***Basic EC2 networking – Instance ENI, IP and DNS***

The first is **public instance** and by public instance I mean an EC2 instance, which has been given a public IP address and it's inside a VPC, which contains any networking required to make that public IP address work. So, in effect, an instance that you can connect to from the public internet.

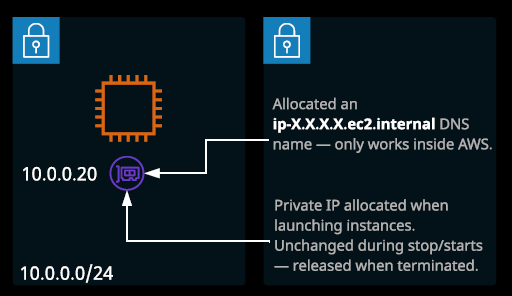
The second term is **private instance** and a private instance is one that can only be communicated with inside a VPC. So that's one without any public internet connectivity. A private instance can be in the VPC, which does have public connectivity only it's not given a public IP address or it's in a subnet is not configured to be publicly accessible.

Private instances. So in this case, we've got this private instance it's got a private IP address, and it's got a single attached network interface or ENI.

I'm going to change the **auto assign public IP address** **from use subnet setting enable to disable**. This is going to make it a private instance. It's not going to have any public connectivity.

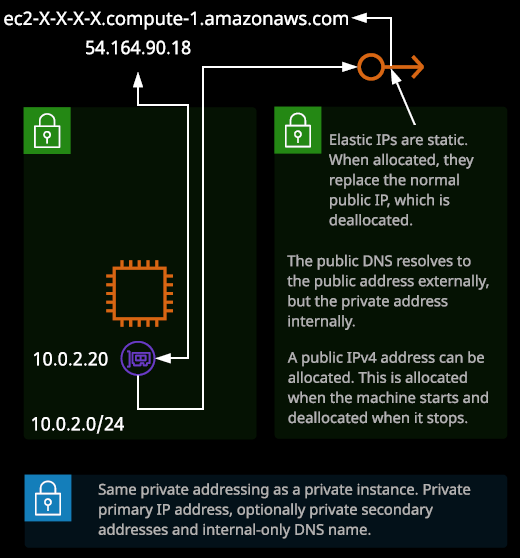
Now, by default, any instance created by EC2 is going to start off with a **default ENI or elastic network interface**. So this is the **network interface card, or virtual network interface card** that the instance starts with and when you launch any instance into EC2 with an elastic network interface, it's going to be allocated an IP address within the range that the subnet uses. So this particular instance is located in U.S. East 1c and it's located in this particular subnet that has an IP address range associated with it. The instance as I launch it into that subnet is going to be given a private IP address, so this is an IP address that I can't connect to over the public internet. It's known as a **private IP address**. It's not one that's routable across the public internet. **Now this IP address is associated with the instance for the lifetime of that instance, so it stays the same**. Note that it's currently 172.31.28.6. If I stop the instance so go to instance state and then stop the IP is going to remain associated with this instance so note that even though it's going through this stopping state and it will finally move to a stopped state that IP address will stay associated with EC2 instance or rather the network card that's associated with the EC2 instance. So **private IP addresses inside AWS, they don't change they're static for the lifetime of the EC2 instance**. So after a couple of refreshes, we can see that it's now in a stop state. The private IP addresses still associated with it, and if I start that instance up again, it will still have that same IP address.

Now the **default network interface** that's associated with an EC2 instance is known as **eth0**. If I click on eth0 of this particular instance, I'll be presented with the interface ID. So the **ENI elastic network interface ID for this network interface**. We can assign new Private IP addresses to the instance. We're only allowed to have two private IP addresses associated with it. That is something that differs based on the size of the instance. So I'm able to have **multiple private IP addresses associated with the networking interface and I could also have multiple network interfaces**.



Now, we also have for private instances is a **private DNS name**. So this is an internal only DNS name. You'll note that it's got the IP address built into it. So its IP- and then the IP address but instead of dots, it has dashes. If the IP address were to ever change this internal DNS name would also change. It's based on the IP address, but **because the IP address belongs to this instance for the lifetime of the instance, so does this internal only DNS name**. **I'm not able to ping this externally. It only works inside the VPC, and it always resolves to the private IP address of the EC2 instance**, so I'm going to go ahead and terminate this instance. So **I'll go to instance state and terminate and that will result in the IP address this private IP being deallocated from this instance and return to the pool**.

**Public instances**



So instances that have a public IP address and are accessible from the public internet. Now, public instances share the same private addressing as private instances. So they have the **private IP and private DNS** but they also have **public variants**.

I'll make sure that the **IP address setting is used subnet setting enabled** that will make sure that it's given a public IP address.

So just to confirm this instance, even though it is public, has the same private addressing. We've got the same private IP address and the same private DNS name and both of those behave in exactly the same way as a private instance. Now in addition to this private addressing, public instances receive an **IP version four public IP address**. This is a publicly routable IP address that provided you don't have any security restrictions in place, you'll be able to reach from your machine traversing over the public internet and you can connect to this over SSH for example and if you connect over SHH you're able to access the shell of the instance.

Because this is a Linux machine I'm able to run the ifconfig command to display an overview of the network interfaces inside the guest operating system. This command will list the configuration of the network interfaces as far as the operating system is aware. What I want to draw your attention to is how in none of this output, the external IP address of the EC2 instance is listed. As far as this EC2 instance is concerned, it does not have a public IP address. It only has this private IP address and if we move back to the console, we'll see that this is the private IP address of this instance but it doesn't have this public IP address.   
The way that this works there is a networking device known as an **internet gateway**. **Its job is to be told when any device is inside AWS a given public IP addresses and when a device that has an associated public IP address attempts to communicate with the internet or when any internet devices attempt to communicate with its public IP address its job is to swap those IP addresses**. So if this instance which doesn't have a public IP address on the operating system attempts to do let's say a ping to the internet so I ping 1.1.1.1 what will happen is this internet gateway will receive data from this private IP address and it will swap that private IP address for this public IP address. So when 1.1.1.1 receives the traffic, it will see it as from this IP address. The same process is true in reverse. When I connect to this instance over SSH, I connect to this IP address. The job of the internet gateway is to swap this IP address with the private IP address, which is what allows my SSH connection to reach the EC2 instance.   
You might get a question where an EC2 instance cannot reach any machines on the public internet or any machines on the public internet can't reach your EC2 instance and one of the answers might be you need to apply a public IP address on the guest operating system, and that is not a valid way to fix the problem. **You cannot apply a public IP address on any operating system inside AWS**. That's not how it works. **Everything in AWS is natively private when using IP version four and it's the job of the internet gateway to translate that.** So that's critical to understand. **The internet gateway actually uses a process called NAT or network address translation*. It translates a private IP to a public and a public IP to a private.***

Now, as well as this public IP address EC2 instances also receive a **public DNS name**. If you take that public DNS name and I'll go back to this EC2 instance. So this EC2 instance is inside this VPC. If I ping that public DNS name, note that I get a private IP address the private IP address of the EC2 instance. So I want to cancel that process and I'm going to exit out to the local shell of my machine and I'm going to do that same process. I'm going to ping this public DNS name. This time when a ping that address, even though it's timing out and that's fine because I don't have the necessary rules in the security group but note that it resolves to the public IP address. So this is a key way that this public DNS works. **If you attempt to ping it from the public internet, it resolves to the public IP address. If you attempt to ping it from with inside the VPC, it resolves to the private IP address and that means that you can use the same DNS name, whether you're in the VPC or on the public internet and it will always talk to the best IP address of that EC2 instance whether it's private or public.**

I'm going to go ahead and I'm going to restart this instance. So I'll go to instance state and then reboot. Now that will only take a couple of seconds but when I reboot the instance from the console, the IP address won't change. **So this is critical to remember the public IP address of an EC2 instance of the public IP version four address doesn't change when you reboot** **an instance** but what happens if I go to instance state and then stop? Is that this public IP address is deallocated from this EC2 instance, **EC2 instances can have two different types of public IP version four addresses. They can have what are known as dynamic addresses and dynamic addresses change. They change when the instance moves between EC2 hosts when it stopped and when it started. So when the instance is stopped, the IP address, the public dynamic IP version four address, is deallocated from the instance.** Now when I do the reverse I want go to instance state and start what will happen is a new public IP address will be associated to a brand new public address but in addition, because the public DNS is based on that IP address, the instance will also get a brand new public DNS.   
**By default all public IP version four IP addresses that could be associated with EC2 are dynamic. They change if the instance is stopped and started, or if the EC2 host that in instance is running on changes**.

Now there is another type of public IP version four address that could be associate it with EC2 instances and that's known as **elastic IP addresses. With elastic IP addresses these are IP addresses that are valid for the entire AWS region.** **So it's not an availability zone specific thing.** AWS have a pool of these elastic IP addresses and the first thing that you need to do is to allocate a new address with your account and you can do that by clicking allocate new address. Now it is possible to associate your own IP addresses with your AWS account but for this demonstration, I'm just going to pick the Amazon pool. So I select that and then click on allocate. Now that will result in an elastic IP address being allocated with my AWS account in this region, **it's important because if you don't associate this address with a resource, you're actually charged for it. So the more on allocated elastic IP addresses you have in your account the more cost associated because public IP addresses are in short supply AWS don't want you to waste them and so if you're not using them, you can go to actions and then release IP addresses.** **So releasing it just moves it out of the pool allocated to your account and moves it back into the general AWS pool for that region.**

So I'm going to go ahead and allocate another IP address once I've got an elastic IP address. I'm able to associate this address with another resource in this case and EC2 instance. So I'm going to go ahead and select the EC2 instance, the public instance that I was just demoing a moment ago and then I associate this elastic IP with a private IP address on that EC2 instance so that architecture is critical to understand **when you're associating elastic IP addresses what you're actually doing is matching it up with the private address.** Remember, t**hat doesn't change**. So you're creating a link between the private address and this elastic IP address. I'm going to go ahead and associate that now when you associate on elastic IP address with an EC2 instance, what happens is any existing dynamic or changeable IP addressing is removed so it loses that existing public address. In its place, the static elastic IP is added and that's what this is 3.211.112.173 is the elastic IP address. We get a new public DNS name which matches this IP address and unlike the previous IP address that we had, this one is with this instance for its life time. So I can stop and start this instance I can restart it whatever I do this IP address will be with this instance for its lifetime. **I can, though move this elastic IP address between instances.** So if I wanted to have a consistent IP address for my application so users could connect to this specific IP address or this specific DNS name, then I could take that IP address a map it to another instance, maybe I was doing an upgrade. Maybe I was dealing with a fault, and I wanted to take a new AMI make a new instance and associate this elastic IP with this new instance. Well, I could do with that elastic IP addresses are separate entities. **They could be moved between resources so moved between EC2 instances.** Now, if I remove this elastic IP from this EC2 instance, **I'll disassociate that address with this instance. Once I've done that, I'll release it from the account, so I'm not being charged for it but what happens when I do that if I go back to instances, is because this instance is a public instance, it has a public IP.** When I added the elastic IP, it removed the existing public IP when I removed the elastic IP, it adds an additional IP and this is another dynamic IP address. So this IP address will change now if I stop and start the instance. So **whenever you add an elastic IP to an instance it replaces its existing IP. When you remove it, It's given a new, dynamic public IP not the same one a brand new IP address.**

**So with public instances, you've got the private addresses which never change .You've got the private DNS which never changes and that's the same as with private instances. In addition, you've got the normal type of public IP address, which can change whenever the instance is stopped and started and associated DNS name that can change also whenever the instance is stopped and started and you've got the option for an IP version four elastic IP address which is a static address which never changes.**

IP version four is limited and it's actually running out of public IP addresses, which is why we generally use private ranges for most things. IP version six was designed to fix that shortage and so all IP version six addresses inside AWS are public. You don't pick which addresses you want to use they're allocated by AWS but if your VPC is enabled for IP version six and your subnet is enabled for IP version six, then your instance, will be allocated an IP version six address at launch and it'll keep that until terminated.