**Route-53**

**Public Hosted Zones**

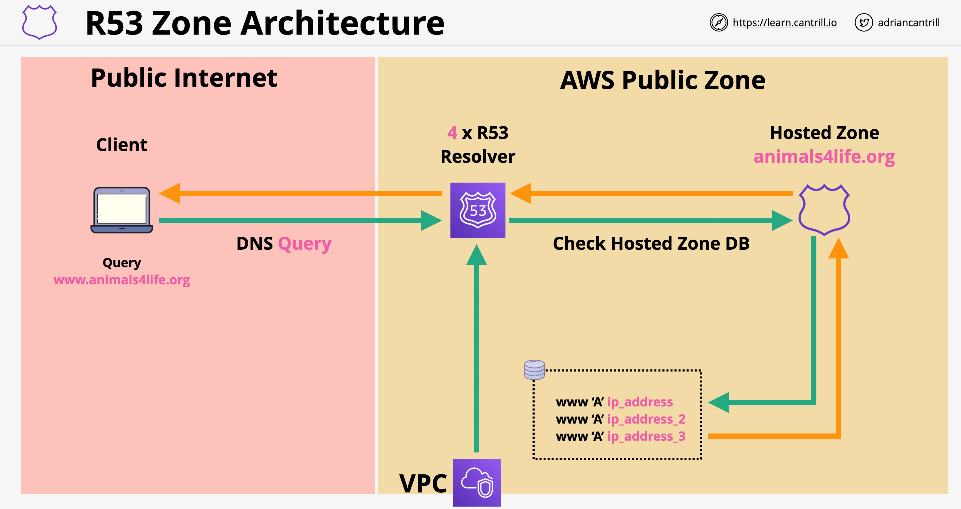
A **hosted zone** is a **DNS database** for a given section of **global DNS data**.

A **public hosted zone** is a type of **R53 hosted zone which is hosted on R53 provided public DNS name servers**.

When creating a hosted zone, AWS provides at least **4 DNS name servers** which host the zone. This is **globally resilient** service due to multiple DNS servers.

Hosted zones are **created automatically when you register a domain using R53**.

Hosted zones **can be created separately**. If you want to register a domain elsewhere and use R53 to host the zone file and records for that domain, then you can specifically **create a hosted zone and point at an externally registered domain at that zone**. There is a **monthly fee** to host each hosted zone within R53 and a fee for any queries made to that service.

**Hosted Zones** are what the DNS system references via delegation and name server records. A hosted zone, when referenced in this way by the DNS system, is known as being **authoritative for a domain**. It becomes the single source of truth for a domain.

**Route 53 Health Checks**

Route checks will **allow for periodic health checks on the servers**. If one of the servers has a bug, this will be **removed from the list**.

If the bug gets fixed, the health check will **pass** and the **server will be added back into a healthy state**.

Health checks are separate from, but are **used by records inside R53**. You don't create health checks inside records themselves.

These are performed by a **fleet of global health checkers**. If you think they are bots and block them, this could cause alarms.

Checks occur every 30 seconds by default. This can be **increased to 10 seconds for additional costs**. These checks are per health checker. Since there are many you will automatically get one every few seconds. The **10 second option will complete multiple checks per second.**

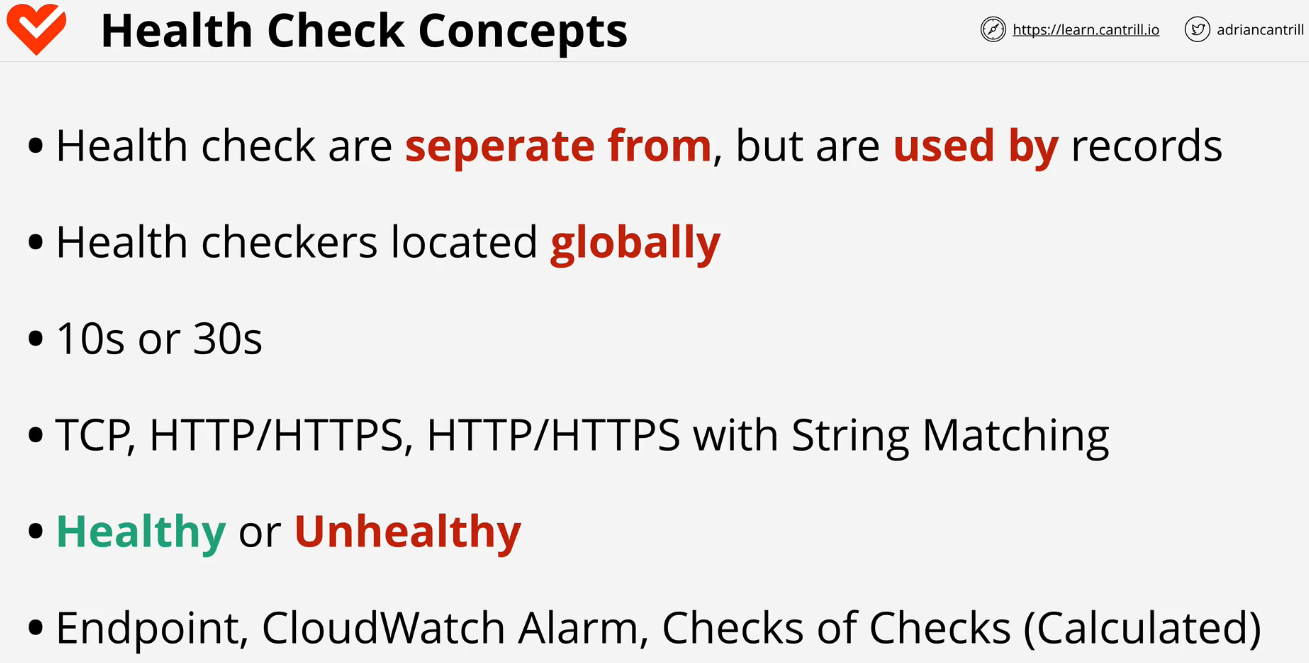
There could be one of three checks

* **TCP checks:** R53 tries to establish TCP with end point within **10** seconds.
* **HTTP/HTTPS:** Same as TCP but within **4** seconds. The end point must respond with a 200 or 300 status code within **3** seconds of checking.
* **String matching:** Same as above, the body must have a string within the first 5120 bytes. This is chosen by the user.

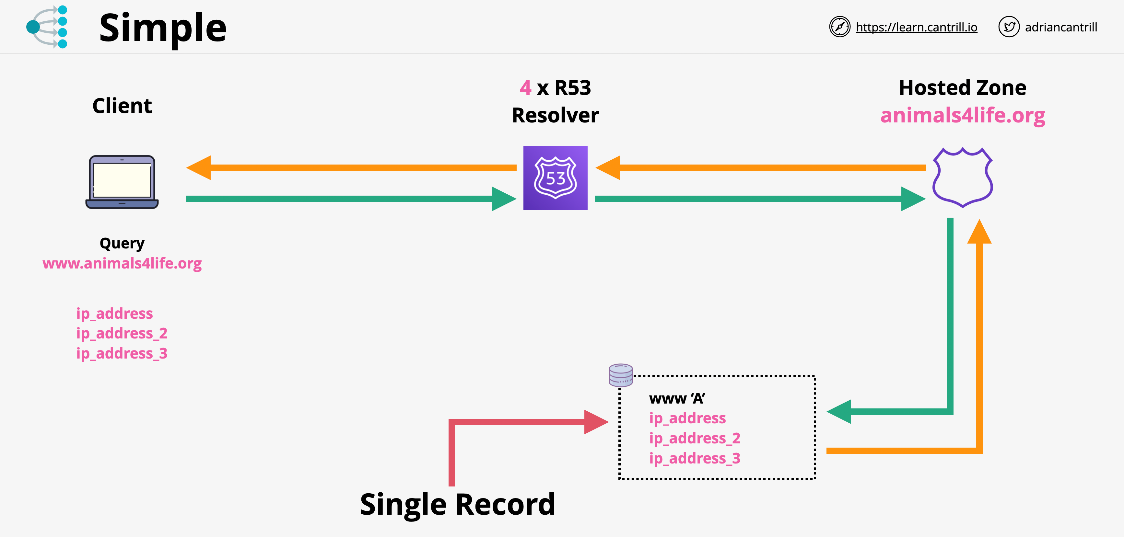
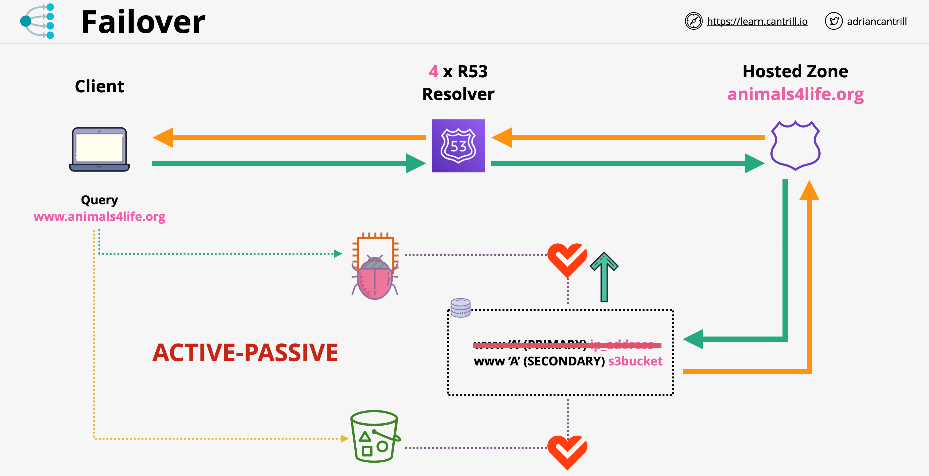
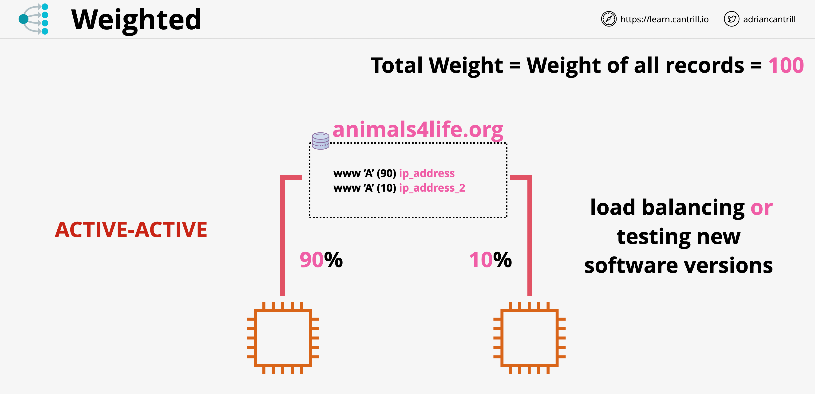
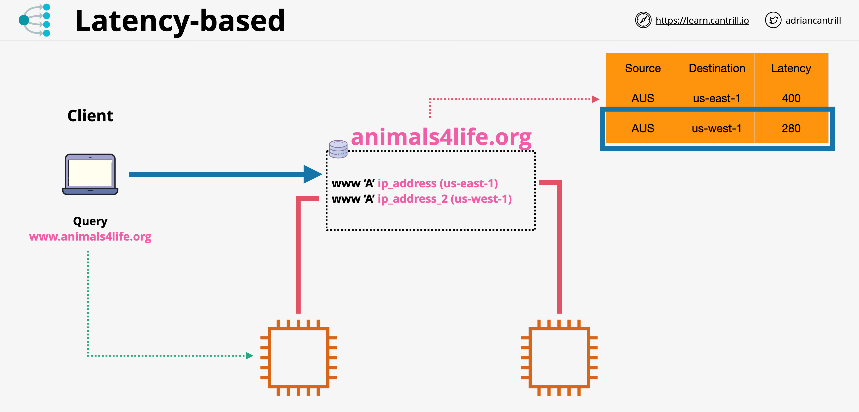
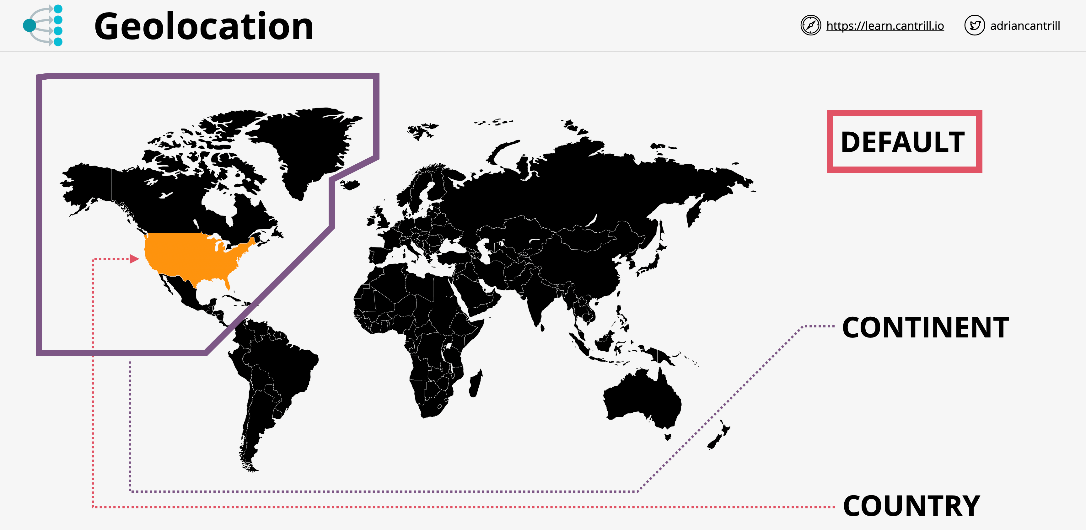
It will be deemed healthy or unhealthy.

There are three types of checks.

* **Endpoint checks:** Asses the health of an actual end point that an user specifies
* **CloudWatch alarms:** React to cloud watch alarms. Configured Separately. May require a Cloud Watch Agent for In App Test.
* **Checks of checks(Calculated):** Configure different checks for different components of the application. Then set up check to check this health checks and provide an aggregate overview.



**Route 53 Routing Policies Examples**

* **Simple:** Route traffic to a single resource. Client queries the resolver which has one record. It will respond with **3 values** and these get forwarded back to the client. **The client then picks one of the three at random**. This is a single record only. **No health checks**.
* **Failover:** Create **two records of the same name** and the **same type**. One is set to be the **primary** and the other is the **secondary**. This is the same as the simple policy except for the response. **Route 53 knows the health of both instances**. As long as the primary is healthy, it will respond with this one. If the health check with the primary fails, the backup will be returned instead. This is set to implement active - passive failover.
* **Weighted:** Create **multiple records of the same name within the hosted zone**. For each of those records, you **provide a weighted value**. The total weight is the same as the **weight of all the records of the same name**. If all of the parts of the same name are healthy, it will **distribute the load based on the weight**. If one of them fails its health check, it will be **skipped over and over again until a good one** gets hit. This can be used for **migration to separate servers**.
* **Latency-based: Multiple records in a hosted zone can be created with the same name and same type**. When a client request arrives, it knows which region the request comes from. It knows the **lowest latency and will respond with the lowest latency**.
* **Geolocation:** Focused to **delivering results matching the query of your customers**. The record will first be matched based on the country if possible. If this does not happen, the record will be checked based on the continent. Finally, if nothing matches again it will respond with the default response. This can be used for **licensing rights**. If overlapping regions occur, the **priority will always go to the most specific or smallest region**. The US will be chosen over the North America record.
* **Multi-value:** Simple records use **one name and multiple values** in this record. These will be health checked and the **unhealthy responses will automatically be removed**. With multi-value, you can have multiple records with the same name and each of these records can have a health check. **R53 using this method will respond to queries with any and all healthy records**, but it removes any records that are marked as unhealthy from those responses. **This removes the problem with simple routing where a single unhealthy record can make it through to your customers**. Great alternative to simple routing when you need to improve the reliability, and it's an alternative to failover when you have more than two records to respond with, but don't want the **complexity or the overhead of weighted routing**.