CHICAGO
INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE 2015

goto; conference

Deploy Like A Boss

Oliver Nicholas

follow us @gotochgo

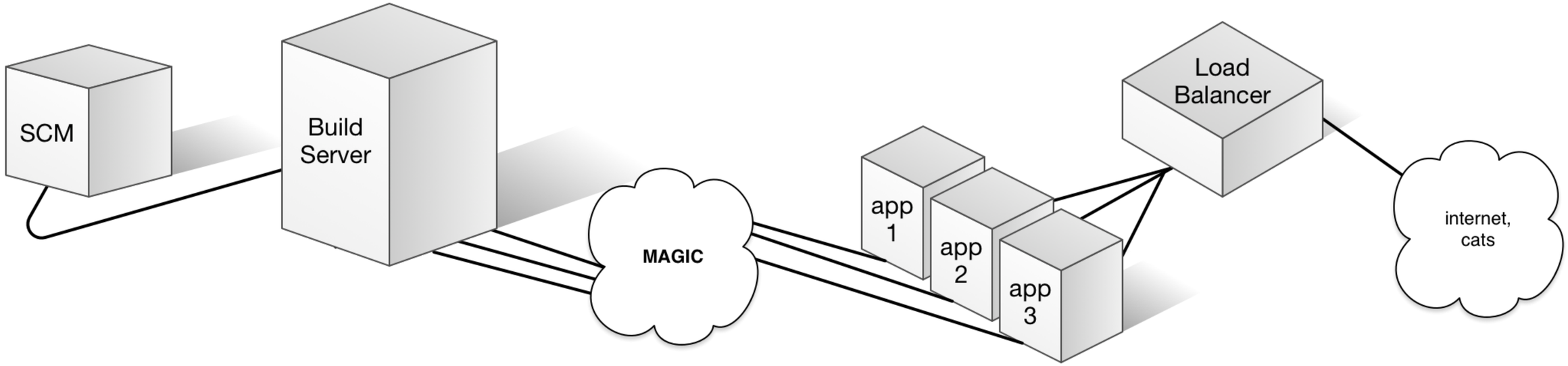
Conference: May 11-12 / Workshops: 13-14

DEPLOY LIKE A BOSS

THE JOURNEY FROM 2 SERVERS TO 20,000

UBER

THE DEPLOYMENT PIPELINE



UBER TECHNOLOGIES, INC

BUSINESS METRICS

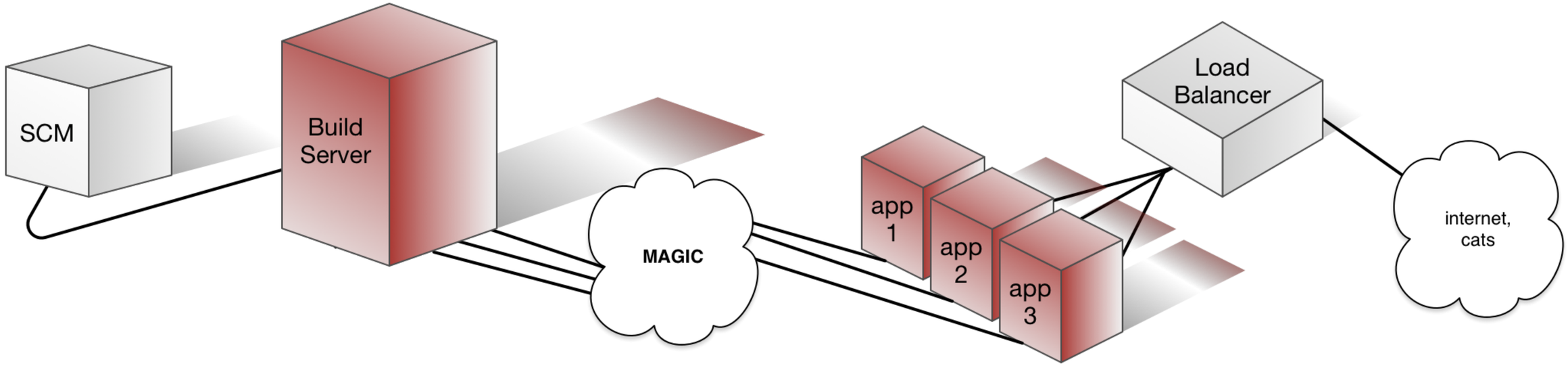
- 311 Cities
- 57 Countries
- 1,000,000+ Rides per Day

ENGINEERING METRICS

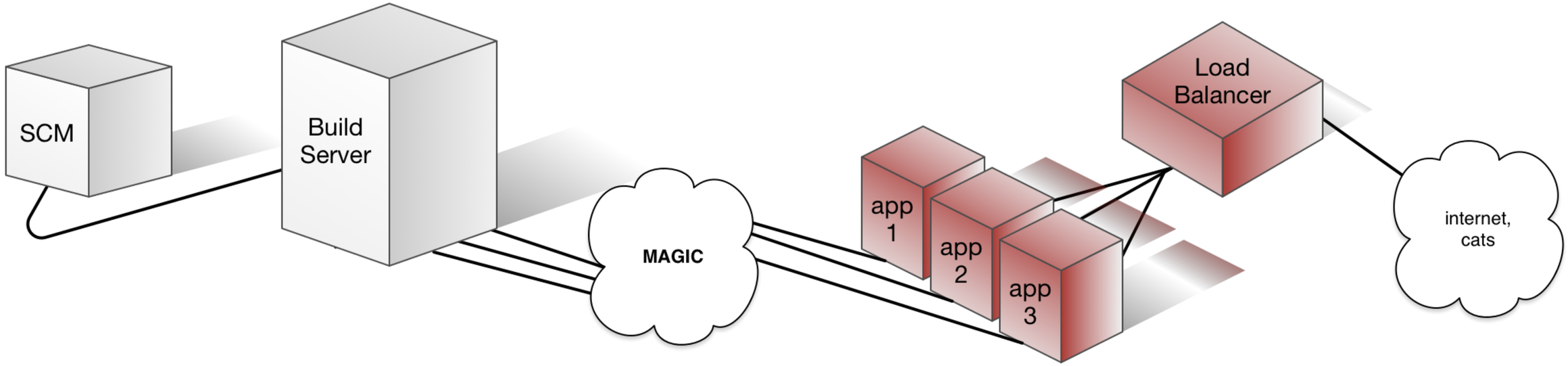
- 300+ Services
- 2500 servers per DC
- 2-4 Datacenters (ABS)
- 10's of deployments per day

OLIVER NICHOLAS

DISTRIBUTION



ORCHESTRATION



THE EARLY DAYS

"DISASTER DRIVEN DEVELOPMENT"

EARLY-STAGE DEPLOYMENT SYSTEMS

DEPLOY AND PRAY

SIMPLE UNIX TOOLS:

```
1. history | grep scp
2. tar zcvf -  proj/ | ssh user@server "cat > /var/www/proj.tgz && tar xfz proj.tgz && /etc/init.d/project restart"
3. rsync -avz proj user@server:/var/www/ && ssh user@server /etc/init.d/project restart
```

DRAWBACKS:

- Not atomic
- Performance impact during deploy
- No load balancer management
- Brittle

PROS:

- We don't care about any of the drawbacks yet.

THE MIDDLE AGES

"GOOD ENOUGH FOR WAY TOO LONG"

MIDDLE-STAGE DEPLOYMENT SYSTEMS

EASY TO BUILD, HARD TO LEAVE

OPEN-SOURCE SOLUTIONS:

- Capistrano, Fabric
- Convenience wrappers for shell scripts.
- Encapsulate most of the SSH complexity.

TYPICAL FLOW:

- Build Code
- Sync to deploy targets
- Take target out of LB
- Shutdown app
- Swap symlink
- Start app up
- Healthchecks, Warmup
- Put target back into LB
- Move onto next host

EXAMPLE:

```
bigo@bigo-proforce/~$ cat deploy.rb
set :application, "uber"
set :scm, :git
set :repository, "git@git.uber.com:/proj.git"
set :user, "uber"
role :app, "server1", "server2", "server3"
set :deploy_to, "/var/www/"

namespace :deploy do
  task :restart, :roles => :app do
    run "/etc/init.d/proj restart"
  end
end

bigo@bigo-proforce/~$ cap deploy
```

MIDDLE-STAGE DEPLOYMENT SYSTEMS

EASY TO BUILD, HARD TO LEAVE

PROS:

- Open-source libraries
- Lots of recipes out there for special cases
- Realistically, **good enough** for tens of servers

CONS:

- Not good enough for hundreds of servers.
- Still essentially utilizing tar-scp pipeline
- Though you can extend away from that (git pull from deploy target hosts)
- Poor support for multi-user environments (no deploy lock)

ASIDE:
BUILD DISTRIBUTION

MOVING BITS FROM A -> B

BUILD DISTRIBUTION

ALL DRESSED UP WITH SOMEWHERE TO GO

NAIVE DESIGN

- Early systems almost always just consist of tar-scp or equivalent
- Single build+distribute server
- SPOF, slow (good way to overload a rack switch and cause a packet storm)

MORE SCALABLE APPROACHES

- Tiered....rsync hosts :)
- HDFS or equivalent distributed filesystems
- BitTorrent

THE MODERN ERA

MODERN ERA DEPLOYMENT SYSTEMS

CALL ME, MAYBE

OUT WITH THE OLD...

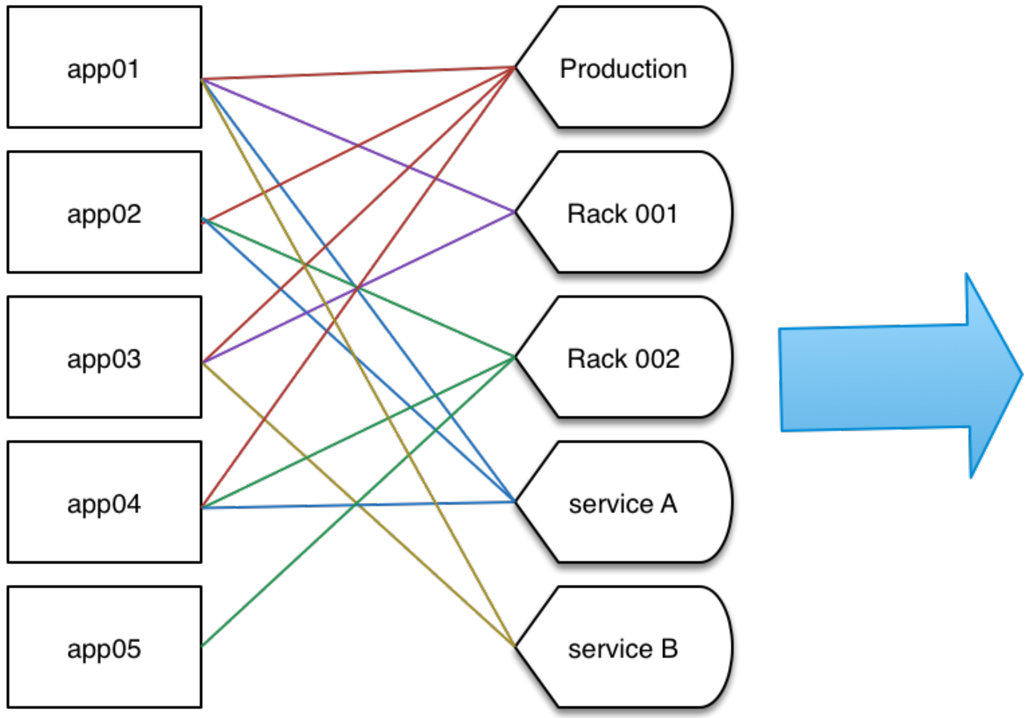
- Earlier systems were all **push** based.
- When scaling, active O(1) -> O(**n**) work where **n** scales with traffic tends to go sideways.
- Consistency issues when servers are inaccessible.

IN WITH THE NEW..

- Pull ("poll") based model.
- Leaf nodes (app servers) contact deploy master, rather than the other way around.
- Goal-based - the hard work happens on the app servers.
- **Database-driven server lists.**

CLUSTO

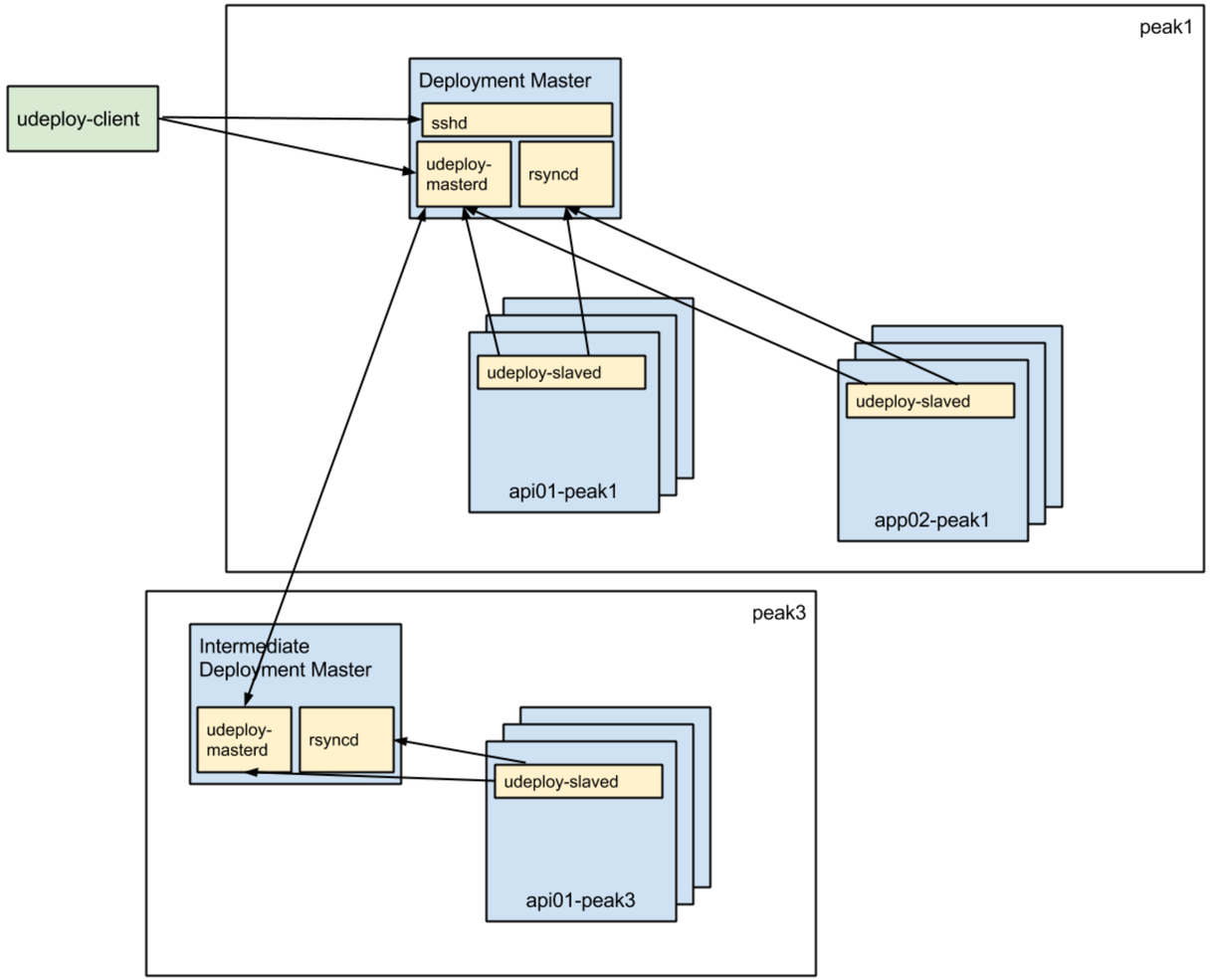
STOP LOSING SERVERS



app01	production, rack001, serviceA, serviceB
app02	production, rack001, serviceA
app03	production, rack001, serviceB
app04	production, rack002, serviceA
app05	rack002

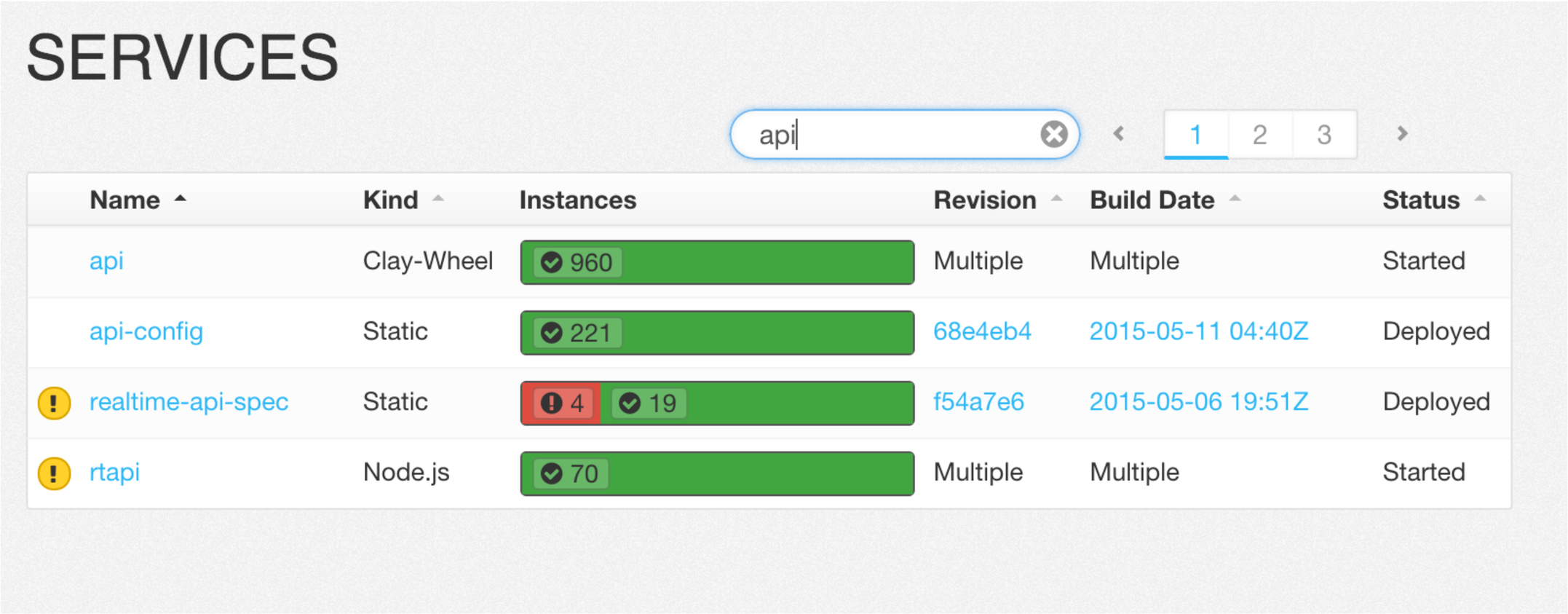
UDEPLOY ARCHITECTURE

- Coordinator in each datacenter.
- Worker on each deploy target.
- Workers poll Coordinator for target state.
- Coordinator has a priori concept of deploy procedure, according to hardware database and deployment policy.
- Deployment policy is customizable for things like stuck machines.
- Intermediate Coordinators take cue from primary; multiple datacenters deployed in parallel.



UDEPLOY TOUR

SERVICE SELECTION



UDEPLOY TOUR

SERVICE VIEW

SERVICE | API

DEPLOYMENT | PIPELINES | BUILDS | TASKS | EVENTS | CONFIGURATION | STATS

canary: 36

production: 924

DEPLOYMENT
canary

ACTIVE REVISION
deployed/another/2015-05-11T04-42-59-5-g574185a

BUILD DATE
2015-05-11 04:49Z

STATE
Started

Upgrade | Start | Restart | Stop

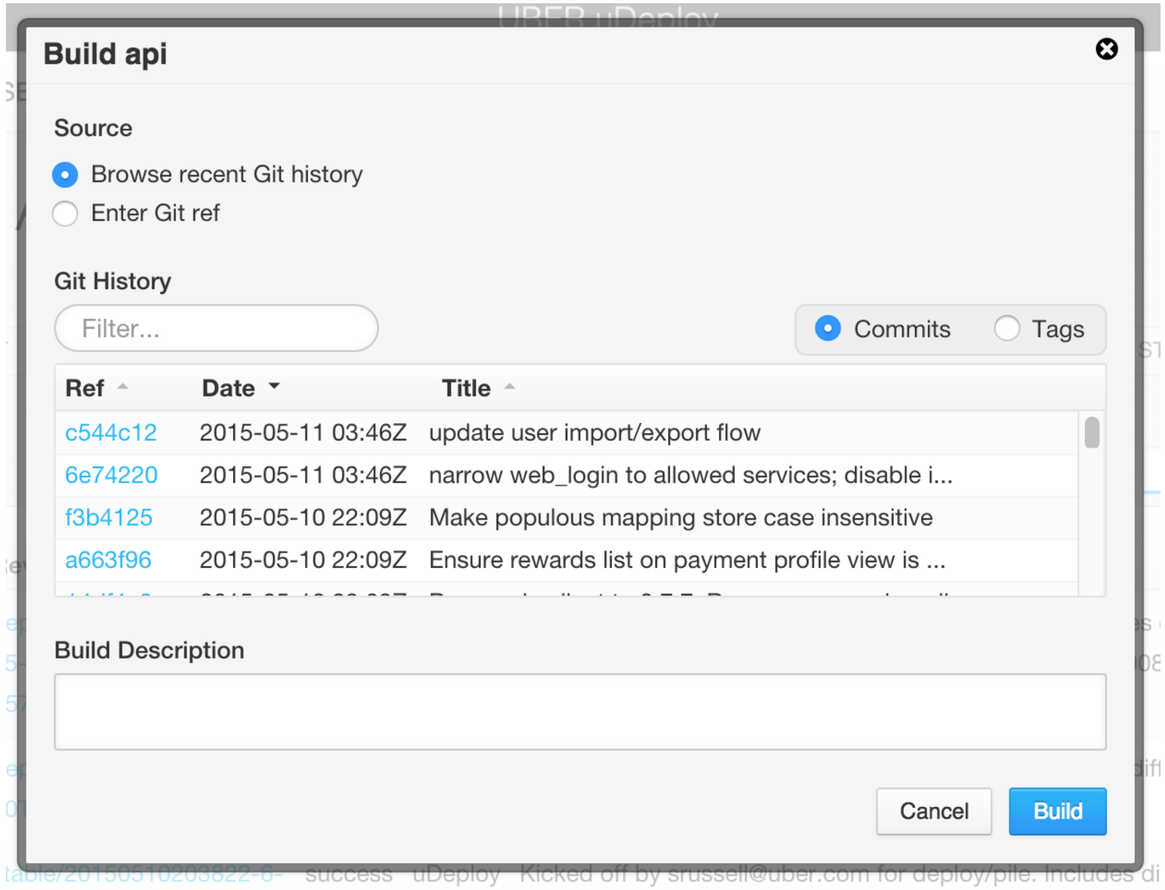
Filter...

1 2 3

Host	Pipeline	Type	Status	Revision	State	Messages
adhoc03-sjc1	us1	adhoc	✓	deployed/another/2015-05-11T04-42-59-5-g574185a	Started	
adhoc04-sjc1	us1	adhoc	✓	deployed/another/2015-05-11T04-42-59-5-g574185a	Started	
adhoc05-sjc1	us1	adhoc	✓	deployed/another/2015-05-11T04-42-59-5-g574185a	Started	
adhoc06-sjc1	us1	adhoc	✓	deployed/another/2015-05-11T04-42-59-5-g574185a	Started	

UDEPLOY TOUR

BUILD VIEW



UDEPLOY

STILL NOT PERFECT

STRENGTHS

- Coordinator only does **passive** work - responding to requests - in O(n) of # of Workers.
- High-level fault detection and rollback triggers
- Offloads most of the work to the Workers.

DRAWBACKS...

- Static server pools!
- Deployment relies on the static mapping of service -> server.
- Deploys must be **windowed** rather than full red/black.

THE FUTURE

THE FUTURE: MESOS

THE MISSING BUILDING BLOCK

MESOS IS A...

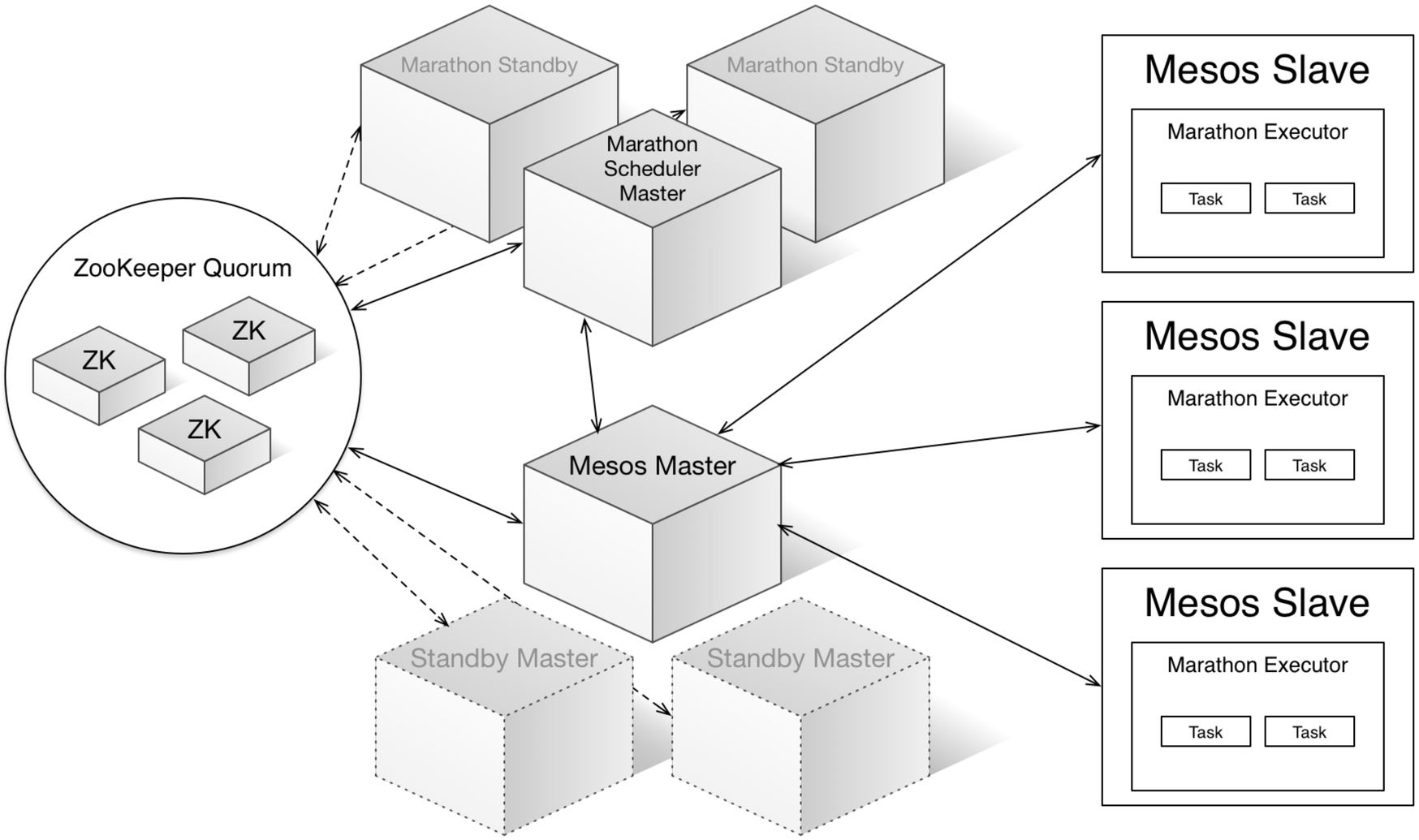
- ...resource management engine.
- ...pluggable conduit for scheduling tasks against server resources.
- "...distributed systems kernel"

THE FUTURE: MESOS

THE MISSING BUILDING BLOCK

HOW IT WORKS

- Slave makes resource offer to the Master ("I have 22 CPUs and 32GB of RAM available").
- Master sends offer to its frameworks.
- Framework accepts portion of offer and informs master ("Take 1 CPU and 64MB of RAM and run `yes`").
- Next slave resource offer takes decremented resources into account.

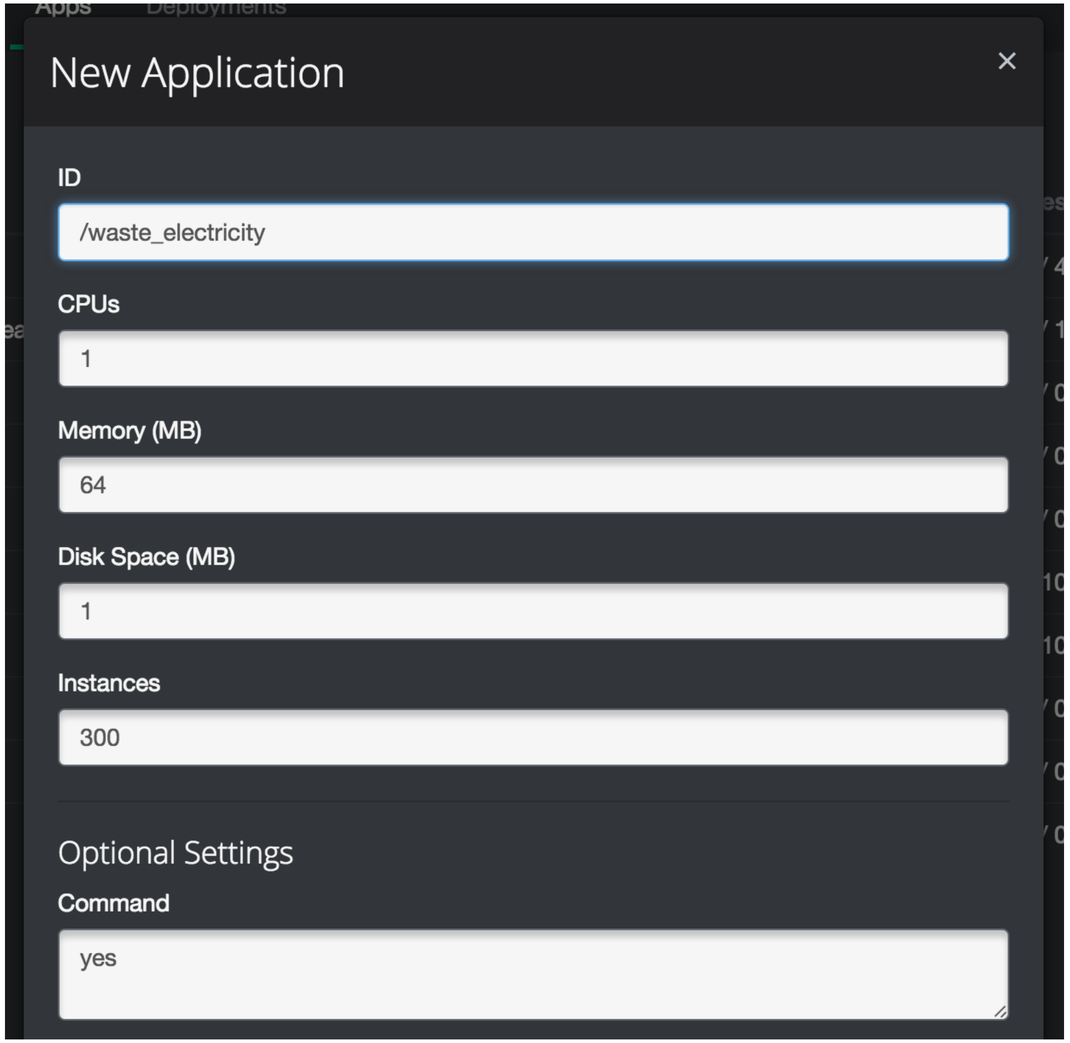


THE FUTURE: MESOS AND MARATHON

A FRAMEWORK FOR LONG-RUNNING TASKS

MARATHON:

- "A cluster-wide init and control system for services in cgroups or Docker containers"
- Built-in support for various deployment policies, healthchecks, automated rollbacks.
- Rich constraint system:
  - "distribute app across racks"
  - "no more than one instance per server"
  - "only deploy to machines with kernel version > 3.13"



THE FUTURE: MESOS AND MARATHON AND UDEPLOY

NIRVANA?

UDEPLOY KEY FEATURES:

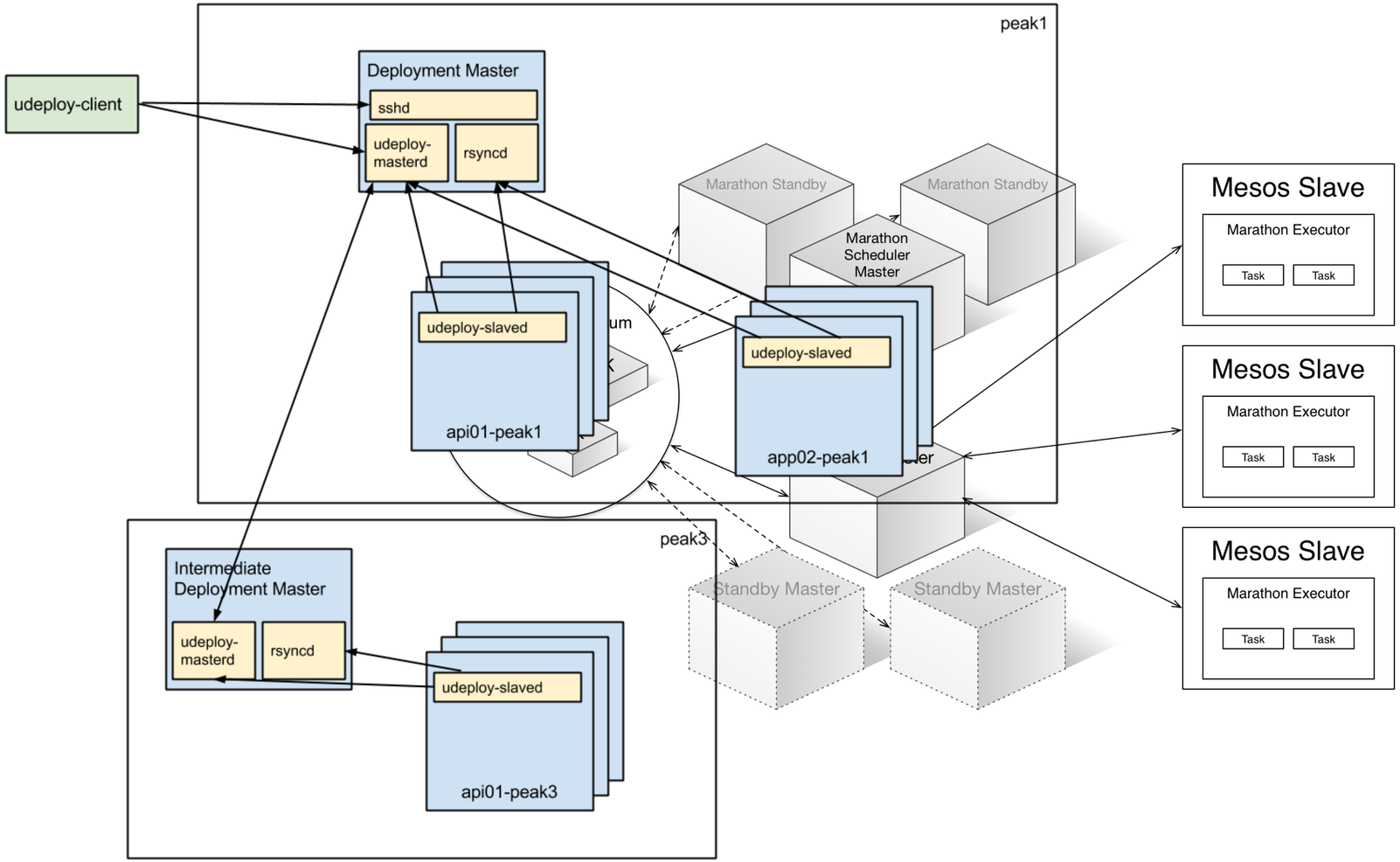
- Beautiful interface for kicking off builds.
- Authentication and Authorization support.
- Coordination across multiple datacenters.
- Higher-order healthchecking/rollback functionality.

MESOS/MARATHON KEY FEATURES:

- HA masters with ZK coordination already built for us.
- Shifts focus from instances to clusters.
- Homogenizes cluster resources - "server" no longer the basic unit of organization.
- Rich features around resource distribution.

THE FUTURE: MESOS AND MARATHON AND UDEPLOY

NIRVANA?



MESOS/MARATHON/UDEPLOY AUTOSCALING

FIRE YOUR OPERATIONS TEAM?

SHIFT OF FOCUS MAKES SCALING EASIER:

- Mesos+Marathon helps us solve the "doing the work" part of scaling.
- We let the software handle all the placement questions.

AUTOMATE THE **DECISION** TO SCALE:

- cgroups and Docker containers deployed under Mesos
- fine-grained CPU utilization metrics for each service instance
- good proxy for scaling decisions

MORE MESOS FRAMEWORKS

KITCHEN SINK INCLUDED

OPEN SOURCE MESOS FRAMEWORKS:

- Task Schedulers
  - Aurora, Marathon, Hadoop, Spark, Storm, Chronos
- Lots more
  - HDFS, Mysos, Cassandra, HyperTable
  - ElasticSearch
  - Jenkins
  - MPI, Chapel

Mesosphere plug!

THE LONG JOURNEY

DEPLOYMENT SYSTEMS

- Start from simple shell pipelines
- Advance to Capistrano/Fabric-type DSLs
- Invert control flow once scale is achieved
- Abstract away individual servers at even larger scale
- Along the way, higher-availability code distribution models

Go forth and Deploy like an Evil Genius.

CHICAGO
INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE 2015

goto; conference

Questions?

Please remember to evaluate via the GOTO Guide App

follow us @gotochgo

Conference: May 11-12 / Workshops: 13-14