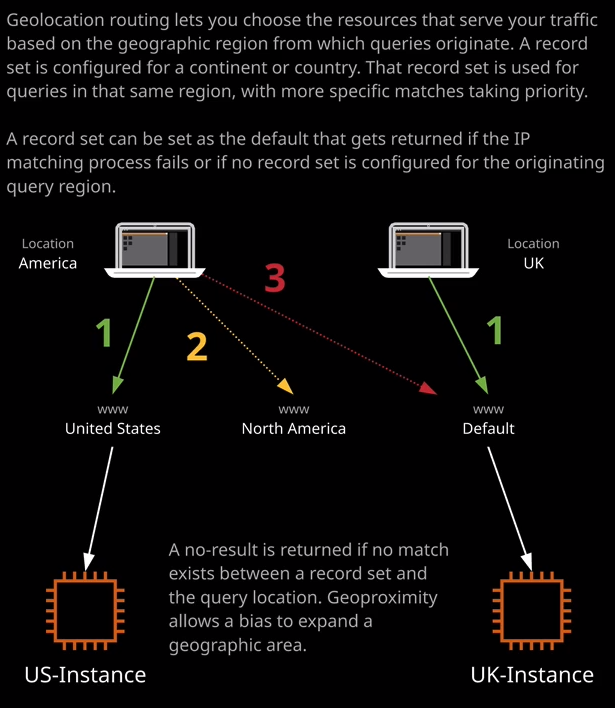
geolocation routing.

Now this is just another routing policy that's available inside Route 53. In the last lesson, I discussed **latency routing and that's where you can allocate a particular region to a given record inside Route 53 and then when any customers attempt to perform resolutions and that record it would select the record with lowest latency to that customer. So you're not really focusing on the geographic location, you're focusing on the network conditions between your customers and the various different resources that you provision inside AWS regions.** So the last lesson was about network performance. It was about latency.



In this lesson I want to talk about physical geography. Now configuring geolocation routing is pretty simple. To demonstrate it quickly and I'm just going to go ahead and delete both of these records that were configured to using latency based routing. I'm going to create a new record set. I'm going to call this record www I'm going to set the TTL to be one minute and then for the value I'm going to move across to the EC2 console and I'm going to select web1. I'm going to copy the IP address of the EC2 instance. Go back to the Route 53 console, paste in this value, change the routing policy down to geolocation, and then with geolocation records, you have to pick a location for this resource. Now the options that you have or either default, continents, or specific countries. So for this example, I'm going to pick North America for the continent. So this version of the website is running on an EC2 instance that is actually running inside U.S. East-1, which is North America. So this is actually correct. Now, as with previous multi answer value routing policy types, you have to specify a set ID, which is unique inside any records with the same name. So for this particular set ID, I'm going to call it web1 and click on Create. Now, there is a crucial difference between latency based routing and geolocation, and I'm going to demonstrate that now by moving across to my terminal and attempting to perform a look up on this fully qualified domain name, so www.associatecats.com. So I'll move to my terminal. I'll clear the screen to make it easier to see. I'll tab up to get the previous command, and I'll do a lock up. Now notice that we get this no answer. It can't find the record in associatecats.com, but I've actually configured this record. So why isn't it returning it? Well, that's one of the benefits of geolocation routing. The way that geolocation works is that when a given resolver attempts to perform a look up to this particular hosted zone, it returns only records, which both match the name of the requests so www.associatecats.com and are matched by a particular location. Now when you're specifying a location, you can do so with different levels of granularity. In this particular case, I specified North America and only request that are coming from North America will be returned this particular record. I could create another record set for example, www.associatecats.com specify the same one minute TTL in this case, let's say I get the web2 instance IP address. I'll copy that into my clipboard, paste that into the value, and then under routing policy change that to geolocation and then, for example, with the location I could scroll all the way down to countries, locate Australia, select that, set ID as web2, and then create the record set. Go back to my terminal and perform that same ns look up command again. This time I will get a response. I'll get the IP address of the web2 server because that's configured in Australia and I have a match set for that location. **Now, this is an important distinction to understand from latency based routing. With latency based routing, you were setting the location of a record and Route 53 was picking the record with lowest network latency. With geolocation routing, what you're actually doing is limiting the region of the world that this record can be returned to.** So because I'm based in Australia, when I perform a look up at www.associatecats.com, it will only return records that are either based in Australia so I can set my specific country as Australia. I could also set it as the region. So, for example, Oceana which represents Australia, in part or I could set default, which is used if no other records apply. So these are different levels of abstraction. If you imagine zooming out on a global map, we could start in Australia. We could zoom out little bit and return Oceana, zoom out a little bit more and return default. You can imagine for demonstrations sake, that default represents the entire planet. Now, if you don't have a particular record that matches at any of those different levels of zoom, then it will return a no answer. So with this architecture that illustrated on my diagram on the right and part of my screen imagine I've got two clients, one located in America and one located in the UK I've configured three different records. I've got one record that set the location to be United States. I've got one record that set the location to be North America and another record where the location is set to be default. Now for the American client, there will be three records that apply the United States, the North America, and the default and the order in which these are returned is based on the level of abstraction on the level of zoom so the most specific is always returned first. So the preference is that the most specific record is always returned first. In this case, it's the United States. If there isn't a record for United States or if it fails a health check then the second priority will returned, which will be the continents so North America. If we don't have a particular continent record configured then the default record will be returned, which is priority number three. If we have a client from the UK because we don't have a record that is set for the UK or for the European continent, the only one that will apply would be the default and as we saw earlier in this lesson, if we have any clients that are attempting to resolve where they don't have a country record, a continent record, and a default record doesn't exist, they'll get no answer.

**It's important to understand that geolocation based routing is used to present different content to customers in different regions. So if you run a major application like Netflix and you want to present a different application based on the country using the same DNS name so www.netflix.com and have UK customers using one platform, US customers using another, Australian customers using another, and then perhaps customers not from any of those specific countries directed at a default platform then you can do so using geolocation based routing.**

Now, before I finish up, there is one more that I do want to comment on, and that is **multivalue answer.** Now I've used this term elsewhere in this topic. I started off with a simple routing policy and if you recall what I said about that one is that you could only have a single record with same name and a single value that gets returned. Now that single value could be one or more IP addresses, but it's always returned in a single response. If you put multiple values, they will get returned to the client at the same time in a randomized order but it's only ever a single answer. We do have this final routing policy type, and this is known as multivalue answer. You can think of this in much the same way as the simple routing policy, but it allows you to specify multiple records with the same name and Route 53 will respond to queries with up to 8 healthy records selected at random. So with this routing policy type, you can create records of the same name, with a unique set ID, pointing an individual resource inside AWS, and then it will respond with eight of these selected at random**. Remember earlier in the topic, where talked about the limitation of the simple routing policy was that you couldn't associate individual health checks with these individual IP addresses. Well, now you could create multiple records for www.associatecats.com and in each of these records have individual IP addresses or have a alias to individual S3 buckets or other AWS resources and then Route 53 could do health checks on each of those individual resources and return up to eight healthy records at random from all of them that are in this list. Now with multivalue answer all you need to make sure is you've got a unique value for the set ID for each of these individual records, and then you can create a large number of these and Route 53 will return eight of them at random.**