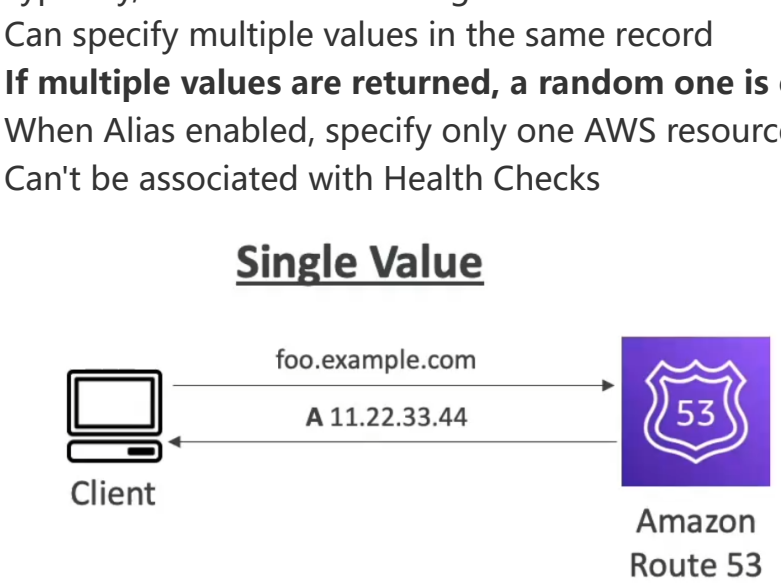
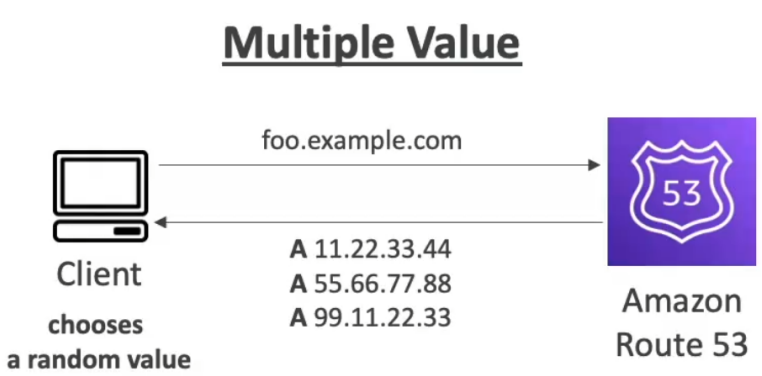
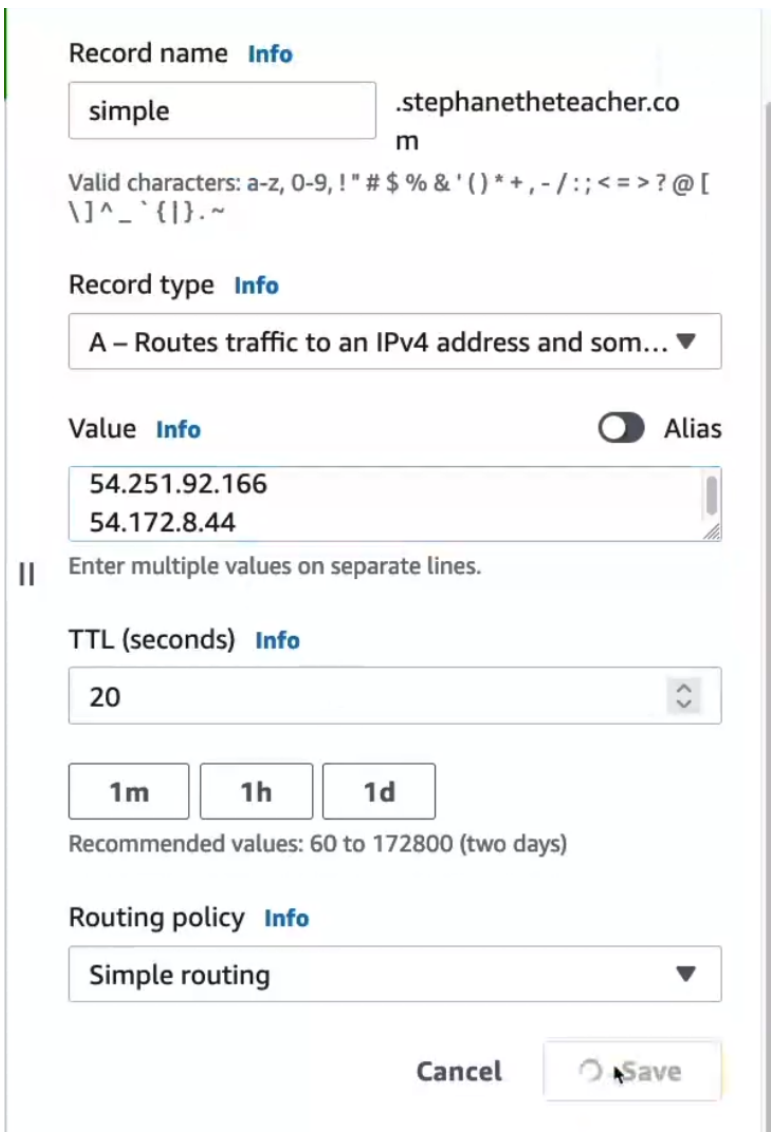
SAA-2

Route 53 - Routing Policies

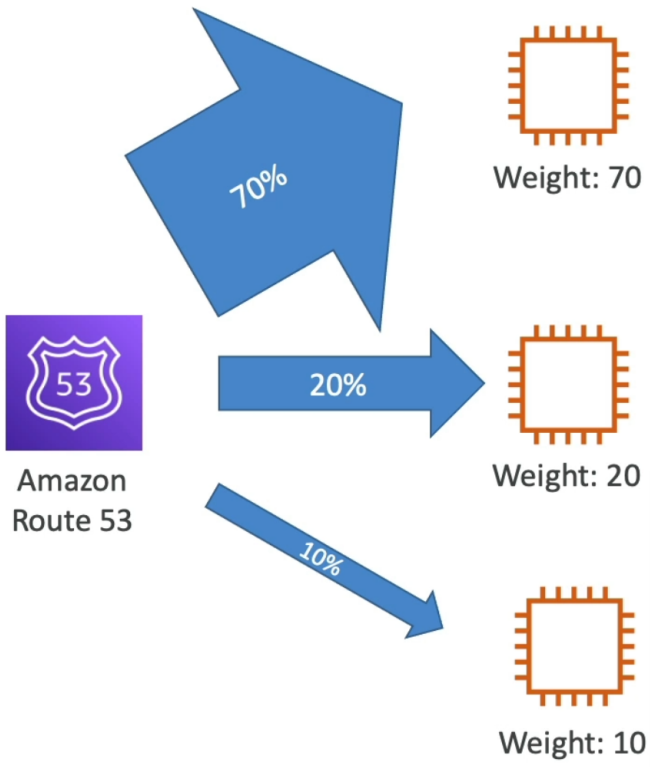
- Define how Route 53 responds to DNS queries
- Don't get confused by the word "Routing"
  - It's not the same as Load balancer routing which routes the traffic
  - DNS does not route any traffic, it only responds to the DNS queries
- Route 53 Supports the following Routing Policies
  - Simple
    - Typically, route traffic to a single resource
    - Can specify multiple values in the same record
    - **If multiple values are returned, a random one is chosen by the client**
    - When Alias enabled, specify only one AWS resource
    - Can't be associated with Health Checks





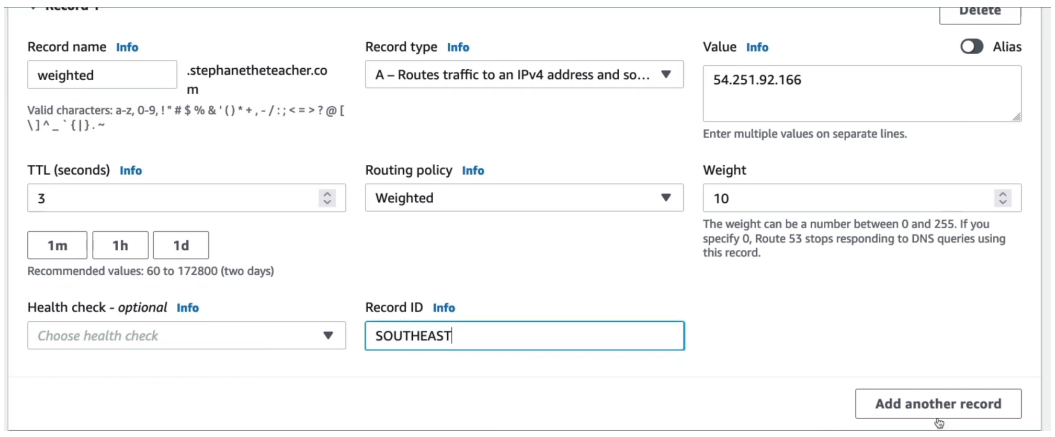


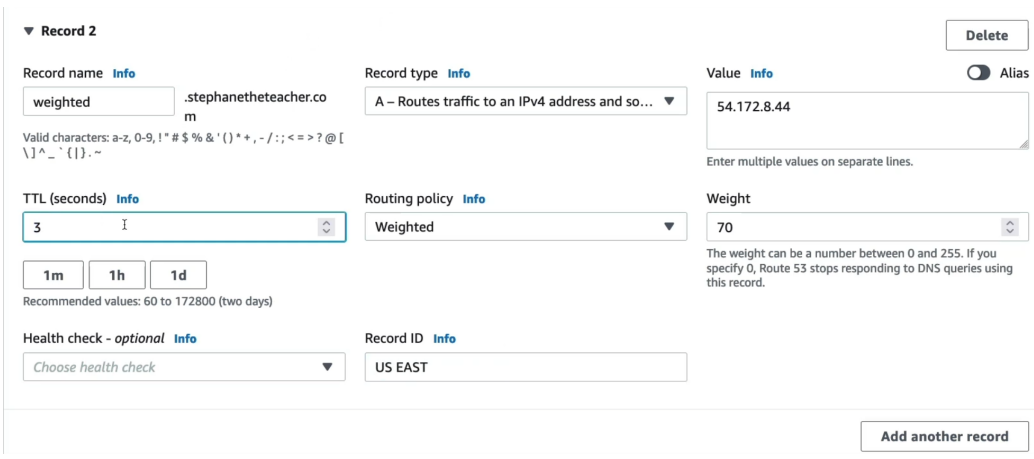
- Weighted
  - Control the % of the requests that go to each specific resource
  - Assign each record a relative weight:
    - $traffic\ (\%) = \frac{Weight\ for\ a\ specific\ record}{Sum\ of\ all\ the\ weights\ for\ all\ records}$
    - Weight don't need to sum up to 100

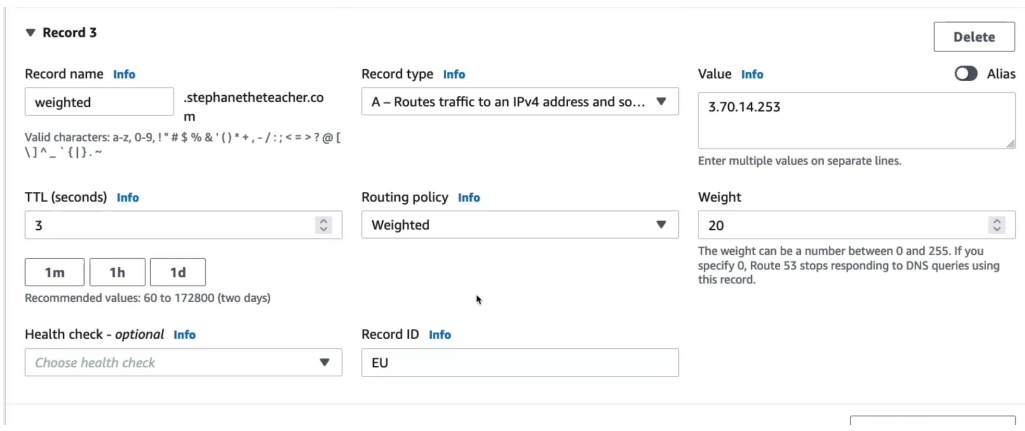


- DNS records must have the same name and type
- Can be associated with Health Checks
- Use cases: load balancing between regions, testing new application versions...
- **Assign a weight of 0 to a record to stop sending traffic to a resource**
- **If all records have weight of 0, then all records will be returned equally**

- hands on

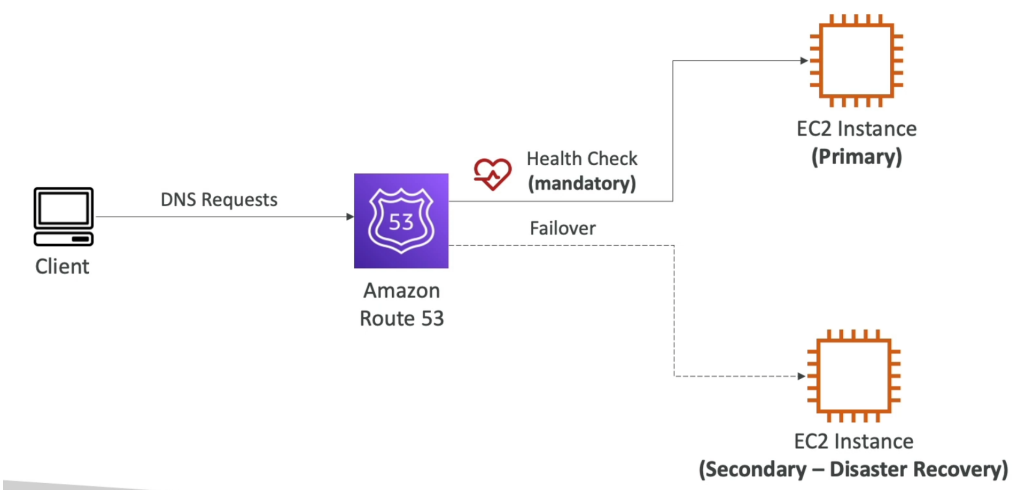




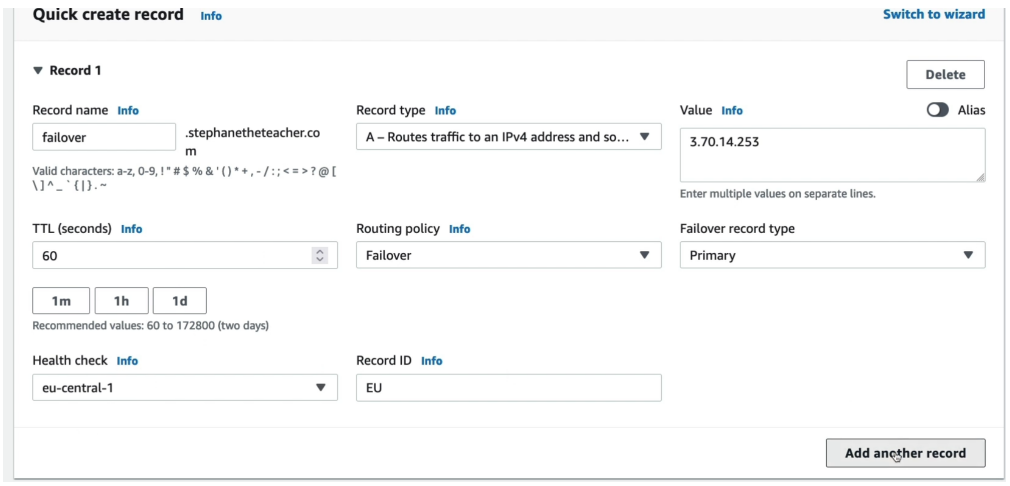


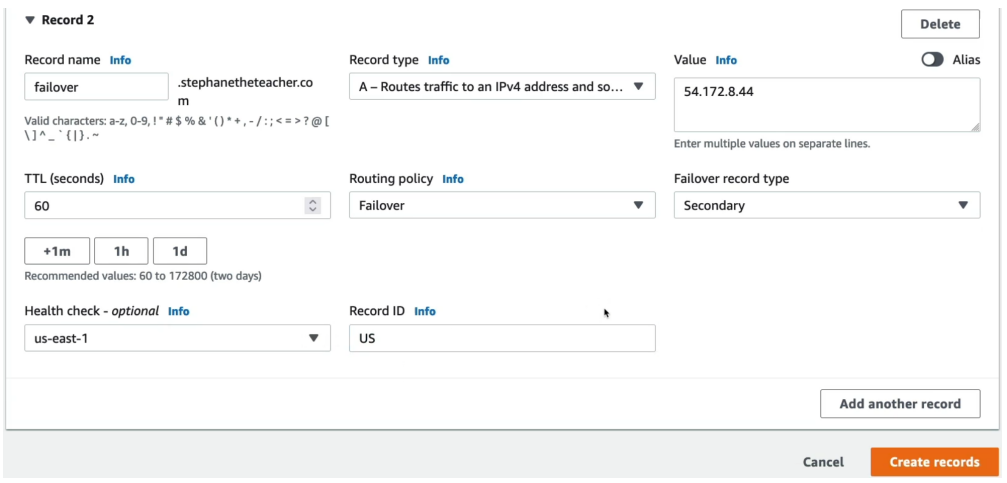
☐	weighted.stephanetheteacher....	A	Weighted	20	3.70.14.253
☐	weighted.stephanetheteacher....	A	Weighted	10	54.251.92.166
☐	weighted.stephanetheteacher....	A	Weighted	70	54.172.8.44

  - Failover
    - Active-Passive

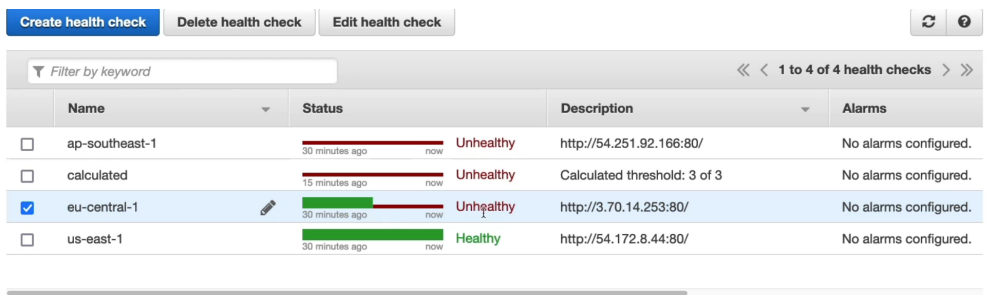


- Hands on

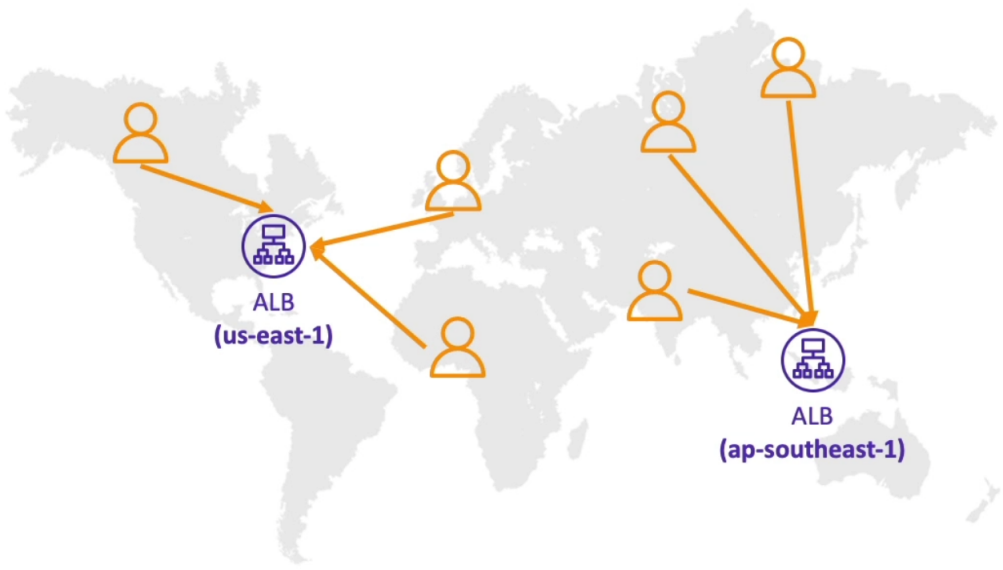




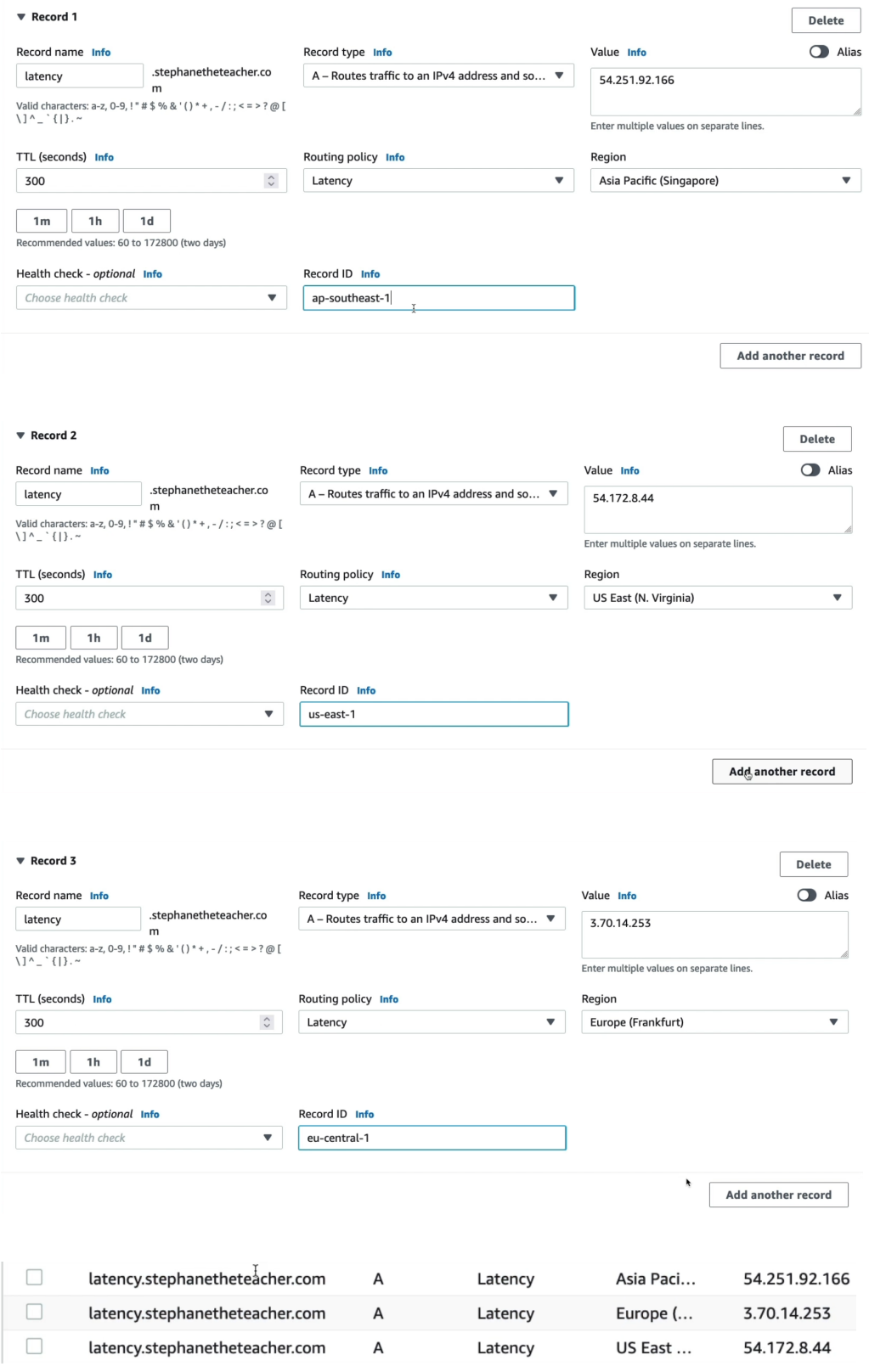
test



- Latency based
  - Redirect to the resource that has the least latency close to us
  - Supper helpful when latency for users is a priority
  - Latency is based on traffic between users and AWS Regions
  - Germany users may be directed to the US (if that's the lowest latency)
  - Can be associated with Health Checks (has a failover capability)



- hands on

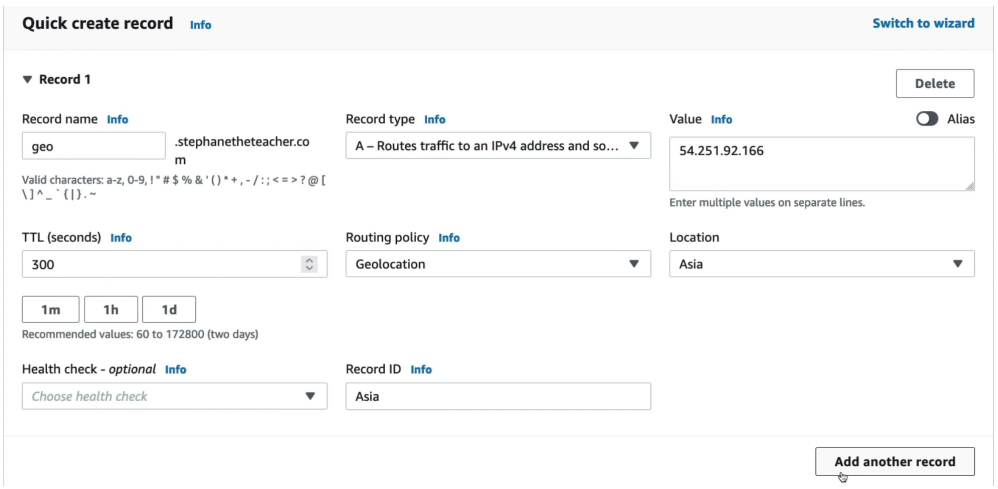


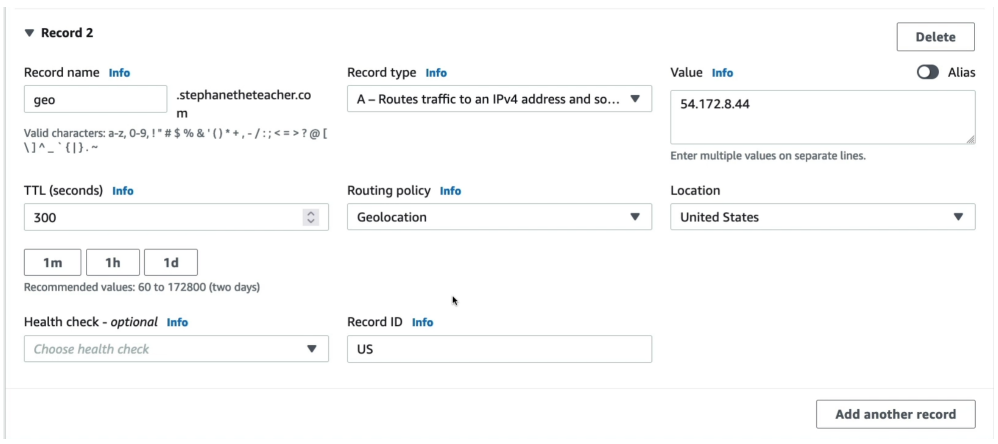
☐	latency.stephanetheteacher.com	A	Latency	Asia Paci…	54.251.92.166
☐	latency.stephanetheteacher.com	A	Latency	Europe (…	3.70.14.253
☐	latency.stephanetheteacher.com	A	Latency	US East …	54.172.8.44

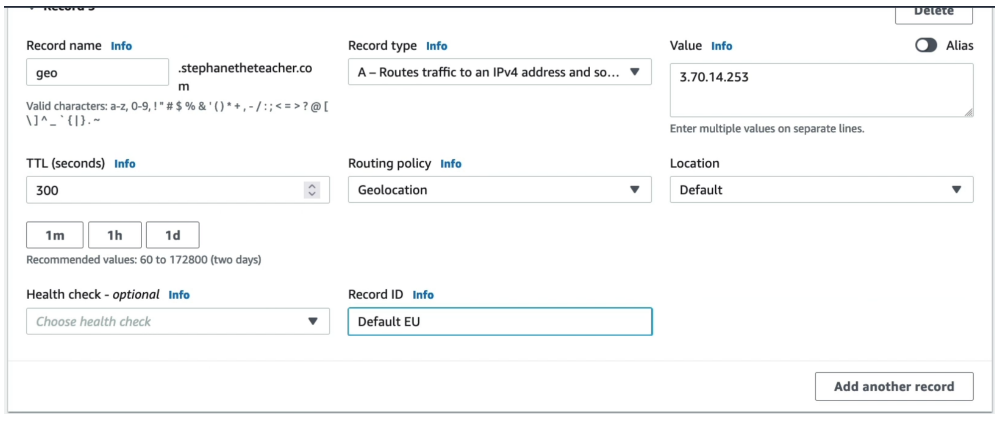
- Geolocation
  - Different from Latency-based!
  - This routing is based on user location
  - Specify location by Continent, Country or by US State (if there's overlapping, most precise location selected)
  - Should create a "Default" record (in case there's no match on location)
  - Use cases: website localization, restrict content distribution, load balancing, ...

- Can be associated with Health Checks

- Hands on



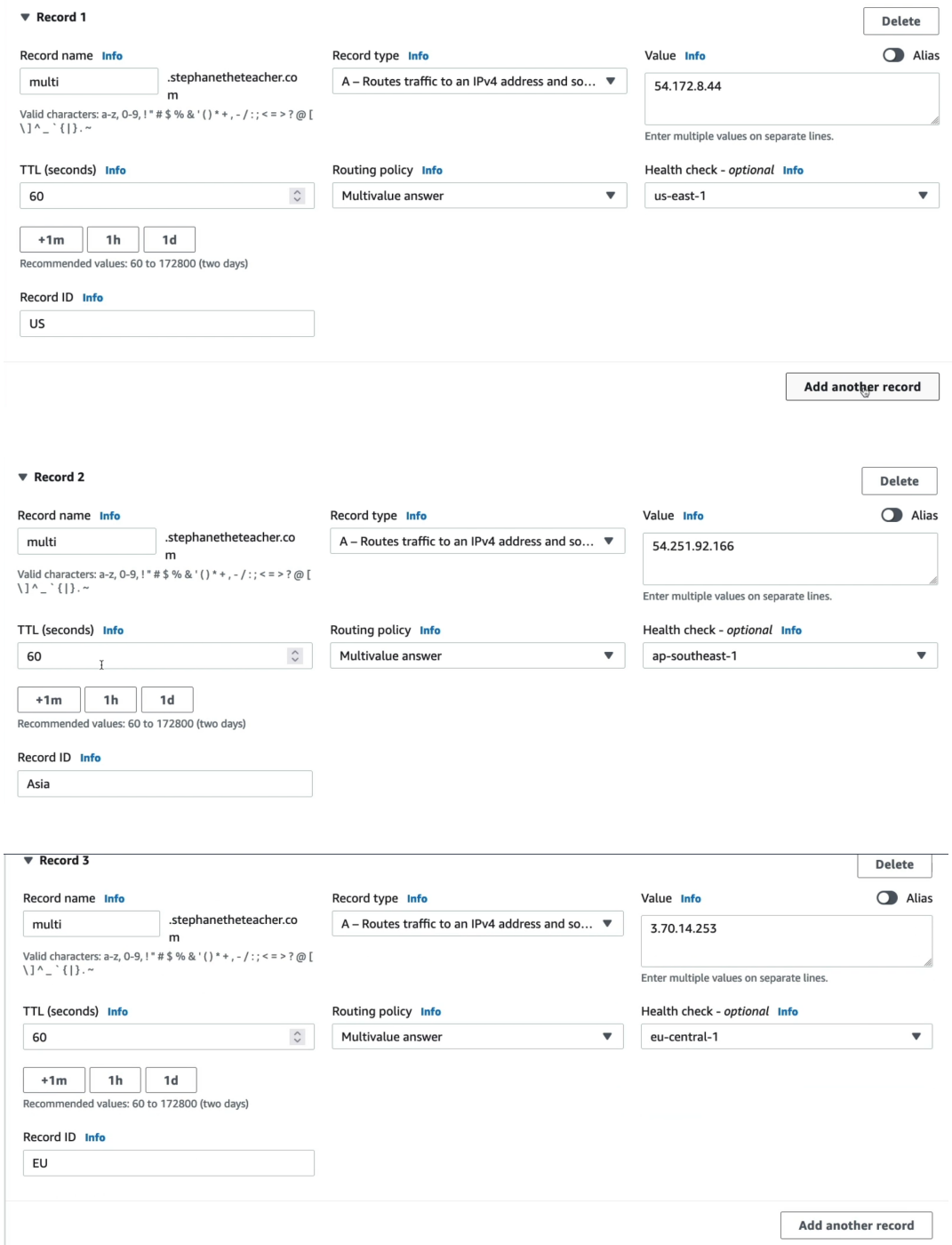




  - Multi-Value Answer
    - Use when routing traffic to multiple resources
    - Route 53 return multiple values/resources
    - Can be assosiated with Health Checks (return only values for healthy resources)
    - Up to 8 healthy records are returned for each Multi-Value query
    - **Multi-Value is not a substitute for having an ELB**

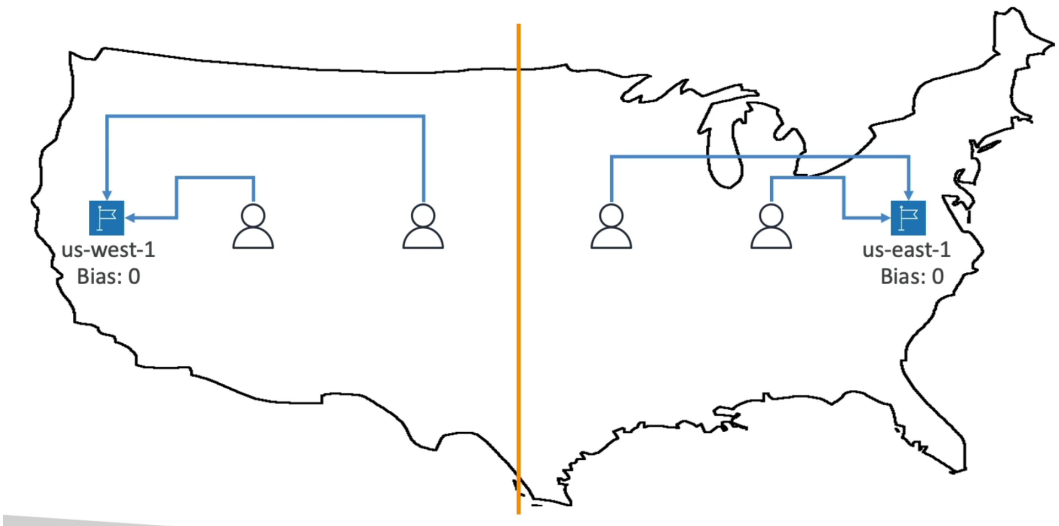
Name	Type	Value	TTL	Set ID	Health Check
www.example.com	A Record	192.0.2.2	60	Web1	A
www.example.com	A Record	198.51.100.2	60	Web2	B
www.example.com	A Record	203.0.113.2	60	Web3	C

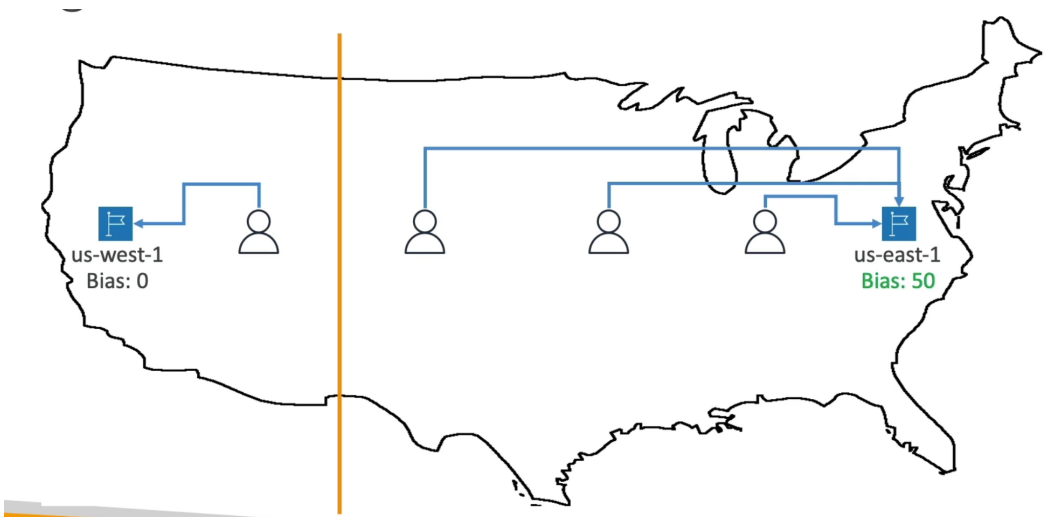
- Hands on



- Geoproximity (using Route 53 Traffic Flow feature)
  - Route traffic to your resources based on the geographic location of users and resources
  - Ability **to shift more traffic to resources based** on the defined **bias**
  - To change the size of the geographic region, specify **bias** values:
    - To exapand (1 to 99) - more traffic to the resource
    - To shrink (-1 to -99) - less traffic to the resource
- Resources can be:
  - AWS resources (specify AWS region)
  - Non-AWS resources (specify Latitude and Longitude)

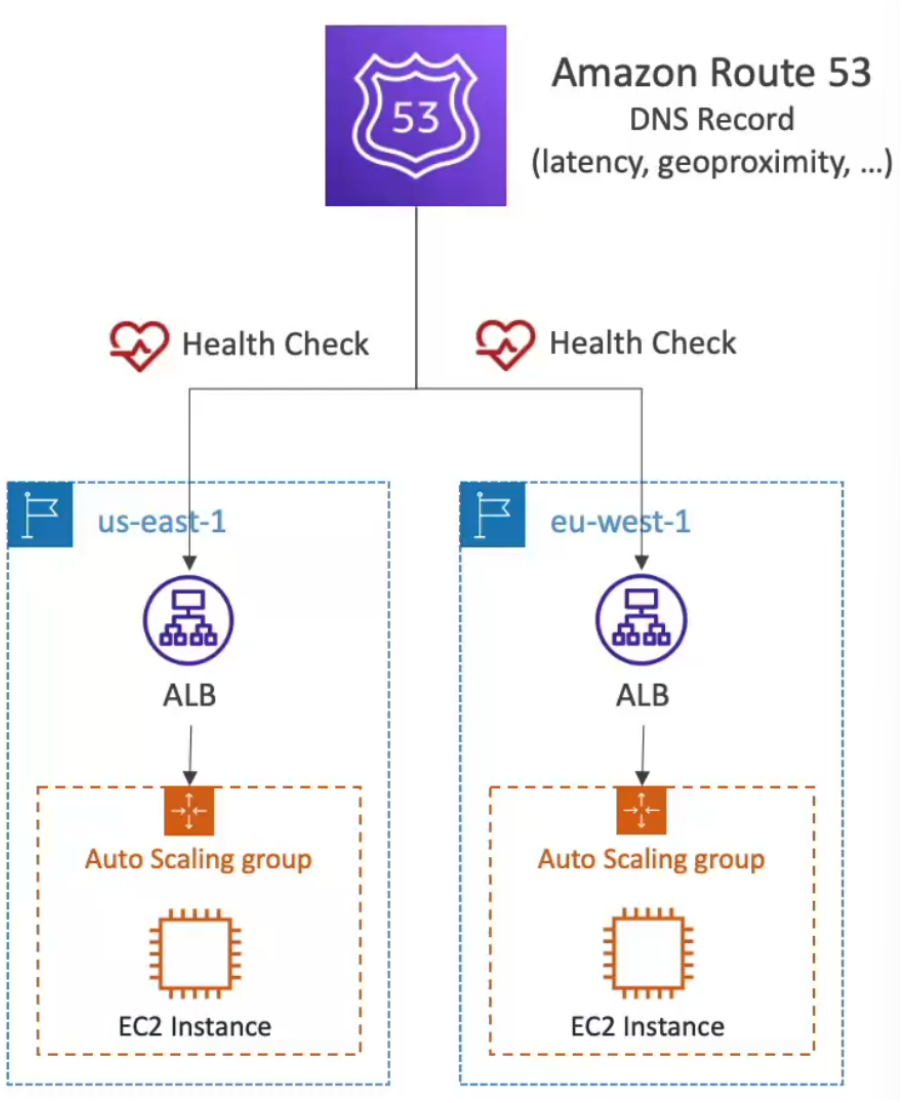
- You must use Route 53 Traffic Flow (advanced) to use this feature





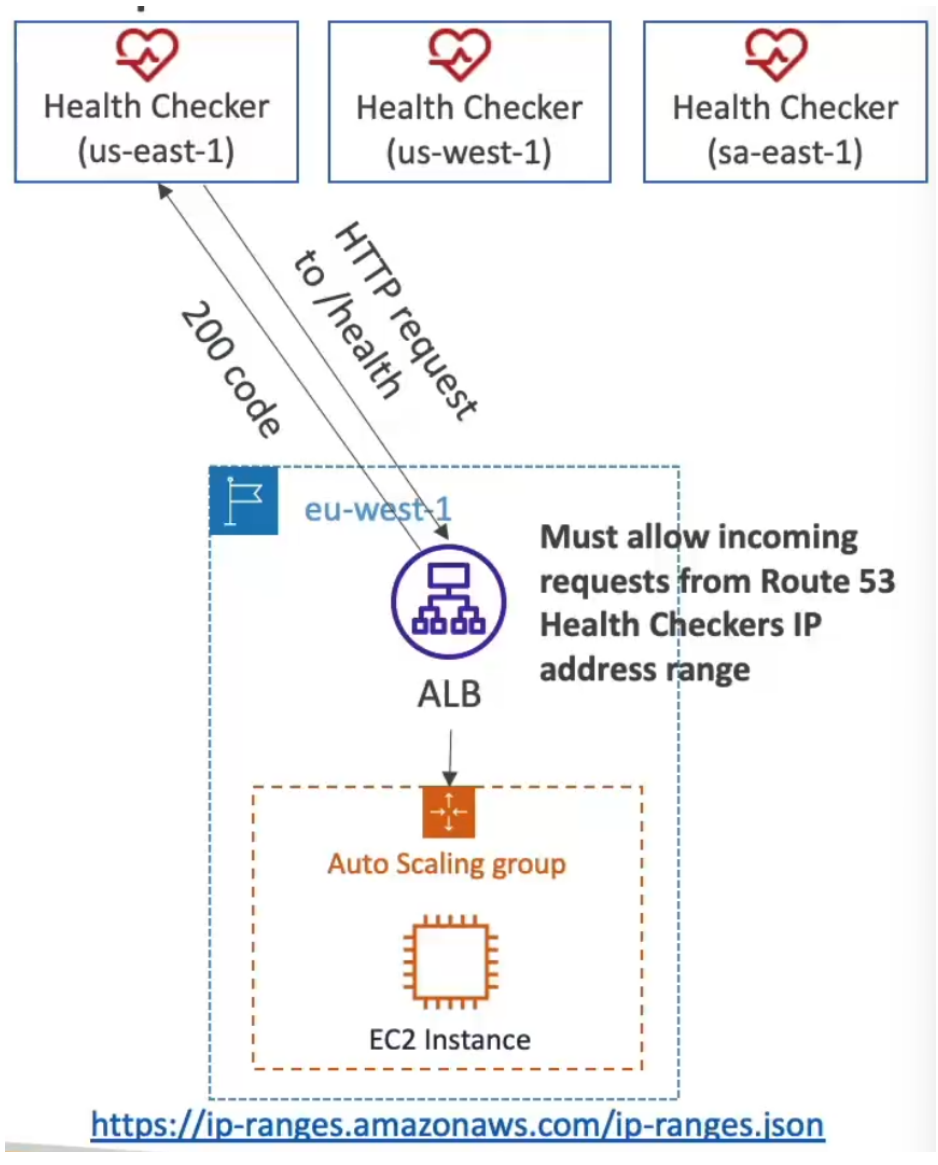
Route 53 - Health Checks

- **HTTP Health Checks are only for** public resources
- Health Check => Automated DNS Failover:
  1. Health checks that monitor an endpoint(application,server,other AWS resource)
  2. Health checks that monitor other health checks(Calculated Health Checks)
  3. Health checks that monitor CloudWatch Alarms(full control!!) - e.g., throttles of DynamoDB, alarms on RDS, custom metrics, ... (helpful for private resources)
- Health Checks are integreated with CW metrics



Health Checks - Monitor an Endpoint

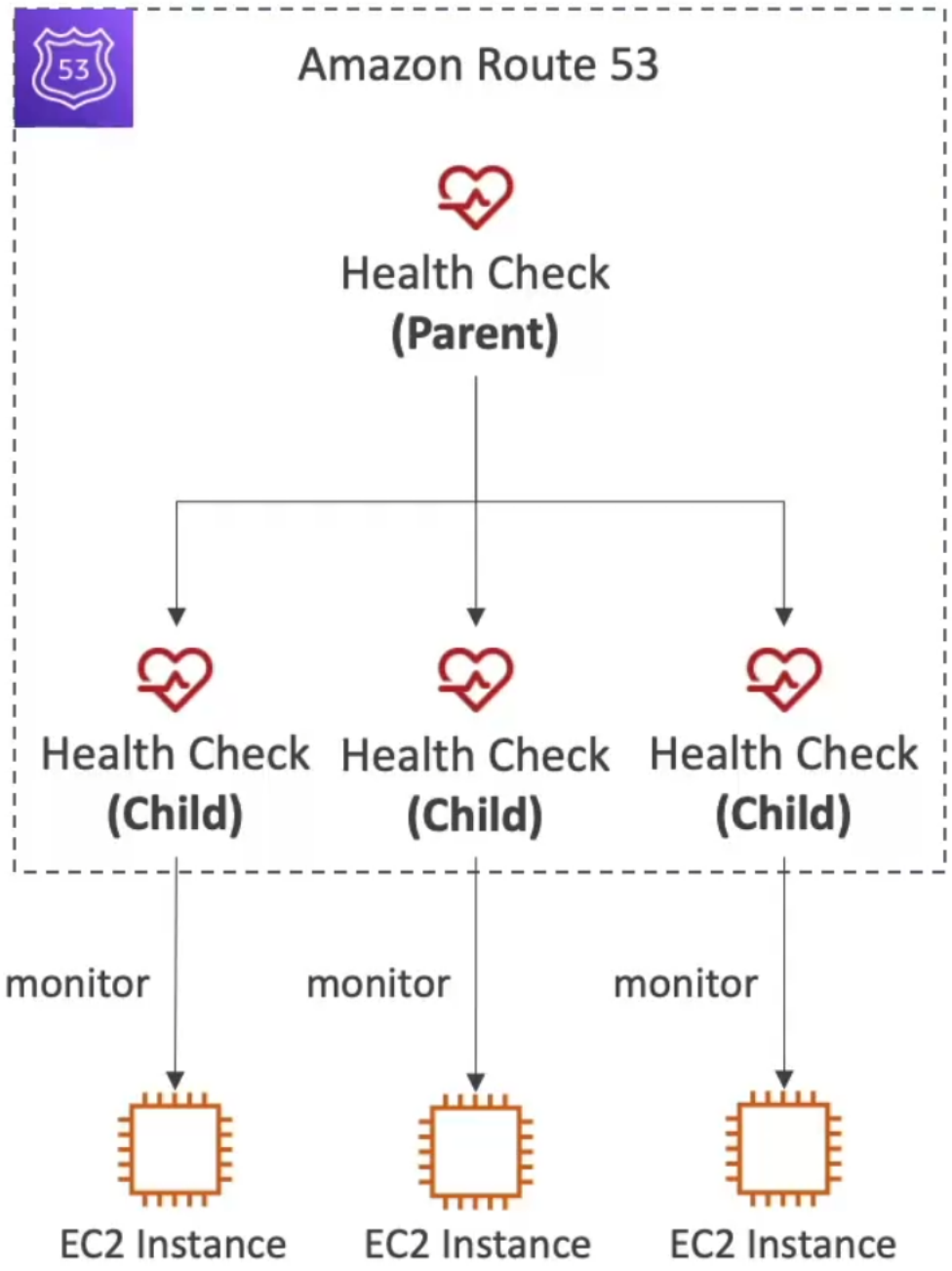
- **About 15 global health checkers will check the endpoint health**
  - Healthy/Unhealthy Threshold - 3 (default)
  - Interval - 30 sec (can set to 10 sec - higher cost)
  - Supported protocol: HTTP, HTTPS and TCP
  - If > 18% of health checkers report the endpoint is healthy, Route 53 considers it **Healthy**.Otherwise, it's **Unhealthy**
  - Ability to choose which locations you want Route 53 to use
- Health Checks pass only when the endpoint responds with the 2xx and 3xx status codes
- Health Checks can be setup to pass / fail based on the text in the first **5120 bytes** of the response
- Configure you router/firewall to allow incoming requests from Route 53 Health Checkers



Route 53 - Calculated Health Checks

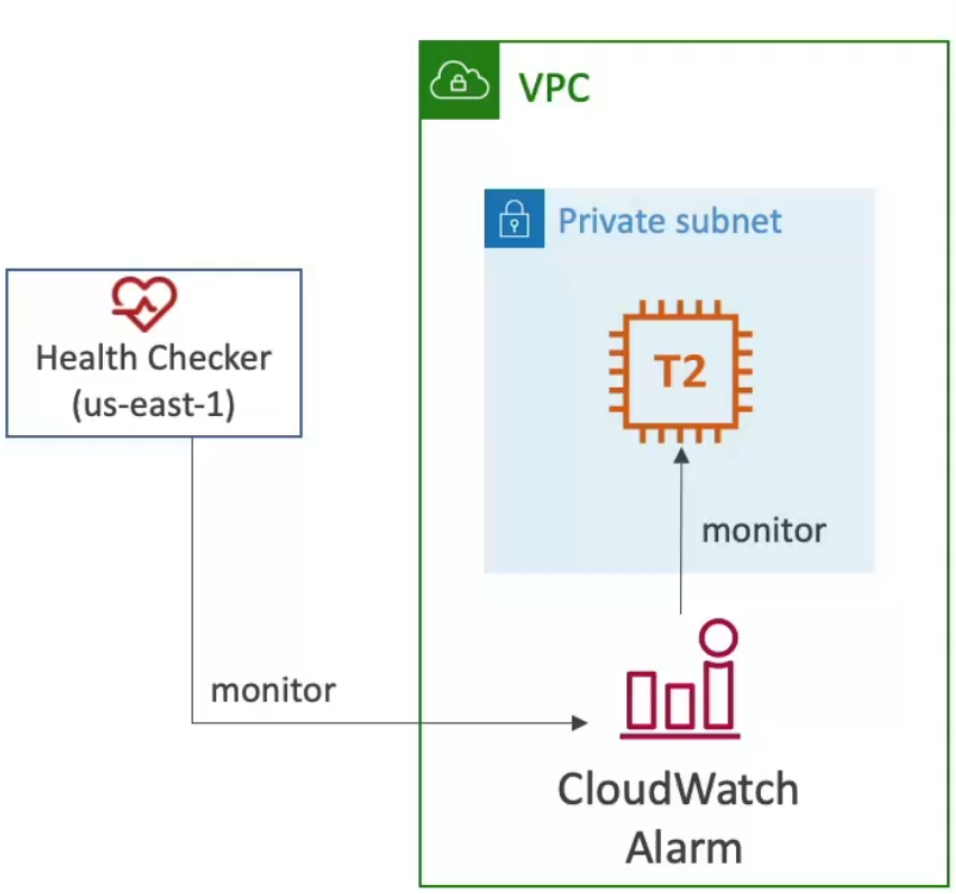
- Combine the results of multiple Health Checks into a single Health Check
- You can use **OR**, **AND**, or **NOT**
- Can monitor up to 256 Child Health Checks
- Specify how many of the health checks need to pass to make the parent pass

- Usage: perform maintenance to your website without causing all health checks to fail

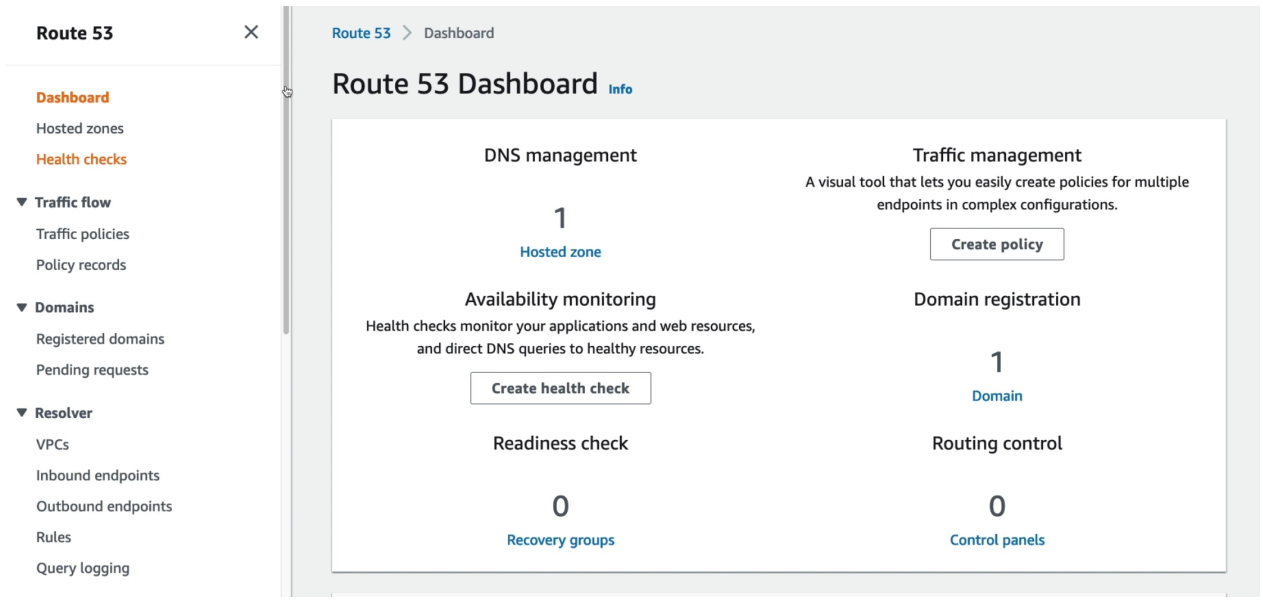


Health Checks - Private Hosted Zones

- Route 53 health checkers are outside the VPC
- They can't access **private** endpoints(private VPC or on-premises resource)
- You can create a **CloudWatch Metric** and associate a **CloudWatch Alarm**, then create a Health Check that checks the alarm itself



Health Checks - Hands on



Configure health check

Route 53 health checks let you track the health status of your resources, such as web servers or mail servers, and take action when an outage occurs.

Name us-east-1

What to monitor
- (•) Endpoint
- ( ) Status of other health checks (calculated health check)
- ( ) State of CloudWatch alarm

Monitor an endpoint

Multiple Route 53 health checkers will try to establish a TCP connection with the following resource to determine whether it's healthy. Learn more

Specify endpoint by (•) IP address ( ) Domain name

Protocol HTTP

IP address * 54.172.8.44

Host name www.example.com

Port * 80

Path / images

▾ Advanced configuration

Request interval (•) Standard (30 seconds) ( ) Fast (10 seconds)

Failure threshold * 3

String matching (•) No ( ) Yes

Latency graphs [ ]

Invert health check status [ ]

Disable health check [ ] By default, disabled health checks are considered healthy. Learn more

Health checker regions ( ) Customize (•) Use recommended

US East (N. Virginia)
US West (N. California)
US West (Oregon)
EU (Ireland)
Asia Pacific (Singapore)
Asia Pacific (Sydney)
Asia Pacific (Tokyo)
South America (São Paulo)

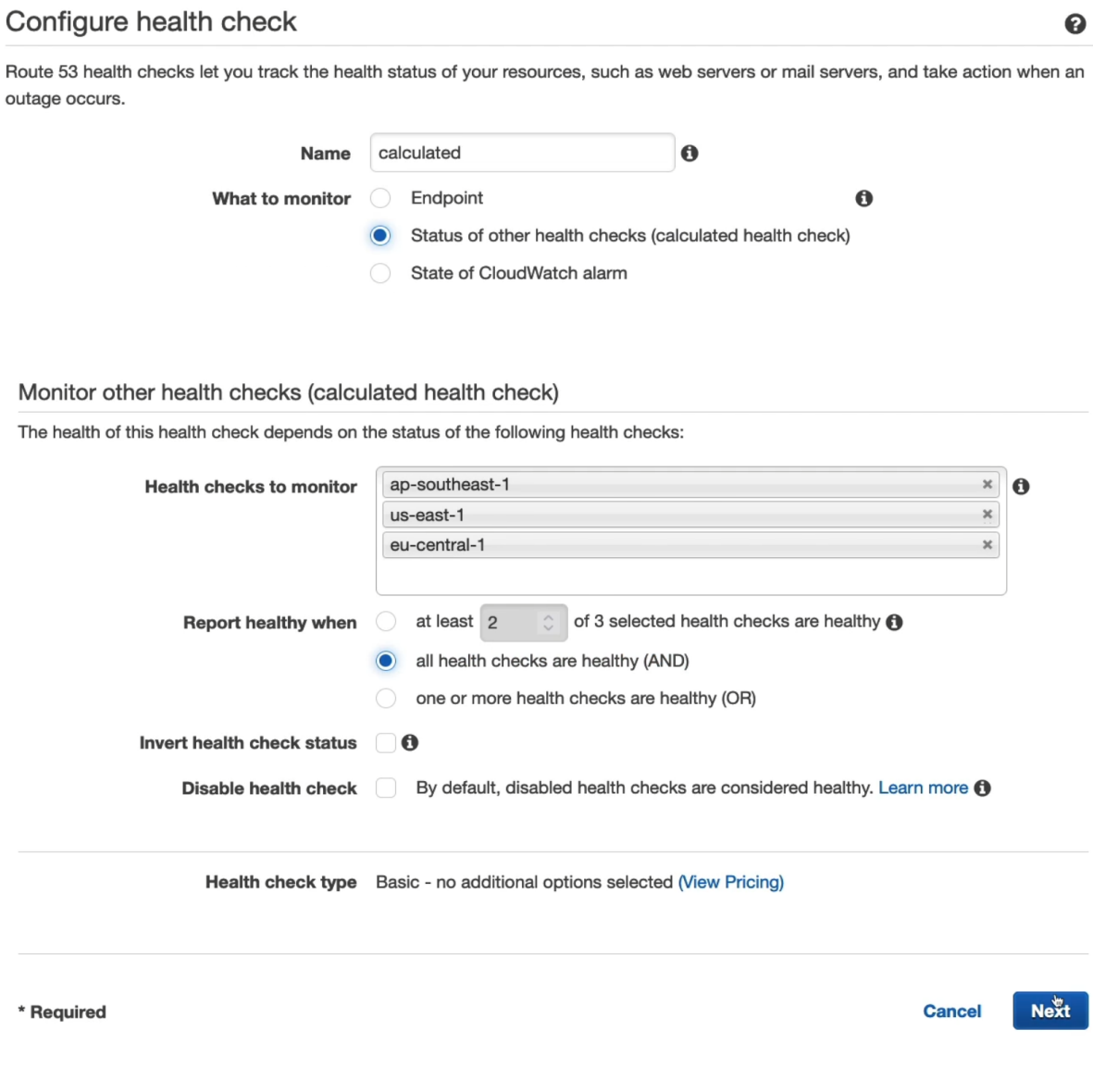
Get notified when health check fails

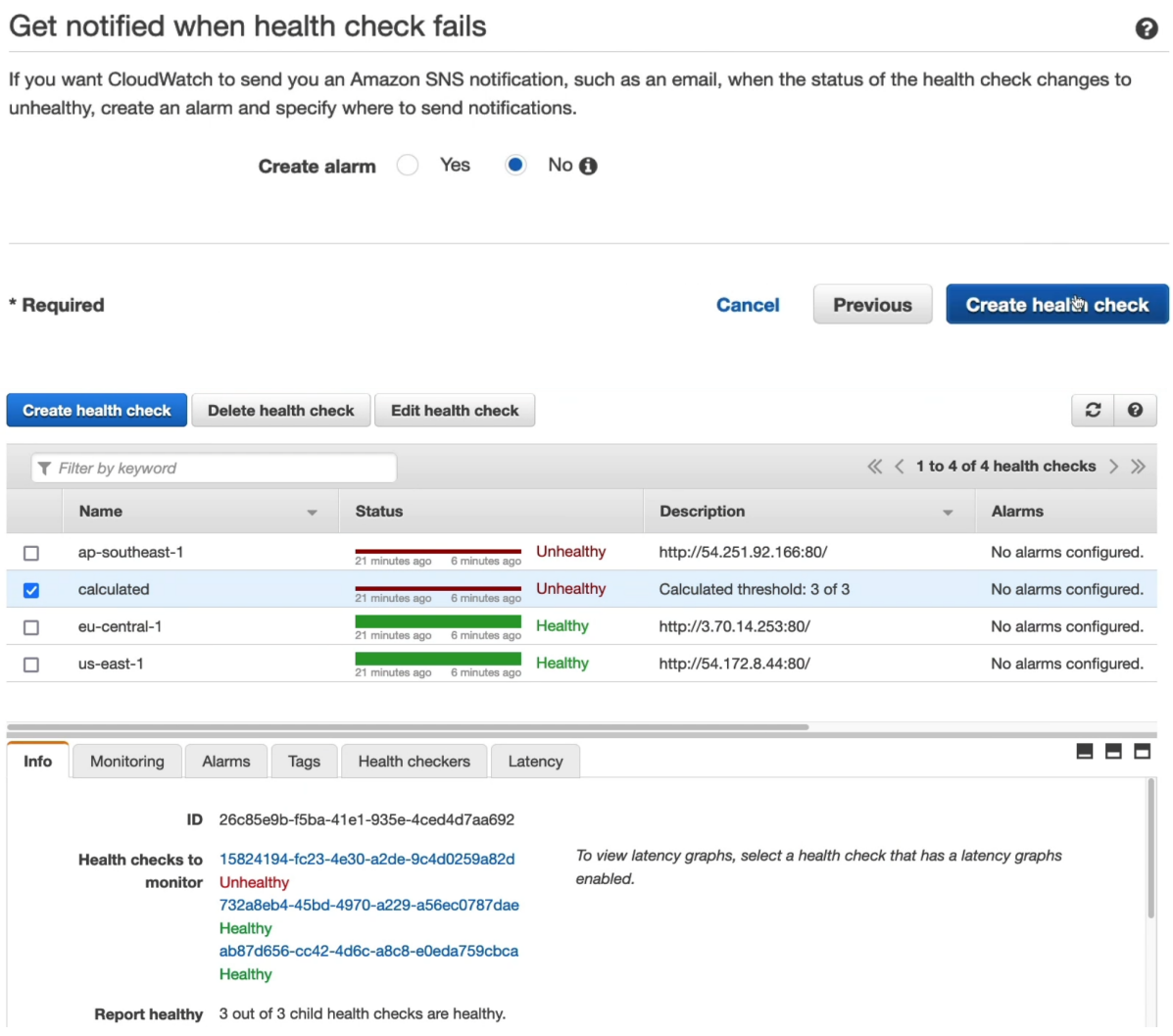
If you want CloudWatch to send you an Amazon SNS notification, such as an email, when the status of the health check changes to unhealthy, create an alarm and specify where to send notifications.

Create alarm ( ) Yes (•) No

*** Required** Cancel Previous Create health check

Hands - on calculate health check

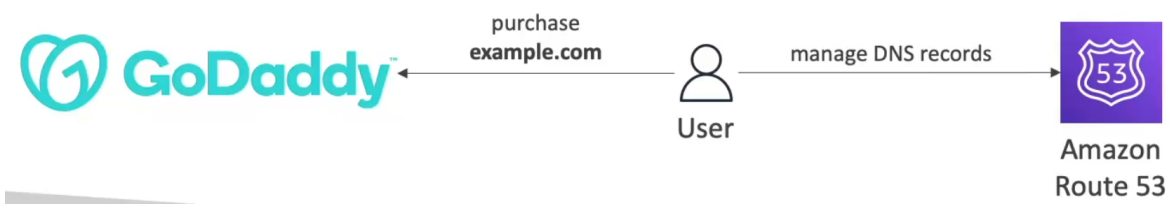




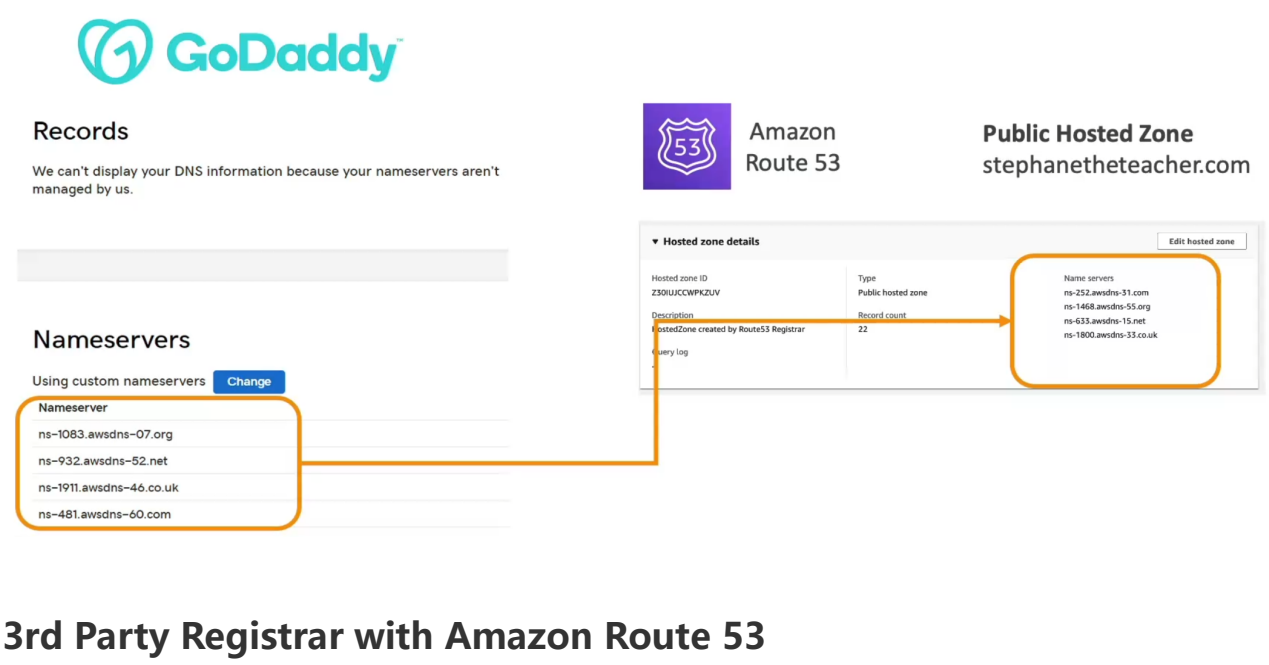
Domain Registrar vs DNS Service

- You buy or register your domain name with a Domain Registrar typically by paying annual charges (e.g., GoDaddy, Amazon Registrar Inc., ...)
- The Domain Registrar usually provides you with a DNS service to manage your DNS records
- But you can use another DNS service to manage your DNS records

- Example: purchase the domain from GoDaddy and use Route 53 to manage your DNS records



GoDaddy as Registrar & Route 53 as DNS Service



3rd Party Registrar with Amazon Route 53

- **If you buy your domain on a 3rd party registrar, you can still use Route 53 as the DNS Service provider**
- Create a Hosted Zone in Route 53
- Update NS Records on 3rd party website to use Route 53 Name Servers
- Domain Registrar != DNS Service
- But every Domain Registrar usually comes with some DNS features

---

Section Introduction

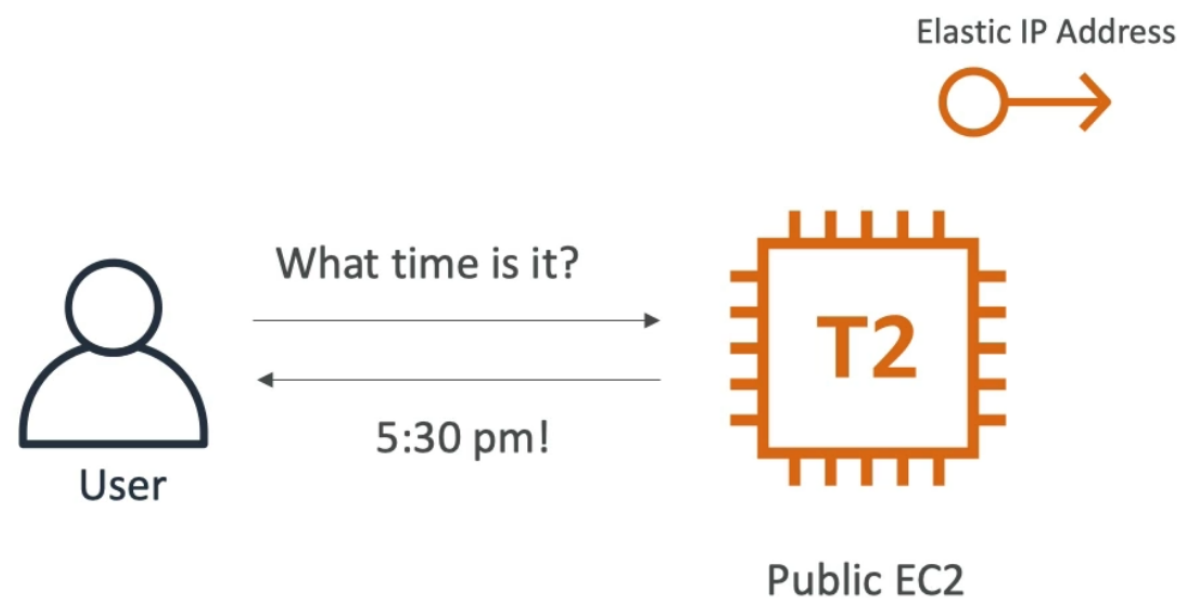
- These solutions architectures are the best part of this course
- Let's understand how all the technologies we've seen work together
- This is a section you need to be 100% comfortable with
- We'll see the progression of a Solution's architect mindset through many sample case studies:
  - WhatIsTheTime.Com
  - MyClothes.Com
  - MyWordPress.Com
  - Instantiating applications quickly
  - Beanstalk

Stateless Web App:WhatIsTheTime.Com

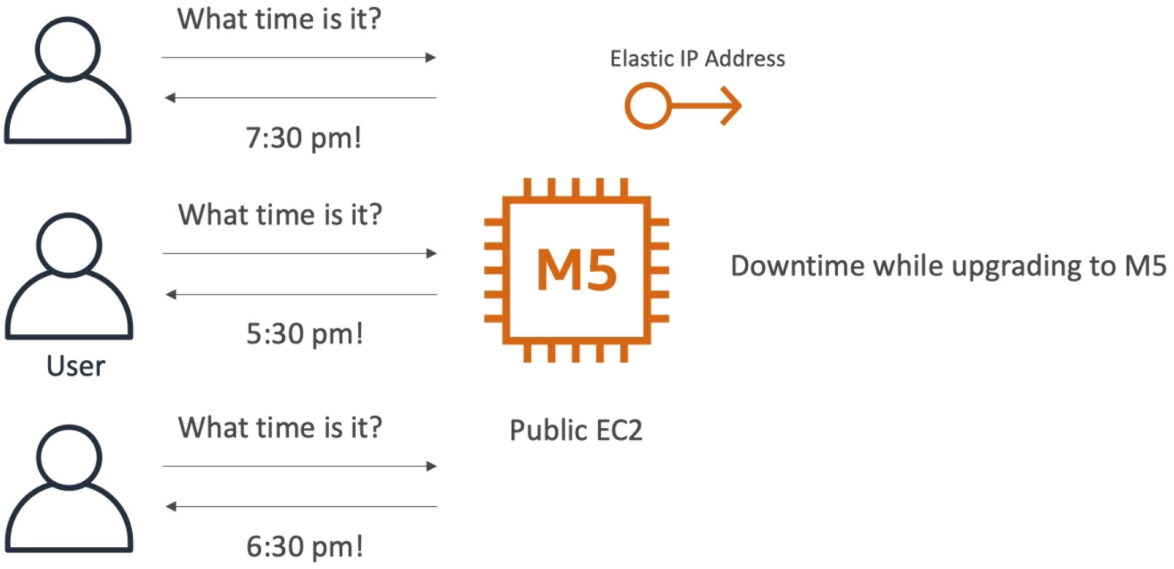
- WhatIsTheTime.com allows people to know what time it is
- We don't need a database
- We want to start small and can accept downtime
- We want to fully scale vertically and horizontally, no downtime
- Let's go through the Solutions Architect journey for this app
- Let's see how we can proceed!

Stateless web app:What time is it?

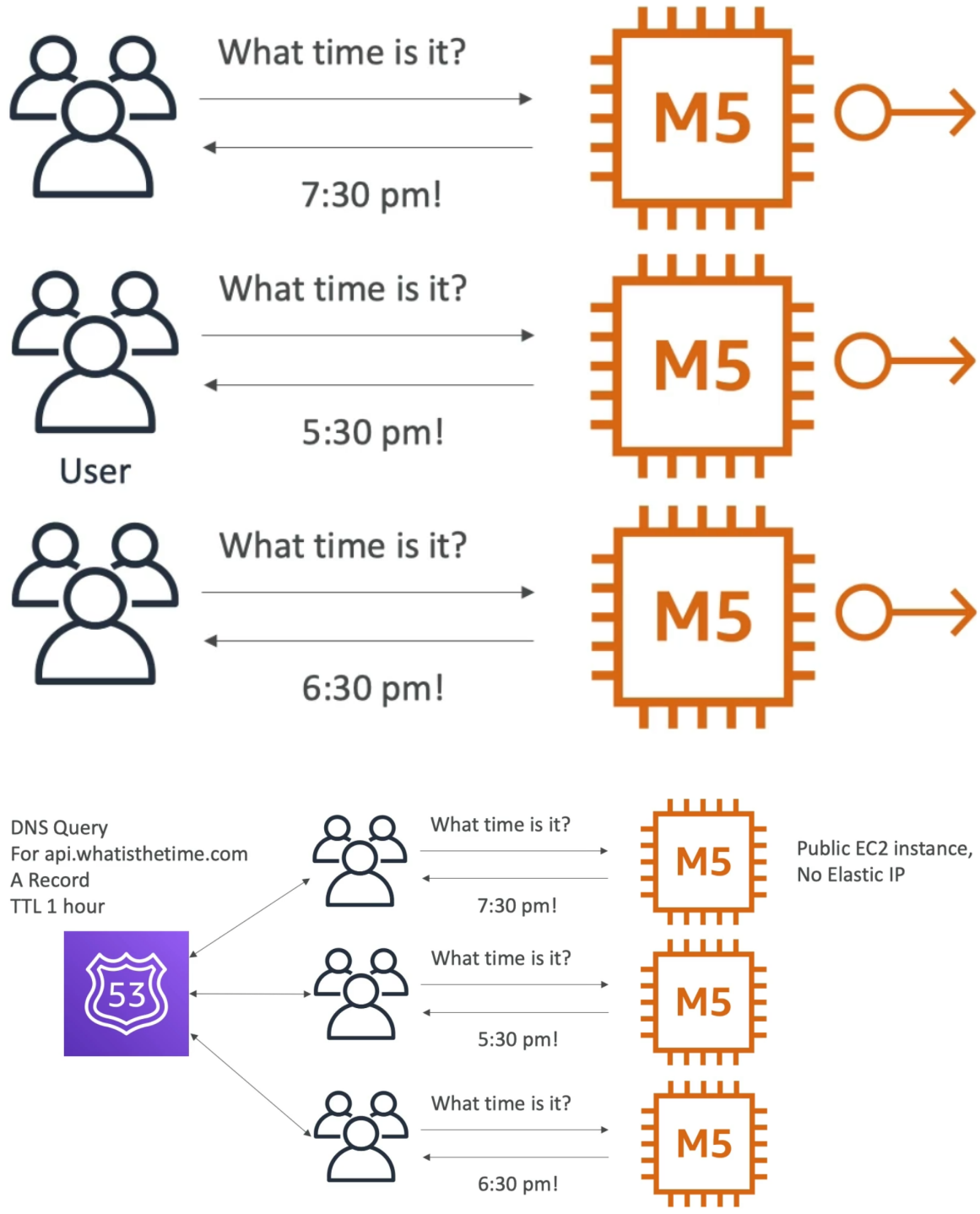
- Starting simple



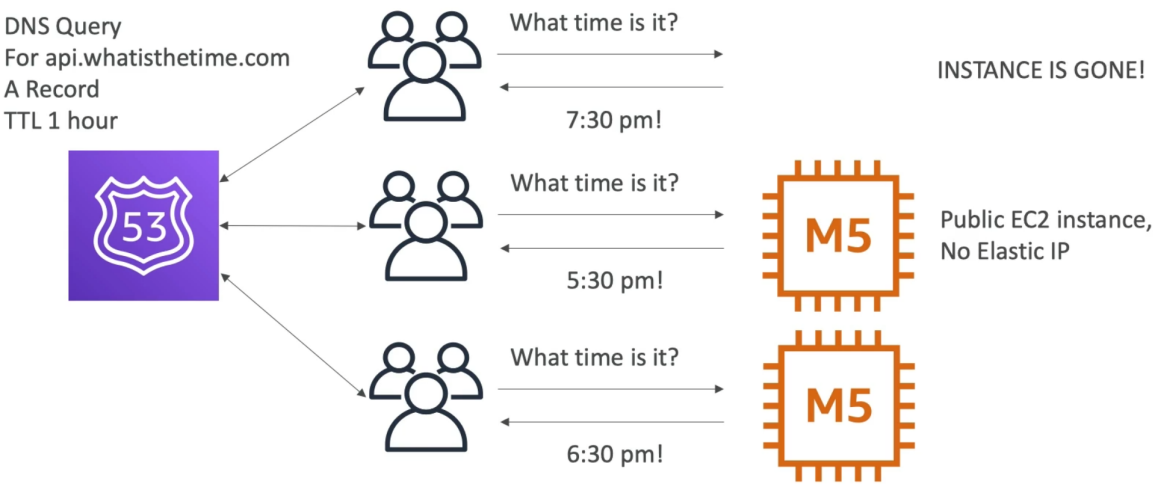
- Scaling vertically



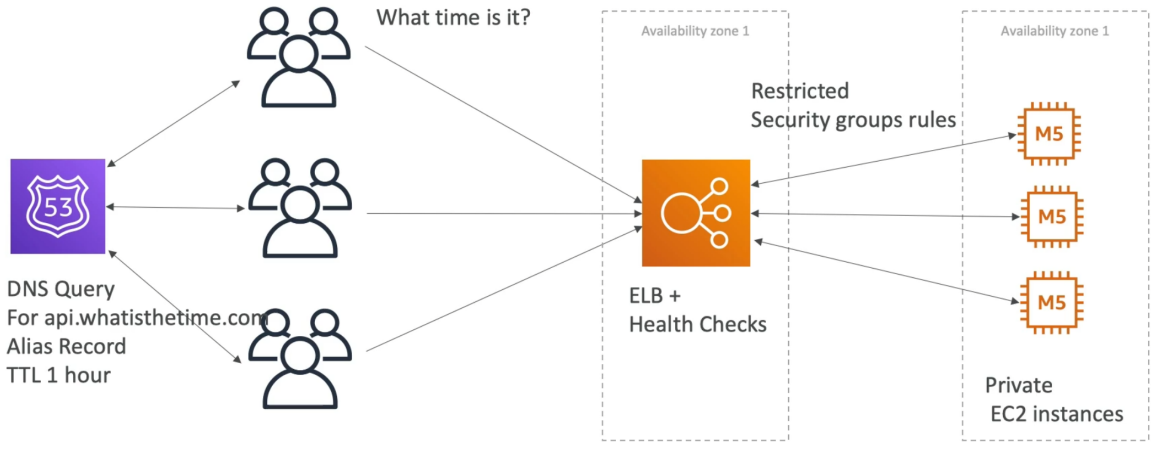
- Scaling horizontally



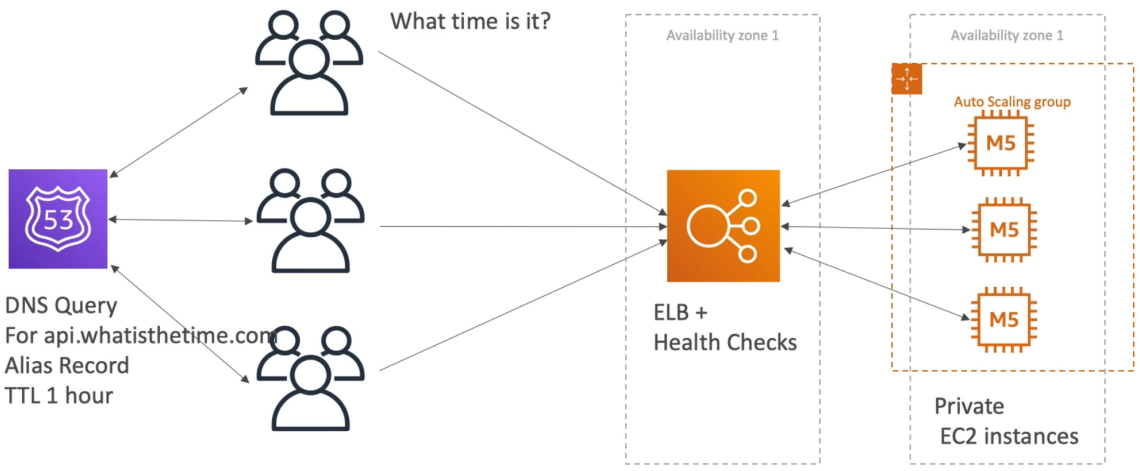
- Scaling horizontally, adding and removing instances



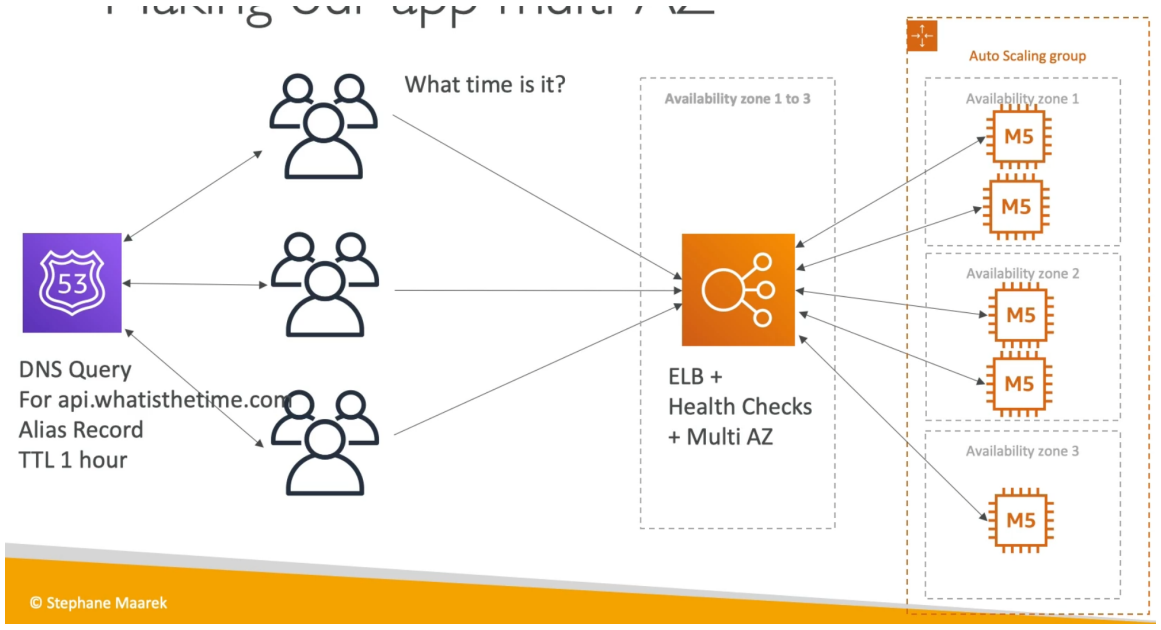
- Scaling horizontally, with a load balancer



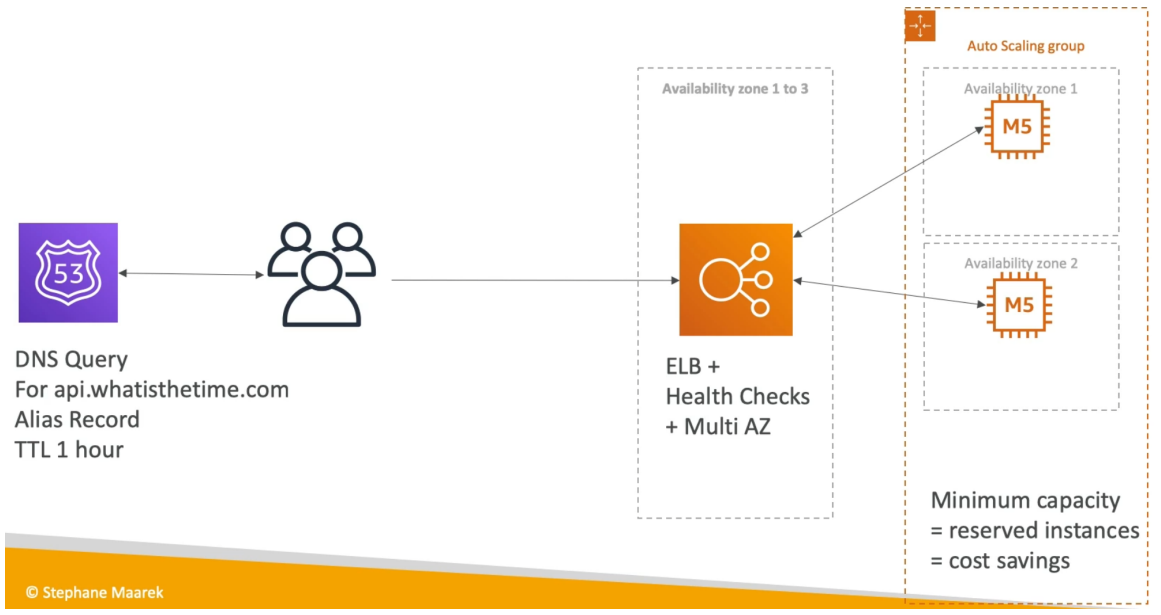
- Scaling horizontally,with an auto-scaling group



- Making our app multi-AZ



- Minimum 2 AZ => Let's reserve capacity



In this lecture we've discussed...

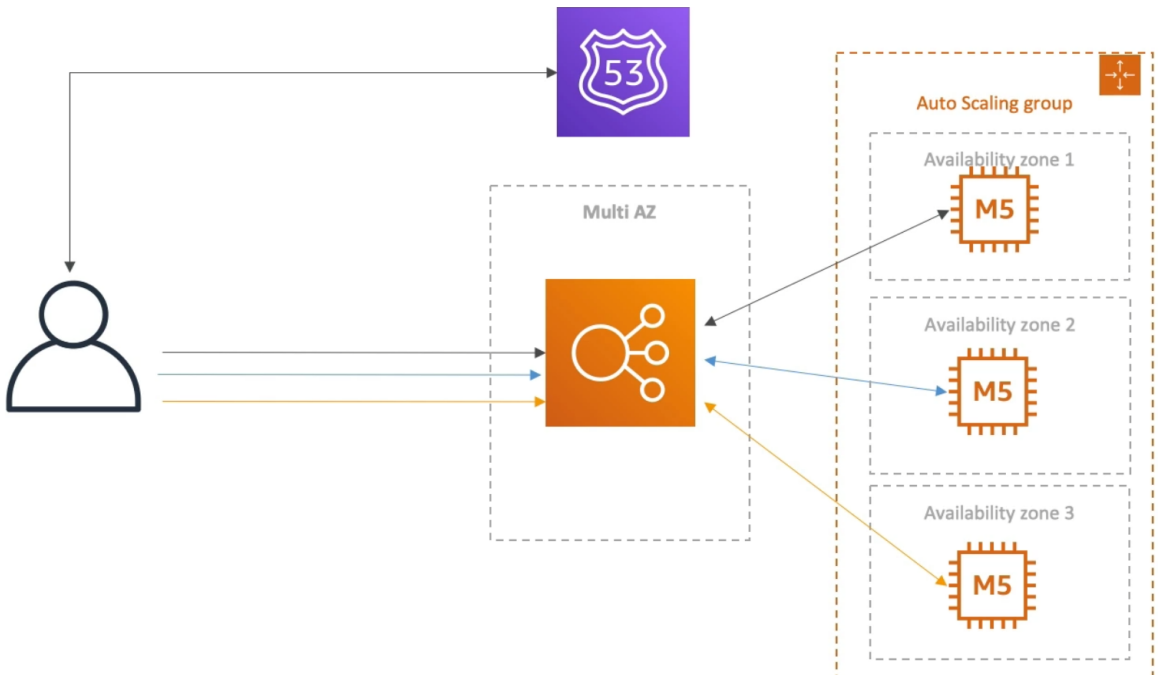
- Public vs Private IP and EC2 instances
- Elastic IP vs Route 53 vs Load Balancers
- Route 53 TTL, A records and Alias Records
- Maintaining EC2 instances manually vs Auto Scaling Groups
- Multi AZ to survive disasters
- ELB Health Checks
- Security Group Rules
- Reservation of capacity for costing savings when possible
- We're considering 5 pillars for a well architected application:costs, performance, reliability,security,operational excellence

Stateful Web App: MyClothes.com

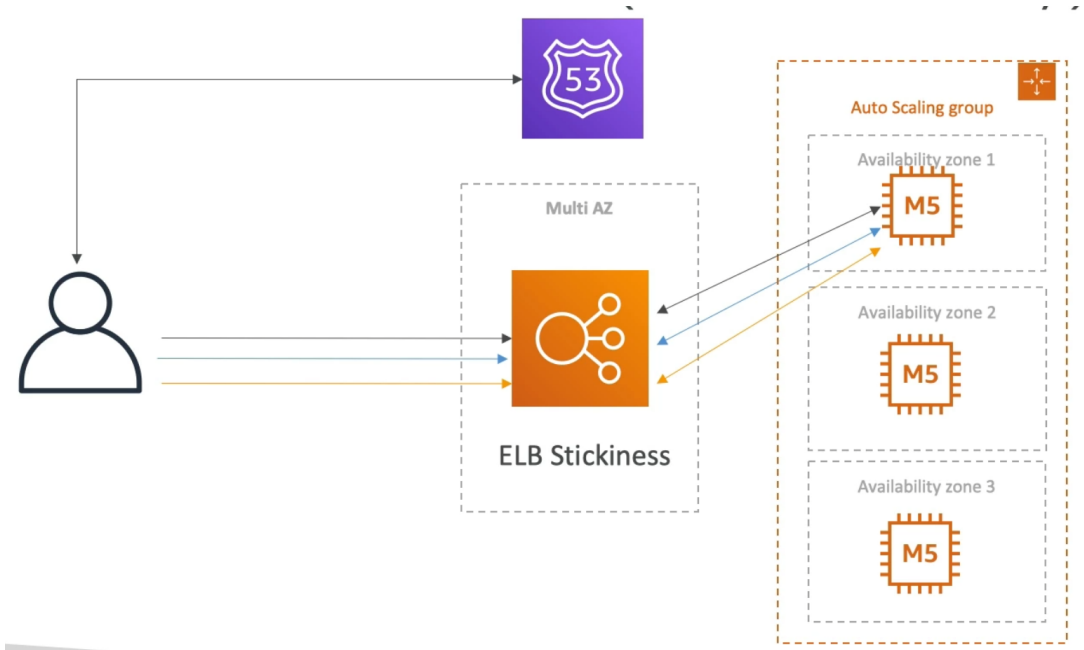
- MyClothes.com allows people to buy clothes online.
- There's a shopping cart

- Our website is having hundreds of users at the same time
- We need to scale,maintain horizontal scalability and keep our web application as stateless as possible
- Users should not lose their shopping cart
- Users should have their details (address, etc) in a database

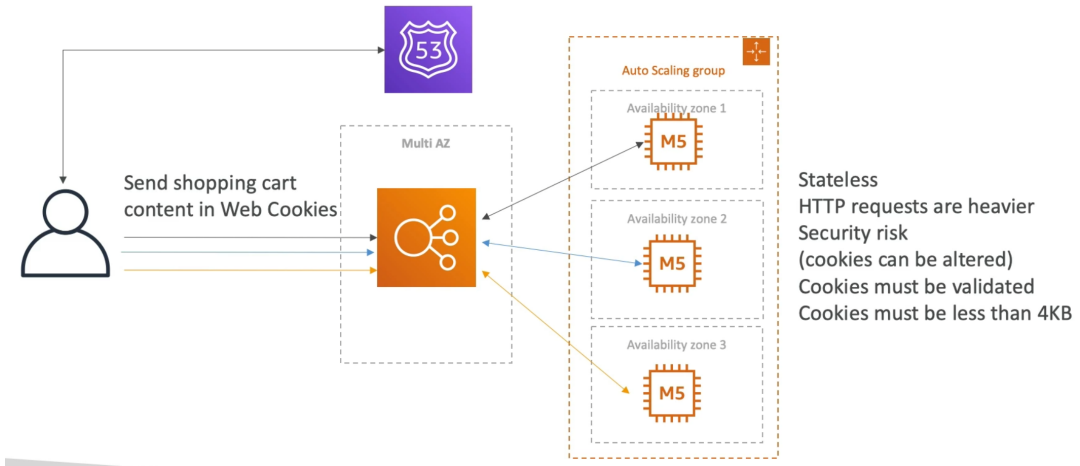
Let's see how we can proceed!



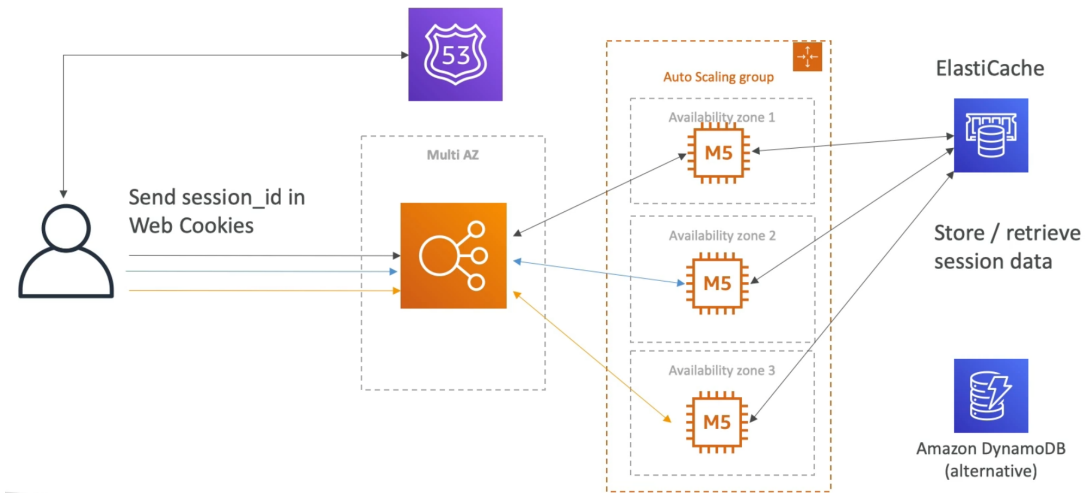
- Introduce Stickiness (Session Affinity)



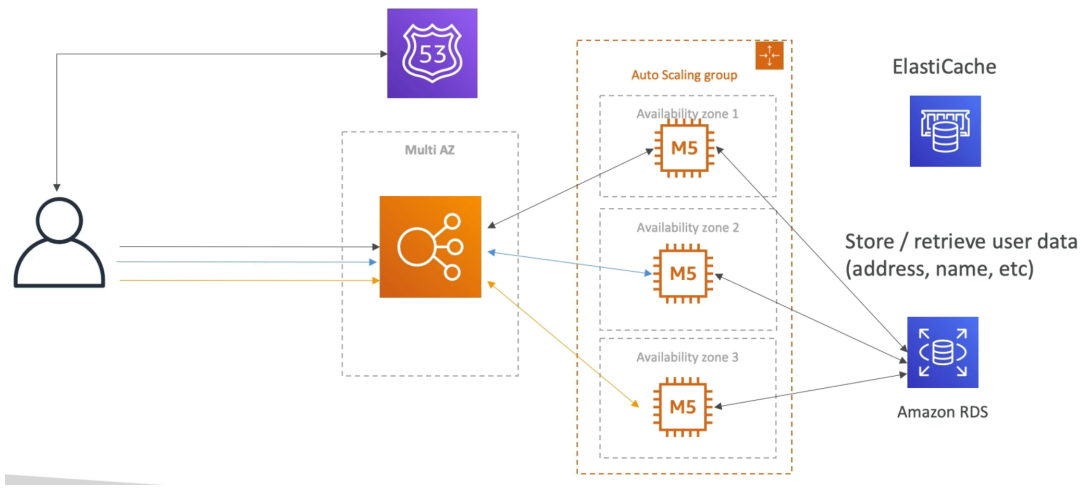
- Introduce User Cookies



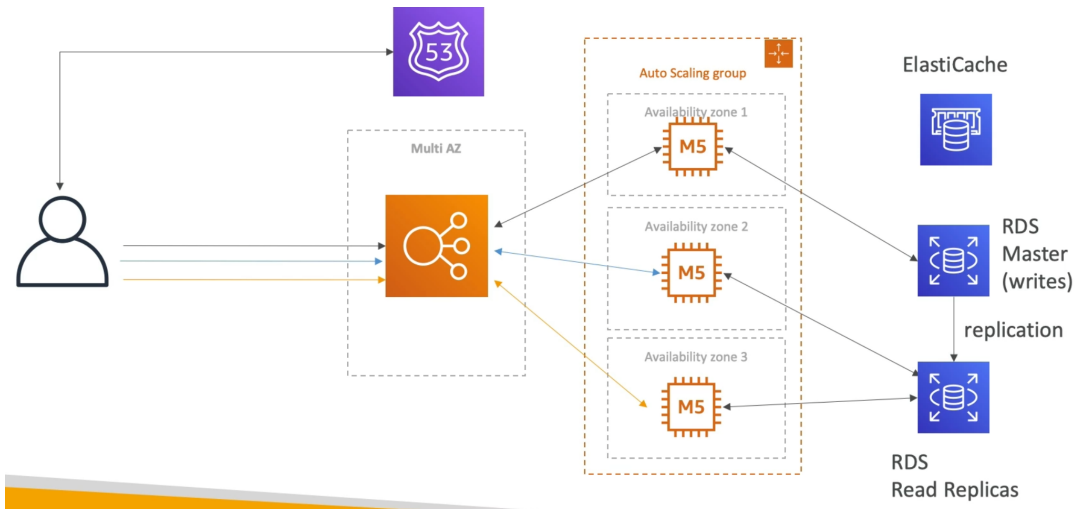
- Introduce Server Session



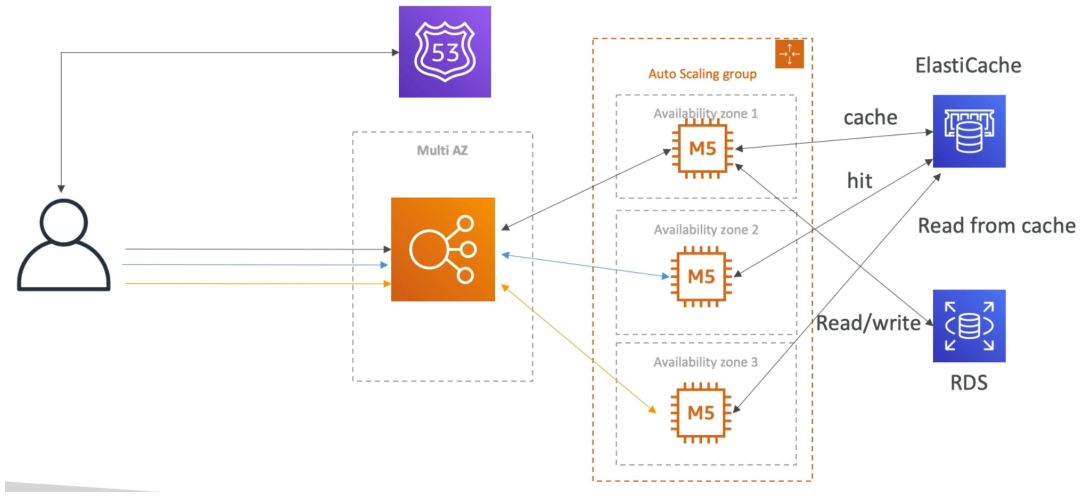
- Storing User Data in a database



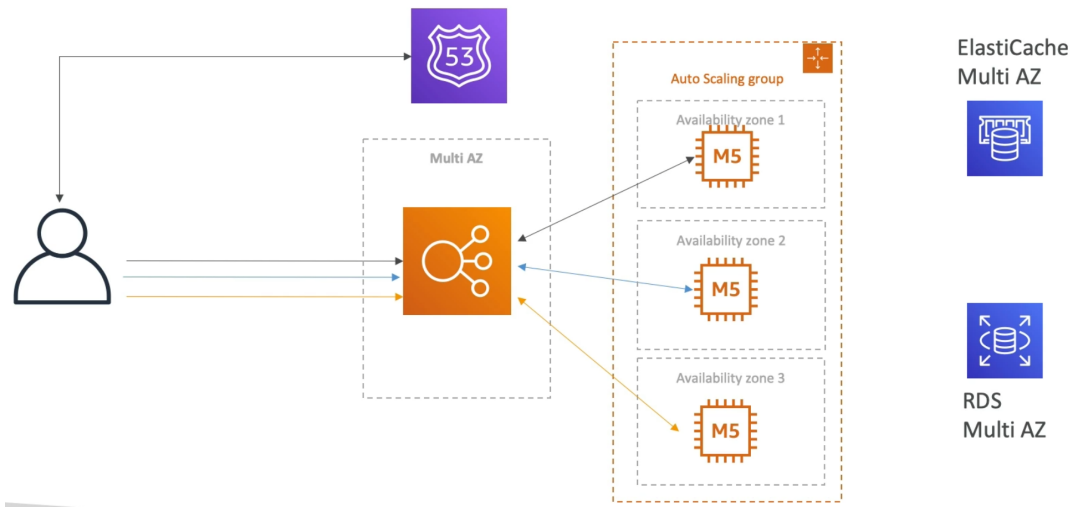
- Scaling Reads



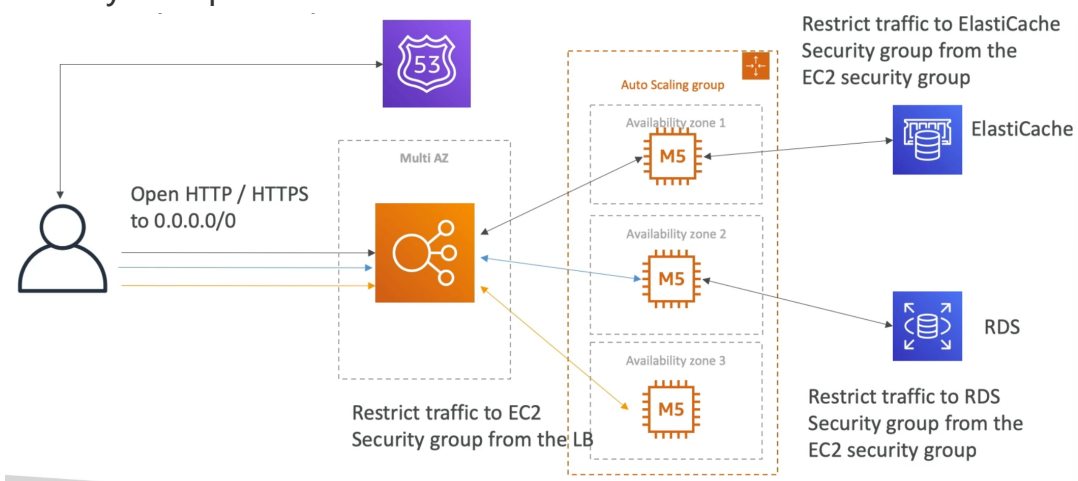
- Scaling Reads (Alternative) - Write Through



- Multi AZ - Survive disasters



- Security Groups



In this lecture we've discussed ...
3-tier architectures for web applications

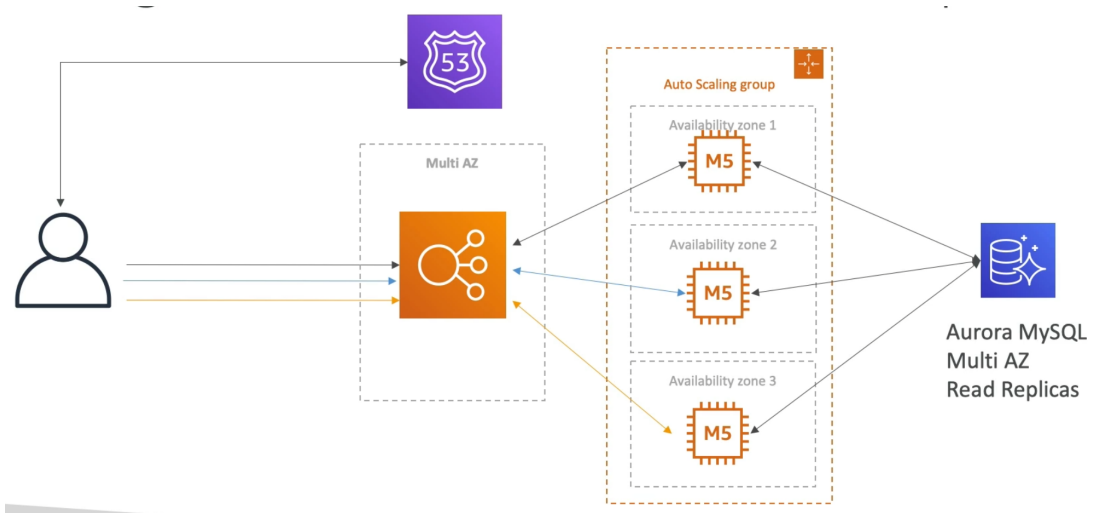
- ELB sticky sessions
- Web clients for storing cookies and making our web app stateless
- ElastiCache
  - For storing sessions(alternative: DynamoDB)
  - For caching data from RDS
  - Multi AZ
- RDS
  - For storing user data
  - Read replicas for scaling reads
  - Multi AZ for disaster recovery
- Tight Security with security groups referncing each other

Stateful Web App: MyWordPress.com

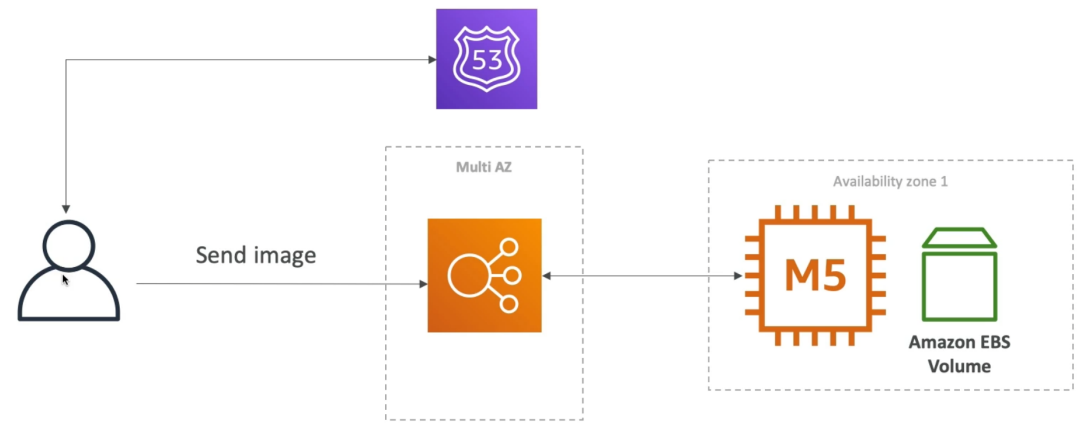
- We are trying to create a fully scalable WordPress website
- We want that website to access and correctly display picture uploads
- Our user data, and the blog content should be stored in a MySQL database

Let's see how we can achieve this!

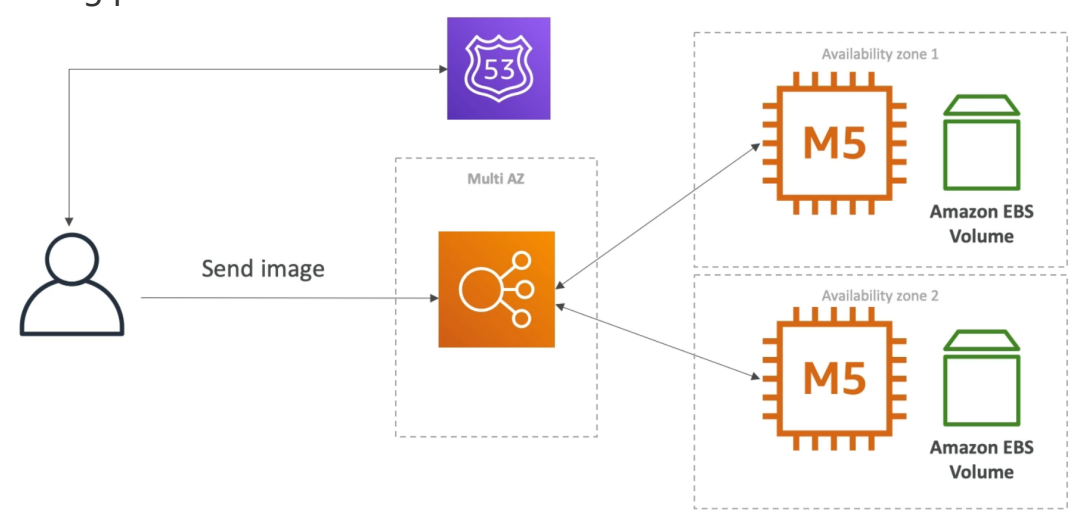
- RDS layer



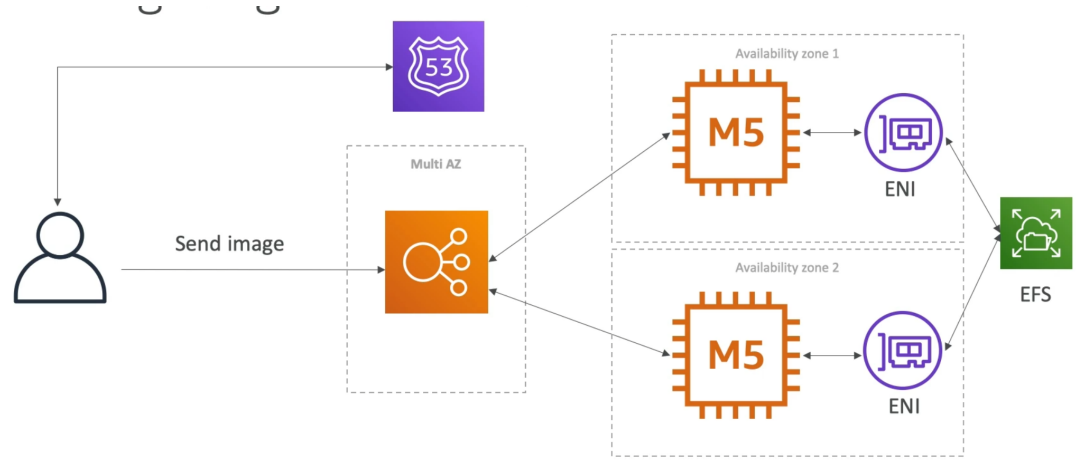
- Storing images with EBS



having problem



- Storing images with EFS



In this lecture we've discussed ...

- Aurora Database to have easy Multi-AZ and Read-Replicas
- Storing data in EBS (single instance application)
- Vs Storing data in EFS (distributed application)

Instantiating Applications quickly

- When launching a full stack (EC2,EBS,RDS), it can take time to :
  - Install applications
  - Insert initial (or recovery) data
  - Configure everything

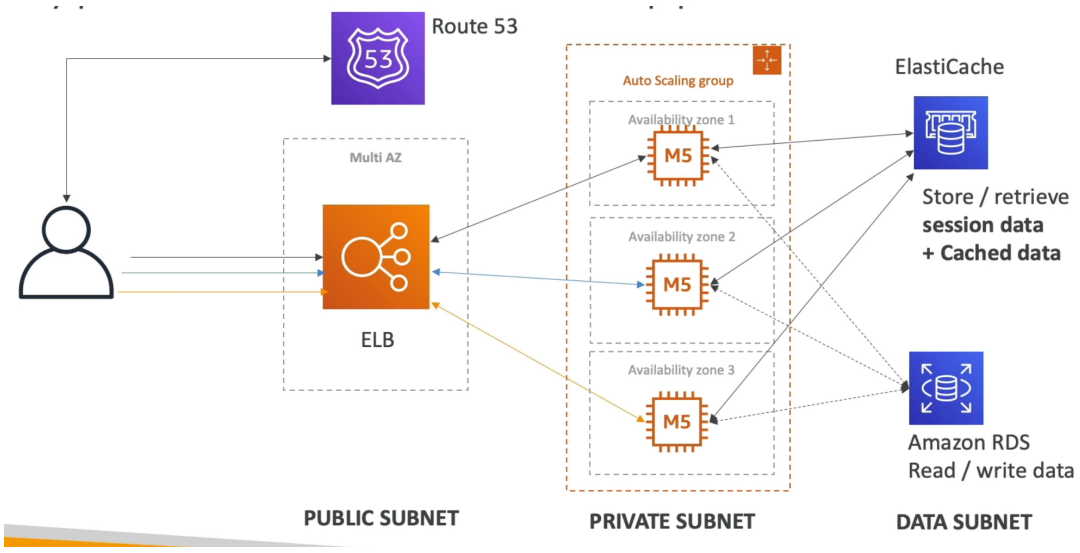
    - Launch the application

We can take advantage of the cloud to speed that up!

- EC2 Instances:
    - **Use a Golden AMI** : Install your applications, OS dependencies etc ... beforehand and launch your EC2 instance from the Golden AMI
    - **Bootstrap using User Data:** For dynamic configuration, use User Data scripts
    - **Hybrid**: mix Golden AMI and User Data (Elastic Beanstalk)
- RDS Databases:
    - Restore from a snapshot: the database will have schemas and data ready!
- EBS Volumes:
    - Restore from a snapshot: the disk will already be formatted and have data!

Elastic Beanstalk

Typical architecture: Web App 3-tier



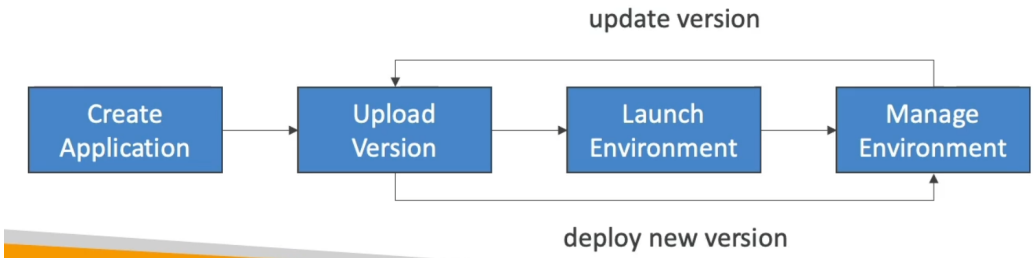
- Managing infrastructure
- Deploying Code
- Configuring all the databases, load balancers,etc
- Scaling concerns
- Most web apps have the same architecture(ALB + ASG)
- All the developers want is for their code to run!
- Possibly, consistently across different applications and environments

Overview

- Elastic Beanstalk is a developer centric view of deploying an application on AWS
- It uses all the component's we've seen before: EC2, ASG,ELB,RDS, ...
- Managed service
    - Automatically handles capacity provisioning, load balancing, scaling, application health monitoring, instance configuration, ...
    - Just the application code is the responsibility of the developer
- We still have full control over the configuration
- Beanstalk is free but you pay for the underlying instances

Components

- **Application:** collection of Elastic Beanstalk components (environments, versions, configurations,...)
- **Application Version**: an iteration of your application code
- **Environment**
    - Collection of AWS resources running an application version (only one application version at a time)
    - **Tiers**: Web Server Environment Tier & Worker Environment Tier
    - You can create multiple environments (dev, test, prod)

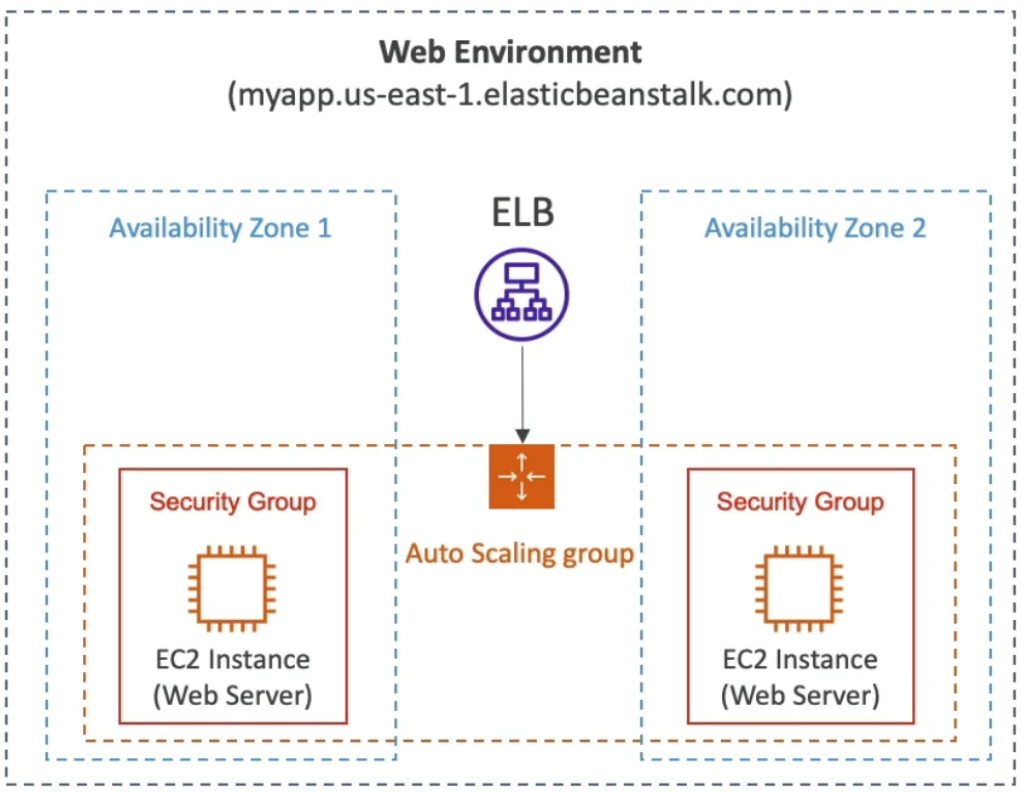


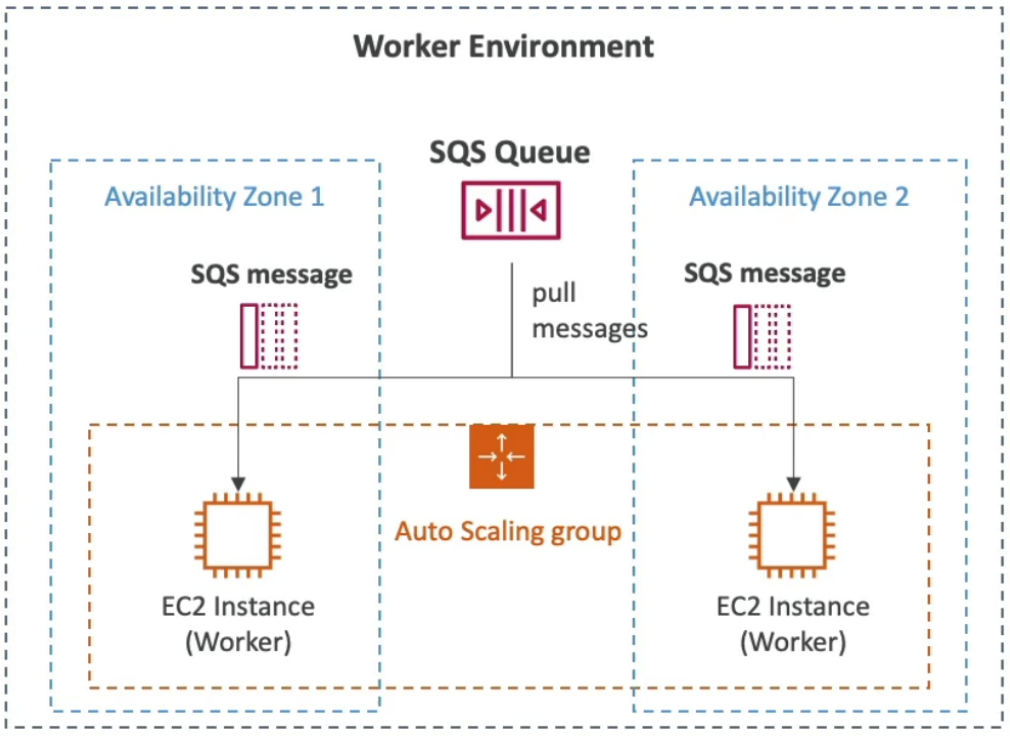
Supported Platforms

- Go
- Java SE
- Java with Tomcat
- .NET Core on Linux
- .NET on Windows Server
- Node.js
- PHP
- Python
- Ruby
- Packer Builder
- Single Container Docker
- Multi-container Docker
- Preconfigured Docker

- If not supported,you can write your custom platform(advanced)

Web Server Tier vs Worker Tier

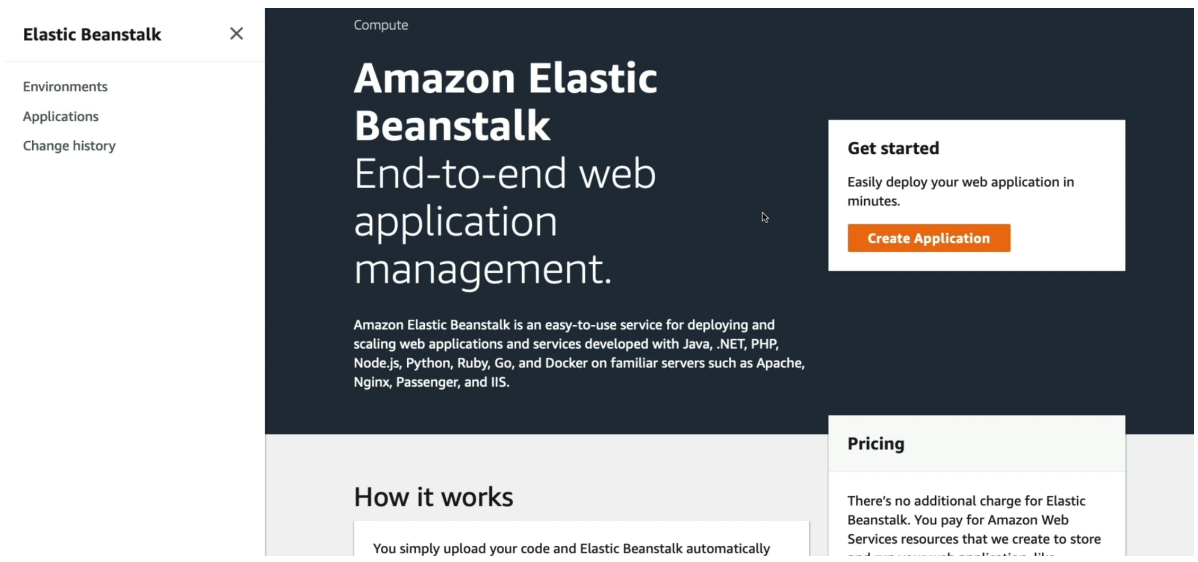


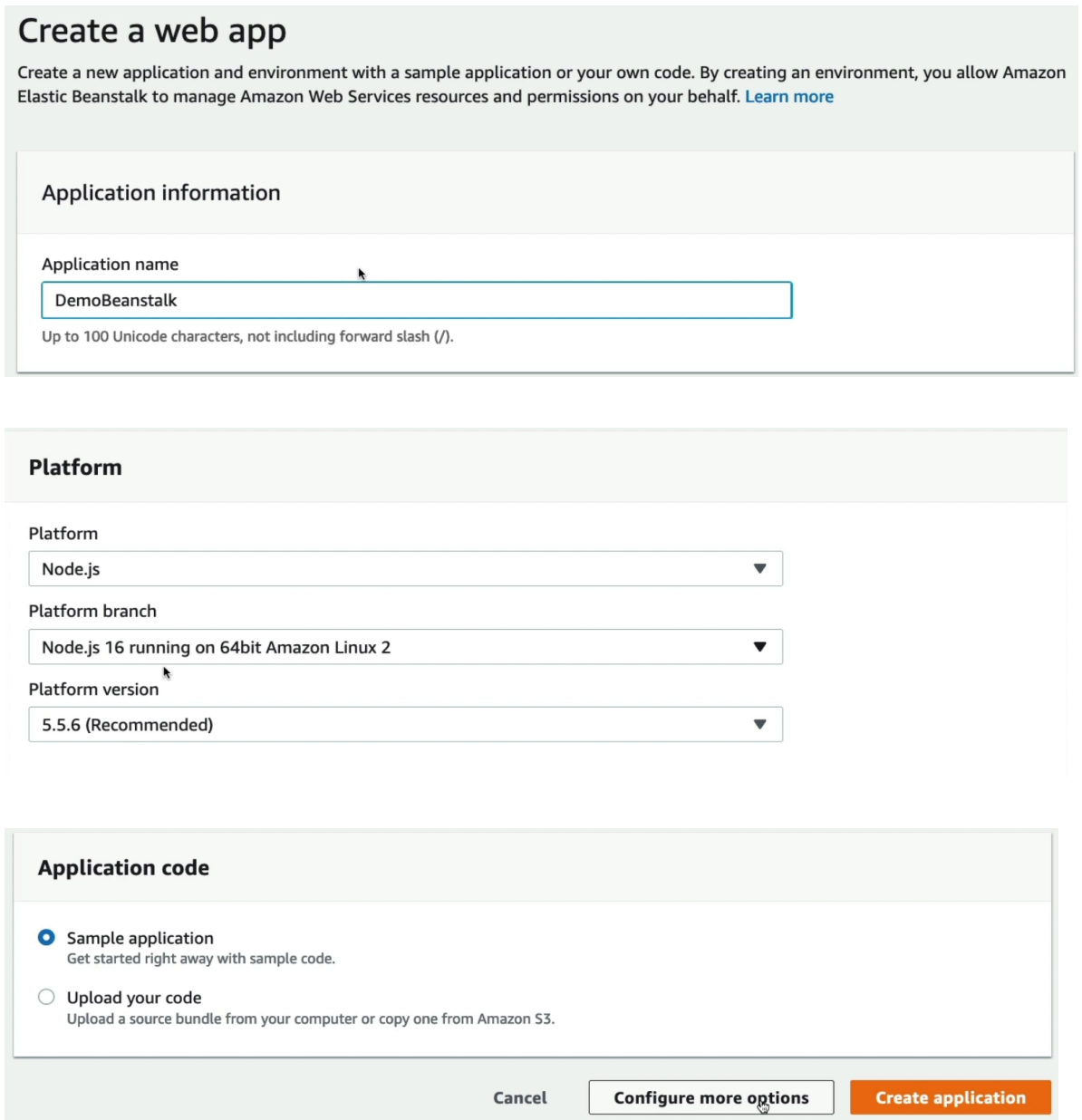


Scale based on the number of SQS messages

Can push messages to SQS queue from another Web Server Tier

Hands on





Configure Demobeanstalk-env

Presets

Start from a preset that matches your use case or choose *Custom configuration* to unset recommended values and use the service's default values.

Configuration presets

- (•) Single instance *(Free Tier eligible)*
- ( ) Single instance (using Spot instance)
- ( ) High availability
- ( ) High availability (using Spot and On-Demand instances)
- ( ) Custom configuration

Platform

Node.js 16 running on 64bit Amazon Linux 2/5.5.6

Change platform version

Software

Edit

Amazon X-Ray:
disabled

Log streaming:
disabled (default)

Environment properties:
0

Rotate logs:
disabled (default)

Instances

Edit

IMDSv1:
disabled

Root volume size (GB):
container default

Root volume throughput (MiB/s):
container default

Root volume type:
container default

Root volume IOPS:
container default

Security groups:
none

Capacity

Edit

Environment type:
single instance

EC2 instance types:
t2.micro,t2.small

EC2 image ID:
ami-0e14fd88b2777054a

Fleet composition:
On-Demand instance

Capacity rebalancing:
disabled

Load balancer

This configuration does not contain a load balancer.

Rolling updates and deployments

Edit

Deployment policy:
All at once

Rolling updates:
disabled

Security

Edit

Service role:
arn:aws:iam::783768293452:role/aws-elasticbeanstalk-service-role

Virtual machine key pair:
--

Virtual machine instance profile:
aws-elasticbeanstalk-ec2-role

Monitoring

Edit

Health reporting system:
Enhanced

Health event log streaming:
disabled

Managed Updates

Edit

Managed updates:
enabled

Weekly update window:
Wed:17:00 UTC

Notifications

Edit

Email address:
--

Network

Edit

This environment is not part of a VPC.

Database

Edit

Engine:
--

Instance class:
--

Storage (GB):
--

Multi-AZ:
--

Tags

Edit

Tags:
none

Cancel | Previous | Create app

Elastic Beanstalk

- Environments
- Applications
- Change history
- DemoBeanstlak
  - Application versions
  - Saved configurations
- Demobeanstlak-env

Elastic Beanstalk > Environments > Demobeanstlak-env

Creating Demobeanstlak-env
This will take a few minutes. .

3:38pm Waiting for EC2 instances to launch. This may take a few minutes.
3:37pm Created EIP: 54.72.161.22
3:37pm Created security group named: awseb-e-5vamafjnjc-stack-AWSEBSecurityGroup-4R1DTWFPTP3H
3:37pm Environment health has transitioned to Pending. Initialization in progress (running for 6 seconds). There are no instances.
3:36pm Using elasticbeanstalk-eu-west-1-783768293452 as Amazon S3 storage bucket for environment data.
3:36pm createEnvironment is starting.

CloudFormation

- Stacks
  - Stack details
  - Drifts
- StackSets
- Exports
- Designer
- Registry
  - Public extensions
  - Activated extensions
  - Publisher
- Feedback

CloudFormation > Stacks > awseb-e-us4qybjafk-stack

Stacks (5)

Filter by stack name

Active | View nested | 1

awseb-e-us4qybjafk-stack
2022-09-11 15:31:22 UTC+0100
CREATE_IN_PROGRESS

StackSet-AWS-QuickSetup-SSMHostMgmt-LA-qeizg-adf11399-ad84-4848-9458-dc3f622d9519
2022-04-08 11:32:07 UTC+0100
CREATE_COMPLETE

ECS-Console-V2-Service-9b8c01b8-360a-4e50-a40d-972504d0b397
2022-04-08 00:17:04 UTC+0100
CREATE_COMPLETE

awseb-e-us4qybjafk-stack

Delete | Update | Stack actions | Create stack

Stack info | Events | Resources | Outputs | Parameters | Templ

Overview

Stack ID
arn:aws:cloudformation:eu-west-1:783768293452:stack/awseb-e-us4qybjafk-stack/68b3fbc0-31de-11ed-b3a5-0a777bb749e7

Description
AWS Elastic Beanstalk environment (Name: 'Demobeanstalk-env' Id: 'e-us4qybjafk')

Status
CREATE_IN_PROGRESS

Status reason
-

Root stack
-

Parent stack
-

Created time
2022-09-11 15:31:22 UTC+0100

Deleted time
-

Elastic Beanstalk

- Environments
- Applications
- Change history
- DemoBeanstlak
  - Application versions
  - Saved configurations
- Demobeanstlak-env
  - Go to environment
  - Configuration
  - Logs
  - Health
  - Monitoring
  - Alarms
  - Managed updates

Elastic Beanstalk > Environments > Demobeanstlak-env

Demobeanstlak-env
Demobeanstlak-env.eba-7rjaw3ii.eu-west-1.elasticbeanstalk.com (e-5vamafjnjc)
Application name: DemoBeanstlak

Refresh | Actions

Health
Ok
Causes

Running version
Sample Application
Upload and deploy

Platform
Node.js 16 running on 64bit Amazon Linux 2/5.5.6
Change

Recent events

Show all | 1

Time | Type | Details