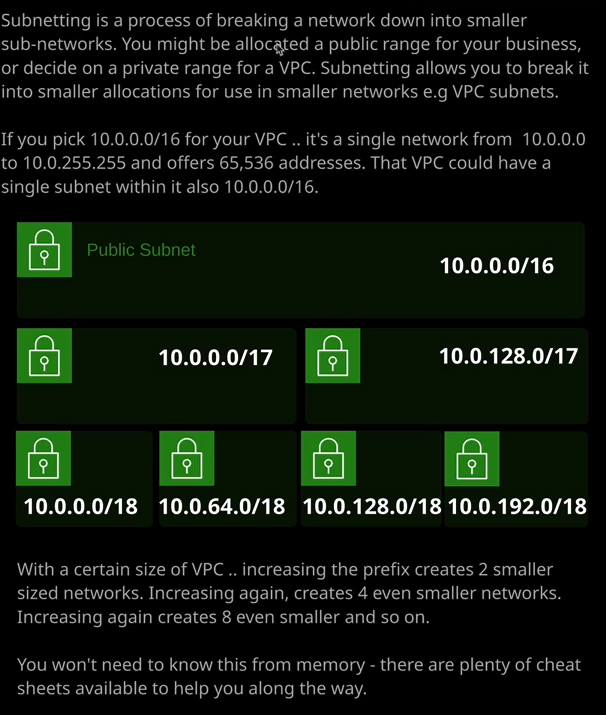
**Subnetting is a process of breaking a network down into smaller subnetworks.**

You might be allocated a public range for your business or decide on a private range to use inside a VPC and subnetting allows you to break it into smaller allocations for use in smaller networks. For example, subnets inside that VPC. So in the next topic of the course, you're going to be looking at how to do private networking inside AWS. You can create a VPC or a virtual private cloud and inside that VPC you can create one or more subnets than having the experience to be able to break up an allocation of IP addressing into individual subnets will make everything a lot easier.



Now in the last lesson I talked about subnet masks and prefixes and within AWS, you'll tend to see the prefix notation used. So recall from the last lesson that the prefix is actually the number of bits inside the binary version of an IP address that represent the network. In essence, the prefix determines the size of the network, so a /16 prefix indicates that's 16 out of the 32 bits so the first two octets for the network. These octets need to be static, so 10.0 represents your network in this notation, 10.1 would be a different network, 10.2 a different network again, but with a /16, it means that anything beyond the first sixteen bits are yours to carve up however you want inside your VPC or inside your physical network. So 10.0.0/16 is a single network from 10.0.0.0 all the way through to 10.0.255.255. So that offers 65,536 IP addresses and, of course, two of those are always reserved the network and the broadcast, so that gives you 65,534 usable IP addresses for nodes and hosts.

Now you haven't covered this yet, but inside AWS, **you could create a VPC and have it allocated this IP address range of 10.0.0.0/16 and then inside that you could create a single subnet, which also uses 10.0.0.0/16 and that would be perfectly valid if you just needed one big subnet to put in hundreds, thousands, tens of thousands of EC2 instances then that would be a perfectly valid way of doing it but inside AWS, as you'll learn over this section of the course, you should really split up the VPC into individual subnets because a subnet is inside an individual availability zone. So by implementing different subnets, you can spread your infrastructure across different availability zones, and that allows you to build in high availability into your infrastructure.** So to do that to add this HA you need multiple subnets and to create multiple subnets you need to understand how to take the CIDR range that your VPC will be given and split it into multiple subnets and that's what I want to cover very quickly in this lesson.

So, essentially, the way to do this is actually really easy. Let's say you start with 10.0.0.0/16 which is a fairly common CIDR for a VPC inside AWS. Well, you can actually split that into two individual networks by just incrementing this prefix. So by incrementing the prefix, what you essentially do is create two networks of half the size, so that would give you 10.0.0.0/17 and 10.0.128.0/17. So all you've done is you've taken your network, which remember, is 10.0 and then anything and you've chopped it in half. So you've got 10.0 and then zero all the way through to 127 and then 10.0 128 all the way through to the end. So you've half the size of the CIDR you created two smaller networks. So by incrementing the prefix you've chopped in half and created two smaller networks but you can do that process again. You can take the half size networks and chop them into two even smaller networks. So remember this first /17 network that we've created. It starts at zero in this third octet and it ends at 127. So by taking that, we can half it again into two /18s. So the first one starts at zero, and it ends at 63.255 the second one starts at 64.0, and it ends at the end of this range. The process is again the same for the second one. This starts at 10.0.128.0. So the first smaller subnet will also start at that same range but instead of going all the way through to the end, it will stop at 191.255. The second small one will start at 192. So 10.0.192.0 all the way to the end. So each time you increment this prefix, you create a subnet half the size. Now there's a little bit of practice to this. So if I'm talking about it, I can do it fairly quickly in my head. What I'll make sure I'll do is I'll include a link in the lesson description that will show you some common quick ways that you can do this in your head. So I'm not doing the whole mathematics on the fly from memory, what I've remembered is a number of rules for this. So I know that if I half the size and increment this to 17 I know that it's half of this third octet. So that allows me to do this and this really quickly. What I want you to do is I'll give you the links needed and you'll be able to do with this in your head. Almost is quickly to start with, and then over time you'll get to the point where you can do it just as quickly as me and if we can do that, then the exam will be really easy because you won't need to worry about making mistakes. You'll be able to really quickly identify valid and invalid subnet configurations. Now this my only be one questions, so don't get scared that it's going to be extensively needed across the exam. It's not. It might just get you one correct answer that you might otherwise struggle with but if we can do that, if we can cope with every eventuality in the exam that we can get you a really good exam grade. So essentially subnetting is the process of taking these larger networks and splitting them up into smaller individual networks. So two /17 inside a 16 and then four /18 inside a 16. So two of them inside each 17 and we could do this even further. We could go to eight individual networks, and they would each be /19. We could go to 16 /20s, 32/21s, 64/22s, 128/23s, 256/24s, and so on. By incrementing this prefix, we create more smaller networks. Now why this is important, why I'm going to include a link, which has some hints on it, is that in the exam you need to be aware of how many IP addresses you'll have for each of these sizes. You don't need to know it from memory or extensively, but just a couple of key values because you might face a question where you need to deploy on elastic load balancer, which is an AWS product into a particular subnet and you might need to know the minimum number of IP addresses that it needs to deploy if you know the minimum number of IP addresses a load balancer needs, and you know the size of these subnets, then you can quickly answer that question and by the end of the course, that's what I want to achieve.