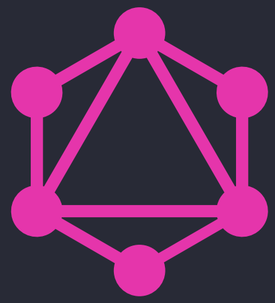
Technology Survey in Data-intensive Application Development And Sample Implementation





Final Project

Mohammad Mehrdad Shahidi

What is Data-intensive Application about?

Primary Challenges:

- Quantity of data
- Complexity of data
- Data changing speed

// On the other hand compute-intensive

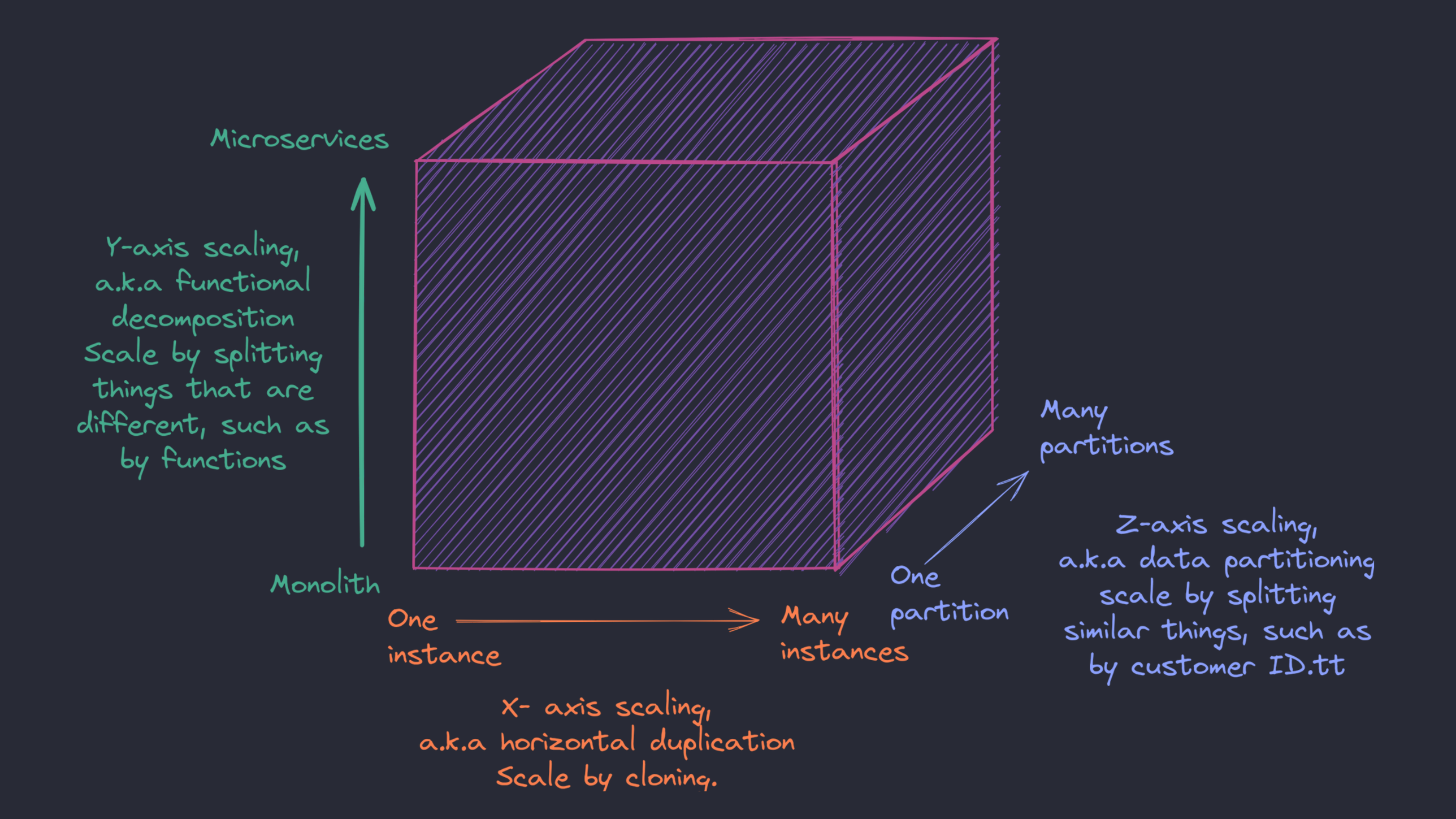
3 Factors That Influence Data-Intensive App

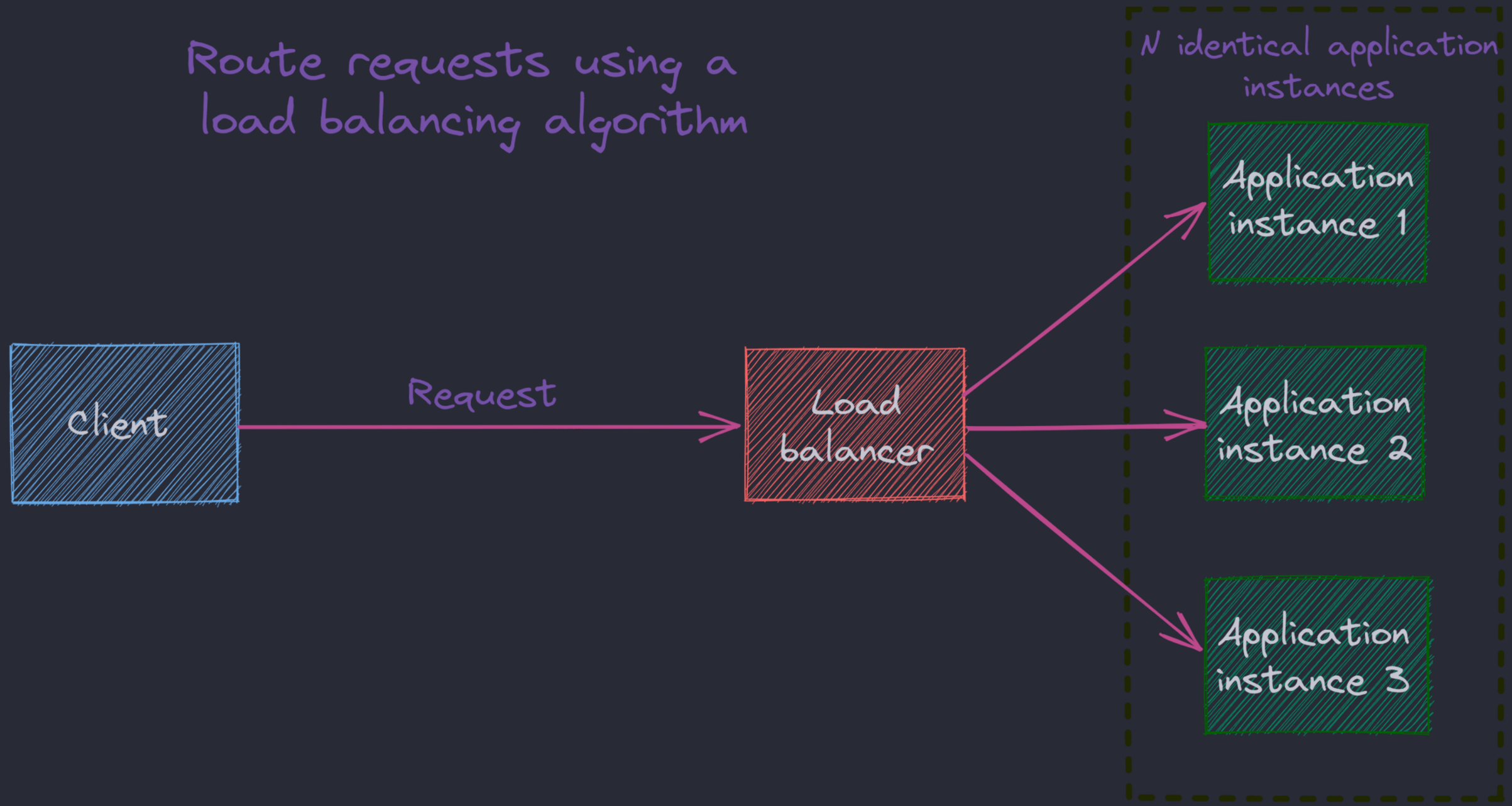
- Reliability
- Scalability
- Maintainability

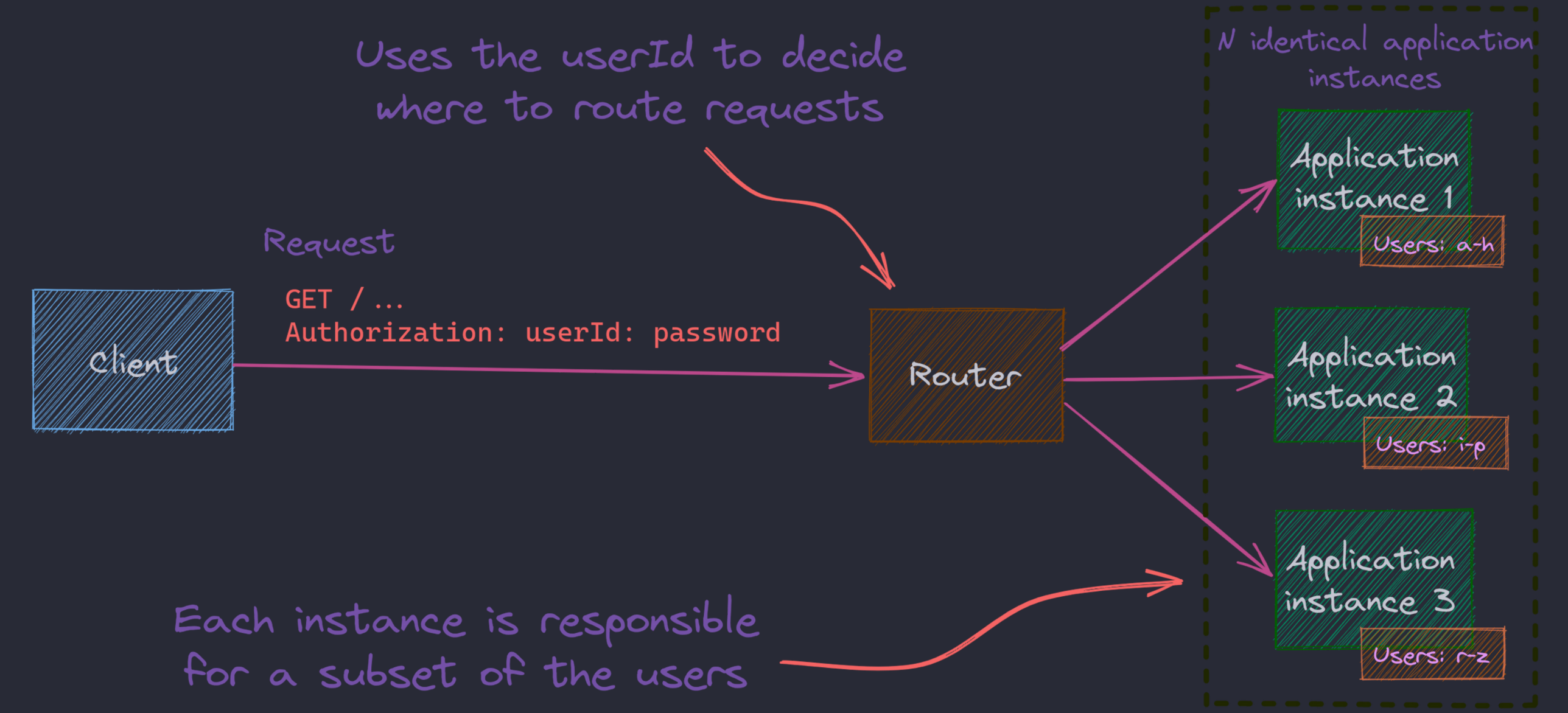
Microservice VS Monolithic

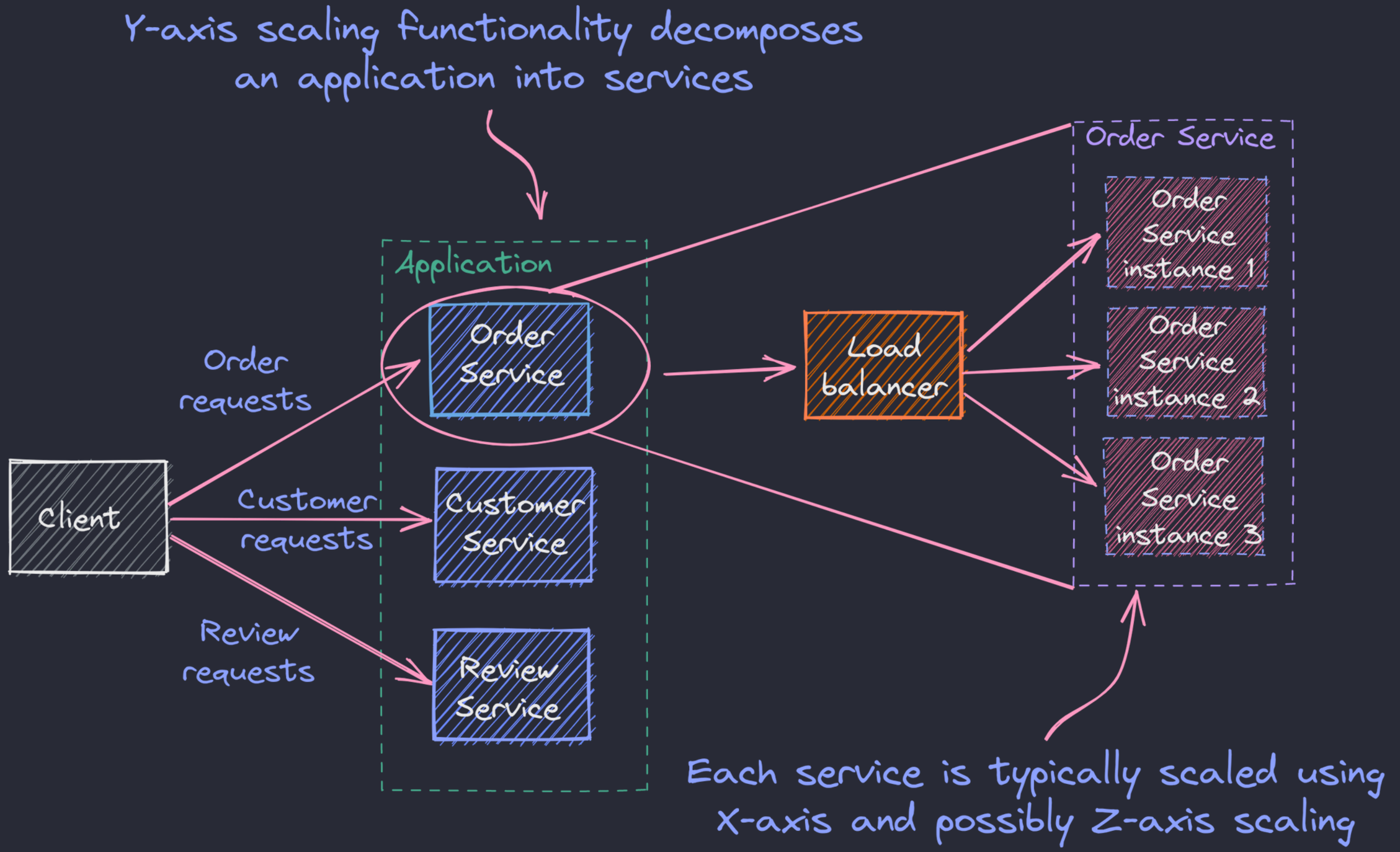
What Is Wrong With Monolithic?

- Complexity in Software
- Slow Down Development
- Difficulty in scaling
- Reliability is a challenge
- Hard to replace obsolete technology


Microservices
Y-axis scaling,
a.k.a functional
decomposition
Scale by splitting
things that are
different, such as
by functions
Monolith
One
instance
Many
instances
X- axis scaling,
a.k.a horizontal duplication
Scale by cloning.
One
partition
Many
partitions
Z-axis scaling,
a.k.a data partitioning
scale by splitting
similar things, such as
by customer ID.tt


Route requests using a load balancing algorithm
N identical application instances
Client
Request
Load balancer
Application instance 1
Application instance 2
Application instance 3


Uses the userId to decide where to route requests
N identical application instances
Application instance 1
Users: a-h
Request
GET / ...
Authorization: userId: password
Client
Router
Application instance 2
Users: i-p
Application instance 3
Users: r-z
Each instance is responsible for a subset of the users


Y-axis scaling functionality decomposes an application into services
Application
Order Service
Load balancer
Order Service instance 1
Order Service instance 2
Order Service instance 3
Client
Order requests
Customer requests
Review requests
Customer Service
Review Service
Each service is typically scaled using X-axis and possibly Z-axis scaling

What Is Good/Bad About Microservices?

Good

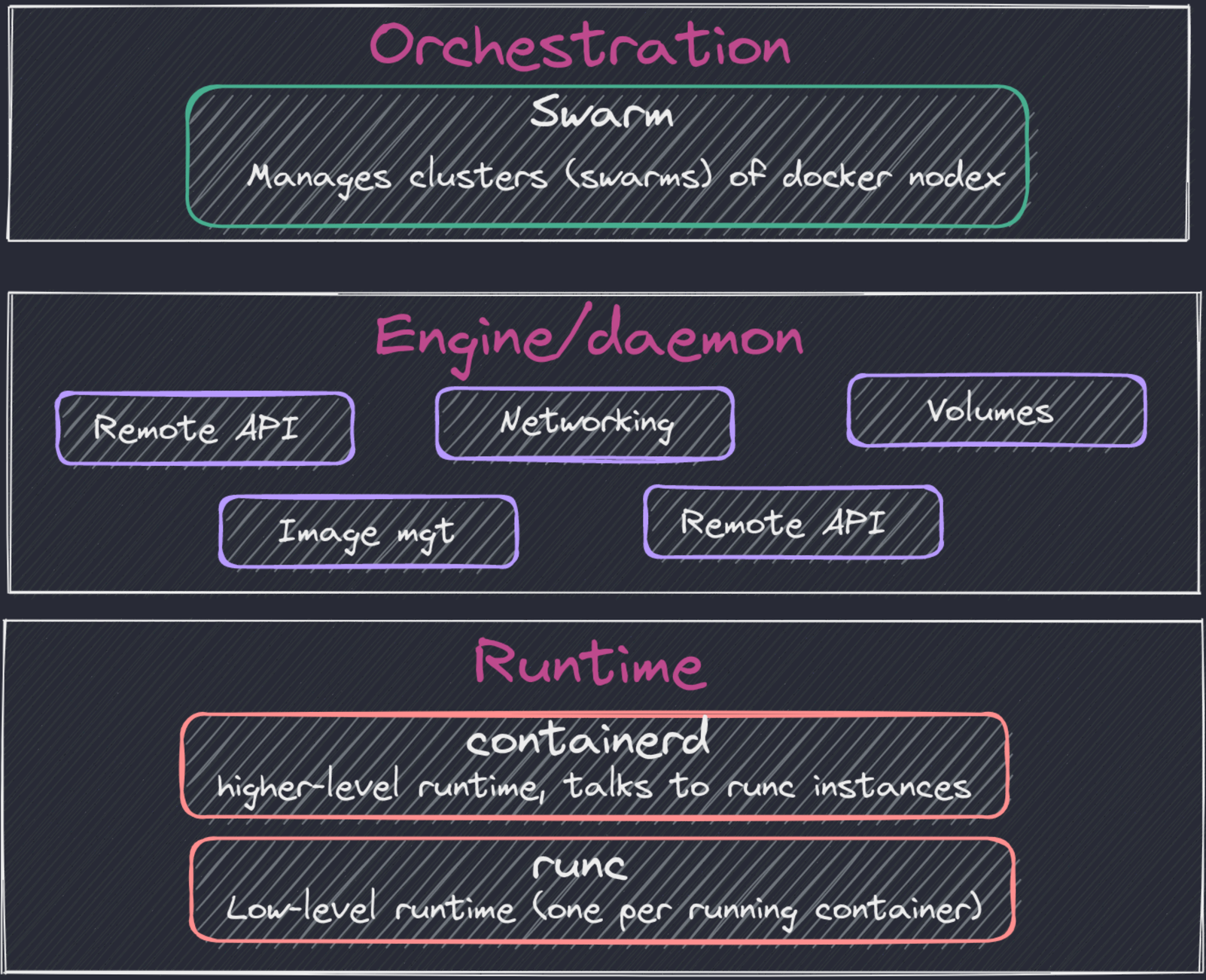
- Better CI/CD
- Small and maintainable
- Independently deployable
- Independently scalable
- Experimenting new tech
- Different team for development
- Fault isolation

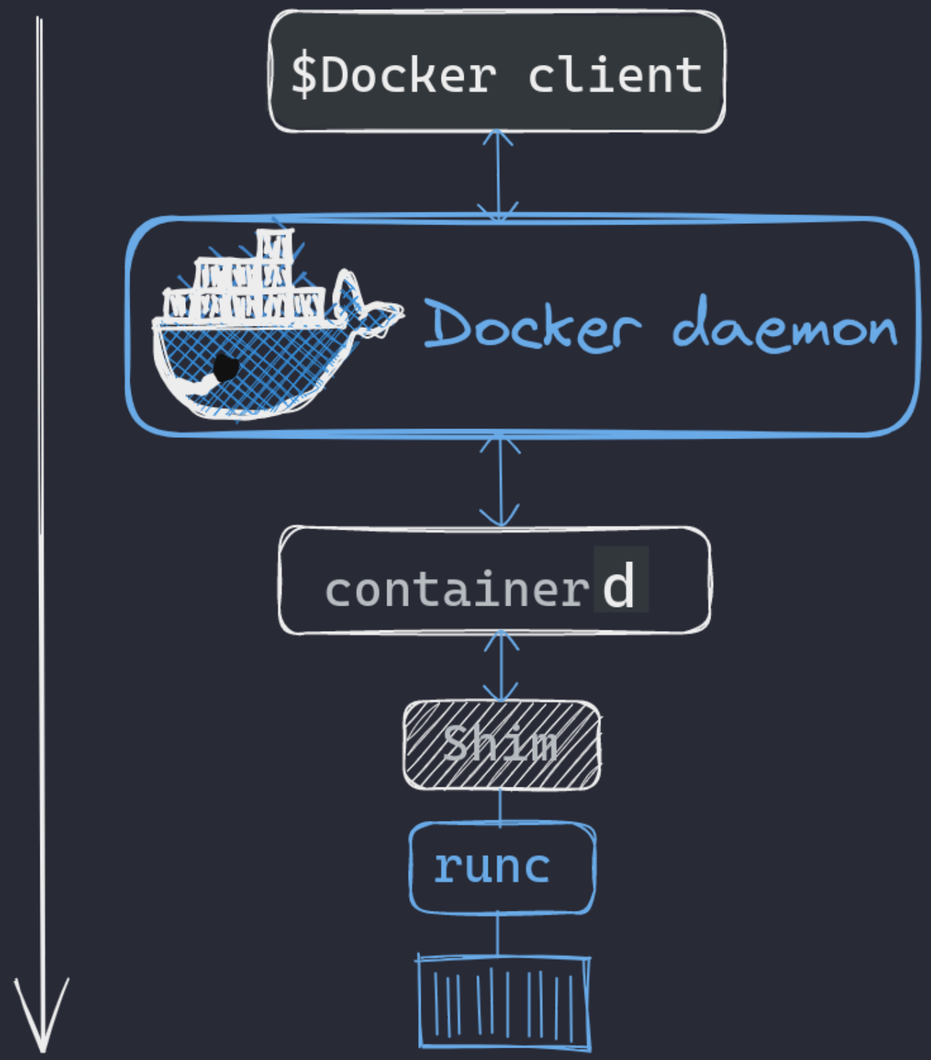
Bad

- Challenge in breaking down
- Complexity in Distribution
- Difficulty in coordination
- When we should use it?

Docker

- The runtime
- The Daemon //Aka engine
- The orchestrator //Swarm


Orchestration
Swarm
Manages clusters (swarms) of docker nodex
Engine/daemon
Remote API
Networking
Volumes
Image mgt
Remote API
Runtime
containerd
higher-level runtime, talks to runc instances
runc
Low-level runtime (one per running container)



Issue 'docker container run' command to Docker API exposed by Docker daemon

Receive instruction at API endpoint.

instruct containerd (via gRPC API) to start new container based on OCI bundle and ID provided

Receive instruction to create containers

Instruct runc to create container

Build and start container
runc exit after container start
shim become container's parent process

Image

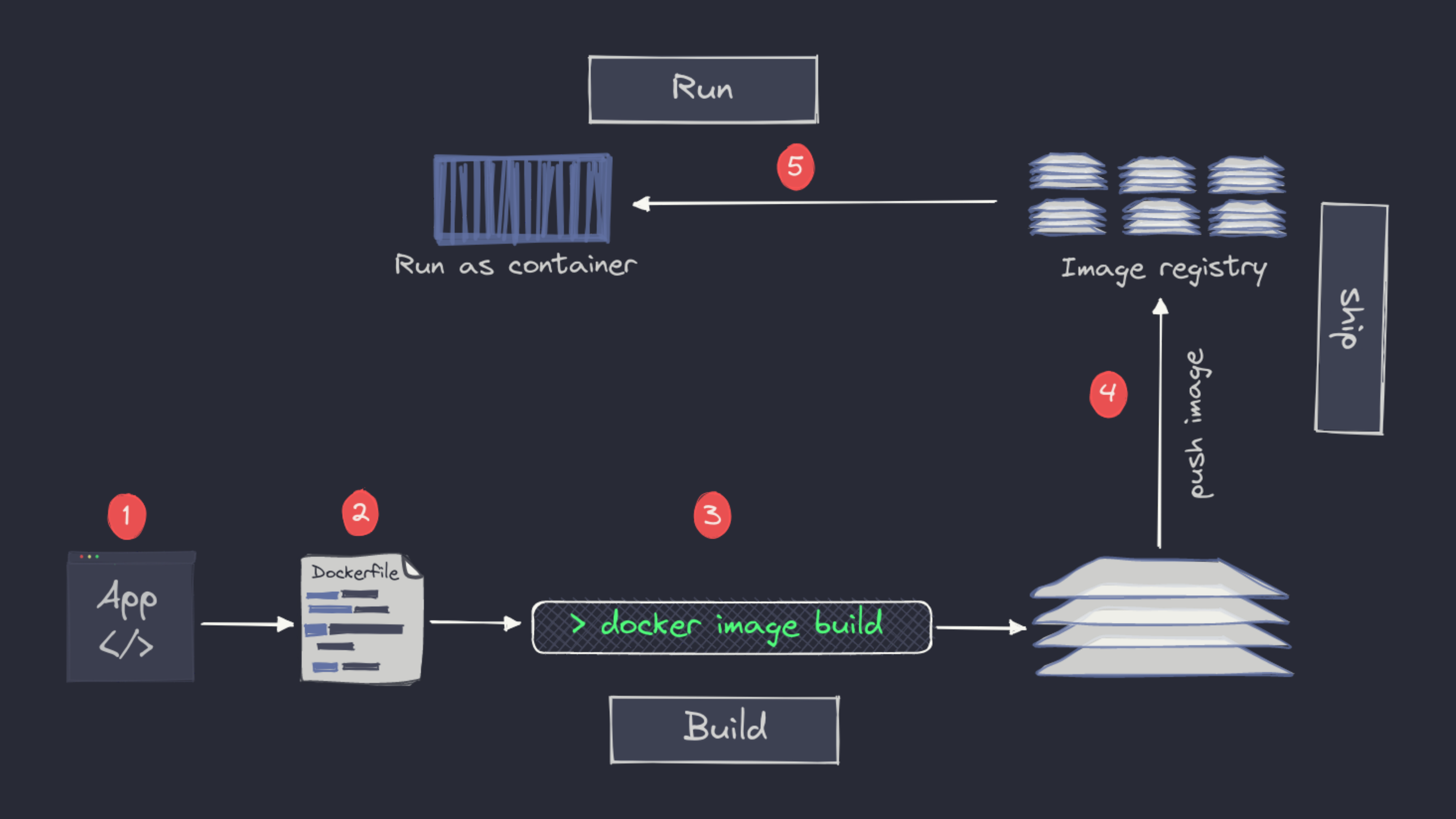
//Class

Container

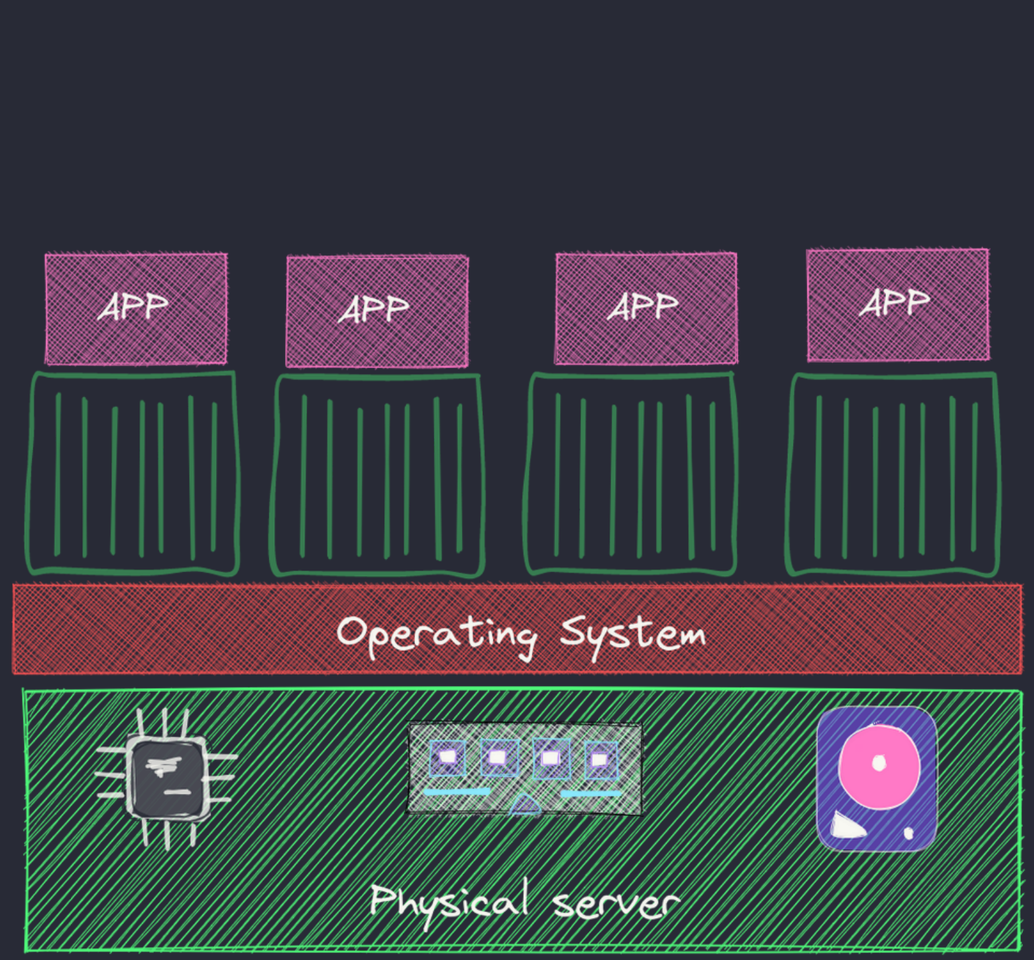
//instance of class

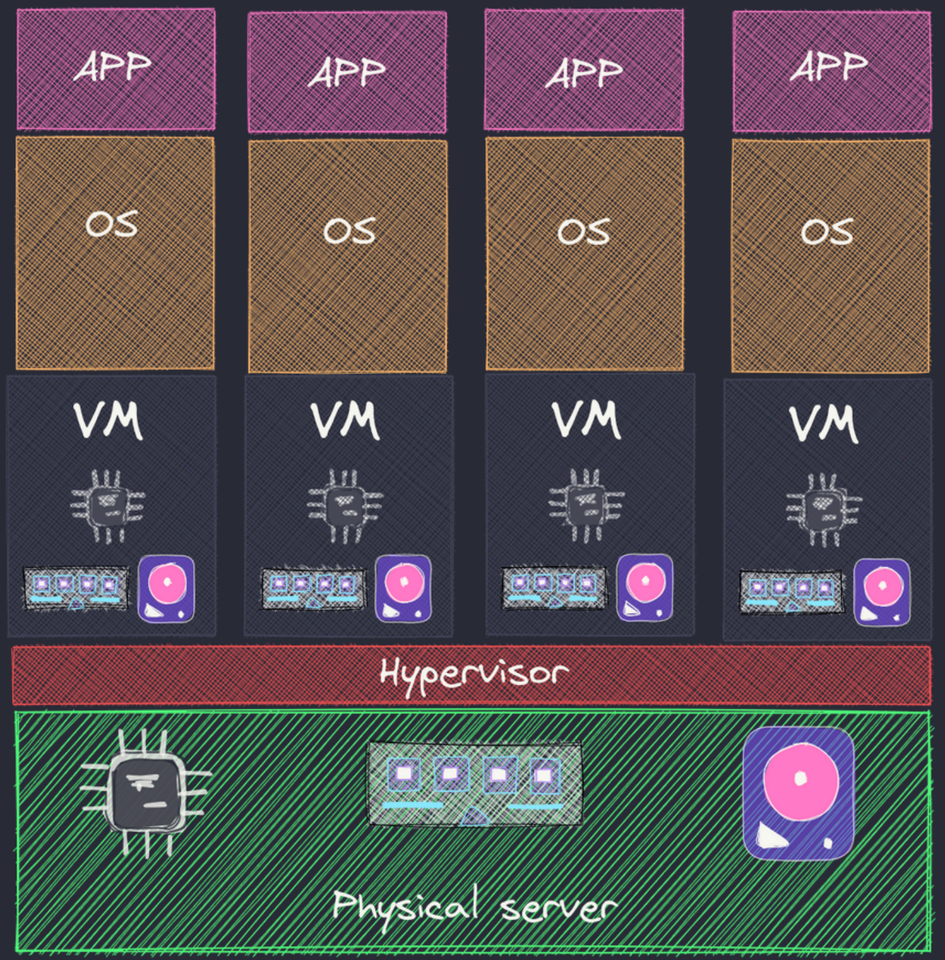
Dockerfile

```
FROM alpine
LABEL maintainer="m.mehrdadshahidi@gmail.com"
RUN apk add --update nodejs nodejs-npm
COPY . /src
WORKDIR /src
RUN npm install
EXPOSE 8080
ENTRYPOINT ["node", "./app.js"]
```


Run
5
Run as container
Image registry
ship
4
push image
1
App
</>
2
Dockerfile
3
> docker image build
Build

Container VS VM


APP
APP
APP
APP
Operating System
Physical server

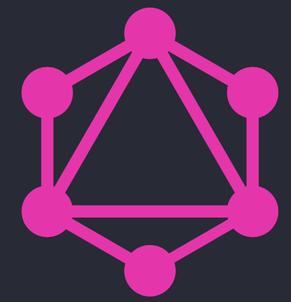

APP
APP
APP
APP
OS
OS
OS
OS
VM
VM
VM
VM
Hypervisor
Physical server

GraphQL VS REST

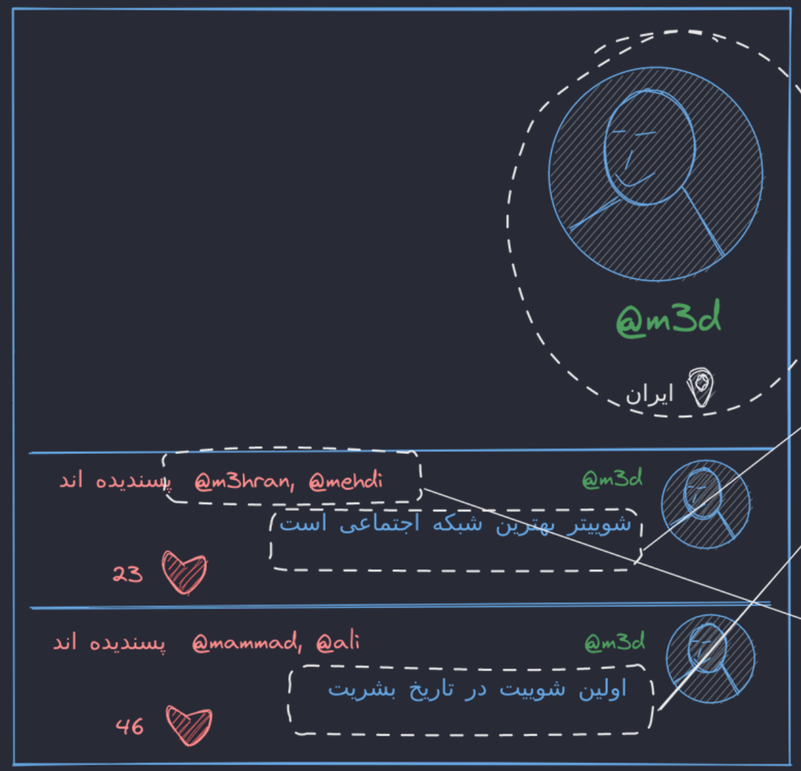
What Is Wrong About REST?

- Over-fetch
- Under-fetch //N+1 Problem
- inflexible

GraphQL



- SDL
- Query
- Mutation



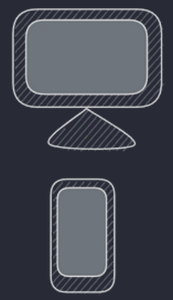

```
user {
    username: "m3d",
    location: "iran",
    avatar: "https://rers.dsf...."
}
```

```
shweets[
    {
    id: 1
    content: "....",
    likes: 23
    },
    {
    id: 2
    content: "...",
    likes: 46
    }
]
```

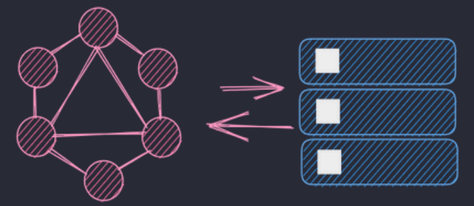
```
LikedShweet(id:1)[
    {
    username: "@mehran
     ...
    },
    {
    username: "@mehdi"
     ...
    }
    ⋮
]
```


1
HTTP GET
users/<id>
users/<id>/posts
posts/<id>/likes
user {
username: "m3d",
location: "iran",
avatar: "https://rers.dsf...."
}
2
HTTP GET
users/<id>
users/<id>/posts
posts/<id>/likes
shweets[
{
id: 1
content: "....",
likes: 23
},
]
3
HTTP GET
users/<id>
users/<id>/posts
posts/<id>/likes
LikedShweet(id:1)[
{
username: "@mehran
...
},
{
username: "@mehdi"
...
}
]

```
user (username: "m3d") {
    username
    location
    avatar
    shweets {
        id
        content
        likes
        likedUsers(take:2){
            username
        }
}
```

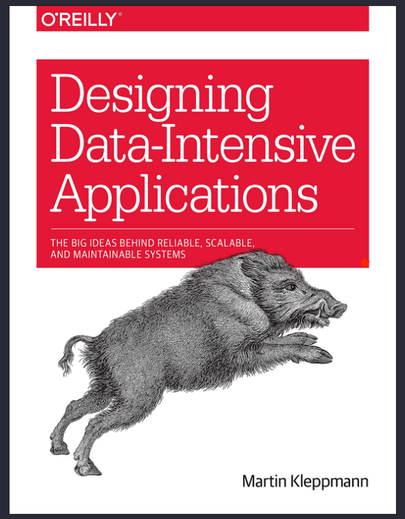


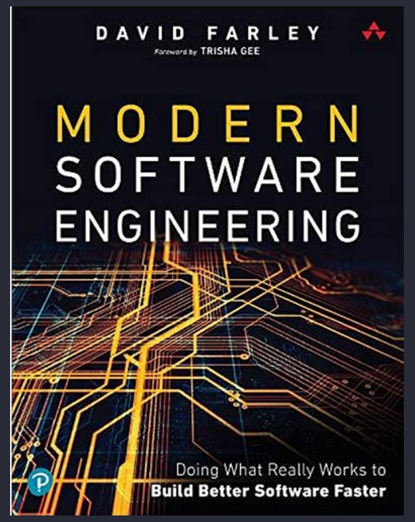
HTTP GET

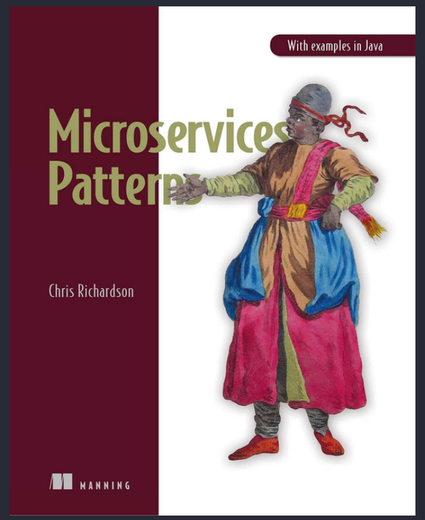


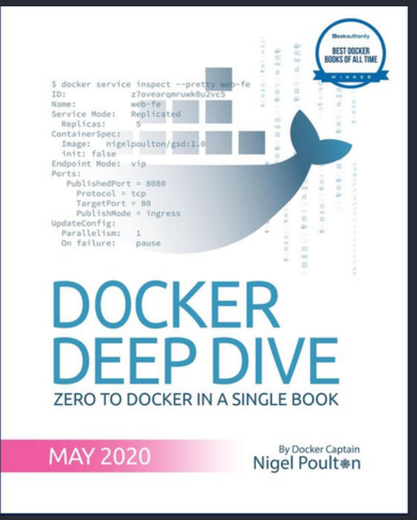
```
user {
    username: "m3d",
    location: "iran",
    avatar: "https://rers.dsf...."
    shweets: [
        {
            id: 1
            content: "....",
            likes: 23,
            likedUsers: [
                {
                 "username": "@mehran"
                 }
                ,
                {
                 "username": "@mehdi"
                 }
        }
        :
        :
    ]
}
```

COOL STUFF!!









How to GraphQL

mehrdadshahidi.ir //all the links

shwitter.mehrdadshahidi.ir

github.com/m3dsh/shwitter //Source code

canva.com //Used for this presentation

excalidraw.com //For drawing