cphbusi

COPENHAGEN BUSINESS ACADEMY

DevOps part 3/4: Scaling & deployment

Jens Egholm Pedersen
<jeep@cphbusiness.dk>

Learning how to learn

- Meta-cognition
  - Think about how you think
  - Dunning-Kruger effect
- Use what we have prepared for you
- Continuous feedback
  - Where would you like to be when this course ends?
  - Keep evaluating yourself
  - … and be honest!

See also: Dunning-Kruger effect, Metacognition improves your grade!

Dev + Ops

- We expect you to develop your system
- And we expect you to keep an eye on them
- Do this for your own sake
  - You would like to learn about operating systems now
  - **Not** when you crash your production system for the first time
  - Trust me on this

Recap

- Service-level agreement (SLA)
- Monitoring
- Logging
- Post-mortem analysis

A word on bottlenecks

- All design problems can be solved by a layer of indirection
- No performance problem can be solved by removing a layer of indirection

- Bottlenecks appear in any good design

- Three problems:
  - Single point of entry
  - Single point of failure
  - Congestion

See also: Profiling in software

The problem with bottlenecks

- Single point of entry
  - How can we solve this?
- Single point of failure
  - How can we solve this?
- Congestion
  - How can we solve this?

Scaling the bottlenecks

- Single point of entry
  - Content delivery network
- Single point of failure
  - Redundancy
- Congestion
  - Scaling!

Kubernetes

- “Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications” - Google

- Powerful
- Complex

- We won’t be using this

See also: Docker swarm and Kubernetes compared

Docker swarm mode

- Based on swarmkit
  - "A toolkit for orchestrating distributed systems at any scale"
- Orchestration
  - Manages nodes and services
- Scheduling
  - Resource aware task scheduling
- Security
  - We'll talk about this next week

See also: swarmkit on GitHub

Docker commands

- `docker-machine`
  - Manages Vms
- `docker swarm`
  - Manages our cluster (swarm)
- `docker service`
  - Manages the containers in the swarm

Docker swarm: Concepts

- Nodes
  - A VM instance participating in a swarm
  - Worker nodes
  - Manager nodes
- Services and tasks
  - A collection of tasks to be executed on a node (= container)
  - Replicated services (fixed instances)
  - Global services (one instance per node)
- Load balancing

See also: Swarm mode key concepts

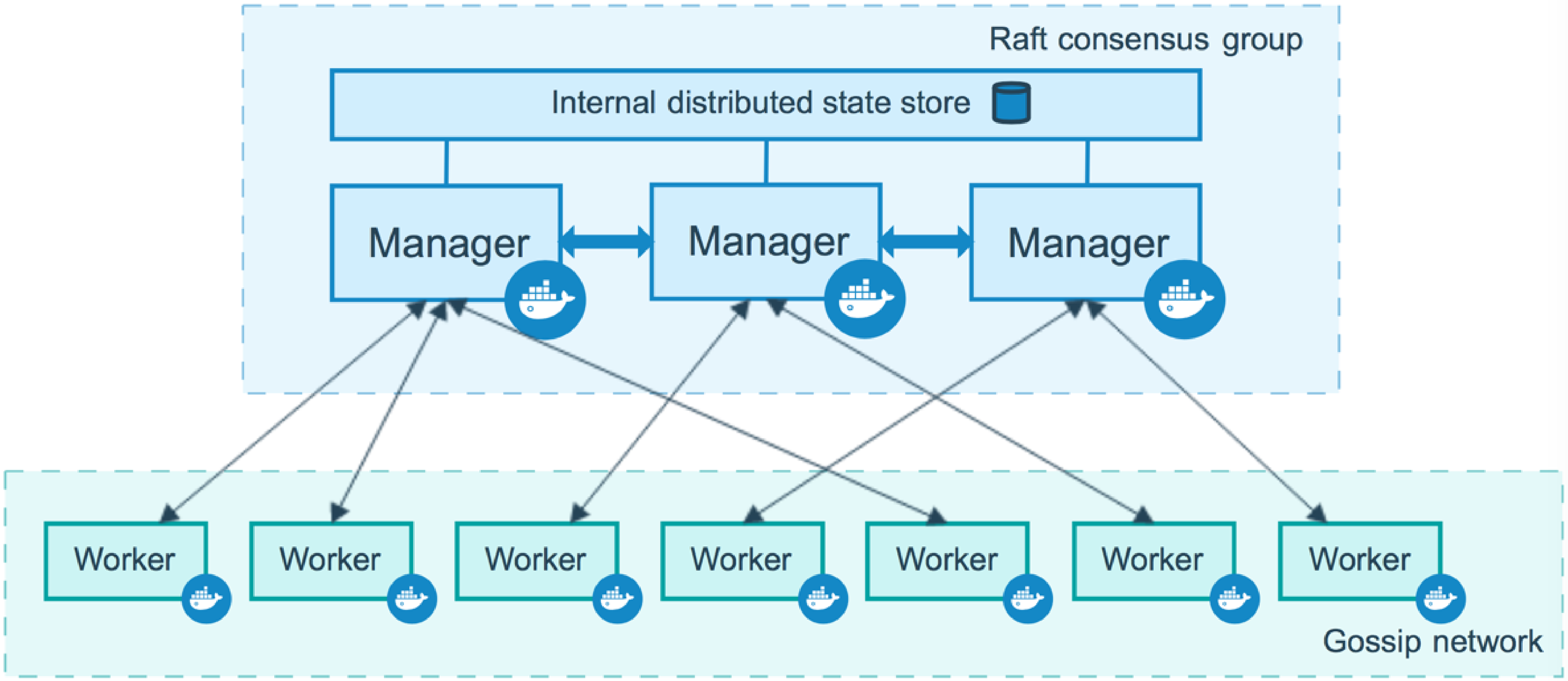
Docker swarm: Nodes

- Worker nodes
  - Contain a number of services
  - Need at least one manager node
- Manager node
  - Maintains cluster state and schedules services

See also: Swarm nodes

cphbusiness

Docker swarm: Nodes



See also: Swarm mode key concepts

Introduction to Docker machine

- Can create and manage virtual machines for us

- Input parameters
  - Driver
    - Virtualbox, digitalocean, aws, etc.
  - Container name
  - Optional: Driver parameters
    - DO access token, DO region, etc.

See also: Docker machine documentation

Introduction to Docker machine

- Can create and manage virtual machines for us

- Locally with virtualbox

```
docker-machine create --driver virtualbox mybox
```

- Remotely with Digital Ocean

```
docker-machine create --driver digitalocean
   --digitalocean-access-token=xxx
   --digitalocean-region ams3 mybox
```

See also: Docker machine documentation

Introduction to Docker machine

- Can create and manage virtual machines for us

- Listing the available machines

```
docker-machine ls
```

- Getting the ip of a machine

```
docker-machine ip mybox
```

- Removing a machine

```
docker-machine rm mybox
```

- Ssh’ing into the available machines

```
docker-machine ssh mybox
```

See also: Docker machine documentation

Docker swarm: Managers

- Manager node
  - Maintains cluster state and schedules services

- Creating a swarm
  - **On your swarm manager!!**

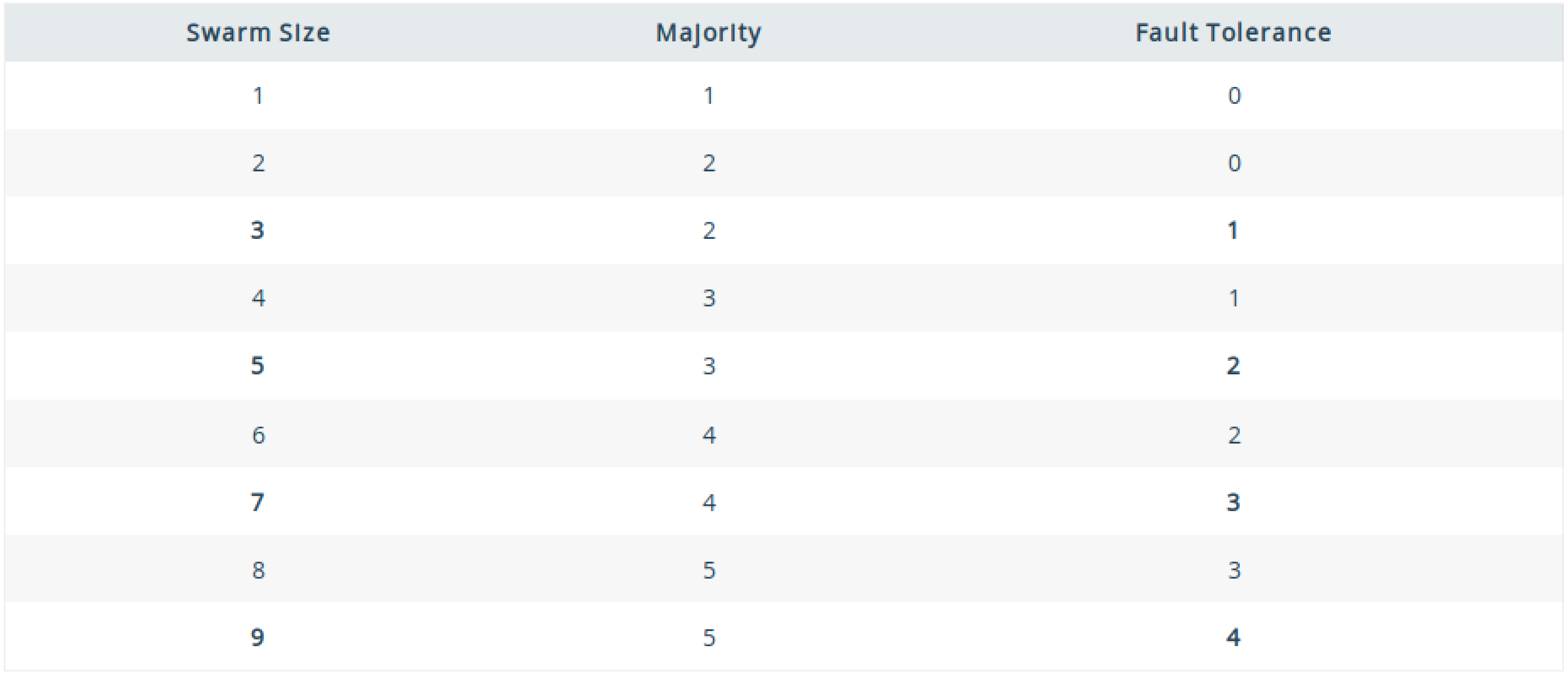
    ```
    docker swarm init --advertise-addr x.x.x.x
    ```

- Now we have a cluster!
  - With one node…

See also: Creating a swarm

Docker swarm: Managers

- Fault tolerance



Swarm Size	Majority	Fault Tolerance
1	1	0
2	2	0
3	2	**1**
4	3	1
5	3	**2**
6	4	2
7	4	**3**
8	5	3
9	5	**4**

See also: Docker swarm admin guide, Raft consensus algorithm

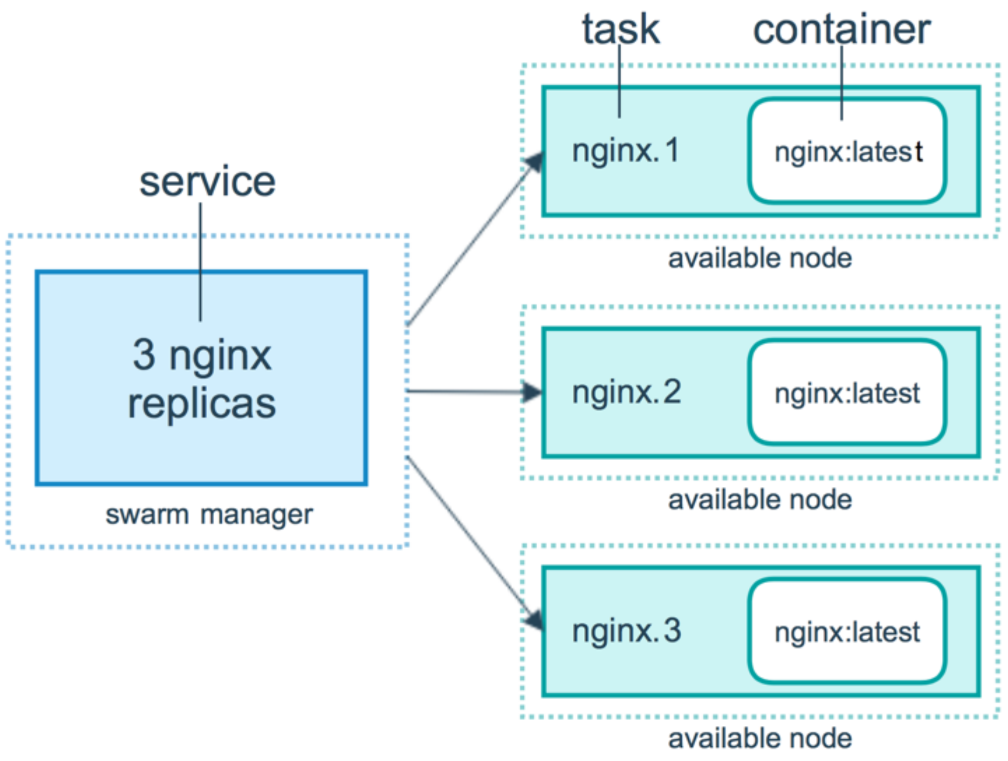
Docker swarm: Services

- A service your cluster provides
  - Typically a container that can scale
- Input parameters
  - Mode: Replicated vs. global
  - Name
  - Network
  - Publish (external port)
  - Container id

cphbusiness

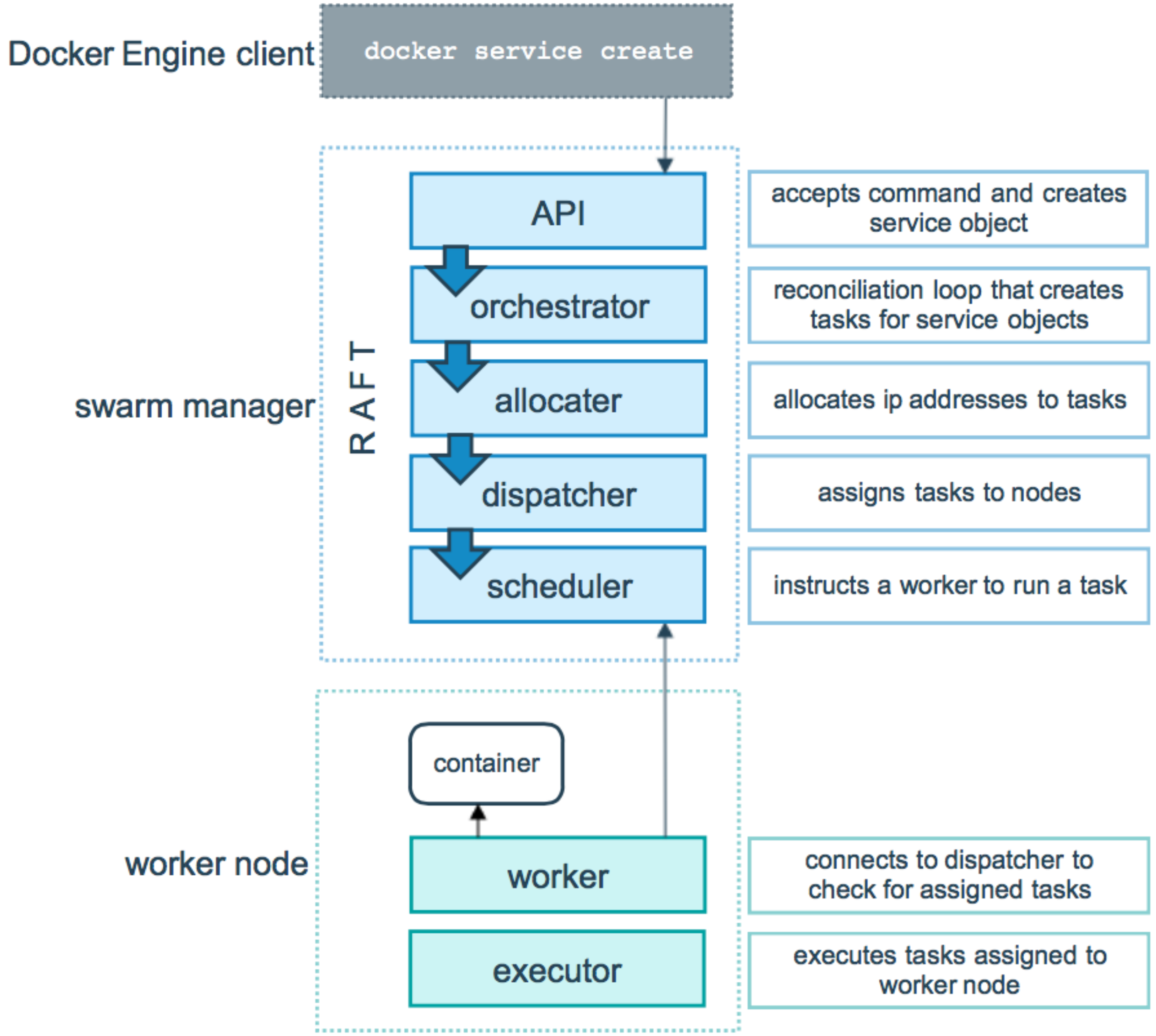
Docker swarm: Services

- A service your cluster provides
  - Typically a container that can scale



cphbusiness

Docker swarm: Services


Docker Engine client
docker service create
swarm manager
RAFT
API
orchestrator
allocater
dispatcher
scheduler
accepts command and creates service object
reconciliation loop that creates tasks for service objects
allocates ip addresses to tasks
assigns tasks to nodes
instructs a worker to run a task
worker node
container
worker
executor
connects to dispatcher to check for assigned tasks
executes tasks assigned to worker node

cphbusiness

`docker service`

- **Run this on your manager node!**
- Equivalent to `docker run`
- Input parameters
  - Mode: Replicated (default) vs. global
  - Name
  - Network
  - Publish (external port)
  - Container id

```
docker service create
   --name webserver
   --publish 80:80 nginx
```

docker service

- **Run this on your manager node!**
- `docker service ls`
- `docker service ps webserver`
- `docker service inspect webserver`

Docker swarm: Scaling

- The beauty of docker
  - Containerize once, run anywhere!

- `docker service scale webserver=5`

- `docker service ls`

- `docker service inspect webserver`

See also: Scaling services

The problem with bottlenecks

- Single point of entry
  - Content delivery network
- Single point of failure
  - Redundancy
- Congestion
  - Scaling!

Recap

- Docker-machine
- Docker swarm
- Docker service

- Coming up:
  - Upgrade strategies
  - Adding/removing nodes
  - Load balancing
  - Service discovery

Upgrade strategies

- Blue-green
  - Two identical environments
  - Only one is live at any time
- Canary
  - Deploy to a small group first, then deploy to the rest
- Rolling
  - Deploy in rolling iterations

See also: Colorful deployments

cphbusiness

Docker swarm: Rolling updates

- What Docker swarm does
  1) Stop the first task
  2) Schedule update for the stopped task
  3) Start the container for the updated task
  4) If the update to a task returns RUNNING, wait for the specified delay period then start the next task
  5) If, at any time during the update, a task returns FAILED, pause the update
- PS: You need at least two replicas!

See also: Rolling updates

Docker swarm: Updating

- The beauty of docker
  - Containerize once, run anywhere!
  - In different versions

-
```
docker service update
       --image nginx:1.12 webserver
```

- … or in this case downgrading

See also: Scaling services

Docker swarm: Adding nodes

- Two types:
  - Managers
  - Workers

- Security
  - You shouldn’t be able to join anywhere
  - Solution: tokens (worker/manager)

- `docker swarm join --token X IP`

See also: Adding nodes

Docker swarm: Removing nodes

- Two types:
  - Managers
  - Workers

- From the worker:

```
docker swarm leave
```

- From the manager:

```
docker node update --availability drain <node>
docker swarm rm
```

See also: Draining nodes

Recap

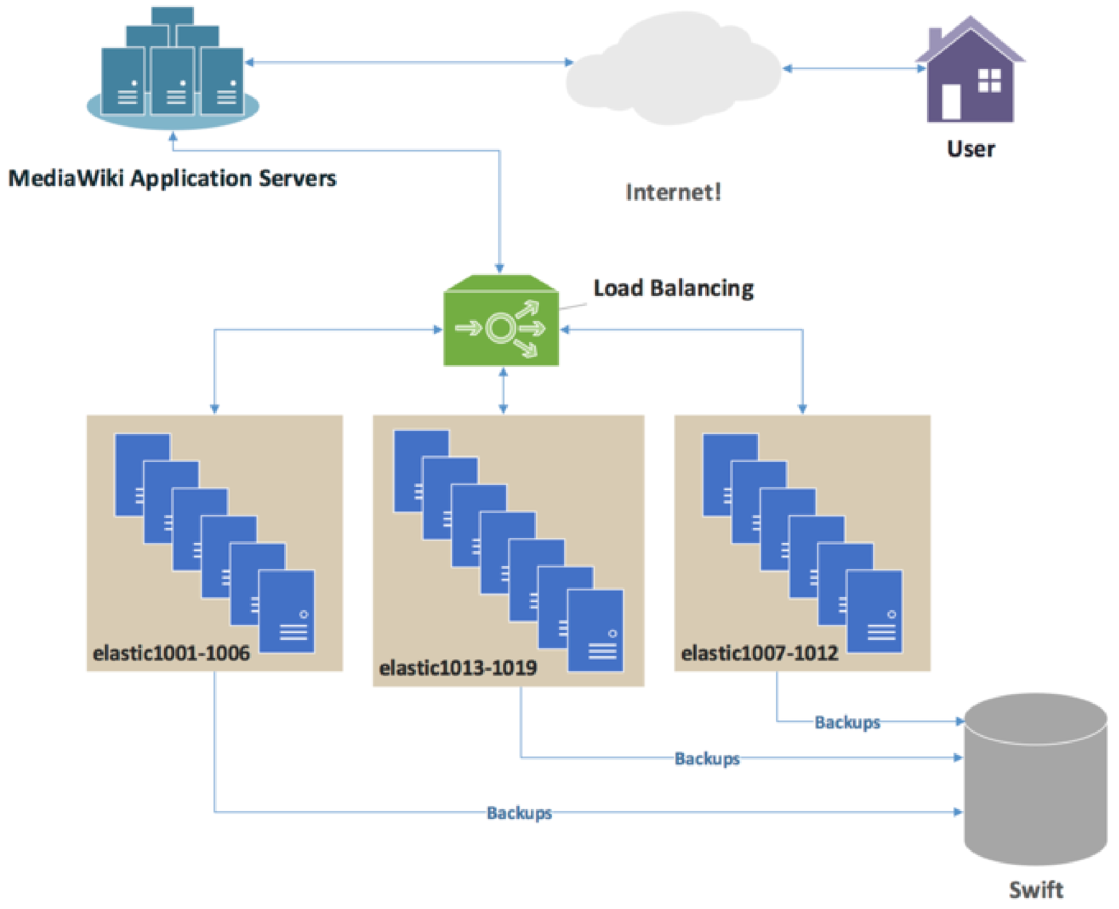
- Docker-machine
- Docker swarm
  - Creating a swarm
  - Adding/removing nodes
- Docker service
  - Upgrading services

- Coming up:
  - Load balancing
  - Service discovery

The problem with bottlenecks

- Single point of entry
  - Content delivery network
- Single point of failure
  - Redundancy
- Congestion
  - Scaling!

Load balancing



Load balancing strategies

1) Round robin delegation
    - Delegate to servers one by one
2) DNS delegation
    - Delegate by DNS zones
    - Distribute geographically
3) Client-side load balancing
    - Clients have a list of servers
    - Choose one randomly

See also: Load balancing on Wikipedia

Load balancing scenarios

- Peak capacity
  - You’re Facebook, users gets home from work
- Reduced capacity
  - You’re Facebook, users are sleeping
- DDOS

Other load balancing strategies

- Caching
- Compression (less data)
- Multi-threading
- Blocking
  - IP regions
  - Single targets

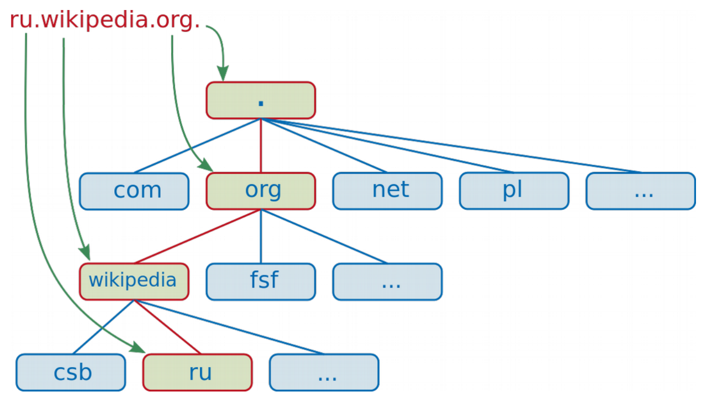
Load balancing software

- Nginx, haproxy, apache server
  - Many types, same purpose

- “Reverse proxy”
  - Not a proxy for connections *out* but connections *in*

Domain name

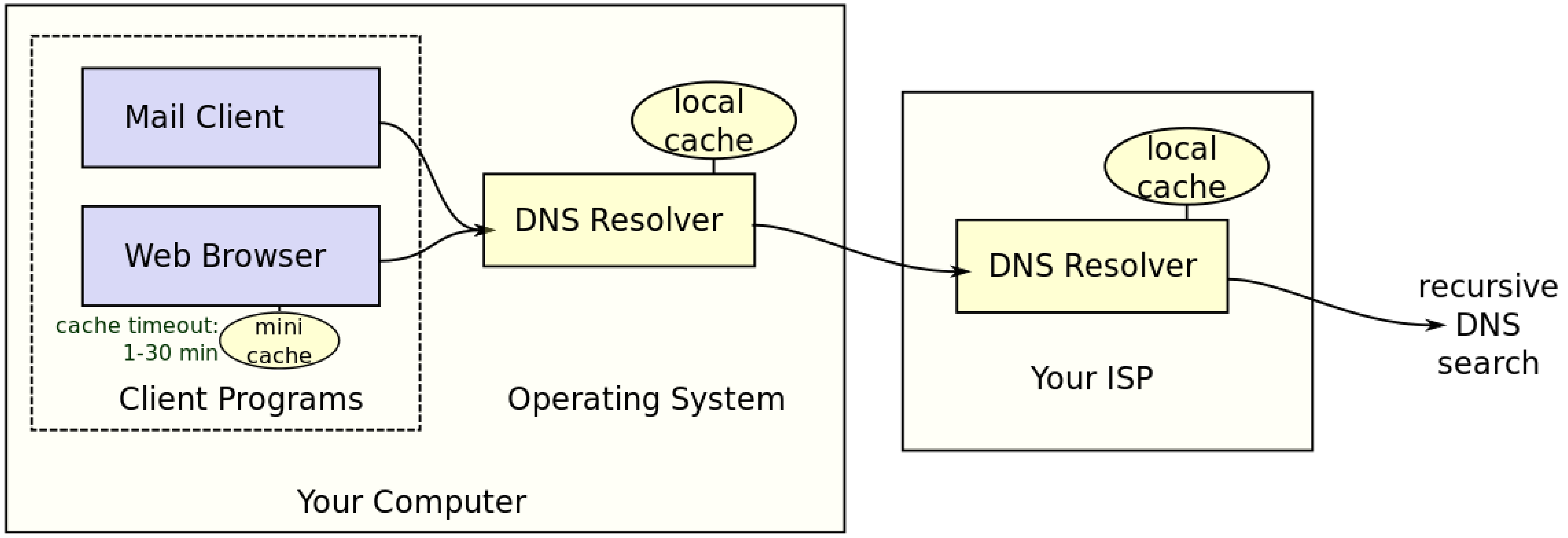
- Read from right to left
- Fully qualified domain name (FQDN)
  - Begins with the root domain "."



1) Root domain
2) Top-level domain
3) Second or third level

See also : Network Address Translation on Wikipedia

DNS Resolving



DNS scaling

- Domain name system
  - IP addresses cached on DNS servers

- DNS A record with multiple entries:
  1) Request: 0.0.0.1
  2) Request: 0.0.0.2
  3) Request: 0.0.0.3…

- Your DNS provider can route your traffic
  - Example: DDOS

See also: DNS on wikipedia, List of DNS record types

DNS scaling

- Domain name system
  - IP addresses cached on DNS servers

- DNS A record with multiple entries:
  1) Request: 0.0.0.1
  2) Request: 0.0.0.2
  3) Request: 0.0.0.3…

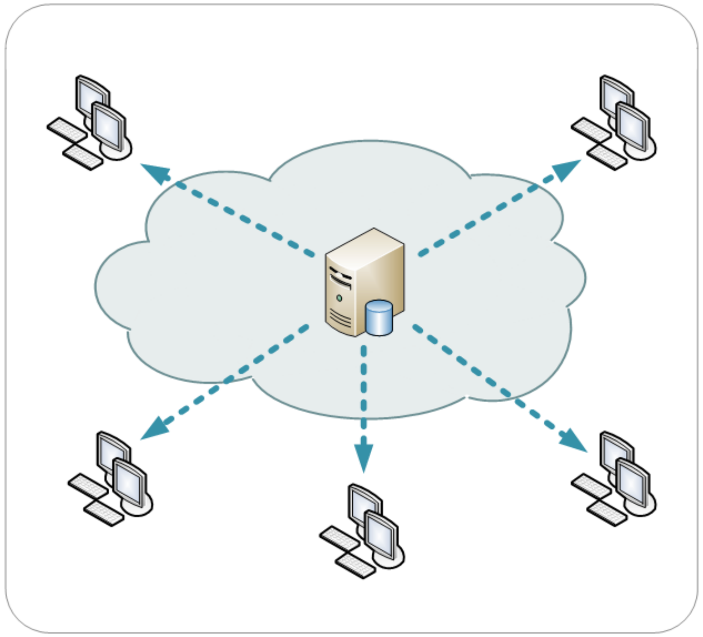
- Your DNS provider can route your traffic
  - Example: DDOS

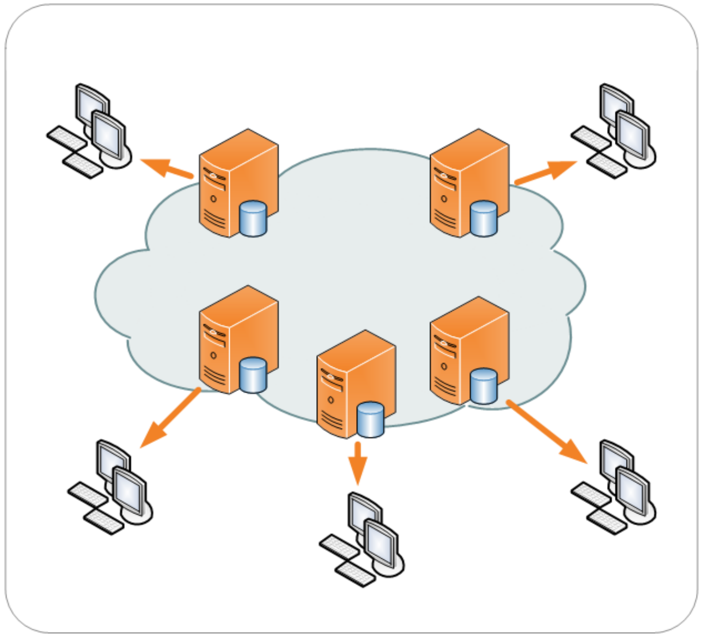
See also: DNS on wikipedia, List of DNS record types

cphbusiness

Content delivery network (CDN)

- "Geographically distributed network of proxy servers and their data centers"





See also: CDN on Wikipedia

Load balancing strategies

1) Round robin delegation
   - Delegate to servers one by one
2) DNS delegation
   - Delegate by DNS zones
   - Distribute geographically
3) Client-side load balancing
   - Clients have a list of servers
   - Choose one randomly

See also: Load balancing on Wikipedia

Docker swarm load balancing

- By default: ingress network
- The ingress network is a special overlay network that facilitates load balancing among a service's nodes.
- When any swarm node receives a request on a published port, it hands that request off to a module called IPVS. IPVS keeps track of all the IP addresses participating in that service, selects one of them, and routes the request to it, over the ingress network.
- The ingress network is created automatically when you initialize or join a swarm.

See also: Docker swarm networking

Recap

- Docker-machine
- Docker swarm
  - Creating a swarm
  - Adding/removing nodes
- Docker service
  - Upgrading services
- Docker network
  - Ingress: load balancing

- Coming up:
  - Service discovery

Service discovery

- Automatically discover machines providing the same service
- DNS A record with multiple entries:
  1) Request: 0.0.0.1
  2) Request: 0.0.0.2
  3) Request: 0.0.0.3…

See also: DNS-SD on Wikipedia, RFC2782

Service discovery in docker

- Overlay networks
  - manage communications among the Docker daemons participating in the swarm

```
docker network create
       --driver overlay monitoring
```

```
nslookup tasks.docker-exporter
```

See also: Docker swarm networking

cphbusiness

Monitoring via docker-machine

- Docker-machine experimental feature
  - Inbuilt prometheus monitoring

-
  ```
  docker-machine create
        --driver virtualbox
        --engine-opt experimental
        --engine-opt metrics-addr=0.0.0.0:4999 mybox
  ```

See also: Docker metrics in Prometheus

Monitoring via docker-machine

- Docker-machine experimental feature
  - Inbuilt prometheus monitoring
- One small problem… We have to expose it on a specific docker network

```
docker
  service create
  --mode global
  --name docker-exporter
  --network monitoring
  --publish 4999
  -e IN=172.18.0.1:4999
  basi/socat:v0.1.0
```

See also: Docker metrics in Prometheus

A note on configuration

- No longer a common file system
- How to hangle configuration?

```
docker config create prometheus-config prometheus.yml
```

```
docker service create --name prometheus
   --config=prometheus-config,target=/etc/prometheus/prometheus.yml
```

See also: Docker metrics in Prometheus

cphbusiness

Monitoring via docker-machine

- Putting it all together
- We have docker-exporters and a prometheus configuration file to listen for the dns service discovery

```
docker service create
   --name prometheus -p 9090:9090
   --config src=prometheus,
      target=/etc/prometheus/prometheus.yml
   --network monitoring prom/prometheus
```

See also: Docker metrics in Prometheus

Recap

- Docker-machine
- Docker swarm
  - Creating a swarm
  - Adding/removing nodes
- Docker service
  - Upgrading services
- Docker network
  - Ingress: load balancing
- Service discovery
  - Prometheus monitoring via DNS service discovery

Next hand-in

Deadline: **22nd of November 23:59:55**

1) Create a docker swarm with at least 3 vms
2) Create your services in docker swarm
    1) At least three: Your service, Prometheus and logging (ELK)
3) Optional: Add monitoring to your docker swarm
    1) Create an overlay network
    2) Create the socat global service
    3) Reconfigure Prometheus to use DNS service discovery
        - Remember to put Prometheus on the same network!

Next hand-in

Deadline: **22nd of November 23:59:55**

Hand-in:

1)Four lines of text describing

- What Docker swarm is and why we need it
- Why it can help us to eliminate bottlenecks

2)Printout of:

- `docker node ls`
- `docker service ls`