
Distributed scheduler hell



# Story of ..

## How we moved 100(s) of virtual machines, onto containers

# Take aways

- What do Distributed schedulers do?
- Why?
- Best Practices
- What not to use it for

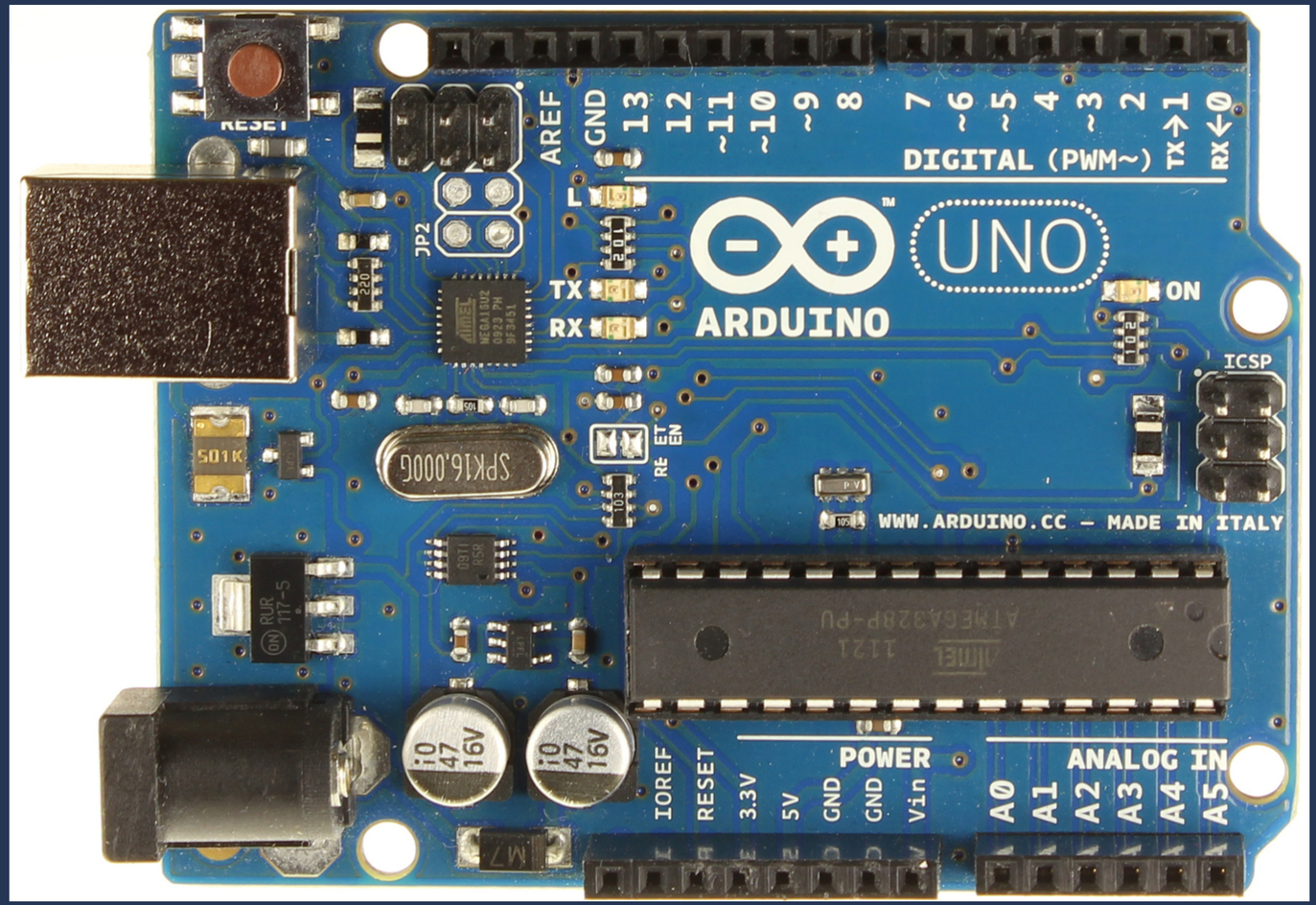




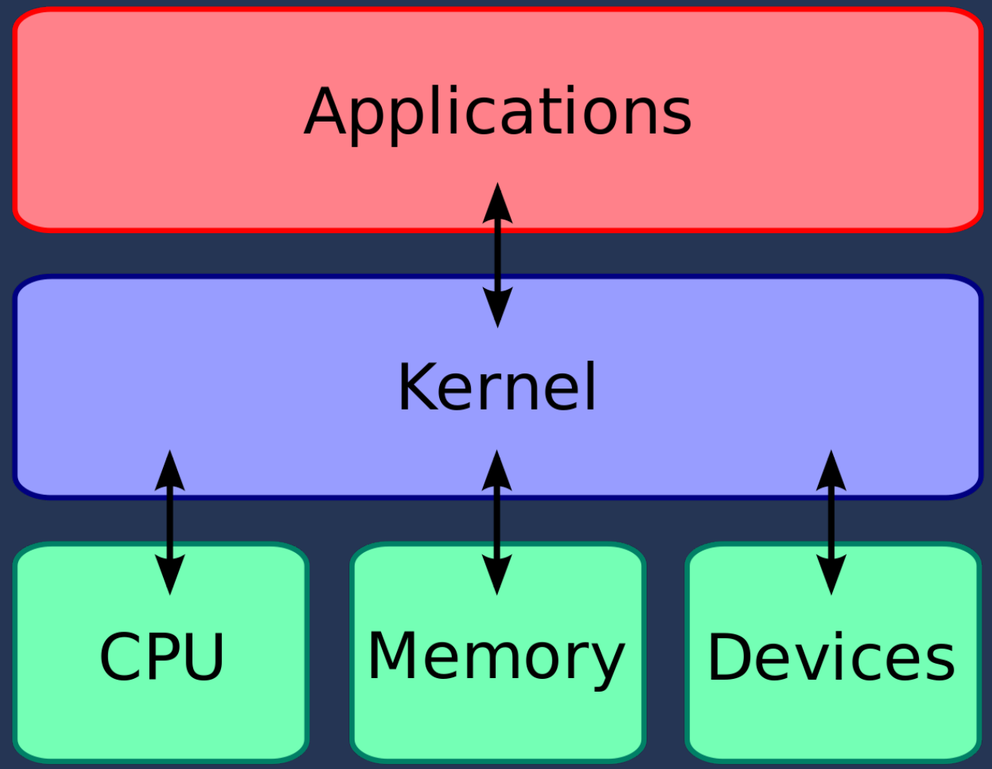
# What is a distributed scheduler

Your cloud provider: Digital Ocean, AWS, Google.
For containers: Kubernetes, Nomad, Mesos

# Single process



# Hypervisor
# Linux Scheduler



## Kernel Provides

- Virtual Memory
- Process Isolation
- Disk storage
- Networking
- CPU scheduling

# Distributed Scheduler Provides




- Container Deploy
- Virtual Machine Deploy
- Memory Quota
- Disk storage
- Networking
- CPU scheduling
- Scaling Instances


Why?
Microservices...
Deployment...
Lifecycle mangement

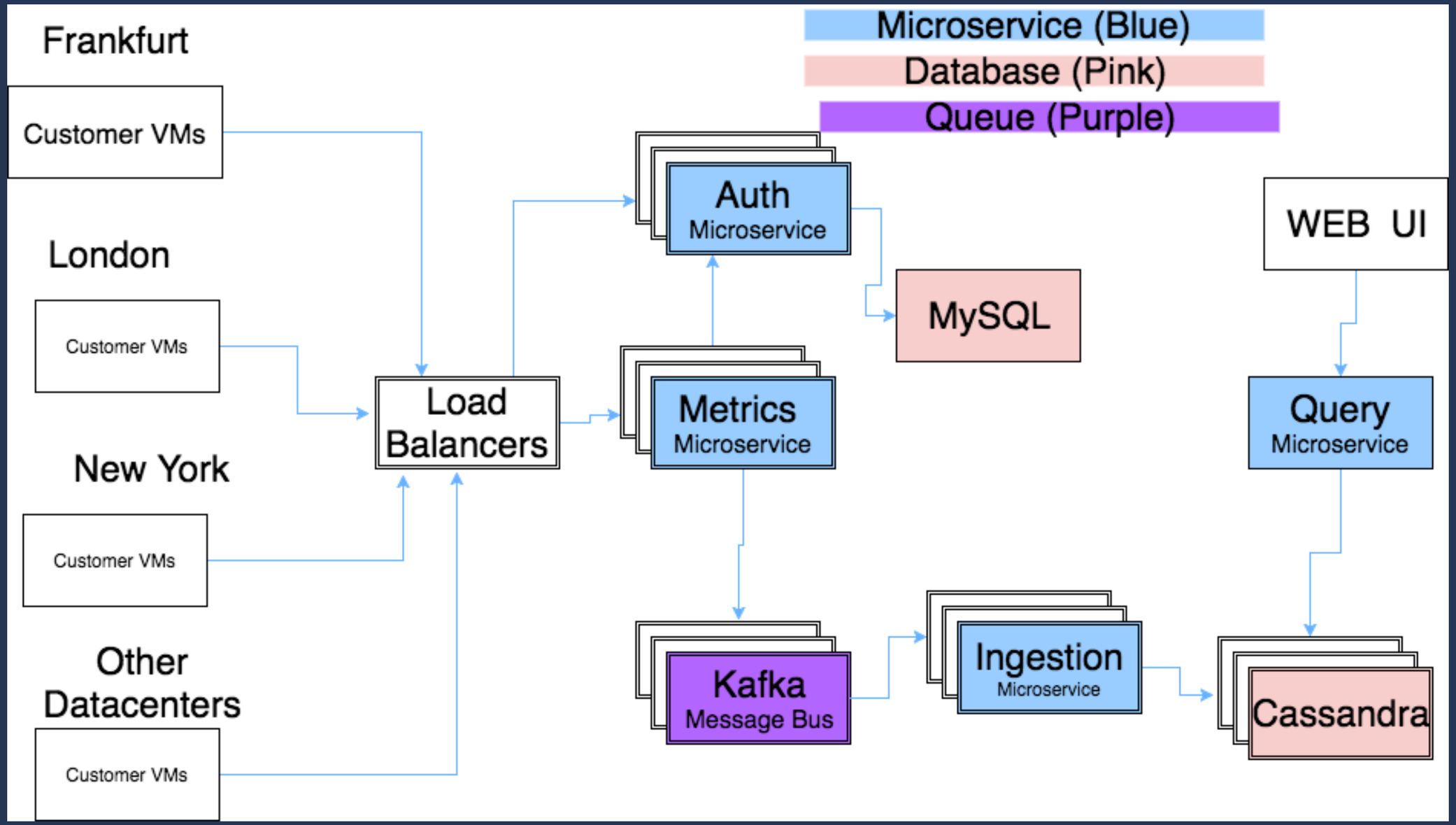
# Vulcan : Distributed timeseries database



https://github.com/digitalocean/vulcan

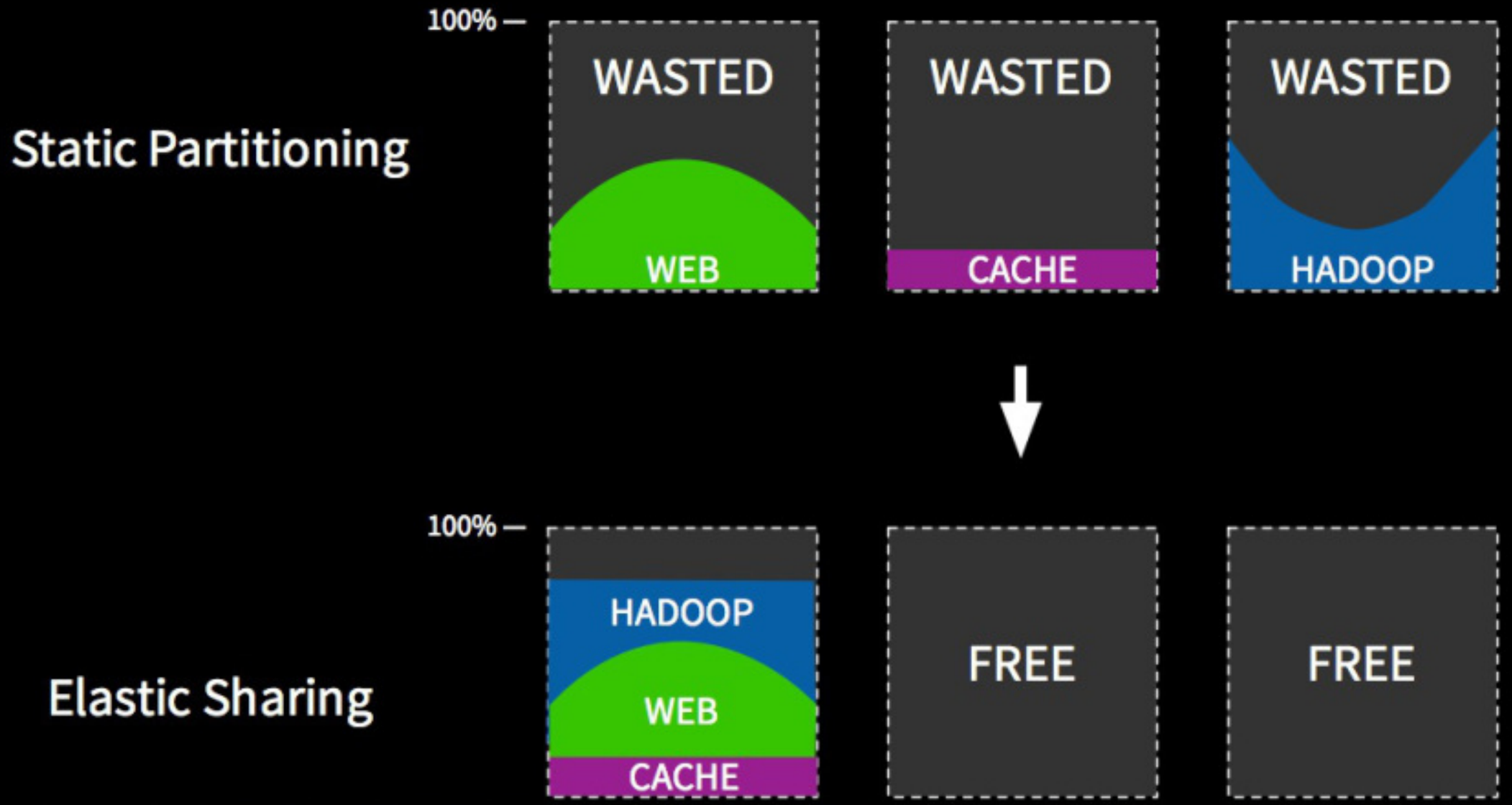
# Requirements

- 3 Gbits a second of Network traffic
- 20 TB Storage
- Sub 100ms Read times
- 100k writes a second


Microservice (Blue)
Database (Pink)
Queue (Purple)
Frankfurt
Customer VMs
London
Customer VMs
New York
Customer VMs
Other Datacenters
Customer VMs
Load Balancers
Auth Microservice
MySQL
Metrics Microservice
Kafka Message Bus
Ingestion Microservice
Cassandra
WEB UI
Query Microservice


dcos-production
172.03.12.1
Dashboard
Services
Datacenter
Mesosphere DCOS v.1.6
Dashboard
CPU Allocation
47%
65.1 of 114.2 Shares
100%
50%
0%
Last 60 Seconds
Memory Allocation
32%
24.6GB of 76.8GB
100%
50%
0%
Last 60 Seconds
Task Failure Rate
2%
Current Failure Rate
100%
50%
0%
Last 60 Seconds
Mesos
Service Health
marathon-production Healthy
chronos Healthy
cassandra-dev Sick
marathon-dev Healthy
hdfs-billing Healthy
View all 64 Services
Tasks
402
Total Tasks
263
Running Tasks
139
Staged Tasks
Datacenter Health
127
Total Nodes
123
Connected
4
Disconnected

# From Static Partitioning to Elastic Sharing




Search Service
Jetty
Rails



# Deploying our App On Mesos



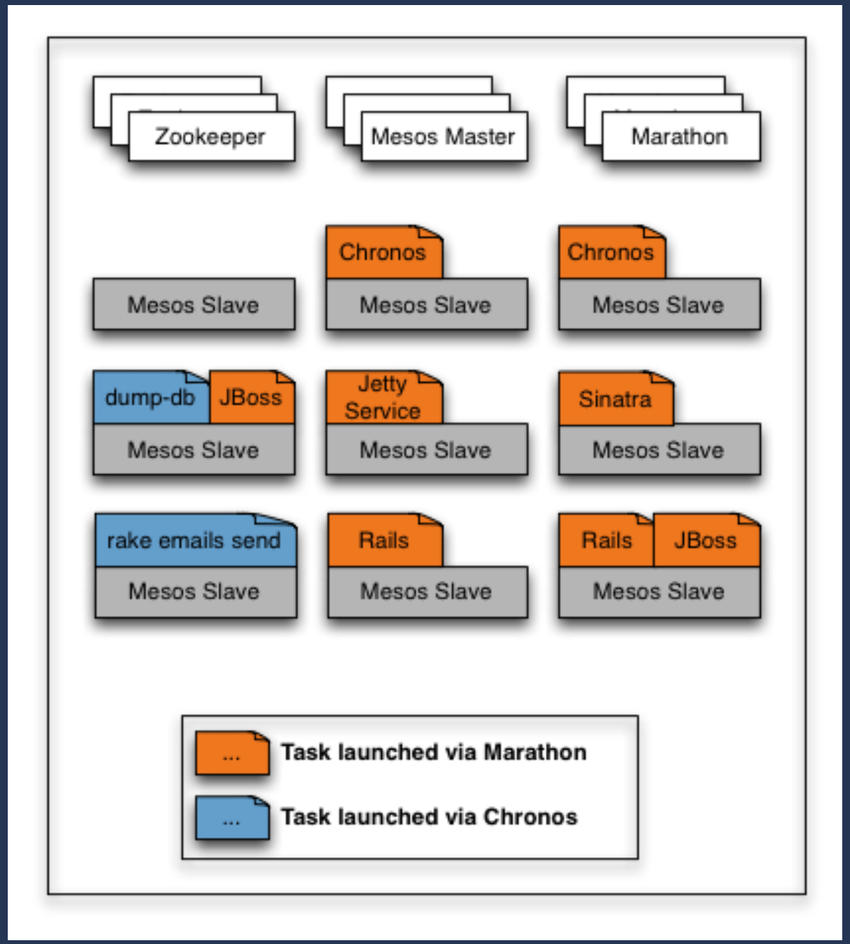
MARATHON

Apps 1 Deployments About Docs ⇗

+ New App

| ID ▲ | Memory (MB) | CPUs | Tasks / Instances | Health | Status |
|---|---|---|---|---|---|
| /kibana | 512 | 0.5 | 0 / 1 | | Deploying |
| /mesos-dns | 512 | 0.2 | 3 / 3 | | Running |
| /search | 2048 | 1 | 2 / 2 | | Running |

# Marathon Architecture




Kafka
Custom Scheduler



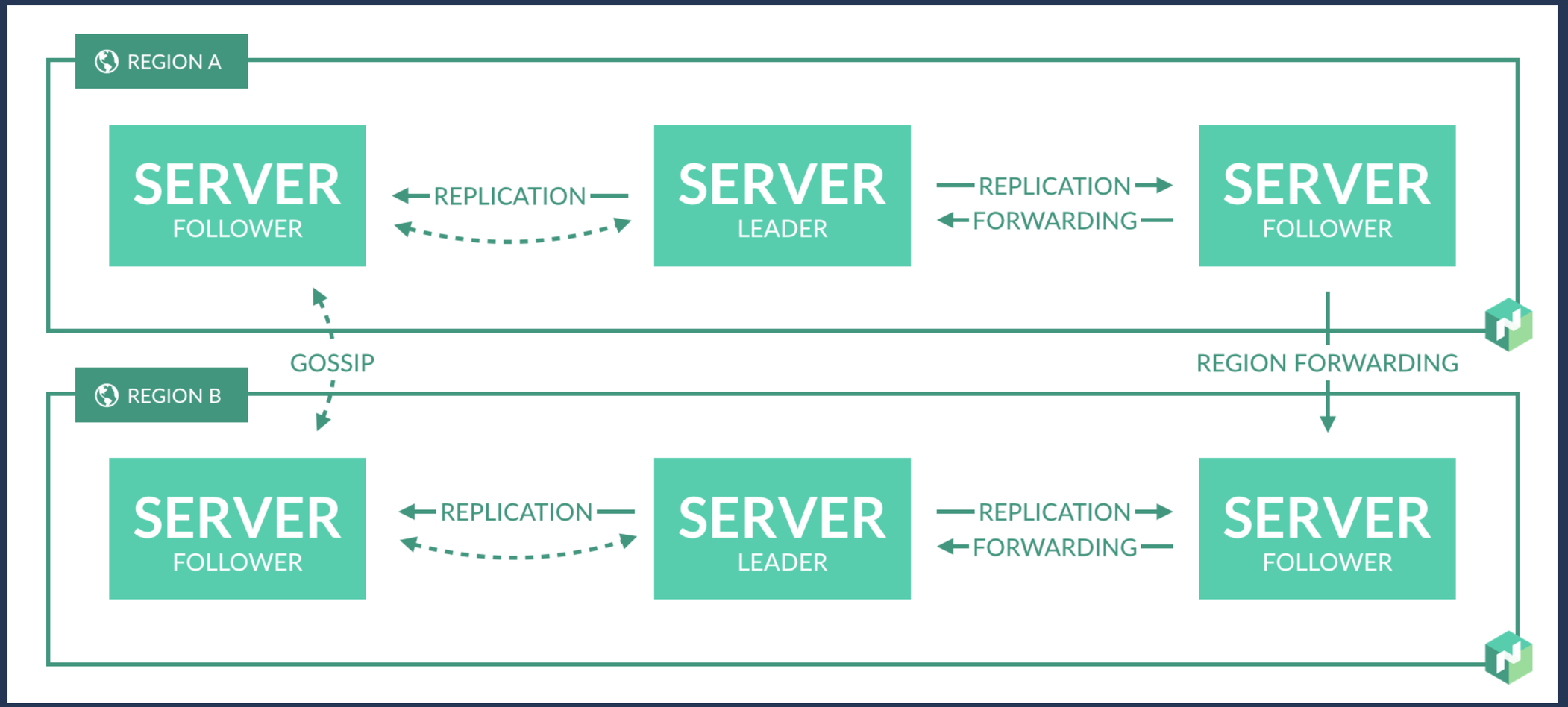
# Cassandra

## Pinning to specific Mesos nodes

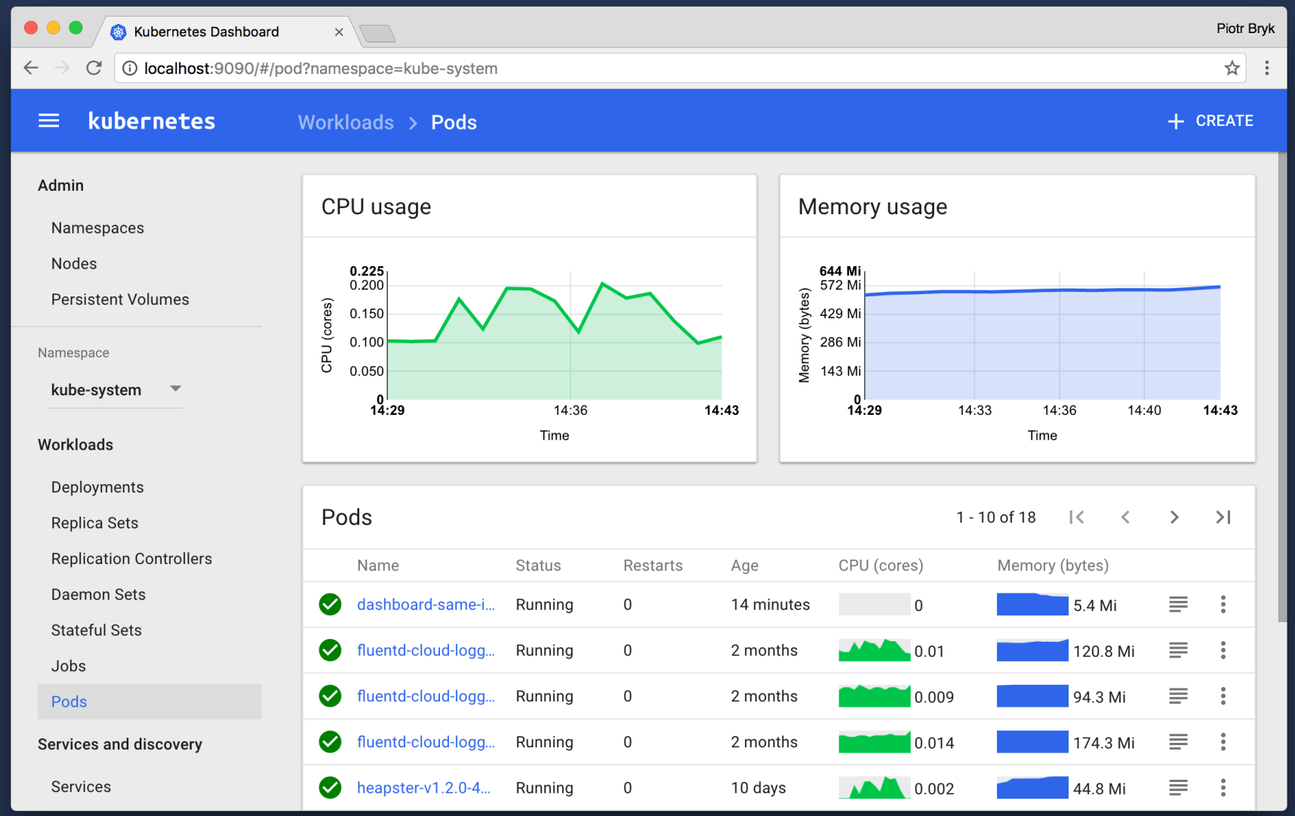

Mesos Failure Modes
Network Partition FAIL

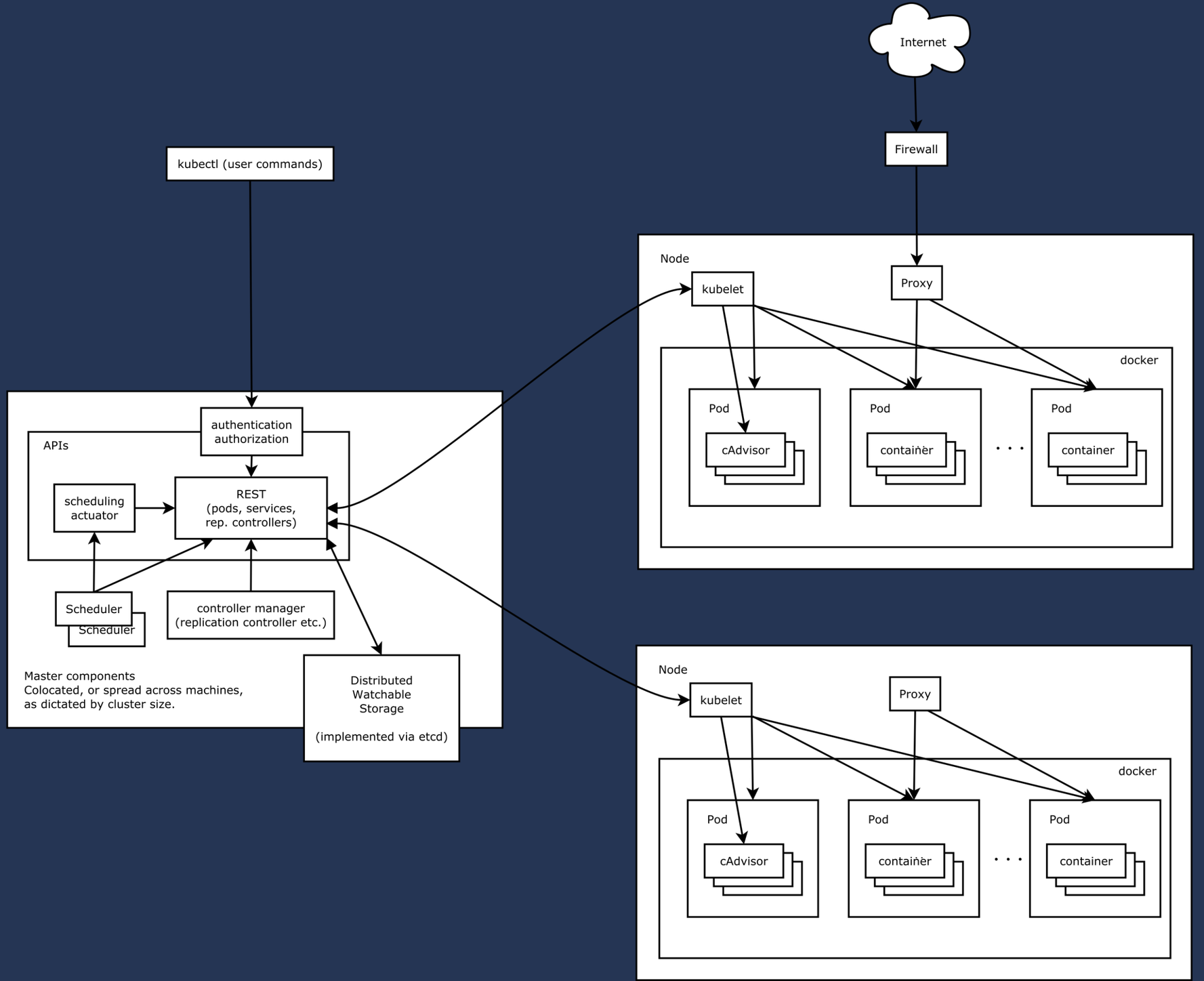
# Counter Example: Nomad

# Nomad Architecture



# Kubernetes


Kubernetes Dashboard
Piotr Bryk
localhost:9090/#/pod?namespace=kube-system
kubernetes
Workloads > Pods
CREATE
Admin
Namespaces
Nodes
Persistent Volumes
Namespace
kube-system
Workloads
Deployments
Replica Sets
Replication Controllers
Daemon Sets
Stateful Sets
Jobs
Pods
Services and discovery
Services
CPU usage
0.225
0.200
0.150
0.100
0.050
0
CPU (cores)
14:29
14:36
14:43
Time
Memory usage
644 Mi
572 Mi
429 Mi
286 Mi
143 Mi
0
Memory (bytes)
14:29
14:33
14:36
14:40
14:43
Time
Pods
1 - 10 of 18
Name
Status
Restarts
Age
CPU (cores)
Memory (bytes)
dashboard-same-i...
Running
0
14 minutes
0
5.4 Mi
fluentd-cloud-logg...
Running
0
2 months
0.01
120.8 Mi
fluentd-cloud-logg...
Running
0
2 months
0.009
94.3 Mi
fluentd-cloud-logg...
Running
0
2 months
0.014
174.3 Mi
heapster-v1.2.0-4...
Running
0
10 days
0.002
44.8 Mi


Internet
Firewall
kubectl (user commands)
Node
kubelet
Proxy
docker
Pod
cAdvisor
Pod
container
Pod
container
authentication
authorization
APIs
scheduling
actuator
REST
(pods, services,
rep. controllers)
Scheduler
Scheduler
controller manager
(replication controller etc.)
Master components
Colocated, or spread across machines,
as dictated by cluster size.
Distributed
Watchable
Storage
(implemented via etcd)
Node
kubelet
Proxy
docker
Pod
cAdvisor
Pod
container
Pod
container

# Custom Json Language

```
{
  "application": {
    "name": "timeseries-ingestor",
    "scale": 1,
    "ingresses": {
      "timeseries-ingestor-health": {
        "scheme": "http",
        "container_port": 8001
      }
    },
    "containers": {
      "timeseries-ingestor": {
        "image": "docker.com/timeseries/ingestor",
        "image_tag": "fcb24ca",
        "ports": [
          8001,
          9090
        ],
        "env": {
          "CASSANDRA_ADDRESS_LIST": ""
        },
        "resources": {
          "cpu": "4",
          "memory": "4000"
        }
      }
    },
    "metrics": {
      "port": 8001,
      "path": "/debug/metrics"
    }
  },
  "maintainer": "dummy@digitalocean.com"
}
```

# Command Line Deploys

```
docc --contexts dev_env deploy timeseries-microservice1.json
```

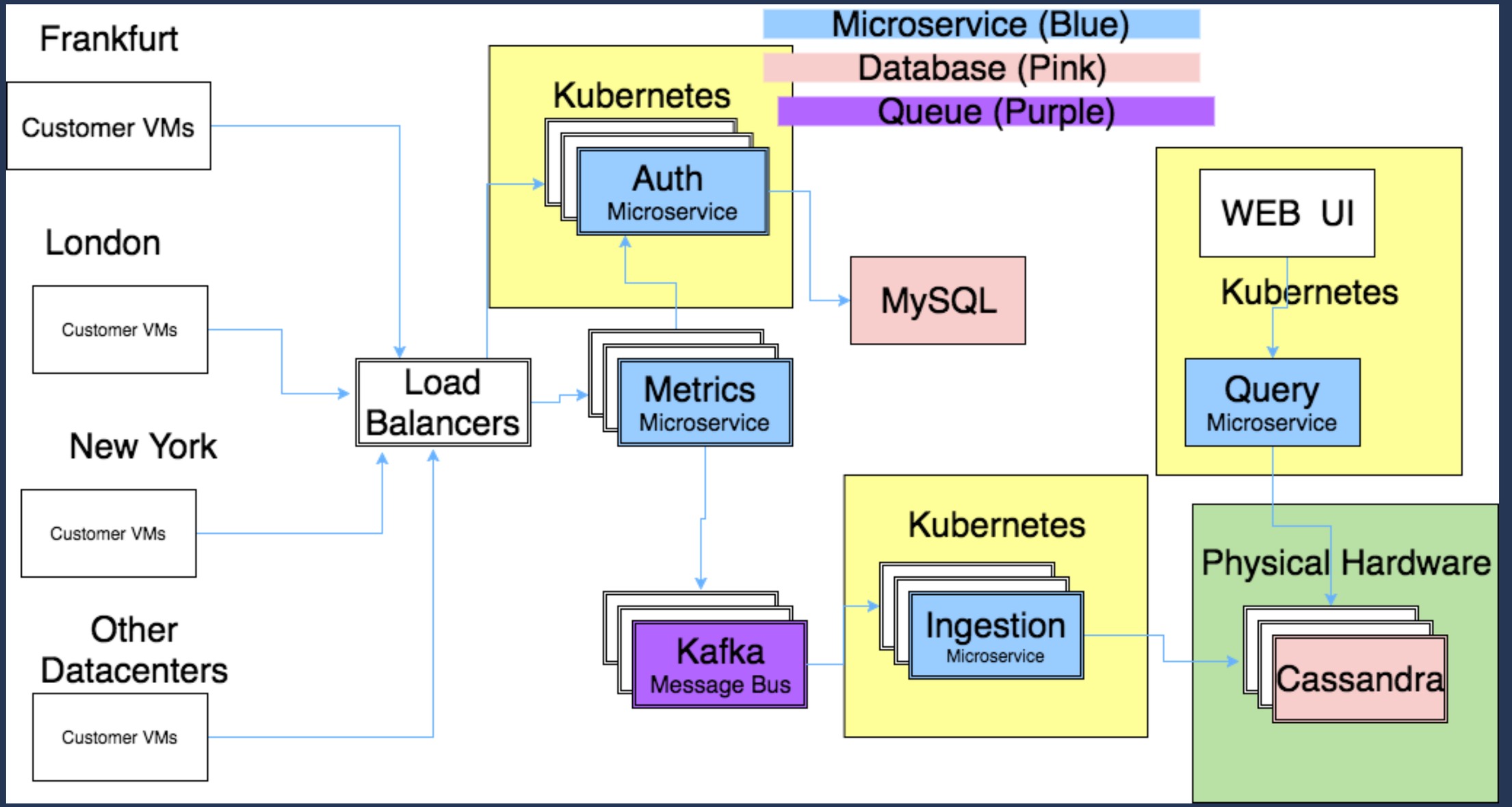
**Docc is our internal Kubernetes tool**

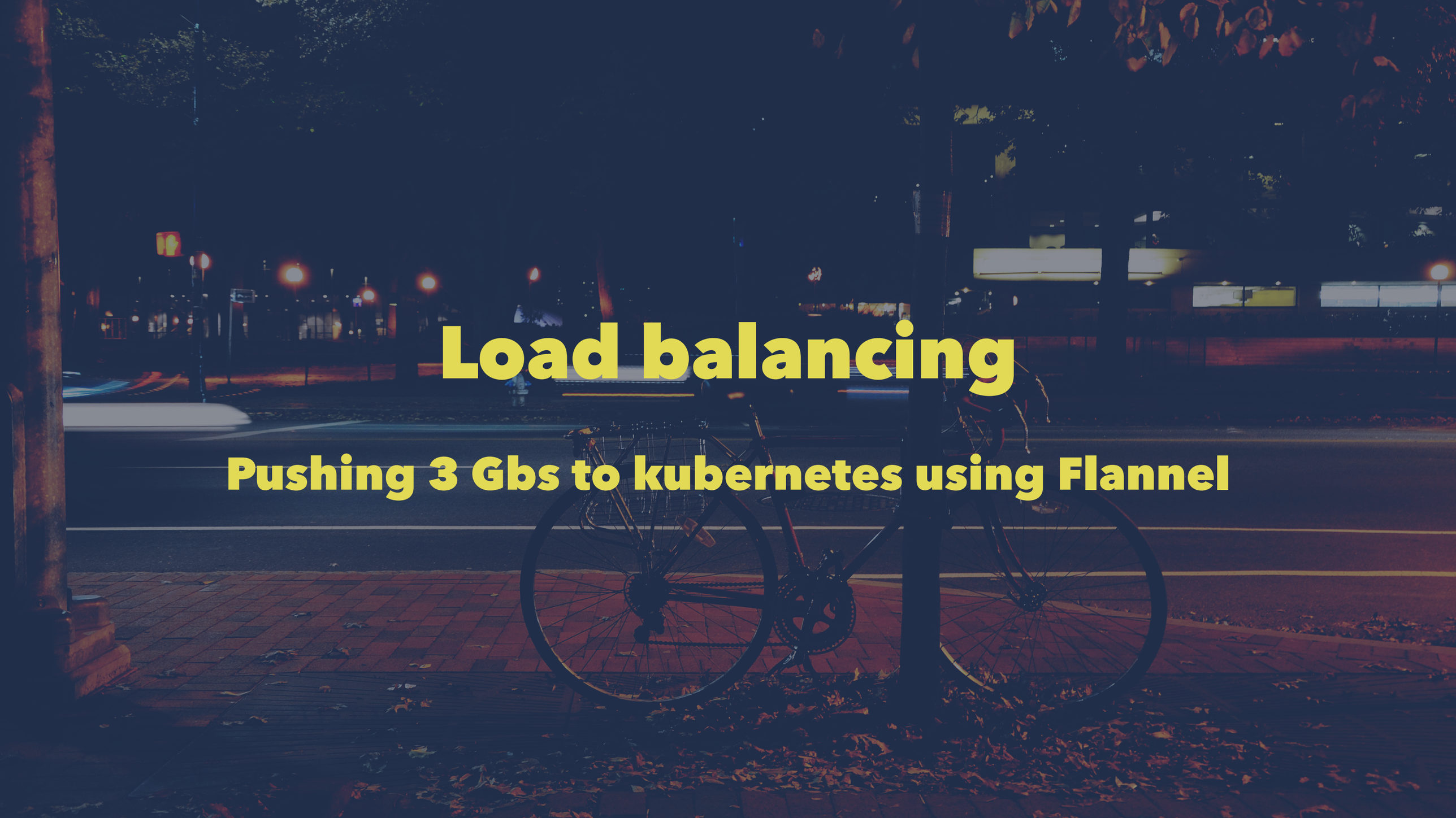
# Deployment tool with diffs to Kubernetes

```
                "env": {
+++                      "CASSANDRA_CQL_VERSION": "3.1.7",
---                      "CASSANDRA_KEYSPACE": "staging_vulcan",
+++                      "CASSANDRA_KEYSPACE": "staging_vulcan_123",
```

# Simliar to KubeDiff

# Final Architecture


Frankfurt
Customer VMs
London
Customer VMs
New York
Customer VMs
Other Datacenters
Customer VMs
Load Balancers
Kubernetes
Auth
Microservice
MySQL
Metrics
Microservice
Kafka
Message Bus
Kubernetes
Ingestion
Microservice
Microservice (Blue)
Database (Pink)
Queue (Purple)
WEB UI
Kubernetes
Query
Microservice
Physical Hardware
Cassandra

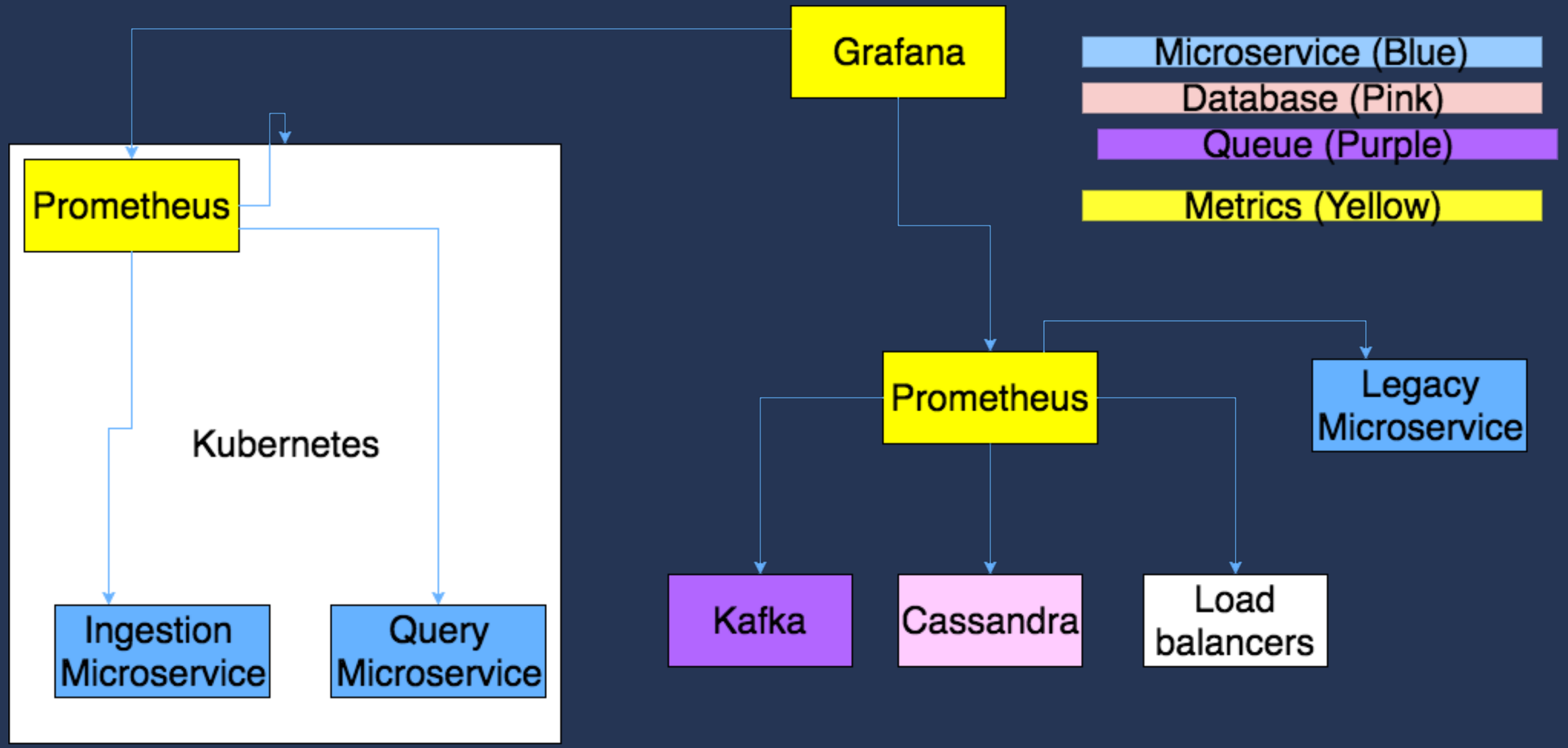


# Load balancing

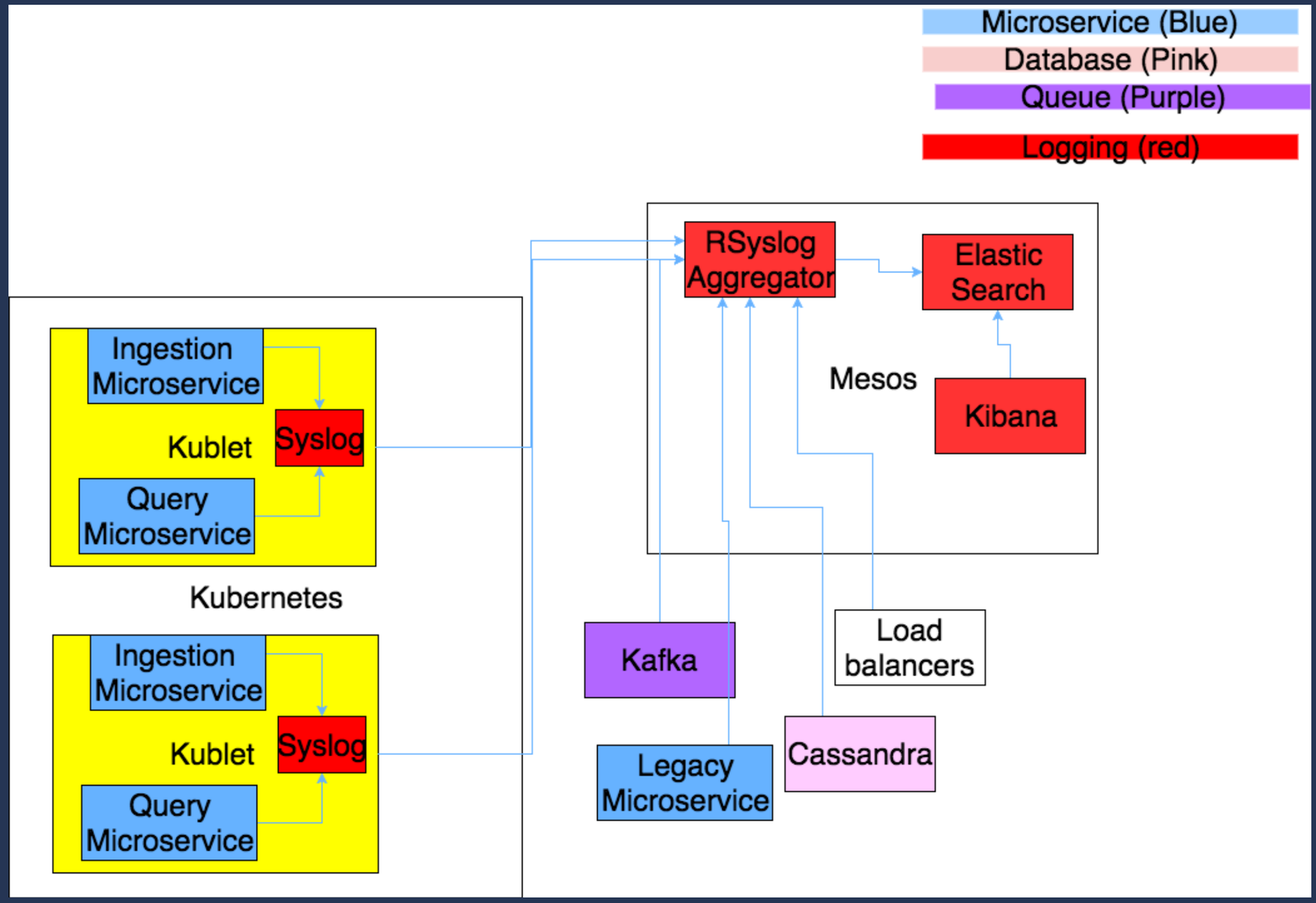
## Pushing 3 Gbs to kubernetes using Flannel



# Metrics


Grafana
Microservice (Blue)
Database (Pink)
Queue (Purple)
Metrics (Yellow)
Prometheus
Kubernetes
Ingestion Microservice
Query Microservice
Prometheus
Legacy Microservice
Kafka
Cassandra
Load balancers


Logging


Microservice (Blue)
Database (Pink)
Queue (Purple)
Logging (red)
RSyslog Aggregator
Elastic Search
Mesos
Kibana
Ingestion Microservice
Kublet
Syslog
Query Microservice
Kubernetes
Ingestion Microservice
Kublet
Syslog
Query Microservice
Kafka
Load balancers
Legacy Microservice
Cassandra



# Upsides to Distributed Schedulers



# How to choice your abstractation




# Questions ?