# Tupperware:
# Containerized Deployment at FB

DockerCon 2014

Aravind Narayanan
aravindn@fb.com

facebook

## Scale makes everything harder

- Running single instance: easy
- Running at scale in production: messy and complicated

Provision machines

Machine decoms

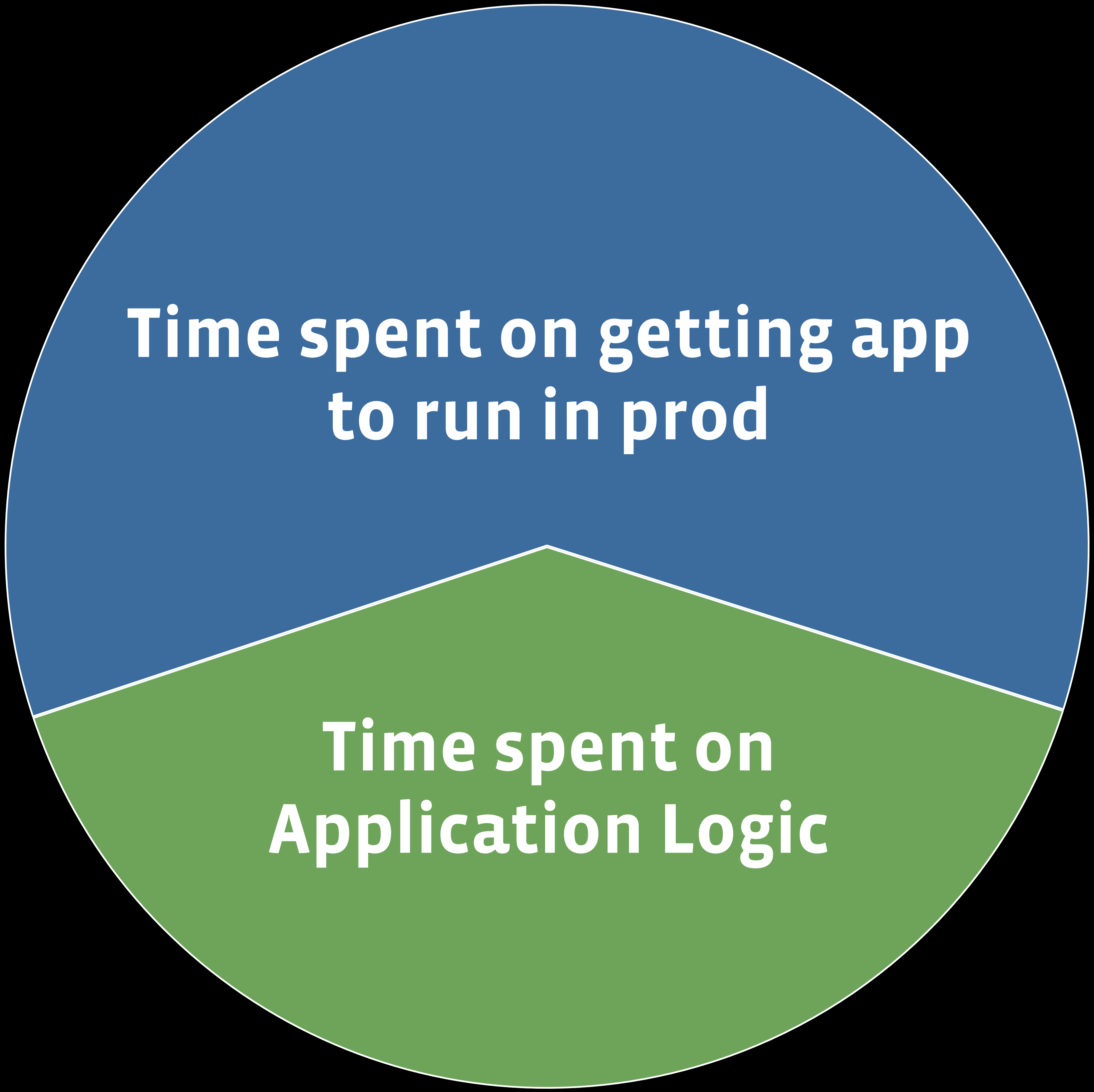
Distribute binaries

Failover

Geo-distribution

Monitoring

Daemonize process


Time spent on getting app to run in prod
Time spent on Application Logic

# Tupperware to the rescue

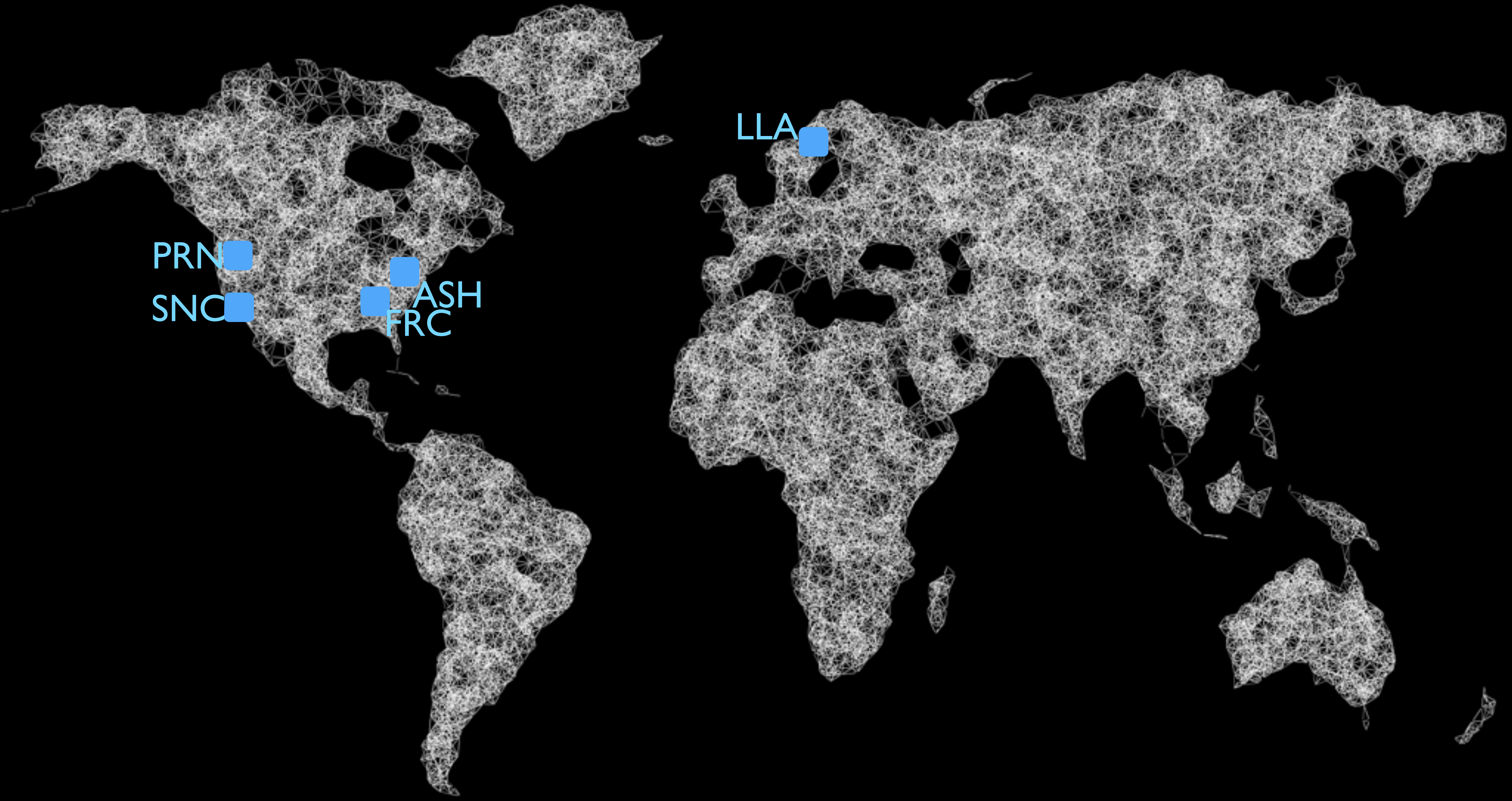
"This is my binary. Run it on X machines!"

- Engineer is hands-off
  - Doesn't need to worry about machines in prod
- Handles failover, when machines go bad
- Efficient use of infrastructure
- 300,000+ processes, spread over 15,000+ services

## Agenda

1. Architecture
2. Sandboxes
3. Ecosystem
4. Lessons learnt

# Facebook Datacenters

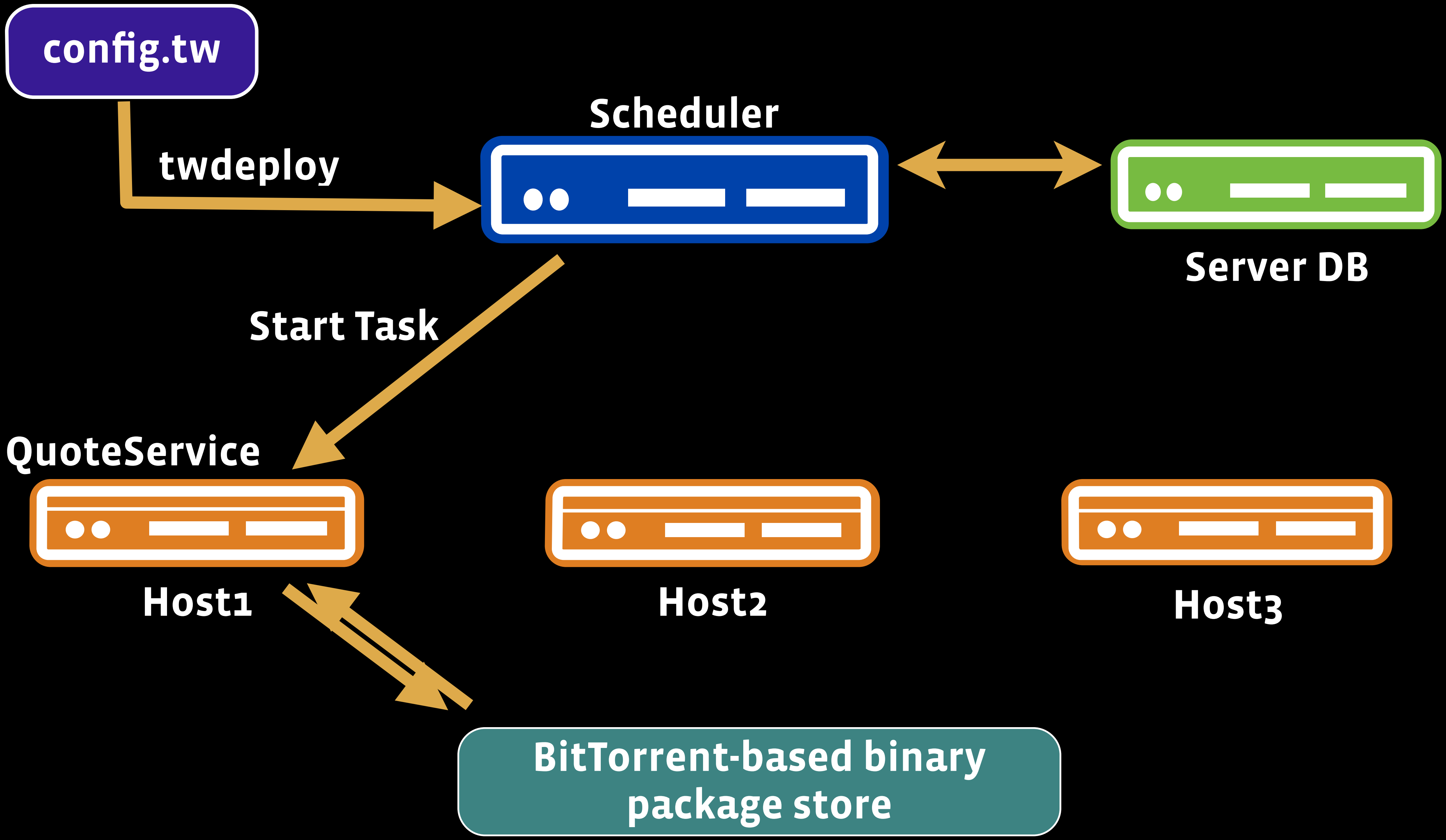


## Terminology

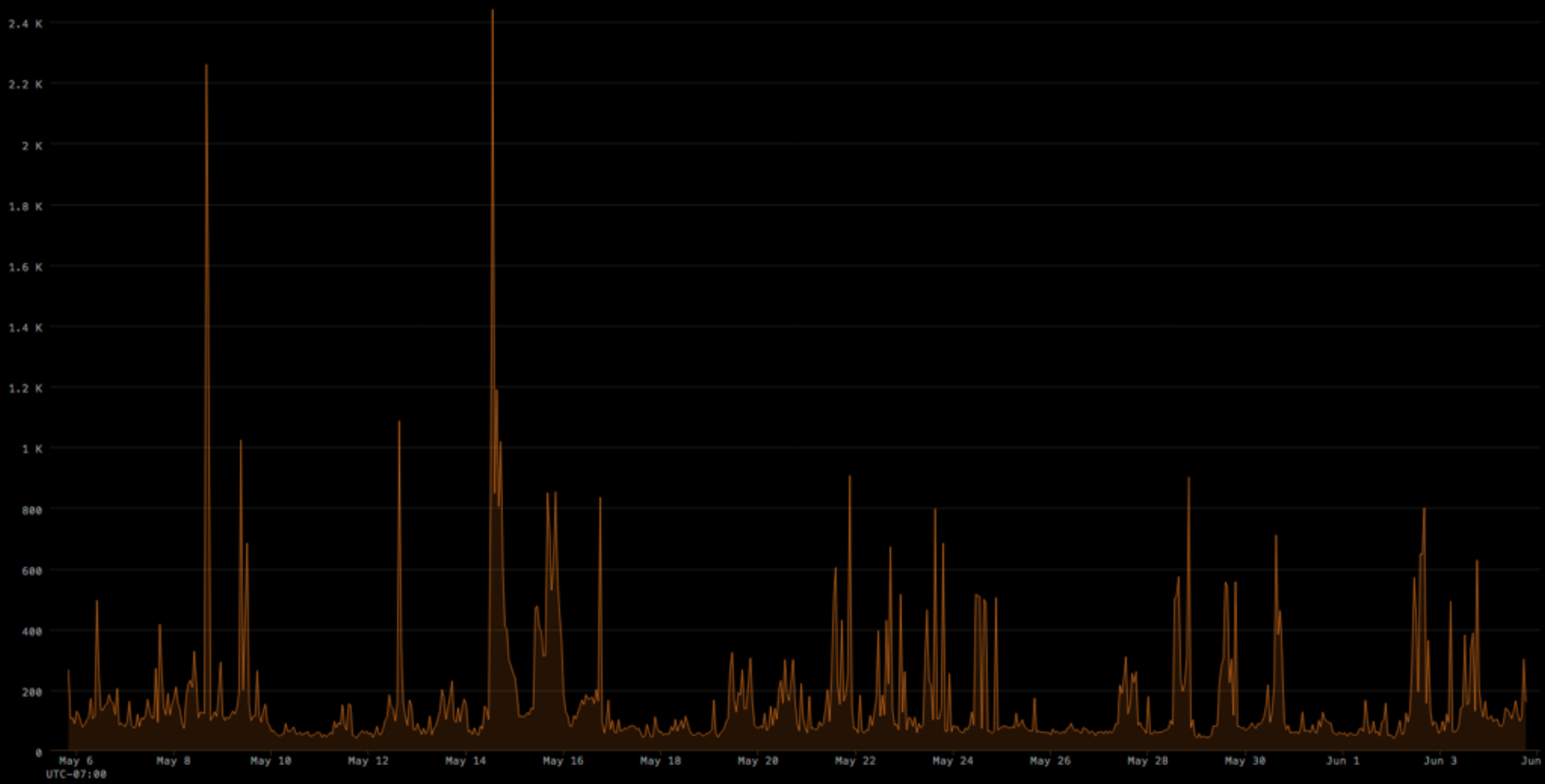
- A DC has one or more clusters
- A cluster has multiple racks
- A rack has multiple machines

- A TW job is equivalent to a service
- A job has multiple tasks, each an instance of the service

# Architecture

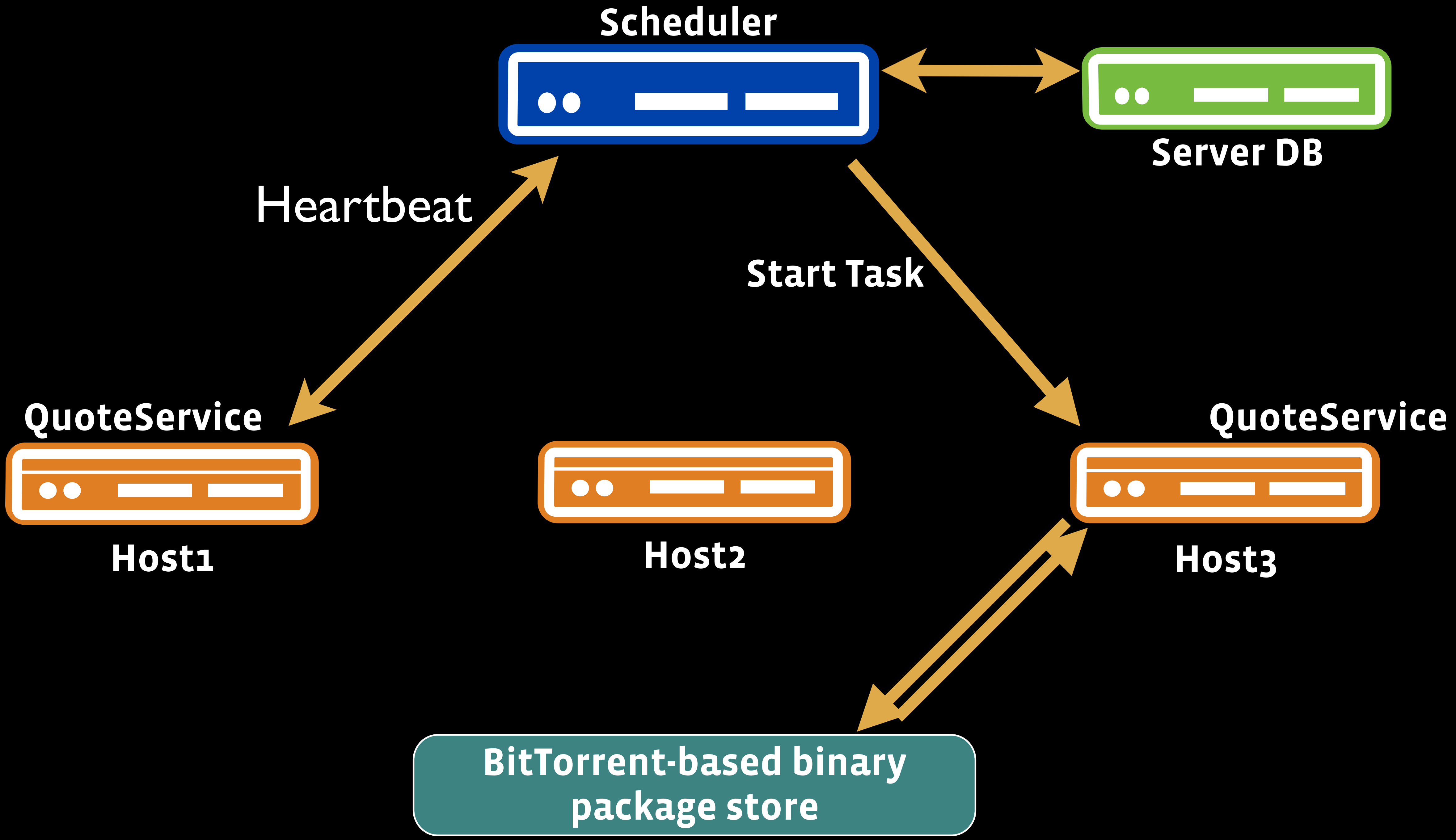


# Failover



Machine "failures" / hour

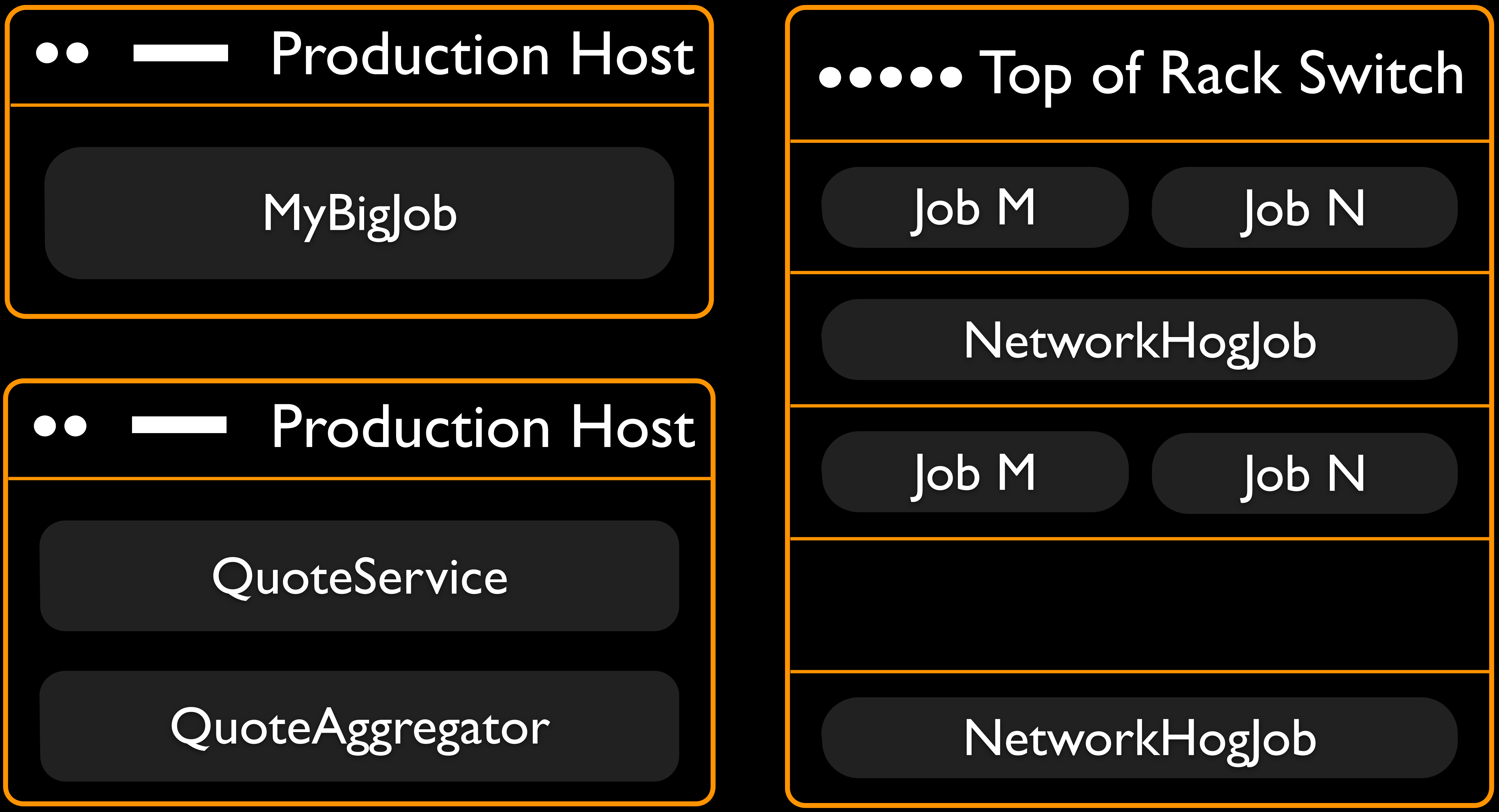
# Failover


Scheduler
Server DB
Heartbeat
Start Task
QuoteService
QuoteService
Host1
Host2
Host3
BitTorrent-based binary package store

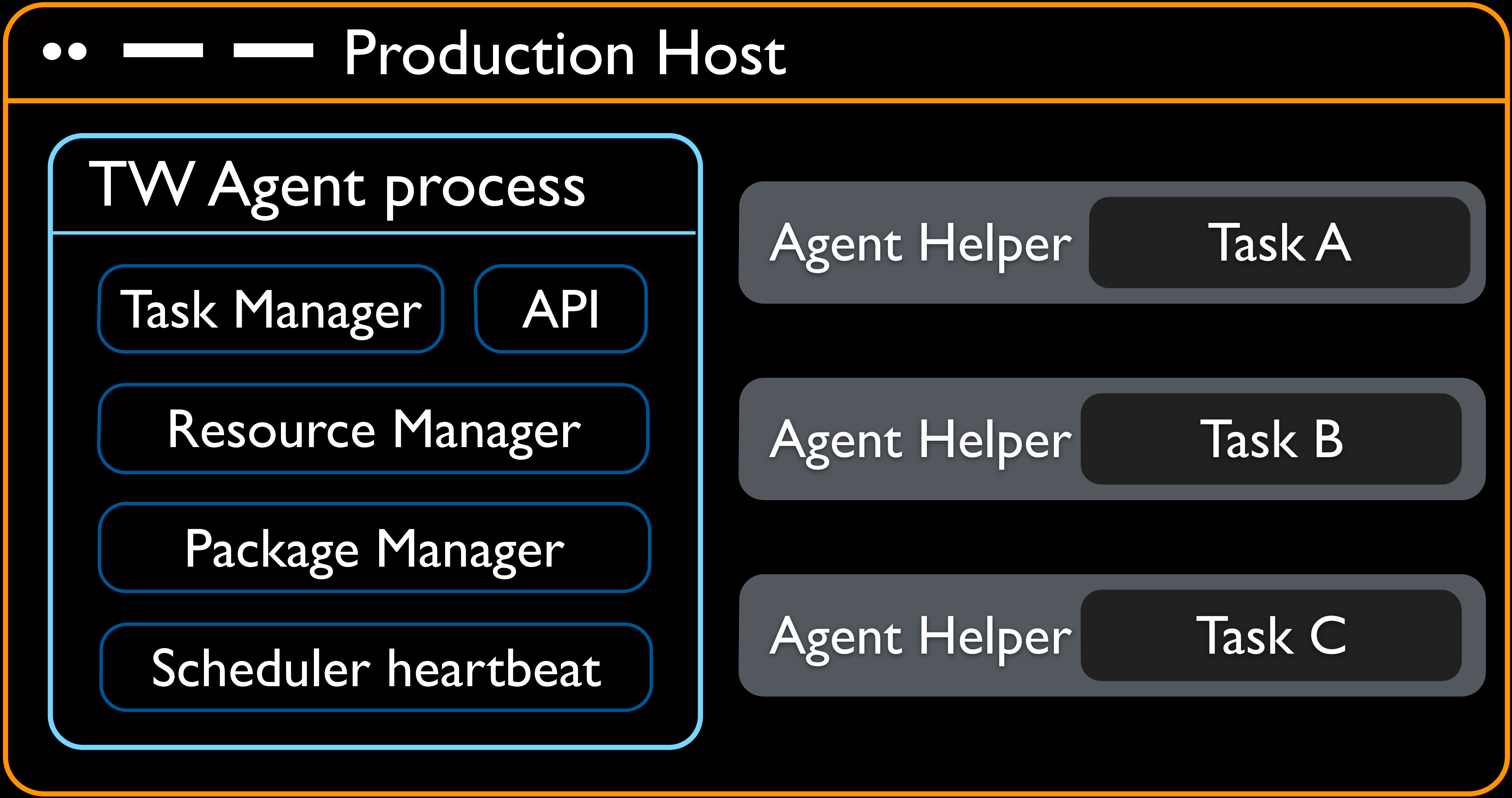
## Painless Hardware maintenance

- Notify scheduler of impending operations
- Scheduler can preemptively move tasks
    - Graceful migration for stateless services
    - Stateful services may endure maintenance

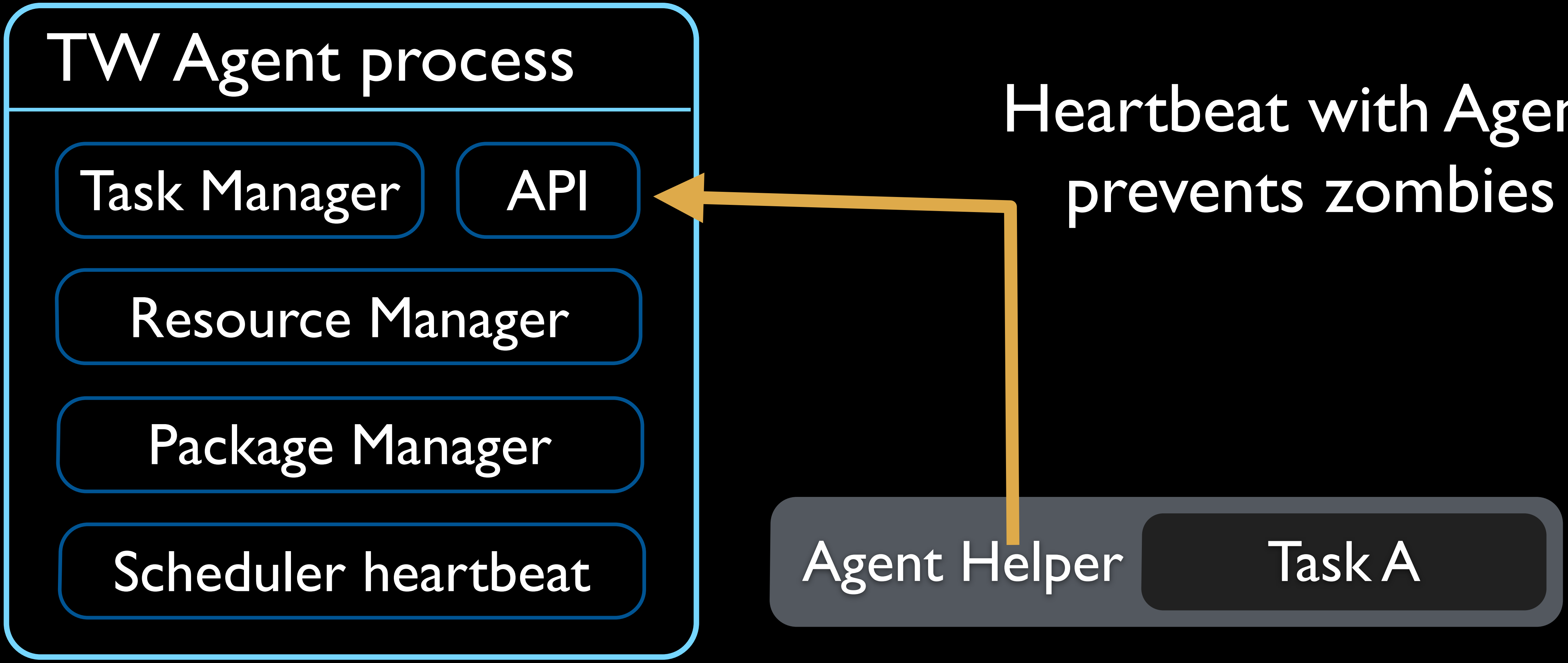
# Expressive allocation policies



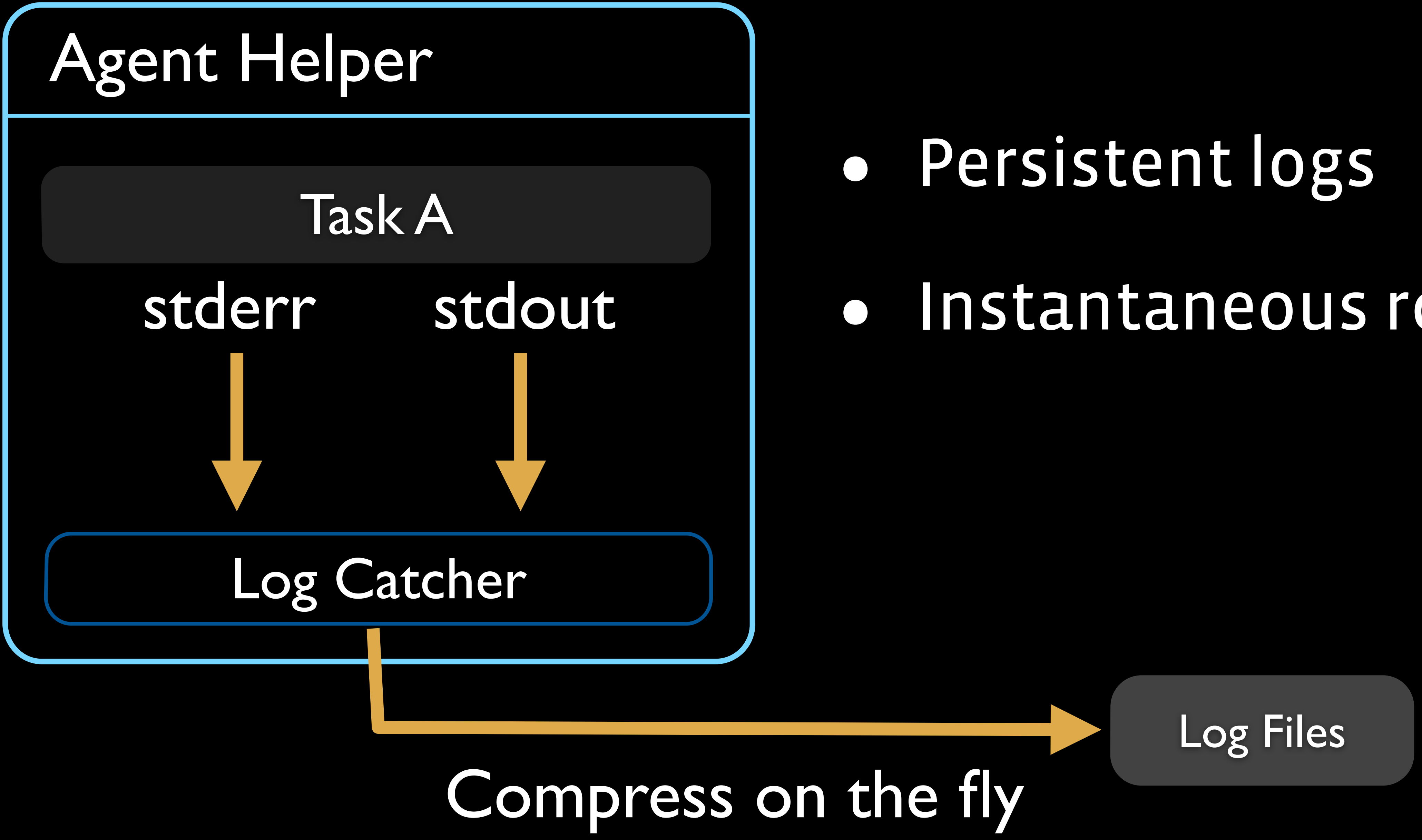
# TW Agent


Production Host
TW Agent process
Task Manager
API
Resource Manager
Package Manager
Scheduler heartbeat
Agent Helper
Task A
Agent Helper
Task B
Agent Helper
Task C

# Agent Helper process



# Logging



- Persistent logs
- Instantaneous rotation

## Sandboxing

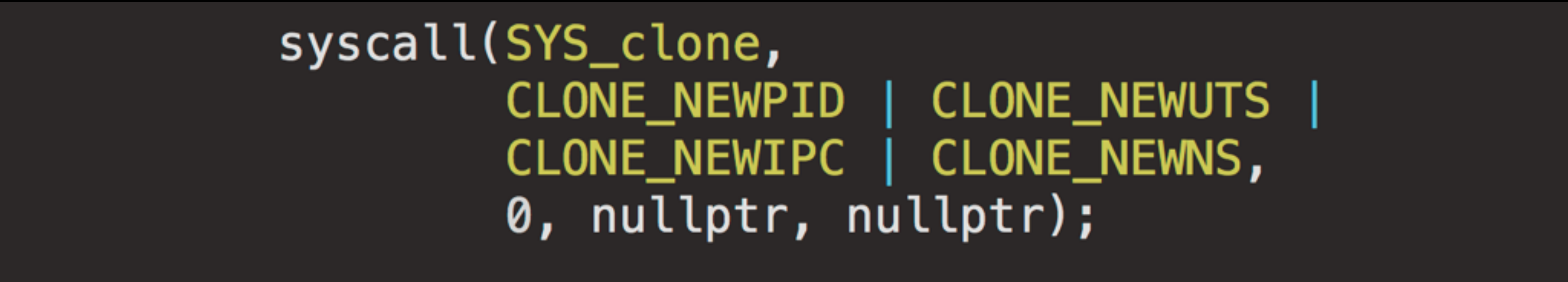
Initially, used chroots to contain processes

- No isolation
- Not secure

```perl
#!/usr/bin/perl -w
chdir "/"; opendir JAILROOT, ".";
mkdir "mysubdir"; chdir "mysubdir";
chroot "."; chdir(*JAILROOT);
while ((stat("."))[0] != (stat(".."))[0] or
       (stat("."))[1] != (stat(".."))[1]) {
  chdir "..";
}
chroot ".";
system("/bin/sh");
```

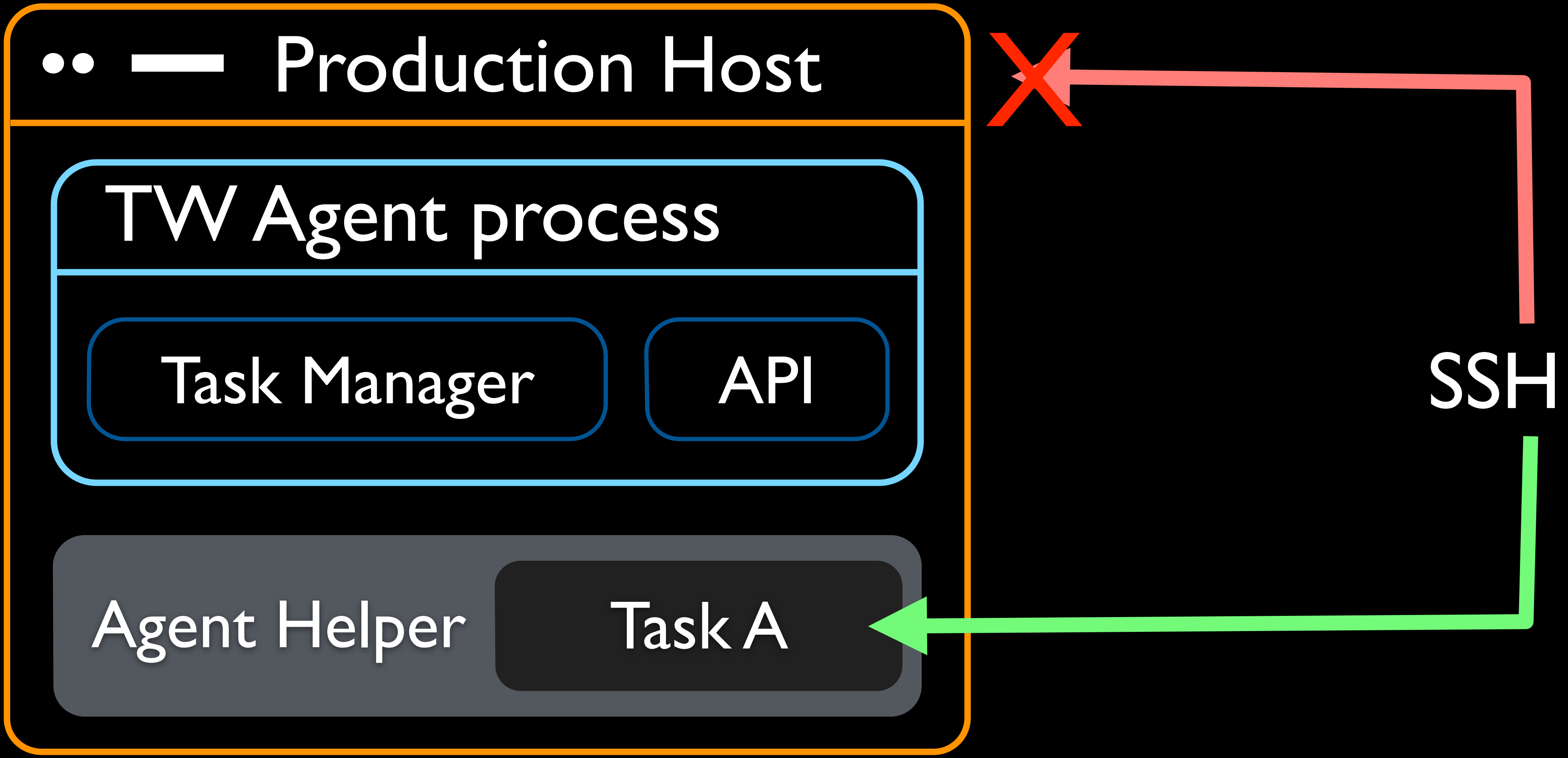
# LinuXContainers

- As tech matured, we switched
- Separate process and file namespaces, set up by Helper
- Mount required resources directly into container
- Secure & isolated



```
syscall(SYS_clone,
        CLONE_NEWPID | CLONE_NEWUTS |
        CLONE_NEWIPC | CLONE_NEWNS,
        0, nullptr, nullptr);
```

# Service permissions



- Every container runs sshd
  - SSH directly into the container
  - Regulate access

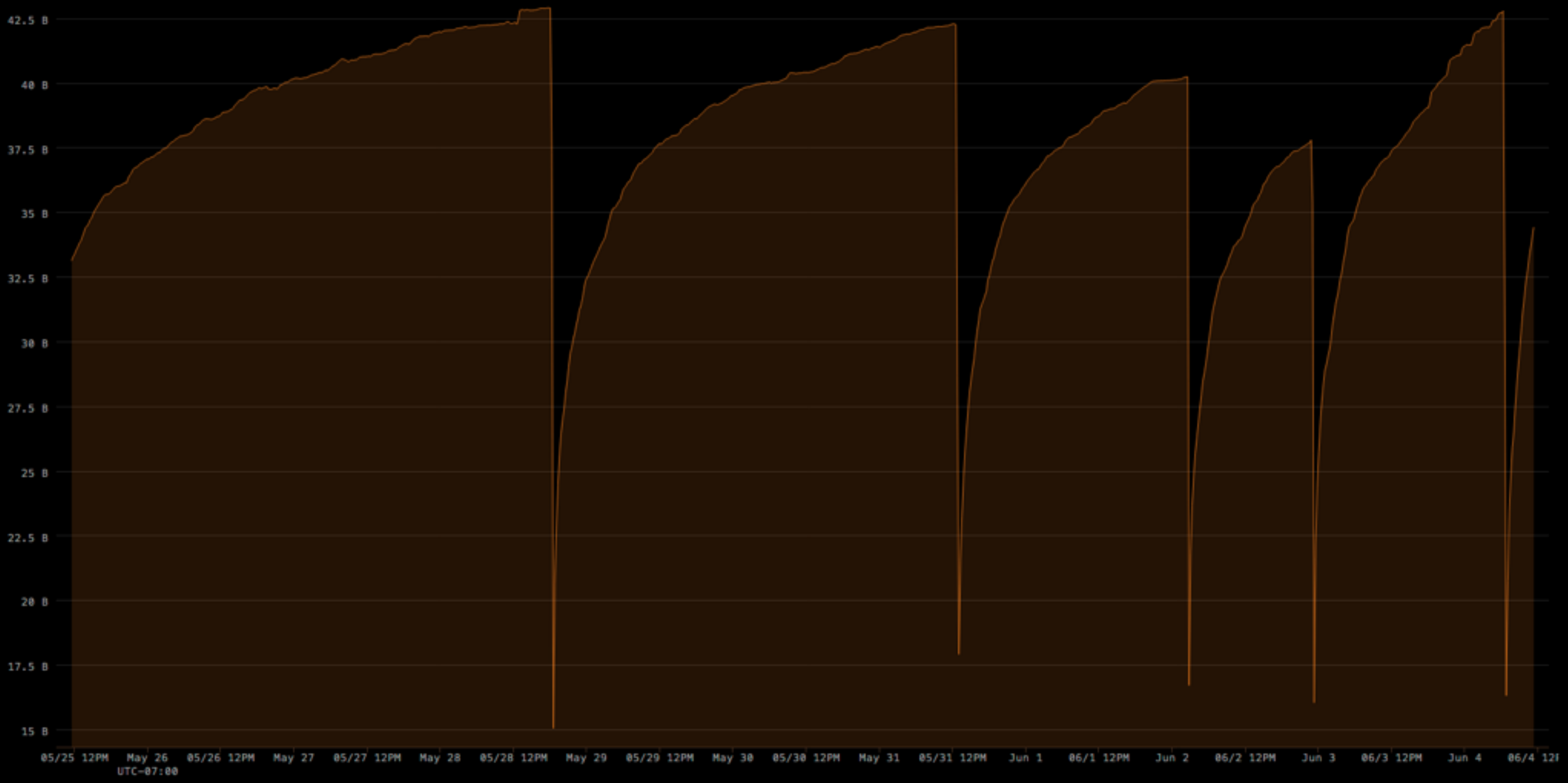
## Configuring the container

```
import 'git.tw'

job = Job(
  command = '/packages/quote_server/server',
  scheduling = Scheduling(replicas = 10),

  profile = 'centos6',  # centos5, ubuntu, ...
  packages = [
    Package(name='quote_server'),
    Package(name='rpm-deps',
            rms=['emacs', 'nfs-utils']),
  ],

  resource_limits = ResourceLimit(
    cpu = 2, ram = '4G', disk = '32G',
  )
)

enable_git_support(job)
```

# Resource limits

- CPU, RAM & disk limits
- Implemented with cgroups
- Agent handles memory limits with cgroup notification API
- Adaptive limits

# Resource Limits in action

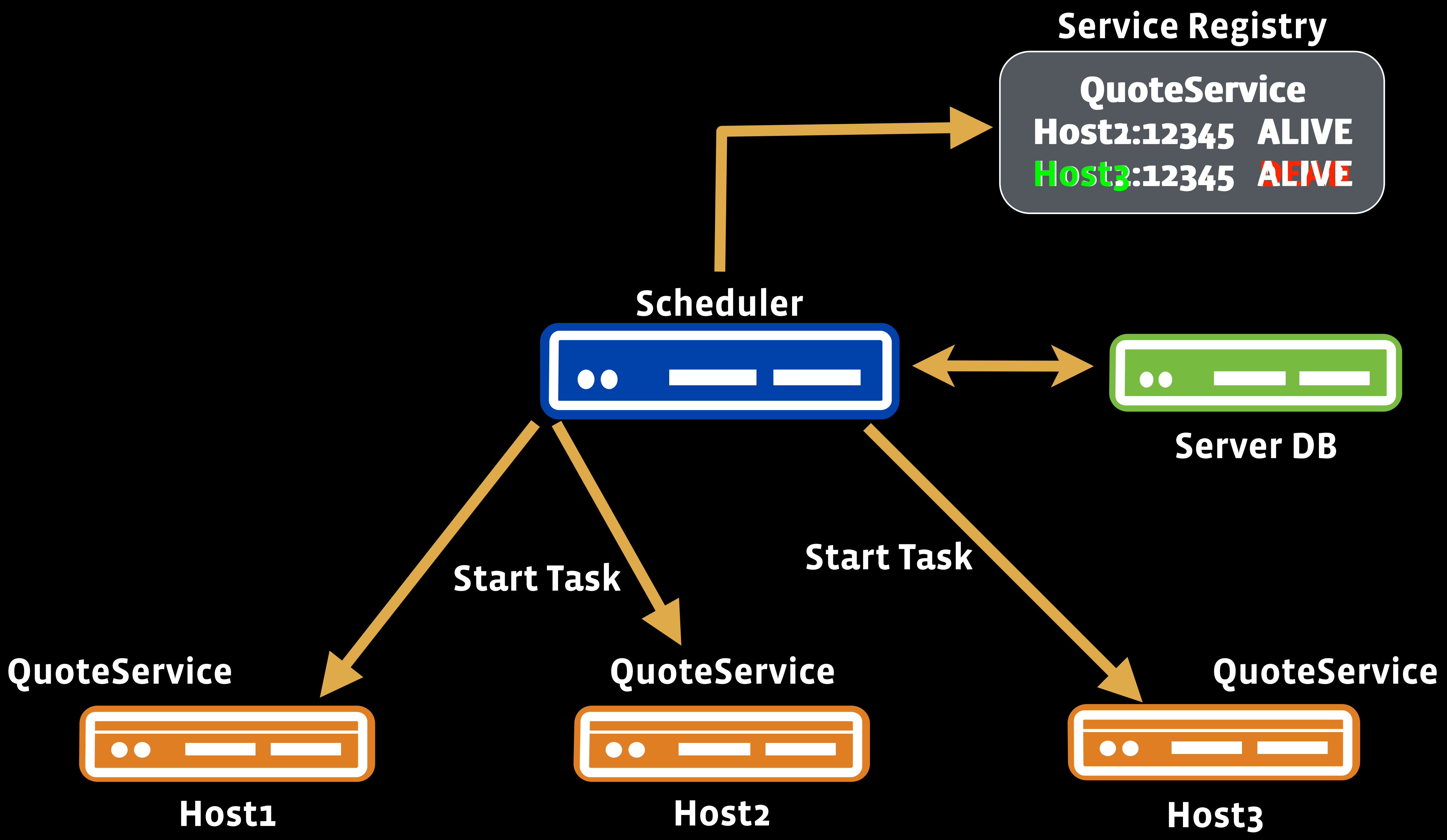


watchdog-service - tw.mem.rss_bytes

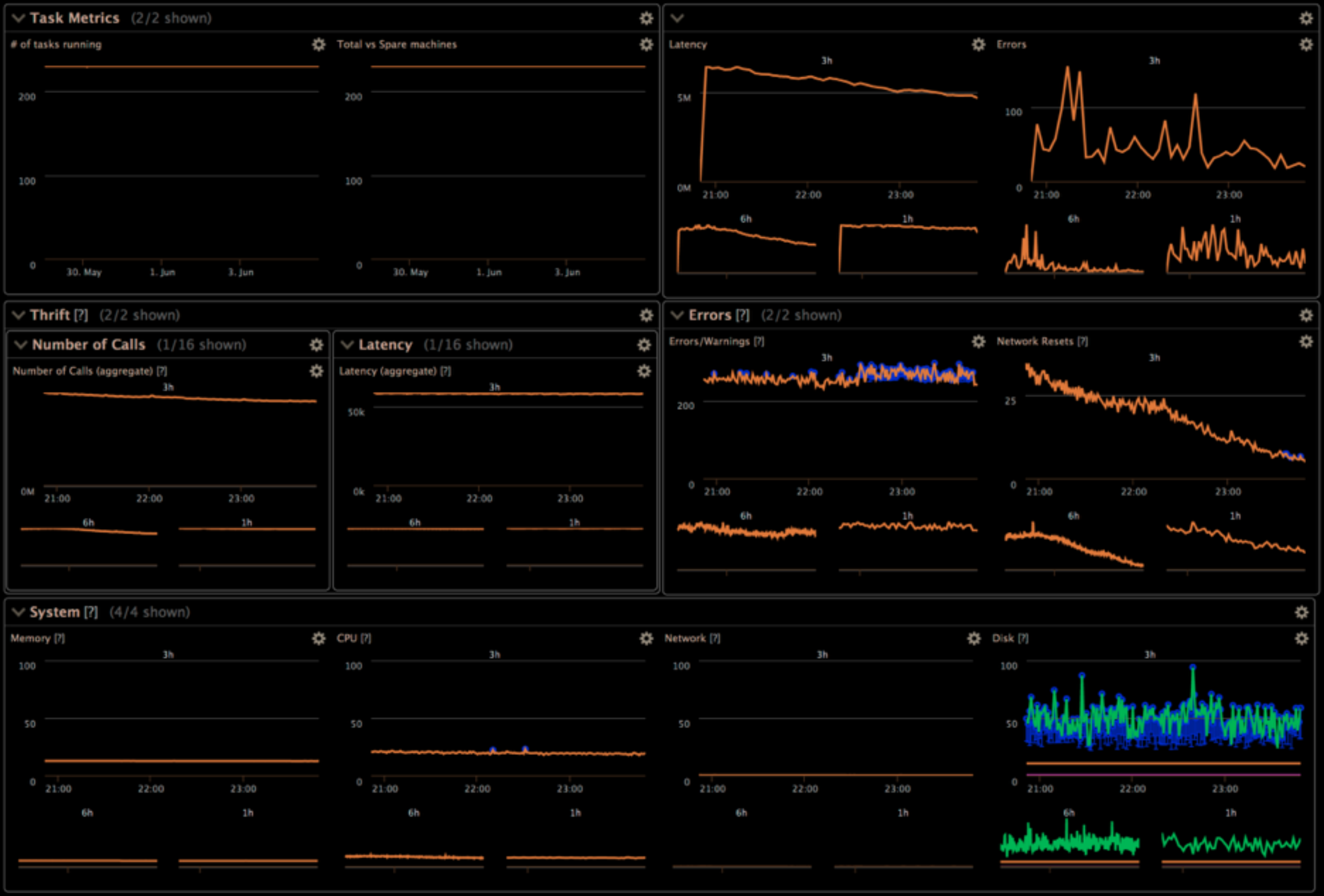
## Migrate from Chroots to Containers

- No-op for most services
- But new namespaces posed problems for some
- Major hurdle was social, not technical

# Service Discovery



# Monitoring & Alerting



## Alternatives to Tupperware

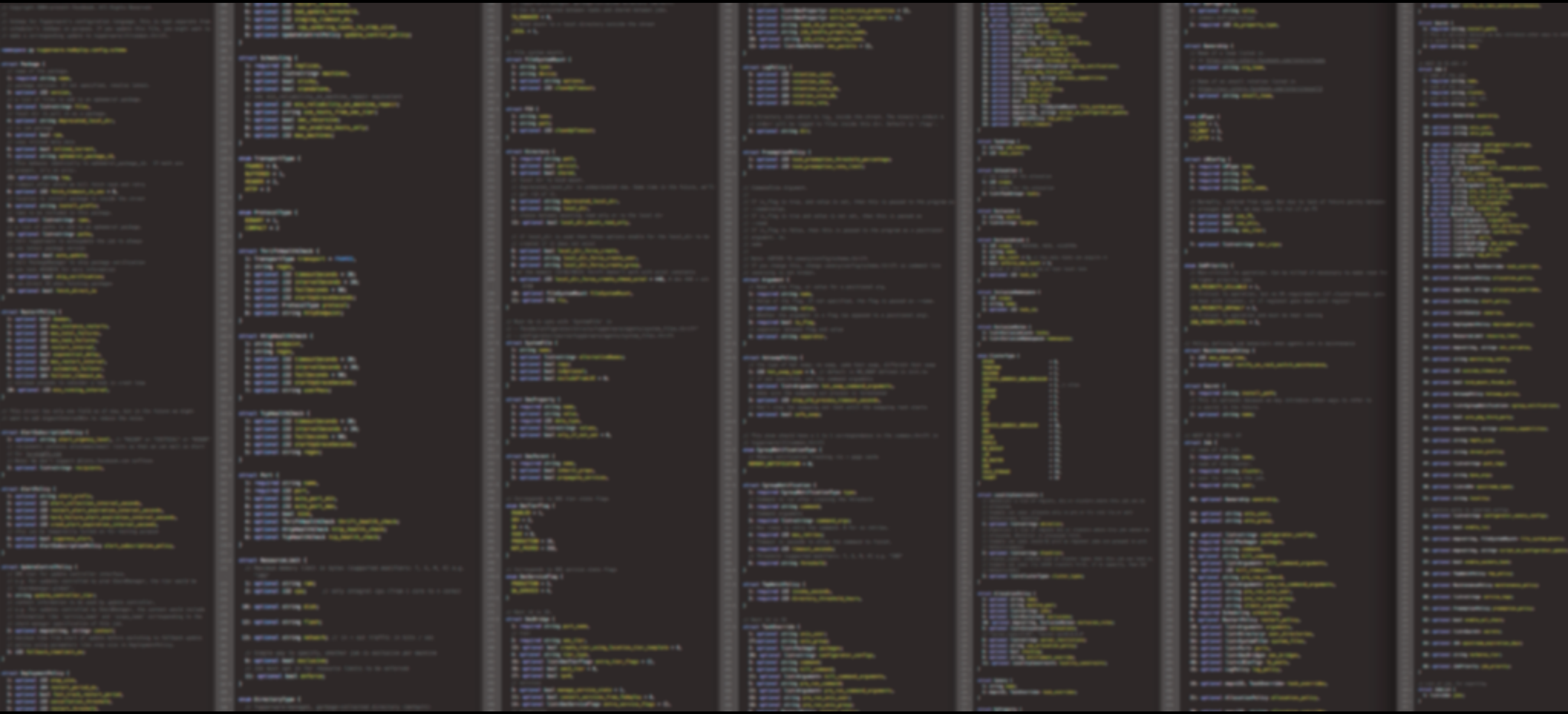
- Why not use Docker / CoreOS?
  - They didn’t exist
  - TW integrates with other FB systems
- Why not use VMs?
  - Performance penalty
  - Hypervisor makes debugging harder

## Lessons learnt

## Releases are scary!

- Release often
- Dry runs
- Canaries are your friends
- Manage dependencies

## Sane defaults



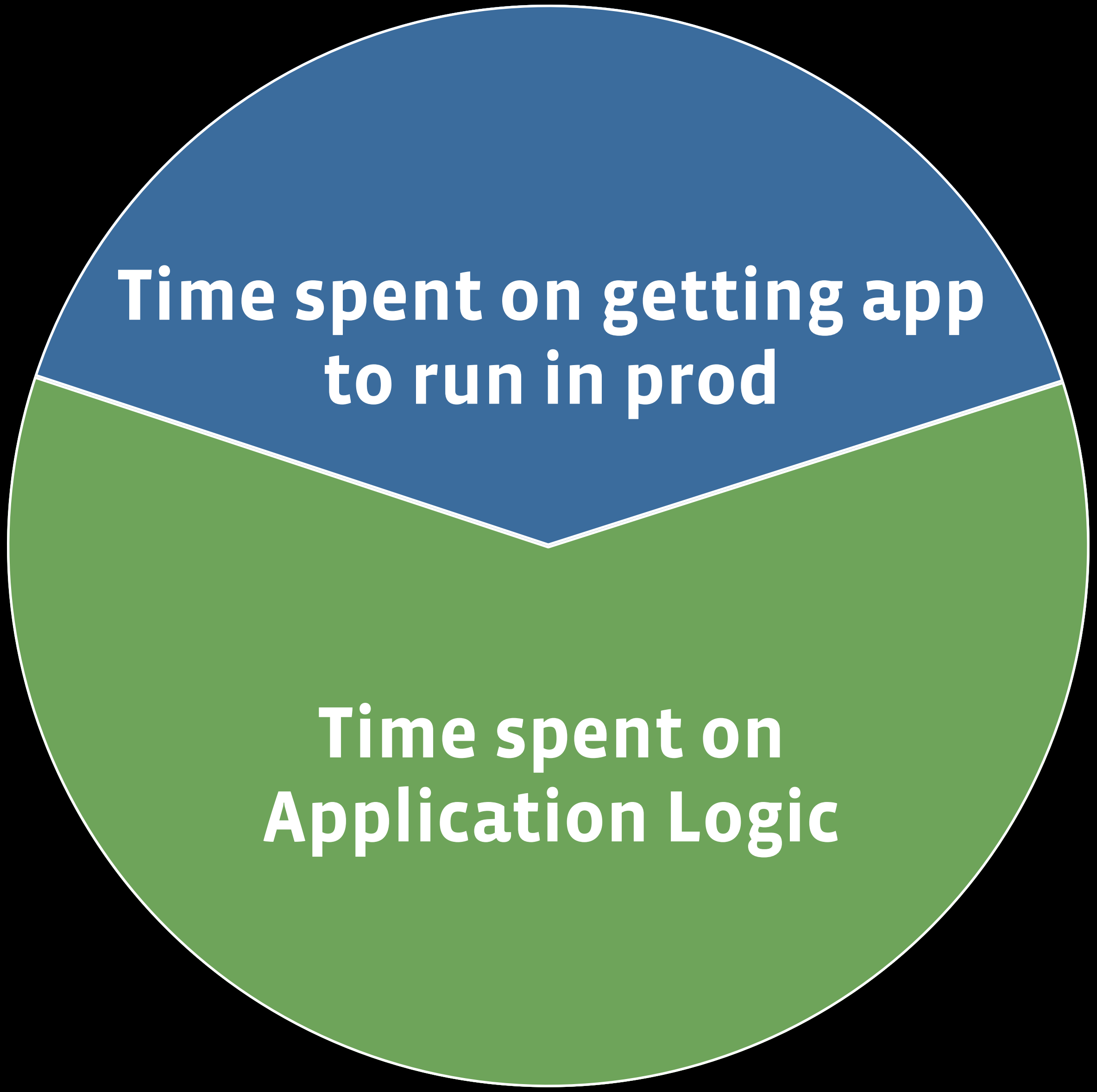
- Users shouldn’t have to read entire manual
- Choose what makes sense for most services

## What went wrong?

- Hard to understand ***why*** TW did something
- It’s not about “what went wrong”, but “what should I do next?”

# Tupperware

- Automated deployment
  - Less work for engineers
- Containers for security and isolation
- Increased efficiency



Questions?