



# AWS Fargate: From Scratch To Production

Marcelo Pinheiro  
@salizzar



# \$ whoami

- Firefighter / Problem Solver / Programmer since 2000
- Ruby, Python, Golang, Java, C#, Classic ASP, PHP, Node.js, Erlang and others
- Not wrote code only... I already fought, made coffee, negotiated deadlines, was a therapist, drank a few beers
- Señor DevOps Engineer @ Work & Co



**WORK  
&CO**

We're hiring.

<https://work.co/careers>

Look for me :-)



# AWS Fargate

## A Brief Introduction

Oh really?

# AWS Fargate - A Brief Introduction

- Fargate is a engine provided by Amazon Web Services to run containers without need to manage servers or clusters
- You can scale up / down containers as you need
- You pay for the resources used (vCPU + RAM)
- Requires a Docker image



# AWS Fargate - A Brief Introduction

- AWS Fargate vs AWS EKS:
  - EKS manages master nodes but you need to provision worker nodes
- AWS Fargate vs AWS ECS:
  - You need to provision at least one server and manage it
- AWS Fargate vs Azure ACI:
  - Far way more complex to setup
  - On the other hand, much more complete
- AWS Fargate vs GCE:
  - Google has no equivalent solution



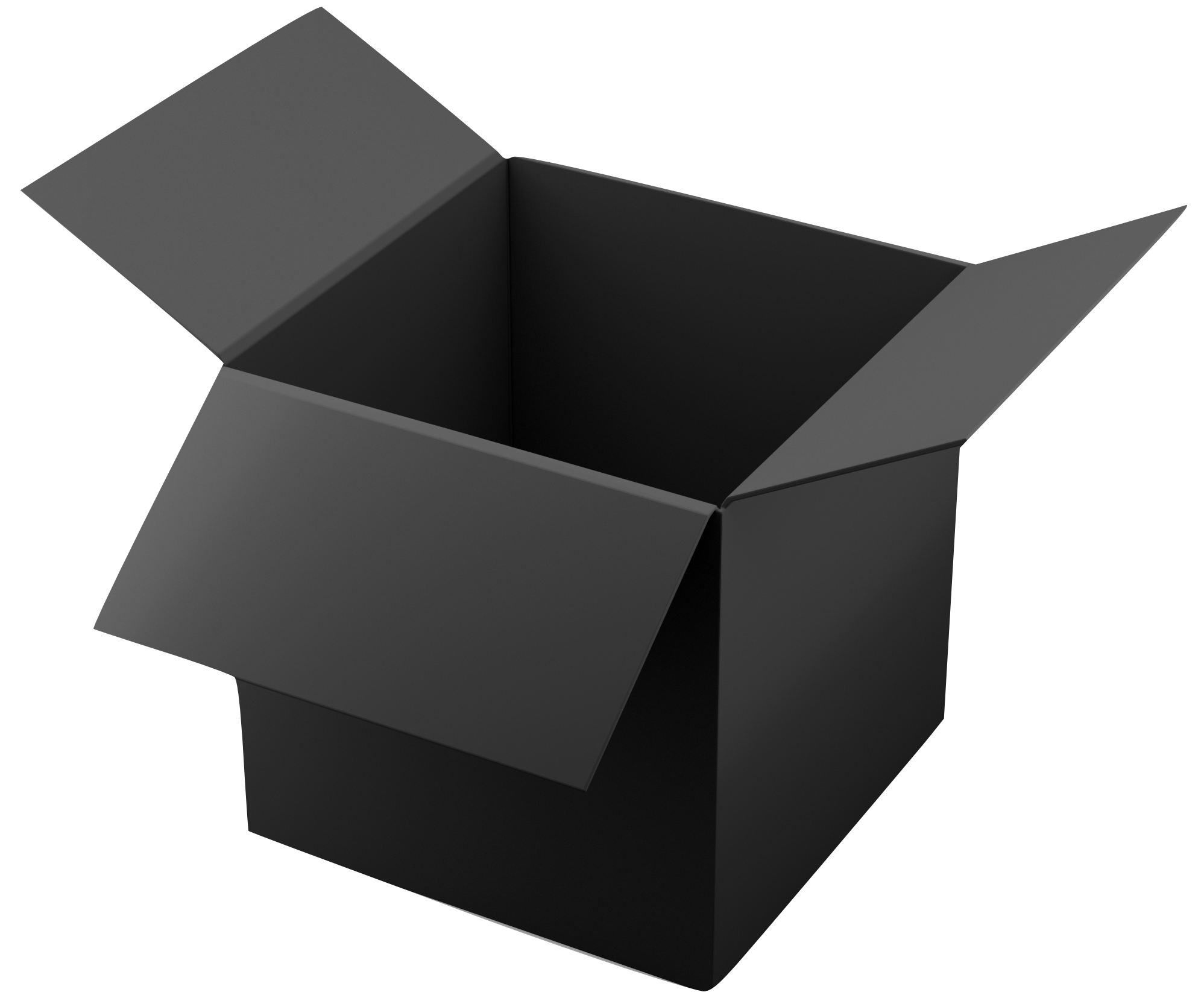


# AWS Fargate - Architecting Your Cluster

These things are not so simple as you expect.

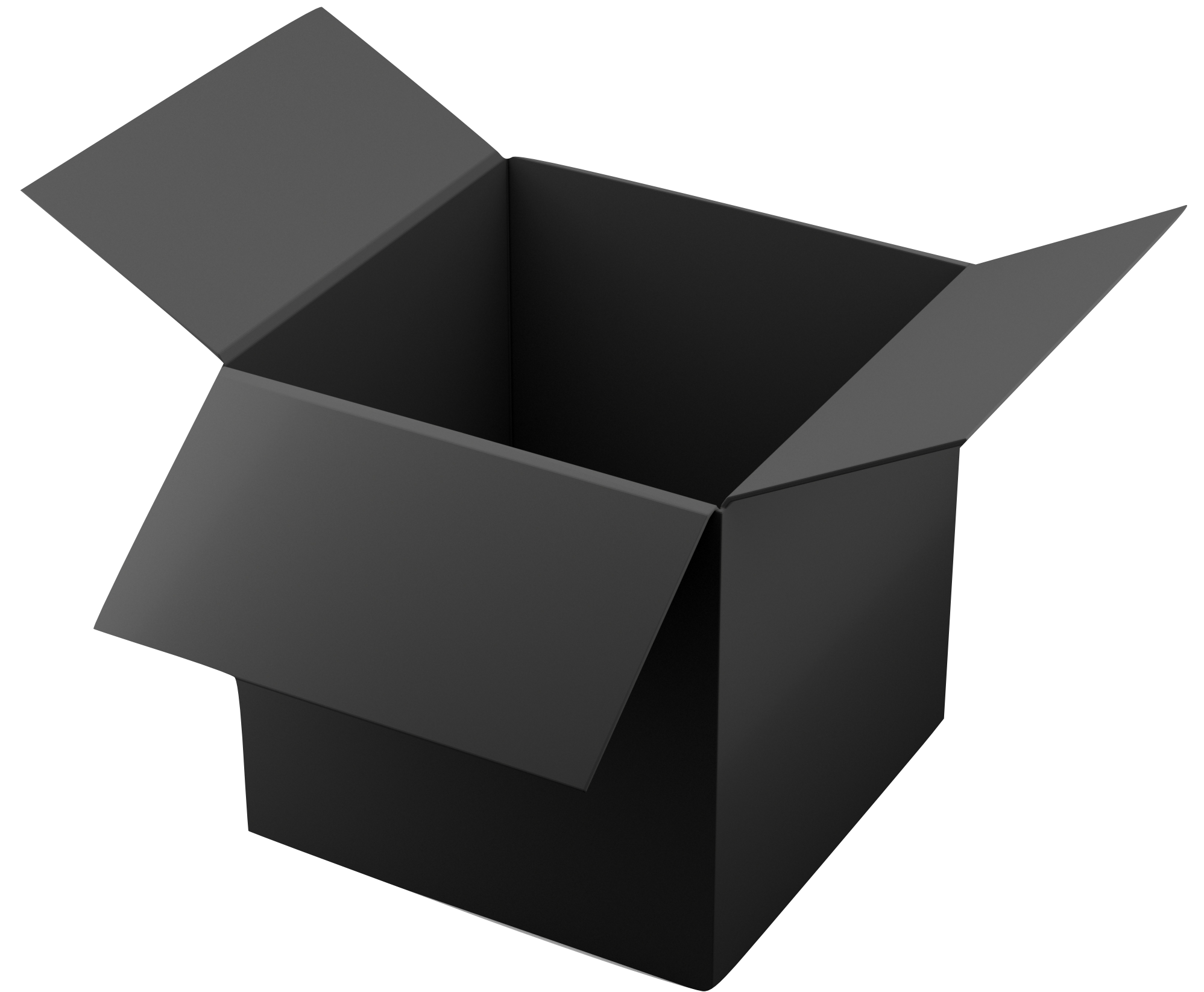
# AWS Fargate - Architecting Your Cluster

- You need to have some experience with the following AWS components to create a Fargate cluster ready for production:
  - ECS Services / Task Definitions
  - VPC
  - ALB
  - Cloudwatch Alerts / Logs / Events
  - IAM
  - SSM Parameter Store
  - KMS
  - Route53
  - ACM



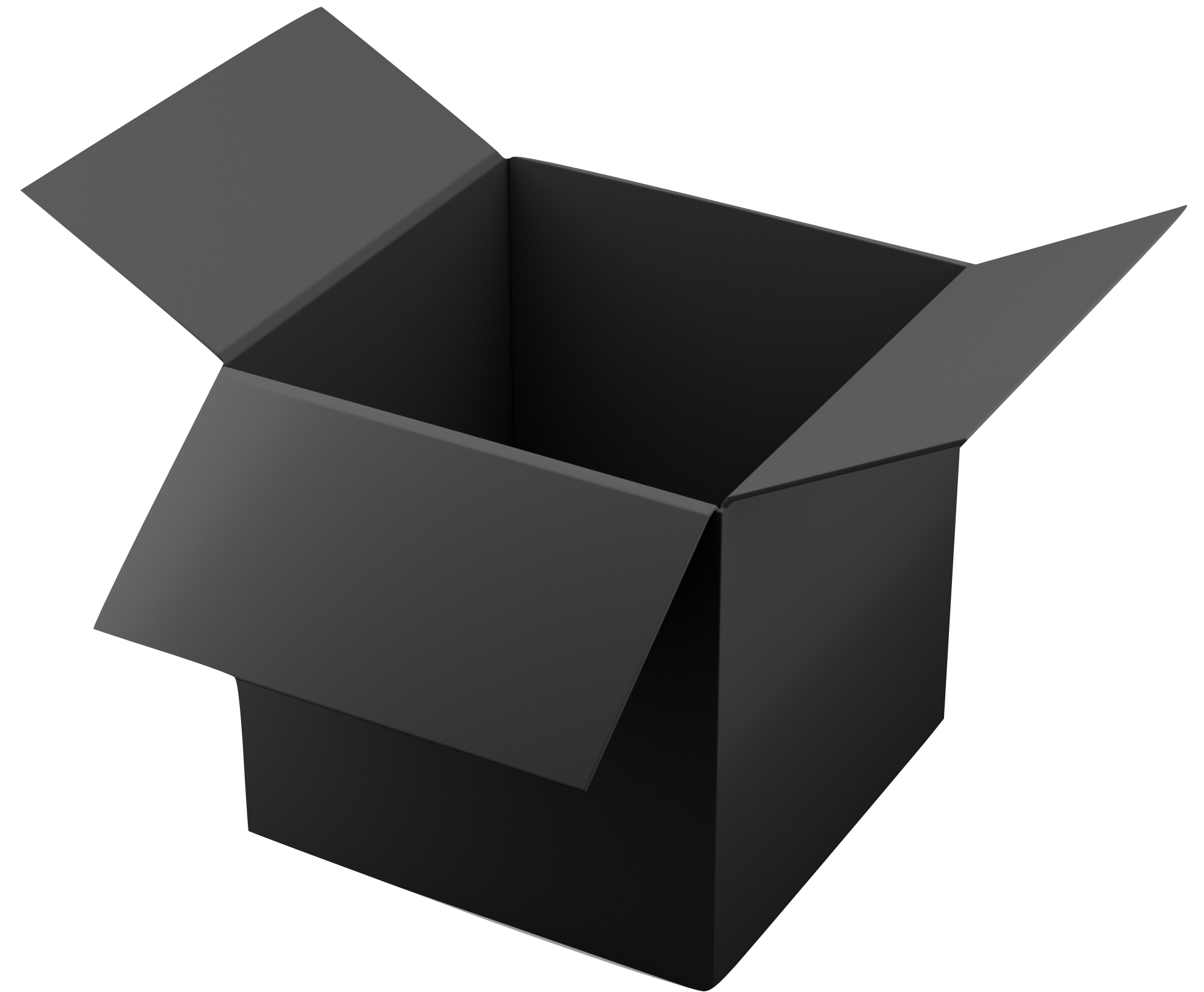
# AWS Fargate - Architecting Your Cluster

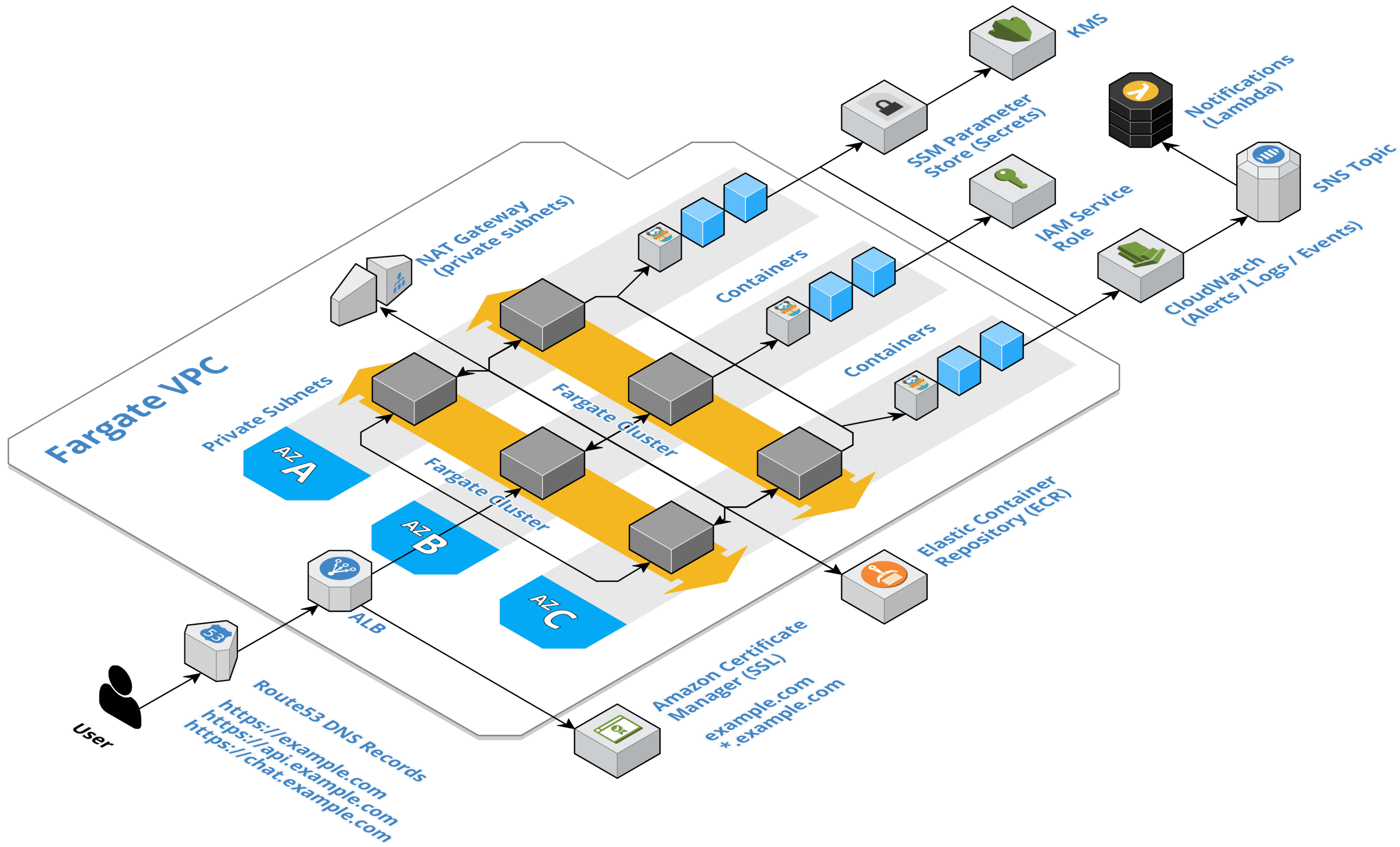
- Some tips:
  - Create a custom VPC
  - Create public subnets for ALBs
  - Create private subnets for Fargate (at least two to provide minimal HA)
  - Attach at least one NAT Gateway to these private subnets
  - Create a IAM Service Role with minimal policies as the default one for every ECS Service that will run inside your Fargate cluster



# AWS Fargate - Architecting Your Cluster

- Some tips:
  - Use SSM Parameter Store to export secrets
    - Preferably using a custom per-project AWS KMS Key
  - Create Cloudwatch Alarms for:
    - CPU scale up / down
    - RAM scale up / down
  - You can use a Lambda to notify every alarm:
    - Requires a SNS Topic







# AWS Fargate & Traefik

Let's do things easier.

# AWS Fargate & Traefik

- Traefik supports container service discovery by using AWS ECS API
  - You need to create a IAM user because Traefik is configured to use traditional AWS envvars:
    - `AWS_ACCESS_KEY_ID`
    - `AWS_SECRET_ACCESS_KEY`
  - You can use named profiles too but `~/.aws/{config,credentials}` files must be mounted (Traefik Docker Image contains only the binary)
  - With minimal setup you can setup Traefik to serve all containers with no need of multiple ALBs



# AWS Fargate & Traefik

- AWS setup requirements:
  - Traefik ECS Service must have access to ALB public subnets
    - Create a Security Group for Traefik
  - Create a ALB Target Group
    - Port: 80
    - Target Type: ip
  - On app ECS Service side, map the ECS private subnets as allowed subnets to be used



# AWS Fargate & Traefik

- AWS setup requirements:
  - Create a additional Security Group to be used as default by your Fargate apps
    - This is mandatory otherwise Traefik service will not dispatch requests to expected containers that responds for that URL
  - Inbound rules: ECS CIDR block
  - Outbound rules: set up as you need
- All ECS Services (in practice your apps) not requires more a ALB, Traefik does the hard work



# AWS Fargate & Traefik

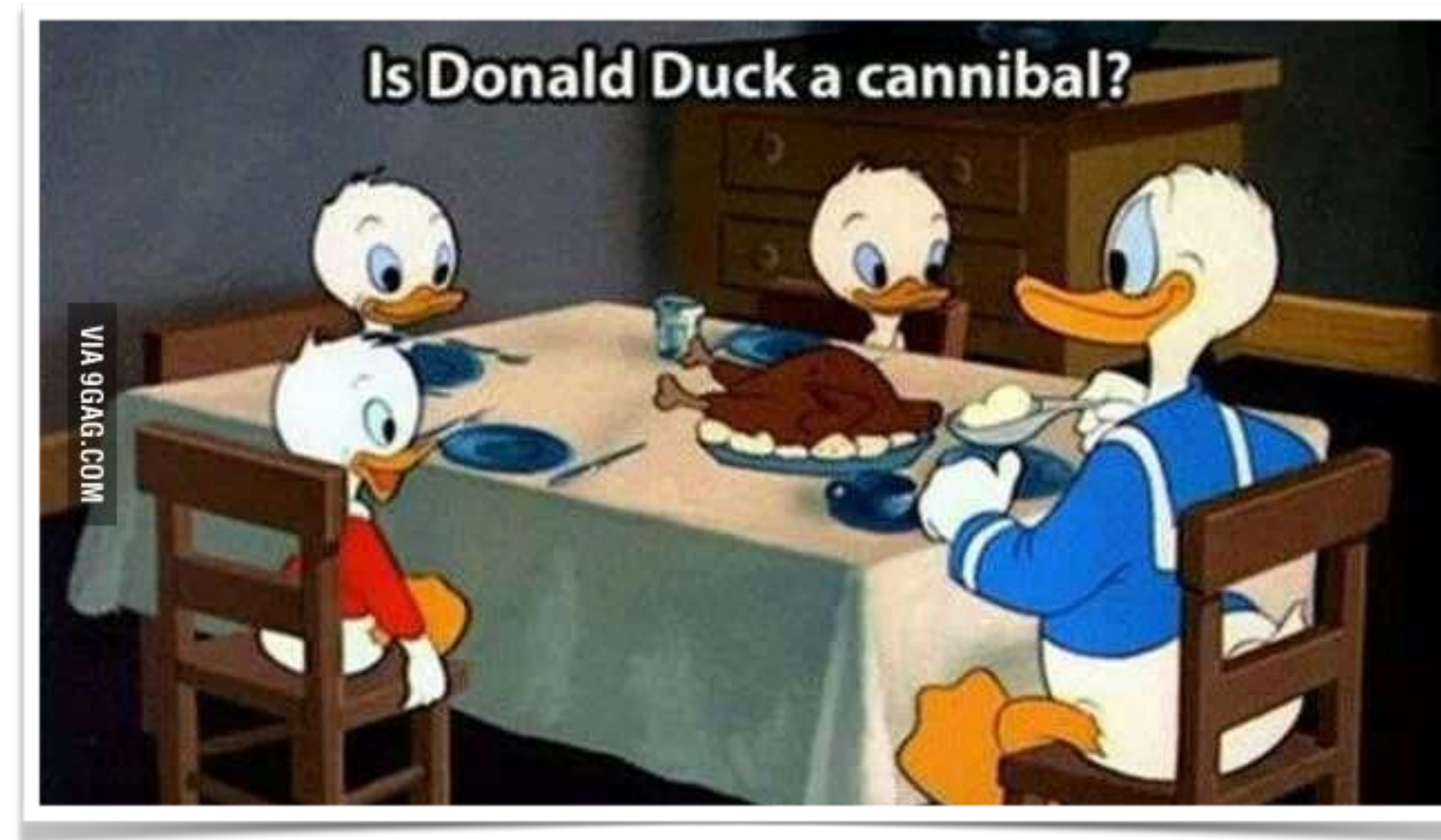
- You need to make a few hacks into Traefik ECS Task Definition Json file:
  - In command:
    - `--maxidleconnsperhost=100000`
      - Keep-Alive connections
    - `--ecs`
    - `--ecs.region=<aws-region>`
    - `--ecs.clusters=<cluster-name>`
    - `--ecs.domain=ecs.localhost`
    - `--ecs.exposedbydefault=false`
      - All these lines are the ECS config for Traefik



# AWS Fargate & Traefik

- You need to made a few hacks into Traefik Task Definition Json file:
  - In ulimits:
    - `softLimit = 1024`
    - `hardLimit = 100000`
    - These configs avoids `ERR_TOO_MANY_OPEN_FILES`
  - Export IAM AWS\* envvars as secrets using SSM Parameter Store





# Questions?

Traefik Docs: <https://docs.traefik.io>

AWS Fargate Docs: <https://aws.amazon.com/blogs/aws/aws-fargate/>

GitHub: <https://github.com/salizzar/aws-fargate-demo/> (under development)

Thank you.