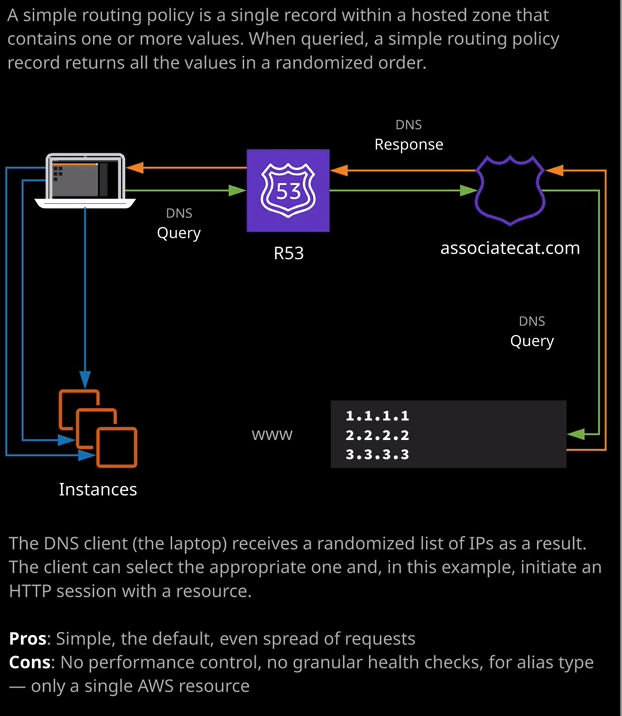
In this topic, I want to step through **the different routing policies that Route 53 offers and a routing policy simply defines how Route 53 handles any requests for any of the hosts inside any of the hosted zones.** So when you do a DNS resolution request for www.associatecats.com, the routing policy defines how Route 53 handles that request. Now routing policies are divided into two main types.

We've got simple and multi value and the first routing policy that I want to talk about is the simple routing policy.



A simple routing policy is one where you define a single record so inside the Route 53 console, I've defined a single host inside this hosted zone. It's test.associatecats.com. It's an A record type, and it resolves to 1.1.1.1. So if I move across to my terminal and I attempt to do a NS look up on that holds name, so ns look up test.associatecats.com, it'll return this single value. So address 1.1.1.1. Now to illustrate exactly how this works, I'm going to go ahead and create three EC2 instances. I'm going to create them inside the default VPC and I'm going to have them launch automatically using user data. Now they're going to be three servers that I'm going to create web1, web2, and web3 and the user data script for each of these is available inside this lesson's resources on the course GitHub repository. So I'm going to start I'm going to go ahead and launch an instance I'm going to skip through this relatively quickly. I'm only going to show the creation of the first one I'm going to follow the same process for the following two. So I'm going to launch using Amazon Linux two. I'm going to pick a T3.micro instance, which is the one that I've been using throughout this course. I'll go to next configure instance details. Make sure that it's in the default VPC and that it does have a public IP address assigned. I'll drop down advanced details and I'll paste in the user data for the web1 server and again that's available inside this lesson's files. So the files you're looking for are inside the lesson's file folder and then 04\_networking. Then we're looking for topic five advanced DNS and then we're inside the simple folder and then I'll be using, in this case, user data web1.txt. That's inside the web1 folder. So I'll take the contents of that file and I'll paste it into the user data box under advanced details. All this is going to do is build a simple web server and download and install some assets into the web folder, so I'll go to add storage. I'll accept the defaults. Go to security group, create a new security group, which I'll call web server. I'll set the description to be the same click Add Rule, select HTTP, and I'll set both of these to my IP just to keep things a little bit more secure. Once I've done that, I'll go to review and launch, click on launch. In this particular case, I'll select to proceed without a key pair because I won't be connecting to the instances. I'll acknowledge that and click a launch instance. Now I didn't provide a tag there. So I'm going to click on the pencil and call this web1. Now, at this point, while this is launching, I'm going to follow the same process for the additional two web servers. So if I go back to the lesson files. I'll be using user data web2 and user data web3. So I'm going to skip ahead to when both of those servers have been created. Now that I've got these three web servers provisioning, I'm going to select web server one and then copy the public IP address into my clipboard because I want to verify that each of these servers is working as expected, so I'll open each of these servers in a new tab. So we've got public catweb1. That's web server one. I'll do the same protest for web2 so I'll open that in a new tab. It's the same picture, but running on web server two and then the same process for web three. That's three different servers web server one, web server two, web server three. Now that I've done that, I'm going to configure a simple routing policy so I want to create a www record for associatecats.com that can resolve to any of these three web servers. So to do that, I'm going to copy the IP address into my clipboard and go to the Route 53 management console. Then inside the associatecats.com hosted zone I'm going to create a record set. I'm going to call it www and then paste in the IP address. I'm going to select to have a TTL of one minute. **So TTL or time to live is just the way in which you specify to DNS how long you want this record to be cached for.** So if you select one minute and any other DNS service is cache this record, then they should do so for only one minute. So I'm going to do this to keep things simple, and I'll go ahead and click on create. Now, at this point, if I was to do an **ns look up** on this record, I'll get this IP address returned. So let me show you that will move across to my terminal and I'll do an ns look up on www.associatecats.com so I get that single IP address returned, and that is the most basic form of DNS record that you can put in Route 53 but what I can do is expand the functionality of this by adding additional value. So right now, we've only got this single value if I go to a new line, go back to the EC2 console, select web server two, copy that IP address into my clipboard. I'll be able to add that to this list of values, and I'll do the same for web server three. So I'll go back to my instances, select web server three, copy that into my clipboard, and paste that in as the third and final values. Once I've done that, we'll go ahead and hit Save. If I go back to my terminal on run another ns look up. This time I'll receive all three of these values that are stored inside this record and this, in essence, is how the simple routing policy type works. **You've got a single record multiple values for that single record, and every time a query occurs to Route 53 those values are returned in a random order. Now when you'd use this type is if you're not aware of how your traffic patterns are and you want a simple starting point for your DNS architecture.** So in this particular case, I might have these three individual servers providing the same web page and a single record pointing at each of these three servers. In this particular scenario, **the pros are is that you will have a somewhat even spread of requests. So over the entire global set of your clients when they're performing DNS requests against this particular record in this hosted zone, they will, most of the time retrieve a different IP address as part of that resolution request**. So if I go back to my terminal on do a look up this time I get the 18.212.55.166 IP address returned as the first entry. Now, if I do that the same ns look up again because it's cached I'll get that same address. When the TTL value expires, I'll get another address. If you were to perform a look up on this host, you would get a different address again and so simple routing generally provides a simple form of spreading the requests against all of the different values but note this is **not load balancing**. **It doesn't ensure that the load is distributed evenly across each of your individual web servers, because every single DNS request that occurs could be cached and if it's cached by a large organization, then you could have all of the users of that organization hitting that single IP address and that's why this TTL value is so important, because if it's set to fairly low, it means that each 60 seconds, even for large organizations they're going to retrieve this list in a different order. So over time you're going to get some form of leveling across each of the individual servers.**

Now the cons is that **you don't get any kind of performance control you're not able to influence which of these IP addresses are returned, and it's not a load balancing architecture. You don't have the ability to do any granular health checks against these individual values.** So if one of them fails, let's say this first one you could still have requests that are directed at this IP address. Now, **if you do decide to pick alias type record and do be aware that you can only select a single AWS resource**. So as you'll see if you proceed throughout this topic of the course, they're different routing policies, and some of them offer much more functionality.

So when it comes to a simple routing policy, **you can only point it at a single AWS target. That might be a single S3 bucket, a single load balancer, a single network load balancer, a single CloudFront distribution, and so on**. Now, don't worry. We'll be covering a lot of these later in the cause but when it comes to the exam, **a simple routing policy can only point at a single AWS resource if you do pick it to be an alias, if it's not an alias and you are able to pick multiple values, but you have no control over how those values are returned. Now, with simple routing policies, you can only have a single record with the same name.** So if I went to create a new record set I wouldn't be able to create a new one with the name www. So if I create that, I'll get an error. That's not the case with the other routing policy types, and I'll illustrate that over the next few lessons. So remember, **for simple routing policy type, it points at one or more values, and they're returned in a random order. You're able to create it as an alias record type, but if you do that, you can only point it at a single AWS resource with simple routing policy types, you don't get to perform any granular health checking** and I'll talk more about how that's different for the other routing policy types over the next couple of lessons but for now, that's all I wanted to cover for simple routing policy. You'll see how all of the other types gives additional functionality over the rest of the lessons inside this topic. For now, I hope it was useful. Go ahead, mark this lesson as complete, and when you're ready, you can join me in the next.