To the Cloud!

A Personal Journey

Server By Default

- I installed Linux on my laptop and set up Apache, MySQL, PHP there, so when I had access to my first server, that's what I did.

- While Hand-Crafted Artisanal Servers (™) are great for learning, they don't scale, and they soon annoy.

- Recipes came on the scene and made things better. Now I could stamp out my Artisanal httpd.conf files quickly but they were still dependent on the server environment and didn't match my dev environment, and it was all still a little annoying.

Let There Be Containers!

- What the hell is a container?
  - Don’t swear.
  - Containers have been defined in a few ways — they’re really not complicated — but let me just say for this presentation, they’re a way to save the state of a server and spin it back up super quick, anywhere, sort of like cloning a VM only way, way, way lighter. They’re also nothing like VMs but I don’t have time for that.

Containers are like `git commit` for server setups.

Isn’t this presentation about the cloud?

What about the cloud?

- First, you don't *need* the cloud. You could run the Docker daemon on your Hand-Crafted Artisanal (™) server and life would still be a little better for you.

- Clouds give me a few things:

  - Cheap, quick backups that I don't have to think about.

  - Monitoring and alerts that I don't have to think about.

  - The pointy-hairs know exactly what it costs to offer the service.

  - And most importantly of all, the platform vanishes entirely. I never have to run `apt-get` or `dnf` again.

I NEVER HAVE TO RUN `apt-get` AGAIN!!>!

But how?

This will be AWS specific but the concepts are useful.

We’re gonna need a few things …

Load Balancer

Image repository

Target group

Task definition

(Task definitions are immutable for some reason.)

Security group

CNAME records

Environment variable-based configuration

(Environment variable-based configuration may require some work)

Something called Fargate

(Whatever ‘elastic’ is referring to.)

Clusters

Services

CloudWatch

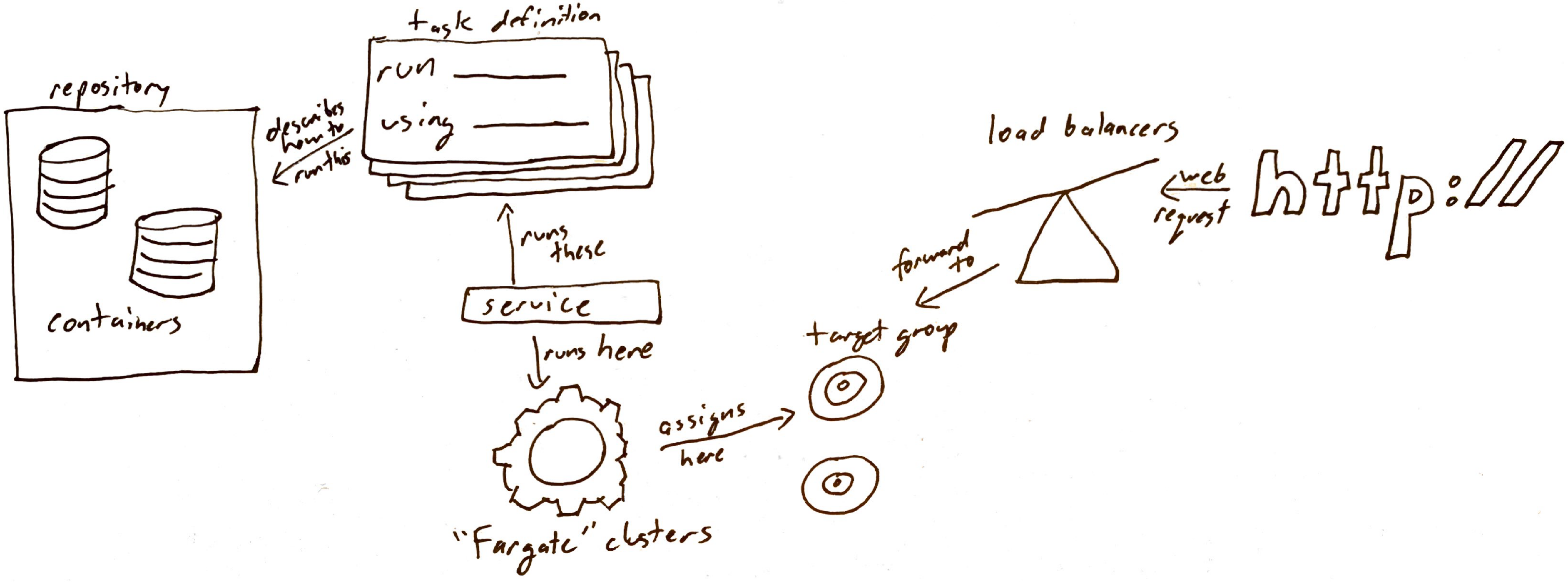
SNS

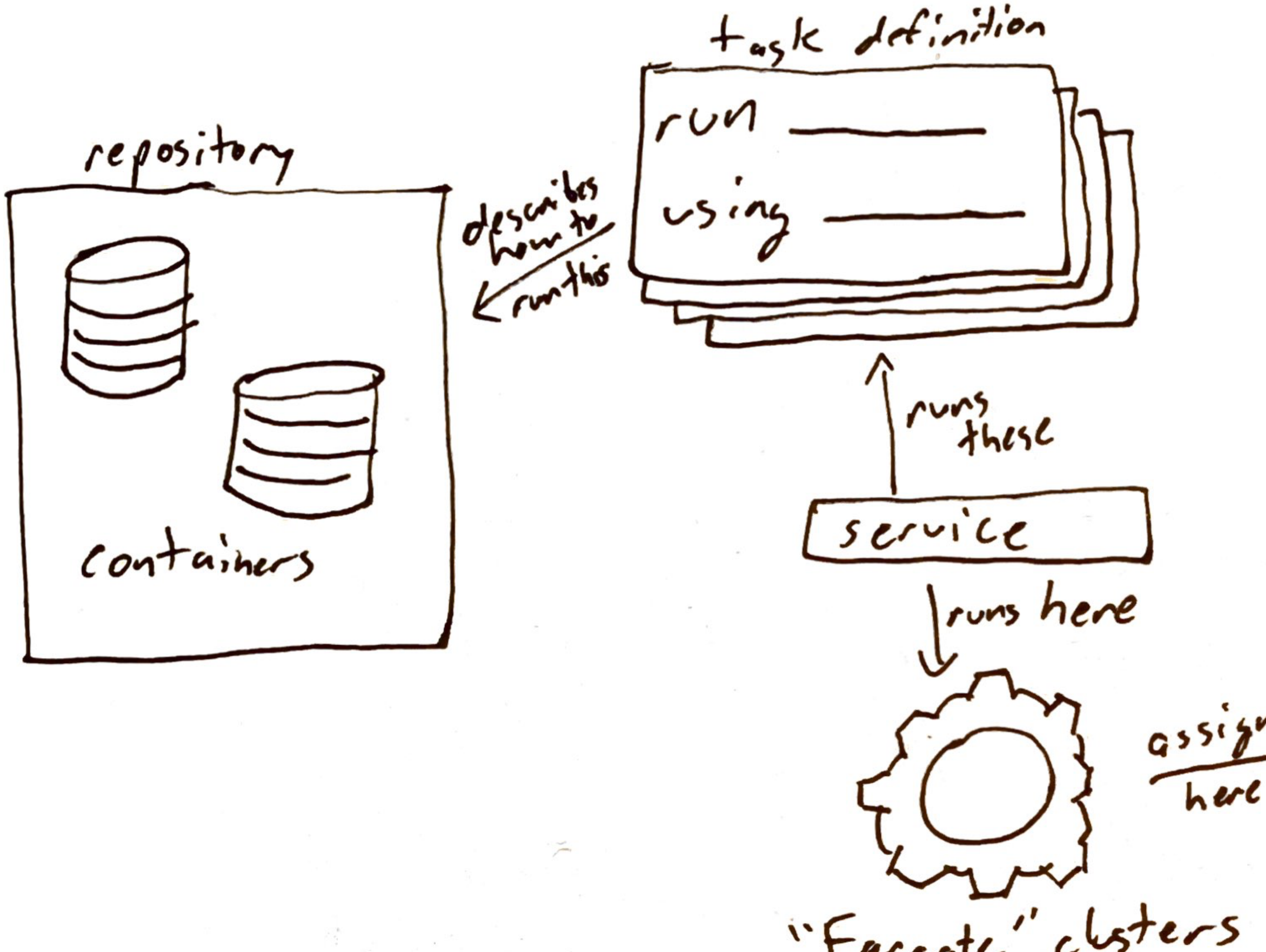
Health checks

(Health checks ironically take down my services on occasion.)

All delivered by an inconsistently-designed web interface that works okay.

Here’s how you container in AWS.


repository
containers
describes how to run this
task definition
run
using
runs these
service
runs here
"Fargate" clusters
assigns here
target group
forward to
load balancers
web request
http://


repository
containers
describes how to run this
task definition
run
using
runs these
service
runs here
assig
here
"Fargate" clusters

Repositories

- Amazon ECR is a managed AWS Docker registry service. Customers can use the familiar Docker CLI to push, pull, and manage images.

Repositories

```
docker push [OPTIONS] NAME[:TAG]
```

- Amazon ECR is a managed AWS Docker registry service. Customers can use the familiar Docker CLI to push, pull, and manage images.

Task Definitions

- **A task definition is required to run Docker containers in Amazon ECS.** Some of the parameters you can specify in a task definition include:
  - The Docker image to use with each container in your task
  - How much CPU and memory to use with each task or each container within a task
  - The launch type to use, which determines the infrastructure on which your tasks are hosted
  - The Docker networking mode to use for the containers in your task
  - The logging configuration to use for your tasks
  - Whether the task should continue to run if the container finishes or fails
  - The command the container should run when it is started
  - Any data volumes that should be used with the containers in the task
  - The IAM role that your tasks should use

Task Definitions

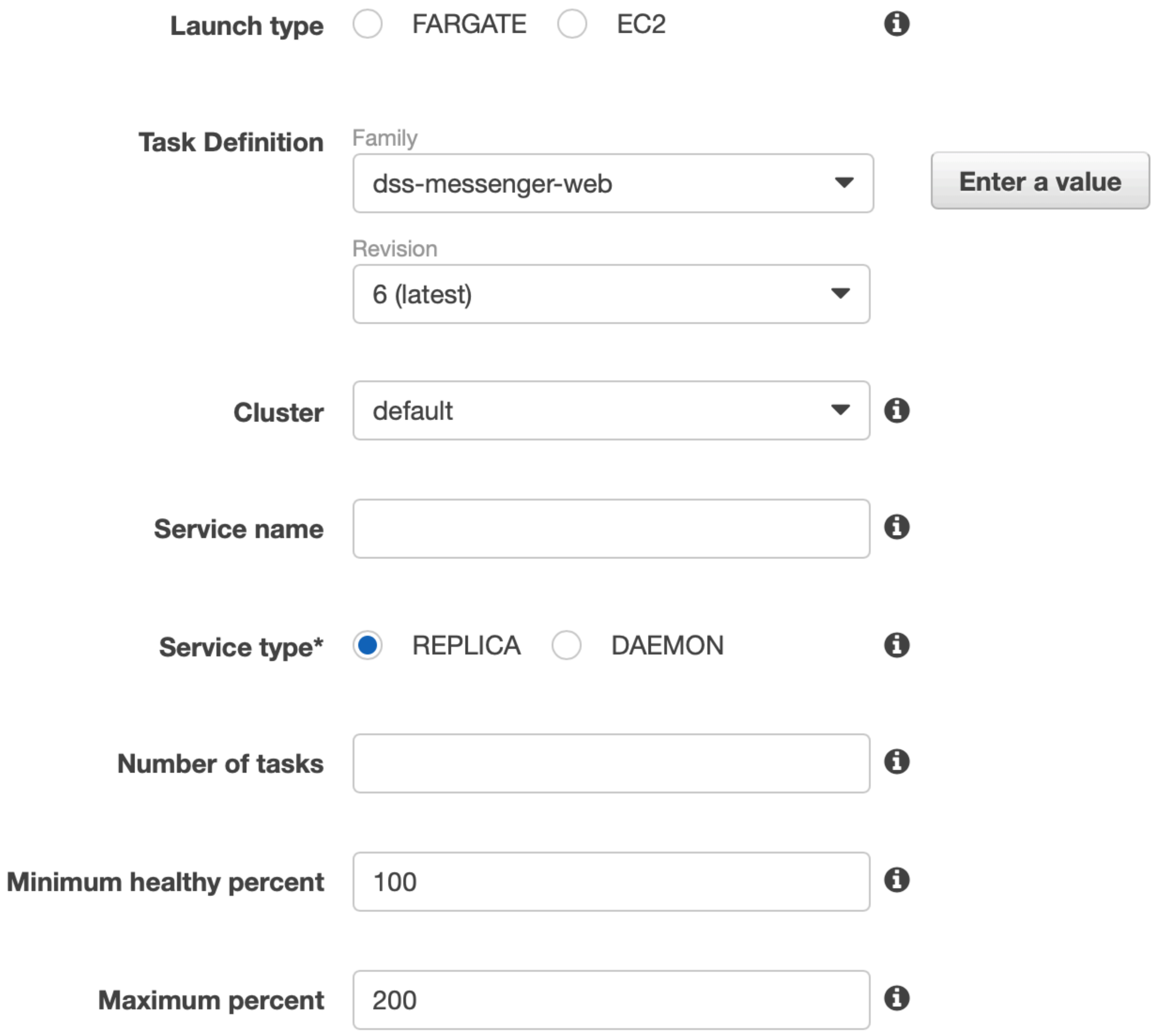
- **A task definition is required to run Docker containers in Amazon ECS.** Some of the parameters you can specify in a task definition include:
    - The Docker image to use with each container in your task
    - How much CPU and memory to use with each task or each container within a task
    - The launch type to use, which determines the infrastructure on which your tasks are hosted
    - The Docker networking mode to use for the containers in your task
    - The logging configuration to use for your tasks
    - Whether the task should continue to run if the container finishes or fails
    - The command the container should run when it is started
    - Any data volumes that should be used with the containers in the task
    - The IAM role that your tasks should use

```
docker run [OPTIONS] IMAGE[:TAG|@DIGEST] [COMMAND] [ARG...]
```

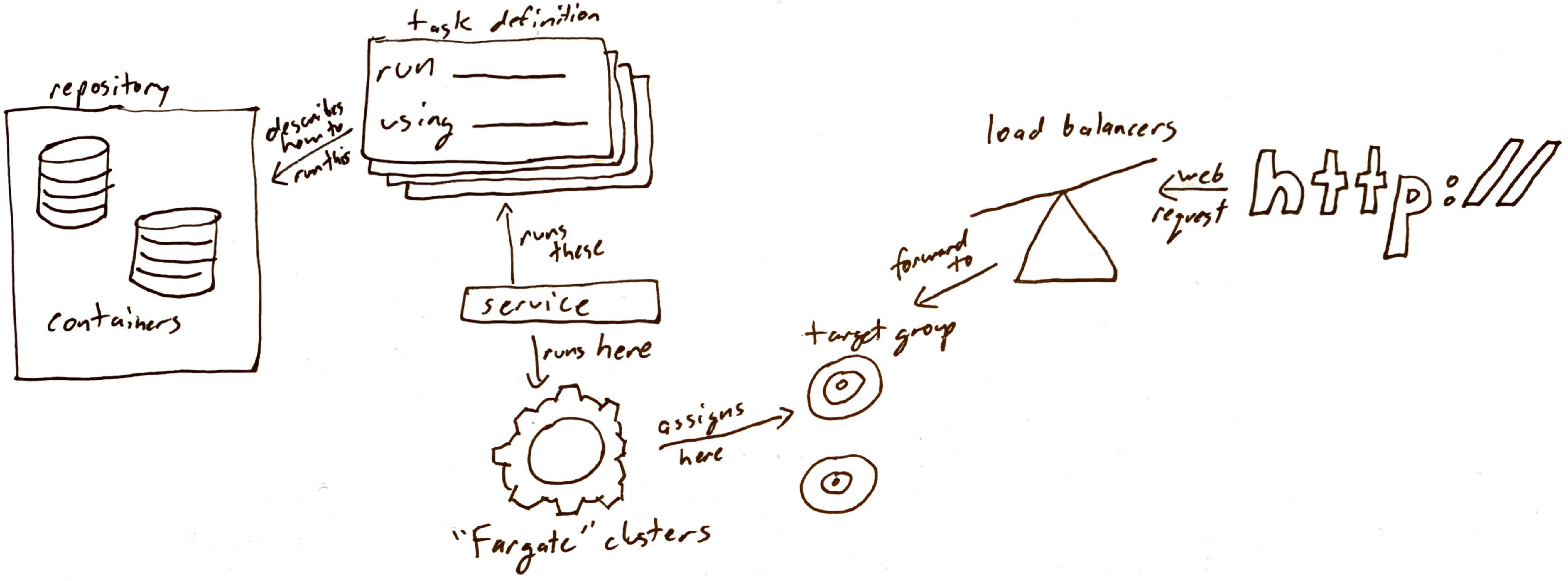
Services

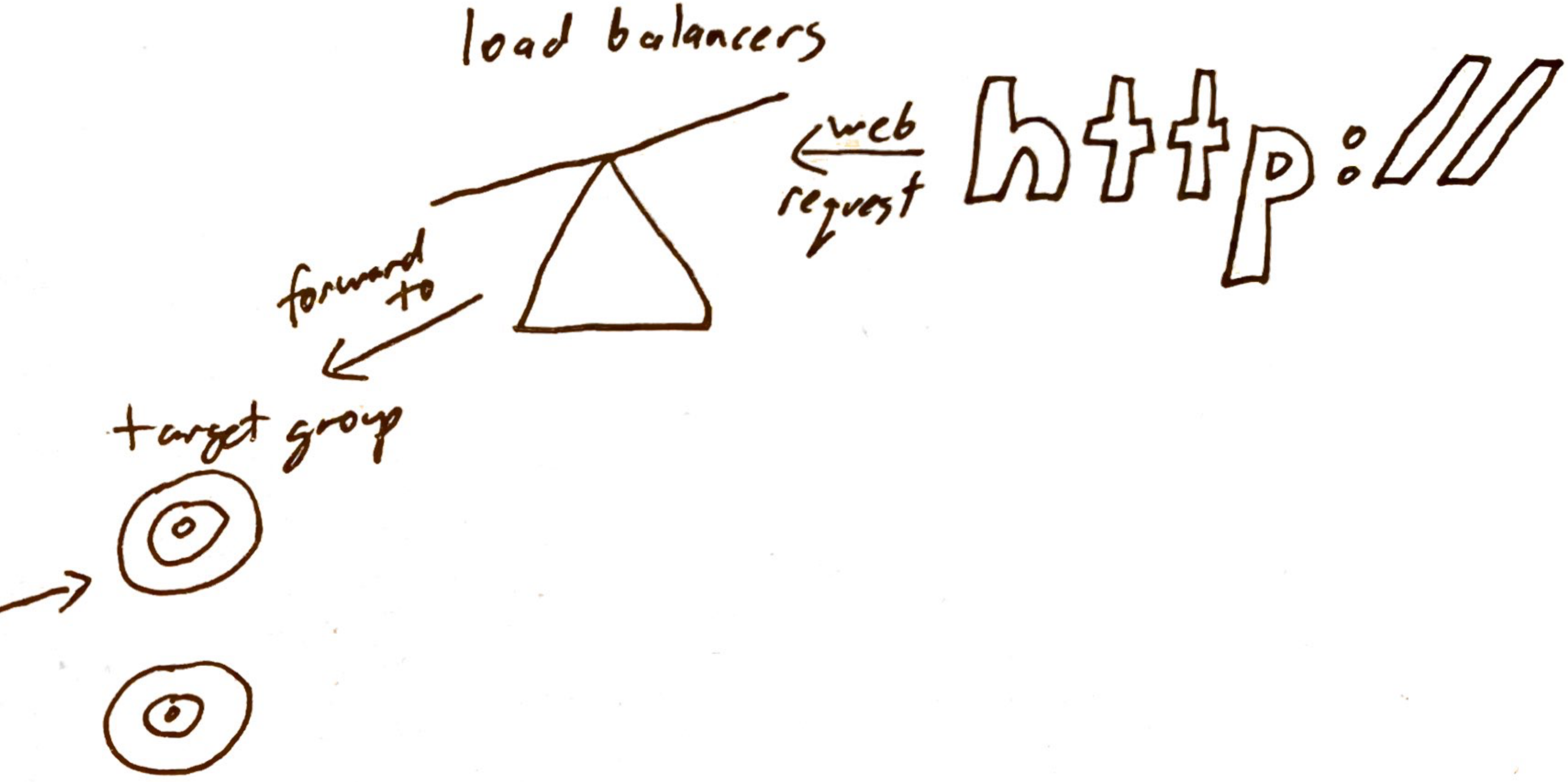
- Amazon ECS allows you to run and maintain a specified number of instances of a task definition simultaneously in an Amazon ECS cluster. This is called a service.

Services



Launch type	○ FARGATE ○ EC2
Task Definition	Family: dss-messenger-web; Revision: 6 (latest)
Cluster	default
Service name	
Service type*	[✓ REPLICA] ○ DAEMON
Number of tasks	
Minimum healthy percent	100
Maximum percent	200

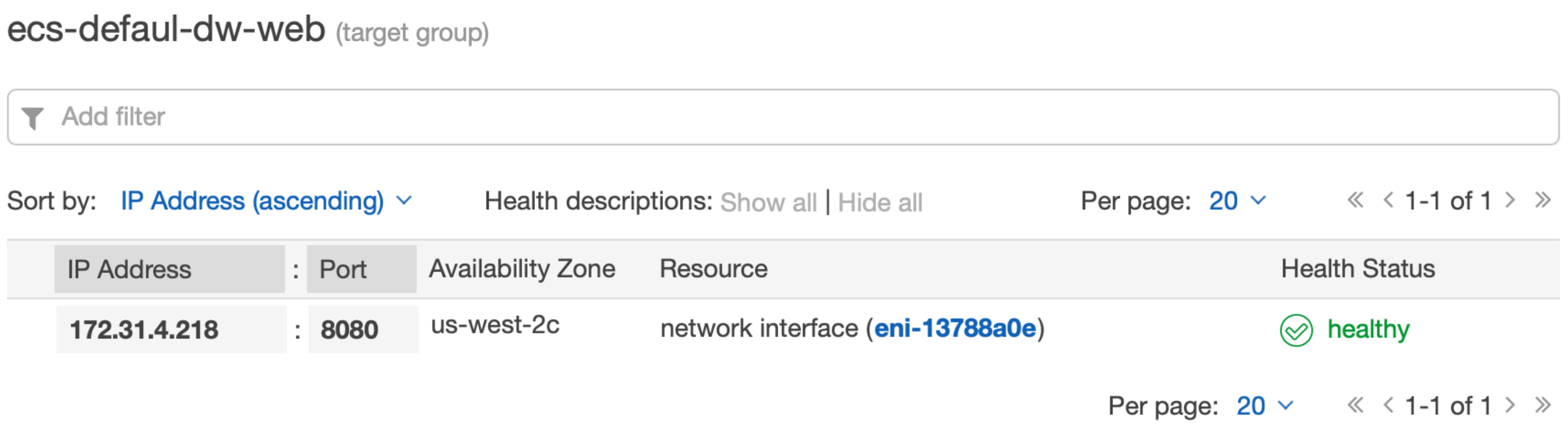

repository
containers
describes how to run this
task definition
run
using
runs these
service
runs here
"Fargate" clusters
assigns here
target group
forward to
load balancers
web request
http://


load balancers
web request
http://
forward to
target group

Target Groups

- Each target group is **used to route requests to one or more registered targets**. When you create each listener rule, you specify a target group and conditions. When a rule condition is met, traffic is forwarded to the corresponding target group

Target Groups



ecs-defaul-dw-web (target group)

Sort by: IP Address (ascending) | Health descriptions: Show all | Hide all | Per page: 20 | 1-1 of 1

IP Address	:	Port	Availability Zone	Resource	Health Status
172.31.4.218	:	8080	us-west-2c	network interface (eni-13788a0e)	✓ healthy

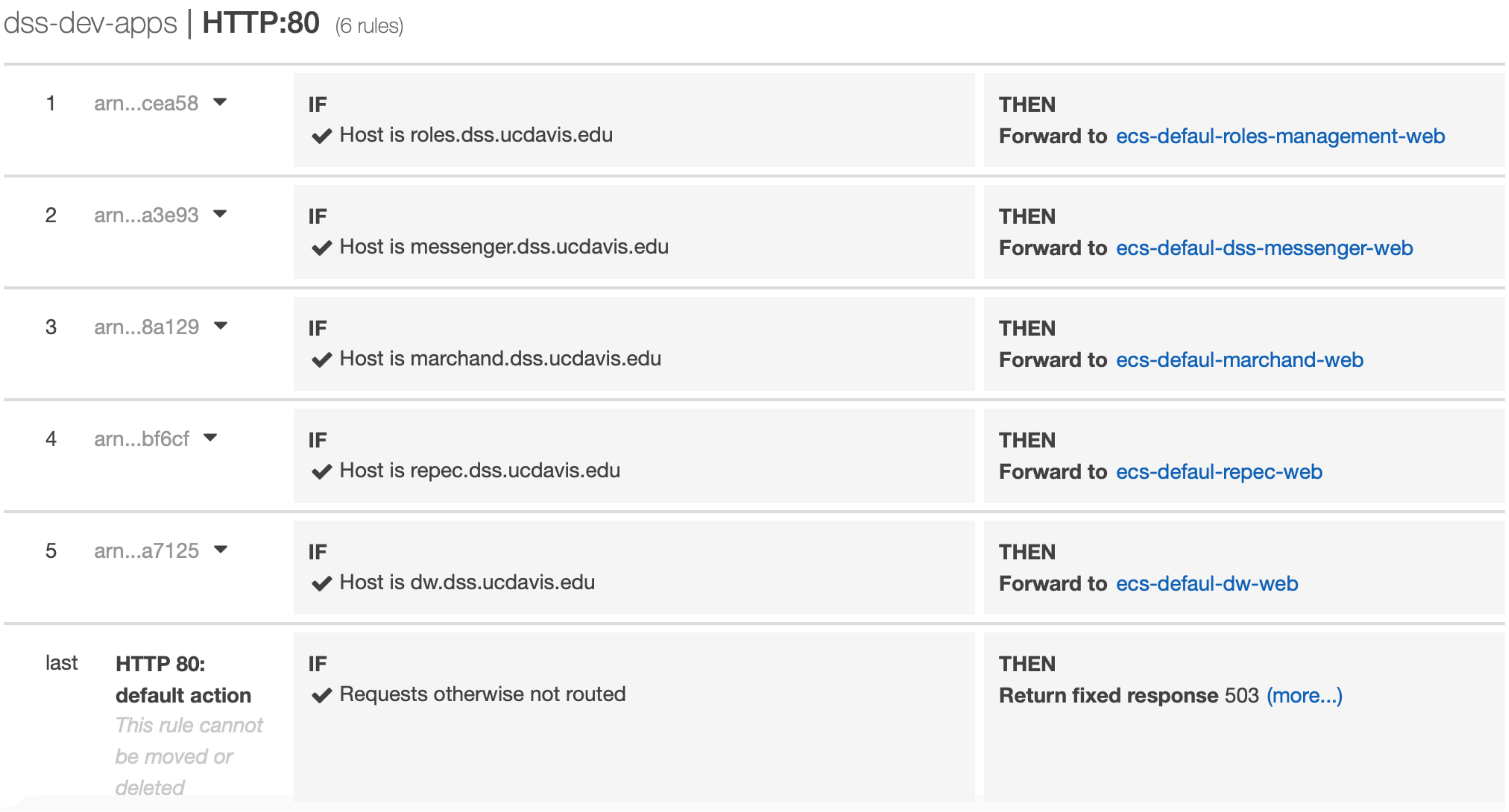
Per page: 20 | 1-1 of 1

Mostly automatic when using the container service.

Load Balancers

- Elastic Load Balancing automatically **distributes incoming application traffic across multiple targets**, such as Amazon EC2 instances, containers, IP addresses, and Lambda functions.

Load Balancers



dss-dev-apps | **HTTP:80** (6 rules)

1	arn...cea58	**IF** ✓ Host is roles.dss.ucdavis.edu	**THEN** **Forward to** ecs-defaul-roles-management-web
2	arn...a3e93	**IF** ✓ Host is messenger.dss.ucdavis.edu	**THEN** **Forward to** ecs-defaul-dss-messenger-web
3	arn...8a129	**IF** ✓ Host is marchand.dss.ucdavis.edu	**THEN** **Forward to** ecs-defaul-marchand-web
4	arn...bf6cf	**IF** ✓ Host is repec.dss.ucdavis.edu	**THEN** **Forward to** ecs-defaul-repec-web
5	arn...a7125	**IF** ✓ Host is dw.dss.ucdavis.edu	**THEN** **Forward to** ecs-defaul-dw-web
last	**HTTP 80: default action** *This rule cannot be moved or deleted*	**IF** ✓ Requests otherwise not routed	**THEN** **Return fixed response** 503 (more...)

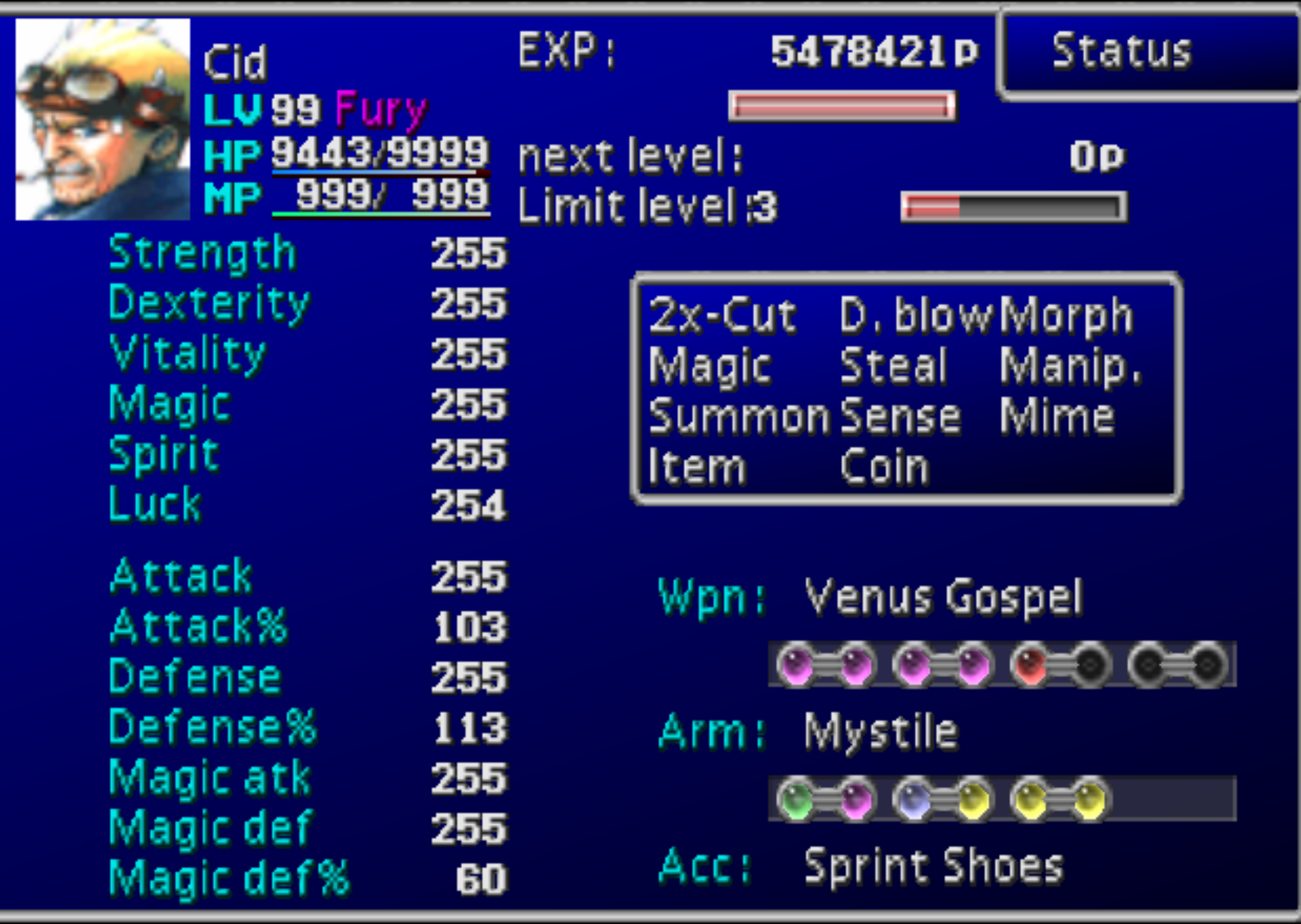
Do I need a load balancer? I don’t get 1000 hits / s.

Sort of. You have options.

Handling Public Addresses

- Load balancer (CNAME record + configure traffic router)
- Elastic IP (very limited static IPs, presumably usable in ECS but double-check me)
- Use the public IP already given to the running task (very dangerous, can change when task shuts down)
- Don’t worry about it (background tasks don’t need public addresses but benefit from containerization)

WISDOM



Cid
LV 99 Fury
HP 9443/9999
MP 999/ 999

EXP: 5478421p
next level: 0p
Limit level :3

Status

Strength	255
Dexterity	255
Vitality	255
Magic	255
Spirit	255
Luck	254
Attack	255
Attack%	103
Defense	255
Defense%	113
Magic atk	255
Magic def	255
Magic def%	60

2x-Cut	D. blow	Morph
Magic	Steal	Manip.
Summon	Sense	Mime
Item	Coin	

Wpn: Venus Gospel
Arm: Mystile
Acc: Sprint Shoes

Wisdom

- Make sure your services are all running **in the proper security group**. This was my biggest headache and it does not reveal itself easily.
- If your tasks appear to run but then shut down minutes later, make sure the **health check isn't killing them**, e.g. checking on port 80 <u>should</u> return a HTTP 302 but the health check is configured to only accept HTTP 200.
- You can use **one application load balancer for many sites**. This will save you $$$. I don't know how I missed that.
- Round-trip **latency between AWS and campus is enough to kill performance** if you need many requests / s, e.g. your application is running on campus but your RDB is on AWS. Just move it all to the same place. Bite the bullet.
- Campus services (Banner!) are correctly firewalled. Take this into consideration. **Moving to AWS means switching subnets**. This may imply other changes.

What about CI? Or automated deploys using Terraform, etc.?

Ask me in a few months. I dunno.

CLI Workflow

```
$(aws ecr get-login --no-include-email --region us-west-2)

docker build -t the-image .

docker tag the-image:latest a.url.amazonaws.com/repo-name:latest

docker push a.url.amazonaws.com/repo-name:latest

aws ecs update-service -service service-name -force-new-deployment
```

Live demo?