

Welcome back and in the last lesson, I stepped you through the simple routing policy fail over enhances this architecture that you saw in the last lesson **by adding the ability to fail over from primary record types to secondary record types** and to demonstrate that in this lesson, I'm going to go ahead and create an S3 bucket. That's what I'll do first, so I'll move to the S3 management console. I'll create an S3 bucket and I'll be calling this bucket www.associatecats.com. I want to keep the name the same as this record and hosted zone, so www.associatecats.com and I'll click next. I'll be accepting the defaults on this page. I'll hit next again. I will be using this as a static website to provide backup functionality for this primary record set. So I will want to uncheck the block all public access. So I'll do that and click on next. I'll create the bucket. Once the bucket is created, I'll open the bucket up and I'll upload some objects and then I'll select the index.html file and the oops.jpeg file that are inside the S3 folder, inside this lessons files and I'll make sure that this is linked in the lesson description. So it's like both of those click on open, click on next, next again, next again, and then upload. Now, because I'll be creating a static website inside. This S3 bucket and don't worry, I know we've not covered that yet. In the course, we will do that in detail in the next section but for now, I'm going to go ahead and select both of these objects, click on actions and then make public that I'll make sure that both these objects are accessible from any anonymous users on the public internet and then to enable this bucket to be accessed from the public internet using HTTP, I'll go to properties, click on static website hosting. Use this bucket to host a website, and I'll enter index.html for both the index document and the error document and then once I've done that, I'll copy this endpoint URL into my clipboard and click on save. So at this point I've configured all that I need to from the S3 point of view, I got this backup S3 bucket that can be accessed over HTTP by my customers. So I'm going to move back across to the Route 53 management console and I'm going to configure this fail over routing policy type. Now, before I do that, I need some method of determining the health of this primary resource. So I'm going to create a health check. So I'll go to health checks, I'll create a health check and I'm going to name this health check www.associatecats.com just so it matches the fully qualified domain name that it will be checking. It's going to be an endpoint check. I'm going to use an IP address to specify the endpoint. It's going to be a HTTP check, and I'll need the IP address of the instance that I'll be doing the health check on. Now they're to keep things simple, I'm going to use a single EC2 instance is the primary resource. So I'll need the IP address of web1. So I'll copy that into my clipboard. Go back to the Route 53 management console paste that in. Now I'll be checking this resource on port 80 and I won't be using a path. It'll just be checking the web root. We'll leave all of that standard and I'll dropdown the advanced options and just for this demonstration to make it easier, instead of performing the health check every 30 seconds, I'm going to make a fast health check. So the check will be performed every 10 seconds and I'll change the failure threshold from 3 to 1. Now what this means is instead of having to wait a period of 30 seconds and wait for three of those intervals in order to change from healthy to unhealthy or vice versa in this case, it's going to be checking every 10 seconds, and it will only need one period. So this will make it quicker to change status, and so it would be more effective for this demonstration. Now, we've got everything that we need here configured, so I'll go ahead and click on next. Once I've done that, I won't want to create an alarm. So I'll just click on create health check. The health check status will start in an unknown state. So before any of the globally distributed health checkers perform their checks it'll be in an unknown status. If I refresh this a couple of times, though, it will change from unknown to unhealthy. The reason it's going to do this is that when I created the EC2 instances are created them with a security group the only allowed HTTP connections from my public IP address. So these health checkers won't actually be able to connect to this EC2 instance, so after a number of refreshes, it's now in an unhealthy state, and if I select the health check, go to the health checkers tab, you've got to see that the reason for that failure is the connection timed out. To fix that problem, I'll go to the EC2 console, I'll select the web1 instance, select the security group that's associated with this instance, which is the web server security group which I created this in the previous lesson. I'll open that group, go to the inbound rules, click on edit, and instead of being my personal IP address, I'm going to change this to anywhere. So this will be anywhere on the public internet. So I'll click on Save that will update, and that will mean that the health checkers for Route 53 will now be able to communicate with this EC2 instance. I'll need to refresh this a couple of times but as soon as I do, you'll note that now we're getting HTTP status code two hundred, which indicate that this resource is now healthy. Now, if you recall earlier in this lesson, when I talked about the health check itself, I talked about the failure thresholds. So because the health checks are running every 10 seconds, it needs to wait for one set of successful health checks to occur before it'll change the status of this health check so it won't change immediately it's still showing as unhealthy. If I select it, go back to the health checkers. We are getting status code two hundreds. So now if a hit refresh a couple of times once we've got one full set of health checks, that comeback as healthy the next set that occur will change the status from unhealthy to healthy. Now it did take a short while for this to occur, but it is now in a healthy status. So based on the architecture diagram on the right of my screen, we're now performing a health check on these primary resources. Now to keep things simple, I have configured this particular records, so if I go to hosted zones and then associatecats.com. I've configured this single www record to only point at a single IP address. This is the IP address of the web1 EC2 instance, this is the instance that health check is performing checks on to determine whether it's healthy or unhealthy. Now to implement its architecture what I need to do is change the routing policy from simple to fail over. Now, this presents additional functionality. A fail over routing policy means that I can define additional records with the same name. Remember, in the previous lesson, I wasn't able to create additional records with the same name and for a given record, I was only able to define one value. That value could have multiple IP addresses, but it was always returned as a single object. Now I can select fail over routing, and I'm able to define multiple records with the same name. Now with fail over routing, you also need to specify two different types of records primary and secondary. The purpose of the primary record is that it will be what Route 53 responds with unless it's unhealthy and if it's unhealthy, the secondary record type will be used. So the first thing I'll do is create this primary record. The set ID needs to be unique amongst all records sets with the same name. So in this case, this is going to be www-primary because this is a primary record, I'm going to associate it with a health check and I'm going to use the www.associatecats.com record set that created in the previous step. Once I've done that, everything else looks good. So I'm going to go ahead and save this record set. So this is the primary record set that I've got shown on this architecture. This means that as long as this is healthy, it will be what Route 53 responds with. So next time I'm going to create the secondary fail over record types, I'm going to go to create records that I'm going to call this the same name so www.associatecats.com I'm going to give it the same TTL of one minute I'm going to specify the fail over routing policy type but this time I'm going to pick secondary. Now what I want to do is point this at the S3 bucket and this S3 bucket to be the resource that's returned whenever our primary website is having issues. Now to point this directly at S3, I'm going to change this to be an alias and for the target name, I'm going to pick the www.associatecats.com is S3 bucket that I created earlier in this lesson. Once I've done that, I'll click create Now note how this allows us to do this. Remember in the previous lesson, I wasn't able to create another record set with the same name, but now I'm able to I could do that because these are both fail over routing policies and using fail over routing we're able to create multiple record sets of the same name. So what does this mean from a functionality perspective? Well, if I browse to www.associatecats.com because it's a fail over routing policy, the primary fail over record type will always be returned. So if I go to my terminal and I do an ns look up on associatecats.com I'll get the www.associatecats.com record set and I'll be returned the value of the IP address. So this is the IP address that's associated with web1. So if I open that in my web browser, I'll browse to this name in a new tab, I'll get public cat web1. So this is the website that's running on the EC2 instance the web1 EC2 instance. So let's now test the fail over. If I go back to security groups and this is the security group that's associated with all three of those EC2 instances, so I'll edit the inbound rules and instead of having HTTP allow from anywhere, I'll delete both of these rules to simulate a server failure. So if go ahead and click on Save that means that the Route 53 health checkers will no longer be able to connect to these EC2 instances to monitor the health. So if I go back to Route 53 go to health checks will see very quickly. The status of this health check will change from healthy to unhealthy. So if I select the health check and go to health checkers and hit refresh, we can already see that these connection time outs are starting to appear. So what will happen is the status of this health check will change from healthy to unhealthy. I'll just refresh a couple of times to illustrate this and after a couple of refreshes, it's changed to unhealthy. What this means is that the primary record set types— so if I go back to hosted zones and associatecats.com and select this record. This is the primary fail over record type. This now reports as unhealthy because it's linked with this health check, which is itself unhealthy. This entire record now reports as unhealthy. What this means is that it will no longer be returned by Route 53 because we've got a secondary or a backup fail over record type this will be returned instead of this primary. Now if I go back to my terminal and do another ns look up instead of this 18.212.55.166 IP address. This time I get this IP address, which isn't one of my EC2 instances. What's happening is because this is an alias record type. It resolves to an IP address that's part of the S3 service. Now if go back to the tab that I previously have this open in and hit refresh instead of the functional primary website this time "oh noes something is brokes" so this is using S3. Now this is a simple example of a primary secondary fail over architecture. Now imagine that this is a real production usage. Of course, this isn't production because I've included some spelling errors and it's using an under construction cat but if it was production, you might use this to provide a maintenance page for your functional application. Maybe your web application is running on EC2 instances, or it's a serverless application using Lambda and API gateway but you want to cope with any situations where the primary website fails, or you want to take it down for maintenance. Well, with fail over routing you can do that. You can specify this secondary record configure a health check on the primary and whenever the primary is out of action, it will redirect to the secondary record type, which in this case is a website that's running inside an S3 bucket It's got a funny message and an amazing cat picture, but if it was in production, you could present a maintenance page, maybe some contact details, how your customers could contact your support teams, or any other information you want available when your primary website is down. So this has been a pretty simple example, but it's illustrated the power of fail over routing.

**Now with fail over routing policies, you are only able to create a single record for both the primary and secondary. So if I was to create another record set.** Let say I hit create record set again, I call it www. Let's say again, it's an alias record type that I'm pointing at this S3 bucket, going to change it to be fail over record type, and secondary and then hit create. I wouldn't be able to create this additional record. This is important to note for the exam. For fail over routing, you can only create a single primary fail over record type and a single secondary fail over record type. If I attempted even to change this set ID. Maybe to add a letter to make it unique an attempt to create it. I wouldn't be able to do that either, because it could only have a single primary and a single secondary. Now, I'll talk about this more as we proceed through the course, but it is possible, for each of these records acts to point them at multiple underlying record sets. So as we'll see later on in this topic of the course, we can have different routing policies. We can have geolocation, latency, and weighted and you can chain these together so I could have the primary fail over record type pointing at multiple underlying records and each of those could be different routing policies. So you can make this really complicated and so, as we proceed throughout this topic of the course and the later topics, you'll see how you can create some pretty complex routing trees inside Route 53 but for the exam, we want to keep it simple. Remember, a single primary fail over record type and a single secondary fail over record type. Now that is a wanted to cover in this lesson. It's a fairly simple concept to grasp but I don't want to give you a demonstration because I often find that demonstration lessons help you absorb the architecture. You've learned a lot of theory about Route 53 so far and in this topic I want to focus on demonstrating how we can implement that theory within practical scenarios. With that being said, go ahead and mark this video as complete and when you ready, join me in the next.