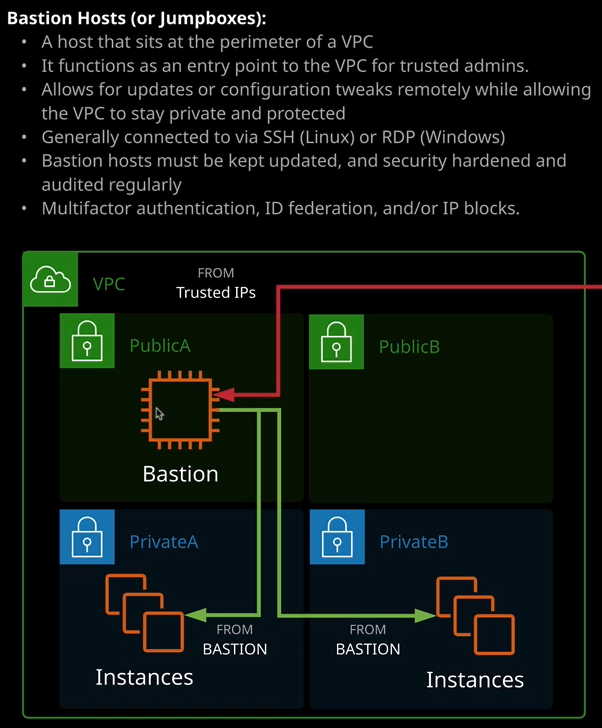
**Bastion Hosts**



Now bastion hosts are also known as jump boxes. Now the concept of a bastion host or jump box is pretty simple. It's generally either a physical or virtual machine, or in this case, an EC2 instance, it occupies a network that's publicly accessible. So in this particular case, it's a public subnet inside the VPC and it's used to be able to manage an otherwise private VPC or a VPC which is highly locked down. Now the premise is that you might have one or more of these instances in your public subnets, and you can make everything but that bastion private and only have the one entry point. So if you've got instances running in private subnets that you need to occasionally connect into maybe to perform configuration changes or software updates, then rather than having these being available from the public internet, **you can have the single bastion host, which is accessible, and then be able to connect into that bastion host, and from there, connect to these internal resources. It essentially functions as an entry point for trusted admins to connect to for system updates, configuration updates, or any other admin tasks. Now you can lock these bastion hosts down. You can conduct regular security audits and restrict who can connect to them using security groups or network ACLs which we'll talk about in the last theory lesson of this topic. Or alternatively, you can make it so that private keys are used for authentication or corporate logins are required to access the bastion host. There's lots of different ways that you can lock it down to ensure the only trusted people can access it.**

**Now bastion hosts generally run Linux and are accessed via SSH but they can also run Windows and be connected to using remote desktop. Now they're often even secured with multi factor authentication, which means they're really hard to breach. The whole point of a bastion host is that it is going to be the thing that's exposed to the public internet, so you need to make sure that it is heavily locked down and it's secured with every tool that's available from inside AWS.**

So because in the last lesson I created a VPC with two public subnets, I'm in a position where I can actually create a bastion host. So I'm going to do that. I'm going to move to the EC2 console and create an instance in one of these public subnets. So I'm going to launch instance. I'll use Amazon Linux 2, I'll select to use a T3.micro, and I'll go to configure instance details. Now because I've got a default VPC in this AWS account and in this region it will select the default VPC in the North Virginia region but because I've got this custom VPC I can select the drop down and change it to the VPC demo. Once I've done that, I want to select one of the public subnets. So either public A or public B and for now I'll go with public A because that matches the diagram. Now**, because I've configured this as a public subnet, remember, it's automatically assigning public IP addresses and so the default here is to automatically assign a public IP.** So that's great. That's what I want. So everything else looks good. I'm going to click on next. The storage, in this case, we can leave as default. It's plenty for this demonstration, but if you did want this bastion host maybe to store administration tools or backups that you're wanting to copy back to your business premises, you might want to add an additional volume but for now, I don't need to do that. I'll accept the default. Now it's always best practice to tag resources specifically, in this case, a name because it is an instance that will have a different security profile. **So we need to make sure that it's immediately identifiable even for inexperienced admins.** So key is name, value is **bastion**, and then I'll move to security groups. Now for security group, this is fairly important.

Remember earlier in the course when I was talking about EC2, I talked about how **security groups operated at a higher level of the OSI networking model. Security groups are capable of referencing logical resources so other AWS products.** Now at a basic level, what I want to do is create a brand new security group for this bastion instance because I want to identify what can get in and out of this bastion at a glance. I want to call it bastion because it might have different security rules than these other instances. Remember, its security profile is different. So in this case, I'm allowing SSH connectivity from the public internet. I might want to narrow this down, which I will be doing. So I'll change this from custom to my IP. In this case, I'm only allowing SSH now from my particular IP address. Now because security groups offer this logical resource referencing capability because I've got a dedicated group for this bastion host. When I come to create instances inside the VPC, I could allow any connectivity from this bastion security group and that means if I create additional bastion hosts they would also have this security group, and it would automatically allow connections in from other public subnets. So **by using this logical referencing capability of security groups, you significantly reduce the admin overhead.** So that's why I want to create one that's specifically named, not shared with anything else because it will have some custom rules and I will want to reference it across this VPC, I'm going to go ahead and click on Review and Launch. Launch the instance and I'll be creating a brand new key pair and I'll call it VPC Demo. I'll need to download the private component of the key pair, which I'll let it do. So that's vpcdemo.pem. Once I've done that I'll launch the instance. Now I'll give that a couple of seconds to create and once it has, you'll see that it's got a public IP address. So I'm going to right click and hit Connect. I will need to modify the permissions of the PEM key. Which I'm going to do is I'm going to copy that into my clipboard, move across to my terminal, paste this command in to address those permissions, move back to the console, get the connection command, copy that into my clipboard, moved back to my terminal, run that command, accept the authenticity of the host and then I'll be connected to this bastion host and once I'm in, I can use this as a jumping point to get to other EC2 instances inside the VPC that I haven't yet created but I will be creating additional EC2 instances and connecting to them over the next couple lessons as I demonstrate other networking capabilities inside a VPC but for now I have demonstrated that I can create this bastion host inside a public subnet and the all of the configuration that I've done in the previous lessons is working as expected.

Now, one thing I want to draw your attention to is notice **how this instance isn't given a public DNS address.** In the previous section of the course where I talked about AWS's Compute Services. I talked about how EC2 instances are given both a private IP address and the private DNS address, and so both of those are present but I also mentioned how it should get a public IP address and the public IP version four DNS address, and it hasn't done. Now the reason for this if I open VPC in a new tab, so that'll take a couple of seconds to load. Once it does, I'll go to your VPCs, I'll select this VPC, click on Actions, and then edit DNS hostnames. Now it doesn't have this enabled. **There are a number of DNS options that you need to specifically enable on a custom VPC these are enabled by default on the default VPC but you do need to enable them if you create a custom VPC.** So I'm going to do that now. I'll click on enable DNS hostnames and I'll do the same process for DNS resolution. So I'll edit the settings for that. Make sure that's enabled and click Save. At this point, if I go back to the EC2 console and hit refresh, we'll see that we've now got this public IP version four DNS name associated with the instance. **So there are a couple of intricacies that you need to be aware of if you are creating a custom VPC and one of those is that you do need to explicitly enable the DNS hostnames**. Now it doesn't impact our ability to connect as I demonstrated moments ago you can always connect using IP, but as a best practice step. I always like to enable DNS hostnames on any VPCs that I create, but as I've just illustrated if you do create a custom VPC, it is something that you'll need to do explicitly. Now questions in the exam might involve bastion hosts or jump boxes just remember that architecturally **they're services that you deploy into a public VPC in order to get secure access to private resources.**

What you'll be learning about in the next lesson is how you can give private resources inside the VPC public access to the internet in an outgoing way, but not allow them to be accessed from the internet. So if you remember, **an internet gateway allows this two directional access so two the internet and from the internet but sometimes you want to create private resources that can access the internet for things like software updates but not be accessed from the internet and that's where a product called NAT gateway comes in handy but if you want to connect to these resources for administration, you can't connect to them using the internet gateway because they don't have public IPs. You can't connect to them through the NAT gateway because that's not a supported feature, and so bastion host give you this functionality. They let you have this gateway EC2 instance, this bastion host this jump box that you can connect to and then from there hopping to your private resources.** So that's the type of question that you'll see in the exam that references bastion hosts or jump boxes. That's everything I wanted to cover. That's all the theory about bastion host. Next up we'll cover NAT Gateways. So giving private resources outgoing-only access to the internet. So go ahead and mark this lesson as complete, and when you're ready, join me in the next.