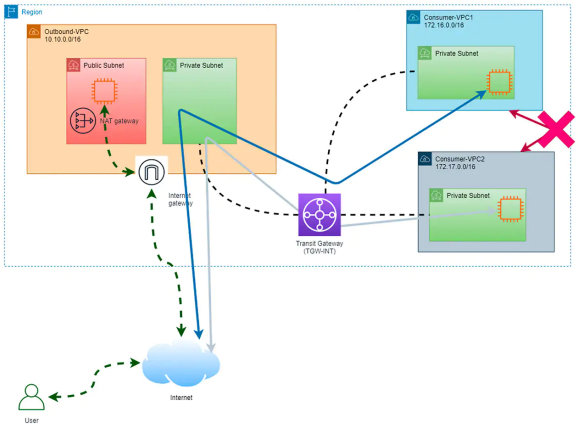
Create a private instance in the Consumer VPC 1 network and allow SSH or RDP access to this instance from the jump box.

Create another private instance in the Consumer VPC 2 network

You should be able to confirm that from either Consumer EC2 instance you can connect to the Internet; however, you cannot connect to the other Consumer EC2 instance (thanks to the blackhole route).

Network Flow

To conclude, all Consumer VPC internet traffic is routed through the Transit Gateway, through the NAT Gateway, out to the internet. Thanks to the blackhole routing entries, the traffic between the two Consumer VPCs will never be allowed.



Arun Daniel *is a Sr. Consulting Engineer for Data Center and Cloud Services at CDW with more than 20 years of experience designing, deploying and managing all aspects of data center and cloud services. For the past 10 years, his primary focus has been migrations from on-premises data centers to Amazon Web Services. Arun holds numerous AWS certifications, as well as HashiCorp, Microsoft Azure, VMware and Cisco certifications.*