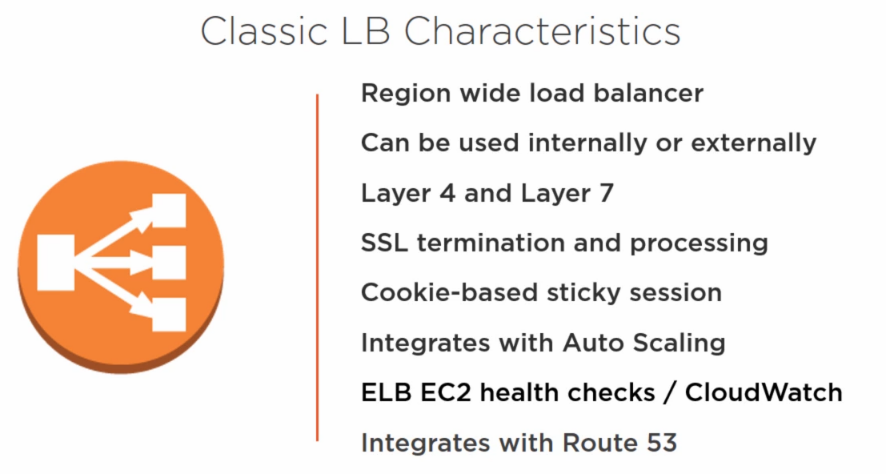
**Classic LB Characteristics**



1.) a.) Classic Load Balancer is **a region wide load balancer**, meaning if you are in a particular region that has more than one Availability Zones you can create one Classic Load Balancer and this Classic Load Balancer is going to load balance traffic across all of the different Availability Zones.

b.)ELB is a platform service, meaning you don't have to manage it,you don't have to worry about high availability, you don't have to update it, nothing, it's hands off, **you configure it, you point to your instances, there's a few other configurations,** and ba-da-bing, ba-da-doom, you're off to the races.

c.)Region wide load balancer, very cool, this is not a VM, it's not an appliance, it's very seamless from your perspective.

2.) This can be used as an **internal or external load balancer**, so if you want it to be internet facing, you can use that, if you want it to be a load balancer for internal workloads only, you can use that as well, it can be used in any one of these configurations.

3.) This is a **Layer 4 and a Layer 7 load balancer.** Now, it's more of a Layer 4 load balancer with some Layer 7 capabilities.

4.) For example, **SSL termination and processing.** Now, why is SSL termination and processing important? It's important because if it's not done at the load balancer level then it's going to be done at the instance level. Now if you're offloading termination and processing to the individual instance, what happens? Well guess what? CPU processing goes up, right?

Because now these instances have to process the SSL certification. **With ELB, that functionality is moved off to the platform to the Elastic Load Balancer, which means those instances are going to perform better, why, because they are not doing extra functions like SSL termination and processing.**

5.) **Cookie-based sticky session**. Now, it's debatable whether or not this is a good thing or a bad thing and it's available because it's available.

It essentially allows you to configure it so that a user that's connecting to a particular instance through a web browser is always routed to that particular instance, hence the name sticky session, because although there's a load balancer, the user is always connected to that instance, obviously in the event of the failure things change, but they're always connected to that instance.

Now AWS best practice will dictate that you use a database instead of using the ELB capabilities for sticky sessions, that way if you need to fail over you're failing over seamlessly and you don't necessarily impact the user. However, this feature is available on the load balancer for use.

6.) **integrates with auto scaling,** which is also super cool

in the event that there's an instance that has gone rogue, or that you have to pull out of the rotation, auto scaling immediately detects that you now have two servers instead of three, and automatically provisions an additional third server.

Converse there consequently what you can do is if you experience very high load against your instances, the auto scaling feature will provision additional instances to offload the high traffic that you're experiencing and then it will, it will shut them down and it will terminate them at a later time.

7.) ELB also gives you **health checks** for your EC2 instances.

Sometimes an EC2 instance will respond, however when the user tries to access it while the server is up and running, it is not necessarily processing the application or the page that the user is trying to access. With health checks you have the ability to do more advanced health checking.

**for example:** you might specify that you want to query a particular page on an instance and wait for a response, that response might be something like the page loading successfully, if the page loads successfully then that's additional assurance to ELB that this particular instance is healthy, and then you can put some metrics around how often to query it and when to take it out, how many times should it respond with healthy or unhealthy.

8**.) ELB integrates with Amazon CloudWatch.**

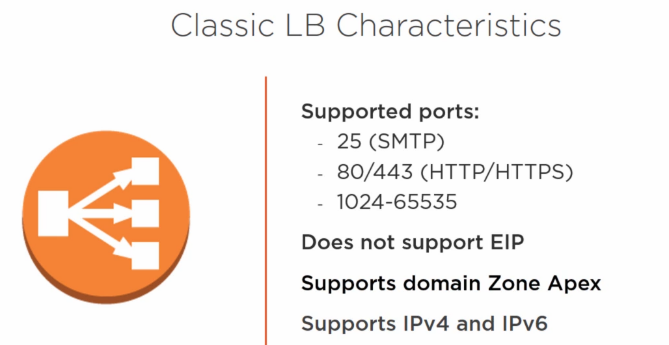
What you do when you integrate ELB with CloudWatch is now you have the ability to have advanced metric load balancing.

For example instead of just doing a round robin and sending users or sending load to instances in an equal way or based on some percentage that you specify, with CloudWatch you can take it one step further and apply load balancing based on metrics like CPU-like memory like this, etc., there's a ton of metrics that you can configure with Amazon CloudWatch.

9.) **integrates with Route 53.**

Route 53 is cloud-based DNS. Now what you do there is you can do cloud-based DNS load balancing if you have multiple ELBs that are in multiple regions.

**ELB Characteristics:**

****

**ELB ports supported:**

1. supports port 25 (SMTP),

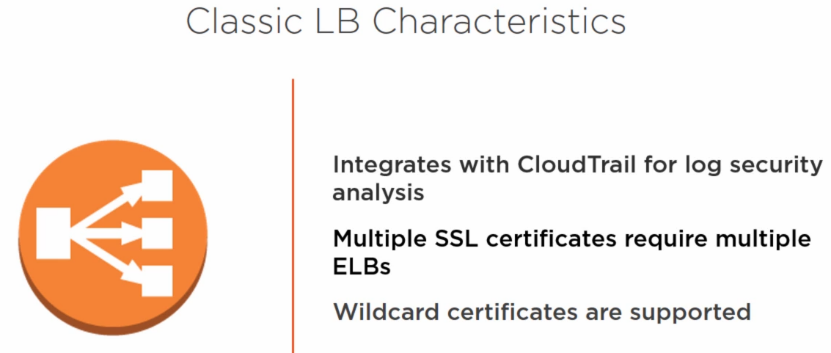
port 80/443 (HTTP and HTTP) and

anything in the range of 1024-65535.

1. **NOTE:** you cannot assign an EIP or a public IP address to your ELB, it only works with a DNS,
2. Also, support domain Zone Apex.

For ex: You can point to domain without necessarily having to put www., so you can point to Pluralsight.com without doing [www.Pluralsight.com](http://www.Pluralsight.com).

1. ELB supports IPv4 and IPv6



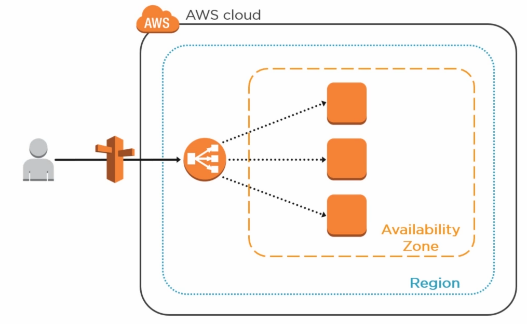
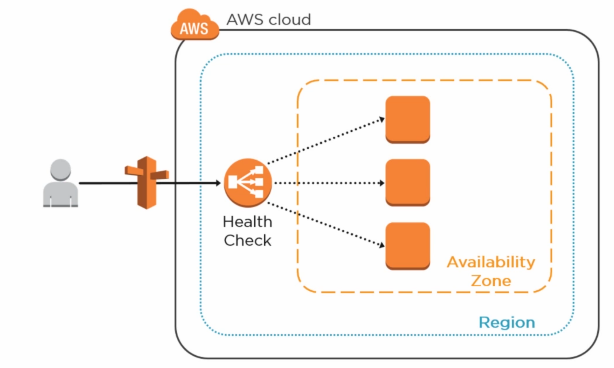
1. **ELB can integrate with CloudTrail for log security analysis**

So that's also important because all the configurations that you're doing on ELB, you can enable CloudTrail to track that, and then you can dump that into CloudWatch and configure monitoring and alarms around that, and we'll cover the connection or the configuration between CloudTrail and CloudWatch later on.

1. If you're going to need multiple SSL certificates, you will need multiple ELBs.

**NOTE:** What that means is that one SSL certificate per ELB.

Now, a lot of you might have something that's known as a wildcard certificate within your organization. What that means is for example \*.Pluralsight.com, which means any child domain that we want to give with the domain Pluralsight.com would be covered under that wildcard certificate, but if for example you had blog.Pluralsight.com assigned to a particular ELB, you wouldn't be able to do forms.Pluralsight.com on the same ELB, you would need to at that point configure a different ELB for it.

**Classic LB Scenarios**

Basic deployment of an ELB.

1.)In this case, a user going through a Route 53 and hitting the load balancer. Now remember, the load balancer cannot be accessible via an EIP or an Elastic IP or a public IP address, at least not as of this recording.

2.)So what happens is it has a DNS, now that DNS is resolved through Route 53, it could also be resolved through a different type of DNS that you're using, but inside of AWS, Route 53 is what's going to manage the DNS names for these different assets and these different components

3.) There is no Elastic IP address that's associated to the load balancer, there is a DNS name that is associated to the load balancer.

4.)Now again this is a very traditional load balancing scenario deployment, the user hits the load balancer, the load balancer is going to take a look at the group or the number of nodes that it has, and it's going to direct the user to the least busy instance or server that it has.

5.) Now you can couple or **integrate your load balancer with health check.** Now what health check will do is it's going to constantly exam the instances that are being load balanced.

Now this could also be in an auto scaling group, and again depending on how you've configured that autoscaling group, based on metrics.

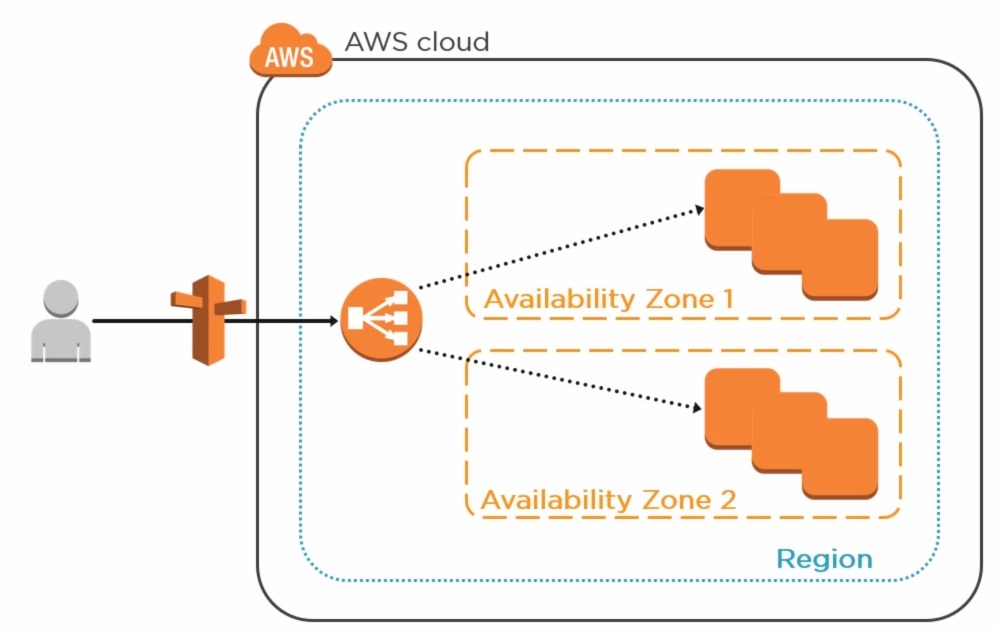
**for example**, if the CPU, the collective CPU of the group hits 80% you add more, if it drops below let's say 70% it removes an instance, again this will have an effect both from a cost perspective, but also from a performance perspective.

6.) So let's say health check detects that you have a particular instance that has gone bad, what health check will do is then remove that particular instance from the load balancer, so the load balancer is no longer sending traffic to this particular instance.

7.)It will only reintroduce this instance once the reasons for why it was removed are addressed, so for example if the CPU hit 80% and as a result health check removed it, then that CPU needs to fall below 80% before it can get back into the rotation.

8.)Now again, this is more the function of CloudWatch, so from a health check perspective there's got to be something wrong with that particular instance, maybe it's sending a message to it and it's not receiving a reply, maybe the ping is not coming back with a reply, so health check will deal with more of these types of scenarios…

9.) **but if you couple health check with auto scaling then you have the ability again to do more advanced things from an Elastic Load Balancer perspective**.

**10.)The next scenario** ELB a platform service that AWS offers, which means you don't have to maintain high availability, you don't have to worry about scalability, you don't have to worry about software updates, nothing, it's a service that AWS offers you, and in this case what's happening is that because it's region wide, the **LB is region wide, you are now load balancing across instances that are in different Availability Zones.** 

11.)Now also remember that we talked about the fact that an ELB can only have one SSL certificate, so you can't have two SSL certificates on the same ELB, you can however have a wildcard certificate on a particular ELB, so if you wanted to on this particular one that we're looking at, you could have a wildcard certificate for star.Pluralsight.com, or star.wiredbraincoffee.com, and any child domain at that point would be fine.

However, if you do for example onlineordering.wiredbraincoffee.com, then that SSL certificate would be tied to one ELB. In order to have another SSL certificate possibly maybe for customer service, that .wiredbraincoffee.com, you would need to deploy a second ELB. Now, you can also deploy a second ELB because maybe you're looking to have high availability and you're not comfortable with one ELB.

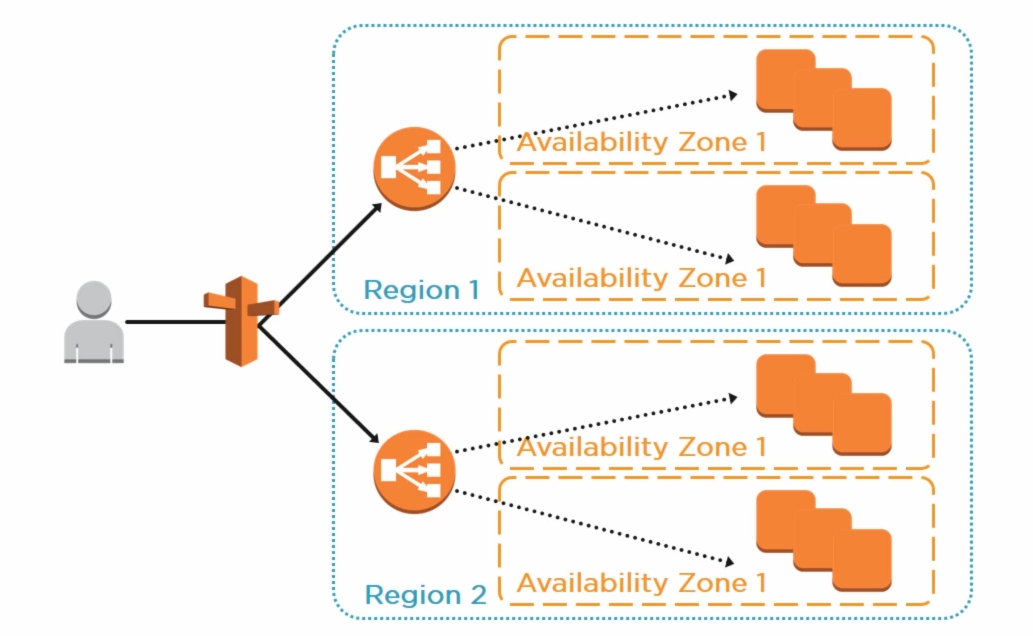
12.) Now the takeaway here is remember this is a platform service, so whether you deploy one or you're deploying more, it's not really going to matter because if there's going to be an issue with the platform that's offering the ELB as a service, regardless of how many ELBs you deploy, **all of the ELBs will be effected.**

13.)Again, this is at the software layer that this is going to happen. However, from a technical perspective, if you wanted to deploy a second ELB and then load balance across, or use health check for example for in the event that one ELB fails, Route 53 and health check will start to route the traffic through another ELB, you can technically do that, however the takeaway here is again remember this is a platform service and the most likely event that's going to happen is a software update that gets rolled out that will affect the software that runs the platform for ELB in particular, so we'll most likely affect both..

However you can deploy it and you can do the scenario, the effectiveness is, it's not recommended but it's not very effective just because again there are no two distinct pieces of hardware, so this is something that we would apply **typically in our data center where** **we would have two pieces of physical load balancer equipment or two virtual appliances that are running load balancer, this would work more optimally there.**

**Here you can do it, not the greatest.**

14.)Now we've talked about deploying the ELB within the same region, now let's talk about deploying ELBs in different regions. So we've got Region 1 and Region 2, you've ELBs that are again load balancing across multiple Availability Zones and different regions, what you can do is again same type of scenario, **here you're doing DNS based load balancing, so** if a particular region goes down, here it's very effective to have a secondary region that's acting maybe as an active active partner, active passive partner, it's your fail overregion, whatever the case is, if one of the regions is having a problem, Region 1 for example, DNS will then forward the users to Region 2 and then it will start to distribute them based on the instances that are in Region 2.



**Creating and Configuring the Classic LB**

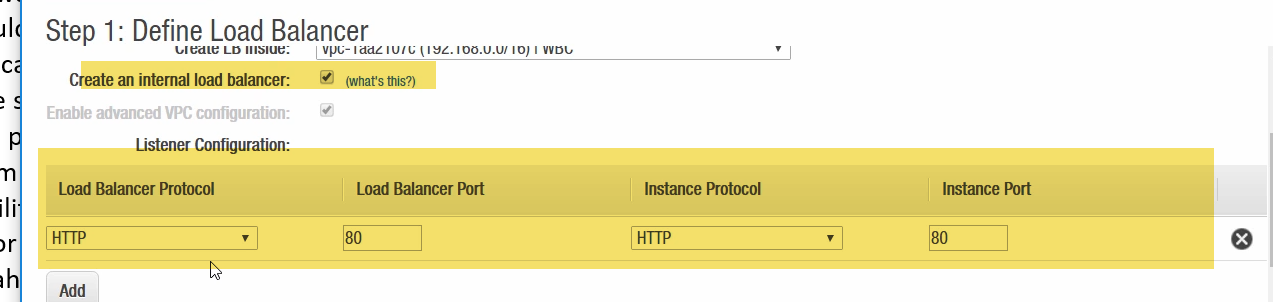
EC2🡪select Load Balancers🡪Create Load Balancers.

1.)External facing load balancer is going to have a public IP address, this is something that you would probably use for things like web servers, if you're trying to put these web servers in the DMZ for instance and you want your users to connect to them. Now, there will come a time when maybe you have web servers for internal use only, maybe this is for your internal use cases, and this is where you would come in and select an internal load balancer.

2.)Now because we've selected external load balancer on pretty much every exercise for the AWS course that we have on Pluralsight, I figured that with this lesson what we're going to do is go ahead and create an internal load balancer.

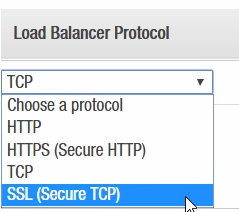
**The process is pretty much the same, except it's not creating a publicly facing DNS for this load balancer.**

a.)left the first thing that you'll be presented with is which port do you want me to listen on, this is when the users are connecting, which port should they be connecting on, which port is the load balancer going to be listening on?



Now best practice will dictate that you switch this to HTTPS, you would want all of your traffic to come in on the securest port possible, the securest protocol possible, so you would probably select HTTPS. Now in some instances, maybe for internal uses, HTTP might be acceptable, there are other cases where you probably want to select TCP and maybe put a random customized workload-specific port that this application is listening on, this is where you can do that.

you can do the same thing for SSL as well.



However, by default and in most cases, you're probably going to put this to 443. Now this is the port you're listening on.

b.) Over here on the right is the port that you want to forward on, so once the load balancer receives this traffic, how do you want it to forward it internally?

Now ideally if the application supports it you would probably want to switch this to 443 as well, so that the traffic is secure from the user all the way to the destination server or the destination service, so you would want 443 across the board. In many cases, or in some cases you'll find that some applications are configured by the software developers to only listen on port 80, so what happens is you're receiving the traffic over the internet, or internally over 443, and then you're forwarding this internally on port 80. I've seen this scenario before, this might come up again, ideally you would want 443 across the board.

4.)For the purposes of our example, and because we don't have an SSL certificate to upload in order to use port 443, I'm going to use HTTP port 80 for the purposes of this example so that we can bypass the SSL certificate upload screen.

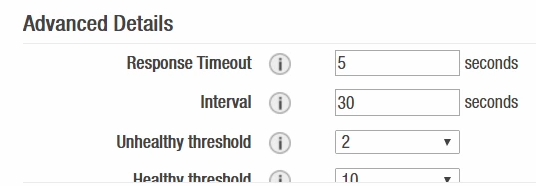
5.)in order to adhere to best practices, the AWS SLAs, you would want to have more than one Availability Zones participating so that you're load balancing traffic across multiple Availability Zones. Now for our purposes I'm going to select the two internal subnets that I have.

6.)How to **handle health checks against your instances?**

**So** for starters you want to choose the protocol that it's going to ping, you can either choose HTTP, TCP, HTTPS, or SSL. For our purposes, we're going to keep it at HTTP, and the Ping Port is port 80.

1. The health check at this point is it will constantly ping port 80, and if it receives a response back from port 80 then it's going to deem this particular instance as healthy and it's going to move on, but you and I both know that **there are a lot of cases where even if you receive a response back on port 80, that doesn't necessarily mean that the server is healthy. The server can be in a state where you can ping it but it can't actually service pages or anything else,** which is where
2. the second option here or the third configuration here comes into play, which is the Ping Path. The Ping Path is helpful because it allows you the specify a page. Now in this case we have /index.html, so what this will do in addition to pinging port 80, what it will do is it will also try to load this particular page. Now if a page loads it's going to deem this particular instance as healthy and it's going to move on, if this page doesn't load, then it's going to deem this instance as unhealthy and as a result it will take corrective action against it.

Now this can be as simple as an HTML page, or it could be an HTML page that has a bunch of scripts that then will load and do a bunch of things and then respond back with healthy or unhealthy.

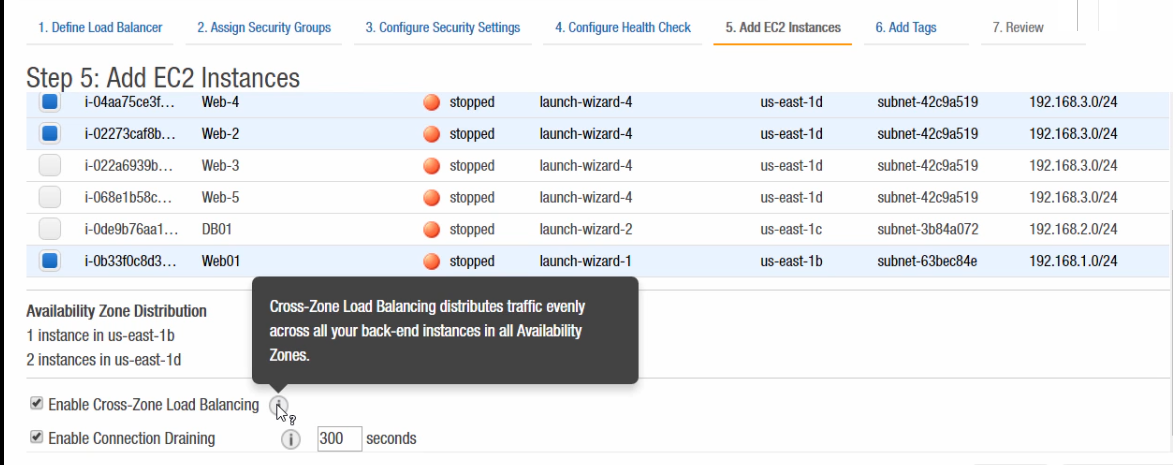
7.) Now down here you are configuring some **advanced options** as far as well how often or how long does this particular instance have to respond, so once you ping it how long does it have to respond? 

The default if 5 seconds, you probably want to lower this down to 2 seconds, which will make a little more sense, before the request will time out. Now the second option is the Interval, how often do you want me to probe this server? Right now it's set to 30 seconds, that's a pretty good metric, you can probably lower this to 15 or 10 if you want it to be aggressive, but 30 seconds is pretty good. So every 30 seconds it's going to probe the server and the server has at this point 5 seconds to respond.

Now, how do you deem a particular instance as unhealthy? Well, right now it's configured so that if it fails the test twice, so basically after 60 seconds or a little more if you're taking into account here the timeout, after 60 seconds or a little more it will deem this particular server unhealthy if it fails this twice.

Now you can obviously increase this to as much as you want here but twice seems to be fair.Now, once you've deemed something unhealthy and it corrects itself or you fix it, how does it get back, how does it get added back into rotation, and this is what this particular metric here deals with. So in order to be added back into rotation as a health instance, it would need to pass the test 10 times, and pass it healthy obviously, before it can get back added into rotation.

8.)select instances that are in different Availability Zones, so I'm going to go ahead and select a couple of these instances here that are in different Availability Zones, and this adheres to again AWS best practices that I'm spreading my load across multiple Availability Zones. If I scroll down a little bit I have two options.



The first option is to **Enable Cross-Zone Load Balancing,** now if you look at the little bubble here that explains this, when this is enabled the load balancer is going to probe the instances that are within the Availability Zones, and it's going to load balance traffic against these instances. If you uncheck this particular setting, what it will do is it will treat the Availability Zone as an instance, as a large instance, so instead of probing the instances within the Availability Zones and distributing traffic to them directly, it is simply going to load balance traffic against Availability Zone A, Availability Zones B, Availability Zone A, Availability Zone B, disregarding the instances that are in that Availability Zone.

**draining timeoutsetting,** so when you take something out of rotation you can either have it immediately disconnect all of the connections that are currently on it, or you can give it a draining timeout period.

You ideally don't want to disconnect your users immediately, if you're taking something out of rotation for maintenance purposes, you don't want to disconnect all of the users that are connected to it so what you would do here is by default this is enabled and the timeout value is 300 seconds, you can increase or decrease this.

**What this will do is it will take it out of rotation so no new connections will be accepted, but the current connections have a period where they're easily being drained out and removed from the server as opposed to immediate disconnection.**

9.) Tags are important for billing, for reporting purposes, we've already talked about that, you can certainly add a tag here, maybe the department tag, the group name, for our purposes

pretty big internal DNS name. If this was an external facing load balancer it would have an external facing DNS name that would probably be just as lengthy. 

Now, your users aren't going to be expected to memorize this right, and remember there's no IP address that's associated here, so what you want to do is go into your DNS and create a C name that is more friendly, something that they can remember,apps.yourorganization.com, or something like that that they can easily remember that then points to this A name so that it is not this complicated or convoluted.

10.)All of the configurations that we just created are available under this description, you can continue to scroll down, here under Port Configuration you can again change the port configuration, you can enable session stickiness which is something we talked about during the presentation, this is a Layer 7 configuration and functionality that you can enable if you wanted this load balancer to handle this, you can modify the security group if you choose to do that, under your attributes again all of these settings we've just finished configuring so you can come in here and modify them if you choose to do so.

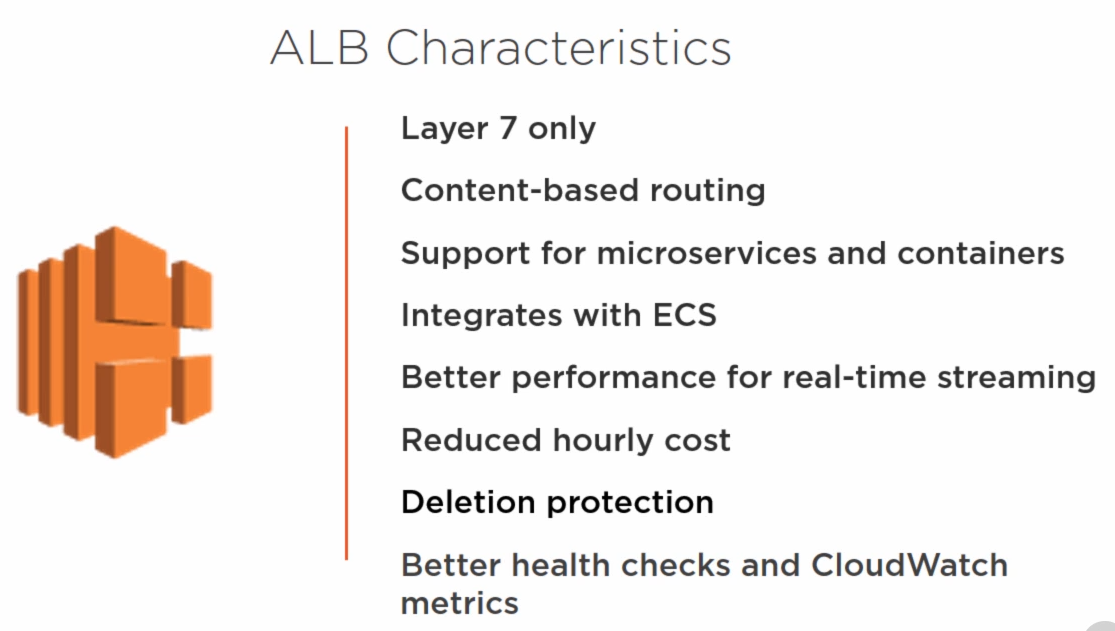
Under **instances**, these are the instances that we've configured, add, remove at your liking.

**Health check**, we've also just configured this, you can modify it if you choose to do so.

**Listeners**, if all of a sudden you get an SSL certificate and you wanted to switch from port 80 to 443, this is where you can come in and do it.

**Monitoring**, this will allow you to monitor the current health of your Elastic Load Balancer and the instances that are on it, you can also create alarms, we haven't talked about alarms yet but this is where you can add them as well, and then

**Tags** you can come in here and add tags.

**Application LB Characteristics**

1.) it's a Layer 7 only load balancer. When you compare this to the Classic Load Balancer, the Classic Load Balancer is primarily a Layer 4 balancer with some Layer 7 capabilities, whereas the Application Load Balancer is strictly a Layer 7 load balancer.

It is still a region-wide load balancer, very similar to the Classic Load Balancer, so you can deploy it in one region and service multiple Availability Zones and multiple instances and containers in different Availability Zones, but it's a Layer 7 only load balancer so that's important to note.

2.) It's content-based routing, and that's important, I'm going to show you different examples later as far as well what does content-based routing mean and how that helps you, especially **when you want to point to different applications behind that load balancer and how that would compare to how you would do it traditionally, whereas traditionally you would require maybe more than one load balancer to accomplish this, now you can have Application Load Balancer that is pointing to multiple back end application services, again with a single load balancer.**

**3.)** It supports micro services and containers, so that's another important improvement here is that it's not just EC2 based, you can support microservices.

4.) It integrates with the Elastic Container Service from AWS.

5.) It will offer you better performance for real-time streaming.So if you're putting a service behind it that has real-time streaming you'll have better performance with the Application Load Balancer.

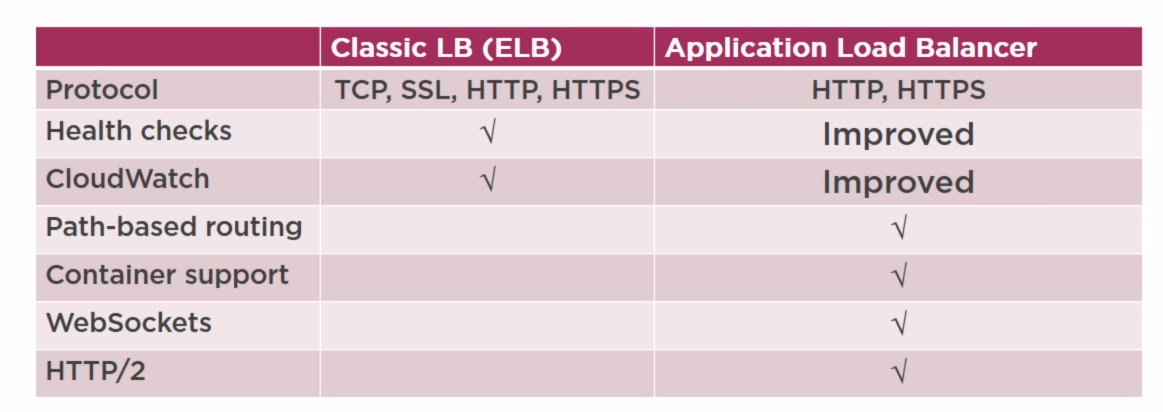
6.)You're going to have a reduced hourly cost as well with the Application Load Balancer, and there's a couple different methods by which you're going to be able to save money, **again scenarios that previously required more than one load balancer can now be consolidated on a single load balancer and as a result you are lowering the cost of using that load balancer.**

Now keep in mind, as I'm going through these different scenarios and as you're starting to think about your existing implementation and how you can optimize it, just because **you can consolidate on a single Application Load Balancer, may not be enough of a reason for you to do that. Or you might want to consolidate by department, maybe consolidate by service, because again you want to avoid that single point of failure exercise, so while you're saving money by consolidating on a single Application Load Balancer, you have to keep in mind how much you're consolidating on this Application Load Balancer.**

So if you currently have, let's say 20 ELBs, you might not want to bring them all back to one ALB, maybe you'll have two or more ALBs again that are divided either by service or by department.

7.)You'll also have deletion protection, deletion protection is if you have, if you've enable deletion protection on a particular instance then that instance cannot be deleted until you actually go through and remove that deletion protection, so that also helps in these cases.

8.)You'll have better health checks and integration with CloudWatch metrics bettern than Classic LB.



1.)probably as ELB, the ELB is now the family name for both, but keep in mind if you're still calling it ELB it's now the Classic Load Balancer, that is a Layers 4 and a Layer 7 load balance, but it's connection based, right? So again, this is why you see TCP, you see SSL, it'll support HTTP and HTTPS as well, the Application Load Balancer is strictly an HTTP, HTTPS, so it's strictly a Layer 7 load balancer.

2.)They both support health checks, although the Application Load Balancer has some improvements that 3.we're going to talk about, from a CloudWatch perspective, integration perspective they both support it.

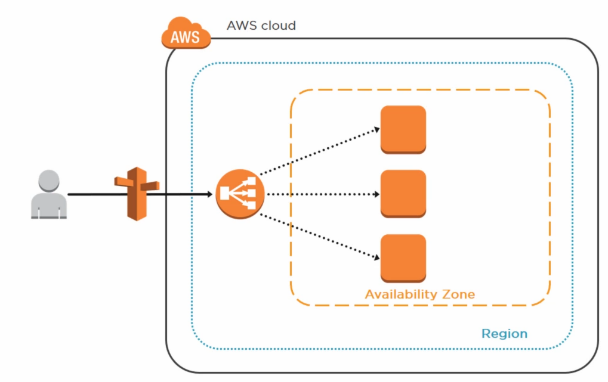
again the Application Load Balancer has some improvements there as well.

3.)Now things that the Application Load Balancer has that the Classic Load Balancer doesn't have is path based routing. Now this is cool because this allows you to now, what would take you two load balancers to accomplish with the Classic Load Balancer, you can now accomplish with one load balancer, and I'm going to show you a diagram that will properly illustrate this, that will do it more justice, but again I just wanted to summarize some of the capabilities and some of the features.

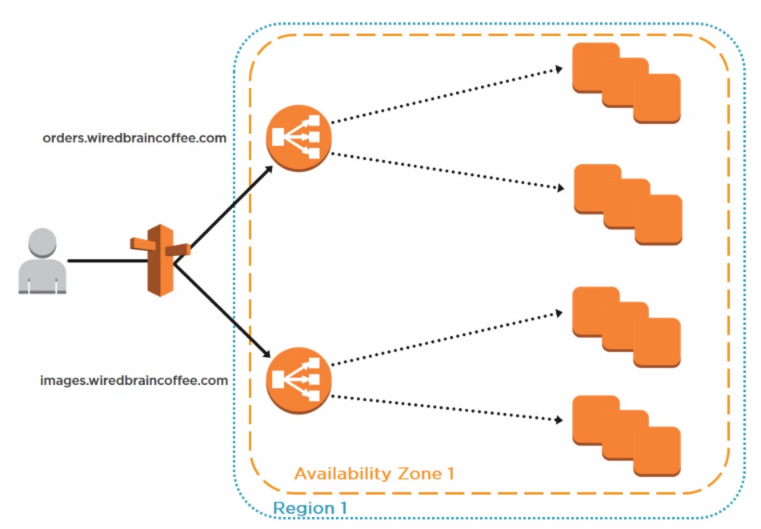
4.) The Classic Load Balancer didn't support containers, with the Application Load Balancer you can support them.

5.)WebSockets weren't supported with Classic Load Balancer, you can support it with Application Load Balancer.

6.) HTTP/2 is also supported with the ALB and that was not supported with the Classic Load Balancer.

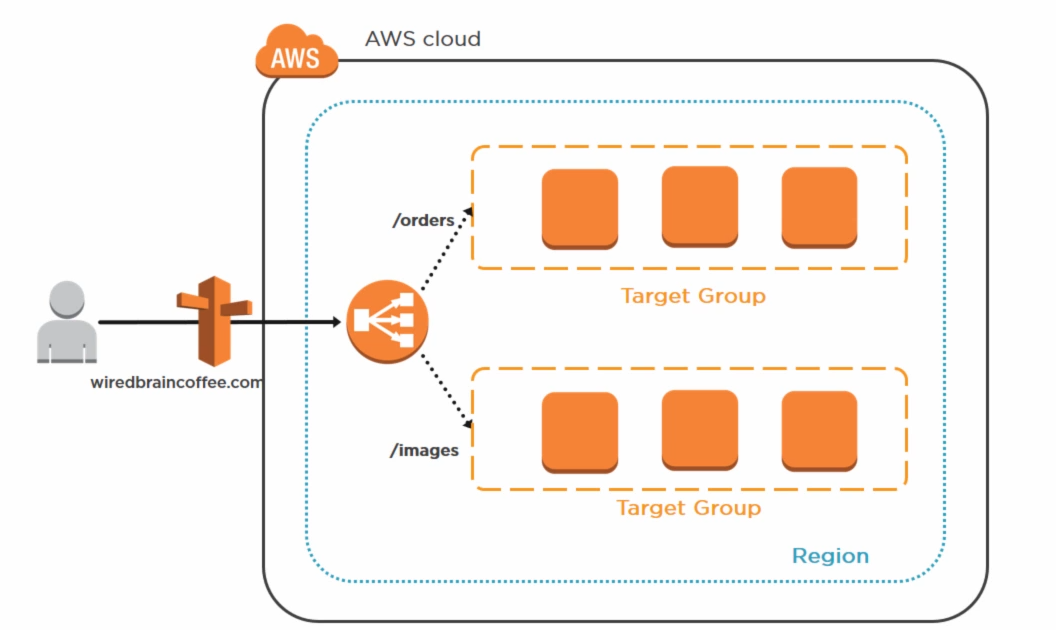
1. let's talk a little bit about path-based routing, and in order to illustrate why this is important, how it's improved, let's take the example and start off with the Classic Load Balancer. Now the Classic Load Balancer was a very simple load balancer that forwarded traffic to a bunch of instances, essentially you could only put one workload, one application in front of that load balancer, if you needed more than one application you would create another load balancer. So this illustration is the very basic, very simple Classic Load Balancer graphic here.

2.) Let me show you if you had more than one application behind the Classic Load Balancer, what it would look like. It would look something like this.



So if you had, for example, yourwebsite.com/orders, or orders.yourwebsite.com and then images.yourwebsite.com, those would have to go to different load balancers and those different load balancers then point it to different EC2 instances, in this case you would need two ELBs, or the Classic Load Balancer, right? Two ELBs meant higher cost. Why? Well, you're consuming two load balancers.

And you can replicate this across multiple applications that you might have in your environment, so it was very difficult to consolidate them onto one load balancer, you would have to spread them across multiple load balancers. So how does the Application Load Balancer solve this?

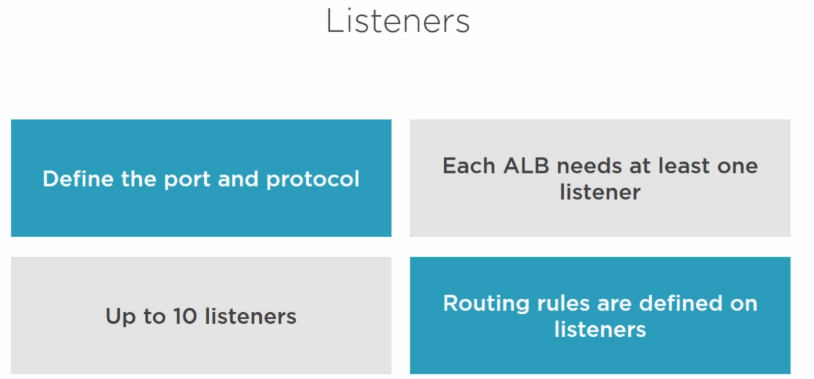
1. 

The Application Load Balancer solved this using path-based routing, so instead of having for example orders.mycompany.com that's going to one Classic Load Balancer and images.mycompany.com that's going to another load balancer, here what you have is mycompany.com/orders, and that is pointing to the same Application Load Balancer, and mycompany.com/images, again point to the same load balancer, the Application Load Balancer is intelligent enough to understand that the /orders I'm going to point it to this back end, to this target group, to this application, the EC2 instances or containers that service that application, and for images I'm going to point it to a different target group that has a different set of EC2 instances or containers.

So this is the advantage because now you can consolidate onto a singleApplication Load Balancer, so you're lowering your cost by again consolidating. Now we talked about the fact that you want to be careful about how you consolidate because sometimes you're saving money and it's great, right, but you want to also make sure that you keep in mind that single point of failure, so make sure as you're consolidating you're consolidating things that are in the same department or in the same service, again to avoid that single point of failure instance.

**Application LB Architectural Diagram**

you have the ALB, and then from the ALB you've got your listeners.



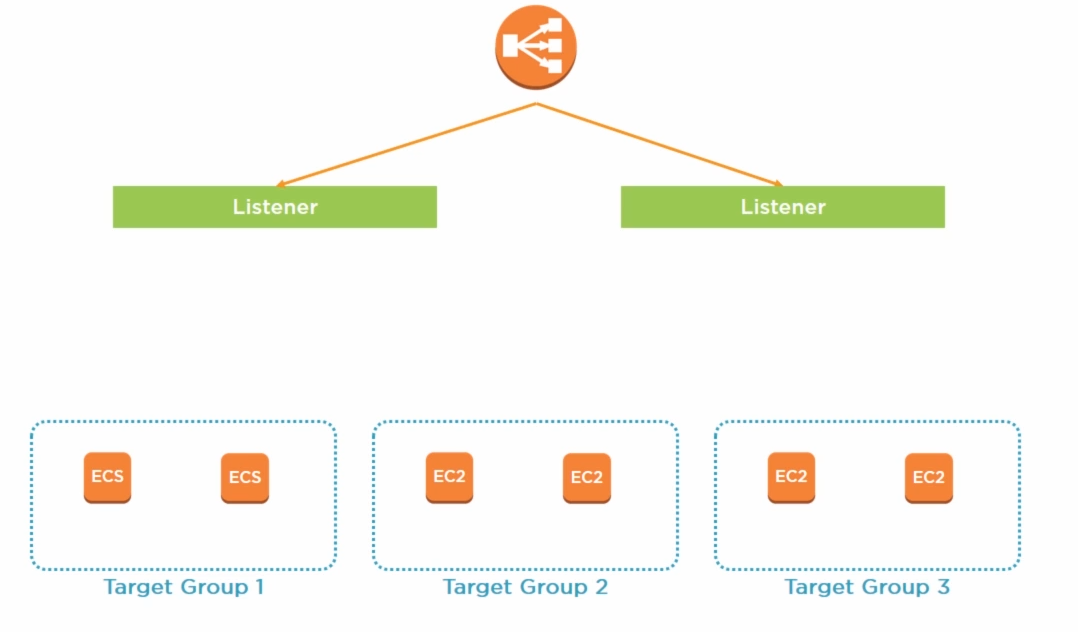
* Now as far as **listeners** are concerned, you can define the port and the protocol that the load balancer must listen on.
* Each Application Load Balancer needs at least one listener in order to accept traffic of course.
* Now each Application Load Balancer can have up to 10 listeners.If you communicate with AWS they might be able to raise this, but you can have up to 10 listeners.
* keep in mind that routine rules are also defined on listeners so that they know kind of where to route the traffic, so the listener is going to accept the traffic, and then based on the routing rules it's going to distribute them.

Next let's talk about **target groups..**

* Now target groups are logical groupings of targets behind the load balancer.
* Now today these target groups can only accept EC2 instances or containers, but again I'm pretty sure AWS is calling them targets just in case in the future they want to add something that is not an EC2 instance or is not a container that that would accommodate for it, but today target groups can only accept again EC2 instances and containers.
* Target groups can exist independently from the load balancer, so you can create your target group, you don't necessarily have to link it to the load balancer, but it's used for that and again that will probably have other purposes in the future.
* It's important to understand that target groups are region based, so they're at their regional level from an architecture perspective, but you can also associate them with an auto scaling group.
* So if you want the auto scaling group to kind of manage that, you can associate that with an auto scaling group.
* Target groups can contain up to a thousand targets, now you cannot mix and match, so your target group can either be EC2 instances or containers, you can't have both in the same target group, so that's another thing to keep in mind.

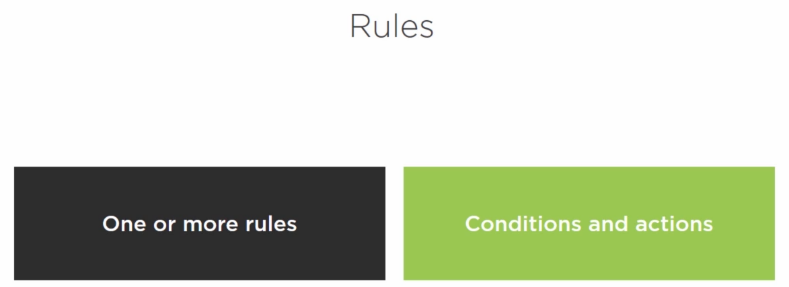
Now within target groups we talked about the fact that you can have targets. Now these targets are EC2 instances or they're containers.

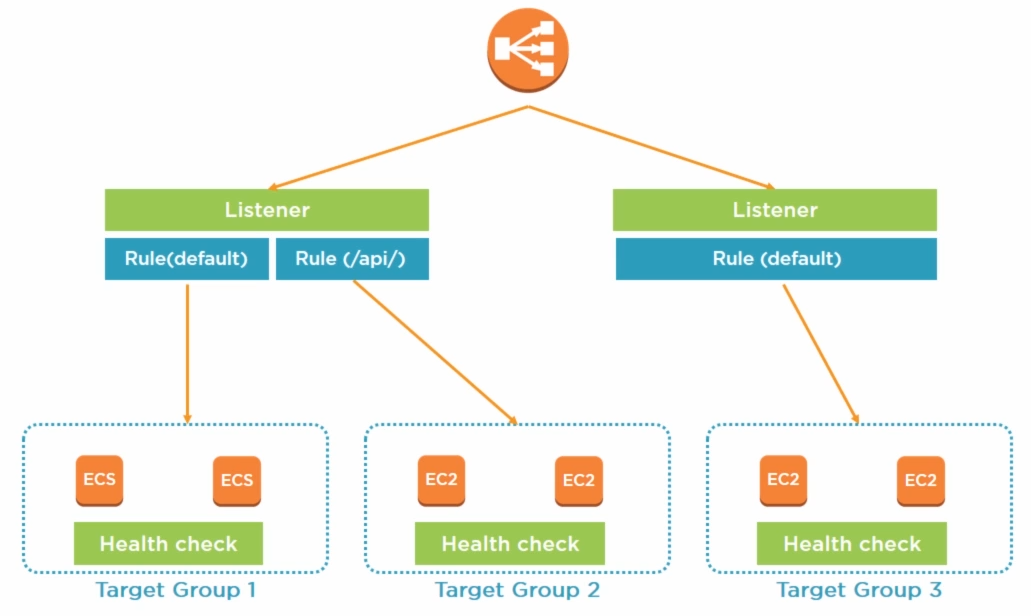
Now EC2 instances can be registered with the same target group using multiple ports. That's another advantage here is that you can configure that with the Application Load Balancer so that it's using different ports. This is especially helpful as you get into containers, and we'll talk about that.

A single target can also be registered with multiple target groups, so that target can be in multiple target groups.

**So, let's continue expanding this diagram and let's talk a little bit about rules.**

1. Now each listener can have one or more rules for routing requests to a target group.

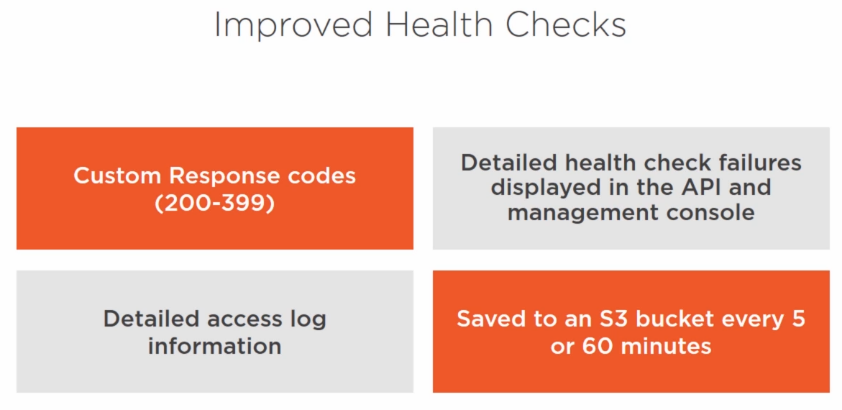


2.) Rules consists of conditions and actions. When a request meets the condition of the rule, the action is taken. It's pretty self-explanatory. Today rules can forward requests to a specified target group. Now the other important thing to note is if a request comes in that doesn't meet any of the conditions, that doesn't meet anything that is available in the rules, then you're using, it uses the default rule and sends it to that target group that's associated with that default rule.So those are things to keep in mind about rules. 

**Now as you can see here within the target groups we've got containers,** and I have them here mentioned as ECS. Now there's a good ECS integration with this Application Load Balancer. It's fully integrated with ESC, so you're able to, you're managing those target groups, those paths of the targets etc., ECS automatically registers tasks with the load balancer using dynamic port mapping, so again when you're using containers, if the application requires port 80, all of these containers on the same instance or on the same target cannot use port 80, so the importance of the Application Load Balancer here is that you can automatically configure it so that it's dynamically doing the port mapping for these containers, so that's very cool there.

Now the other thing to note is that while there's ECS integration with the Application Load Balancer, the Application Load Balancer can also integrate with third party technologies.

**Alright now let's talk a little bit about the improvements and health checks.**

****

1.)So while health checks were always part of the Classic Load Balancer, you had some limitations. So for example with the classic health check you only got a response back for example on response codes of 200, right, so you couldn't customize the list of successful response codes that you can use. With the Application Load Balancer you can now have anywhere from 200 to 399, so you have a widervariety of response codes that you can get, like a 301, etc., so you can define the success criteria a lot easier and you have a lot more, you have a grip on it a lot better than with the Classic Load Balancer.

2.)You also have more details of the health check failures, so in the past you would just know that okay there was a failure, but there wasn't enough information given. Now within the Management Console, they are going to provide you with details as far as well what happened, was it a timeout, was it I don't know, something that the specific outage that happened with that particular instance, that will be displayed as a message in the Management Console.

3.)And since we're talking about health checks, I also want to talk a little bit real quick about access logs. So **you also have the ability to get access logs, it'll provide you detailed information on each request processed by the load balancer, it'll include request time, client IP address, latencies, request path, and server responses, it's delivered to an S3 bucket every 5 or 60 minutes**.

4.) The logs are also indexed by date but include the IP address of the local load balancer node itself.

**Now from a cost perspective,** we talked about the fact that you can consolidate and do all of that, but I just also want to note that from **an hourly change perspective you can expect the Application Load Balancer to be about 10% cheaper than the Classic Load Balancer.**

****

Again, whenever it comes to pricing I have always said this and I'm always going to say this, make sure you're double-checking this against the latest and greatest AWS documentation just to make sure that this is still accurate, but as of this recording the hourly rate for the Application Load Balancer is 10% cheaper than the Classic Load Balancer.

Now from a configuration perspective we're not going to go through and configure an Application Load Balancer, it's outside of the scope of this course, but it's in the same location as where **you configure the Classic Load Balancer, you just select the Application Load Balancer and from a configuration perspective it's very, very similar to the configuration of the Classic Load Balancer.**

**Summary**

characteristics of the Classic Load Balancer, things like it's a region-wide load balancer, you can use it internally, externally, it's a Layer 4 and a Layer 7 load balancer so it's got some Layer 7 capabilities like session stickiness, cookie based sessions, etc., so it has some Layer 7 capabilities, SSL termination, SSL processing.

**prior to the Application Load Balancer, the Classic Load Balancer was known as Elastic Load Balancer**