Exploring the Cloud

Ezra Zygmuntowicz
http://engineyard.com

What is Cloud Computing?

What is Cloud Computing?

IaaS

What is Cloud Computing?

IaaS

SaaS

What is Cloud Computing?

IaaS

SaaS

PaaS

What is Cloud Computing?

IaaS

HaaS

SaaS

PaaS

What is Cloud Computing?

IaaS

HaaS

?

SaaS

PaaS

The Definition has been Clouded...

If you listen to the pundits, anything on the internet is now ‘cloud computing’

I'm Going to Define Cloud Computing as:

Virtualized Compute, Memory and Storage

API accessible

Resources scale up/down on demand

AWS is the prototypical example

So what does it really take to work with these low level cloud resources?

A Different Way of Thinking

It’s all about the Automation

Three Pillars of Automation

1. State Based Configuration Management

Repeatable

Idempotent

1. State Based Configuration Management

Treat an instance like a referentially transparent function

f(config) -> Fully Configured Instance

Calling f(config) will *always* create the same instance whenever config == config

1. State Based Configuration Management

Treat instances as throw away

Prefer rebuilding from scratch

Only persist what truly needs to be persistent

1. State Based Configuration Management

Puppet

CFEngine

Soon to be released Mystery Project ;)

2. Ad Hoc Change

Run this command on this server now

Deploy this new code to these servers now

Gather information from these servers now

2. Ad Hoc Change

Vertebra

Capistrano

2. Monitoring

You can’t fix what you can’t see

2. Monitoring

Nagios
Munin
Collectd
Ganglia
Monit

code name: Engine Yard Elastack

code name: Engine Yard Elastack

Engine Yard abstracted from Engine Yard

code name: Engine Yard Elastack

Clouds are too low level for average humans

code name: Engine Yard Elastack

First Target: AWS

code name:
Engine Yard Elastack

Custom Gentoo Linux

State based config management

Ad hoc Change

Extensive Monitoring

24/7 support

High Level Cloud

code name:
Engine Yard Elastack

Bridge multiple providers

Highly tuned databases on EY

Elastic app servers on ec2

Will support any compelling new cloud platforms

Vertebra:
p2p Cloud Control

Problem:

Problem:

Booting a cluster is a complex process

Problem:

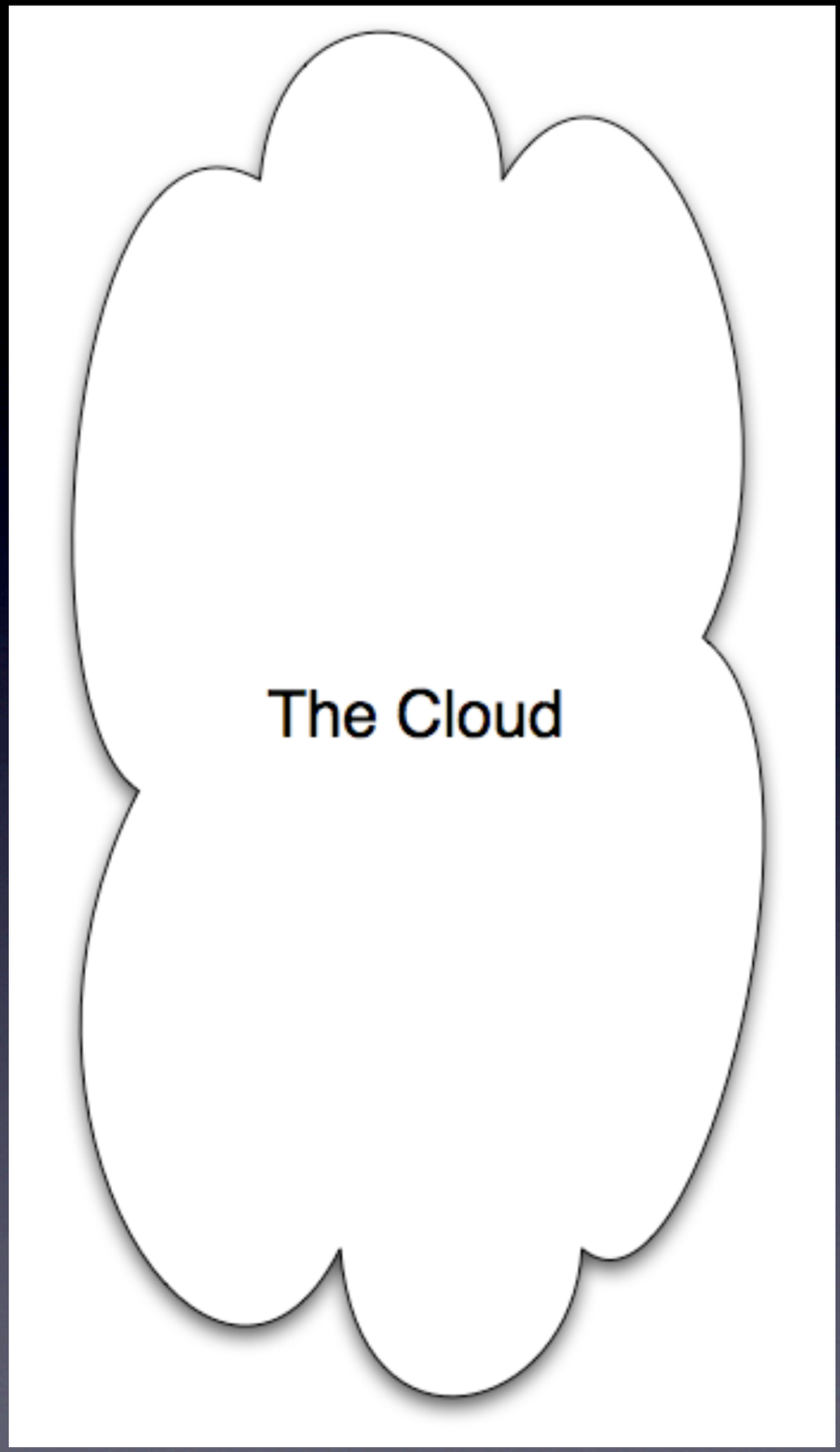
Lots of interdependent tasks to perform

Problem:

No ‘init’ for clusters...

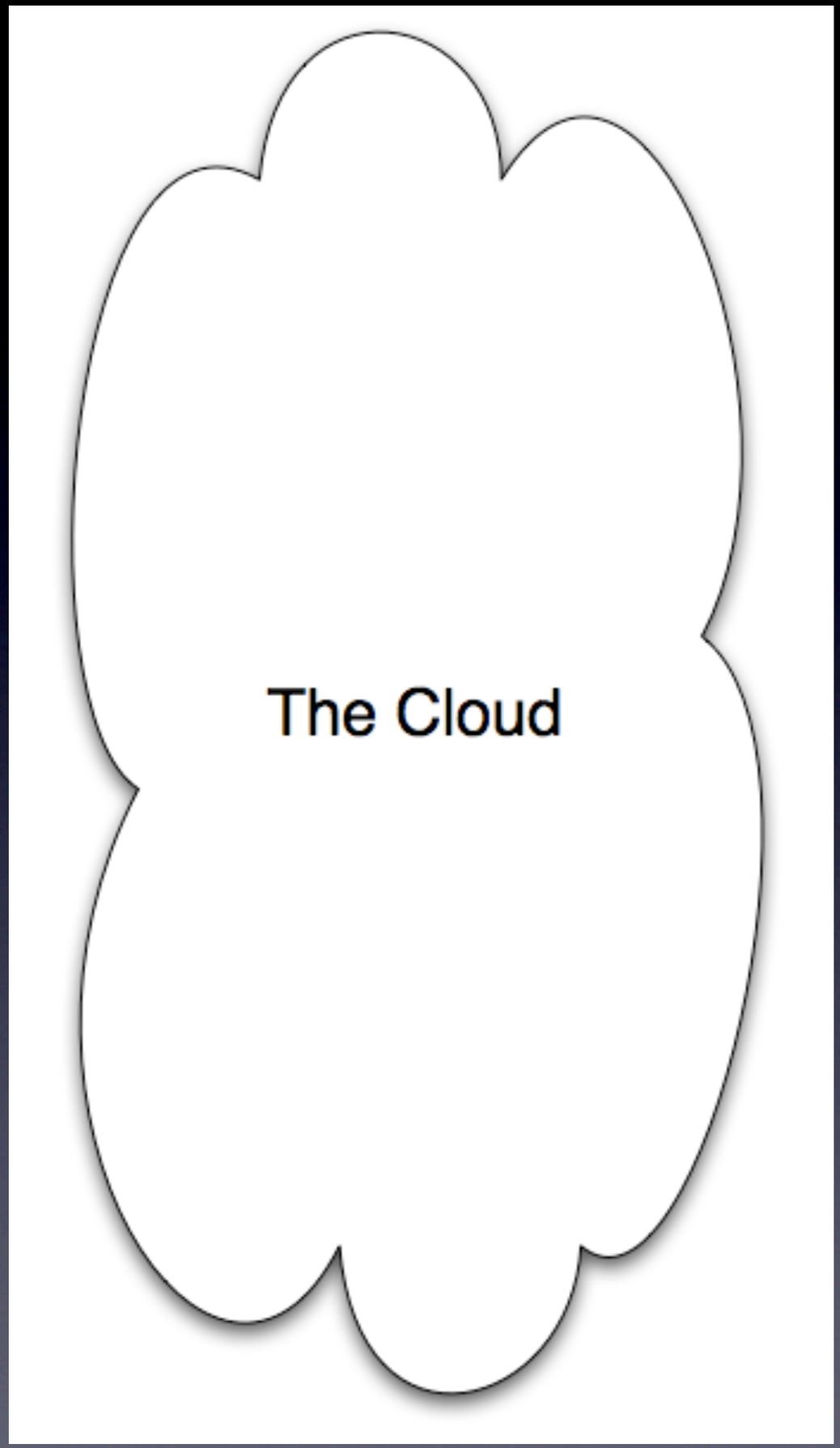
Problem:

Lots of servers to control in the cloud

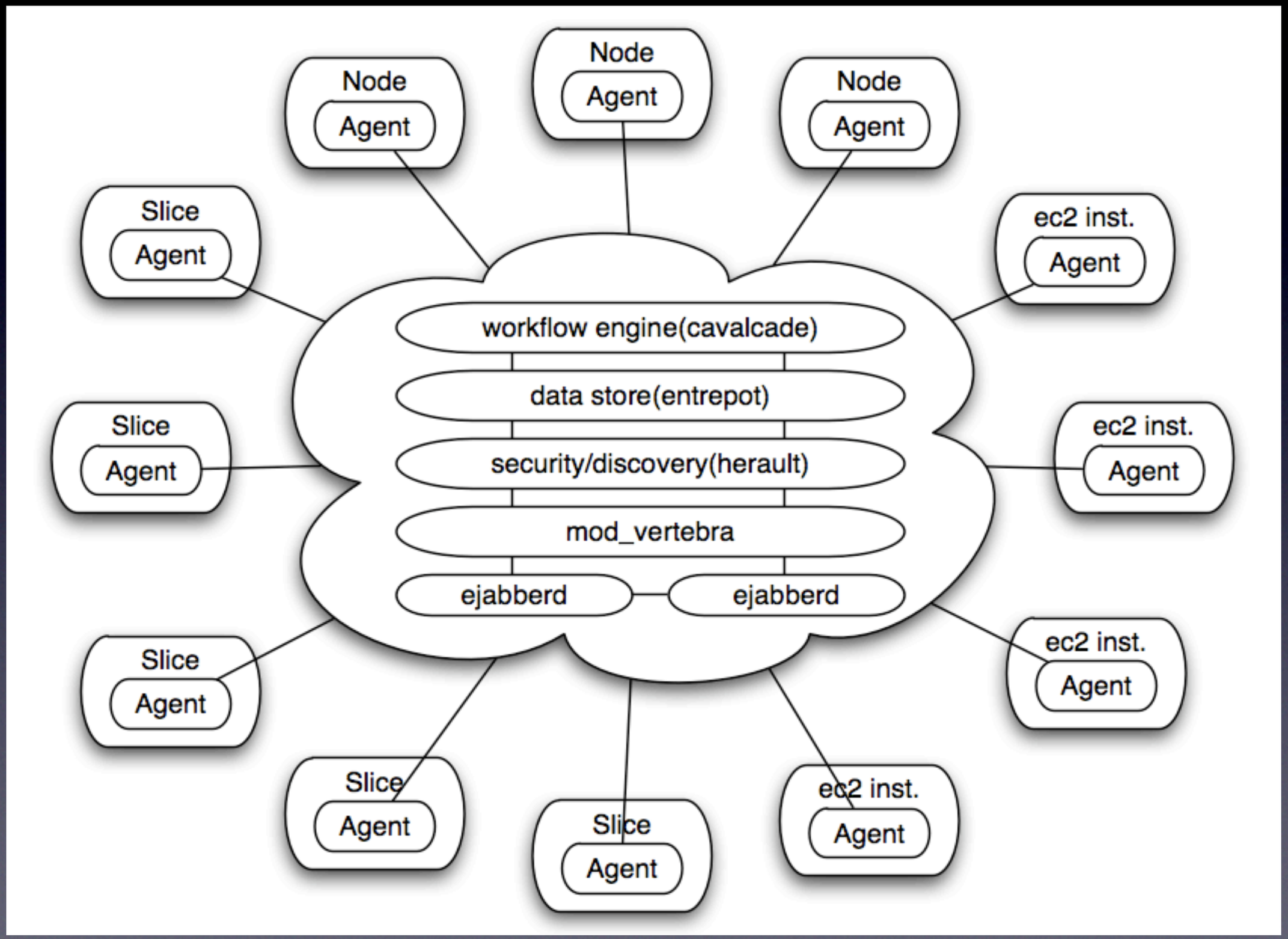


Problem:

Complex interactions and deployment scenarios



Solution: Vertebra



Operations on Resource Sets is Vertebra's Fundamental Abstraction

```
<op token='7e5293c579f4a9fa8da6320e9ef03e3f'
    type='install'>
  <res>/cluster/1</res>
  <res>/slice/42</res>
  <res>/gem</res>
  <string name='gem'>hpricot</string>
</op>
```

Resources are Hierarchical

‘/foo’ == ‘/foo’

‘/’ >= ‘/foo’

‘/foo’ >= ‘/foo/42’

‘/foo/42’ <= ‘/foo’

Resource Set Matching

Needs

/cluster/1
/slice/42
/gem

Resource Set Matching

Needs	Provides
/cluster/1 /slice/42 /gem	/cluster/1 /slice/42 /gem

Resource Set Matching

Needs		Provides
/cluster/1 /slice/42 /gem	>= Matches!	/cluster/1 /slice/42 /gem

Resource Set Matching

Needs		Provides
/cluster/ /slice/42 /gem	>= Also Matches!	/cluster/ I /slice/42 /gem

(slice 42 on *all* clusters)

Resource Set Matching

Needs		Provides
/cluster/1 /slice/ /gem	>= Also Matches!	/cluster/1 /slice/42 /gem

(all slices on cluster 1)

Resource Set Matching

Needs

/cluster/1
/slice/42
/gem

!~

No Match!

Provides

/cluster/1
/slice/42
/xen

(no soup for you)

Resource Discovery

Who provides this set of resources?

Resource Discovery

Who provides this set of resources?

```
>> @agent.discover('/cluster/ey04', '/slice/42')
=> {"jids"=>["ey04-s00042@ey04.engineyard.com/agent"]}
```

Resource Discovery

Who provides this set of resources?

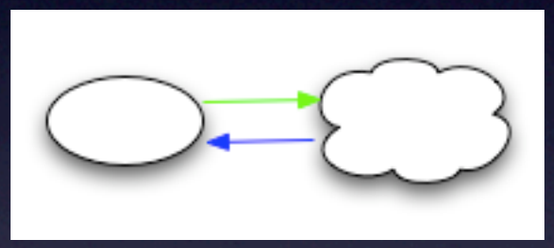
```
>> @agent.discover('/cluster/ey04', '/slice/42')
=> {"jids"=>["ey04-s00042@ey04.engineyard.com/agent"]}
```

Dynamic DNS for Cloud Resources

Map/Reduce
Scatter/Gather Dispatch

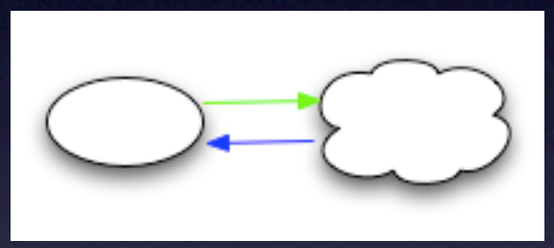
Map/Reduce
Scatter/Gather Dispatch

Discover

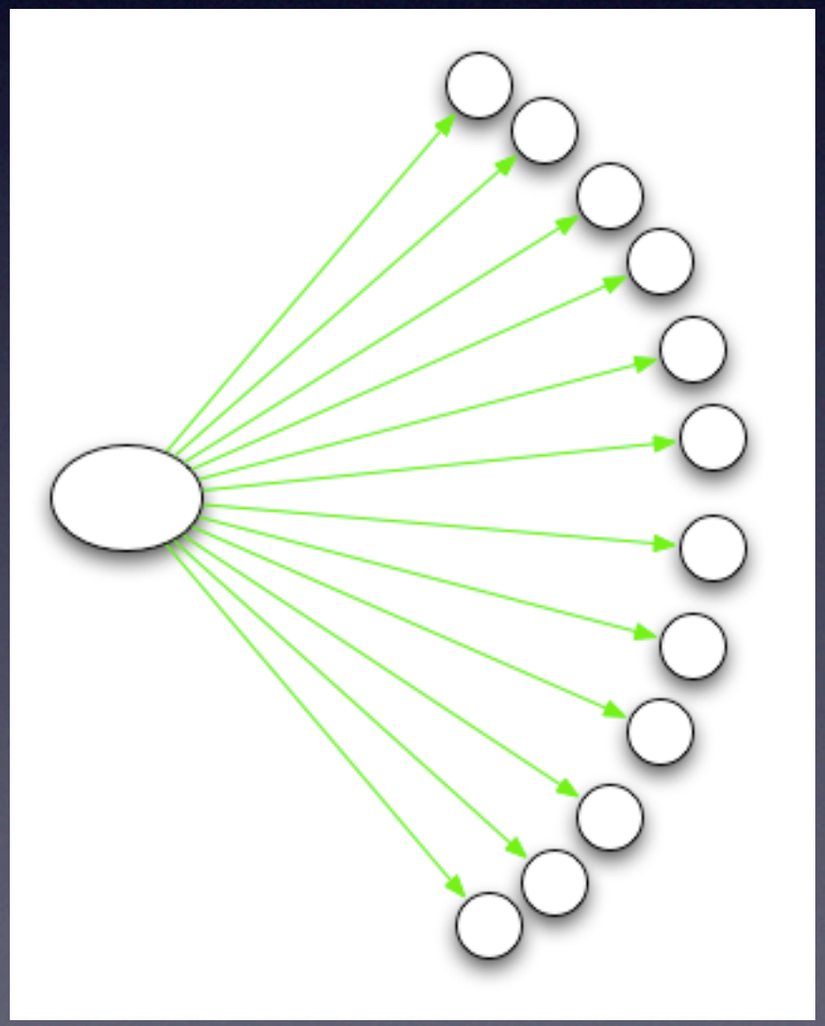


Map/Reduce
Scatter/Gather Dispatch

Discover

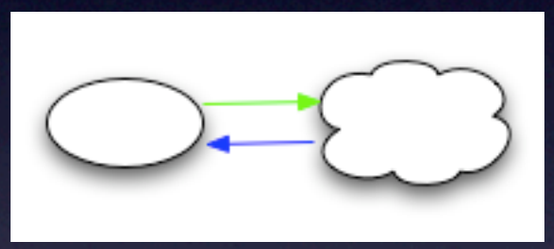


Scatter

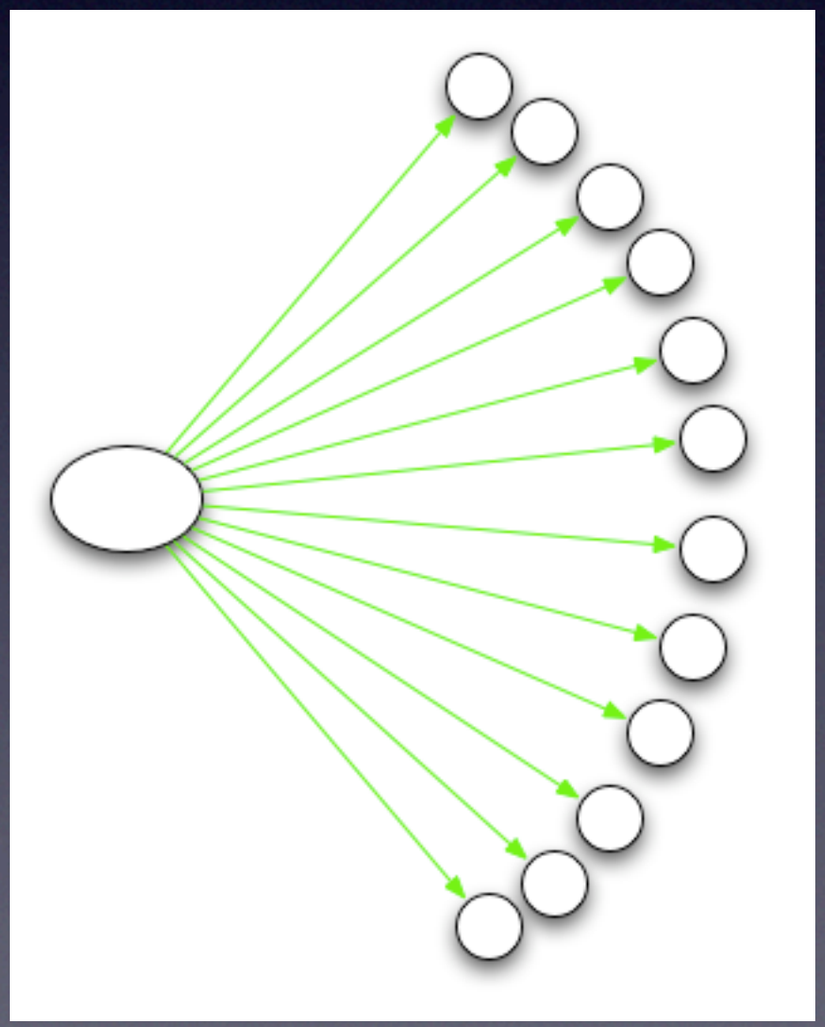


Map/Reduce Scatter/Gather Dispatch

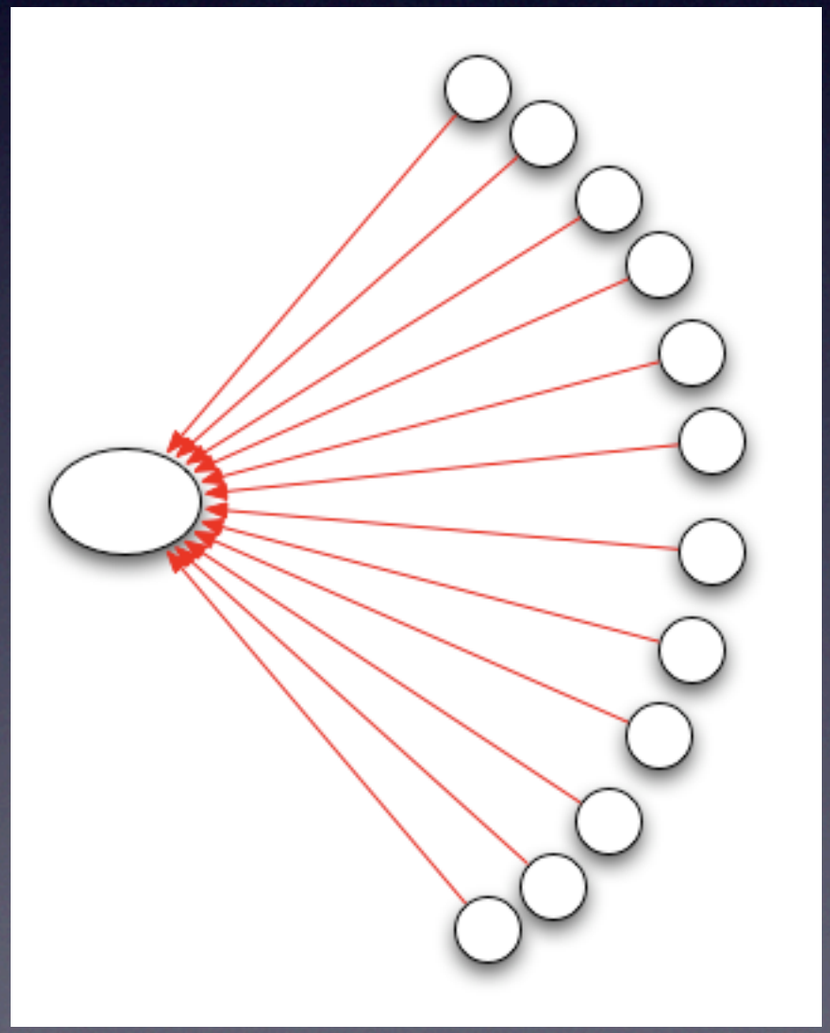
Discover



Scatter



Gather

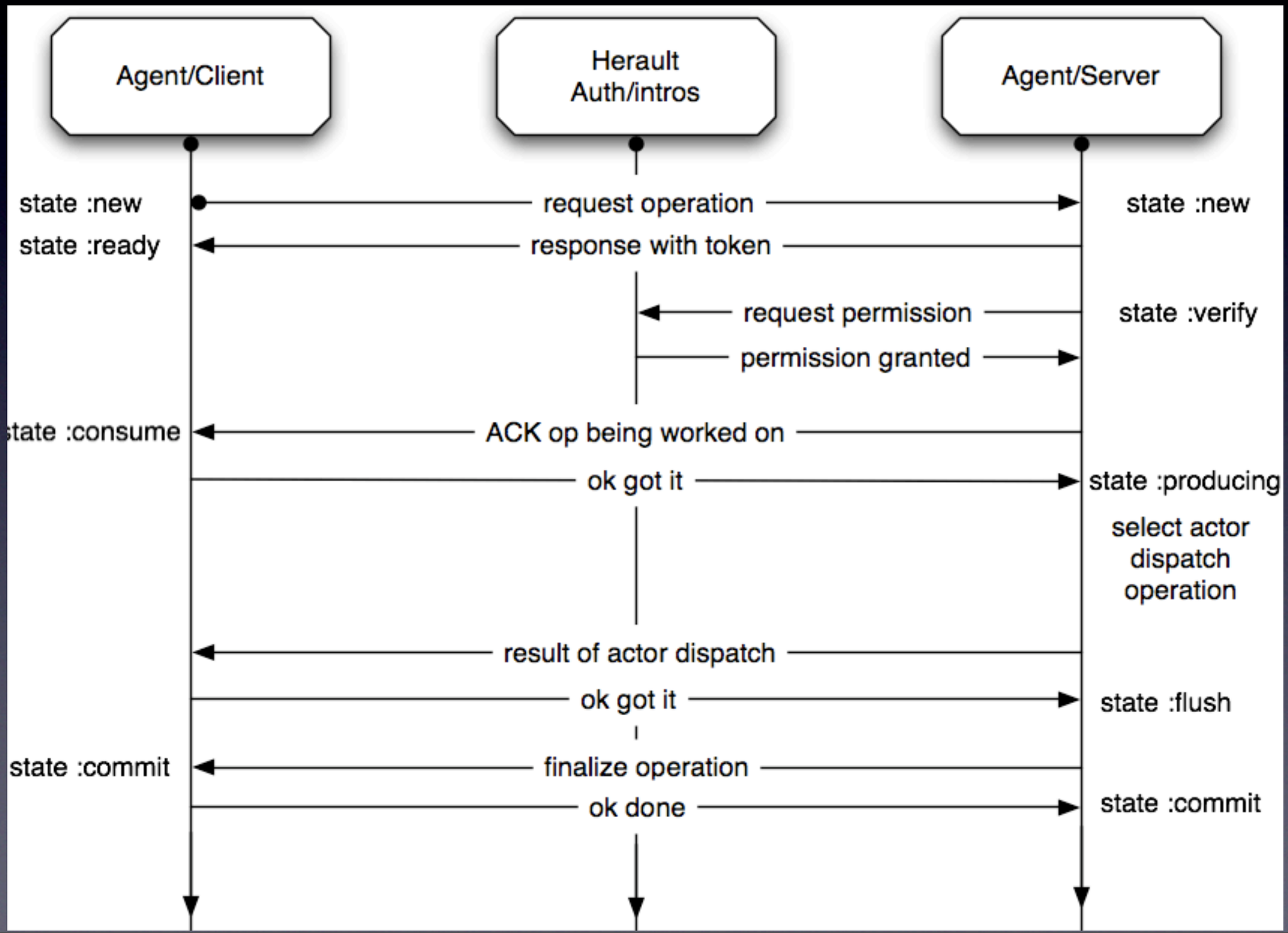


Fault Tolerant Protocol on top of XMPP

Each agent can act as a Client
or a Server or both at once.

State machines for Client/Server
Protocols

Fault Tolerant Protocol on top of XMPP



Key/Value Data Store

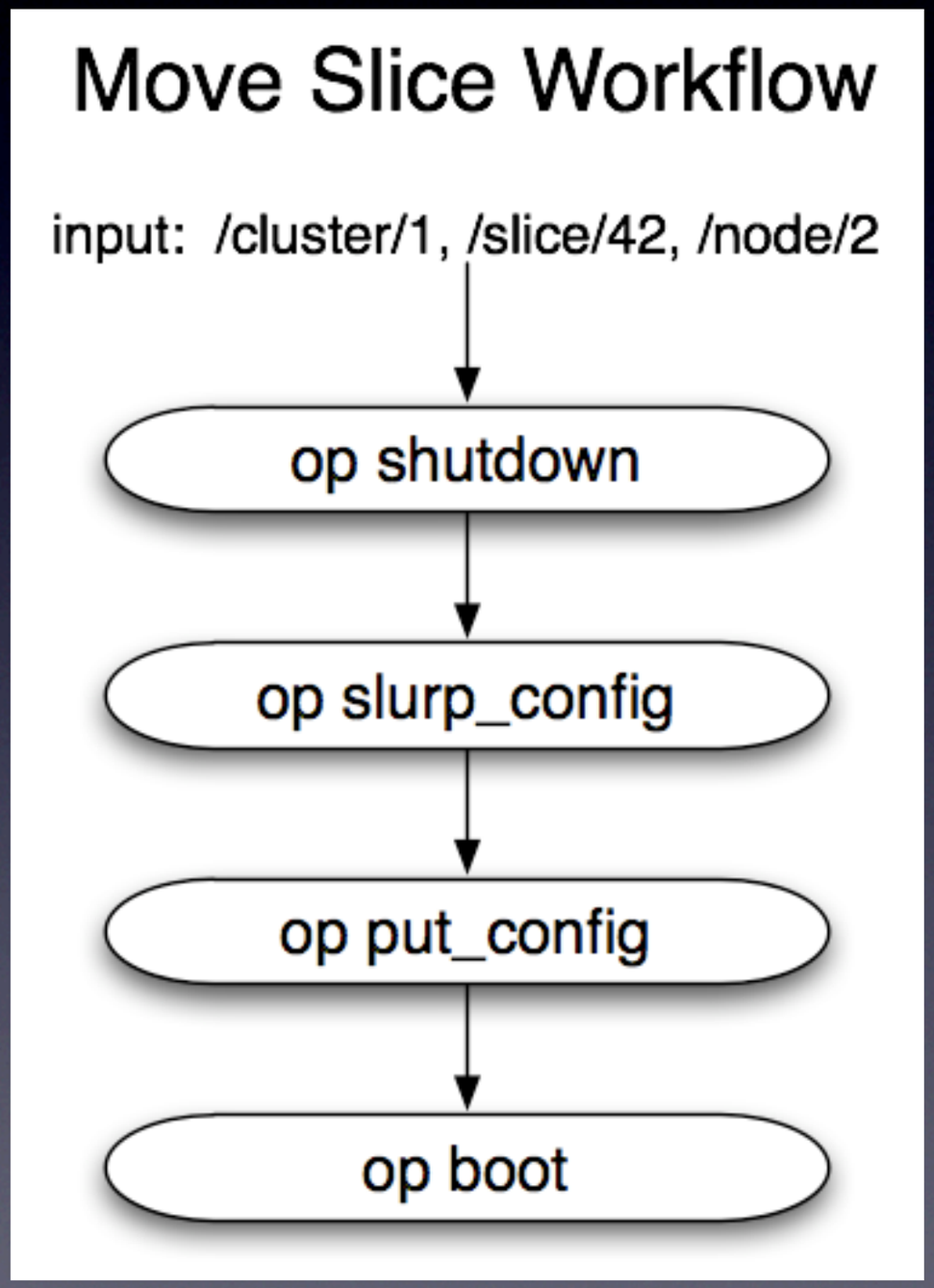
Queried via an ‘op’

Uses Resource ‘Sets’ as
compound keys

key:
/cluster/1
/customer/foobar

value:
{... arbitrary hash ...}

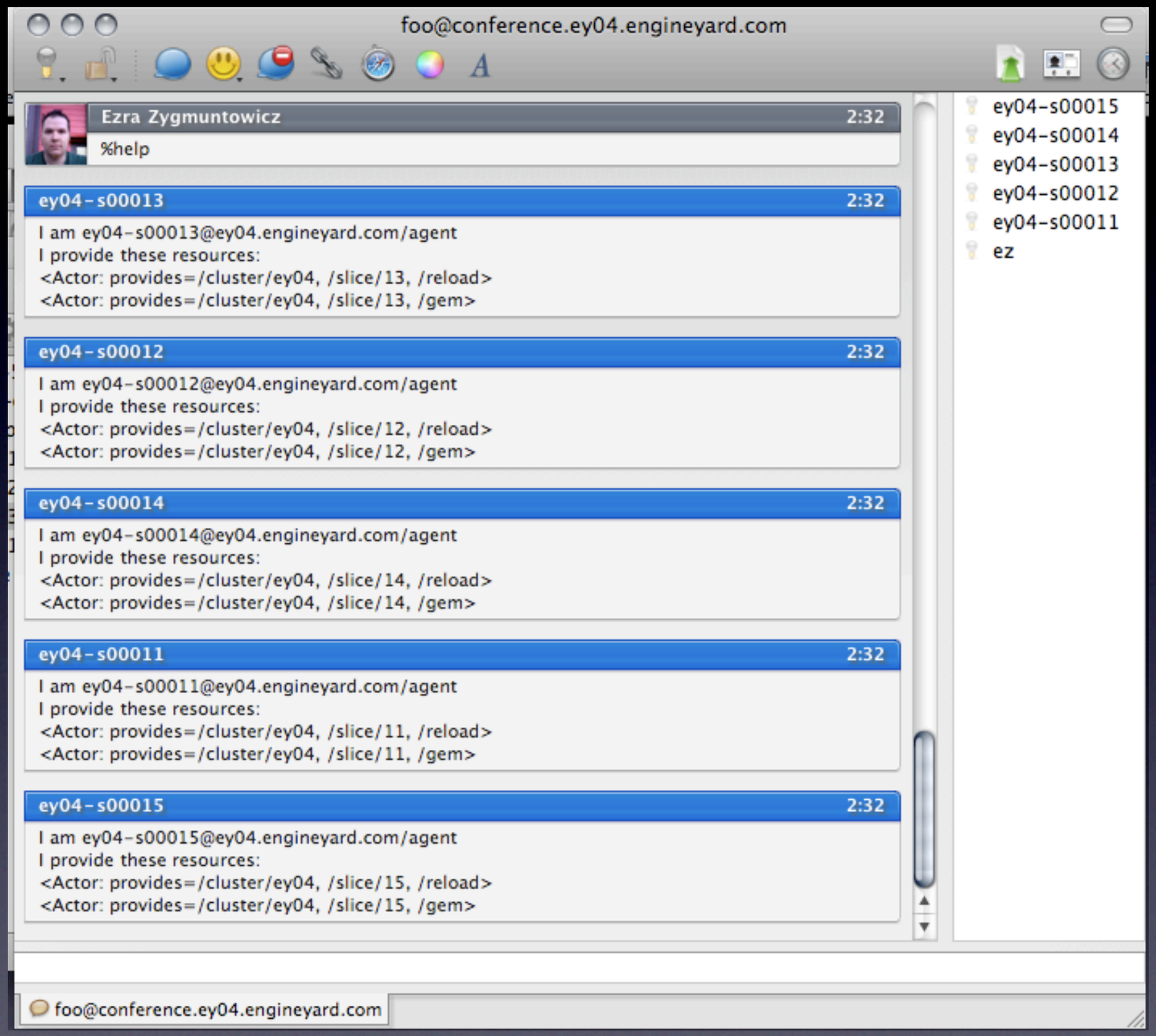
Workflow
coordination of multiple agents



Operations are like unix processes: hash as stdin/stdout

Workflows are like command line pipes tying together multiple operations

Integration with IM via XMPP



Any ‘state’ in a workflow can call out to meatware

If we have time...

Questions?