POLITECNICO
MILANO 1863

Data and computation movement in fog computing

Section 2

Computation movement

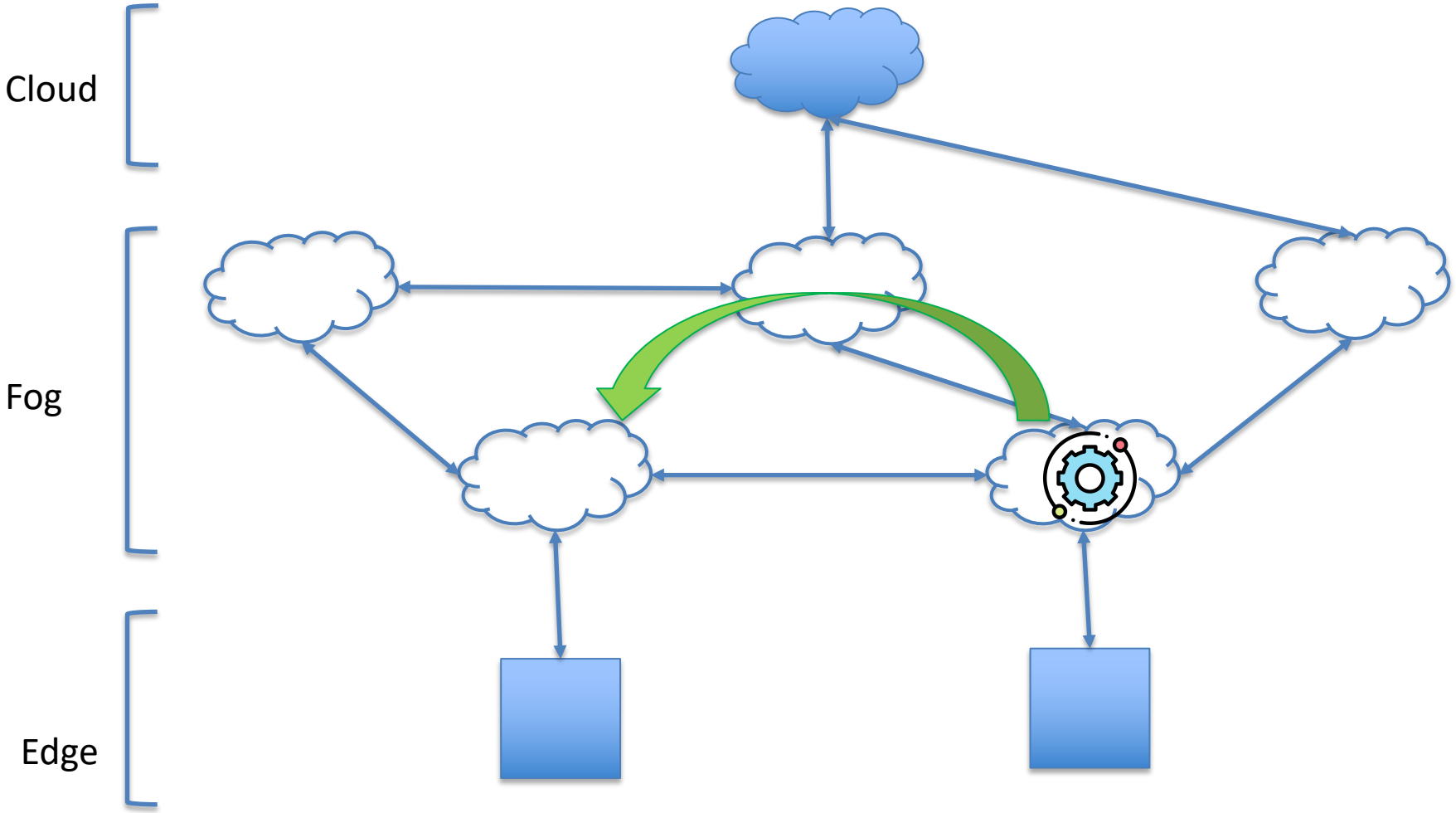
Computation movement

Movement or replication of software services among different fog nodes.

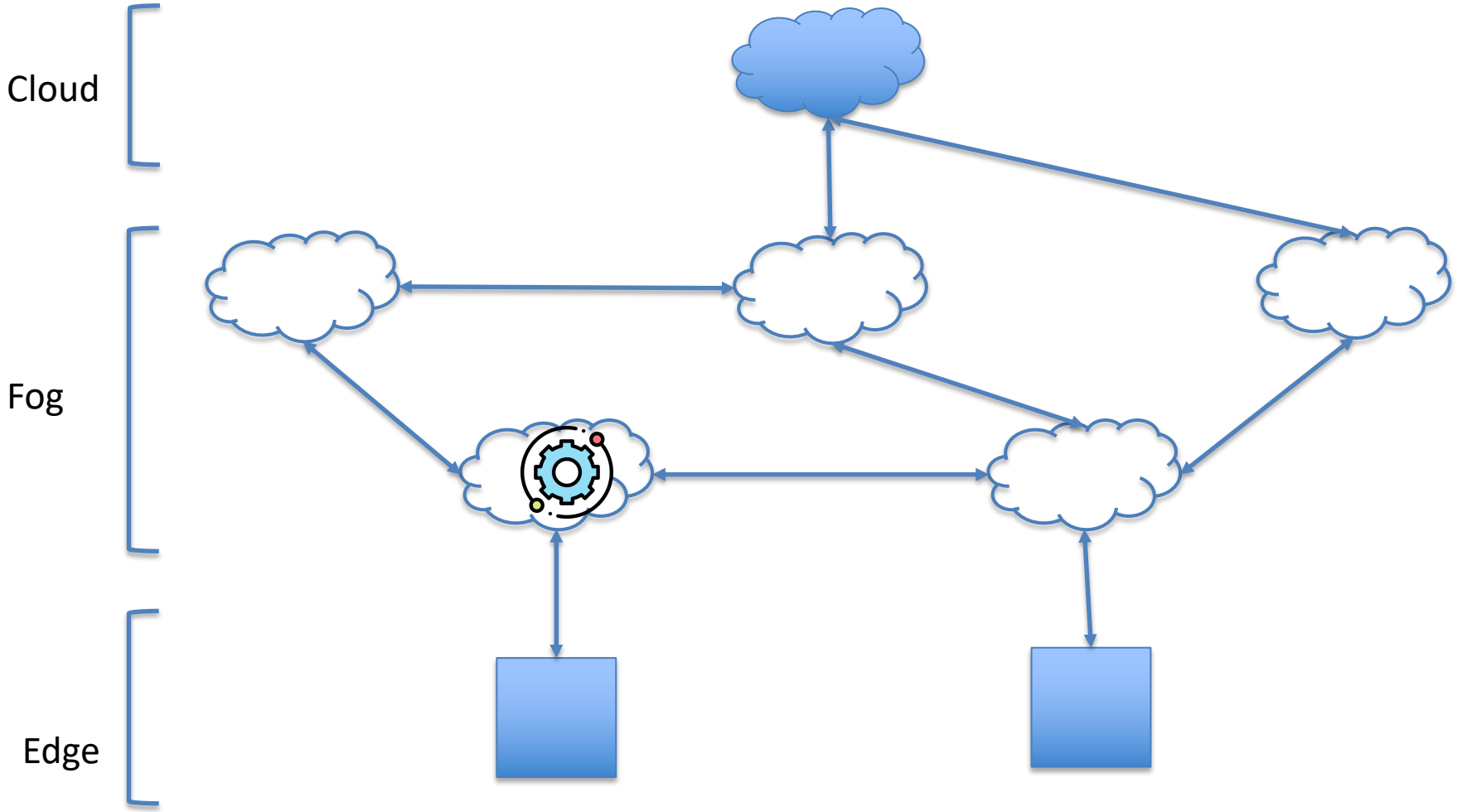
Computation movement vs computation replication

- Movement: a **service** is “copied” in another machine, while the original one is taken down.
- Replication: a service is “copied” in another machine

Typical fog infrastructure



Example of computation movement



Computation movement, why do we need it?

Movement and replication of computation solve many problems

Service too slow	->	move in a more powerful machine
Connection problems	->	move in a nearer machine
Security problems	->	move in a safer machine
Privacy problems	->	move in a machine controlled by the data controller
Pick requests	->	replicate service in another machine
Unreliable service near the edge	->	replicate a service in the cloud

What can be moved?

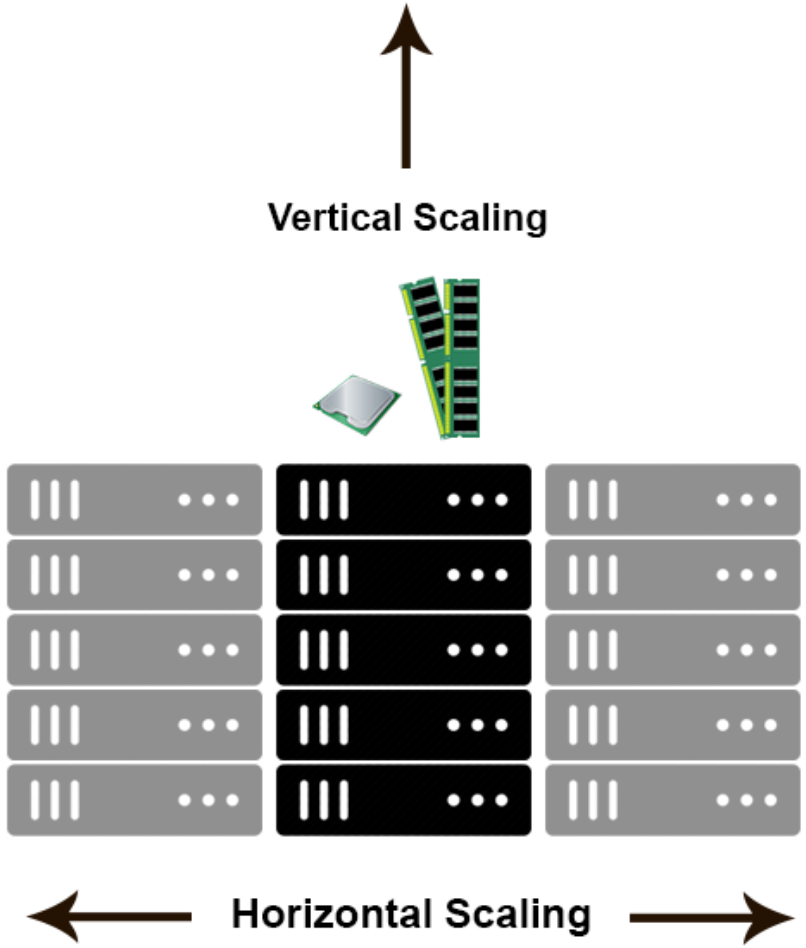
Every software that can be scaled horizontally

- Stateless software
- Software with mechanism to automatically retrieve their status
- Cluster-computing framework

Examples:

- Stateless application
- Spark (MapReduce)

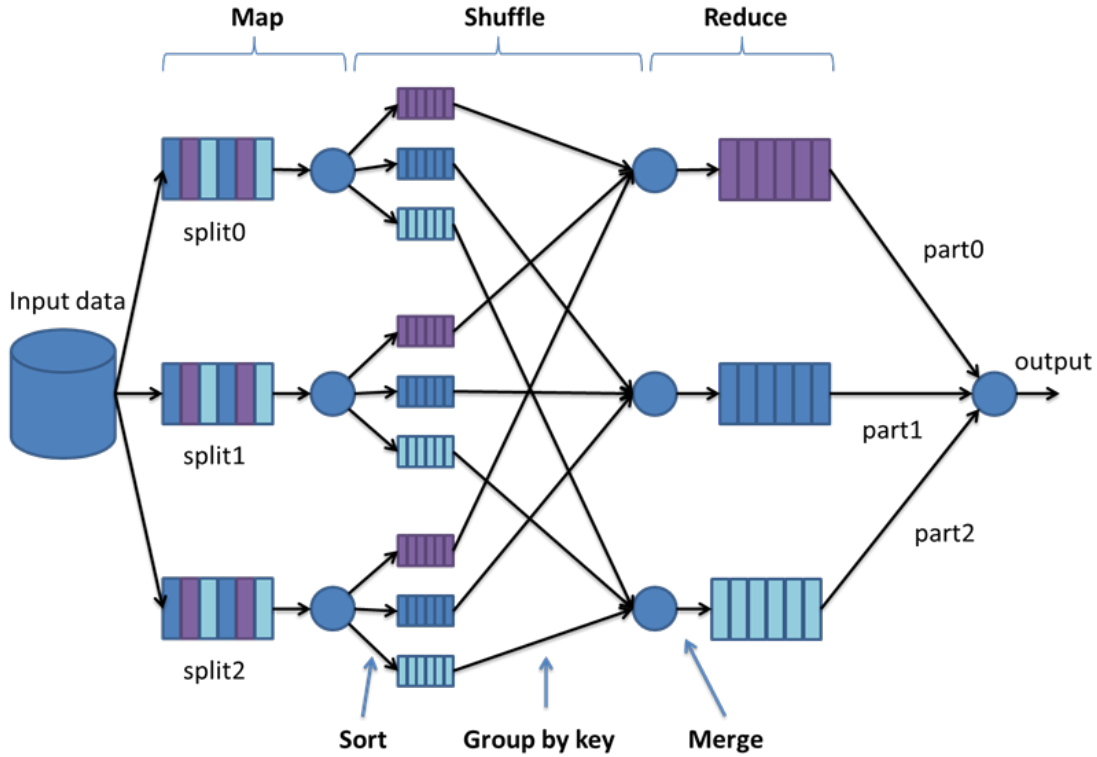
Scaling horizontally and vertically



[1]

Task distribution

Usually based on specialization of the *split-apply-combine* strategy for data analysis.



Stateless applications

In a stateless application the server does not store any state about the client request.

- Allows to deploy a new instance in a new machine without loosing "session" data.

IMPORTANT: state about the client is different from the state of internal resources!

How do we scale horizontally?

Virtualization is the way: there are different type of virtualization, in this case we focus on virtualization of hardware (PaaS, IaaS).

- I can easily emulate a new machine

Virtualization frameworks allow this!

We will focus on Docker and Kubernetes (K8s)

Docker

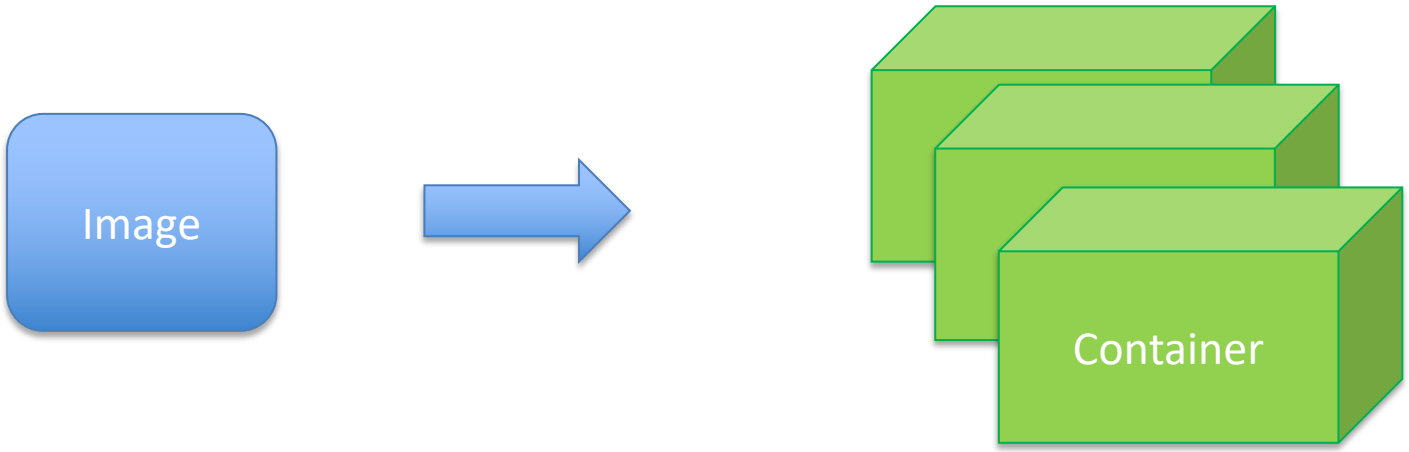
Docker is a virtualization framework that can create virtual machines called **containers.**

Containers are lightweight virtual machines

- they don't emulate the hardware, but they use the API of the operative system that hosts Docker.

Containers are created from **images**.

An image is a description of a container, from an image is possible to create and run many identical containers.

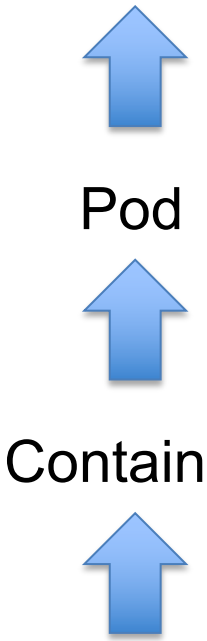


Kubernetes

Kubernetes is an open-source container-orchestration system for automating application deployment, scaling, and management.

It allows to easily create containers and manage them

Kubernetes - Hierarchy

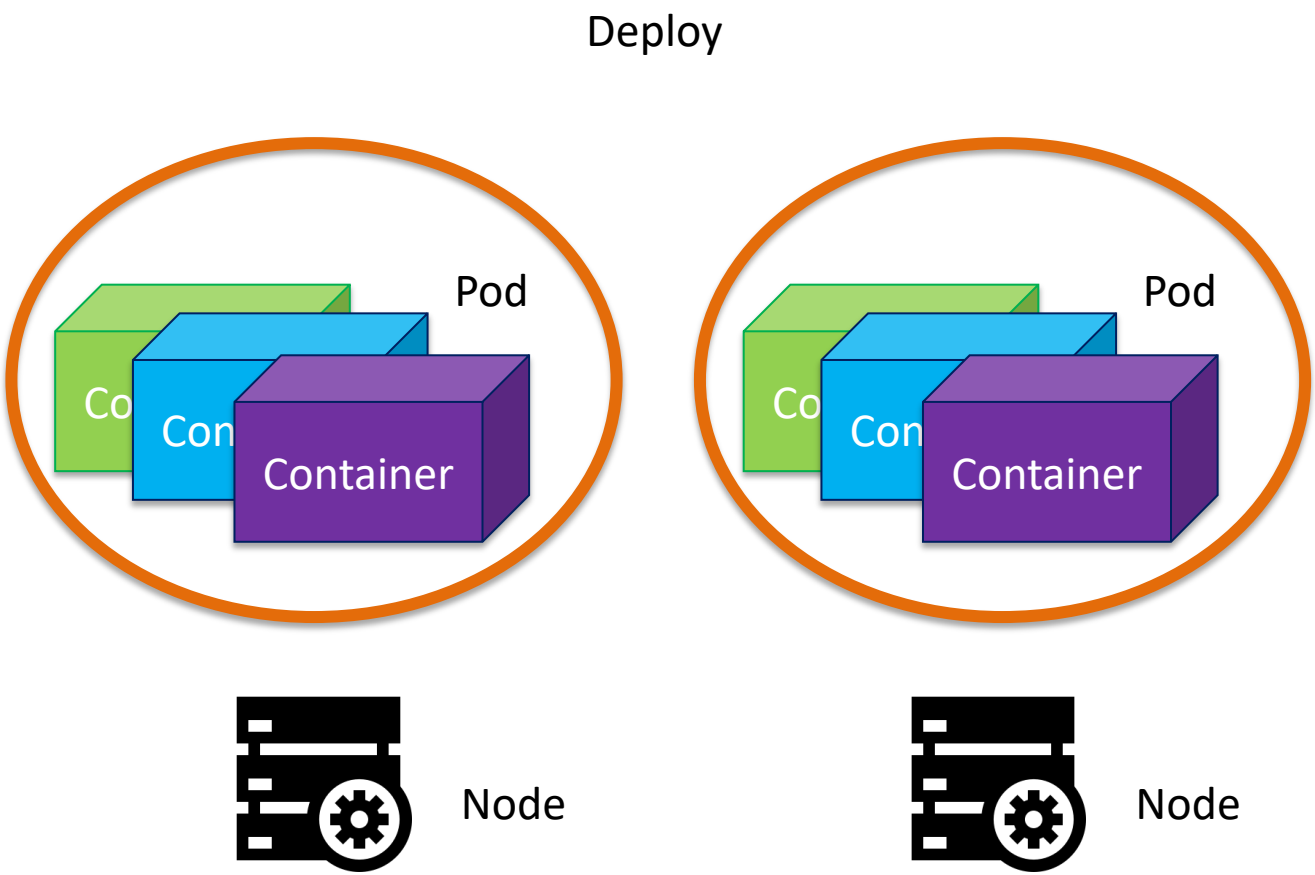


A set of identical Pods
(used for horizontal scaling)

A set of containers

The virtual machine run in the node

Infrastructure
(physical or virtual)


Deploy
Pod
Co
Con
Container
Pod
Co
Con
Container
Node
Node

Kubernetes - Hierarchy

2 deploys of the pod

(Weather forecast, twitter analysis containers)

Twitter analysis container

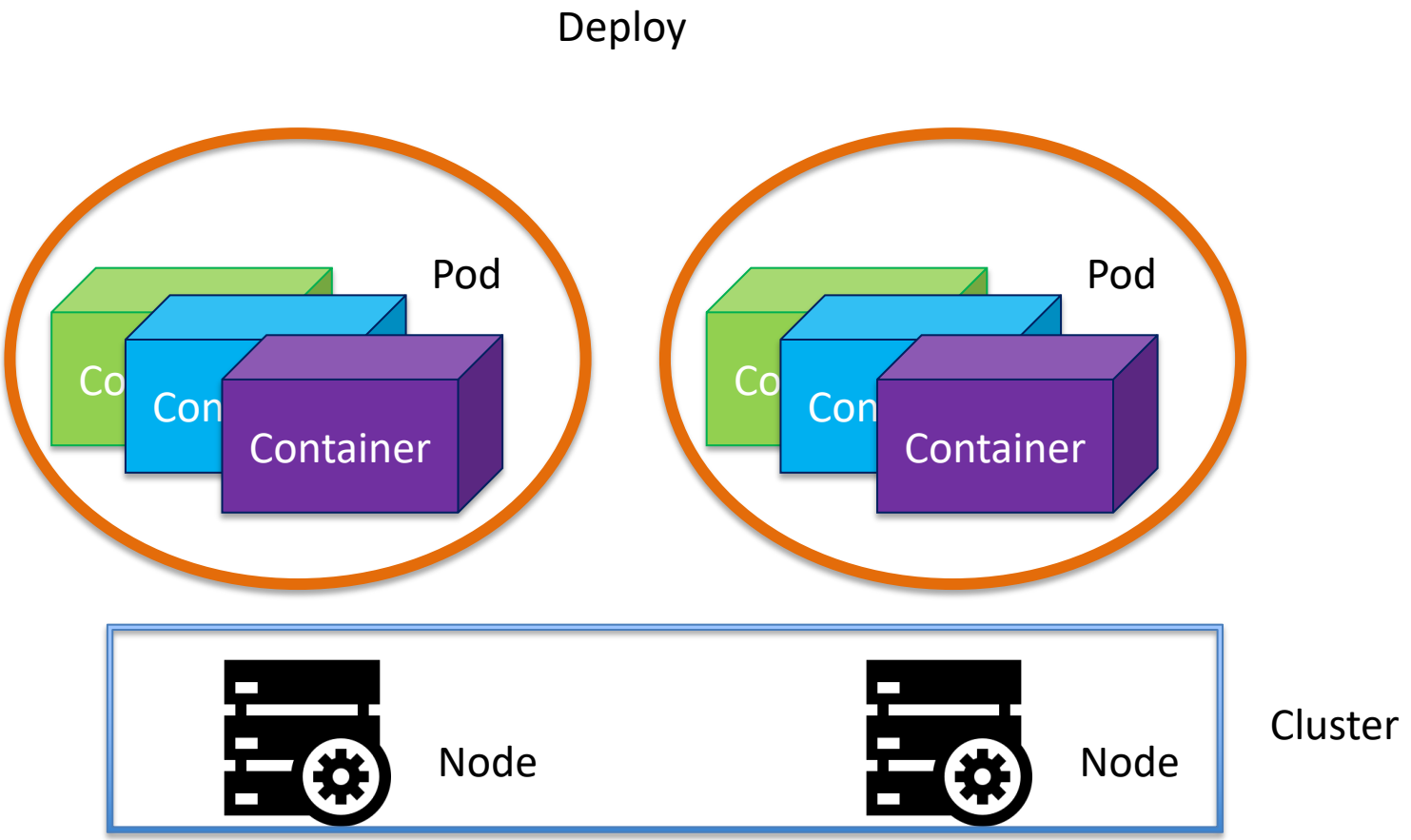
X86 server

Kubernetes - Hierarchy

Cluster — A geographically closed set of machines

Node — Infrastructure (physical or virtual)

Kubernetes


Deploy
Pod
Co
Con
Container
Pod
Co
Con
Container
Node
Node
Cluster

Take away message

Computation movement can be easy with the right choices

- (Micro) services
- Stateless services
- Cluster-computing frameworks

And with the right technology

- Virtualization (Docker, VMWare, VirtualBox)
- Orchestration (Kubernetes, ..)

Take away message

Positive points

- Bring computation task where is needed
- Take computation task from node with limited power (to save battery, ..)
- Distribute computation

Negative points

- Who decide where to move?
- Limitation on location (privacy)
- Overhead (mad-reduce)

Data movement

Data movement

Movement or replication of data among different (fog) nodes.

Data movement vs data replication

- Movement: data is transferred on a new node, the original copy is removed
- Replication: data is copied on a new location and maintained aligned

What can be moved?

Data stored in any form can be moved:

- Raw data
- Data base
    - Relational
    - Non relational
    - Object storage

We focus on databases:

the more advance the technology the faster the movement will be

How can we move/replicate data in database?

Physical movement:

- Copy raw data using the file system

Logical movement:

- Ask the database for processed data
    - In case of relational database (tuples/tables)

Physical copy

Pros

- Fast: data are simply sent , not processed
- Easy to implement

Cons

- Cannot select a subset of data
- Cannot perform transformations
- Highly coupled with technology used

Logical copy

Pros

- We can select what to copy
- We can transform the part that will be copied
- Can support polyglot approaches

Cons

- Slower that physical copy

(Eventual) Consistency (and consensus)

Consistency among replica is a well study problem: how to keep two or more replicas with the same data?

If we examine 2 database in the same time, likely we will have different data

- Writes request arrive at different nodes at different time

Solutions

- Eventual consistency: if we stops all the writes, after a certain amount of time, all node will be consistent.
- Linearizability: one entry point for all nodes
- Distributed transactions: data is available only after all writes are executed in all nodes (2PC)

(Eventual) Consistency (and consensus)

Consistency among replica is a well study problem: how to keep two or more replicas with the same data?

Consistency applies while transferring data too: what will happen to updates of the original copy?

Solutions

- Block updates while transferring (to be avoided)
- Update database when transferring is finished (using writes log)
- …

Time of replication/movement

In fog we can have small amount of data when data comes from the edge and moves to cloud

We can have large amount of data when data moves from aggregated fog node to cloud or from cloud to cloud.

Time is a first class requirement, data is needed fast

An hybrid solution

Logical copy + physical copy

- Take the best of the two solutions

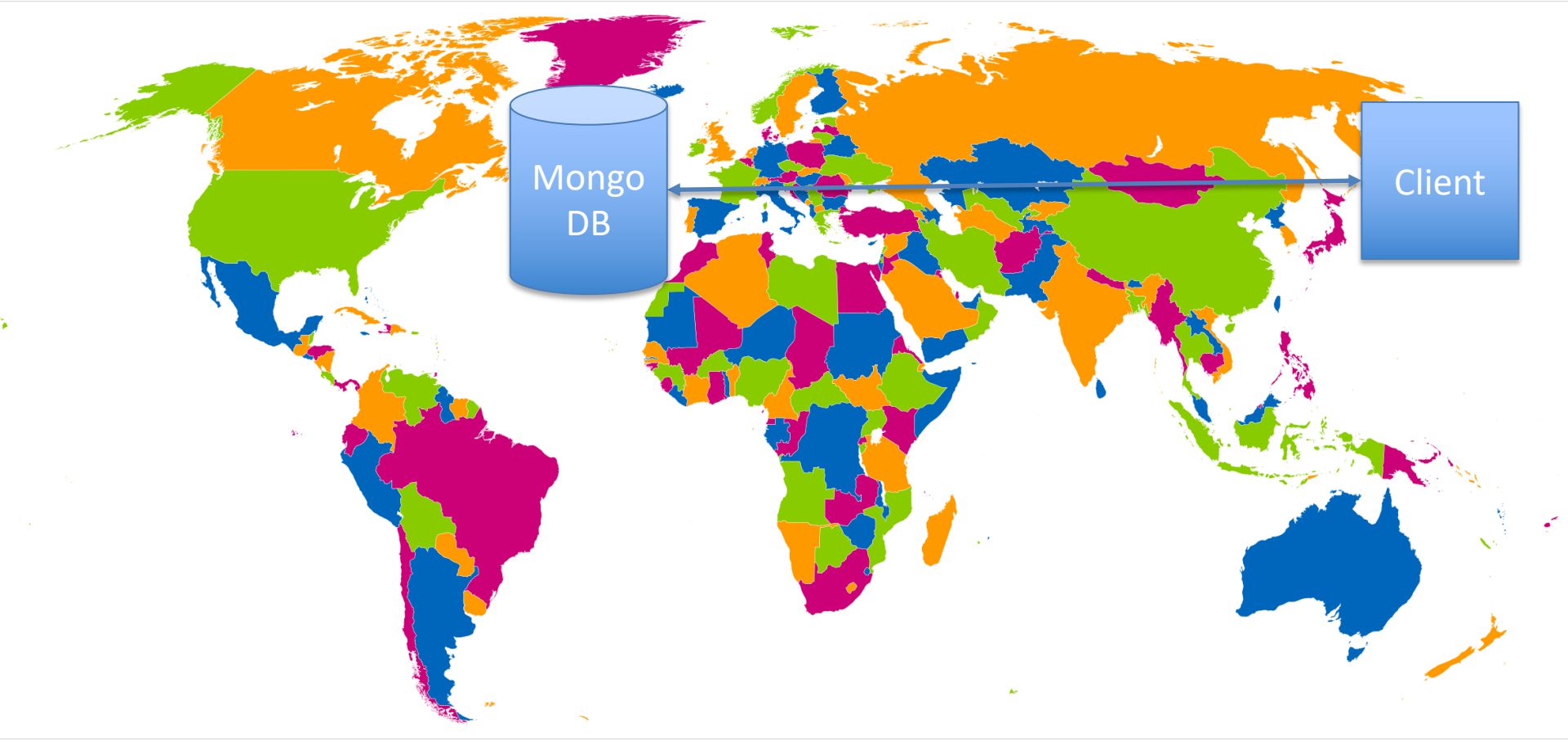
1. A logical copy is made near the original database
2. A physical copy moves the copied data to the new location
3. A logical copy approach keeps synchronized the two databases

Hand on section

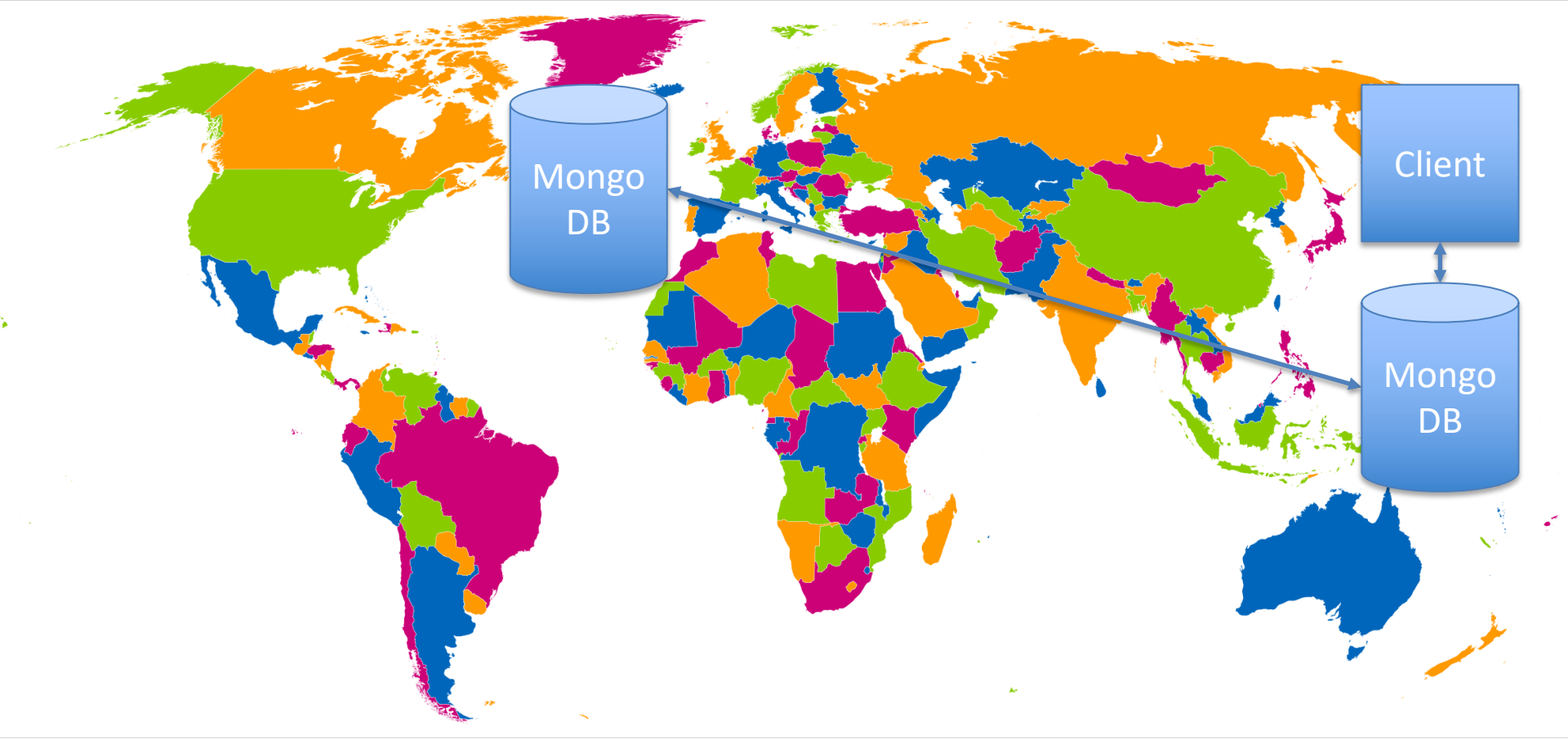
Objective replicate database

Why not move computation? too easy! ☺

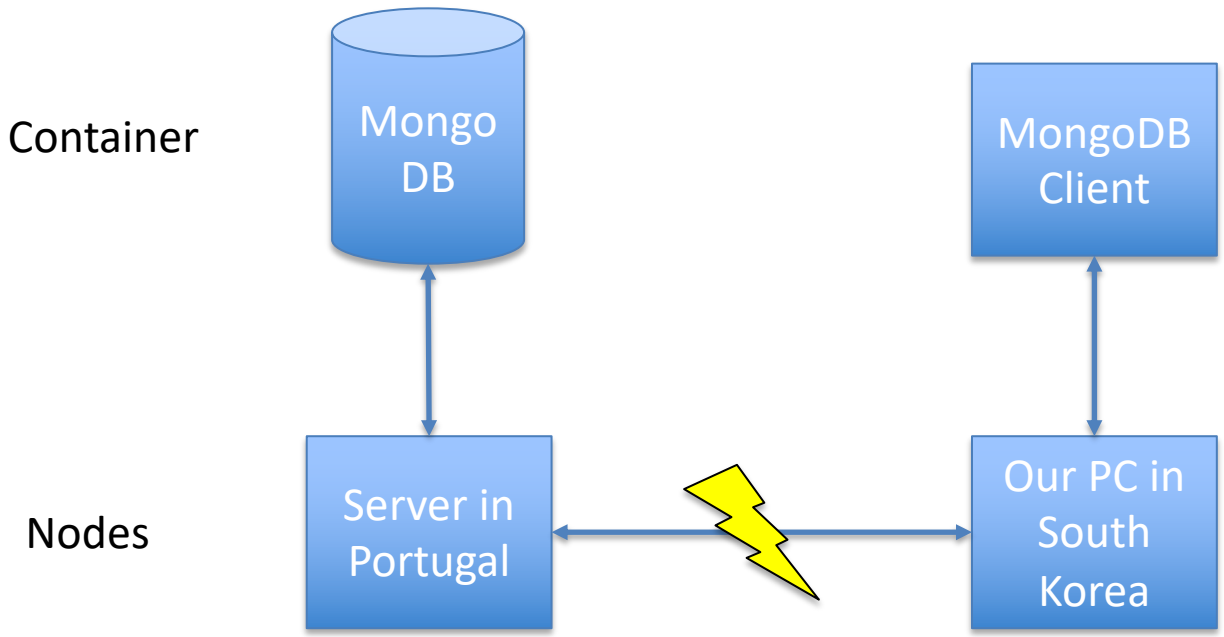
Case study description



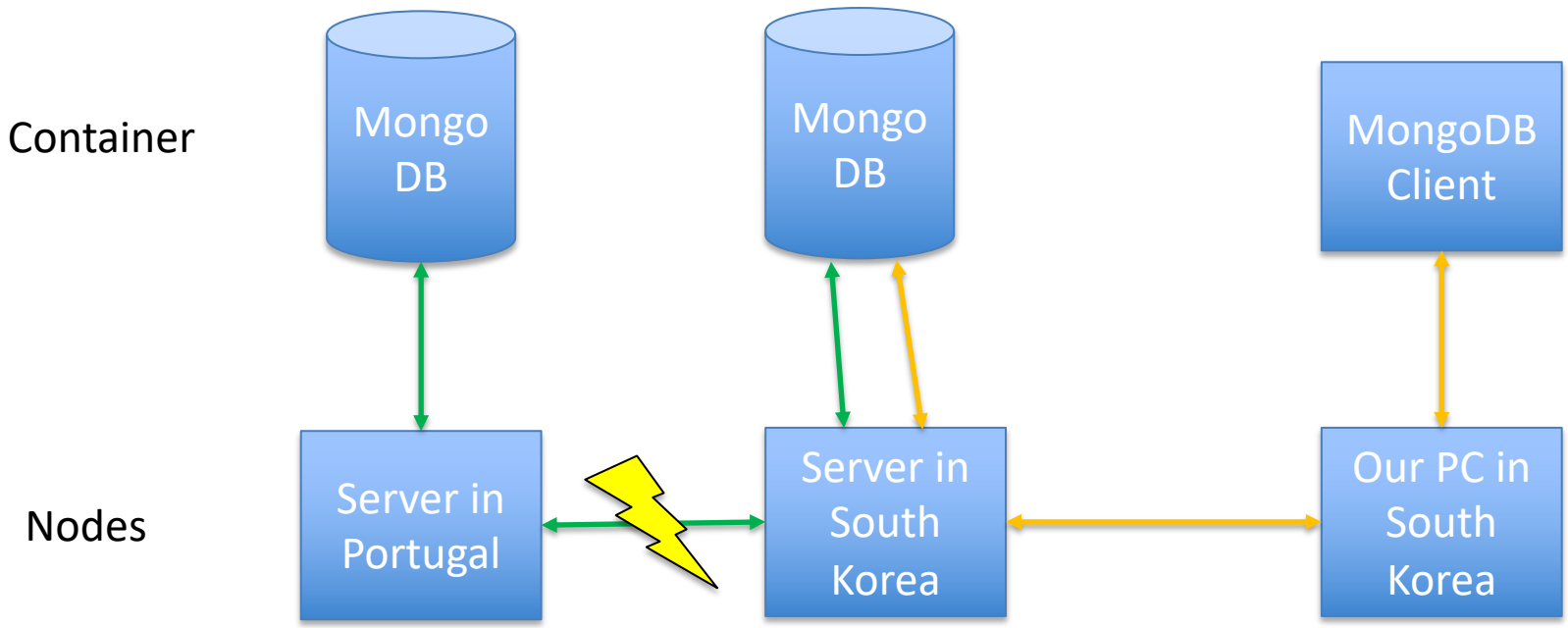
Case study description



Case study description



Case study description



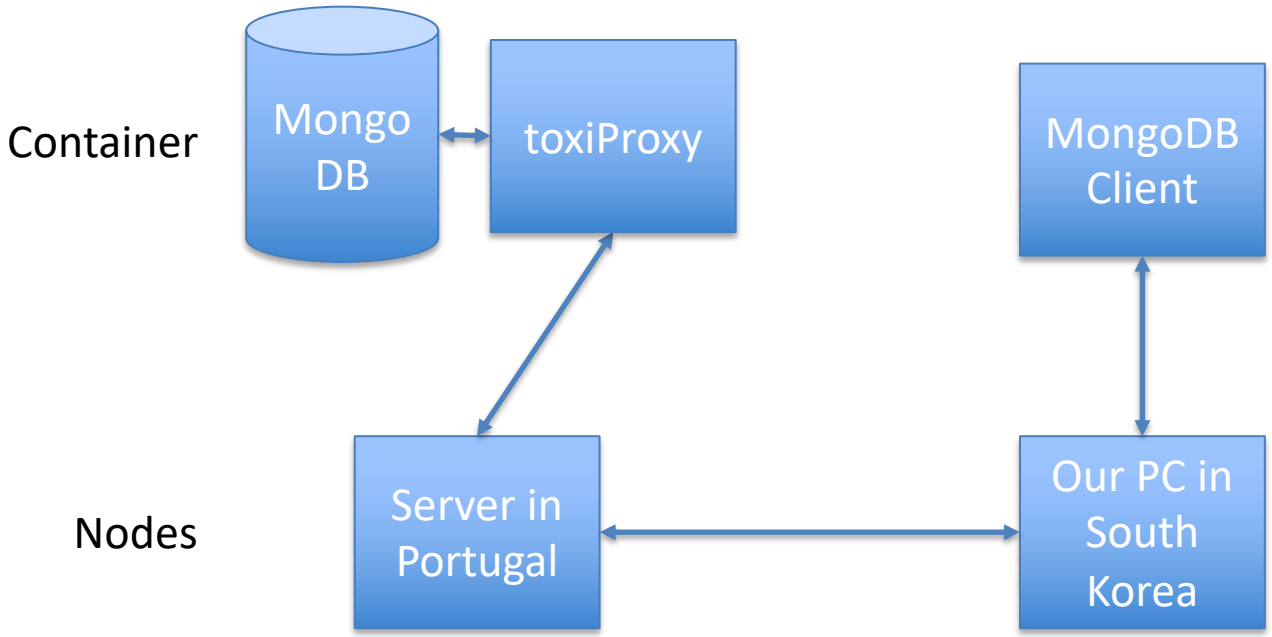
Which option is faster? Requirements matters!

1. Replicate data in South Korea server
2. Leave data in Portugal

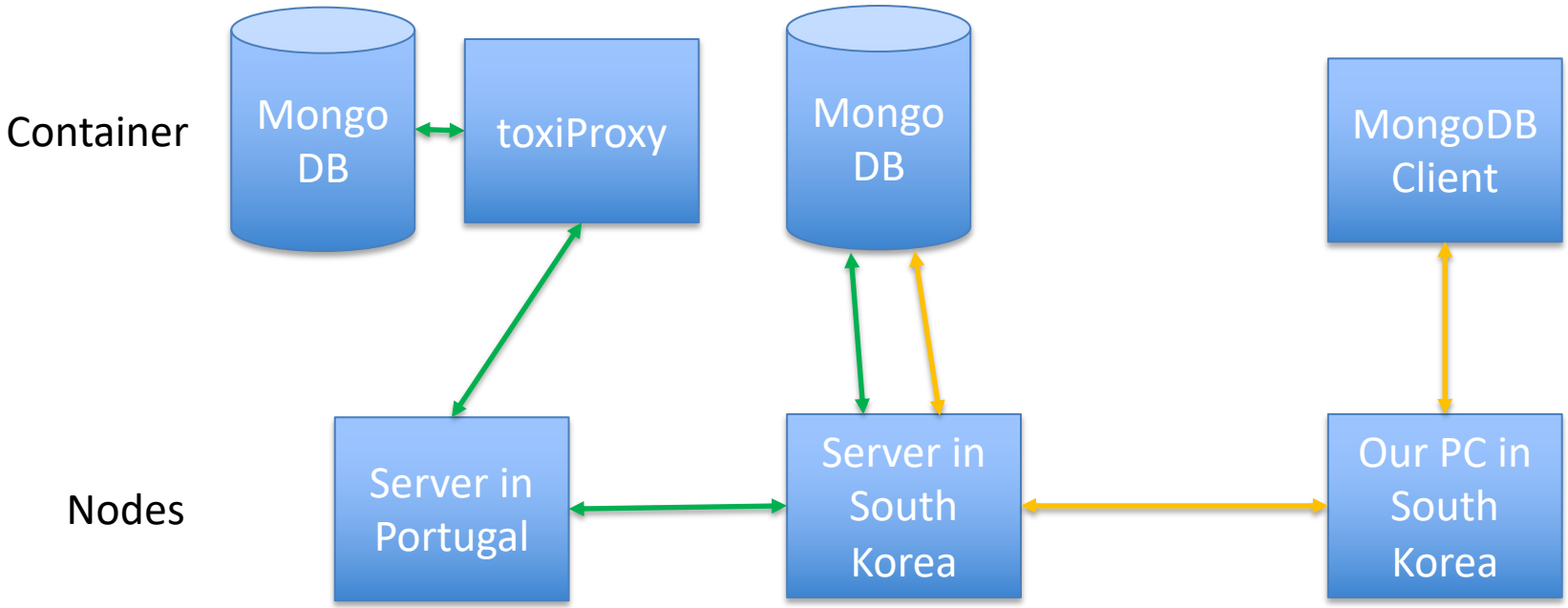
Time to propagate data must be considered:

- If I need a small update as soon a possible:
    - Option 2 is the best
- If large quantity of the dataset is used (historical analysis), or use the same data multiple times (aggregation, statistical analysis)
    - Option 1 is the best

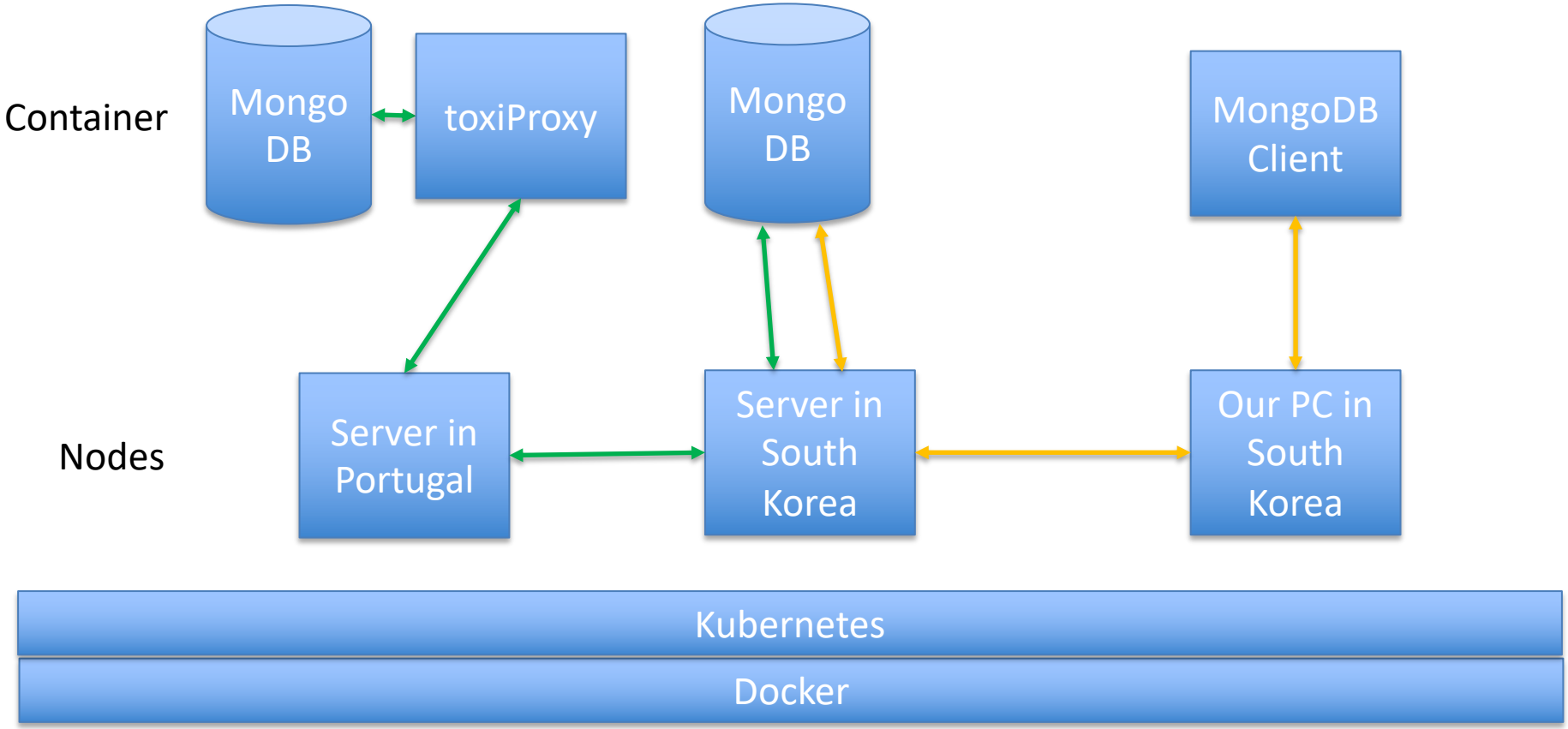
The real configuration on your machines



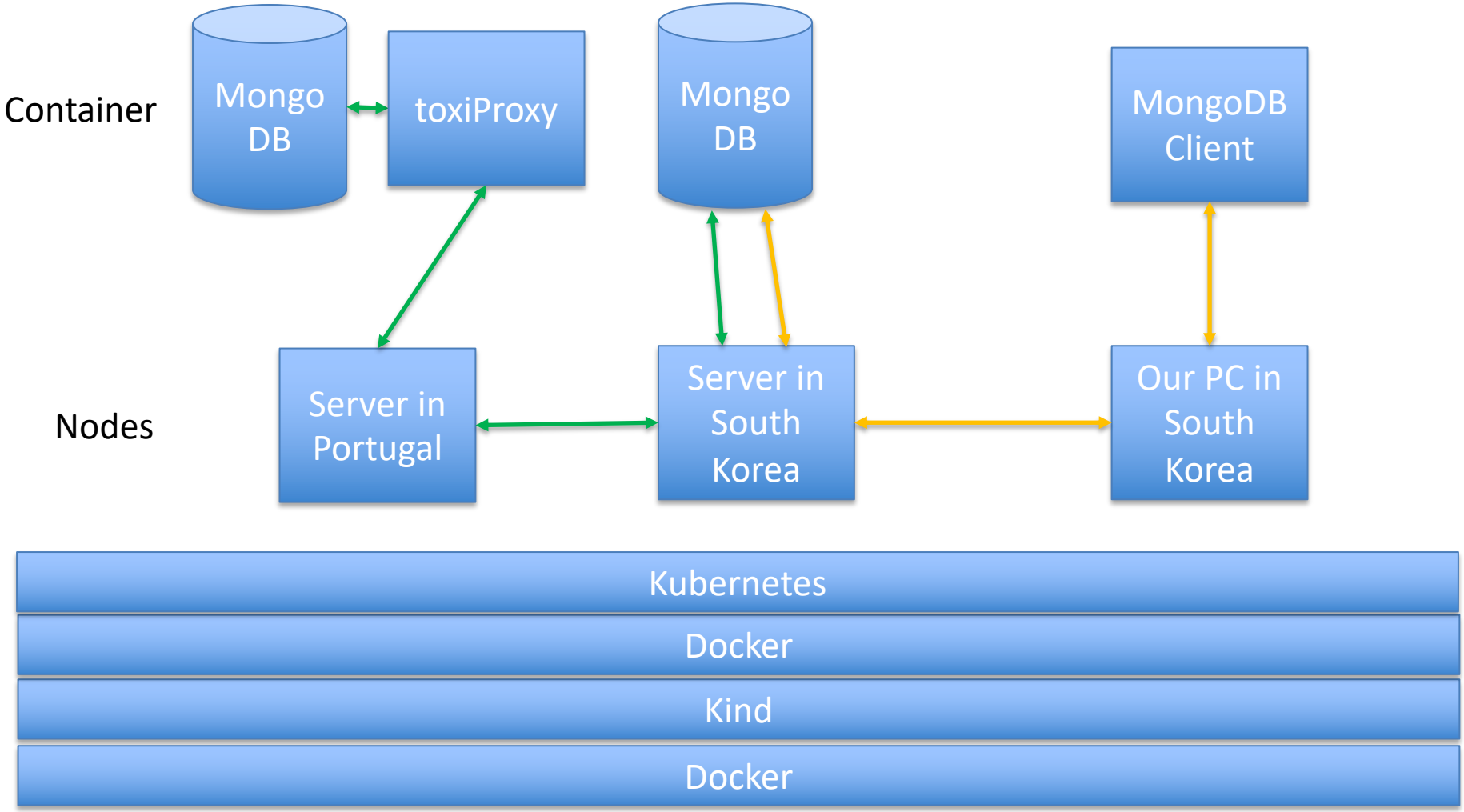
The real configuration on your machines



The real configuration on your machines



Behind the scene
Kind: virtualization over virtualization



Starting point

1. Docker installed
2. Kind downloaded and installed

KubeCTL

Kubectl is a command line interface for running commands against Kubernetes clusters

kubectl [command] [TYPE] [NAME] [flags]

kubectl get pod pod1

Create a virtual cluster with Kind

1. Move files, open a terminal and go to the folder where you extracted the files
2. Create virtual cluster

```
kind create cluster --name my-cluster --config my-cluster-config.yaml
```

Check cluster and export variable

3. Check the cluster

```
kind get clusters
```

4. Export variable (to allow kubectl to point to the new cluster)

```
export KUBECONFIG="$(kind get kubeconfig-path --name="my-cluster")"
```

Check created nodes

5. Checks if (when) nodes are ready

```
kubectl get nodes -w
```

Load images in virtual nodes

6. Load an image that contains ToxiProy in all Kubernetes Nodes

```
kind --name=my-cluster load image-archive toxi-image.tar --loglevel=trace
```

7. Load a custom image that contains the database (MongoDB) in all Kubernetes Nodes

```
kind --name=my-cluster load image-archive mongo-custom-image.tar --loglevel=trace
```

Deploy pods in nodes

8. Deploy pods

```
kubectl apply -f ./yaml_files
```

9. Check status of pods

```
kubectl get pods -w
```

Optional steps

Check ToxiProxy started

```
kubectl logs -c toxi-container <PODNAME-MONGO1> | grep "Proxy successfully created"
```

Check MongoDB container started

```
kubectl logs -c mongo-container <PODNAME-MONGO1> | grep "imported 1000 documents"
```

Query Mongo database

10. Query the slow database

```
kubectl get pods
```

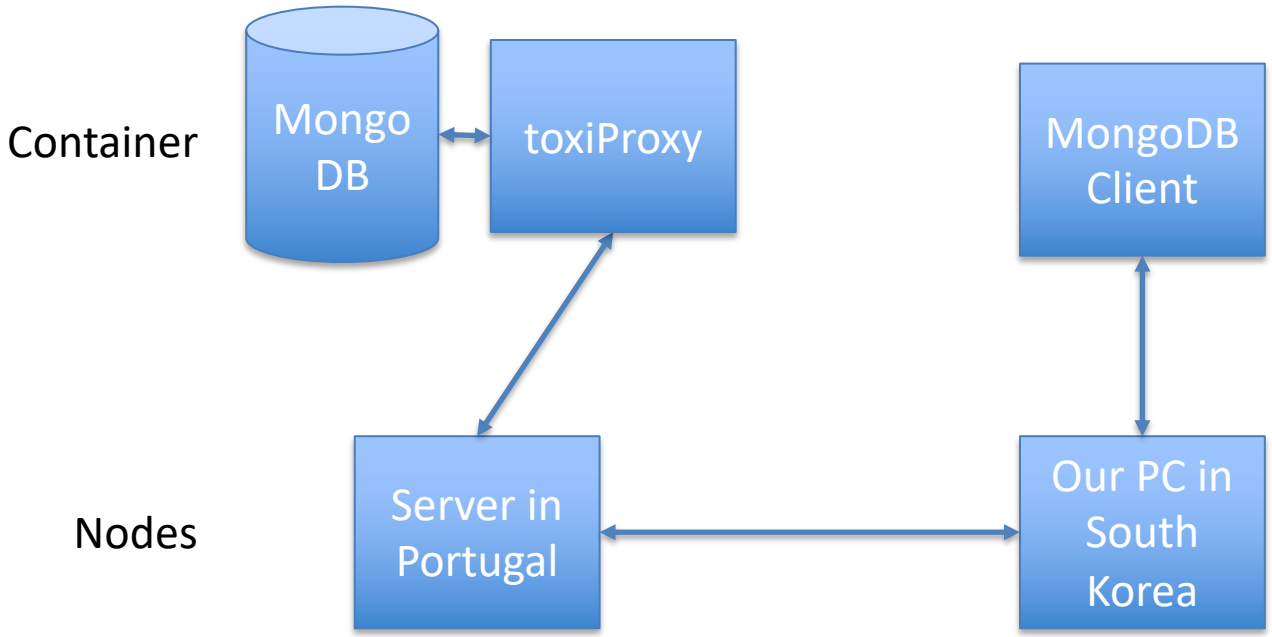
```
kubectl exec -ti <PODNAME-MONGO-CLIENT > -- bash
```

```
mongo --host=mongo-master --eval="db = db.getSiblingDB('universities');
db.students.find().pretty();"
```

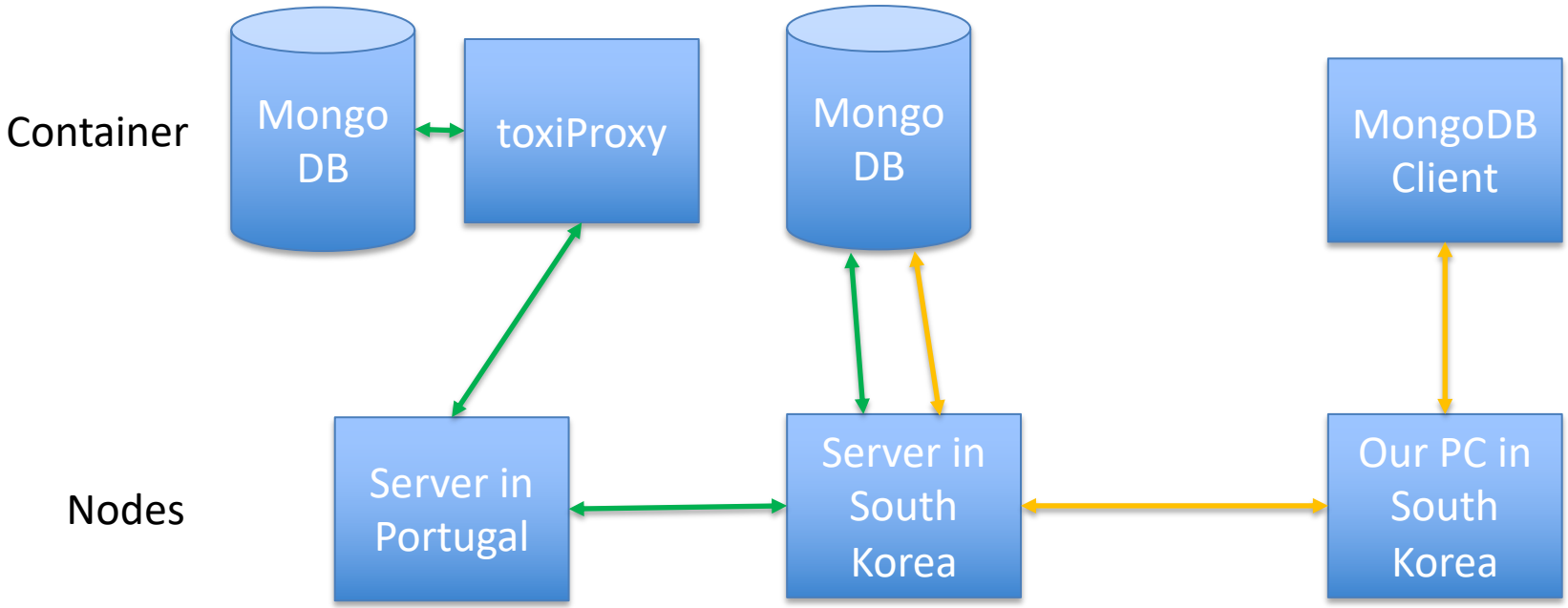
```
exit
```

The database is slow!

Original configuration



Target configuration



Move data

11. Move data

```
kubectl get pods
```

```
kubectl exec -ti <PODNAME-MONGO1> -c mongo-container -- mongo --port=9000
```

```
rs.add( { host: "mongo-slave:27017", priority: 0, votes: 0 } )
```

```
rs.status()
```

```
exit
```

Query Mongo database

12. Query the new database

```
kubectl get pods
```

```
kubectl exec -ti <PODNAME-MONGO-CLIENT> -- bash
```

```
mongo --host=mongo-slave --eval="rs.slaveOk(); db =
db.getSiblingDB('universities'); db.students.find().pretty();"
```

Is it faster??

Cleanup

13. Cleanup Kubernetes Environment

```
kubectl delete deployment mongo-client mongo1-toxi mongo2
```

```
kubectl delete service mongo-master mongo-slave
```

```
kubectl delete secret mongo-secret toxi-secret
```

Take down virtual cluster

14. Delete the virtual cluster

```
kind delete cluster --name my-cluster
```

Icons

	Icons made by Freepik at https://www.flaticon.com/authors/freepik

Bibliography

1. https://github.com/vaquarkhan/vaquarkhan/wiki/Difference-between-scaling-horizontally-and-vertically