Containers in LHCbDIRAC…

… or “How a small idea can turn into a big amount of work”

Christophe HAEN

7th DIRAC User Workshop
29-31 May 2017

Not a status report

The journey is more interesting

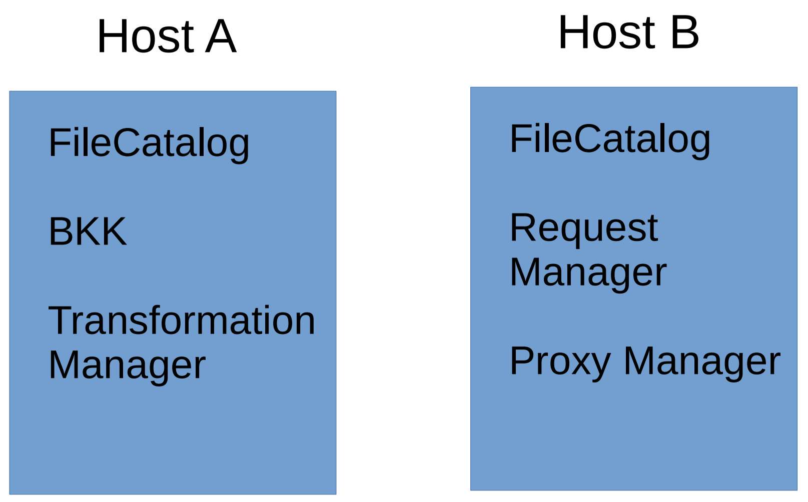
- What choices where made ?
- Why ?
- In practice, how were they applied ?
- Were they good choices ? (make a guess…)

- **I hope to give you some thoughts material, not a solution**

Person power

Hence a work going slowly in a world changing quickly...

Current situation

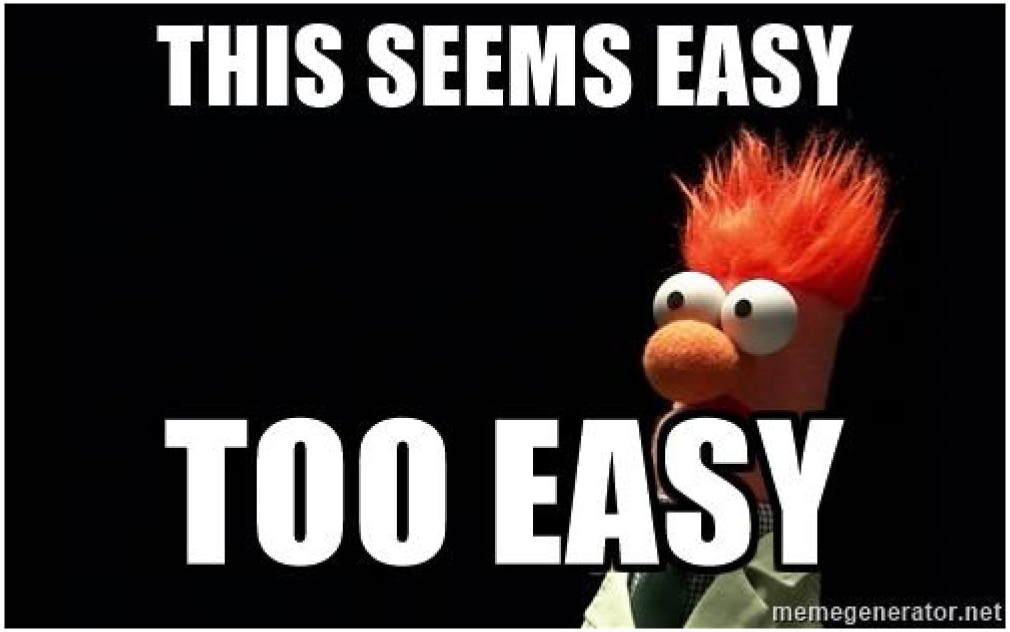


- Static installations
- Placement optimization problems
- Low availability
- Painful updates
- Risk of heterogeneity in the configuration

How ?

Promised land = Orchestrator + Containers

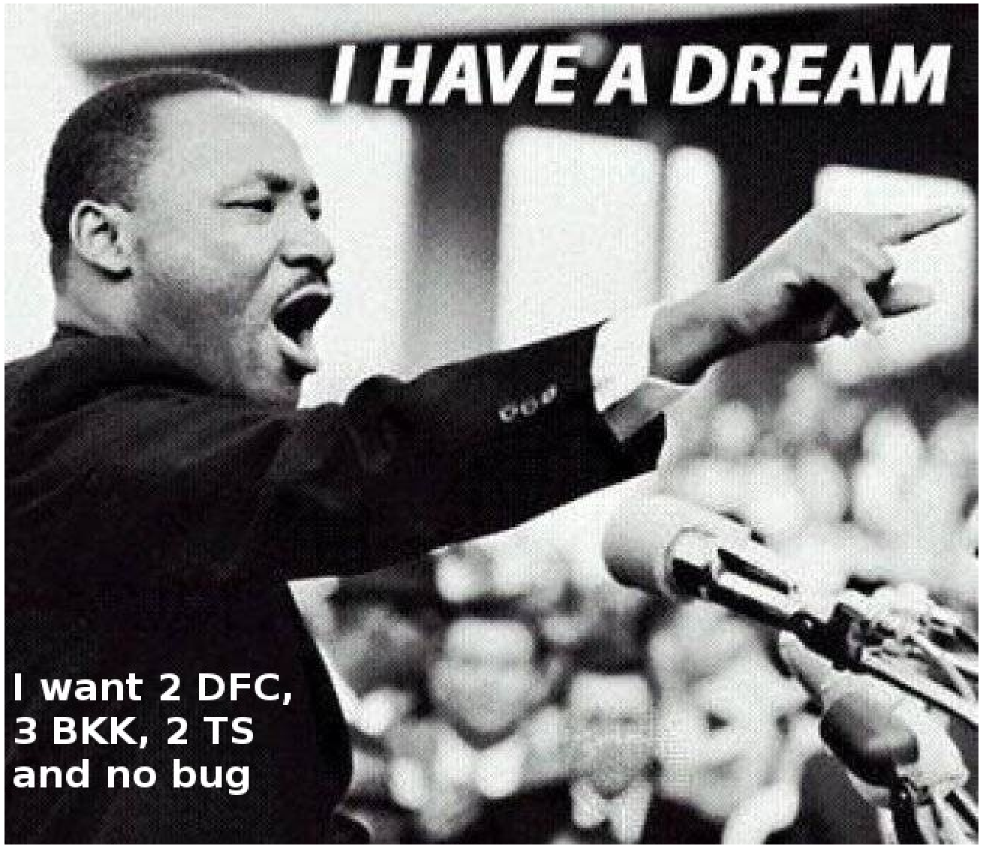
- **Containers: package your application, and ship it all**
- **Orchestrator: runs “somewhere” what you tell it to.**



Let’s package LHCbDIRAC

- Docker, because de facto standard
- Registry integrated to CERN gitlab

I have a dream



Let “something” run it “somewhere” for you

Let’s package LHCbDIRAC

- **What do you put in your container?**
  - Ideally, everything...
- **But maybe not so ideal:**
  - Secrets
  - Configurations
  - Quickly changing information

Let’s package LHCbDIRAC

- **LHCbDIRAC relies a lot on the concept of host for its core infrastructure**
  - Quite antagonist with “running anywhere”
- **Inside the container: just the code/binaries**
- **From the host:**
  - Certificate
  - CRLS
  - Configuration
- **One image to run any setup anywhere**

Orchestration

- **Quite a hype**
  - Give it resources and todo list, and let it handle it
- **Started a year ago: things have changed (quickly)**
- **3 main actors:**
  - Docker swarm
  - Kubernetes
  - Mesos

Orchestration

- **Docker swarm: seemed the least flexible with leastfeatures**
- **Kubernetes: looks good, but very service oriented**
- **Mesos:**
  - Very modular
  - Very generic
  - Solid expertise from RAL admin (Andrew Lahiff)
  - It’s a bazooka (and I like bazooka)

MESOS

- Runs “tasks” on “slaves”
- “Slaves” have “resources” to offer (cpu, mem, etc)
- “Resources” are offered to “Frameworks”
- “Frameworks” contains your work description

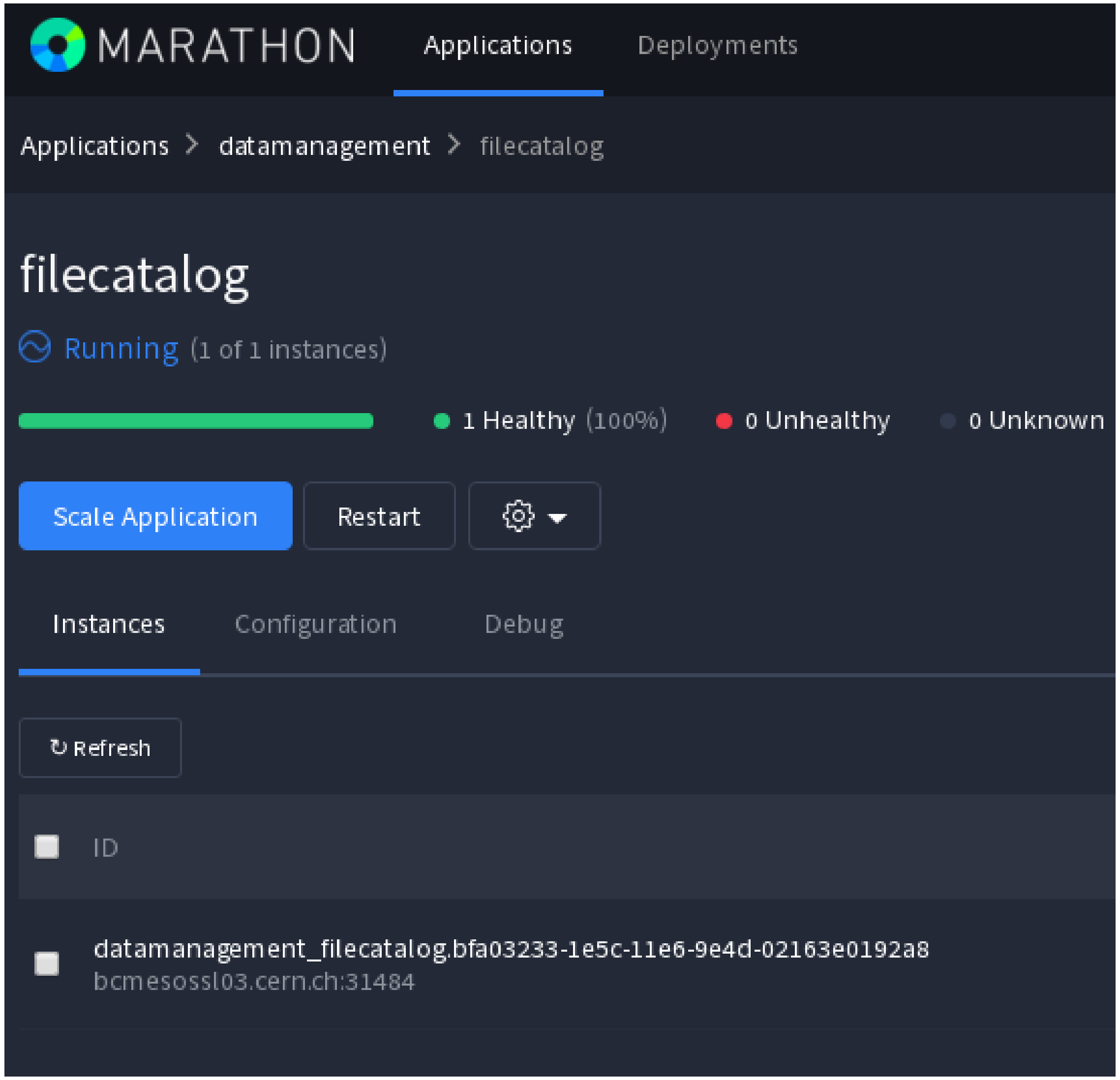
Orchestrate LHCbDIRAC

- **Certification setup:**
  - No impact on production
  - Still very representative
- **Focus on services first:**
  - Stateless
  - Easily moved and duplicated
  - Can still have “bare metal (on a VM) installation” as failover

Marathon

- Distributed init.d for long-running services
- Web + rest interface
- Placement constraints
- Easy scaling
- Rolling upgrades

DFC in Marathon



filecatalog

Running (1 of 1 instances)

● 1 Healthy (100%) ● 0 Unhealthy ● 0 Unknown

datamanagement_filecatalog.bfa03233-1e5c-11e6-9e4d-02163e0192a8
bcmesossl03.cern.ch:31484

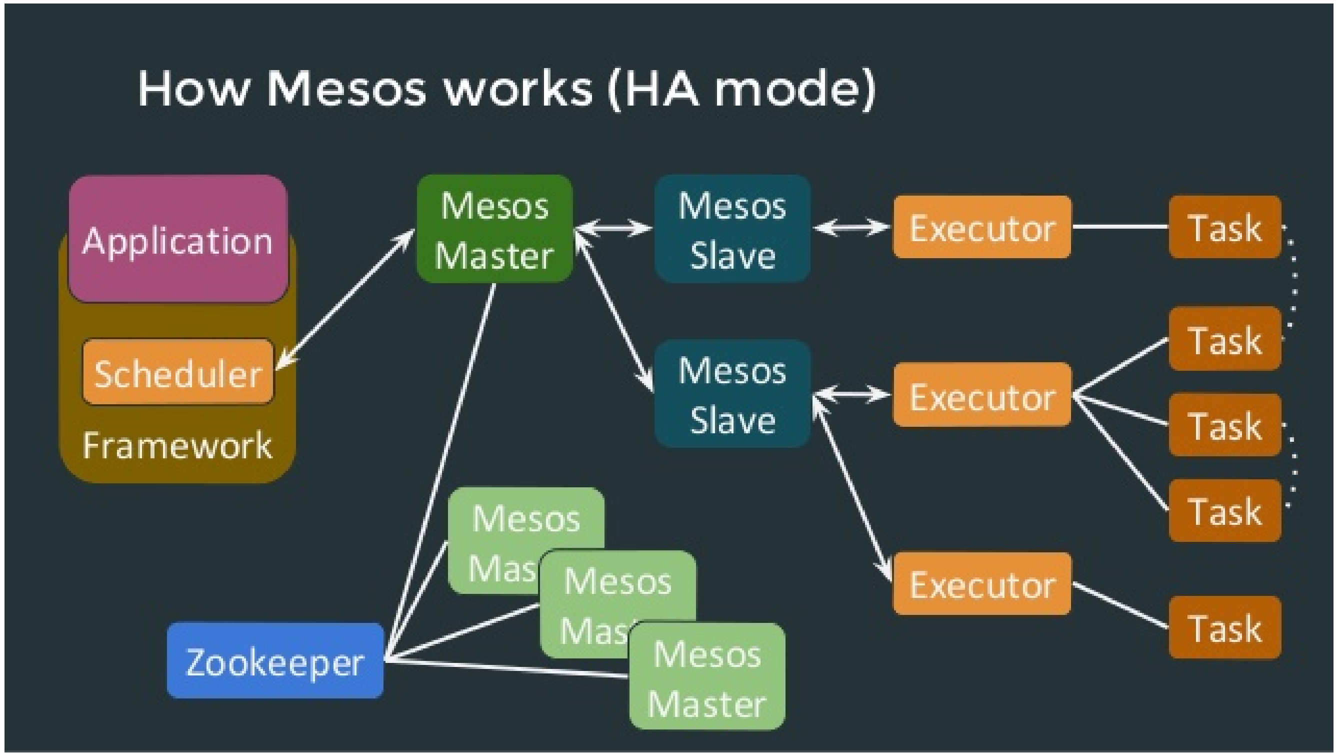
DFC in Marathon

```
{
  "id": "/datamanagement/filecatalog",
  "cpus": 0.8,
  "mem": 600,
  "instances" : 1,
  "cmd" : "dirac-service DataManagement/FileCatalog",
  "container": {
    "type": "DOCKER",
    "docker": {
      "image": "bcmesosms02:5000/registry/lhcbdirac:v8r2p44",
      "portMappings": [
        { "containerPort": 9197, "hostPort": 0 }
      ]
    }
  }
}
```

Good !

- **Easy !**
  - Master: Mesos + Marathon daemon
  - Slave: Mesos agent + docker daemon
- **Now let me just sort out one or two easy detai**

Clusterize the master

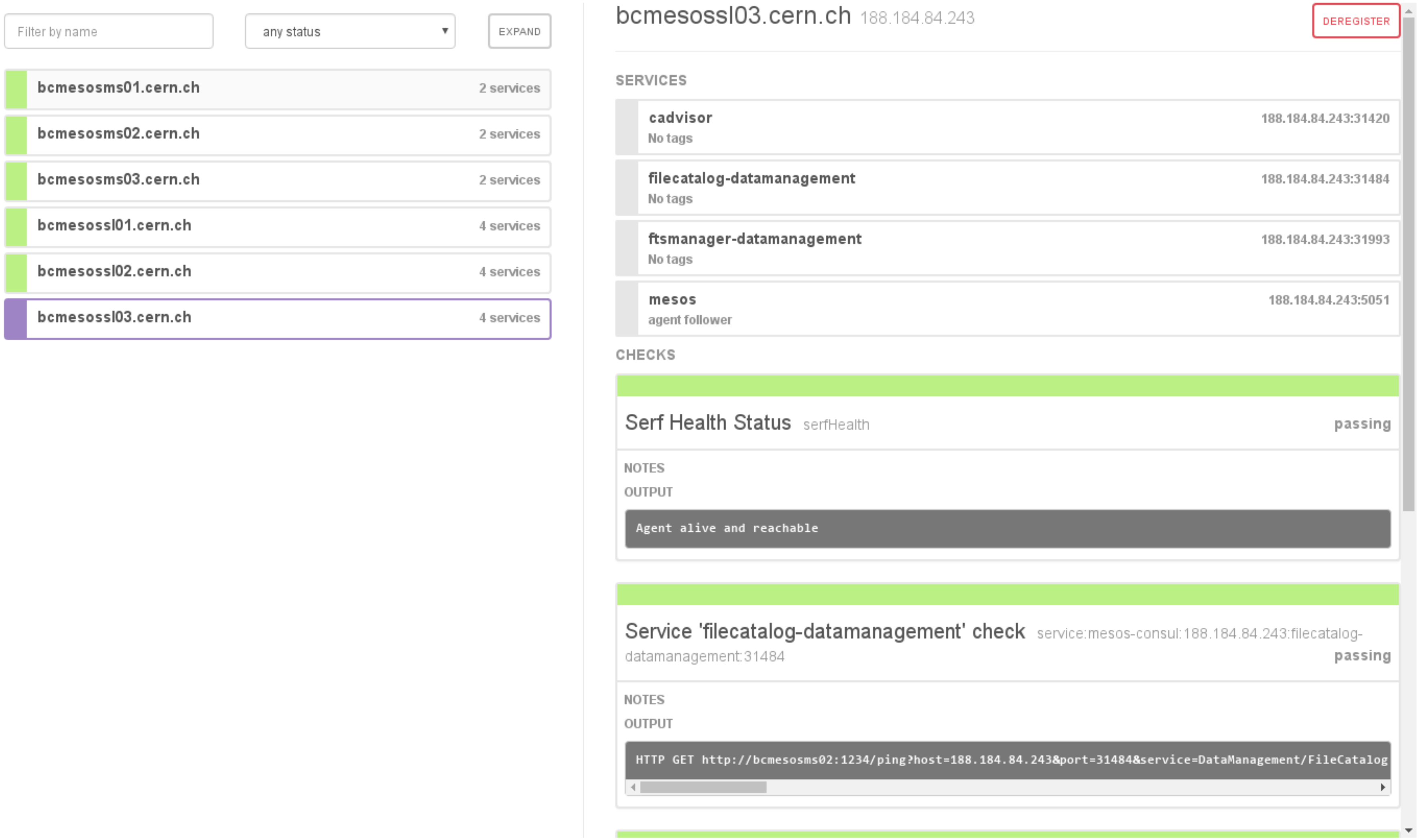


- Zookeeper
- Several masters
- Choose a leader
- Quorum decision
- Failover
- Also for Marathon !

Service discovery

- **What is running ? Where is it ? How do I access it ?**
- **Marathon-lb? No, remember, I like lego**
- **Consul:**
  - Service discovery + health check (see later)
  - Adds a service on every masters and slaves
  - Need to register your services: Mesos-consul (runs as a task in Marathon :-))
  - Use the info: Consul-template (go templating language)

Consul



bcmesosms01.cern.ch 2 services
bcmesosms02.cern.ch 2 services
bcmesosms03.cern.ch 2 services
bcmesossl01.cern.ch 4 services
bcmesossl02.cern.ch 4 services
bcmesossl03.cern.ch 4 services

bcmesossl03.cern.ch 188.184.84.243

SERVICES

cadvisor No tags	188.184.84.243:31420
filecatalog-datamanagement No tags	188.184.84.243:31484
ftsmanager-datamanagement No tags	188.184.84.243:31993
mesos agent follower	188.184.84.243:5051

CHECKS

Serf Health Status serfHealth passing

NOTES

OUTPUT

```
Agent alive and reachable
```

Service 'filecatalog-datamanagement' check service:mesos-consul:188.184.84.243:filecatalog-datamanagement:31484 passing

NOTES

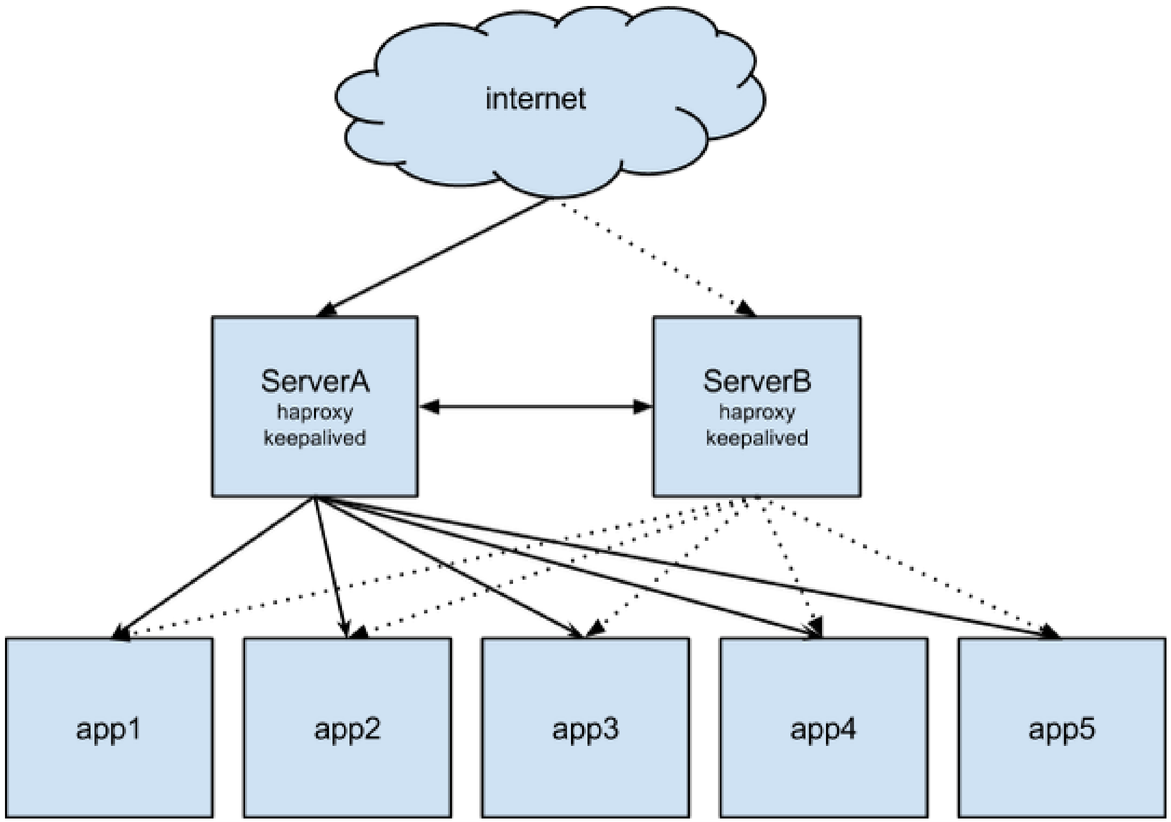
OUTPUT

```
HTTP GET http://bcmesosms02:1234/ping?host=188.184.84.243&port=31484&service=DataManagement/FileCatalog
```

HAPROXY

Powering Your Uptime

Use HAProxy as a gateway to redirect to the correct containers



Health monitoring

- **Marathon:**
  - Failed container are restarted automatically
  - You can monitor the behavior of your container
- **Consul:**
  - Unhealthy entities not returned when Consul is queried
  - Host: nagios checks (generates Mesos slave whitelist)
  - Services: Docker exec/HTTP/TCP (generates HAProxy conf)

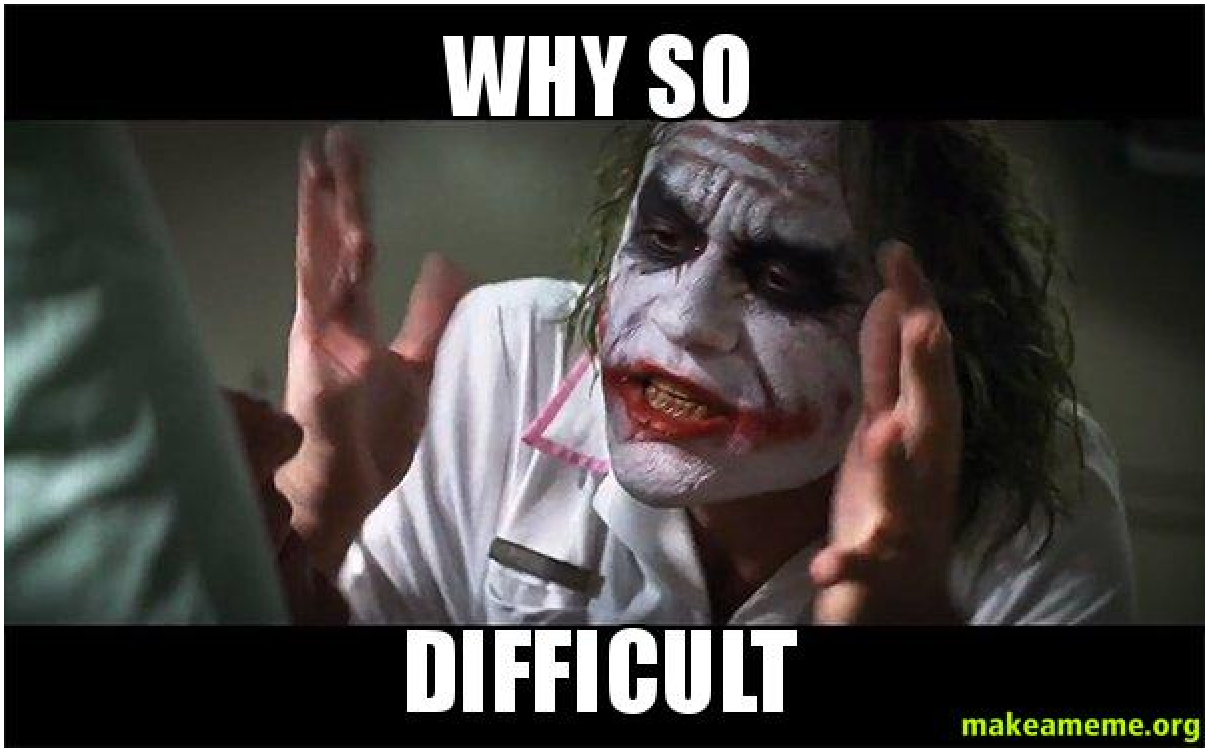
Performance monitoring

- **The users are happy, but you ?**
- **Performance monitoring:**
  - Consul + custom script + influxdb + grafana
  - Still not completely convinced...

Logging

- **"ssh myhost; grep error /var/log/myService.log" does not really work anymore**
- **You need a central logging:**
  - Need an infrastructure (Logstash/Elasticsearch). Where do you get it from ?
  - Either your code is instrumented
  - Or you have to capture the output of your container and ship it (docker-gen + filebeat)

Persist it: puppet

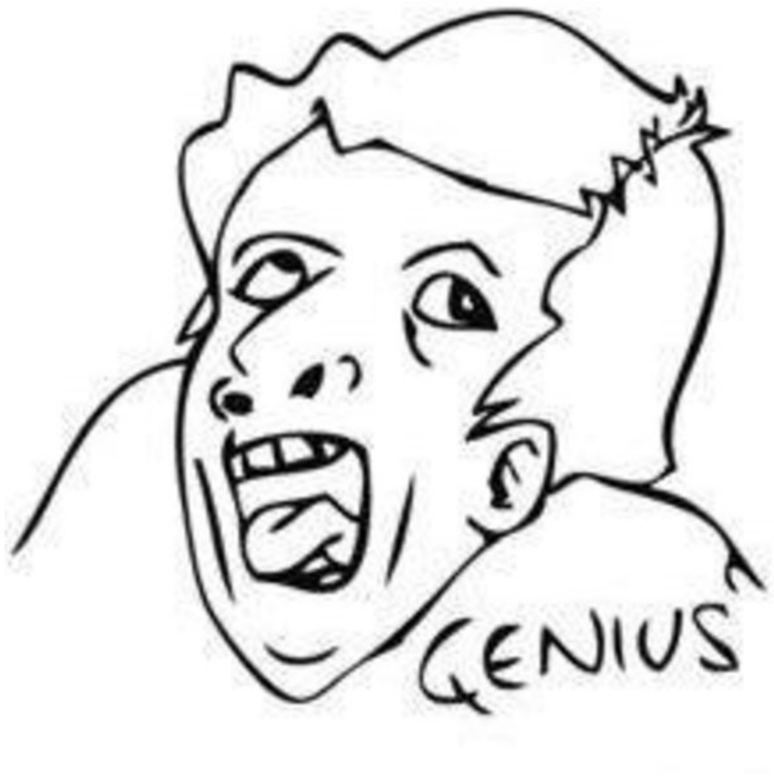


Be ready to invest quite some time if you are not puppet fluent

Security

- Agents and framework are authenticated with shared secrets (Teigi is great!)
- IPTables to open the management ports only within the cluster
- No SSL/TLS communication (would require special compilation)
- Web app: no nice authentication provided out of the box

Secure the web interface: SSO



- **Hide your web app behind an Apache front end**
- **But SSO:**
  - Supports only one app per machine
  - Forces you to disable SELinux (!!!!!!!!!!!!)
  - Requires manual registration (cannot do it all In puppet)
- **Result in complex Apache config and hacking Mesos/Maraton web apps → still not perfect**

Operational aspects

- **The web interface is awesome for routinely aspects**
  - Scale up/down a service
  - Add/remove a service
  - Find out on which machine a given service is running
  - etc

- **More exceptional operations are better done with REST interface**

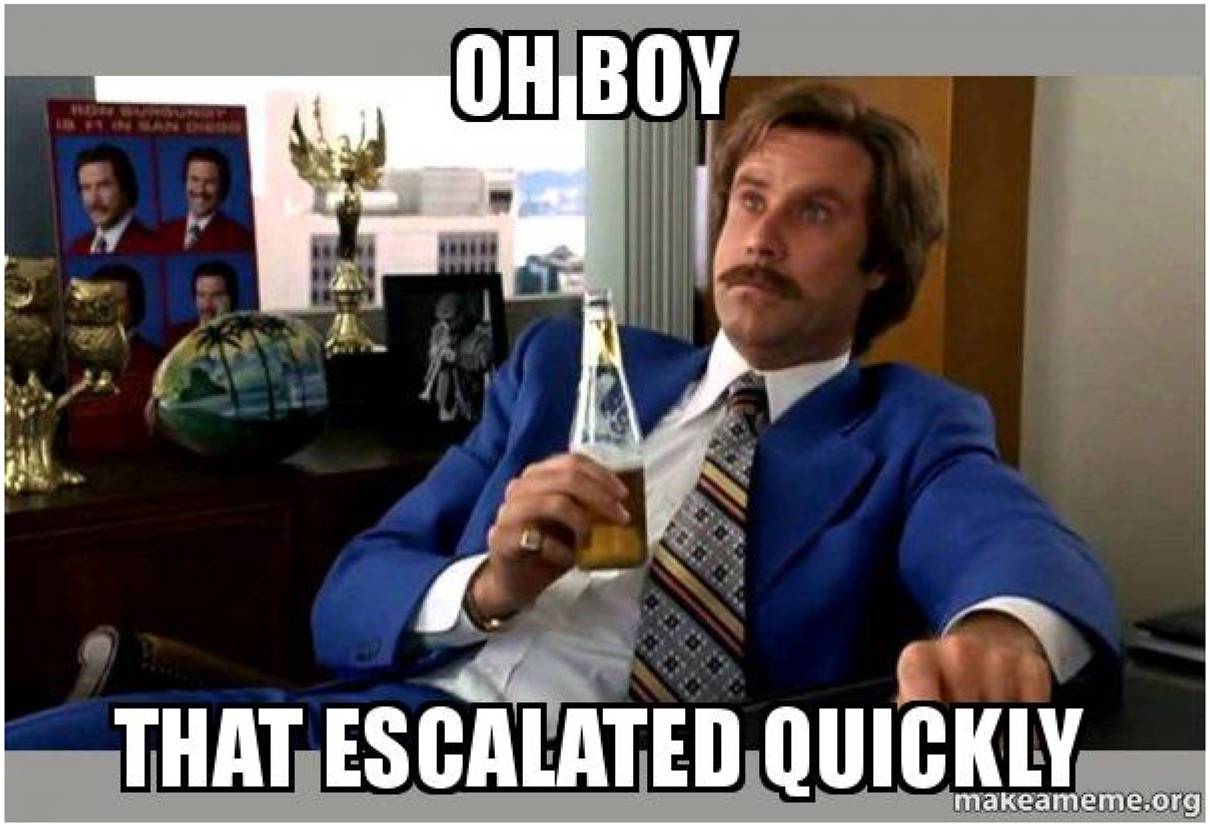
Operational aspects

- **Deploying a new LHCbDIRAC release:**
  - Creating the release tarball and put it on AFS
  - Build and publish the docker image
  - Update the running version →needs to be done for each task definition
- **Gitlab-ci does it all for us !**
  - Tag in LHCbDIRAC triggers build and release of tarball and docker image
  - Commit in another repository updates the running configuration of Marathon

“User/GEOC” point of view

- **Overall: really great**
  - High availability
  - No more heterogeneity problem
  - Releases so much easier
  - No placement problems
  - Nice web interface (for viewing)
  - One json file to administrate everything
  - It all seems simple

“Infrastructure” point of view



“Infrastructure” point of view

- **Do not underestimate the complexity of it all**
  - Requires quite some sys/net admin skills
  - It’s not just one RPM to install
  - The underlying infrastructure ends up being really big
  - Everything can fail at once
- **Writing doc is not enough, you need to train people**

Where do we stand today?

- **The cluster meets its purpose and is stable**
- **Everything in puppet**
- **Certification services are running on it for 6 months**
  - One major issue (Puppet3 → Puppet4)
  - Few small issues for admin
- **Releases are now easy and quick**
- **Almost nothing is LHCbDIRAC specific !! :-)**
- **Still some polishing needed:**
  - Monitoring (working solution, but not convinced)
  - Logging (working, but better coming)
  - Pointed out some bugs in DIRAC
  - Persistent data is a problem
- **Need to train the team**

Would I do it this way again?

- **YES !**
  - Extremely instructive:
    - Skills ++ for me :-)
    - Many lessons learned
  - We have a working system !!
  - Side effect improvements of the production system
  - Docker images available (dev, hackathon, tests, etc)

Should you do it the same way?

- **Things are moving quickly out there**
  - Kubernetes: moving at an incredible speed
  - DCOS: Mesos based full system in a box
  - Docker swarm: better and better (for some use cases)
- **One cluster to rule them all ?**
  - Maybe not...
  - CERN Magnum infrastructure improved a lot
- **In any case: think carefully, and really, talk to people**

What I miss in DIRAC

- **Respect the TLS RFC !**
  - Requires M2Crypto
- **Centralized logging**
  - Alexandre (ISIMA student) working on it
- **Less stateful agent**
  - Graceful stop seems complicated in Mesos
  - MQ and consumers schema ?

"That's all Folks!"