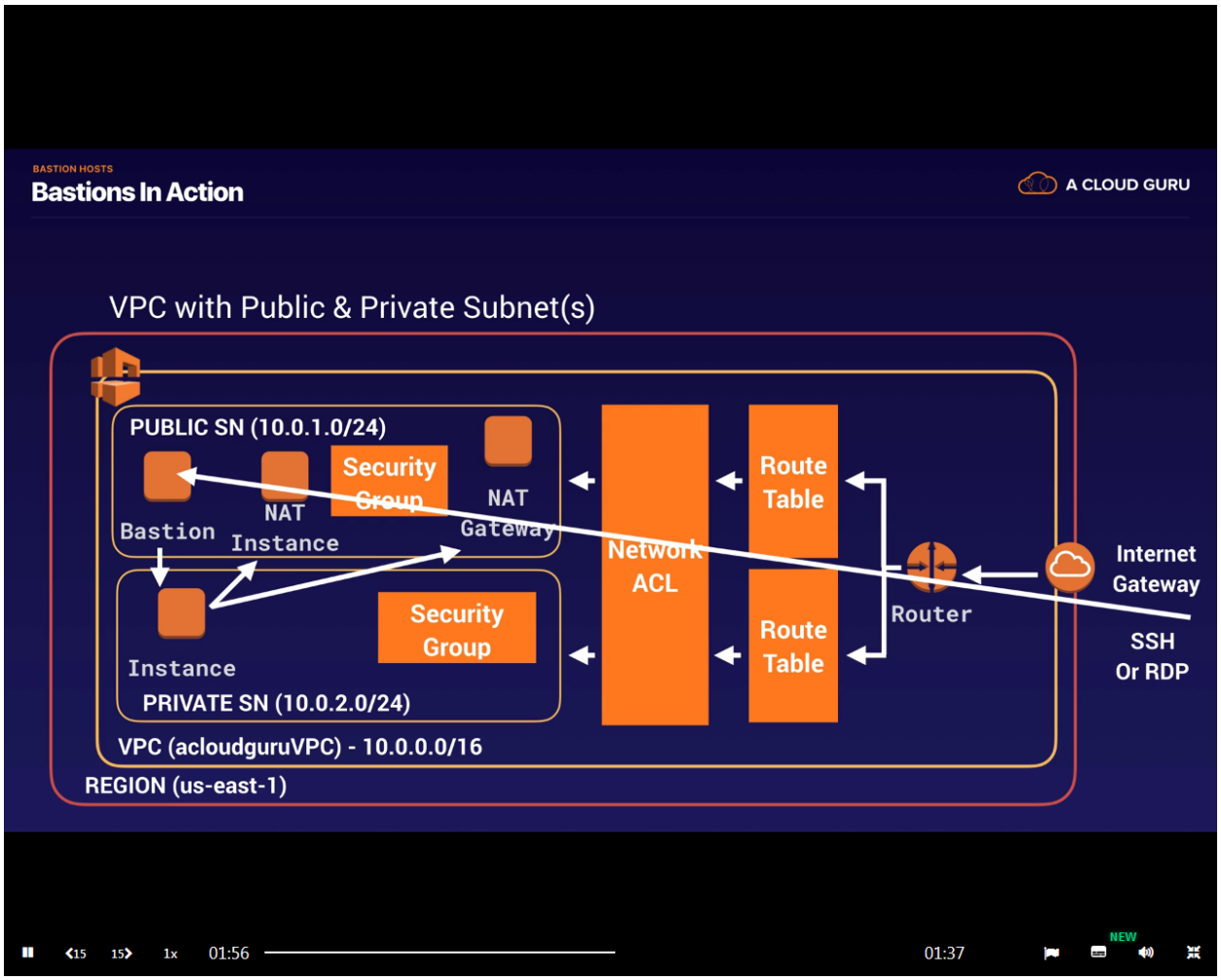
NAT Instances, NAT Gateways, & Bastion Hosts



A Bastion is used to SSH or RDP into an instance in your private subnet.

Whereas, A NAT Gateway (as well as a NAT instance) is used to provide internet traffic to EC2 instances in a private subnet. In other words, it provides a way out to the internet for your instance located in the private subnet. A NAT Gateway is redundant. A Nat instance is an individual EC2 instance located **behind** a security group. Be advised that a Nat instance is a single-point of failure so a NAT Gateway is preferable.

Picking up from where we left on in the Building a Custom VPC in AWS lesson. We currently have no way of communicating to the private instance located in the VPC that we created. If we SSH into our public instance called WebServerInPublicSubnet and from there try to SSH into our our private instance called

DBServerinPrivateSubnet, it will not work. The reason is because these instances are in two separate security groups that do not allow access to each other.

In a production scenario we would use a bastion host to SSH into our private instance. However, for the purpose of demonstrating NAT instances, and NAT Gateways, we will create a new security group for our private instance that will allow us to SSH into it (since the default security group does not have inbound rules that will allow us to do that).

At our VPC dashboard

Click **Security Groups** in the left margin

Click **create security group**

Security Group Name: PrivateDBServerSecurityGroup ← gave it a name

Description: PrivateDBServerSecurityGroup

VPC: OurNewCustomVPC

Inbound Rules

	Type	Protocol	Port Range	Source	
Add →	All ICMP-IPv4	ICMP	All	Custom	10.0.1.0/24 ← the IP range for our public subnet
Add →	HTTP	TCP	80	Custom	10.0.1.0/24
Add →	HTTPS	TCP	443	Custom	10.0.1.0/24
Add →	SSH	TCP	22	Custom	10.0.1.0/24
Add →	MySQL/Aurora	TCP	3306	Custom	10.0.1.0/24

★The above rules allow us to communicate to our DBServerInPrivateSubnet using All ICMP-IPv4, HTTP, HTTPS, SSH, and MySQL/Aurora from our public subnet which has the IP range 10.0.1.0/24! And to reiterate, with regards to security groups if traffic is allowed inbound it is also allowed outbound.

★Alternatively, we could have had the source as our PrivateDBServerSecurityGroup which governs our security for our public instance in our public subnet that has IP range 10.0.1.0/24. This would have accomplished the same thing.

★All ICMP-IPv4 allows us to ping our private instance in our private subnet from our public instance in our public subnet

Click **Create Security Group**

Click **Services**, then click **EC2**

✓DBServerinPrivateSubnet ← Select DBServerinPrivateSubnet

Click **Actions** and then hover over networking and click **change security groups**

Deselect the default security group and **select** the PrivateDBServerSecurityGroup

Click **Assign Security Group**

Click **Services** and click **EC2**

✓DBServerinPrivateSubnet ← Select DBServerinPrivateSubnet

Scroll down and click on the description tab and copy the private IP address of DBServerinPrivateSubnet

The private IP is 10.0.2.126 ← ★The 10.0.2.126 is the private IP of our DBServerinPrivateSubnet. It will likely be a different IP when you do the exercise yourself so bear that in mind.

Now ✓WebServerInPublicSubnet and click the **connect** button to SSH into our public instance

✓EC2 Instance Connect (browser-based SSH connection) ← select EC2 Instance Connect (browser-based SSH connection)

Then type the following:

sudo su (and then hit enter)

clear (and then hit enter)

ping 10.0.2.126 (and hit enter)

Our public instance is now sending pings to our private instance. Hit **Ctrl and C** at the same time to stop the pings.

Now we will SSH over to our private instance. And normally for security reasons we would use a bastion host but this is a demonstration.

Open up the SingaporeKeyPair.pem file with a text editor. Select all the contents and copy it to your clipboard.

Now back in the command line type the following:

nano SingaporeKeyPair.pem (and hit enter)

Now paste the contents from your clipboard and then hit Ctrl + x. Then hit y (for yes I want to Save) and then hit enter and enter again.

You should be back to the regular command line prompt.

Now type the following:

chmod 400 SingaporeKeyPair.pem (and hit enter)

ls (and hit enter) ← that is a lowercase L in case you are confusing it with another letter.

ssh ec2-user@10.0.2.126 -i SingaporeKeyPair.pem (and hit enter) ★The 10.0.2.126 is the private IP of our DBServerinPrivateSubnet. It will likely be a different IP when you do the exercise yourself.

(It might ask you if you want to connect, in which case type yes and hit enter)

We are now SSH'd into our database server located in our private subnet. (We SSH'd into our public server and then SSH'd into our private server)

Now in the command line type the following:

yum update -y (and hit enter)

This command is meant to update our operating system. The command fails because we are in a private subnet that has no route out to the internet. Also, if we try to download software (such as MySQL) it will fail because we have no route out to the internet.

Therefore, we need a NAT instance or a NAT Gateway so that our EC2 instance in the private subnet can download software.

Let's create a NAT instance.

Click **Services** and Click **EC2.** Click **Launch Instance.**

Click **Community AMI's** in the left margin

Type nat into the search tab. Choose the selection at the very top of the list.

✓amzn-ami-vpc-nat-2018.03.0.20200318.2-x86_64-ebs ← will look something like this

✓t2.micro ← then select t2.micro

Click configure instance details.

Network: OurNewCustomVPC

Subnet: 10.0.1.0–ap-southeast-1a-PublicSubnet ← select our public subnet because we will launch our NAT instance into our public subnet.

Auto-assign Public IP Use subnet setting (Enable) (we will then get a public IP address for this instance)

Click Next: Add Storage

Click Next: Add Tags

Key: Name

Value: NAT_Instance

Click Next: Configure Security Group

Select an existing Security Group

SecurityGroupForPublicInstance ← choose this security group as it has http, https, and ssh open

Click Review & Launch

✓Make General Puporse (SSD) the boot volume for this instance ← select this option

Click next

Click Launch

Choose an existing key pair which is SingaporeKeyPair and acknowledge that you have the key.

Click Launch Instance

★Disabling Source/Destination Checks

Each EC2 instance performs source/destination checks by default. This means that the instance must be the source or destination of any traffic it sends or receives. However, a NAT instance must be able to send and receive traffic when the source or destination is NOT itself. Therefore, you must disable source/destination checks on the NAT instance. You can do so using the console or the command line.

So our NAT instance is the bridge that allows the instance in our private subnet to have a route out to the internet.

✓NAT_Instance ← select the NAT instance that we created and click **Actions** and click **Networking** then click **Change Source/Destination checks** and click **Yes, Disable**

Now we want our DBServerinPrivateSubnet to talk to our NAT_instance. So we need to create a route in our default/main route table (because remember our default/main route table associates with our private subnet).

Click **services** and click **VPC**

Click Route Tables in the left margin

✓Select our default/main route table that was created automatically when we created our VPC which is currently associated with our private subnet

Then click the **Routes** tab and click **Edit Routes**

Click Add Route

	Destination	Target	Status	propogated
Already there →	10.0.0.0/16	local	active	No
Already there →	2406:da18:50f:2100::/56 (example of IPv6)	local	active	No
Add this →	0.0.0.0/0	Nat_Instance		

Click Save Routes

So we just added a route to our default/main route table (which is associated to our private subnet) with destination 0.0.0.0/0 with a target that is our Nat_instance.

Click **Services** and click **EC2**

Select ✓WebServerInPublicSubnet and click the **connect** button to SSH into our public instance

✓EC2 Instance Connect (browser-based SSH connection) ← select EC2 Instance Connect (browser-based SSH connection)

Type the following into the terminal:

sudo su (and hit enter)

type clear (and hit enter)

ls (and hit enter)

(Our SingaporeKeyPair will be listed since we saved the file to our instance earlier)

Now type:

ssh ec2-user@10.0.2.126 -i SingaporeKeyPair.pem (and hit enter)

yes (and hit enter)

(We are now SSh'd into our private instance)

Type the following:

sudo su (and hit enter)

clear (and hit enter)

yum update -y (and hit enter)

This will update our operating system and if it is successful it means that we have a route out to the internet as we desired. It is using the NAT instance to go out into the internet and update the operating system for our EC2 instance in our private subnet.

★The problem with a NAT instance is that it is a single point of failure.

Minimize the command line for now.

Back in the console click **Services** and click **EC2**

✓ NAT_Instance ← select the NAT_Instance that we created

Click **Actions** and Click **Terminate**

Once the NAT instance has been terminated go back to the command line and type the following:

yum install httpd -y (and hit enter)

This will not work because our NAT instance has been terminated.

Minimize the command line.

Go to the console and click **Services** and then click **VPC**

Click **Route Tables** in the left margin

✓Select our default/main route table that was created automatically when we created our VPC which is currently associated with our private subnet

Click the **routes** tab

Click **edit routes**

Destination	Target	Status	propogated
Leave this route as is → 10.0.0.0/16	local	active	No
Leave this route as is → 2406:da18:50f:2100::/56 (example of IPv6)	local	active	No
Delete this route → 0.0.0.0/0	NAT_Instance		

Click **Save routes**

(We just deleted the route with target NAT_Instance because we have deleted the NAT_Instance)

Now we will create a NAT Gateway

Click **NAT Gateways** in the left margin

Click **Create NAT Gateway**

Subnet: 10.0.1.0–ap-southeast-1a-PublicSubnet ← select our public subnet because we will launch our NAT Gateway into our public subnet

To the right of Elastic IP Allocation ID* Click the **Allocate Elastic IP button**

This will generate an Elastic IP such as eipalloc-0ee29dd43741bd9f1

Click **Add Tag**

Key: Name Value: NAT_Gateway

Then click **Create a NAT Gateway**

Click **Route Tables** in the left margin

✓Select our default/main route table that was created automatically when we created our VPC which is currently associated with our private subnet

Click the **Routes** tab

Click **Edit routes**

Destination	Target	Status	propogated
Leave this route as is → 10.0.0.0/16	local	active	No
Leave this route as is → 2406:da18:50f:2100::/56 (example of IPv6)	local	active	No
Add this route → 0.0.0.0/0	NAT_Gateway		

Click **Save Routes**

(So we just added a route with a destination of 0.0.0.0/0 (the internet) and the target is the NAT gateway we just created)

Now return to our terminal window where we are SSH'd into our private EC2 instance.

Now type the following:

clear (and hit enter)

yum install mysql -y (and hit enter)

Our private instance will use the NAT gateway that we created in order to install MySQL server

★NAT Gateways are redundant inside the Availability Zone. However, you can only have one NAT gateway inside an Availability Zone. NAT gateways cannot span Availability zones so if the Availability Zone that you are using goes down then your NAT gateway will not work. The great thing about NAT gateways, aside from being redundtant, is that there is no need to disable source/destination checks. Also, notice in the diagram that the NAT gateway does not need to be behind a security group.

Mission Complete. The next lesson will be on Network Access Control Lists aka NACLs and will pick up from this point. If you do not have time to continue on then delete your VPC and delete your instances. Below are important concepts and exam questions pertaining to NAT instances and NAT Gateways.

★**NAT Instance and NAT Gateway**

-When creating a NAT instance you **must disable source and destination checks** on the instance

-NAT instances **must exist in a public subnet**

-You must have a **route out** of the private subnet to the NAT instance

-The size of a NAT instance determines **how much traffic can be handled**

-High availability can be achieved using **Autoscaling Groups**, multiple **subnets in different AZs**, and **automate failover between them using a script.**

- -

-NAT Gateways are **redundant inside an Availability Zone** (can survive failure of EC2 instance)

-You can only have 1 NAT Gateway inside 1 Availability Zone (cannot span AZs)

-Starts at 5 Gbps and scales all the way up to 45 Gbps

-NAT Gateways are the **preferred setup for enterprise systems**

-There is no **requirement to patch NAT Gateways** and there is no need to disable Source/Destination checks for the NAT Gateway (unlike NAT instances)

-NAT Gateways are **automatically assigned a public IP address**

-**Route Tables** for the NAT Gateway MUST be updated

-Resources in multiple AZs sharing a Gateway will **lose internet access if the Gateway goes down**, unless you create a **Gateway in each AZ** and configure **route tables** accordingly

---

Practice Question

QUESTION
You are hosting a web application that runs on a number of Web Servers in public subnets and Database Servers in private subnets. A NAT Instance is being used for connectivity to the internet for the Private Subnets. The NAT Instance is now becoming a bottleneck, and you are looking to replace it with NAT Gateway. Which of the following would ensure high availability for the NAT Gateway?

A) Disable source/destination check on the NAT Instances
B) Deploy a NAT Gateway in 2 Availability Zones
C) Deploy a NAT Gateway in 2 Regions
D) Deploy a NAT Gateway along with the NAT Instance

EXPLANATION:
If you have resources in multiple Availability Zones and they share one NAT gateway, in the event that the NAT gateway's Availability Zone is down, resources in the other Availability Zones lose internet access. To create an Availability Zone-independent architecture, create a NAT gateway in each Availability Zone and configure your routing to ensure that resources use the NAT gateway in the same Availability Zone. Answer is B.