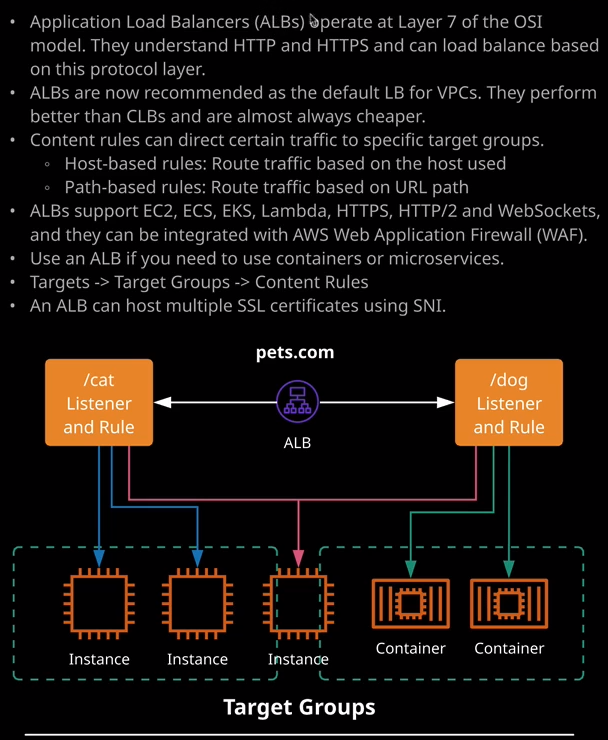
application load balancers.

So application load balancers or ALBs are the new type of load balancer it designed to replace the classic load balancer for most situations. So they're known as ALBs for short, and they operate at layer **seven of the OSI seven layer networking model and that means that they understand the HTTP and HTTPS protocols and can load balance based on this protocol layer**. Now that's an important distinction to understand. With classic load balancers they operated at layer three and four. While they could perform health checks at layer seven, they weren't capable of understanding the protocol details of layer seven. So they couldn't, for example, decided to do a certain thing if the URL contained the word cat and a different thing if the URL contained the word dog and that is what I want to highlight in this lesson the **differences between an application load balancer which is layer seven capable, and a classic load balancer which is layer three and four capable**.



Now application load balancers are created in much the same way as a classic load balancer. So I can click on load balancers, go to create load balancer, and instead of selecting classic, I'll select application. Now much of the configuration is exactly the same as a classic load balancer, so I'll need to give it a name, which I'll pick ALB for application load balancer. The scheme is to pick between an **internet facing load balancer or an internal load balancer, the differences between those are the same as with classic load balancers. It's all about the IP addressing the load balancer nodes and whether they're public or private.** For this demonstration it's going to be an internet facing load balancer. Now, an application load balancer is capable of supporting both **IP version four and dual stack**, which includes version four and version six. Version six is not something that's deployed across the board with all AWS services, or even all on premise networks yet. So in most situations, at least right now you'll be picking IP version four. Now, an application load balancer still uses a listener configuration. So you're still defining the listener configuration exactly what ports and protocols the load balancer can listen to. Now with an application load balancer, what AWS have done is taken the functionality offered by a classic load balancer, extended that, but then split that out into **two different load balancers. The application load balancer, which is capable of operating an application level and the network load balancer which handles network level load balancing.** So with an **application load balancer, the only two options that you have at this point are HTTPD and HTTPS. So an** application load balancer, and this is especially important for the exam is generally going to be used in situations where you're attempting to load balance either standard HTTP or secure HTTPS. Now, for this demonstration, I'm going to select HTTP to keep things simple, and it's going to be on the default port of 80.

Now, just as with a classic load balancer**, you do need to define which availabilities zones the load balancer will use and this, of course, determines which availability zones and which subnets get the load balancer and nodes placed in**. So in this case, I'm going to pick the same as the classic load balancer that's U.S. East 1a, U.S. East 1b, and U.S. East 1c. Now, if you do have multiple subnets in those availability zones, you will be able to select those in the dropdown but because I'm using the default VPC, which only has one subnet availability zone, I don't get the option of selecting multiples. There is only one subnet per AZ. So I'm just going to go ahead and make sure pick the same AZ as a classic load balancer and then I'll move on to configuring security settings.

Now **it's the security settings where you would configure SSL and any SSL related configuration** but because I'm limiting this demonstration to HTTP I can safely skip past this, but I will talk about SSL certificates later in this lesson because there is a very important difference between application load balancers and classic load balancers in terms of how they support SSL. For now, I'm just going to move on to configuring security groups and as with classic load balancers, **you need to create a security group and that's a security group that is associated with the load balancer** and the same logic applies. **It controls both which ports and protocols can get to the load balancer but also, it can be used to reference so that you can control direct access or not to your instances.** So I'm going to go ahead and call this one alb\_securitygroup for application load balancer underscore security group and I'll set that for the description as well. It's already set to allow port 80 TCP which is HTTP so I can go to next configure routing, **the major difference between how an application load balancer works and how a classic load balancer works is it introduces the terms target and target groups**.

**So a target is something that you can point a connection at.** Now because an application load balancer is newer. It supports lots of different target types. So I've got EC2 instances which I have demonstrated so far we've got **ECS and EKS which are both container service** is that I'll cover elsewhere in the course. Application load balancers can also point at Lambda functions**. You can actually have a Lambda function responding to connections**. That's something that's beyond the scope of this course but it's useful to understand that that is a capability of application load balancers and because of those different types of targets, the way that they're organized is a little bit more complex than classic load balancers. **So you've got the concept of being able to create a target group, which is a collection of targets.** **So targets are these different types of compute services application load balancers support. You group these together in targets, and then a target could be pointed at for application load balancers to direct connection towards, so you don't point them directly at instances**, **you point them at target groups.**

Now, **individual targets can be members of multiple target groups. So that's a lot more flexibility and** I've got that shown on the diagram on the right of my screen. So for now, what I'm going to do is I'm going to create this application load balancer and I'm going to point it at the same three cat web instances that created in the previous lesson. So I'm going to go ahead and create a new target group, and I'm going to call the target group Cats. Now the type of target inside this target group, I'm going to leave for now as instance. For target groups, you can directly reference either instances, IPs, or Lambda functions, but to keep things simple for now, I'm going to pick instance, which will mean it will operate in much the same way as a classic load balancer. Now, the protocol that will be used to connect to this target group is going to be HTTP and it's going to utilize port 80. As with the classic load balancer, you can create a health check, and I'm going to do that so the protocol is going to be HTTP but remember, because I've got some dynamic content on those cat web servers are need to use index.php and this is going to mean that this is what's going to be checked with health check. So the instance/index.php. Now just as with classic load balancers, you're able to define a number of advanced health check settings. Some of these will be familiar from the previous lesson. So the healthy threshold defines the number of checks that are required before an instance is viewed as healthy, I'm going to change that to two. The unhealthy threshold defines the number of health checks that need to fail before the instance is marked as out of service. **The time out is the amount of time that the load balancer will wait for a success or a failure state from the instance.** So remember, in the previous lesson, I set this to four. **The interval is the amount of time between health checks and as with the previous lesson, I'm going to set this to five to keep things quick and then the success code, this is an extra piece of functionality. So this allows us to define the success codes that we're expecting from the instance if it's reporting as healthy. The default of course 200 which is a HTTP status code of okay.** It means everything's good, but you are able to define other codes and this is e**specially useful if you got some custom application functionality where you want to deliver a slightly different code.**

Now, another new feature **is you're able to define a port on which this health check occurs. The default is the same port that the traffic is being directed towards, and that was the only option with classic load balancers. With application load balancers, though, you can override** **that and define a specific custom port for these health checks.** We're going to keep things simple, of course, and not do that but I want you to be aware the functionality exists. So next I'm going to move on, and I'm going to go to register targets.

Now, remember, **because the application load balancer works using target groups, what I'm essentially doing at this point is registering these instances inside this target group**. So I'm going to go ahead and pick each of the cat web instances and add them to this target group as registered instances. I'm going to go ahead. They're added, I'm going to go to next review, and then create. Now that is going to create the load balancer but what's going to happen straight away is because in the last lesson, I restricted what could communicate with these instances to only the security group of the classic load balancer. I need to follow the same process. Go back to instances, select them, select the cat web security group, click on inbound, go to edit, and then a need to add a brand new rule. Also for HTTP but this time I'm going to add the security group of the application load balancer. Once I've done that, I'll click on save on that'll just make sure that the application load balancer it on its health checks are capable of reaching these EC2 instances.