EC2

Elastic Computing Cloud

Cloud Terminologies

- Region
- Availability Zone (AZ)
- Edge Location / Point of Presence (POP)

EC2

- Elastic Computing Cloud
- Service which provides resizable computing capacity in the cloud!
- Launch a VM in a minute!
  - OS / AMI (Amazon Machine Image)
  - CPU / Memory
- Scale up and down depending on the requirements!

EC2 - Pricing Model

Model	Description
on-demand	Launch a VM. Use it as long as you want. Pay / hour basis
reserved	Reserve a VM for 1 year or 3 years
dedicated	Fully dedicated physical hardware in the cloud for your use!
spot	Up to 90% discount (Unused VMs in the Data Center. AWS might reclaim these instances any moment - with a **2 minute** warning)

Security Groups

- By default, all the inbound requests are denied!
  - We have to explicitly add the "***allow***" rules.
    - Port
    - Source
- There are no "***deny***" rules. (It has to be handled separately - NACL)
- A Security Group can be attached to multiple EC2 instances
- An EC2 instance can be attached with multiple Security Groups

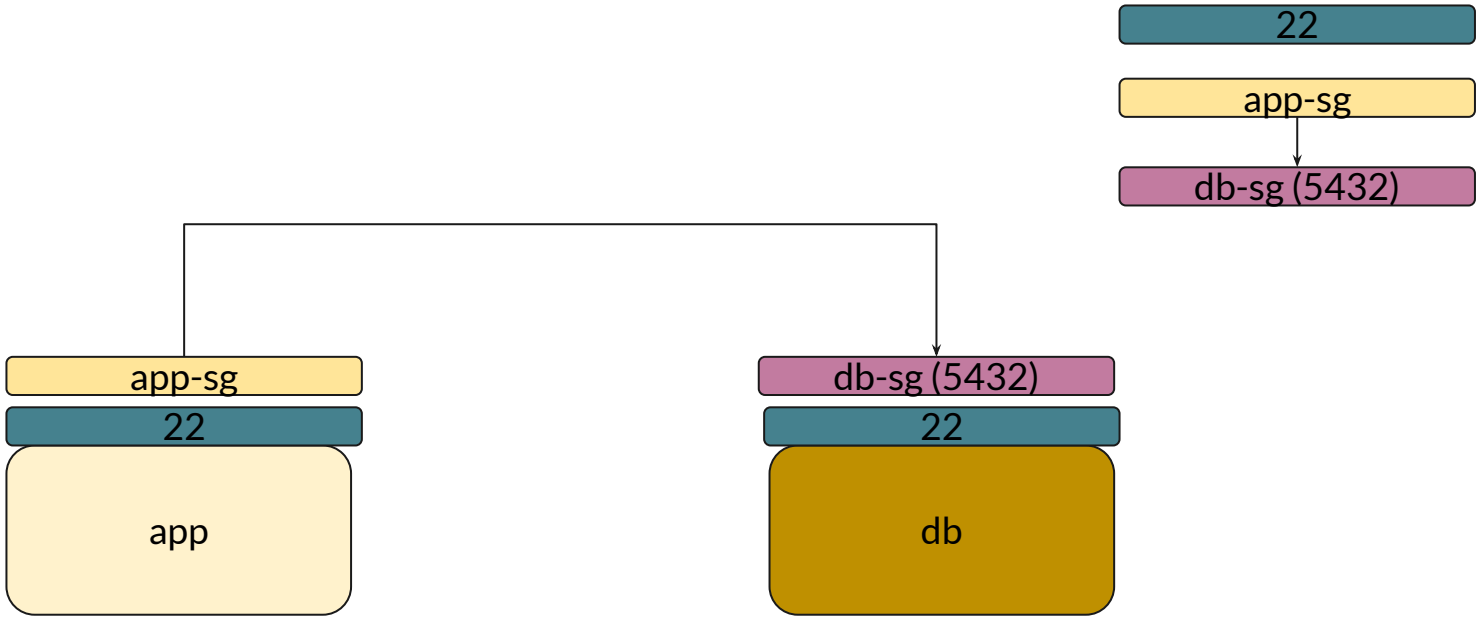
Security Groups

- By default, all the inbound requests are denied!
  - We have to explicitly add the "***allow***" rules.
    - Port
    - Source
      - What if I do not know the source IP / CIDR?
      - IP addresses might not be reliable as It might change!

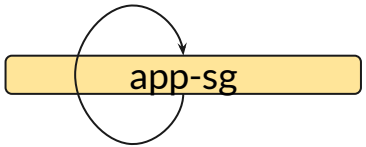
Setup

- Security Groups
  - ssh-22 - allow 22
- EC2
  - client
    - ssh-22
  - app
    - ssh-22
  - db
    - ssh-22

app → db



Self Referencing Rule



S3

Simple Storage Service

S3 Bucket

- Globally unique directory
  - **s3://vins**
- Web address
  - https://vins.s3.amazonaws.com/
  - https://vins.us-east-1.amazonaws.com/

S3 - Summary

- Secure, Durable and Highly Available **Object** Storage
- Object ⇒ Any File (txt, csv, img, css, movie, mp3, zip, backup …..)
- An Object can be 0 bytes to 5 TB
- "***Unlimited***" storage
    - You can store N number of objects.

S3: Data Breaches

- 2022 Pegasus airline
- 2021 Twitch
- **2019 Capital One**
- …
- ….

A Misconfigured Amazon S3 Exposed Almost 50 Thousand PII in Australia

November 06, 2017

Alteryx data breach exposed 123 million American households' information

By Tracey Lien

Dec. 22, 2017 9:40 AM PT

S3 - Summary

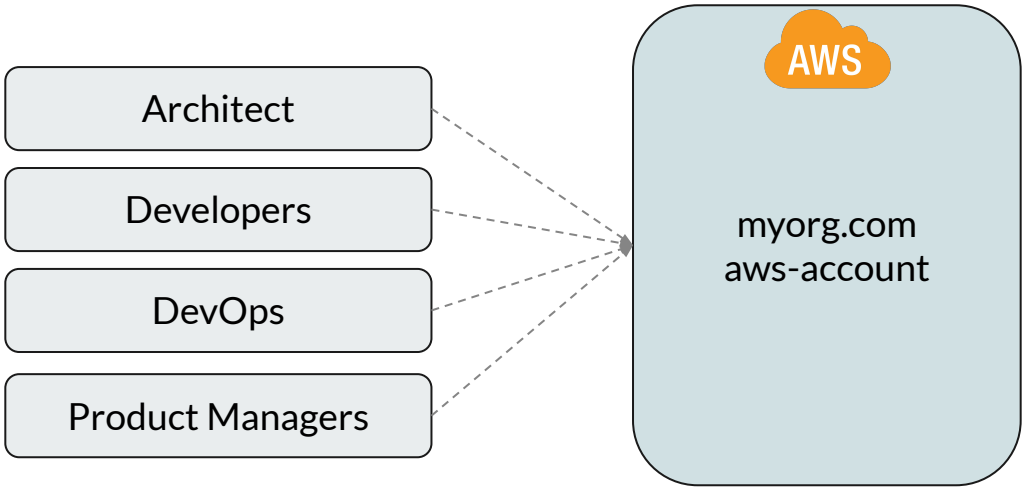
- If you have any sensitive info
  - Block all public access
  - Client side encryption
  - Server side encryption - KMS
  - Bucket Policy
  - Object Lock - to prevent deletion
  - Versioning - to maintain history
  - IAM

IAM

Identity Access Management

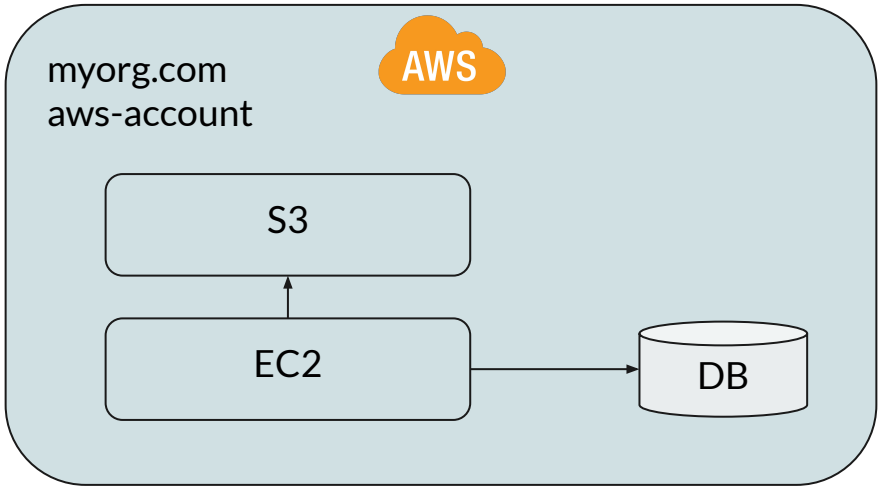
IAM

- To manage users and their level of access



IAM

- Service to service access



IAM

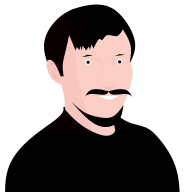
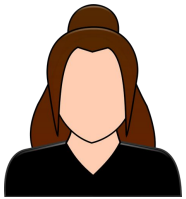
- Policy
- Users
- User Groups
- Roles

IAM - Policy

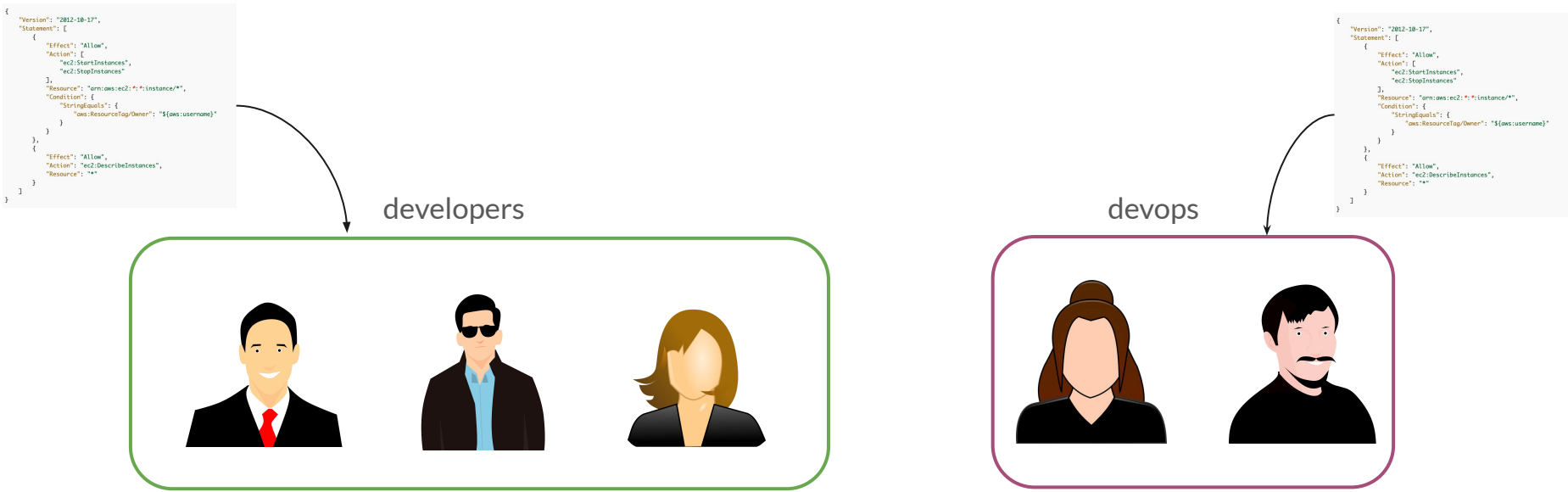
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Owner": "${aws:username}"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DescribeInstances",
            "Resource": "*"
        }
    ]
}
```

IAM - Users

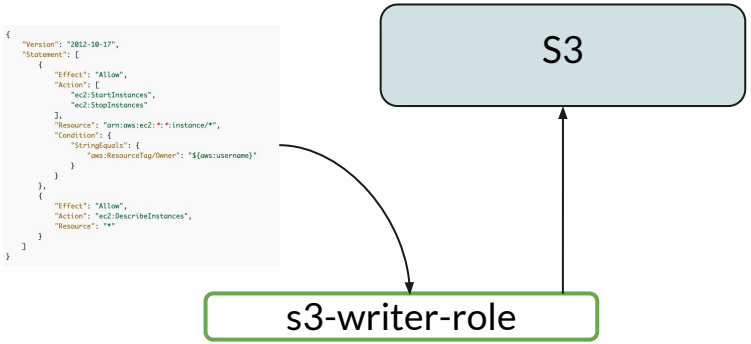



IAM - User Groups



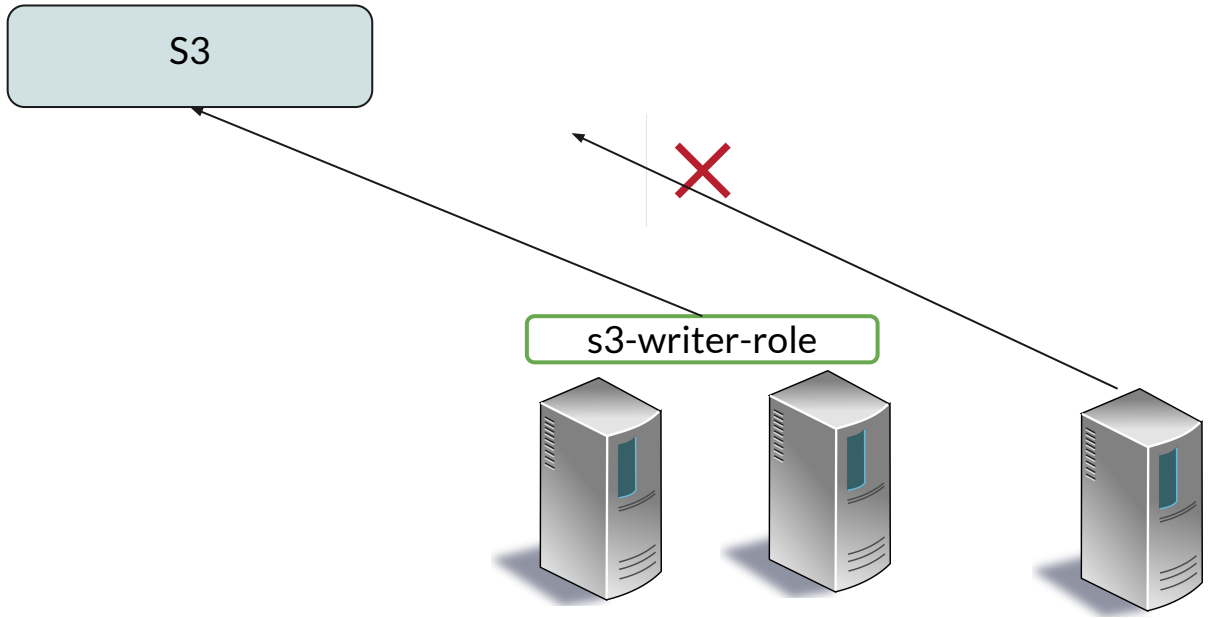
IAM - Roles



review-export-service

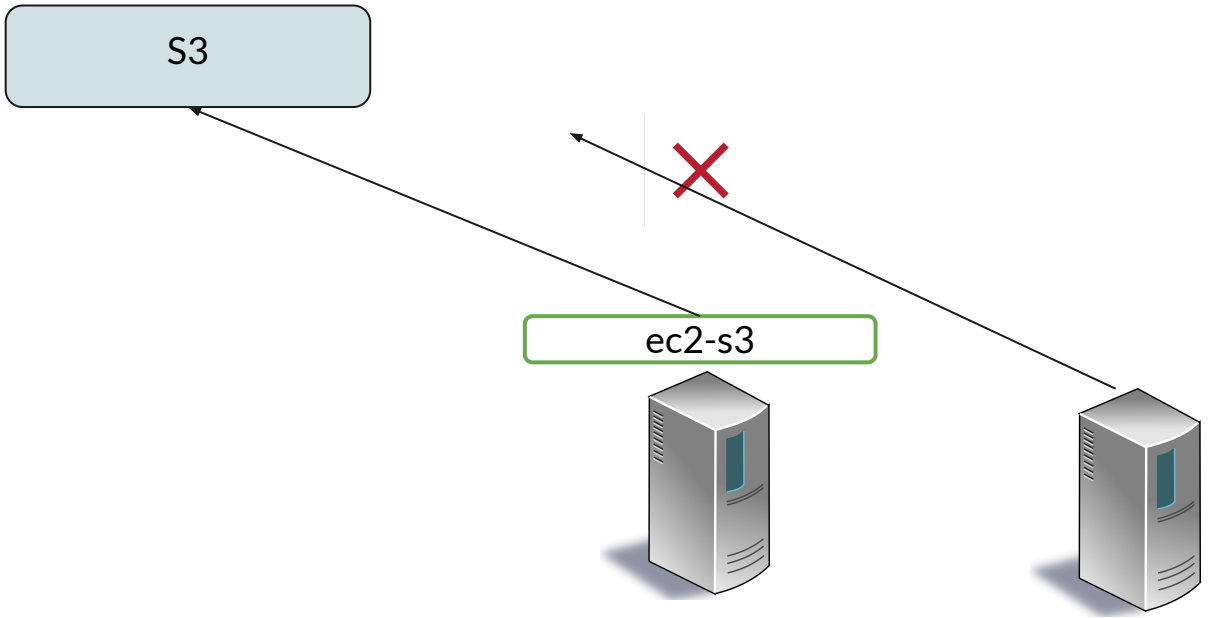
IAM - Roles

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "ec2:StartInstances",
               "ec2:StopInstances"
           ],
           "Resource": "arn:aws:ec2:*:*:instance/*",
           "Condition": {
               "StringEquals": {
                   "aws:ResourceTag/Owner": "${aws:username}"
               }
           }
       },
       {
           "Effect": "Allow",
           "Action": "ec2:DescribeInstances",
           "Resource": "*"
       }
   ]
}
```

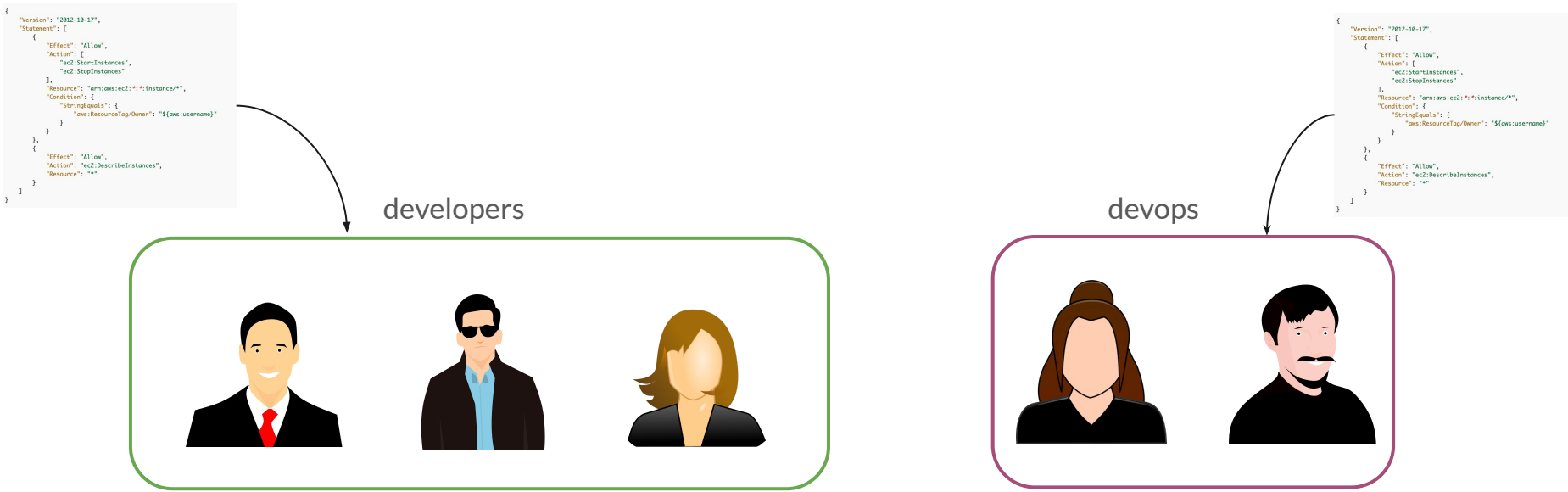


IAM - Roles

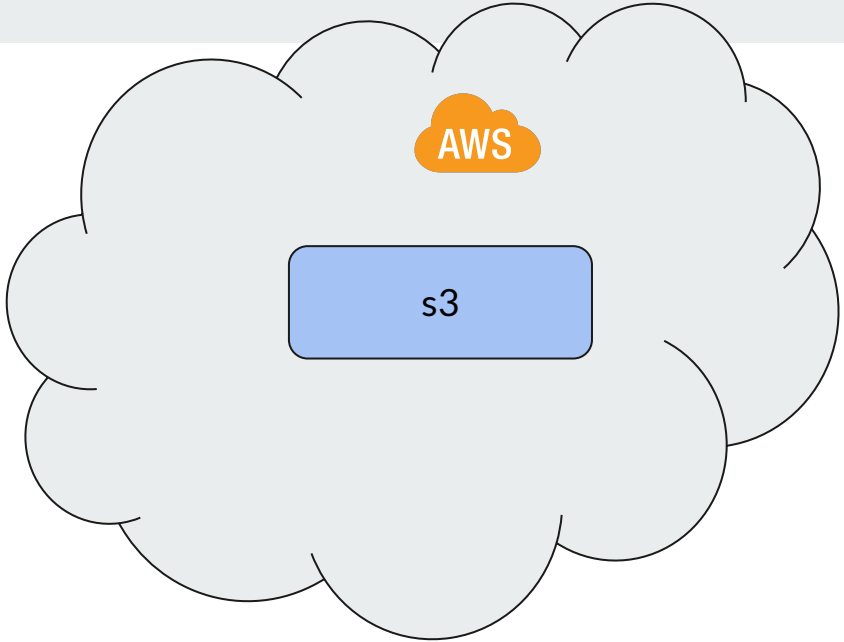
```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "ec2:StartInstances",
            "ec2:StopInstances"
         ],
         "Resource": "arn:aws:ec2:*:*:instance/*",
         "Condition": {
            "StringEquals": {
               "aws:ResourceTag/Owner": "${aws:username}"
            }
         }
      },
      {
         "Effect": "Allow",
         "Action": "ec2:DescribeInstances",
         "Resource": "*"
      }
   ]
}
```



IAM - User Groups

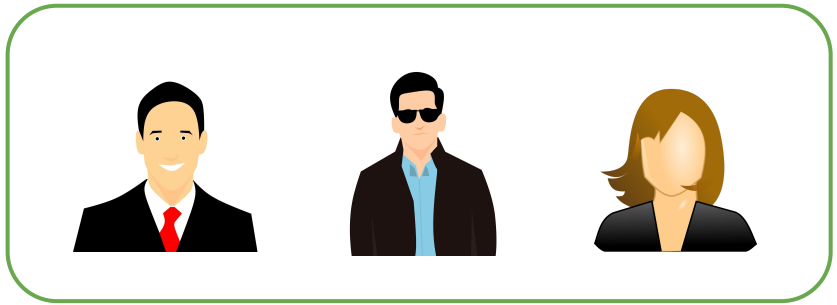


User Local Development

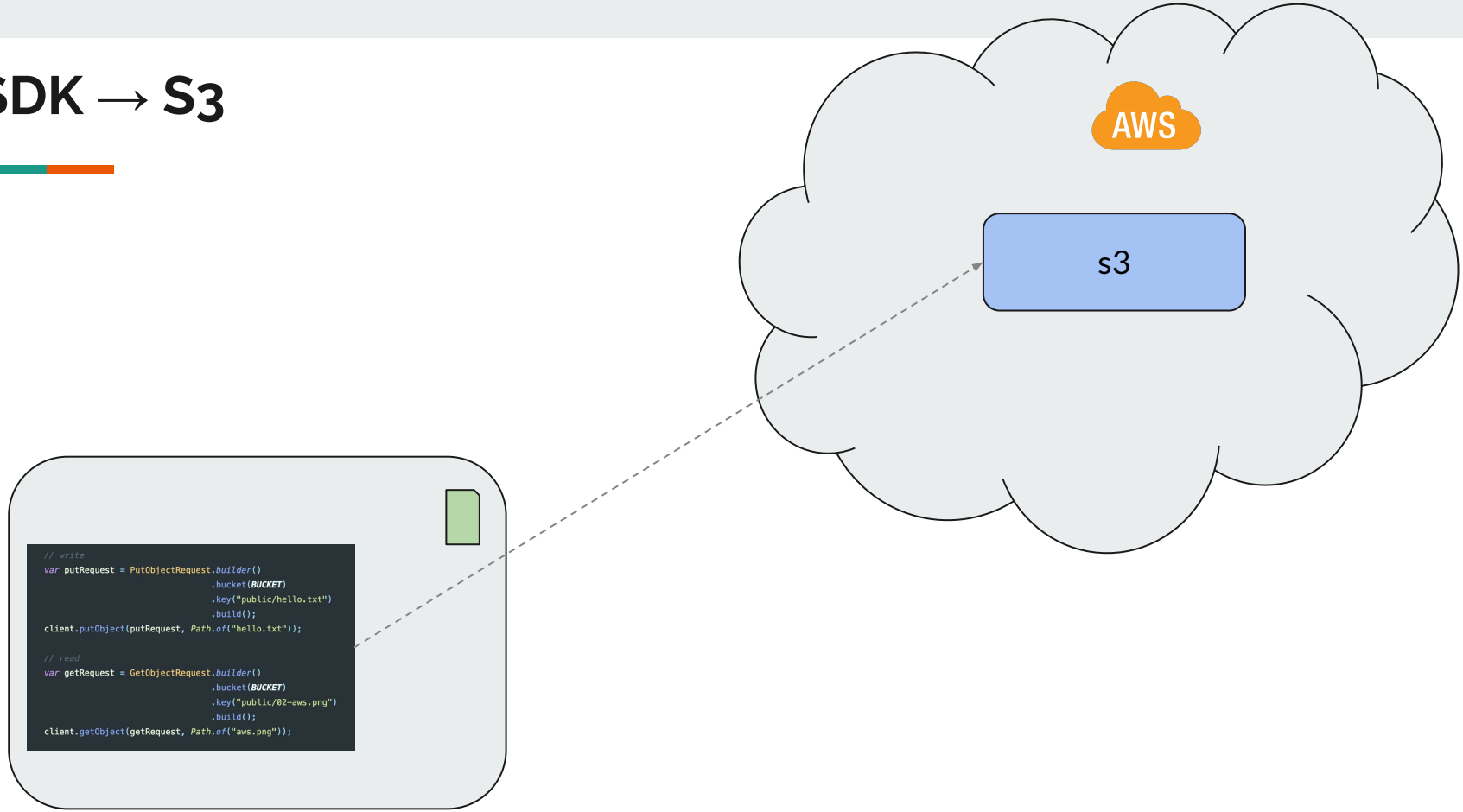



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Owner": "${aws:username}"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DescribeInstances",
            "Resource": "*"
        }
    ]
}
```

developers

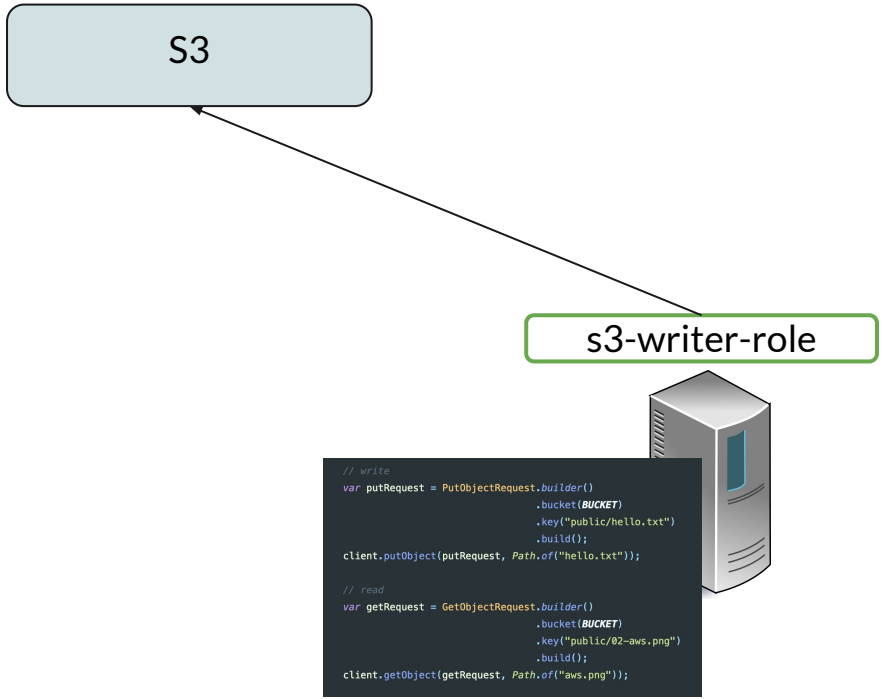


SDK → S3


AWS
s3
// write
var putRequest = PutObjectRequest.builder()
.bucket(BUCKET)
.key("public/hello.txt")
.build();
client.putObject(putRequest, Path.of("hello.txt"));
// read
var getRequest = GetObjectRequest.builder()
.bucket(BUCKET)
.key("public/02-aws.png")
.build();
client.getObject(getRequest, Path.of("aws.png"));

IAM - Roles

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Owner": "${aws:username}"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DescribeInstances",
            "Resource": "*"
        }
    ]
}
```



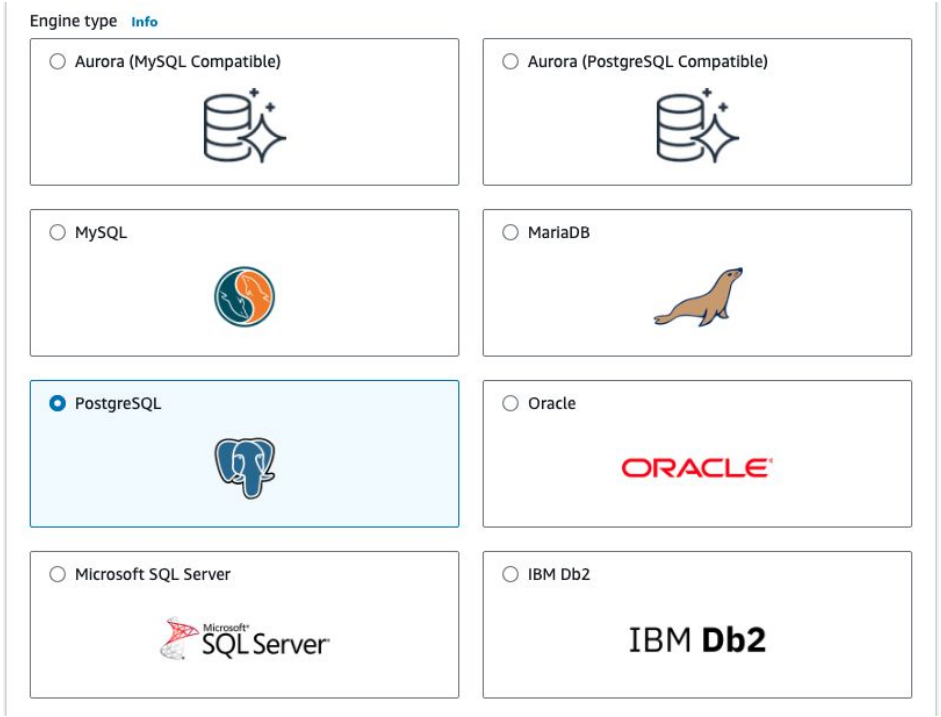
IAM

- Least Privilege Principle
- Root Account
  - Do NOT use for everyday tasks!
  - MFA
  - Create another user account
- Access keys
  - Do NOT use in the code
  - Rotate
  - Delete unused access keys

RDS

Relational Database Service

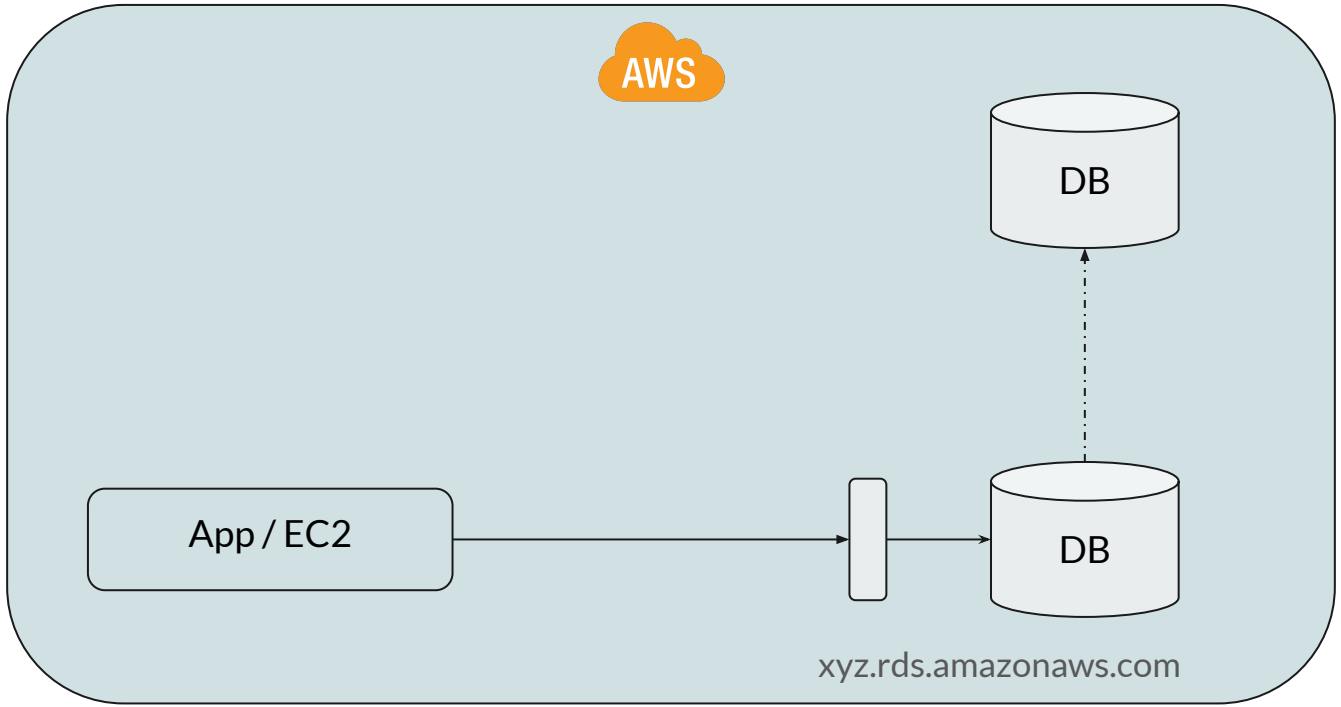
RDS


Engine type Info
Aurora (MySQL Compatible)
Aurora (PostgreSQL Compatible)
MySQL
MariaDB
PostgreSQL
Oracle
ORACLE
Microsoft SQL Server
Microsoft SQL Server
IBM Db2
IBM Db2

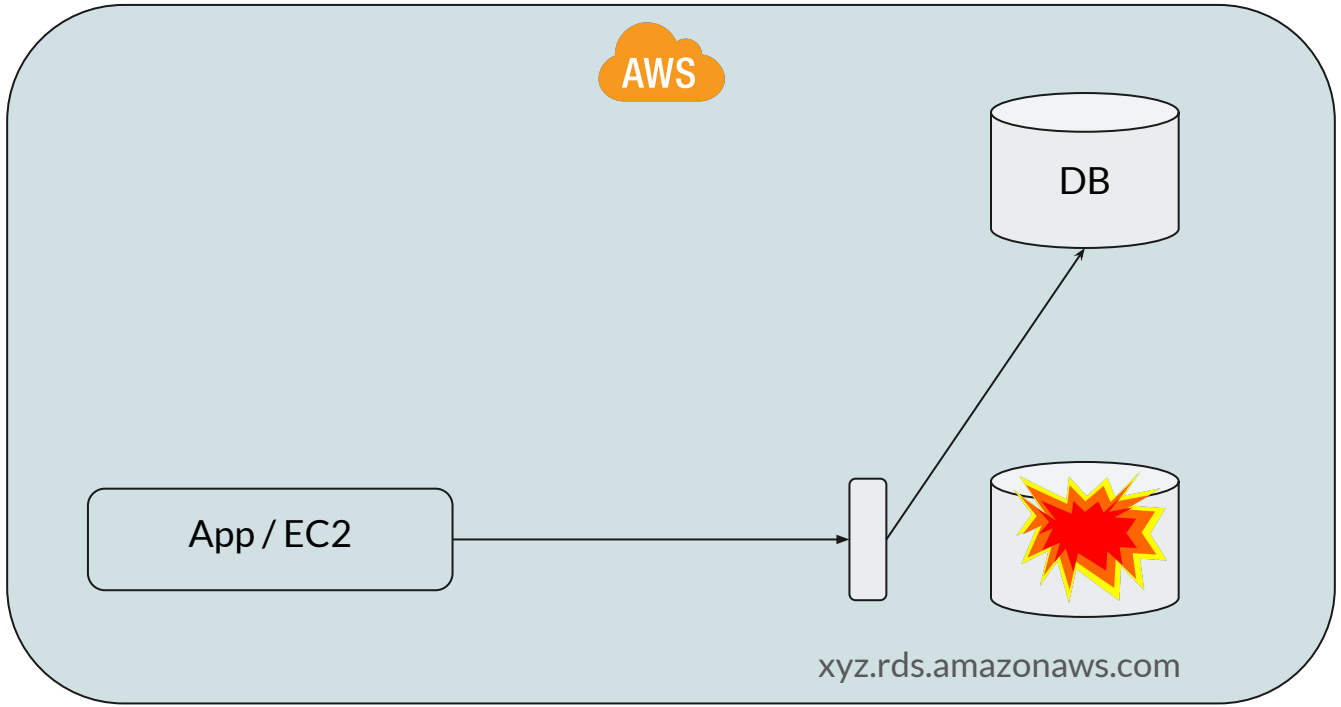
RDS

- Fully Managed by AWS
  - Highly Available
  - Administrative Tasks
    - Software patching
    - Scheduled maintenance
    - Periodic backups
- Point-in-time recovery

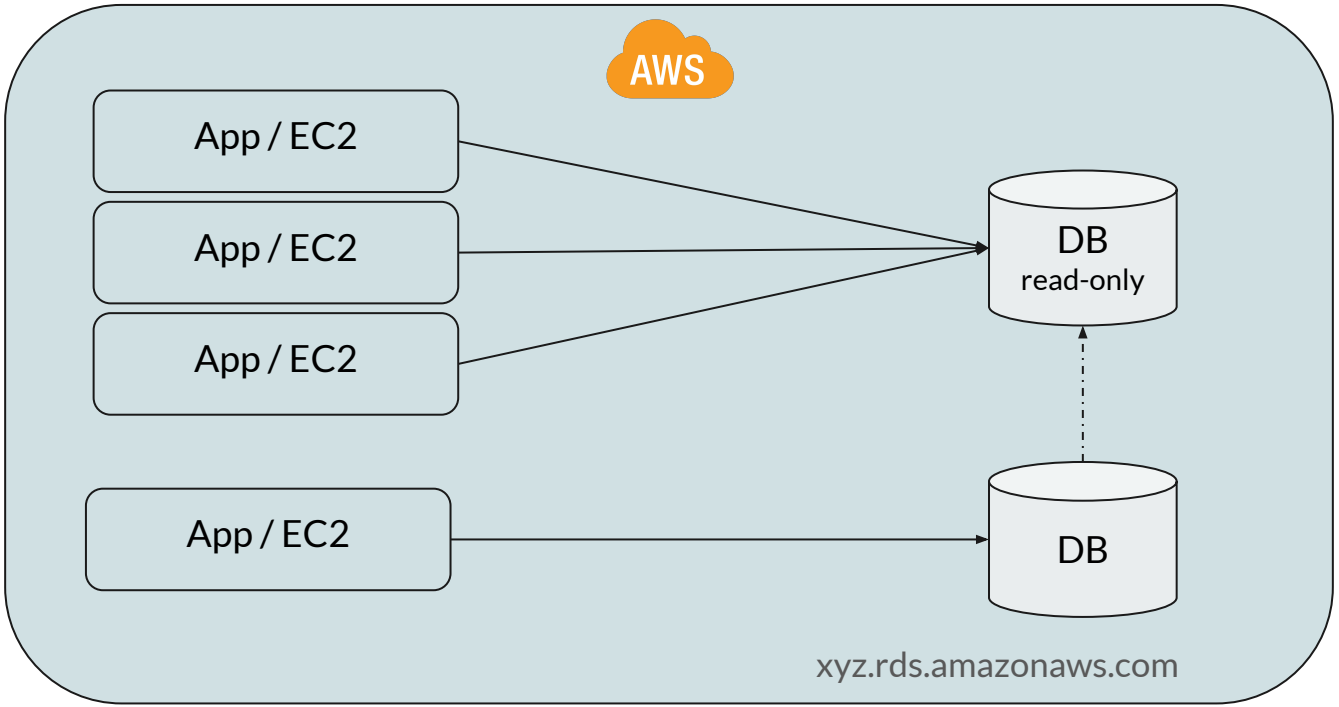
RDS - Multi AZ



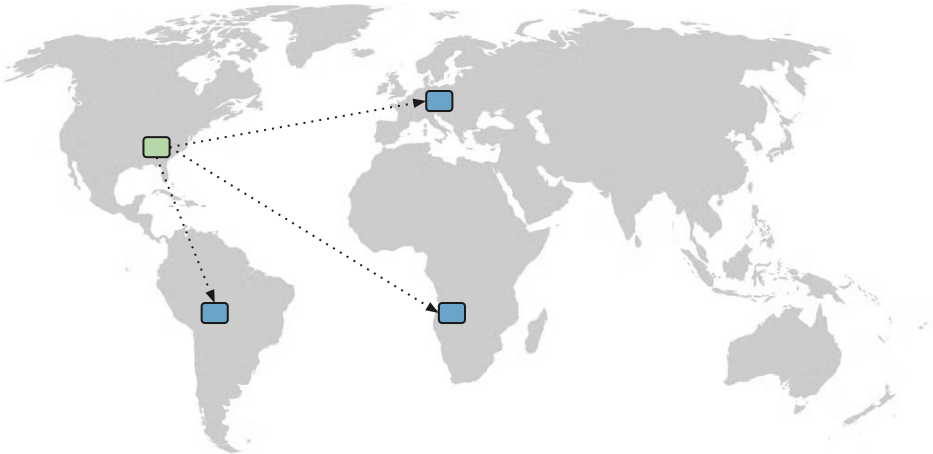
RDS - Multi AZ



RDS - Read Replica



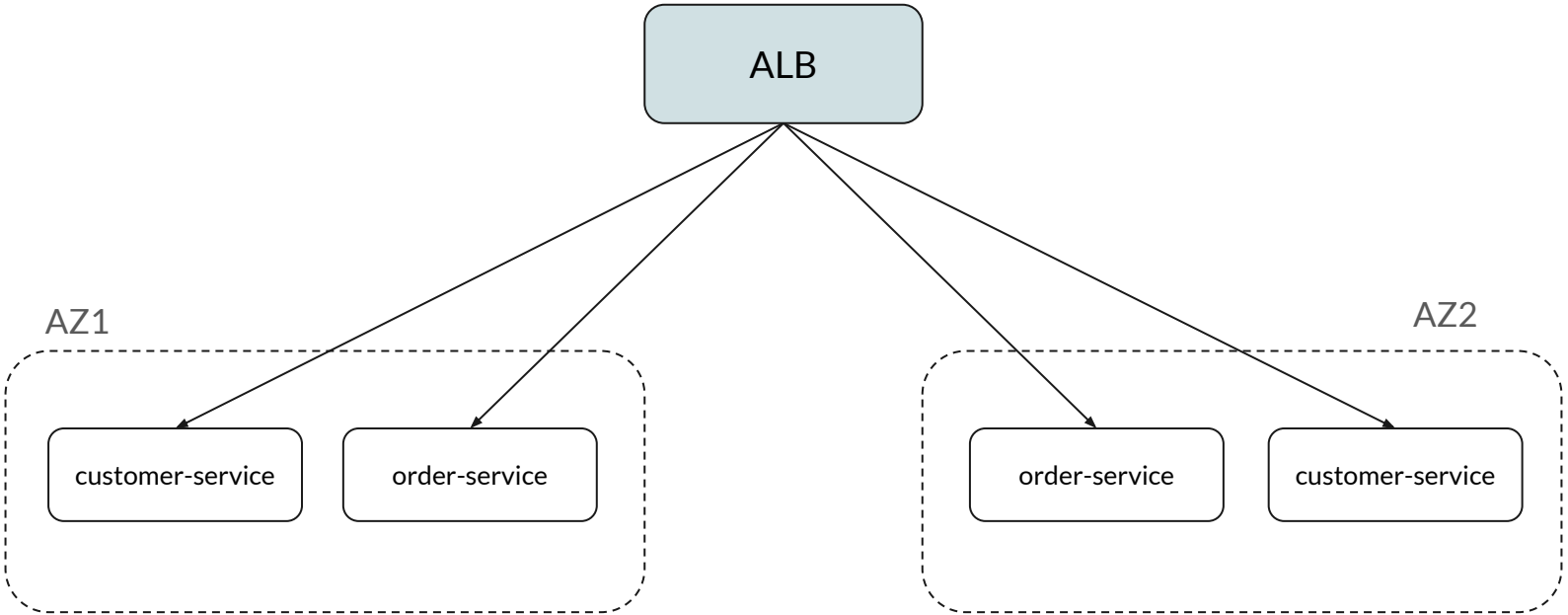
RDS - Read Replica



ALB

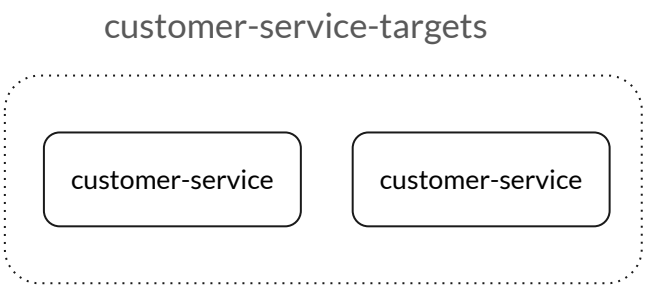
Application Load Balancer

Setup


ALB
AZ1
customer-service
order-service
AZ2
order-service
customer-service

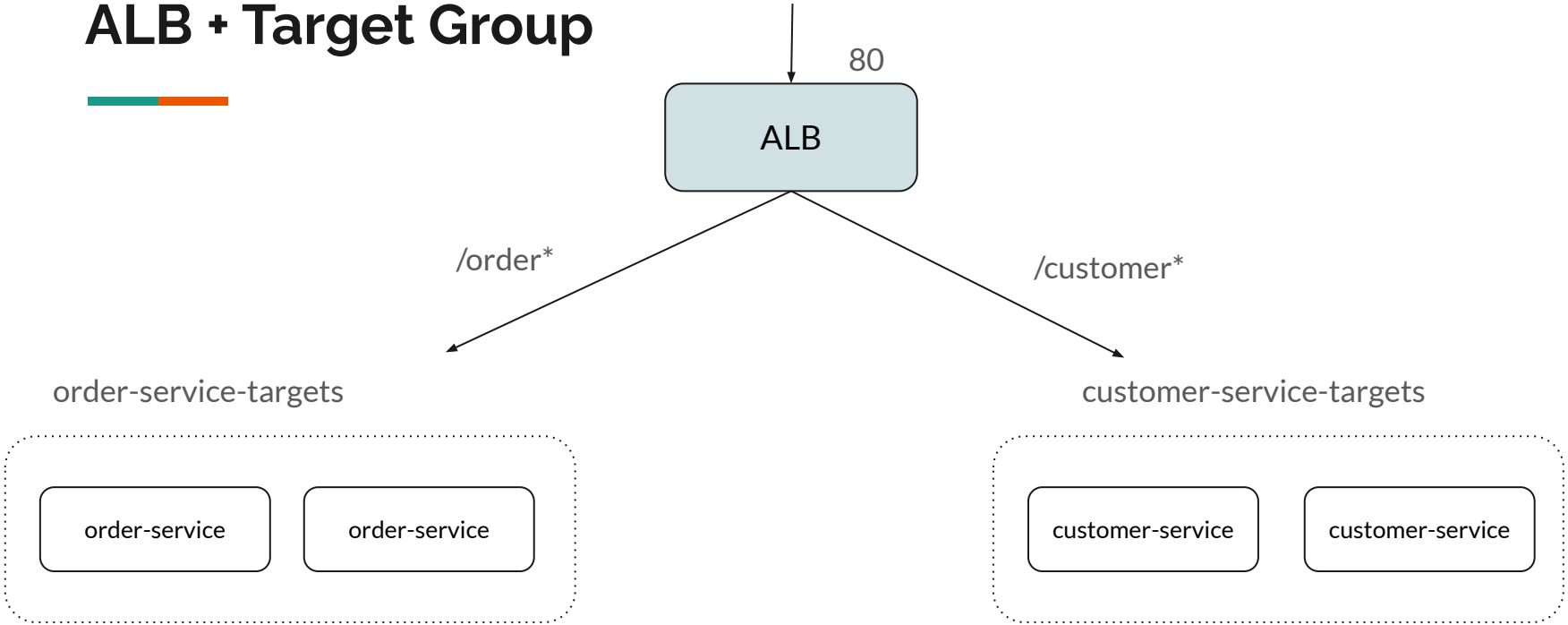
Target Group

- 80
- /
- 5 seconds interval
- 3 consecutive success ⇒ healthy
- 2 consecutive failures ⇒ unhealthy



- 8080
- /actuator/health
- 30 seconds interval
- 3 consecutive success ⇒ healthy
- 3 consecutive failures ⇒ unhealthy

ALB + Target Group



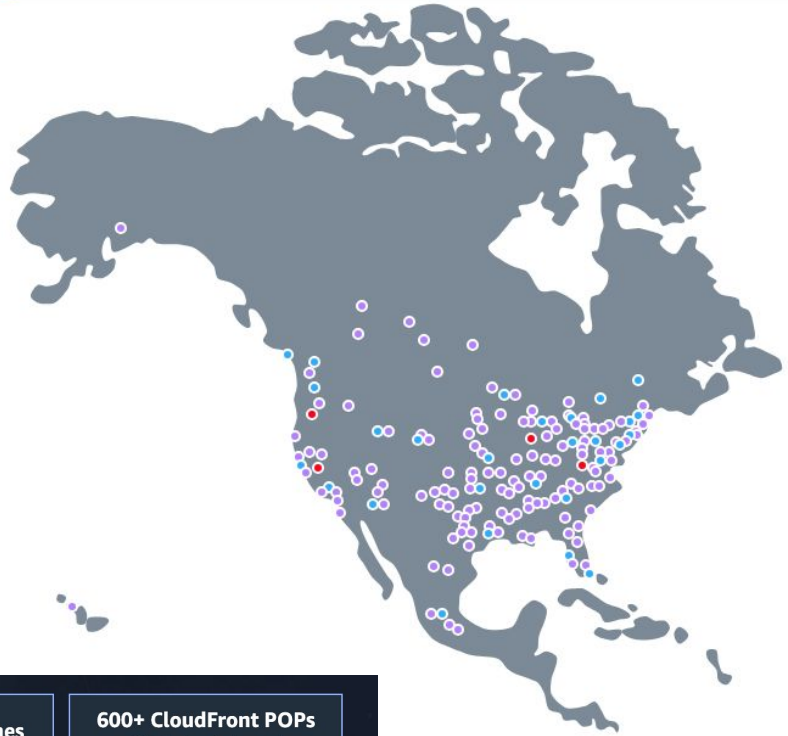
- 80
- /
- 5 seconds interval
- 3 consecutive success ⇒ healthy
- 2 consecutive failures ⇒ unhealthy

CloudFront

Content Delivery Network

Edge Locations

North America | South America | Europe | Middle East | Africa | Asia Pacific | Australia and New Zealand

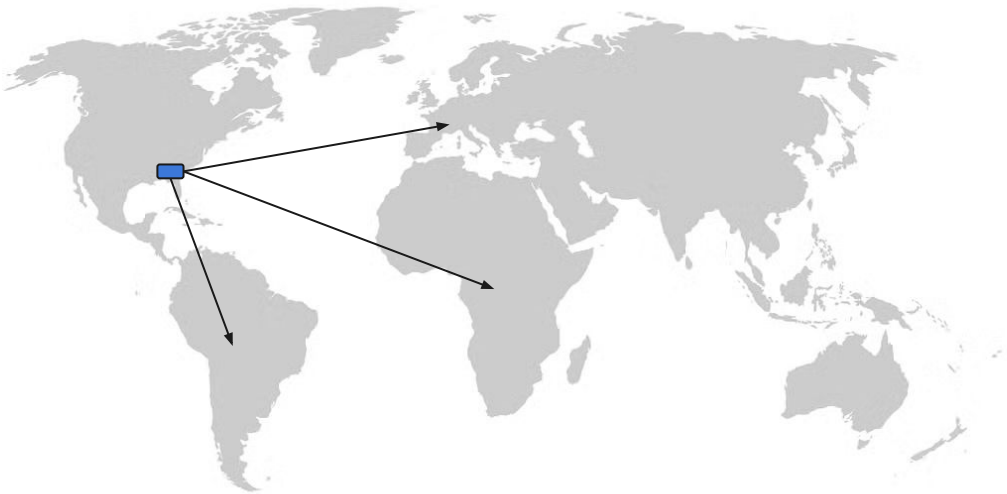


33 launched Regions
each with multiple Availability Zones

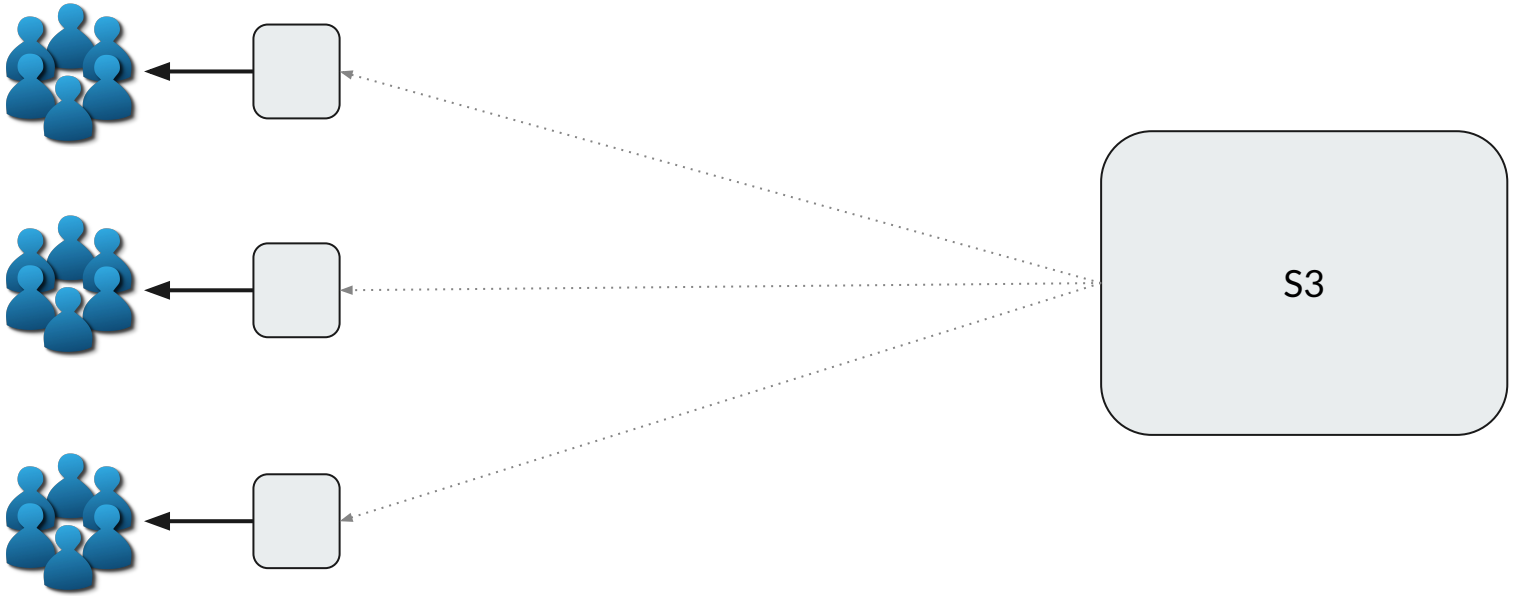
105 Availability Zones

600+ CloudFront POPs
and 13 Regional edge caches

Content Delivery



Content Delivery

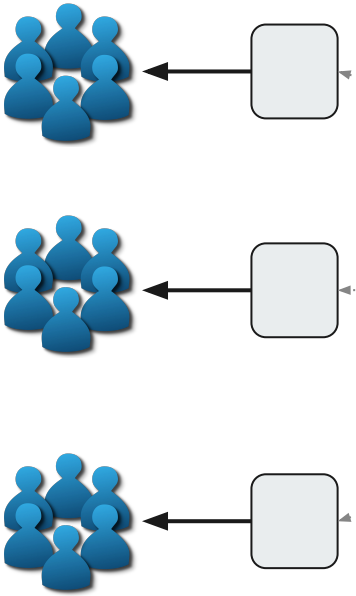


Additional Features!

- DDoS protection
- Geo Restrictions
- Signed URLs
- SSL Negotiation
- Edge Computing

Setup

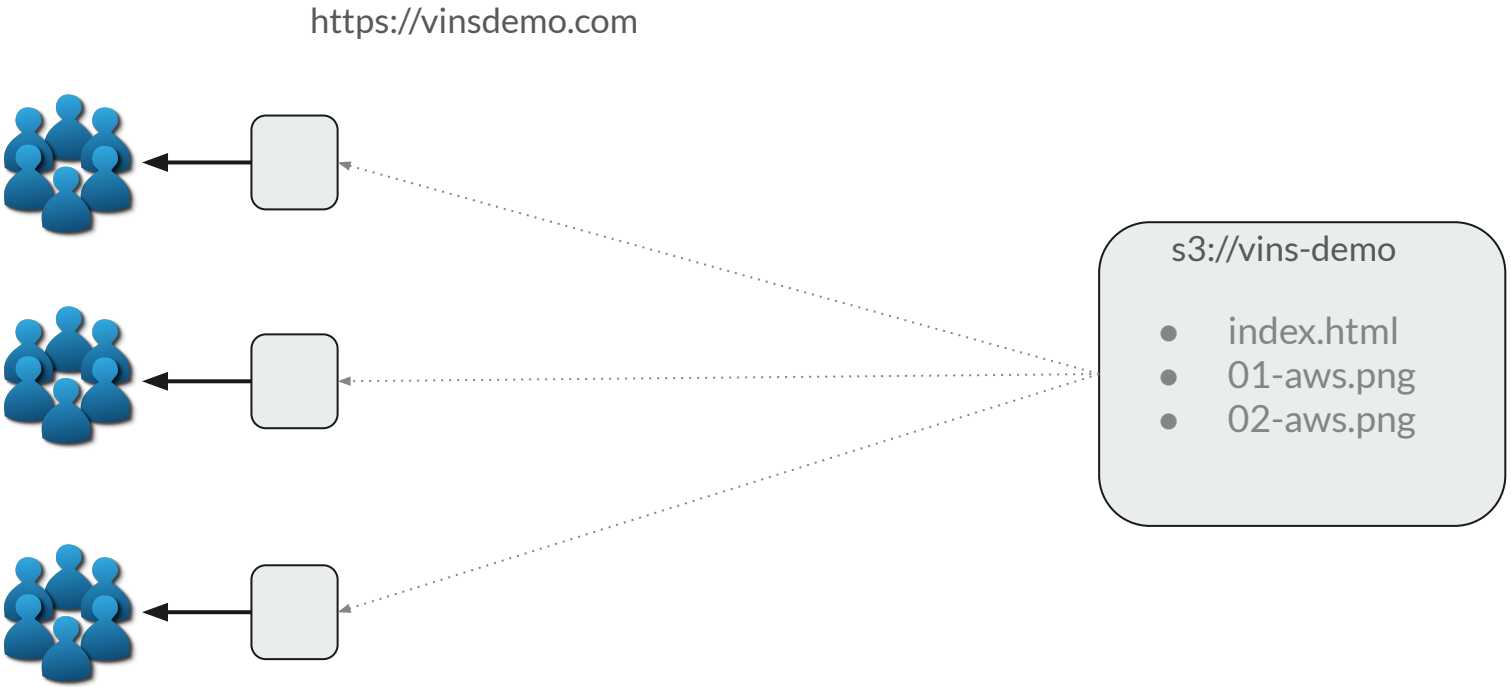
We can serve static content w/o enabling public access to the bucket by using CloudFront



s3://vins-demo

- index.html
- 01-aws.png
- 02-aws.png

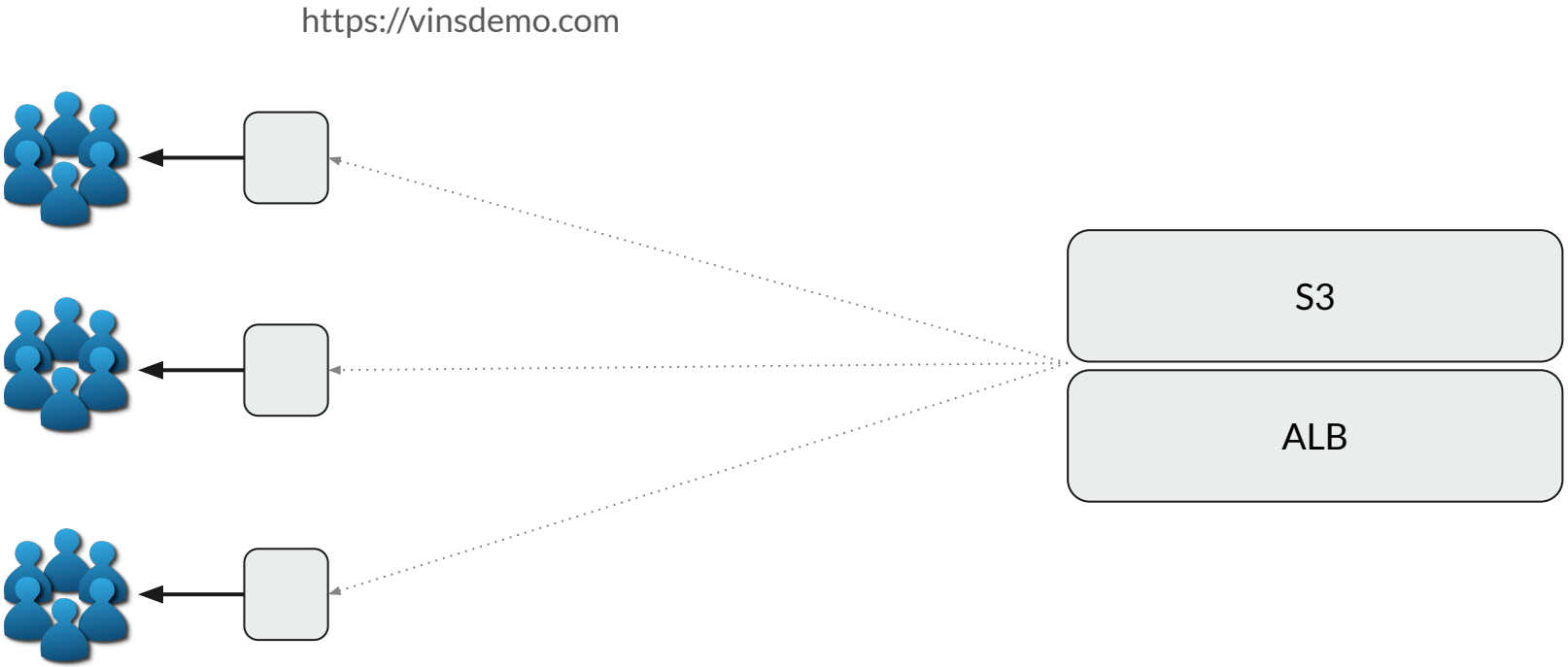
How To Integrate With My Domain



CloudFront Distribution - Multiple Origins

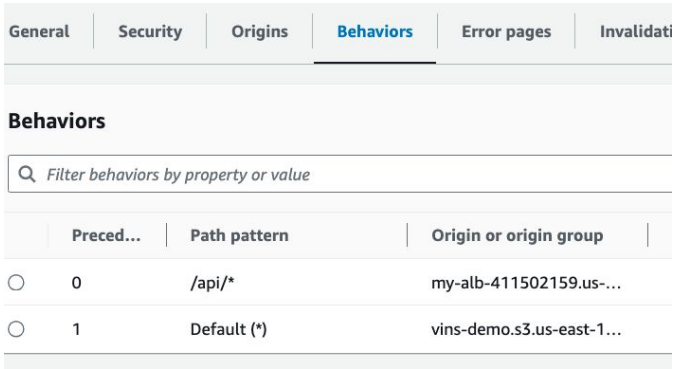
- static
  - img, js, css… ⇒ S3
- dynamic
  - api ⇒ ALB

Multiple Origins



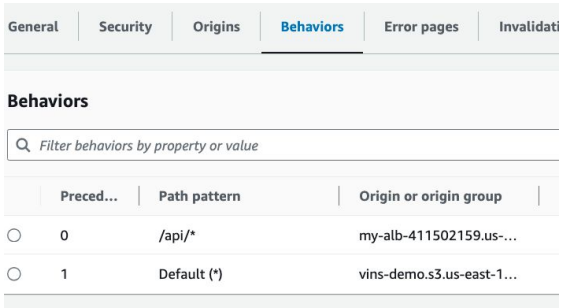
CloudFront Distribution

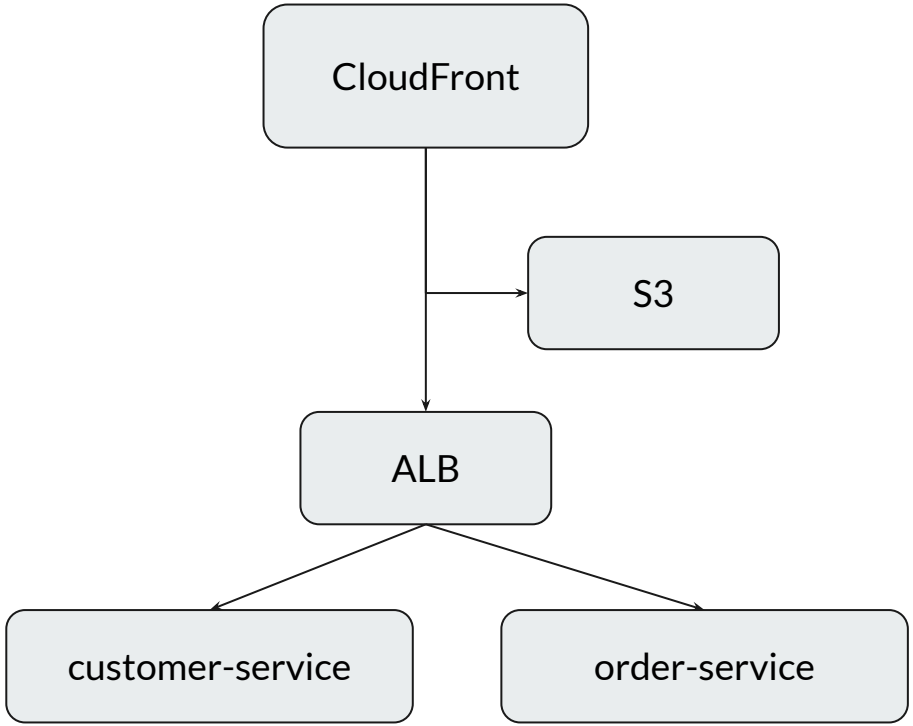
- https://xyz.com
  - ⇒ S3
- https://xyz.com/api/customer/1
  - ⇒ ALB



FAQ: Multiple Path Based Routing?

- https://xyz.com
  - ⇒ S3
- https://xyz.com/api/customer/1
- https://xyz.com/api/order/1
  - ⇒ ALB





VPC

Virtual Private Cloud

VPC - Summary

- Virtual network dedicated to your AWS account.
- Logically isolated section in the AWS Cloud where we create our resources like EC2, RDS, ALB..etc
- We have complete control on ***IP address ranges, subnets creation, route table configuration*** etc

Subnet - Summary

- Logical subdivision of IP address range within VPC
- Subnet resides within a single AZ
- We create EC2 instances, Databases, ALBs into specific subnets.
- Benefits
  - Isolating resources within VPC
  - Control inbound and outbound traffic

Default VPC - Summary

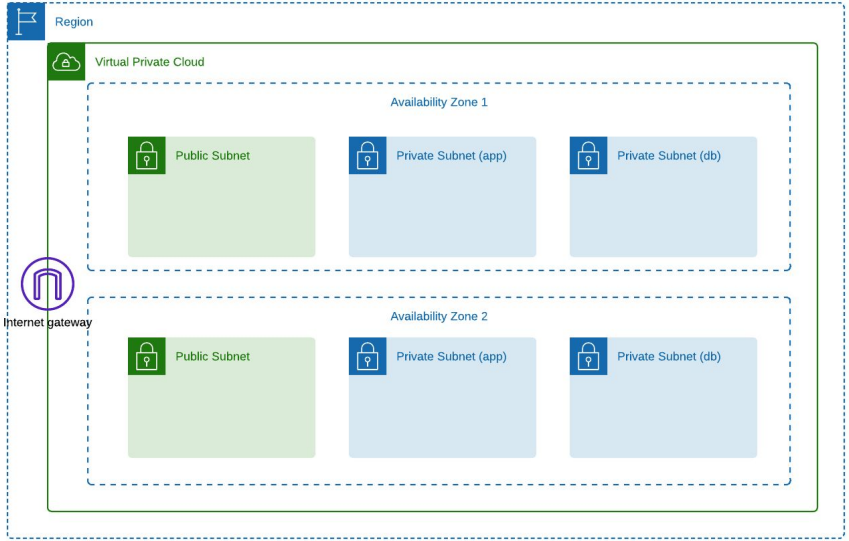
- Each region comes with a ***default VPC***
  - Default VPC has 1 ***subnet per AZ.***
    - ***It is a public subnet.***
    - A subnet which has direct access to the internet!

Public Subnet - Summary

- Security Risk
- ***Security Group*** is not enough!
  - People make mistakes!
- We have had many data breaches!
  - Not because of AWS.
  - Poor understanding of network config / firewall rules etc!

Recommended Approach - Summary

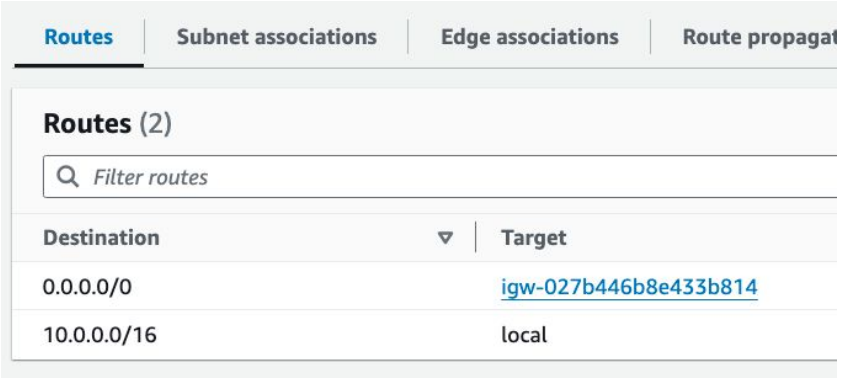
- Create your own VPC for your application!
  - With multiple subnets across multiple AZs to provide high availability and fault tolerance.
- Customize network settings / IP address ranges
- Deploy our application in **private subnets** which are NOT directly reachable in the internet.
- Use AWS resources like CloudFront / ALB as front-facing while Our application is hidden behind these AWS resources in private subnets.
- Our app can be accessed only via ALB / CloudFront (using security groups / bucket policies).

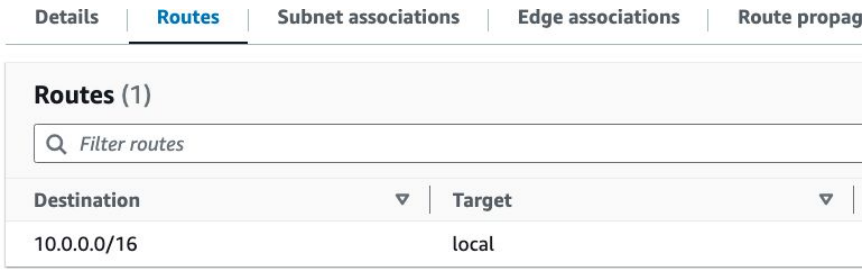


Internet Gateway - Summary

- Provides internet connectivity to the VPC.
- End users request can reach VPC (ALB / EC2 etc).
- Our apps can send request to internet.

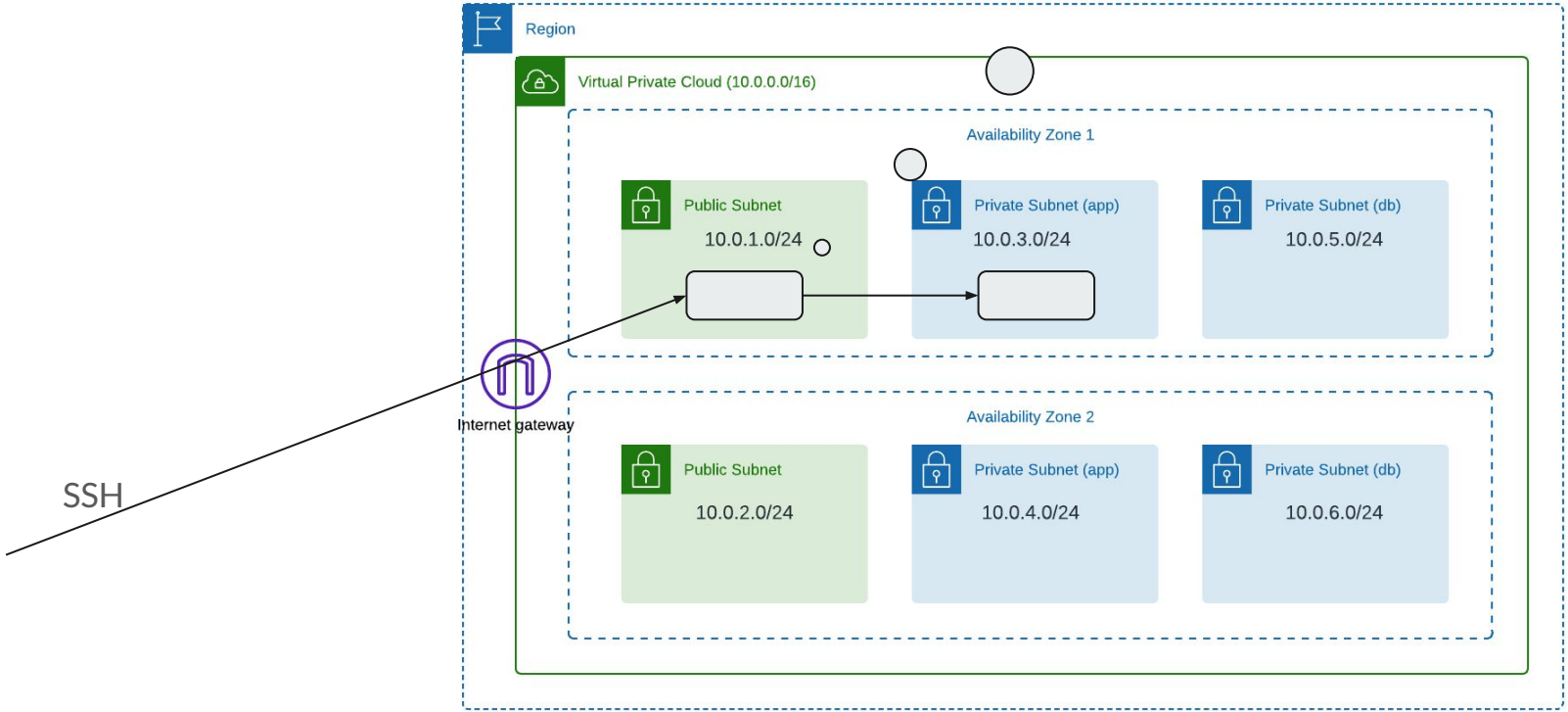
Public / Private Subnet - Summary





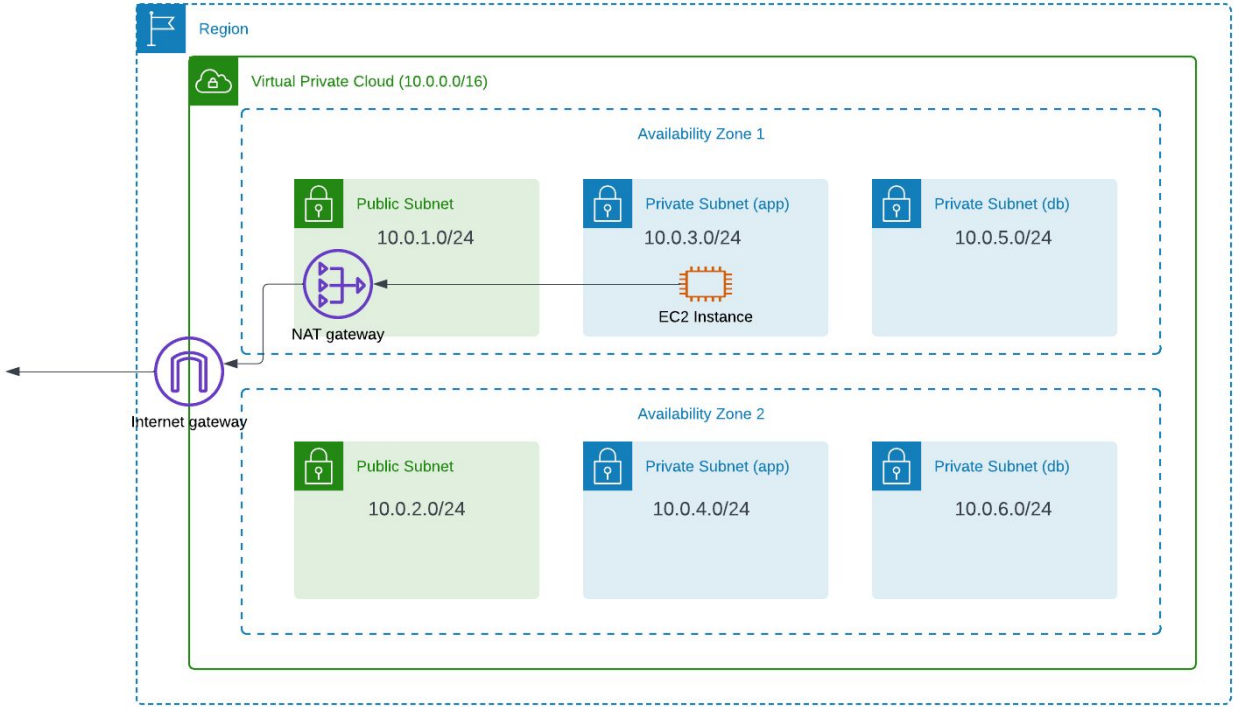
SSH - Private Instance

Bastion Host
open port 22 for corporate network

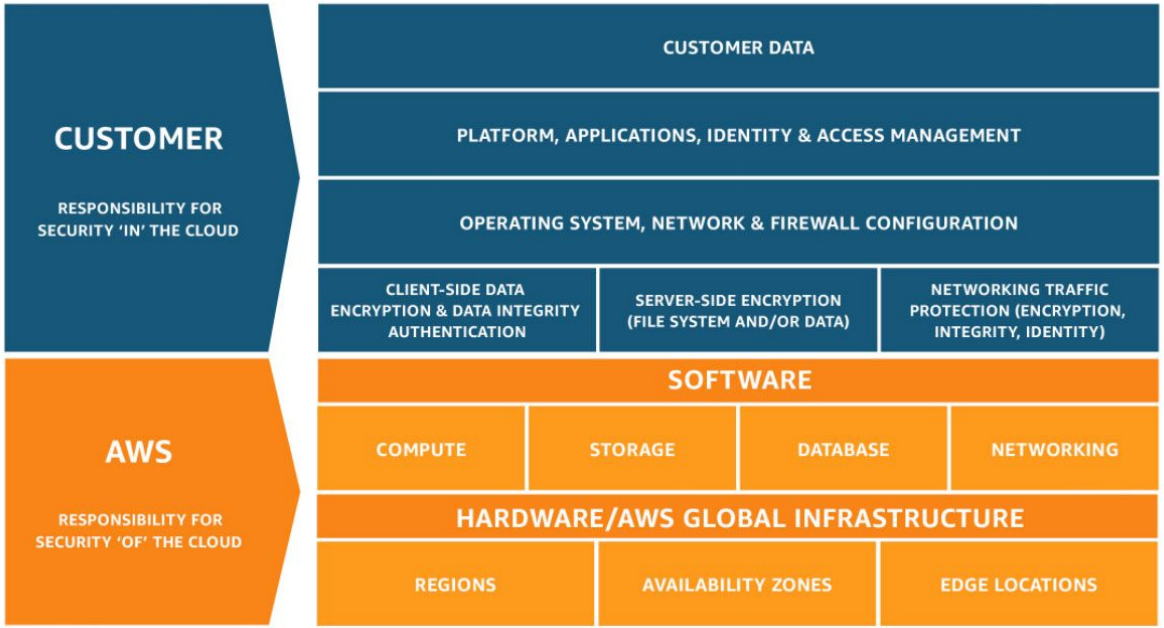


NAT Gateway

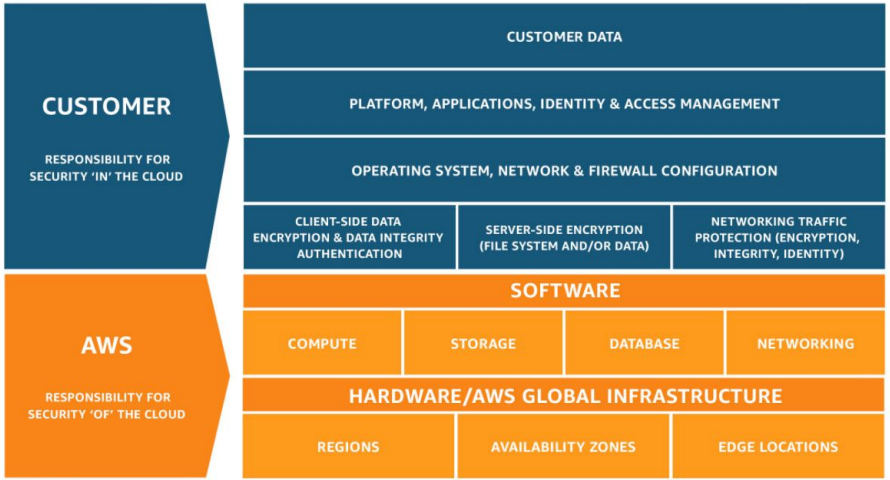
EC2 instance in the private subnet can access internet using NAT gateway! 1 NAT Gateway per AZ

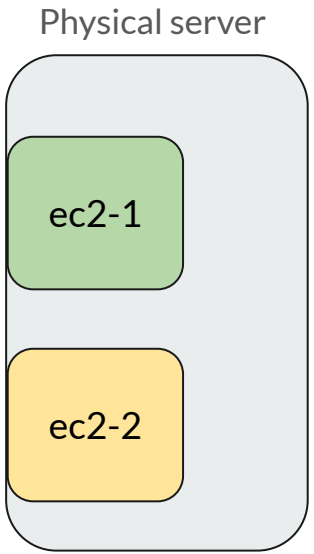


AWS Shared Responsibility Model



AWS Shared Responsibility Model





NACL - Summary

- Network ACL - can be used to block requests using the given CIDR!

Microservices

Java + Spring Boot

Netflux - Movie Service

Table: Movie

id	title	release_year	genre
1	Inception	2010	ACTION
2	The Godfather	1972	CRIME

```
enum Genre {

    ACTION,
    COMEDY,
    CRIME,
    DRAMA;

}
```

```
record MovieDto(Integer id,
                String title,
                Integer releaseYear,
                Genre genre) {
}
```

API Details

GET	/api/movies	***List<MovieDto>***	returns all the movies.
GET	/api/movies/{genre}	***List<MovieDto>***	returns all the movies for the given genre.

Netflux - Customer Service

Table: Customer

id	name	favorite_genre
1	Sam	ACTION
2	Mike	CRIME

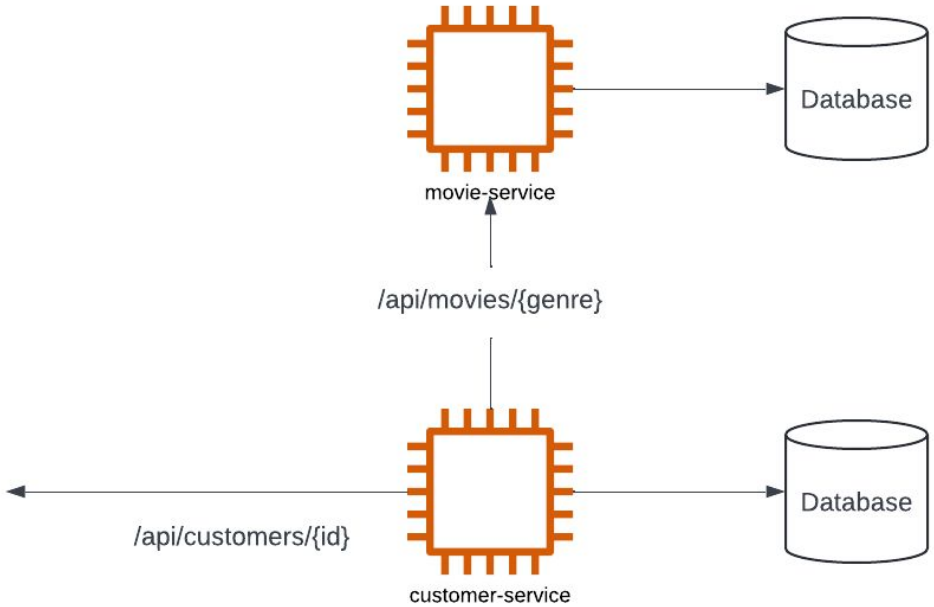
```
record CustomerDto(Integer id,
                   String name,
                   Genre favoriteGenre,
                   List<MovieDto> recommendedMovies) {
}
```

```
// PATCH /api/customers/1/genre
{
  "favoriteGenre": "ACTION"
}
```

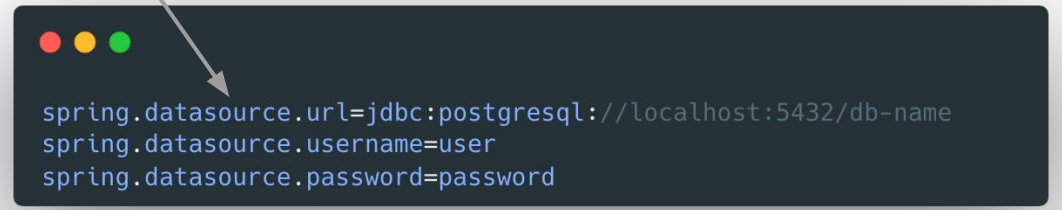
API Details

GET	/api/customers/{id}	**CustomerDto**	returns Customer info along with recommended movies based on the customer's favorite genre!
PATCH	/api/customers/{id}/genre	**Void**	Updates customer genre.

Netflux


Database
movie-service
/api/movies/{genre}
/api/customers/{id}
Database
customer-service

Spring Boot 3.1+



ECR - Public Registry

- ECR → Elastic Container Registry (AWS Cloud)
- We will use ECR public registry to pull ***postgres***
- We will run Integration Tests in the AWS Cloud as part of CI/CD
- Docker Hub has rate limiting

Problem Detail

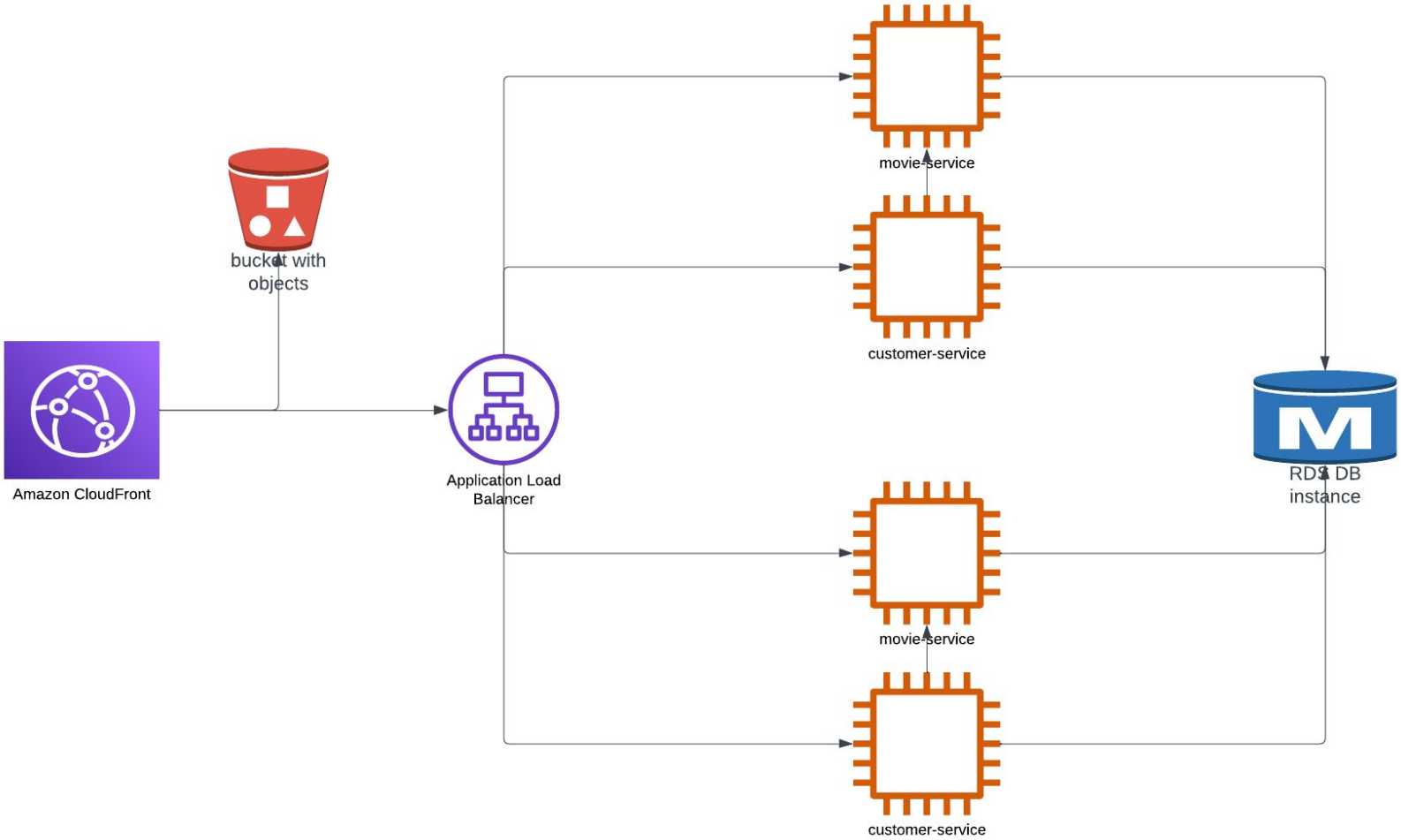
Properties	Description
type	A link to the documentation for the callers to read more about the problem. If it is not provided, “about:blank” is assumed.
title	Human readable summary of the problem
status	HTTP status code
detail	Detailed message specific to the problem
instance	The URI which caused the problem

Customer + Movie Service

- netflux
  - movie-service
    - port: 7070
    - application.properties
  - customer-service
    - port: 8080
    - application.properties
  - postgres
    - docker-compose.yaml
    - init.sql

ECS + Fargate

End To End Infrastructure Creation & Deployment


movie-service
bucket with objects
customer-service
Amazon CloudFront
Application Load Balancer
RDS DB instance
movie-service
customer-service

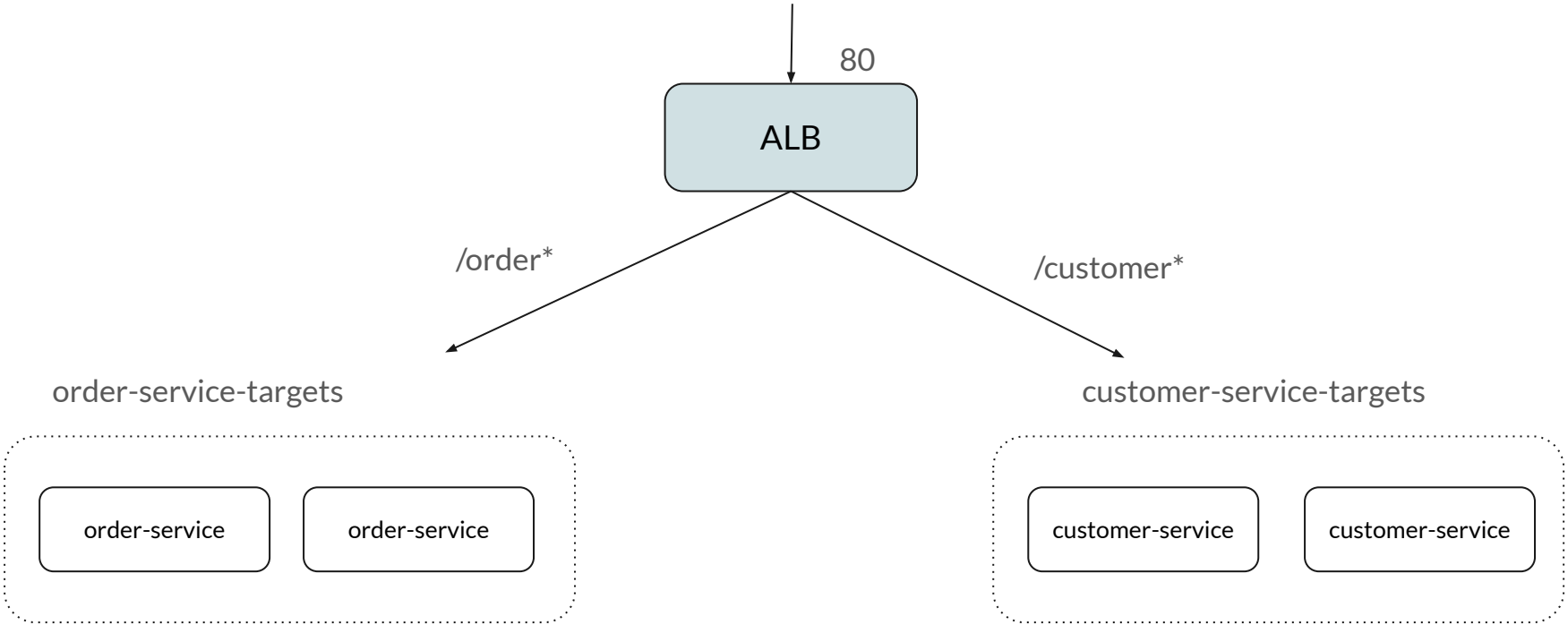
Netflux - VPC

- VPC
  - 2 public subnets
  - 4 private subnets
    - 2 for backend apps
    - 2 for DB
- NAT (Do not create now.Lets do it later)
- SG
  - ALB → APP → DB

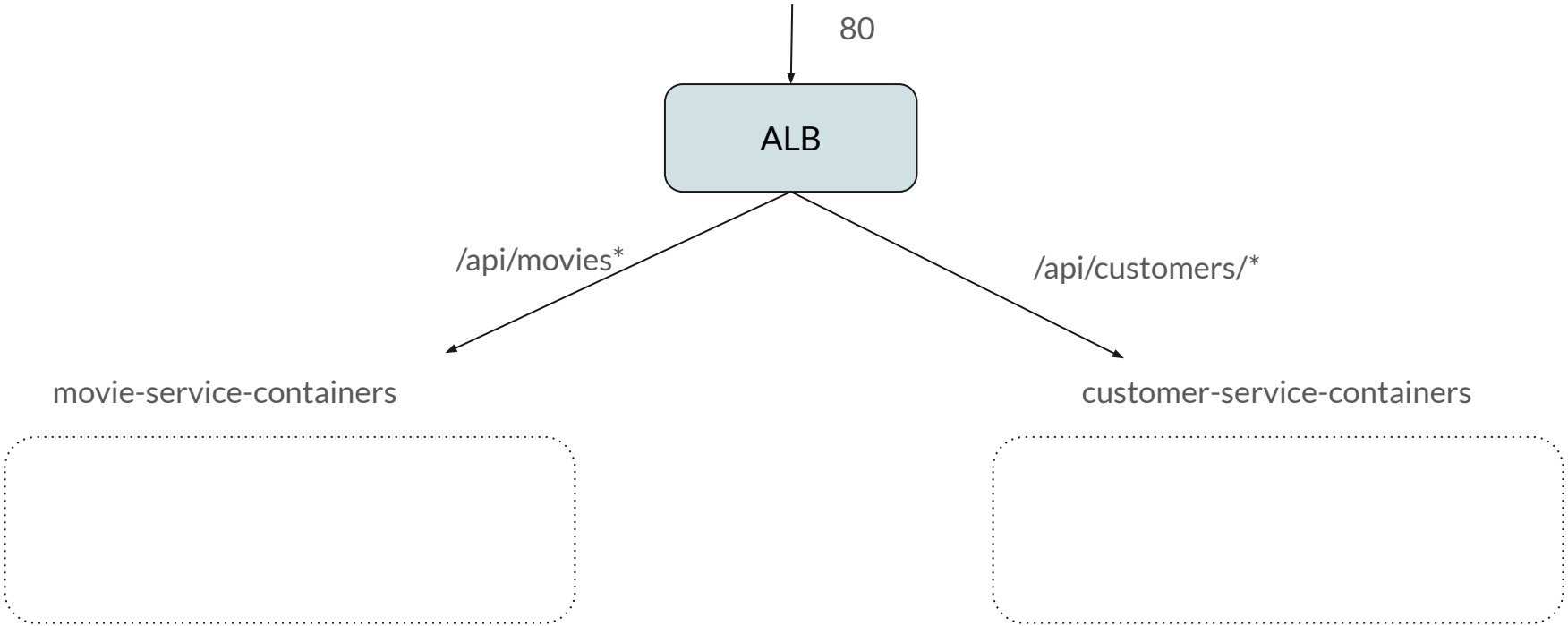
Netflux - ALB & TG

- Target Groups
  - movie-service-containers
  - customer-service-containers
- ALB
  - Listener Rules
    - /api/movies* ⇒ movie-service-containers
    - /api/customers/* ⇒ customer-service-containers

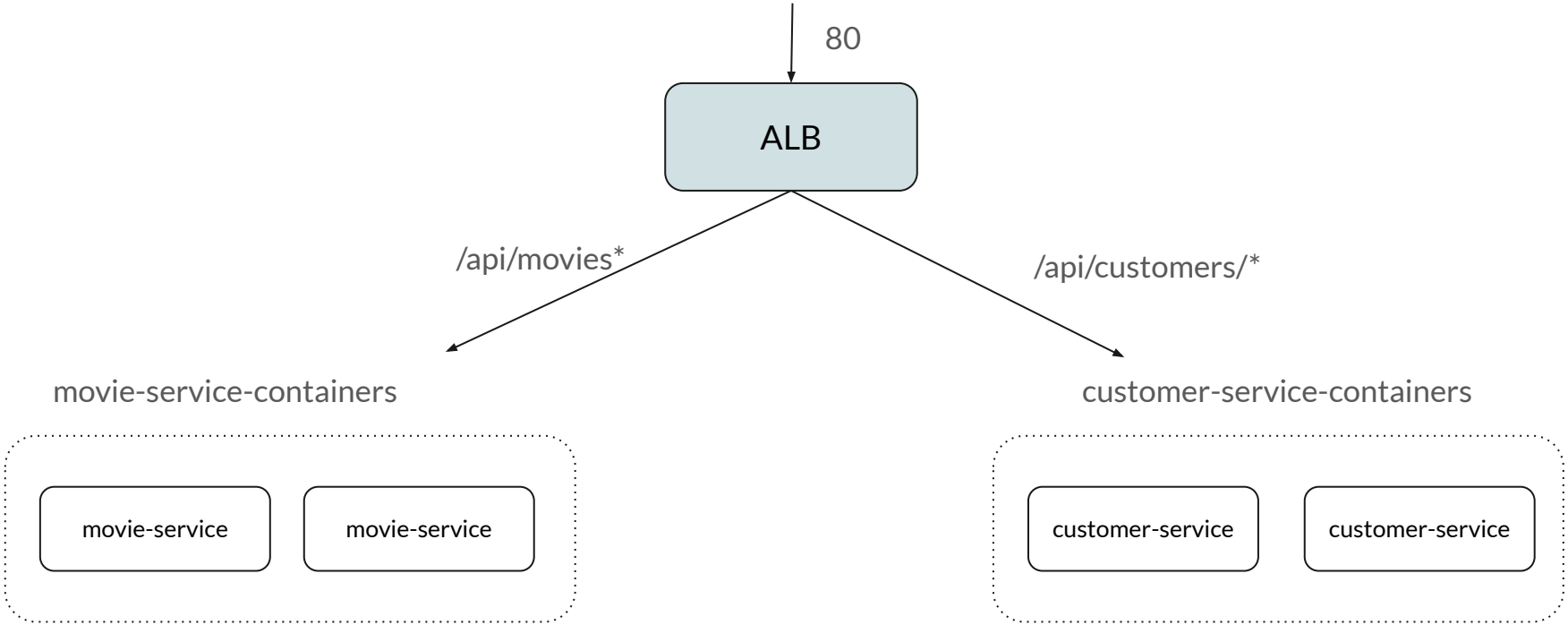
ALB + Target Group



ALB + Target Group



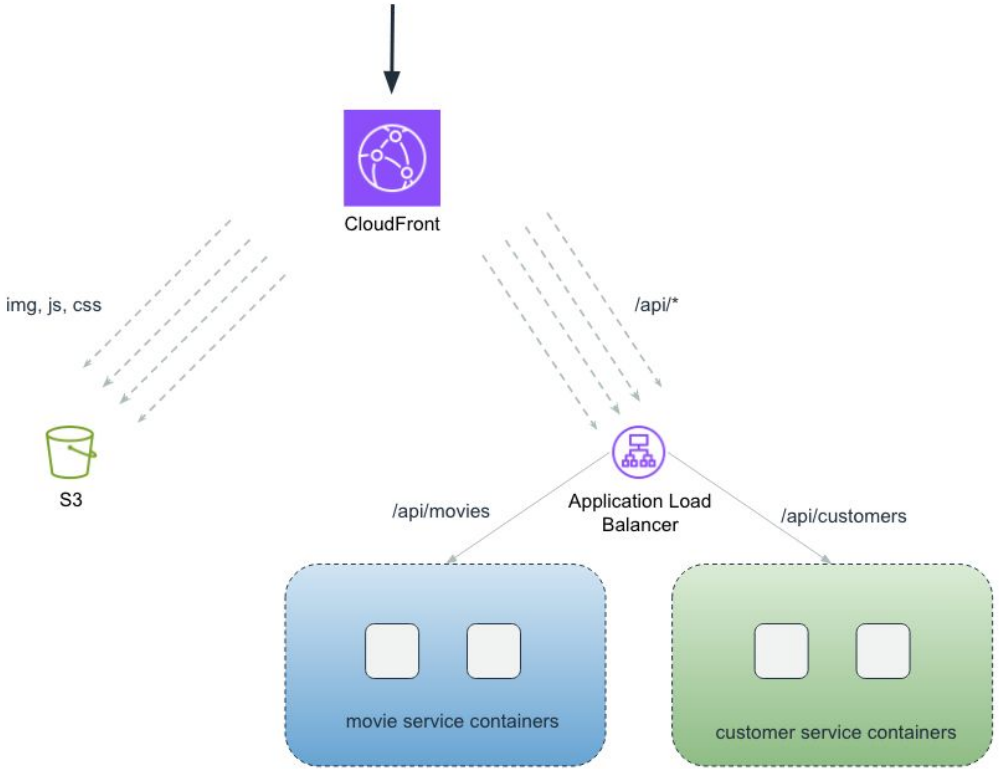
ALB + Target Group



Netflux - S3 + CloudFront

- S3
  - Create a bucket [name-netflux]
  - Upload a dummy index.html
- CloudFront
  - Create a distribution with S3 as origin.
  - Update Bucket policy
  - Add ALB as origin for /api/*
- Test

Netflux - S3 + CloudFront + ALB



Netflux - RDS

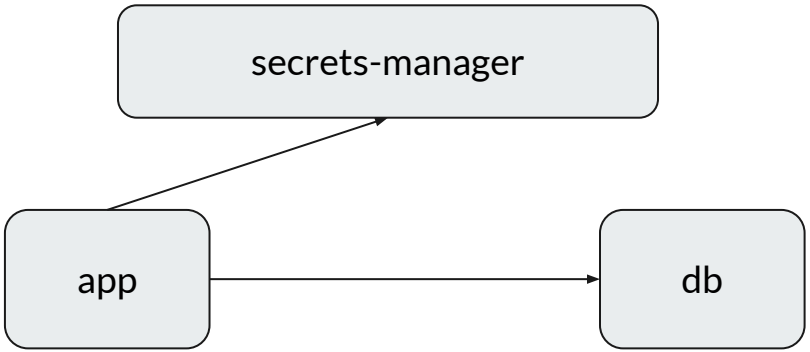
- RDS
  - Create Postgres DB Instance
  - Initialize the DB
- NOTE
  - We will use 1 single DB Instance. We will create 2 databases.
    - movie
    - customer
  - We will NOT use Multi-AZ/Read-Replica in this course to save some cost.

Netflux - RDS

- Production Applications
  - Multiple RDS Instances if you want!
  - ***Multi-AZ***

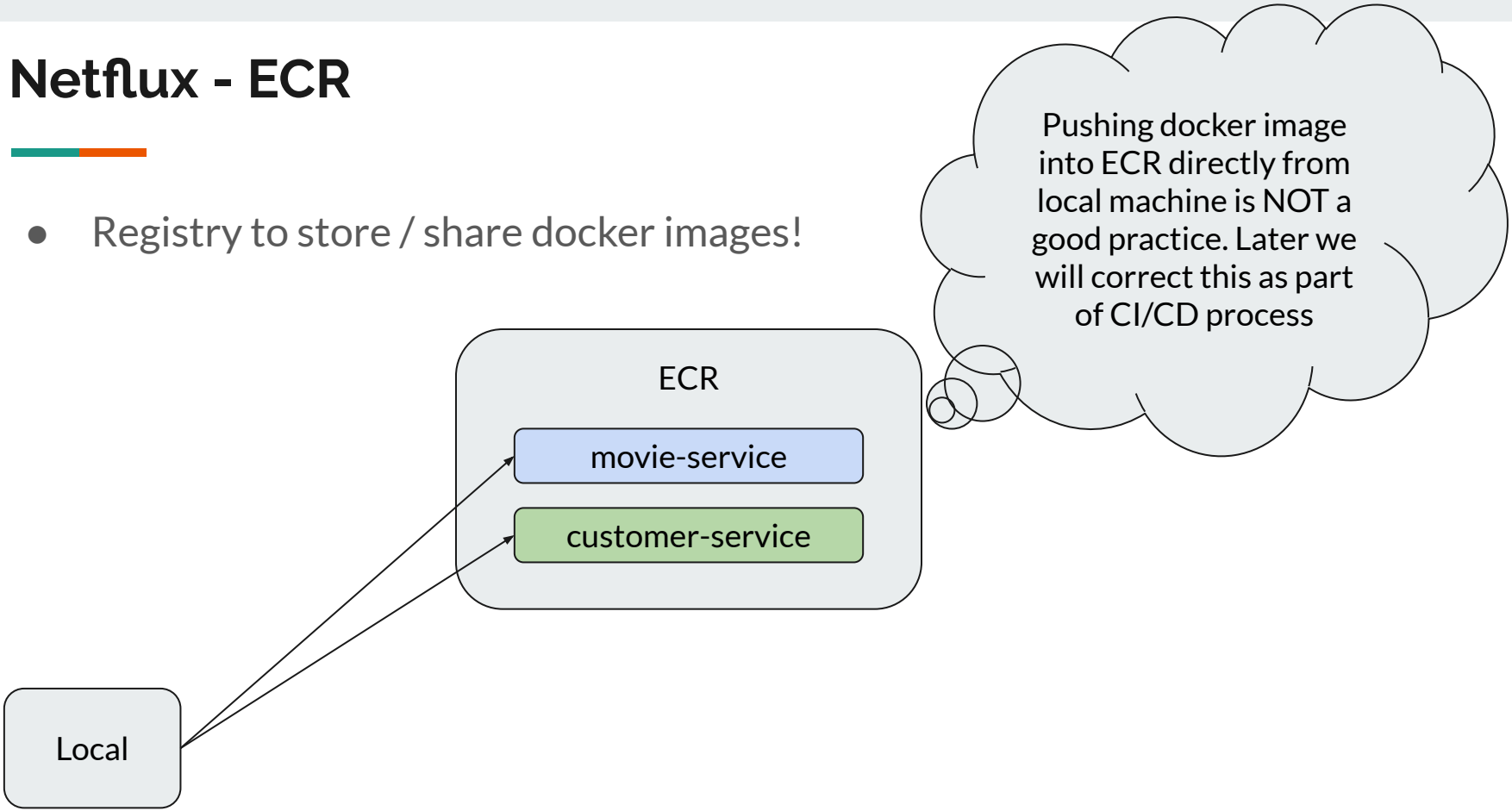
Netflux - Secrets Manager

- To centrally manage the life cycle of the application/DB secrets
  - DB Credentials
  - API Keys
  - TLS Certificates etc

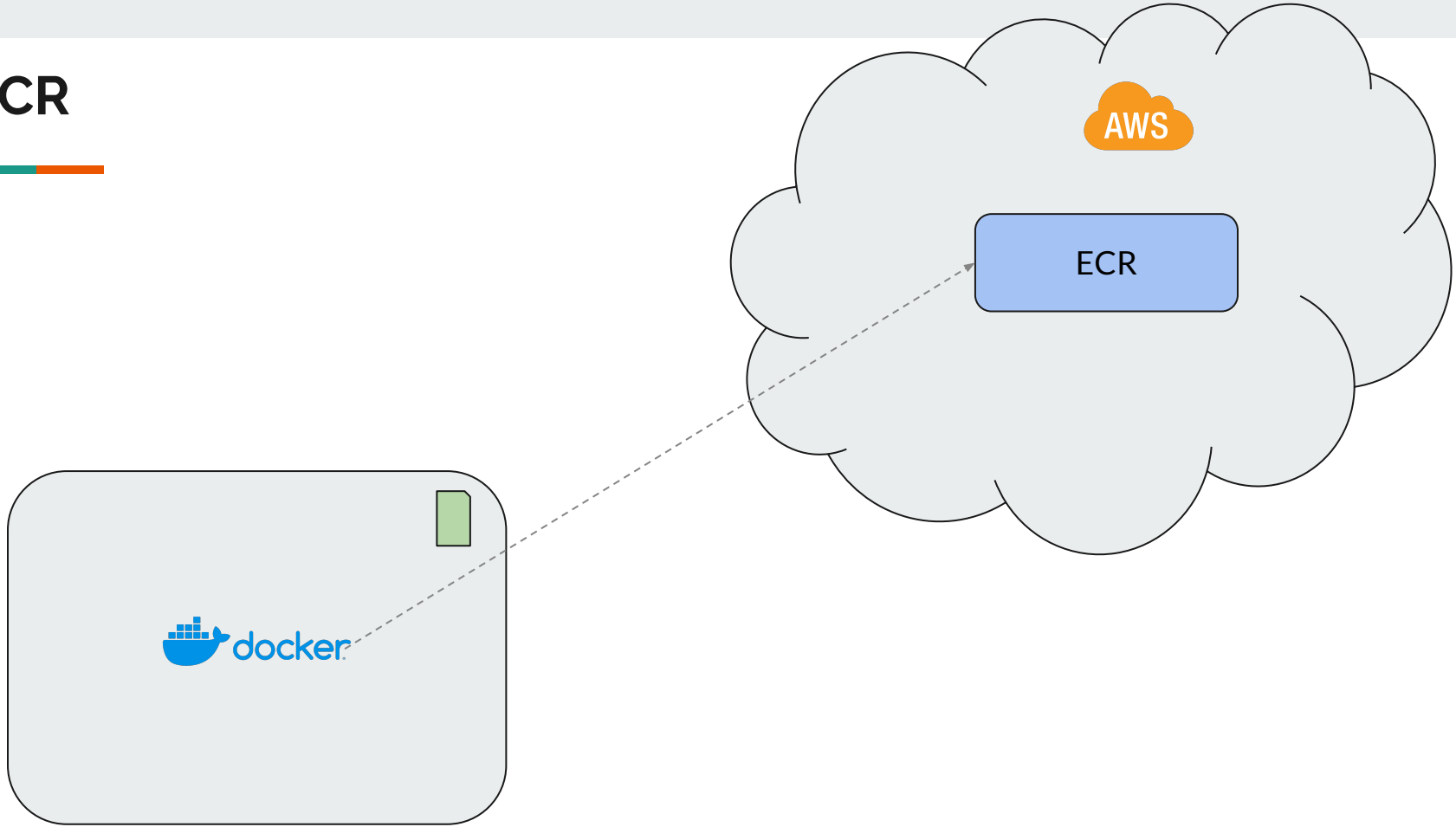


Netflux - ECR

- Registry to store / share docker images!



ECR

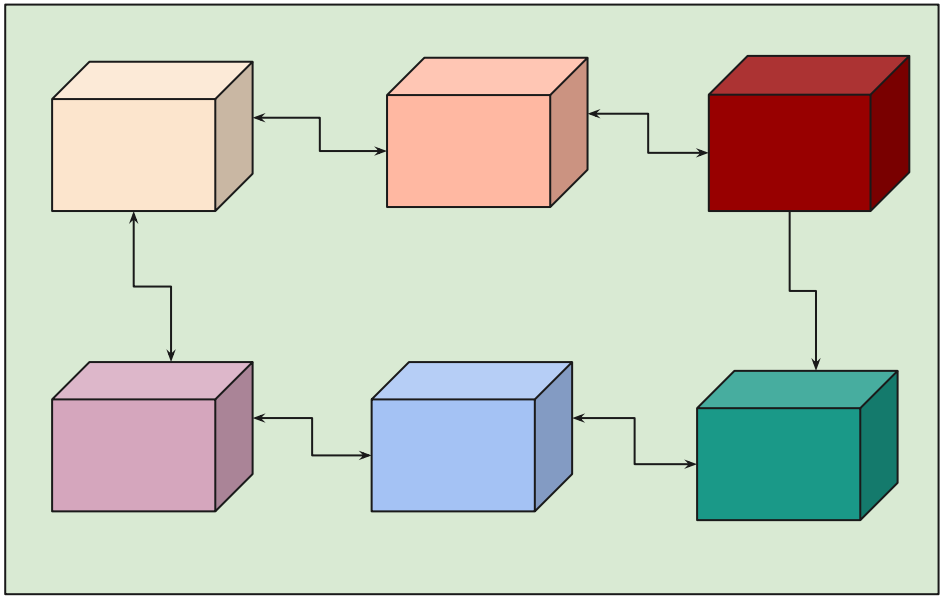


M Series Mac Users / ARM

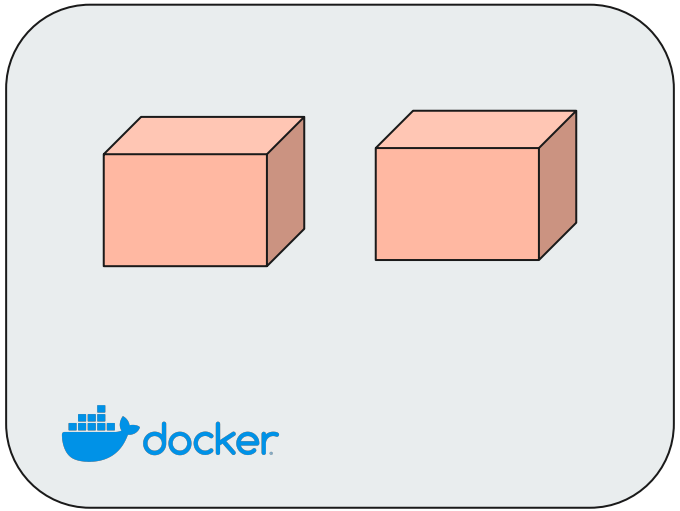
- docker build **--platform=linux/amd64** -t movie-service .

ECS with Fargate!

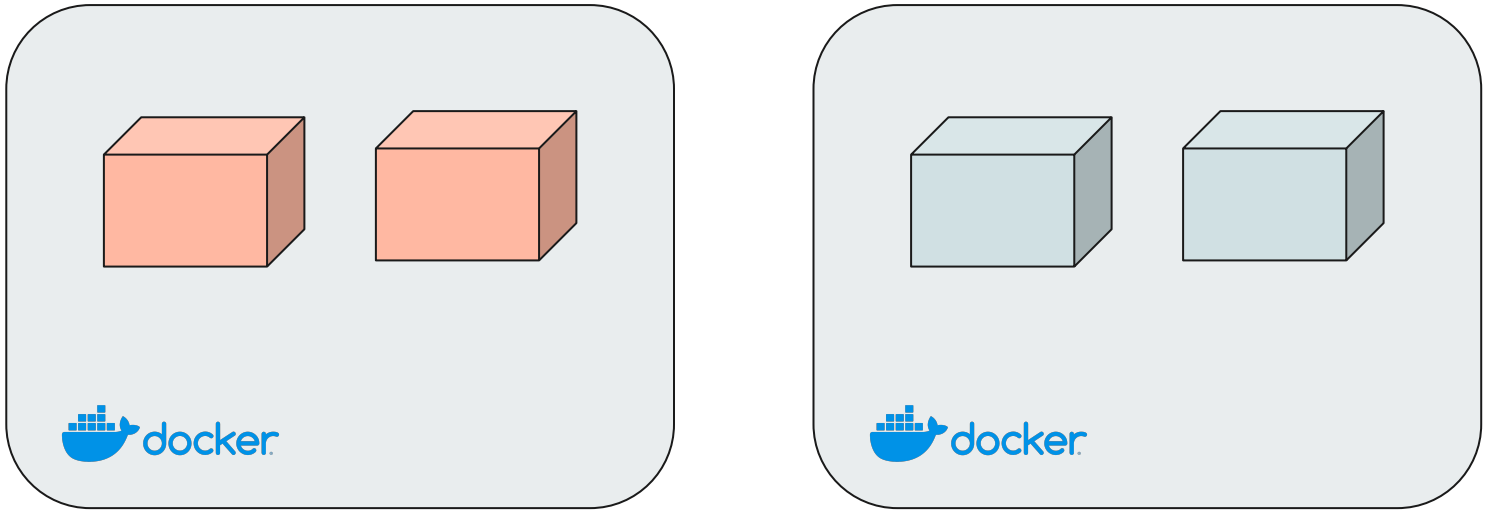
- Elastic Container Service
- Container Orchestrator
- Seamless integration with other AWS services.



ECS



ECS

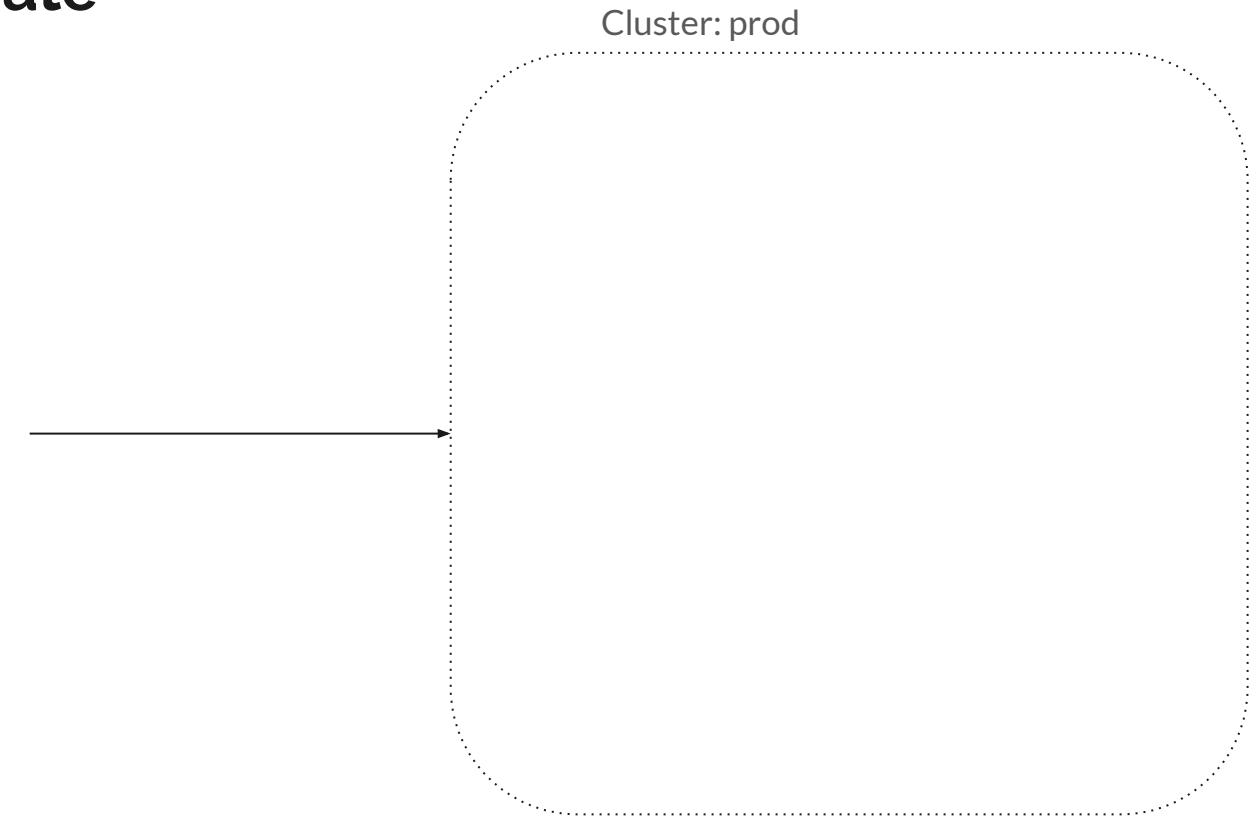


Fargate

- Serverless compute engine for containers!
- No need to worry about the underlying infrastructure!

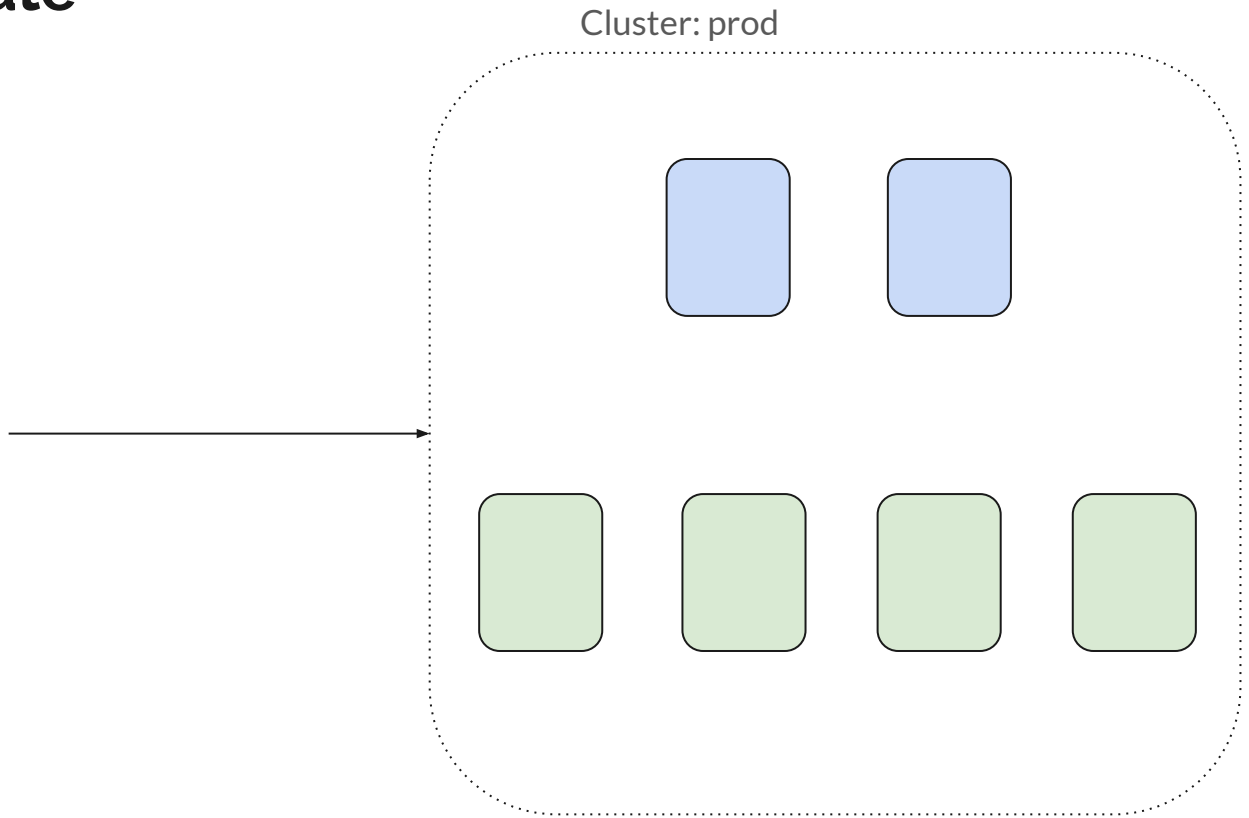
ECS with Fargate

2 movie-service
4 customer-service



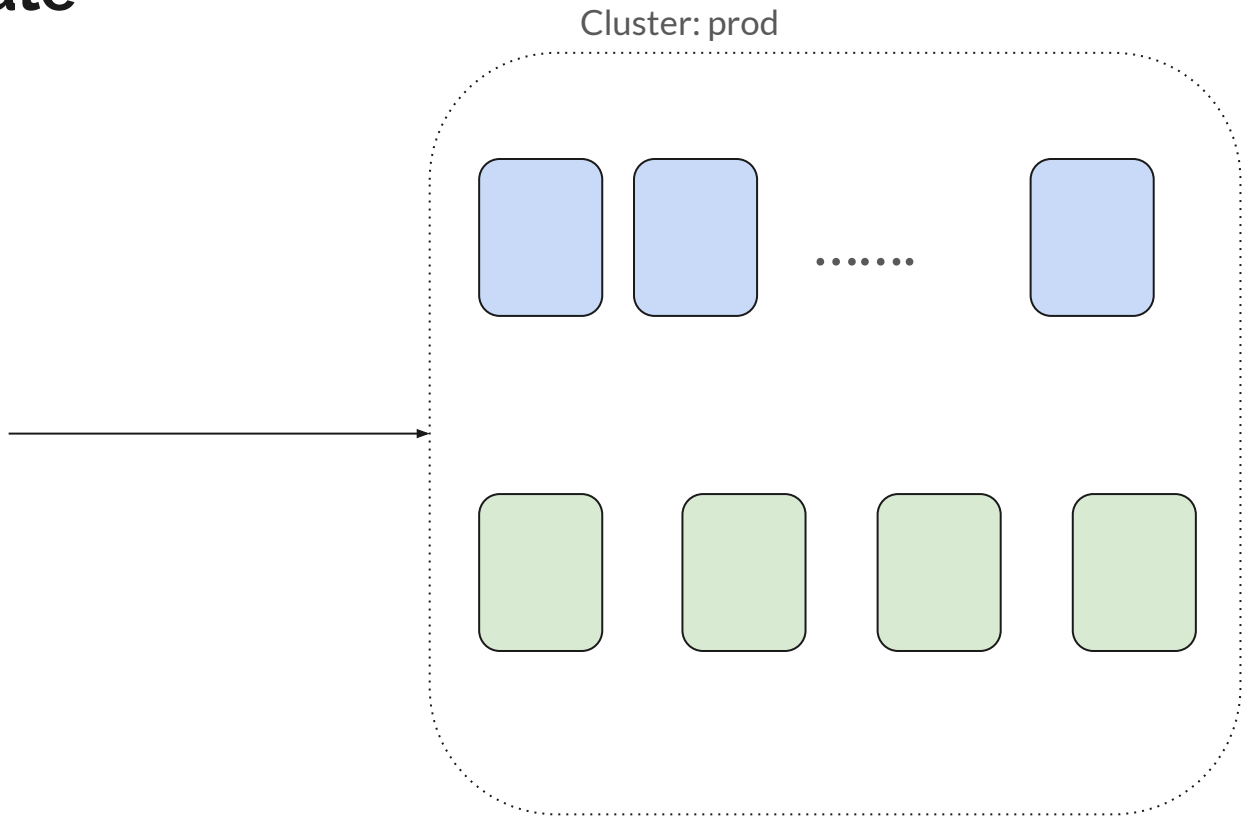
ECS with Fargate

2 movie-service
4 customer-service



ECS with Fargate

2…20 movie-service
4 customer-service



ECS + Fargate

- *Task Definition ⇒ Template*
  - docker image name/version
  - CPU / Memory
  - Port
  - Environment Variables
  - Disk Storage!
- 1 Microservice ⇒ 1 Task Definition

ECS + Fargate

- *Task*
  - actual running instances of those containers defined in the Task definition.

ECS + Fargate

- *2 IAM Roles*
- Task Role
  - Your application might want to talk another AWS service.
    - S3
    - Secrets Manager
    - ...
- Task Execution Role
  - AWS creates itself! This role is required to pull the docker images from our ECR ..etc on behalf of us!

How To Deploy?

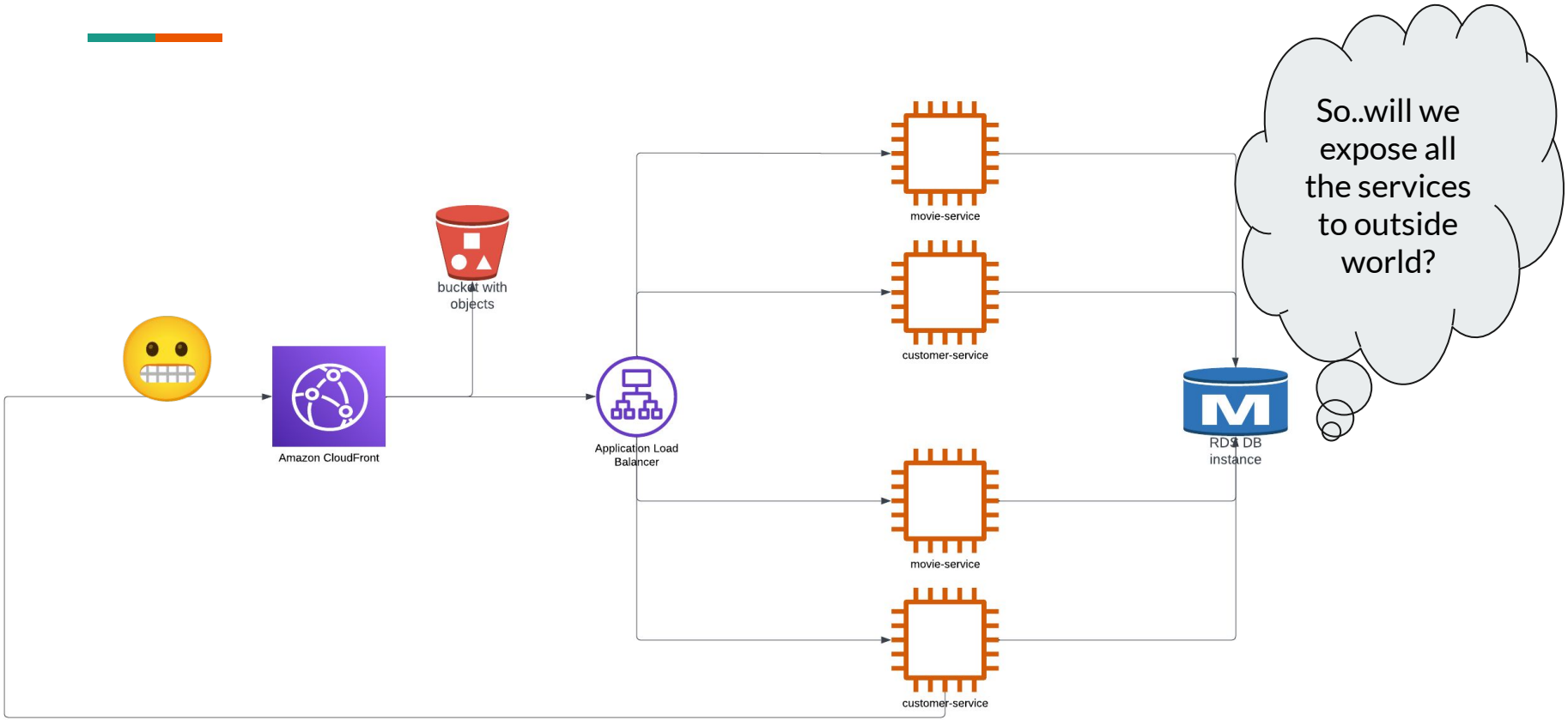
- Run forever / batch job?
- How many instances?
- Deployment type
- Interservices communication
- VPC/Subnets/Security Groups?
- Target Groups?

Service ⇒ Service Communication

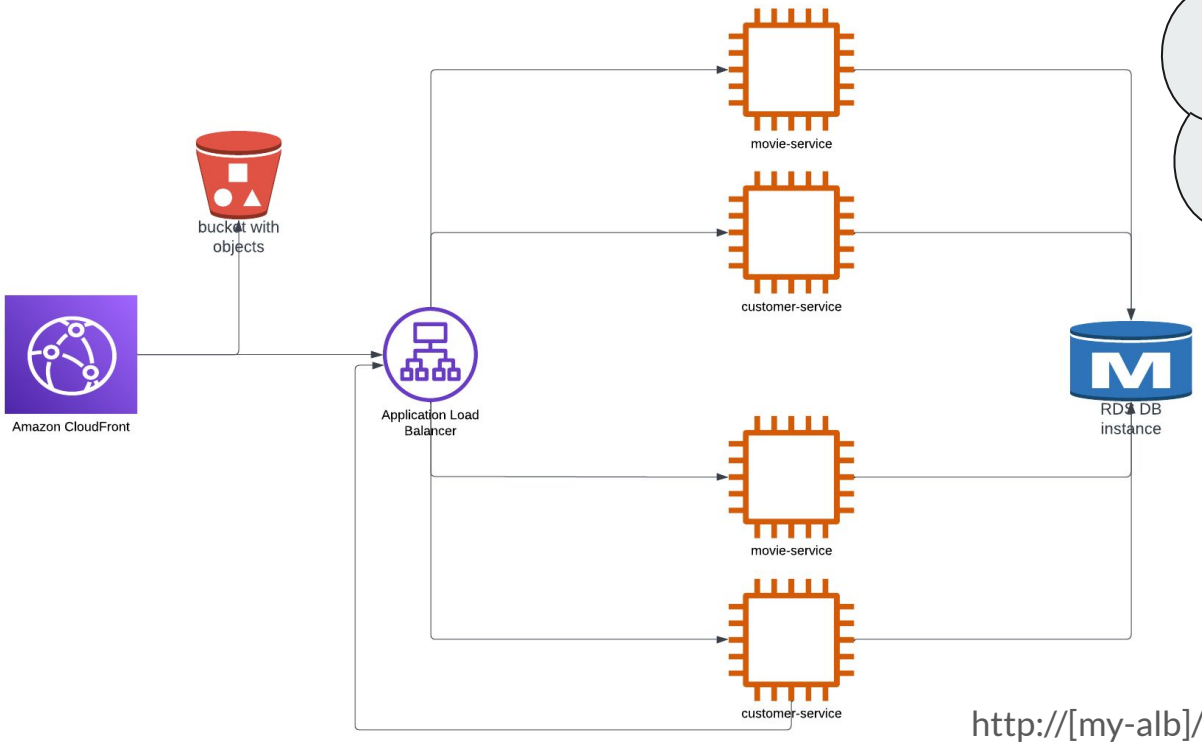
- If ***netflux.com*** is our domain,
  - ***https://netflux.com/api/movies/ACTION*** should return the movies for the given genre. !!?

So, can we use this?
movie.service.url=https://netflux.com

Service ⇒ Service Communication via Internet!!



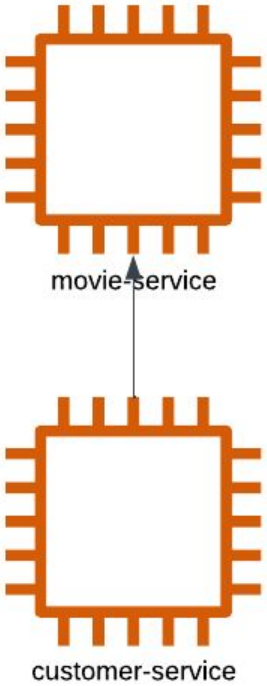
How About ALB?



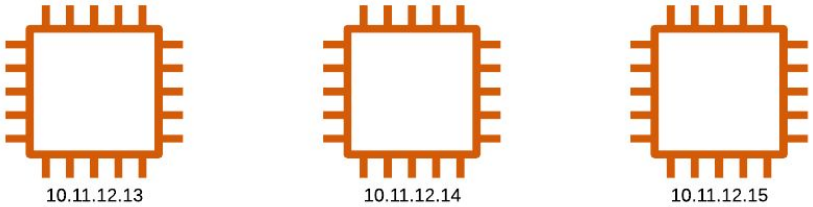
We might not need DDoS protection within VPC.
We can consider internal ALB.
Look at the pricing for ALB as well.

http://[my-alb]/api/movies/ACTION

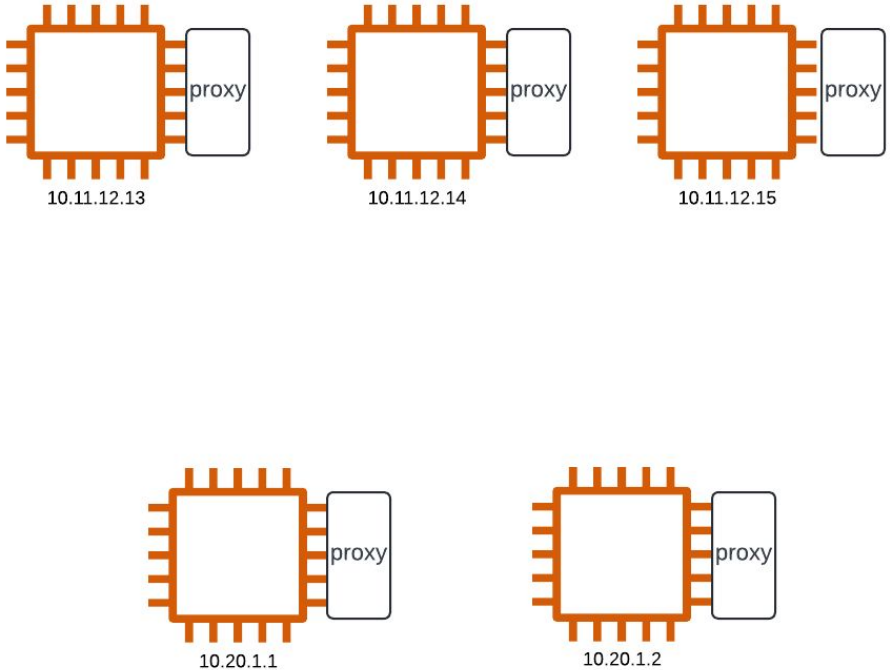
Service ⇒ Service Communication



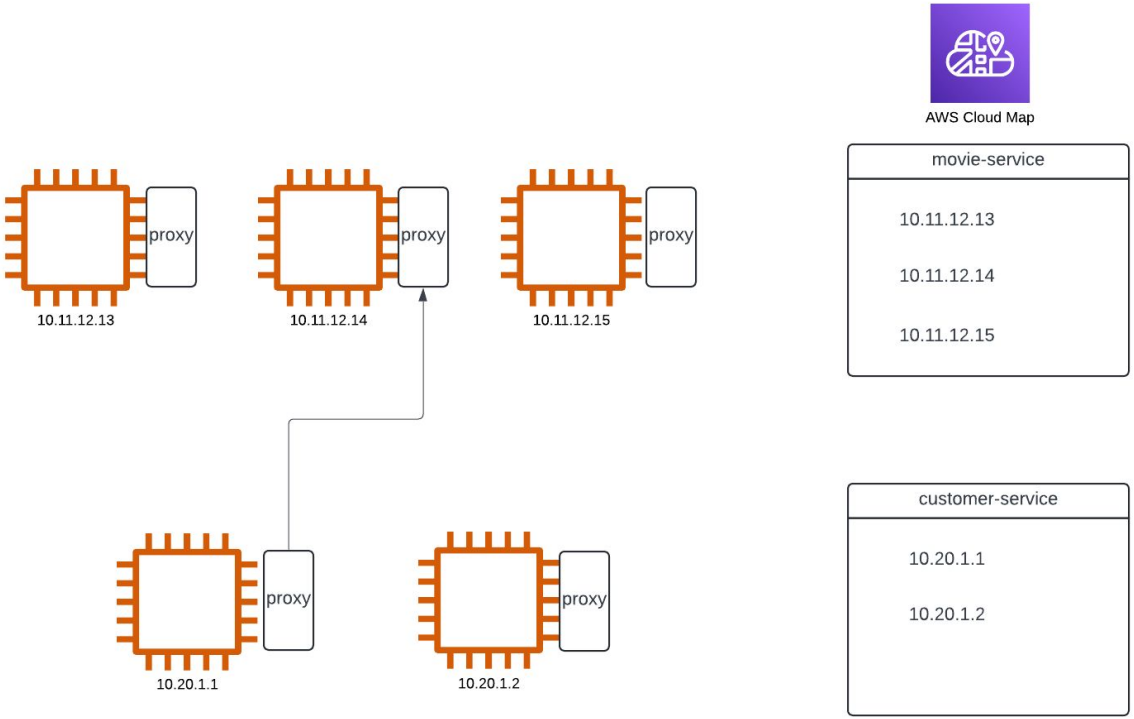
Service ⇒ Service Communication



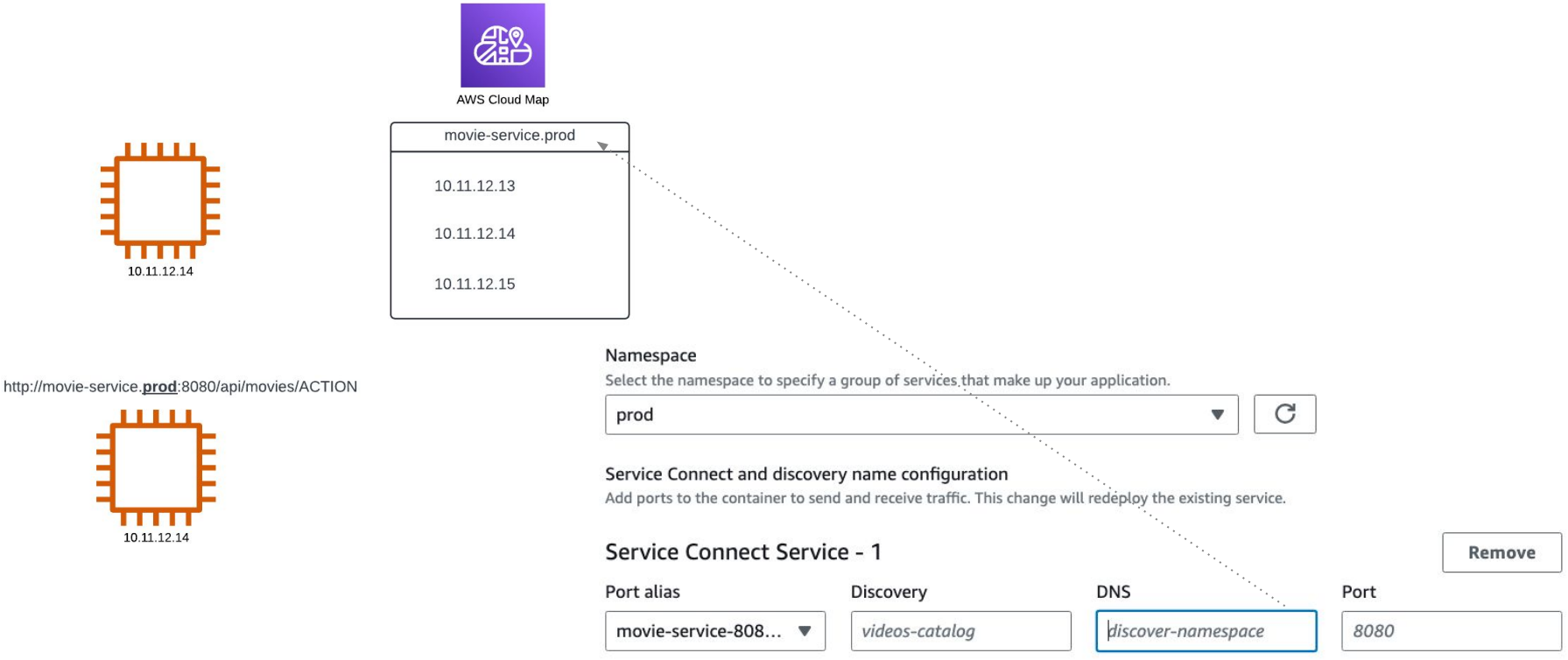
Service ⇒ Service Communication



Service ⇒ Service Communication

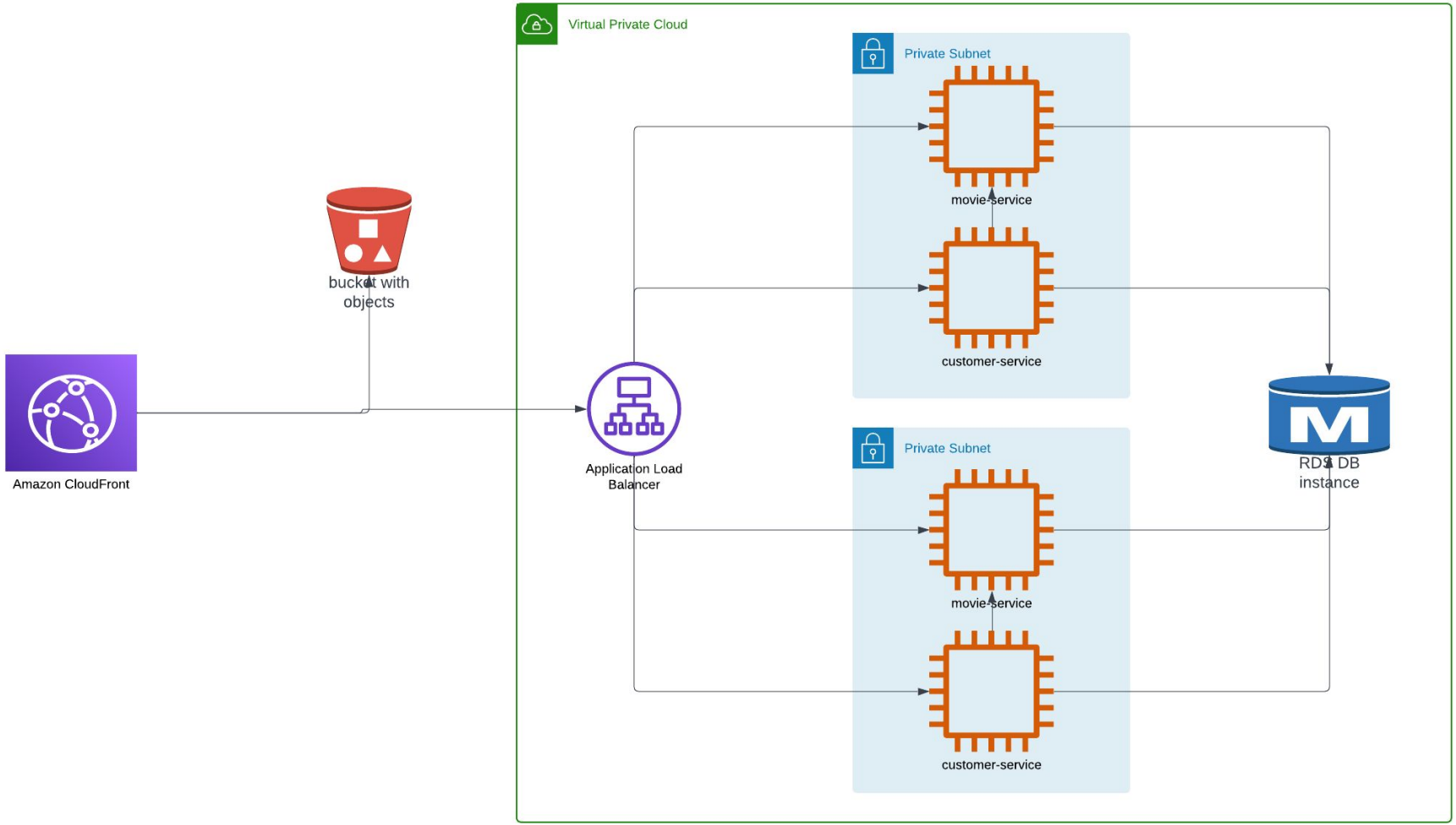


Service ⇒ Service Communication



Summary

- Deep understanding of the Infrastructure
  - VPC + Security Groups
  - ALB + Target Groups + Listener Rules
  - S3 + CDN
  - RDS + Secrets Manager
  - ECR
  - ECS + F*argate*
  - Service Connect
- Creating a reliable, scalable and secure Infrastructure


Virtual Private Cloud
Private Subnet
movie-service
customer-service
bucket with objects
Amazon CloudFront
Application Load Balancer
Private Subnet
movie-service
customer-service
RDS DB instance

Auto Scaling

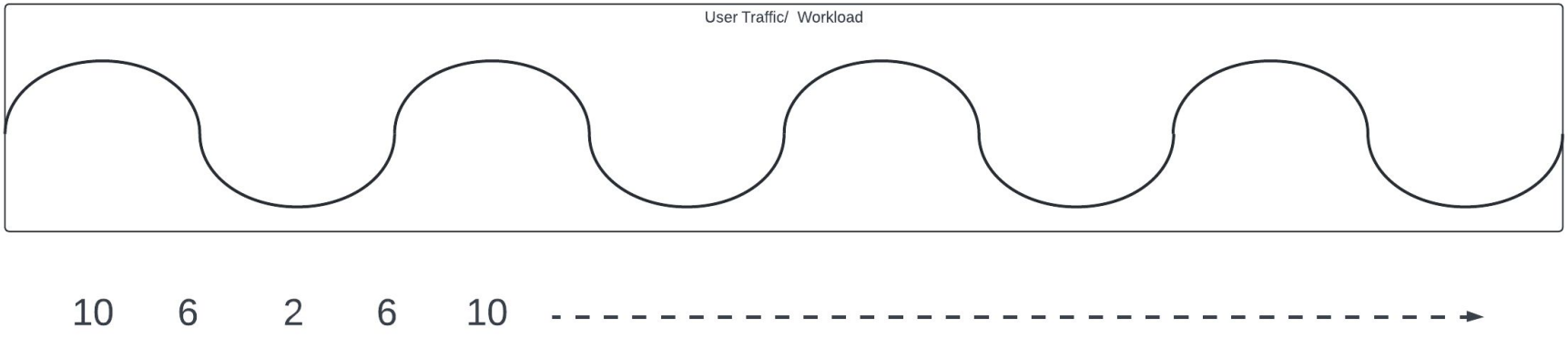
Let Cloud Scale Seamlessly

Need For Auto Scaling

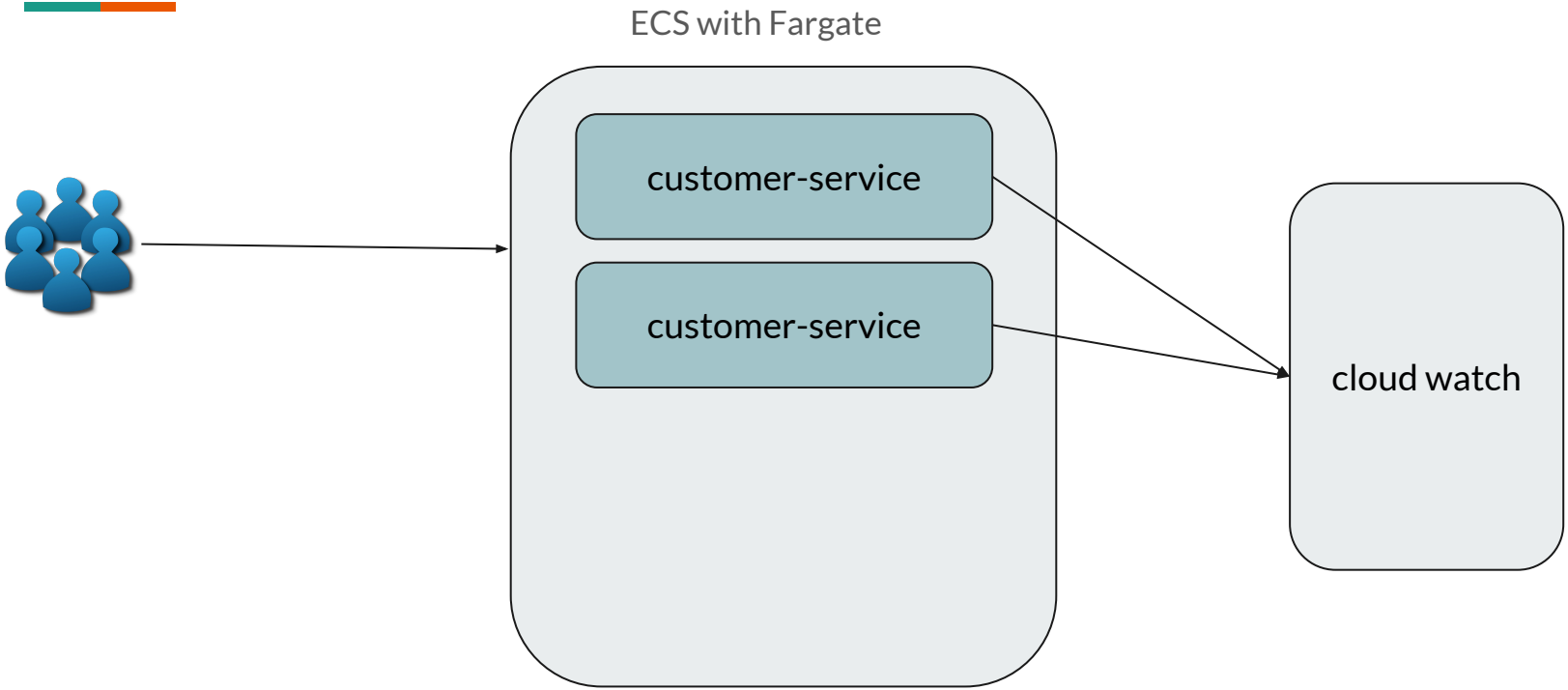
- Unpredictable demand
- Resource Inefficiency
  - Over provisioning / Under provisioning
- Operational Complexity
  - Labor intensive / error prone
- Reduced Application Availability
- Business Agility

Auto Scaling

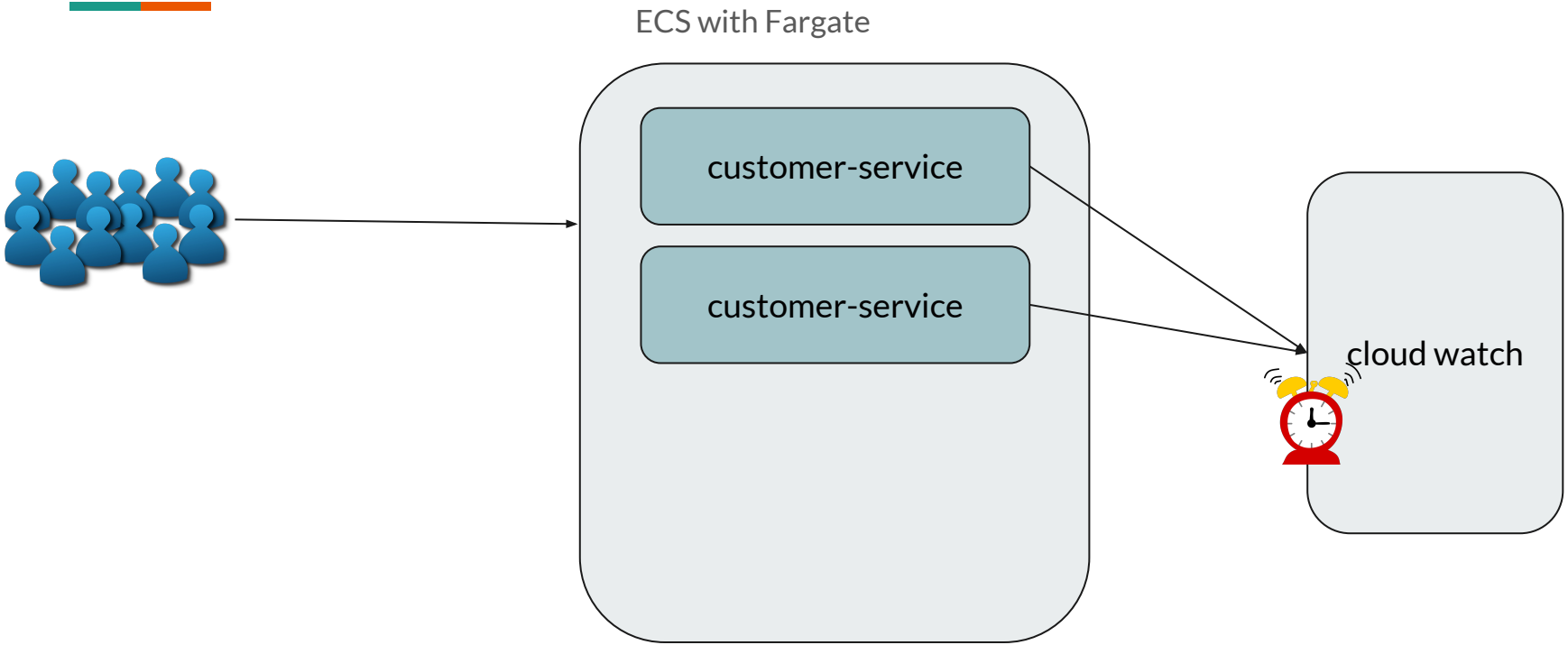
- Automated resource adjustment!
  - Based on CPU / Memory / incoming traffic
- Scale up and down based on demand.



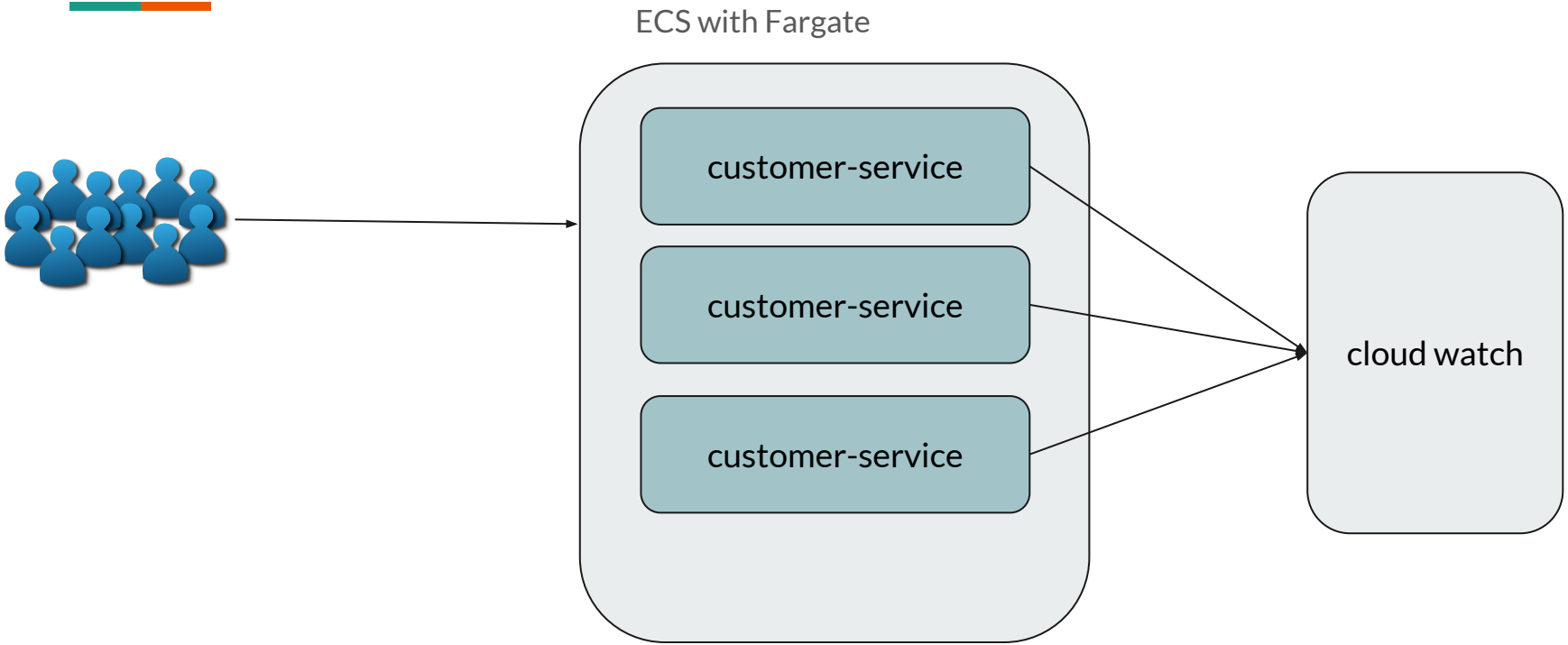
Cloud Watch


ECS with Fargate
customer-service
customer-service
cloud watch

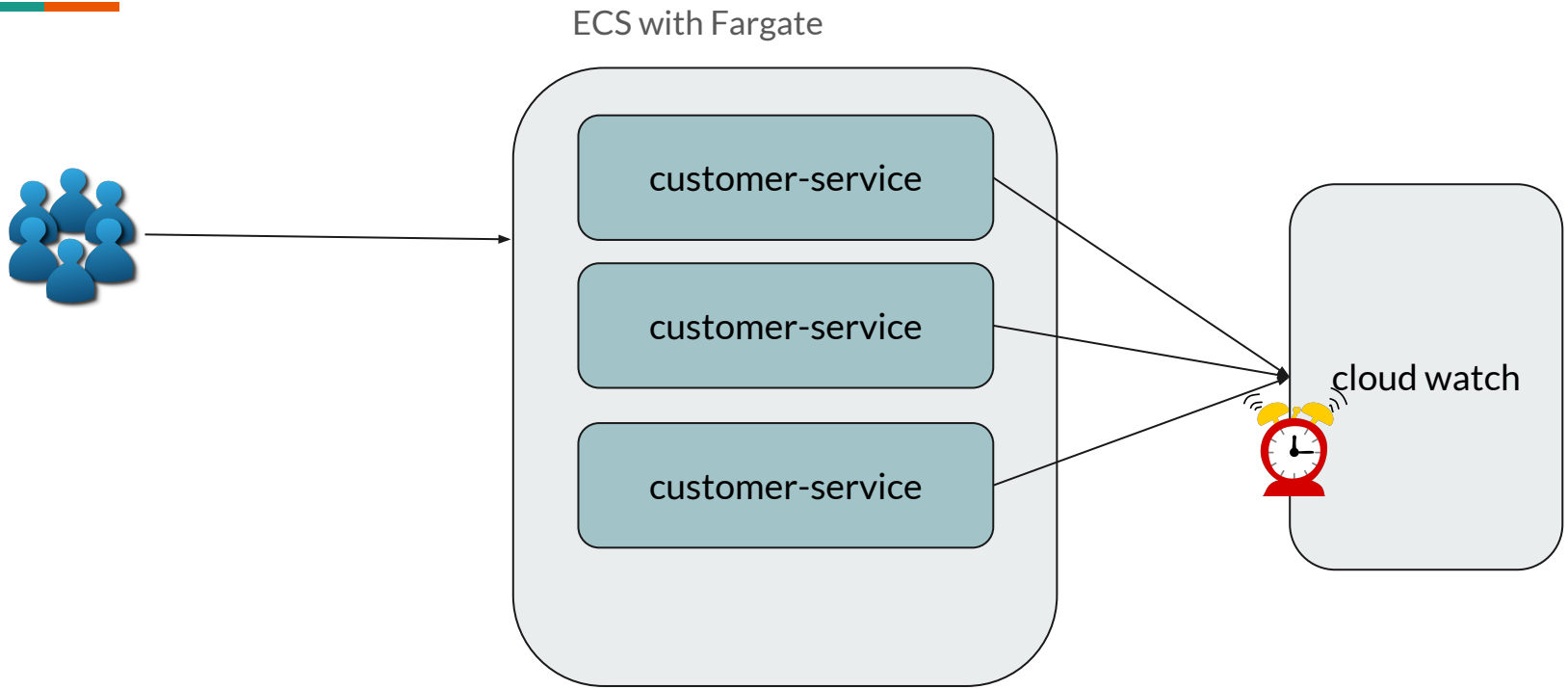
Cloud Watch


ECS with Fargate
customer-service
customer-service
cloud watch

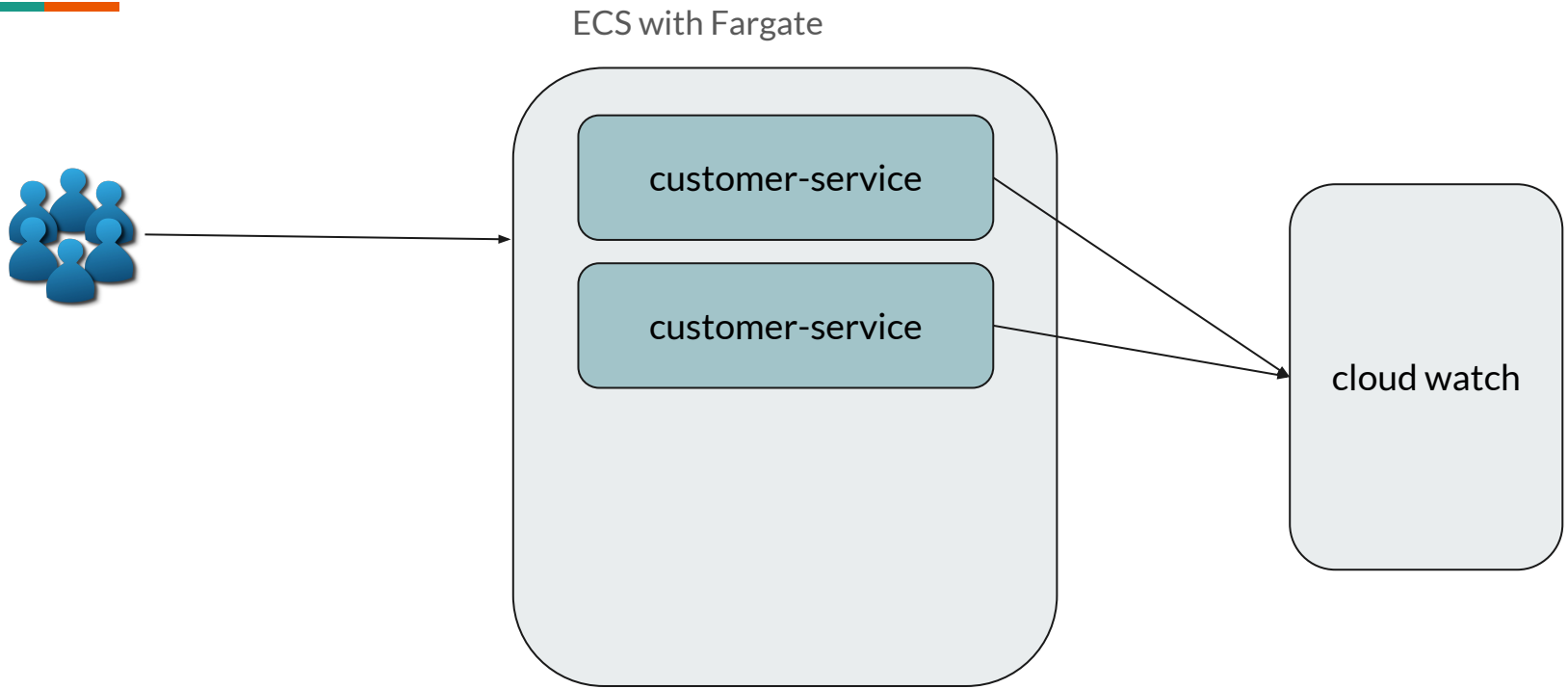
Cloud Watch


ECS with Fargate
customer-service
customer-service
customer-service
cloud watch

Cloud Watch


ECS with Fargate
customer-service
customer-service
customer-service
cloud watch

Cloud Watch


ECS with Fargate
customer-service
customer-service
cloud watch

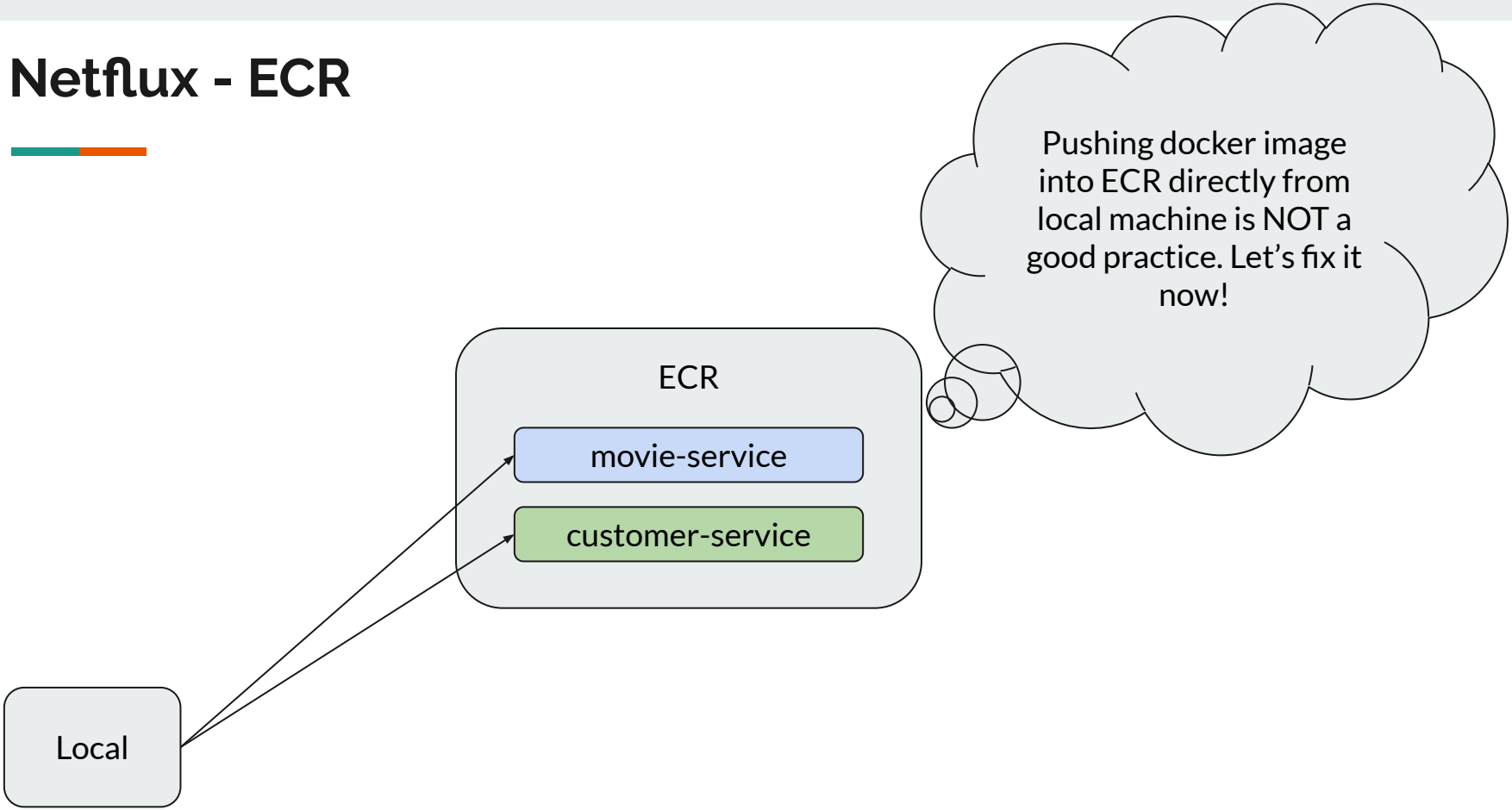
Auto Scaling

- Consistent Application Performance
  - Scaling resources to match varying workloads
- Cost Efficiency
  - Pay per usage
- High Availability
  - Reducing downtime risks
- Operational Efficiency
  - Automated!
- Flexibility
  - Adapts to changing business requirements/market conditions

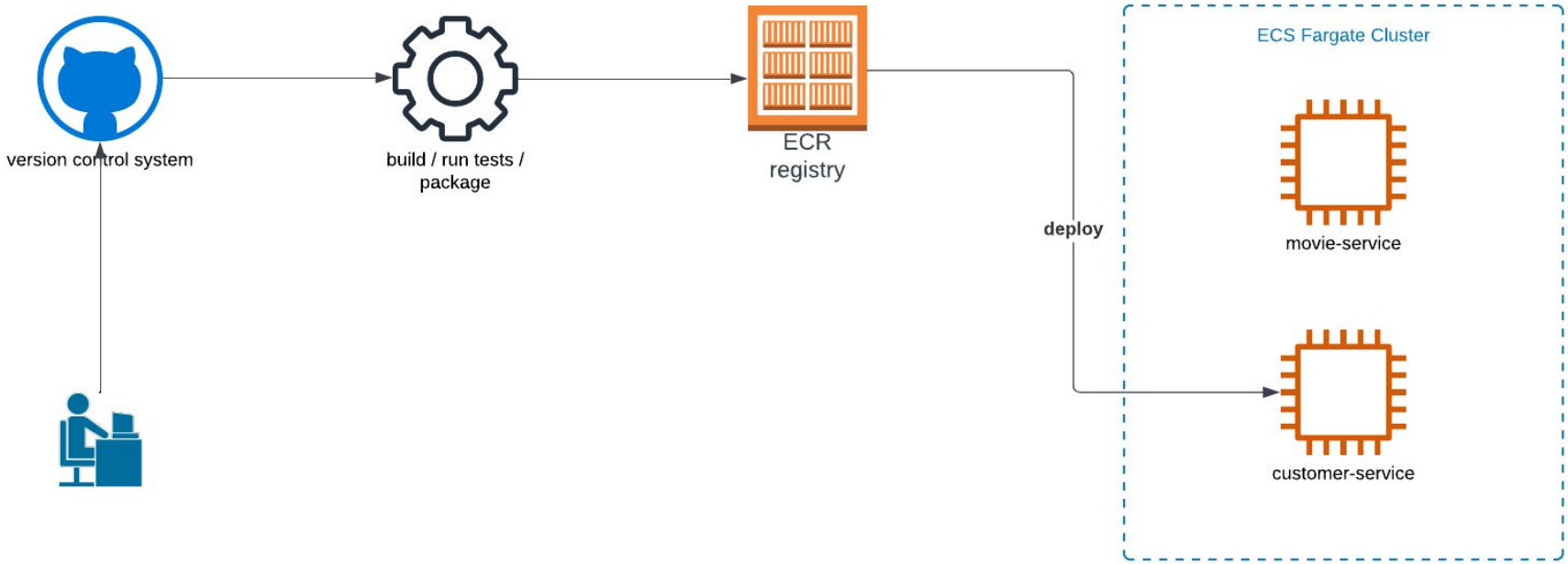
CI/CD

Continuous Integration & Deployment

Netflux - ECR

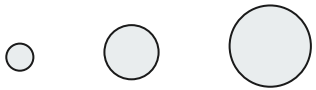

Pushing docker image into ECR directly from local machine is NOT a good practice. Let's fix it now!
ECR
movie-service
customer-service
Local

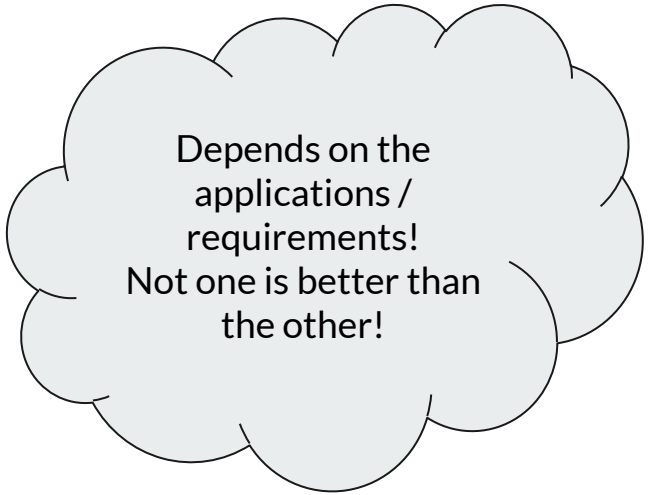
CI/CD Process



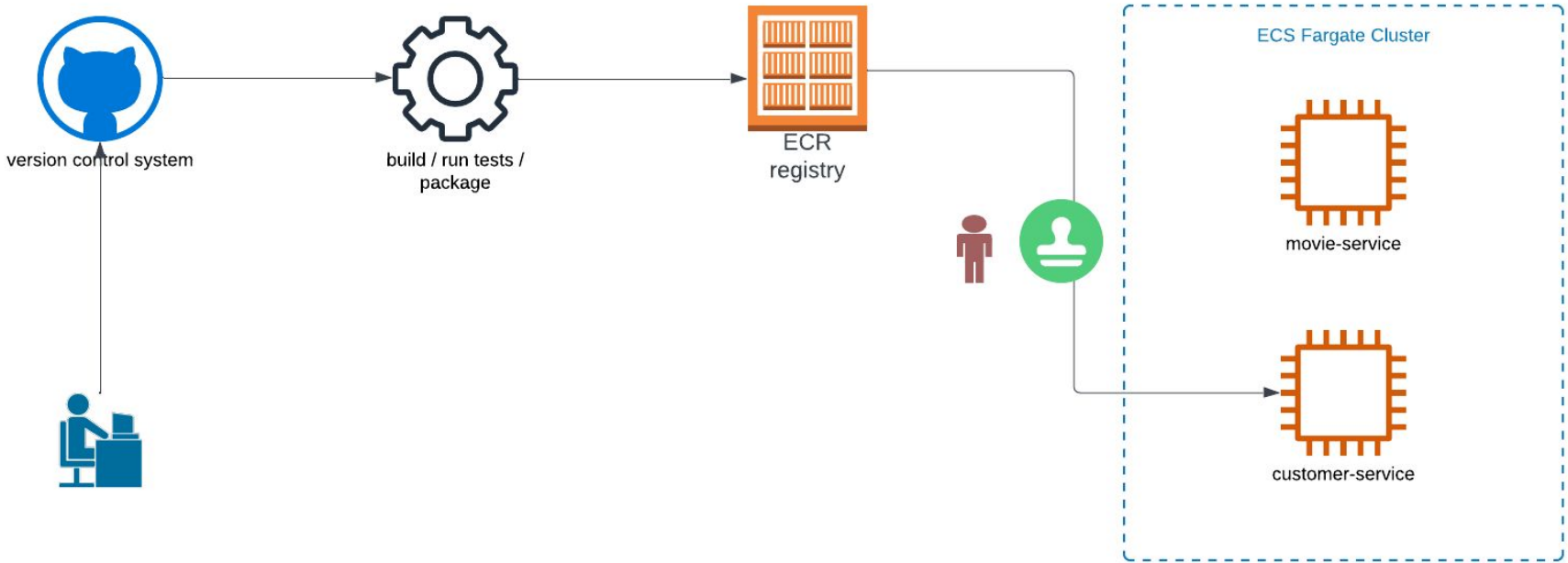
CD

- Continuous Delivery
- Continuous Deployment

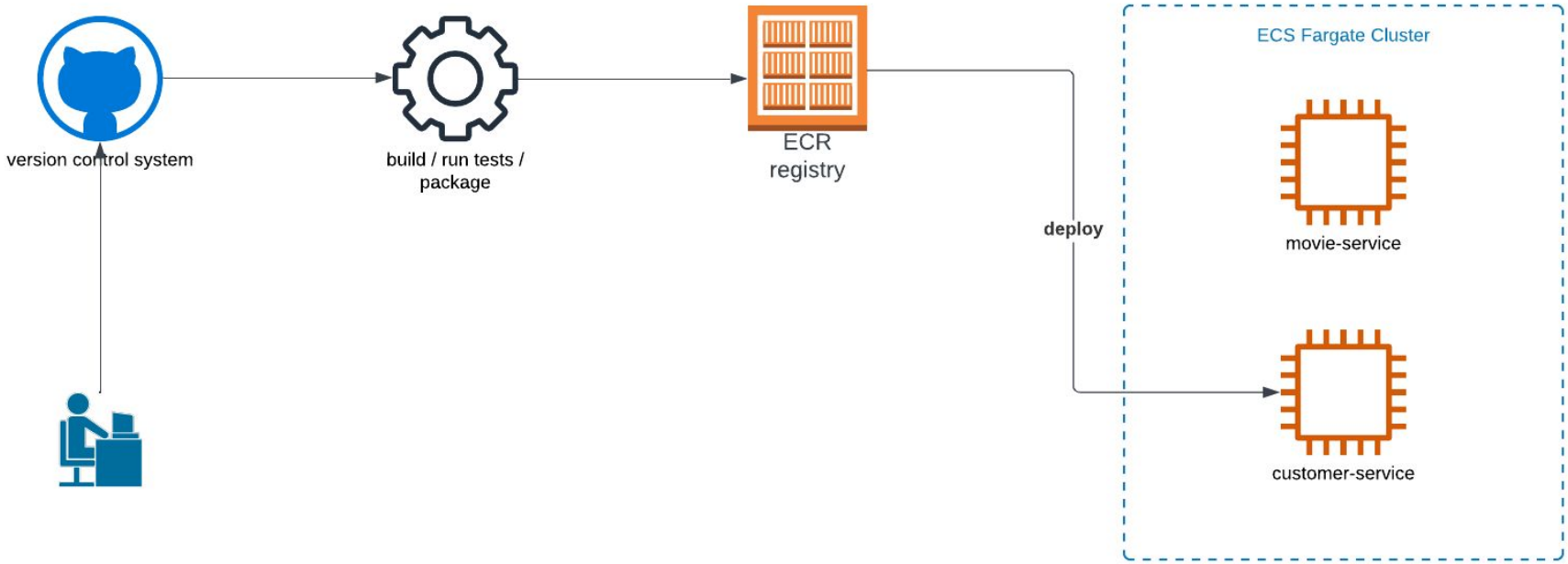




CI/Continuous Delivery

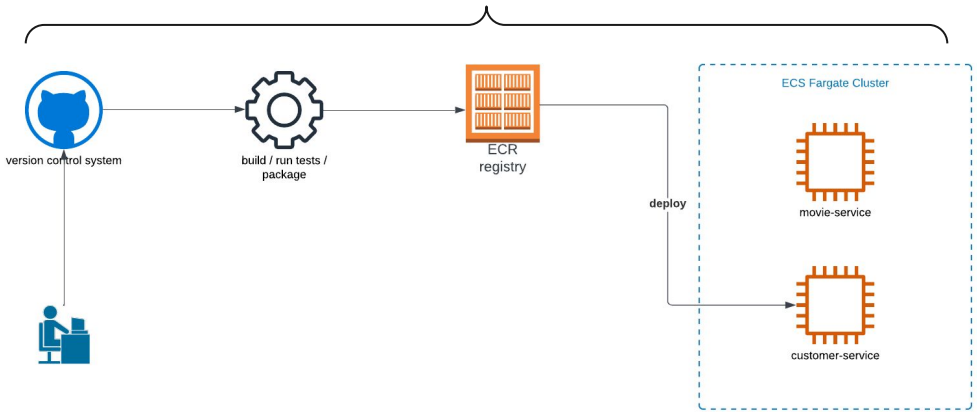


CI/Continuous Deployment



AWS Services

- ***Code Commit***
  - AWS's GitHub/BitBucket
- ***Code Build***
  - Code ⇒ Test ⇒ Package
  - Push to ECR
- ***Code Pipeline***
  - An Orchestrator

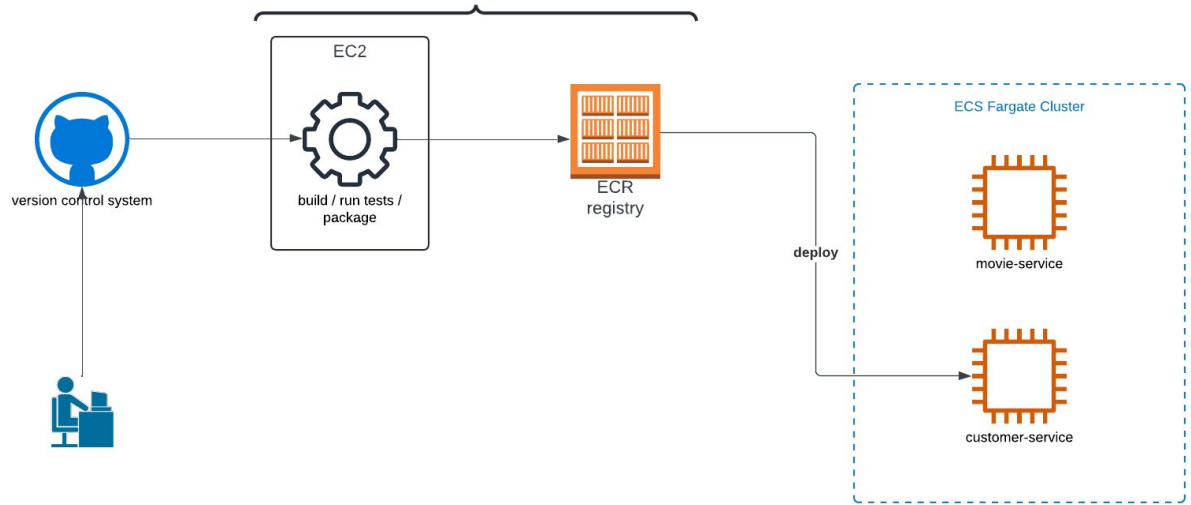


Code Commit

- Create Repository
  - ***customer-service***
  - ***movie-service***
- Developer should be given the ***permission*** to push code to CodeCommit

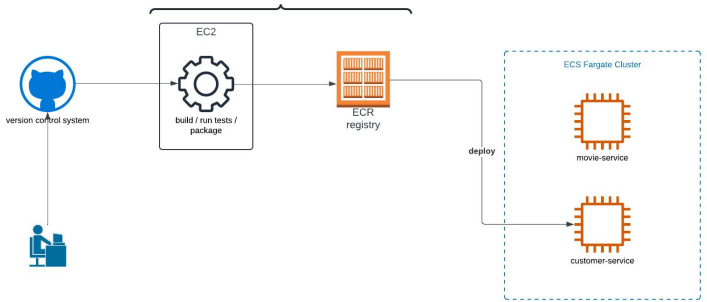
Code Build

- Continuous Integration service



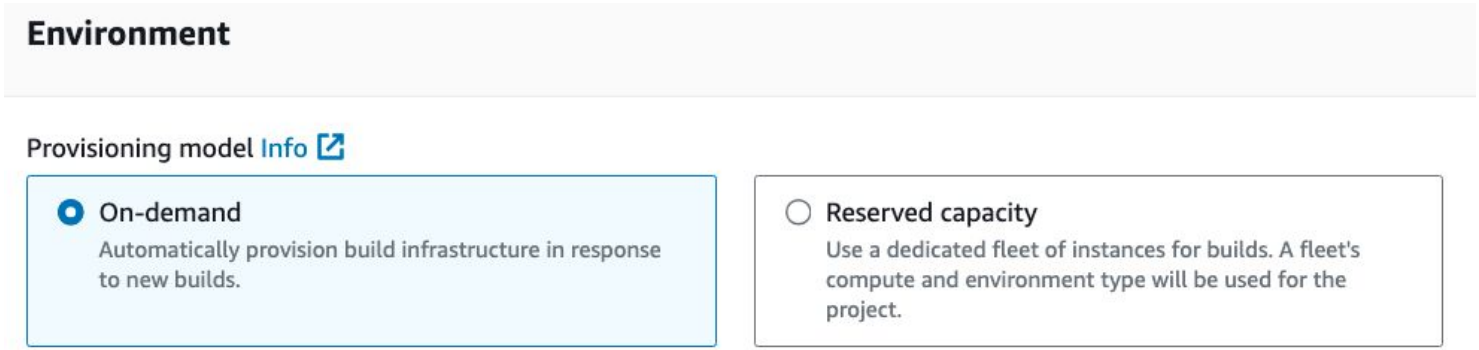
Code Build

```
// build jar
mvn clean package

// docker login / build and push
// region, account number etc can be parameterized

aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 941077029185.dkr.ecr.us-east-
1.amazonaws.com

// build
// docker image name, account number. version can be commit hash
docker tag customer-service:latest 941077029185.dkr.ecr.us-east-1.amazonaws.com/customer-service:[VERSION]
```

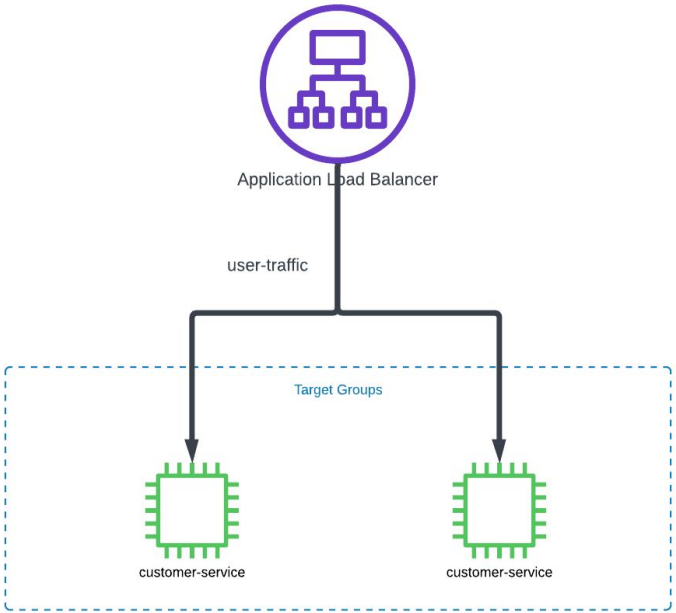


FAQ: Maven Cache / Local Repository

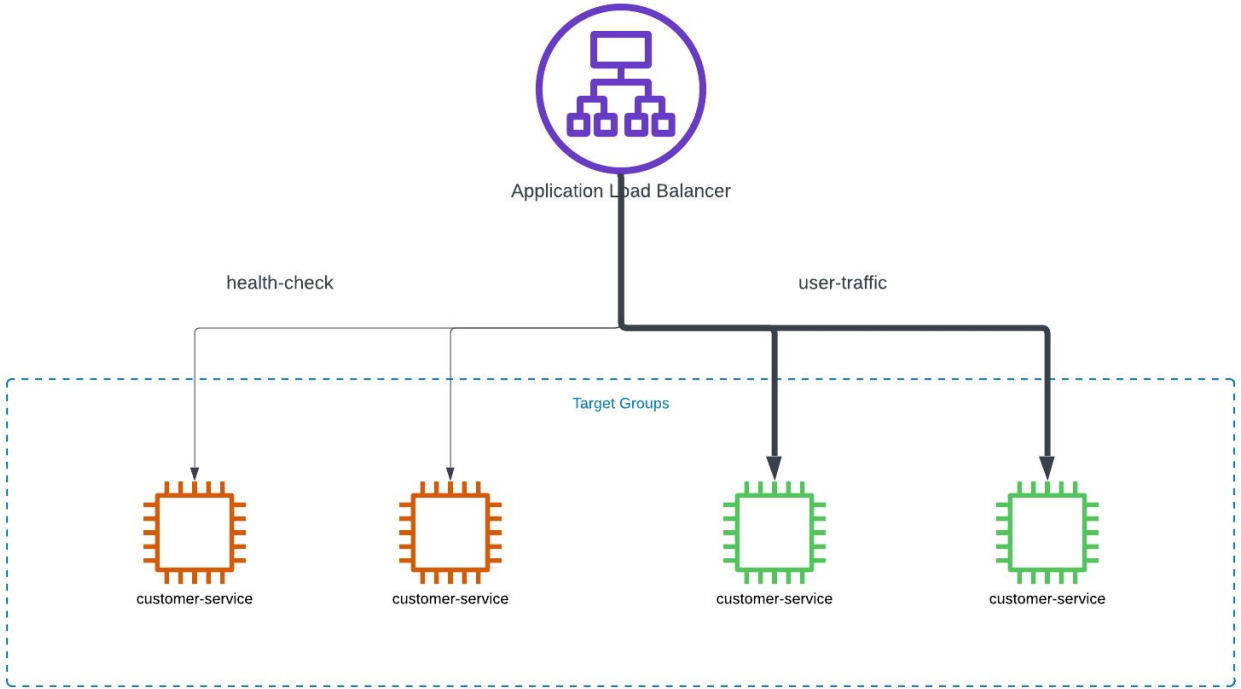
- Amazon recommended to use S3
  - Store the ~/.m2 in a S3 bucket
  - Download from S3 next time
- No significant improvement! (It might depend on the project)



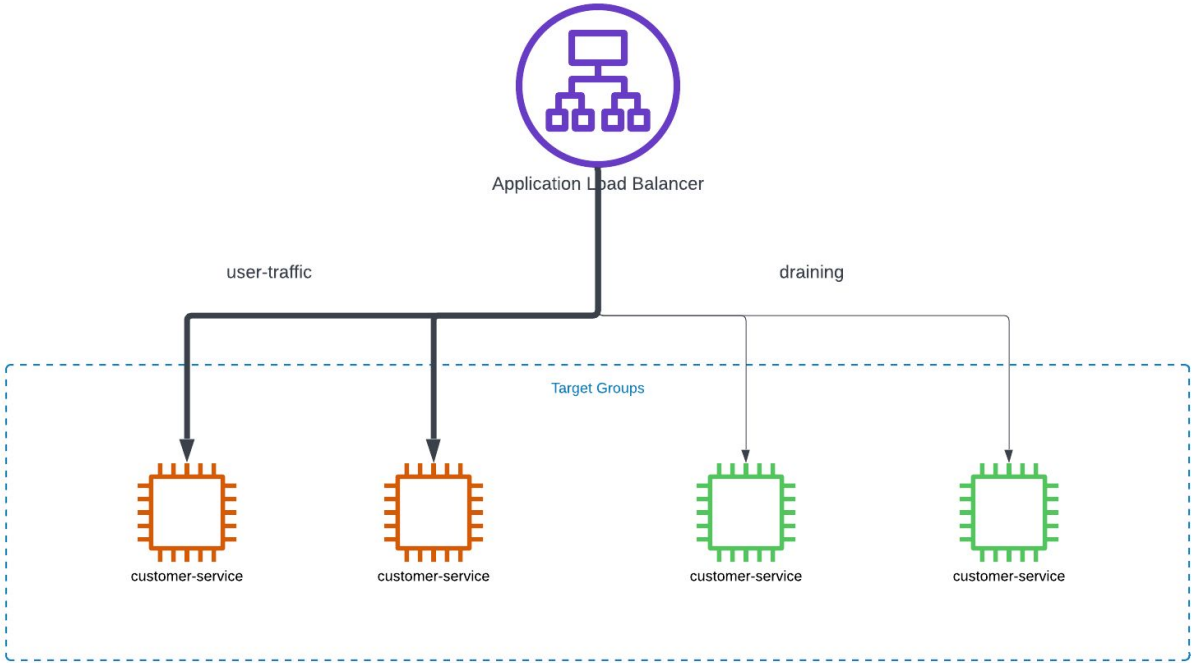
Rolling Update



Rolling Update



Rolling Update



What About QA/Staging Environment?