Microservices

何李石 @ikbear
helishi@qiniu.com

Unix Philosophy

"Do One Thing and Do It Well"

```
docker images | awk '{print $3}' | sort | uniq -c
```

Unix Tools (cmd)

=>

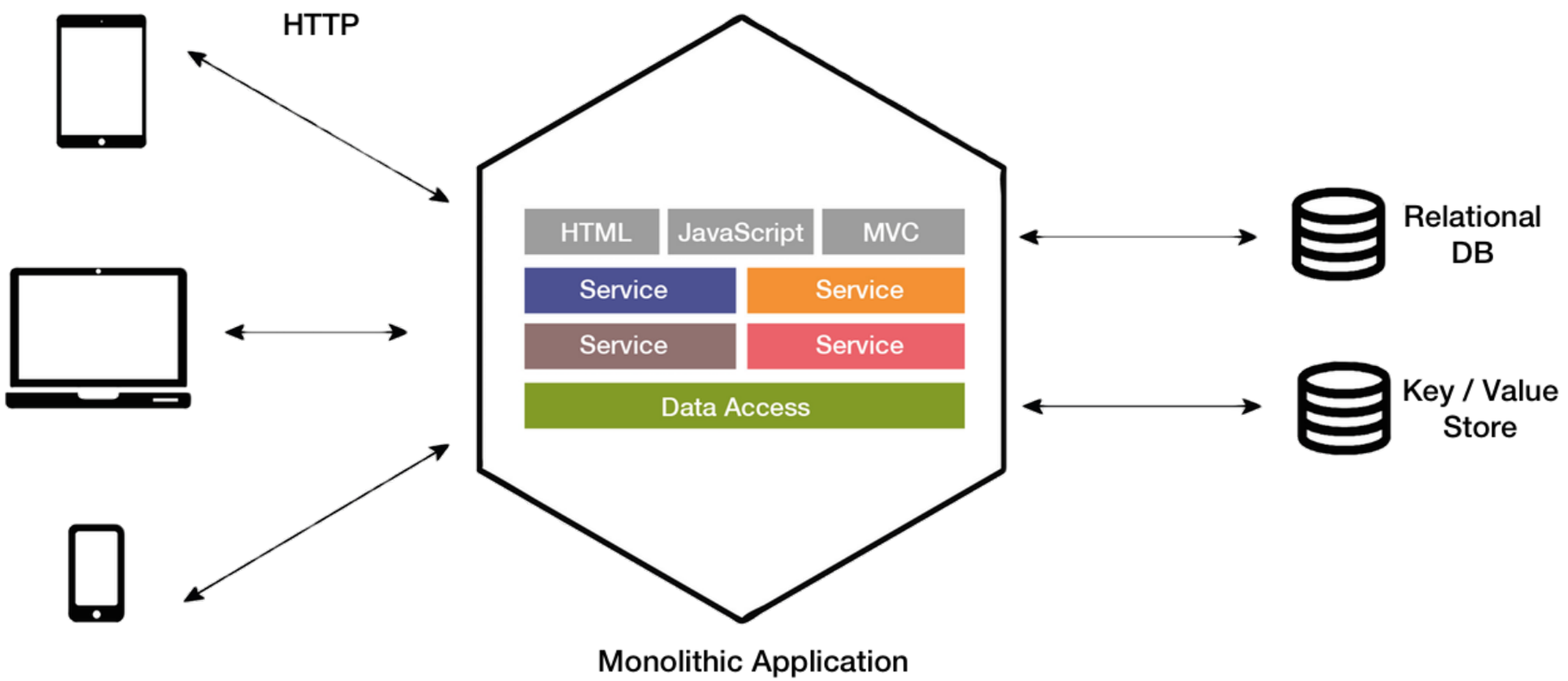
Microservices (API Endpoints)

Microservices

Small, autonomous services that work together.
— 《Building Microservices》

- Communicate between API endpoint
- Decentralized governance
- Simplify “Build => Ship => Run”

Monolithic App



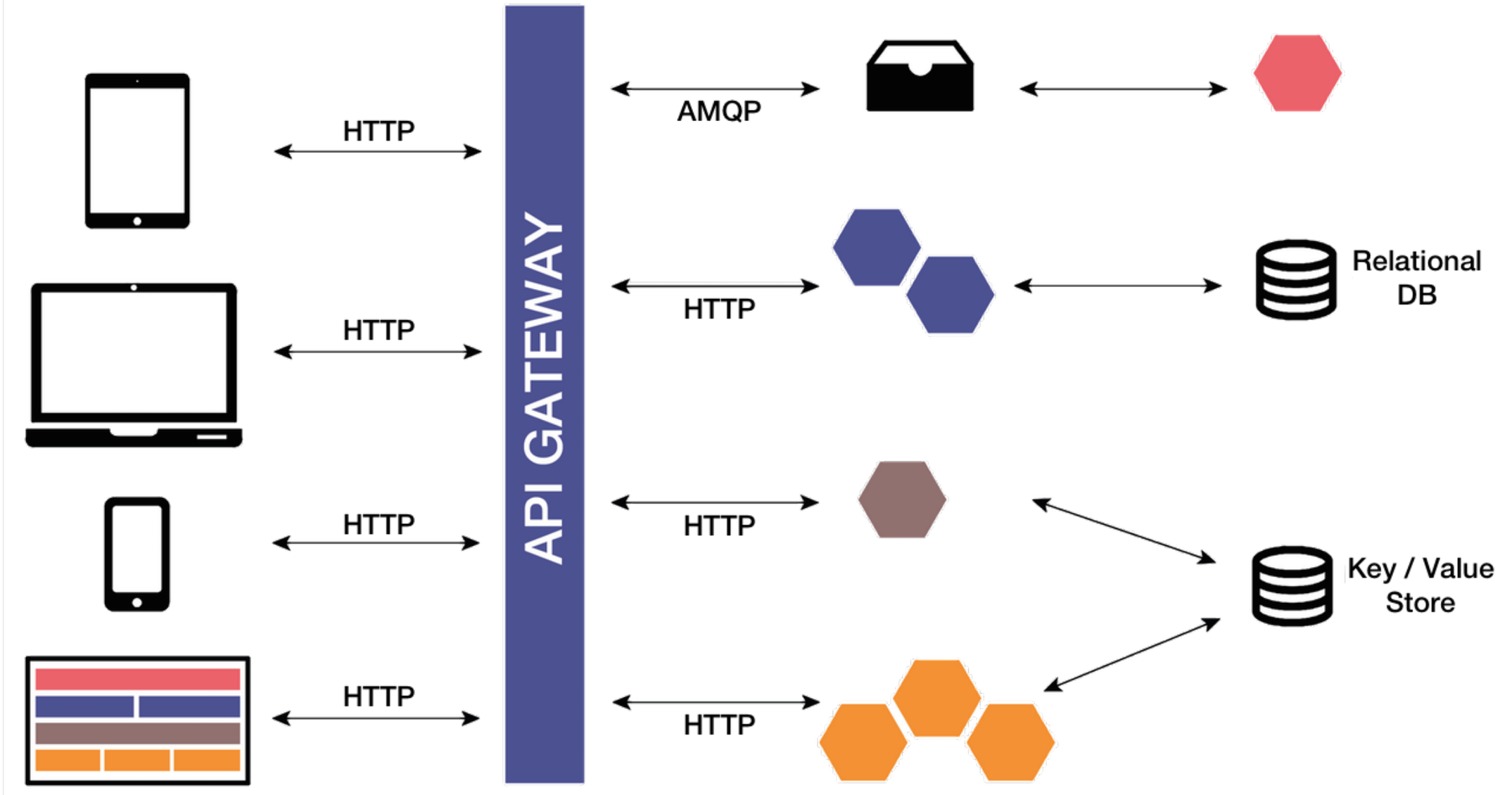
Features: All in One

- Handling all http requests (Controller)
- Authentication (Controller)
- Business logic (Controller)
- Database interaction (Model)
- Email / IM / Message Queues

Problems

- Agile, but very strict release procedure(monthly)
- A huge RoR Project
    - Various DBs (rake db:migrate RAILS_ENV=test)
    - Ruby / Rails Upgrade/Degrade（1.8 => 1.9）
    - ……
- A big team
- ……

Microservices



Features: Single Responsibility

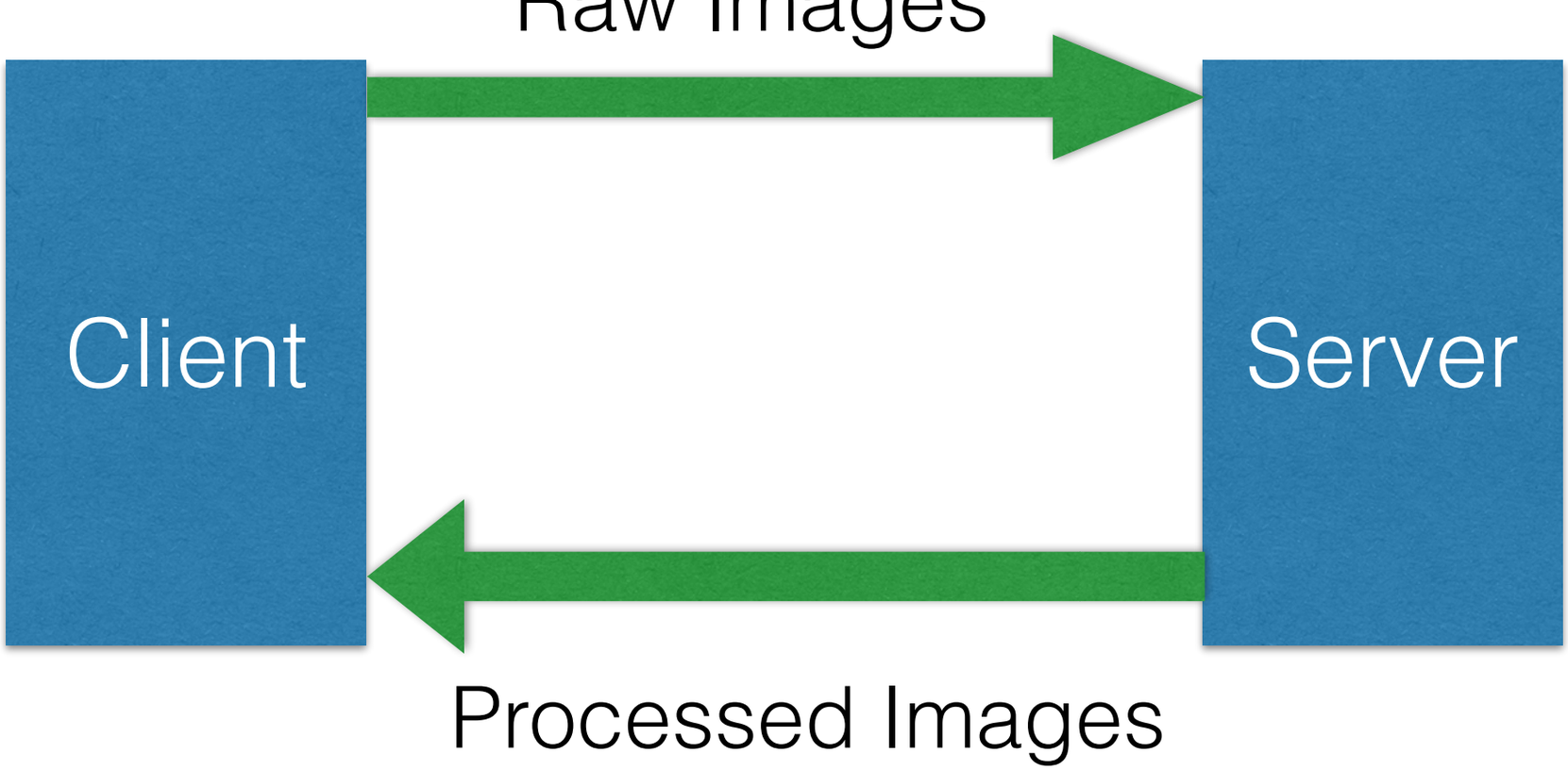
- Faster and simpler deployment and rollback
- Fault isolation: single responsibility, single failure
- Scalable services
- Single and strong DB/Cache clusters
- Small teams with diverse technology stack

Problems

- Increased network communication
- Serialization increment between micro services
- Hard to test a distributed system
- Hard to deploy so many micro services

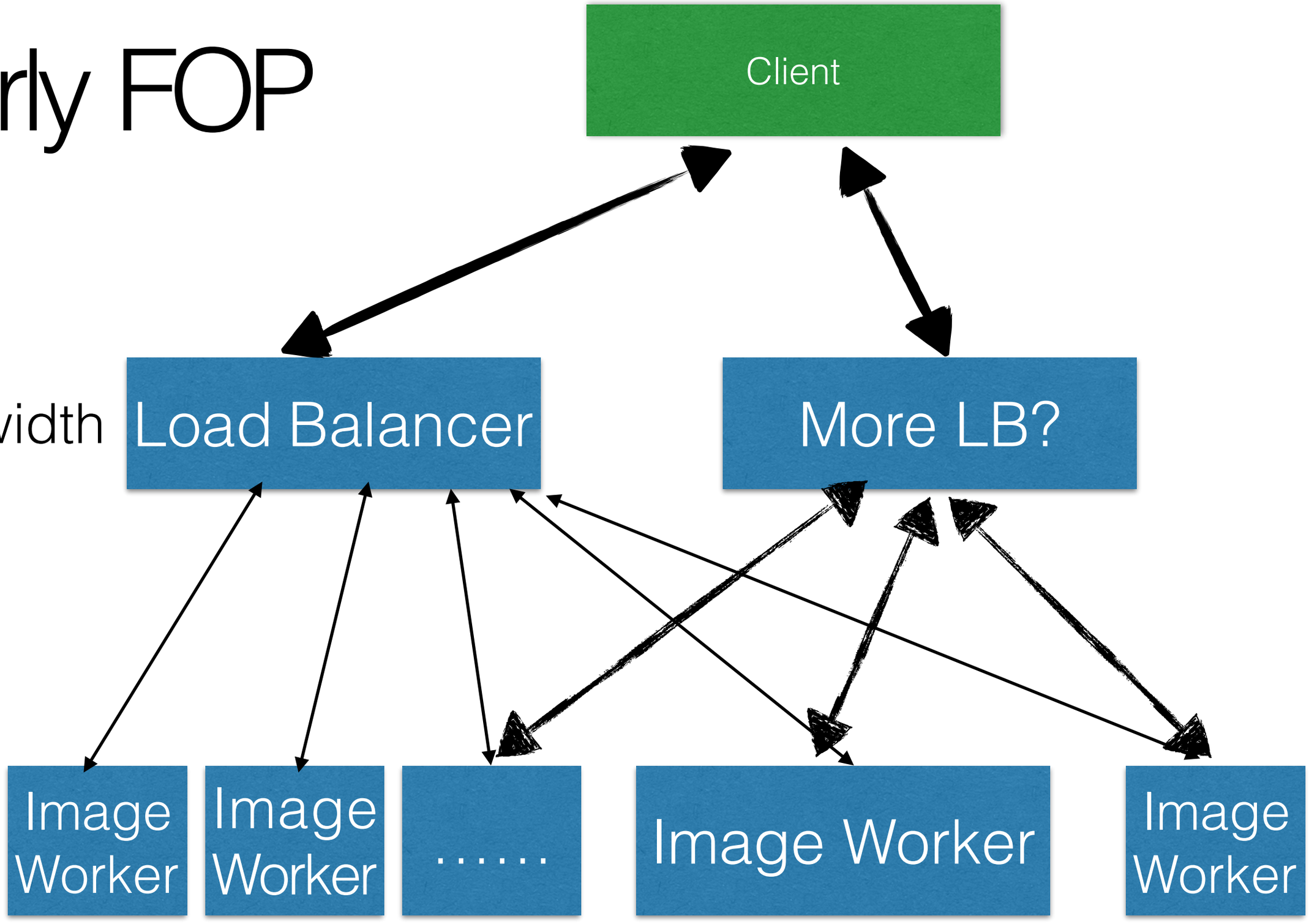
Microservices in Qiniu

FOP: File Operation



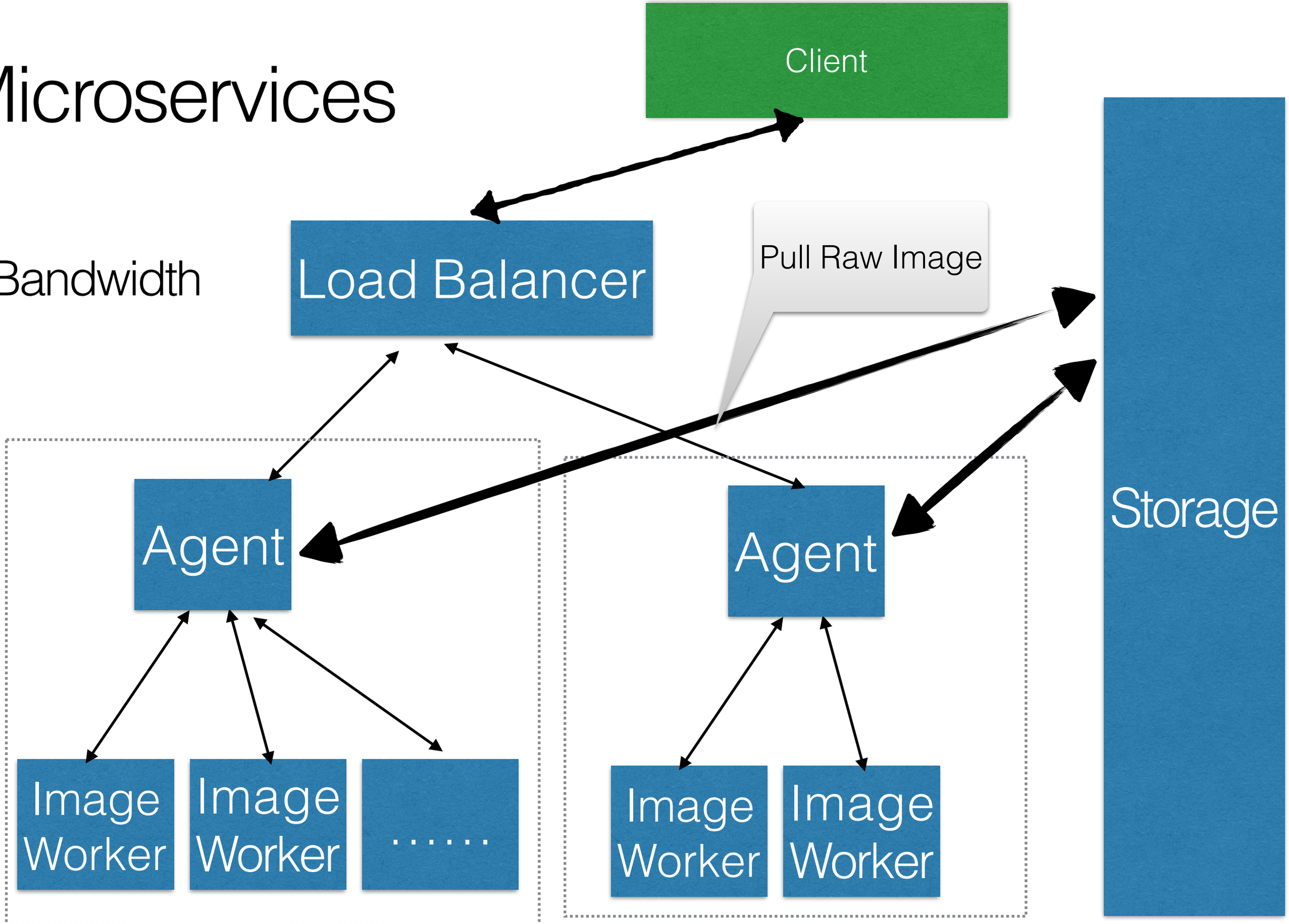
Early FOP

Bandwidth

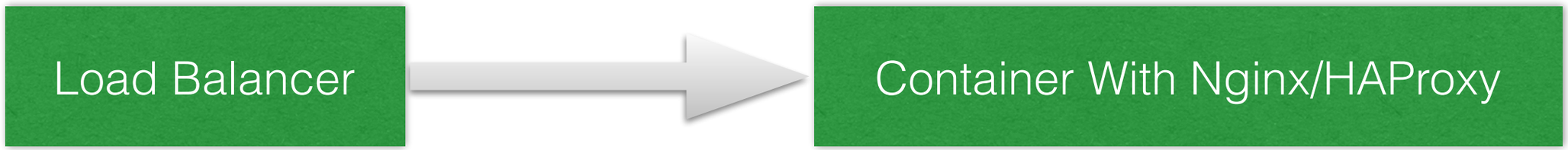


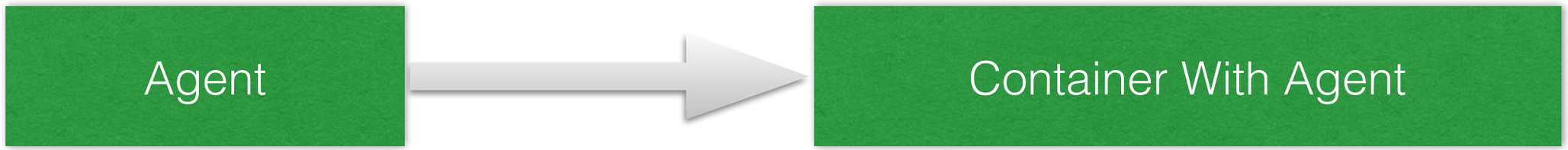
Microservices

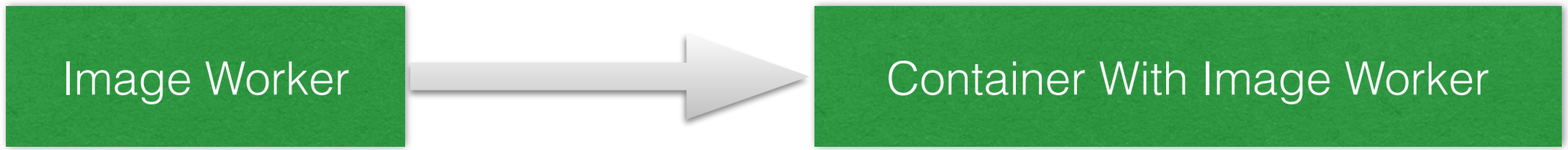
Bandwidth



Running as Containers







Everything is a Container

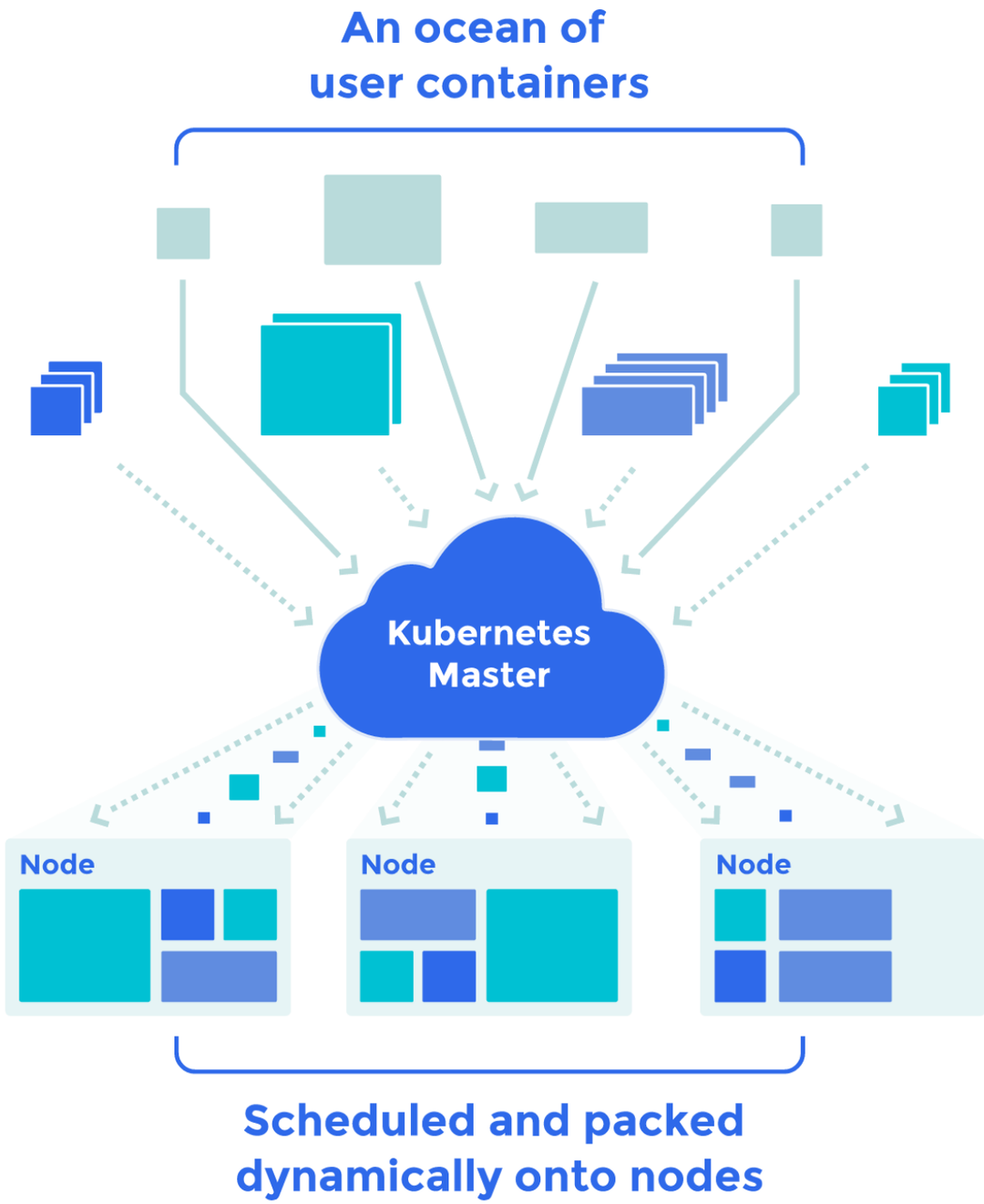
Problems Solved

- ~~Increased network communication~~ (SDN)
- ~~Serialization increment between micro services~~ (API Endpoint for HTTP, not a big problem)
- ~~Hard to test a distributed system~~ (Gray deployment, Gray testing, partly solved)
- ~~Hard to deploy so many micro services~~ (Immutable Image, Running Containers, Auto Scheduled Pods)

Why Docker?

- Isolation, Boundary & Communication Mechanisms
- Immutable Deliver & Unified Deployment (Kubernetes)

Kubernetes


An ocean of
user containers
Kubernetes
Master
Node
Node
Node
Scheduled and packed
dynamically onto nodes

Docker in Development

- git clone ikbear@repo
- docker-machine start dev
- Start Microservices:
  - ➡ ./dev-registry-up.sh
  - ➡ ./dev-up-group.sh 3

Package your Project

```
FROM golang:1.5
MAINTAINER helishi@qiniu.com

# Install apt based dependencies.
RUN apt-get update && apt-get install -y \
  build-essential \
  nodejs

# Configure the main working directory. This is the base
# directory used in any further RUN, COPY, and ENTRYPOINT
# commands.
RUN mkdir -p /app
WORKDIR /app

# Copy the main application.
COPY . ./

# Expose port 3100 to the Docker host, so we can access it
# from the outside.
EXPOSE 3200

# The main command to run when the container starts. Also
# tell the dev server to bind to all interfaces by default.
CMD ["make", "web"]
```

- ➡ docker build -t demo .
- ➡ docker run -it demo
- ➡ docker export c5f7b5ff7c6d | gzip > dev.tar.gz
- ➡ gzcat dev.tar.gz | docker import - dev

Pull and Push(like git)

- ➡ docker pull ubuntu:latest
- ➡ docker tag ubuntu:dev 192.168.99.100:5000/ikbear/ubuntu-dev:latest
- ➡ docker push 192.168.99.100:5000/ikbear/ubuntu-dev:latest #share between buddies

Thanks and Q&A