CloudCRV & Virtual Cluster Appliance

Deploy Your Cluster to the Cloud with 1-Click

Yushu Yao
Lawrence Berkeley National Laboratory

Image vs. Recipe

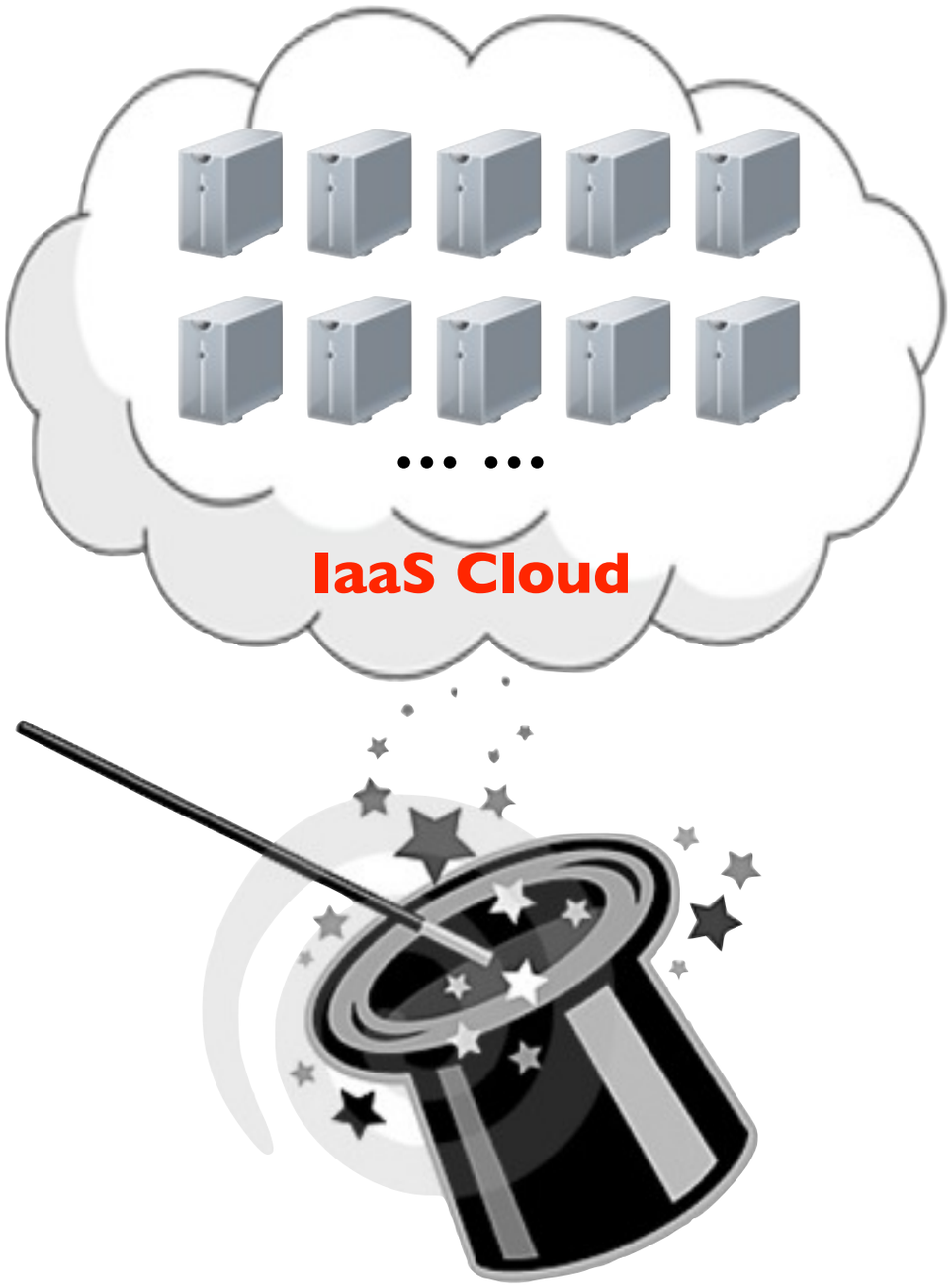
- Same direction as the HEPiX VM working group
  - Many work that applies to Images also works with Recipes
  - Recipes will make the process simpler and more flexible
- Sites are not doing extra work:
  - Most of the work sites have to do them anyways (authorizing, contexualization, auditing, etc)
  - Works to be done in each site are very similar
  - Recipe is a good way to share common security policies
- Performance of VM as Worker need to be studied carefully
  - Suggest investigating raw-hardware provisioning - instead of handing the image to a hypervisor, boot the image on a physical box (e.g. IPMI + PXE)

Overview

- We are building an Auto-Scaling ATLAS Cluster on the Cloud
- We made a Tool (+Concepts behind it) to achieve above, which is also a general purpose tool for Cloud Deployment
  - So why don't we share it with the community
- Many thanks to the Workshop Organizers

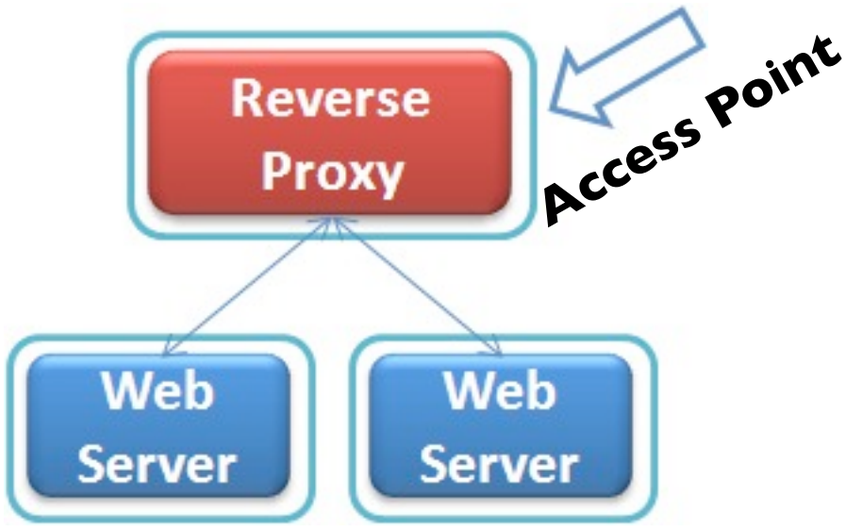
The Cloud is Good

- With the help of IaaS* cloud (Such as Amazon EC2), anyone can lease a large amount of Computing Resource
- E.g. get 100 linux computers
- The idea is: Any cloud user can setup their own cluster, and benefit from the scalability that the cloud provides.
- However, what do I do with the 100 computers?



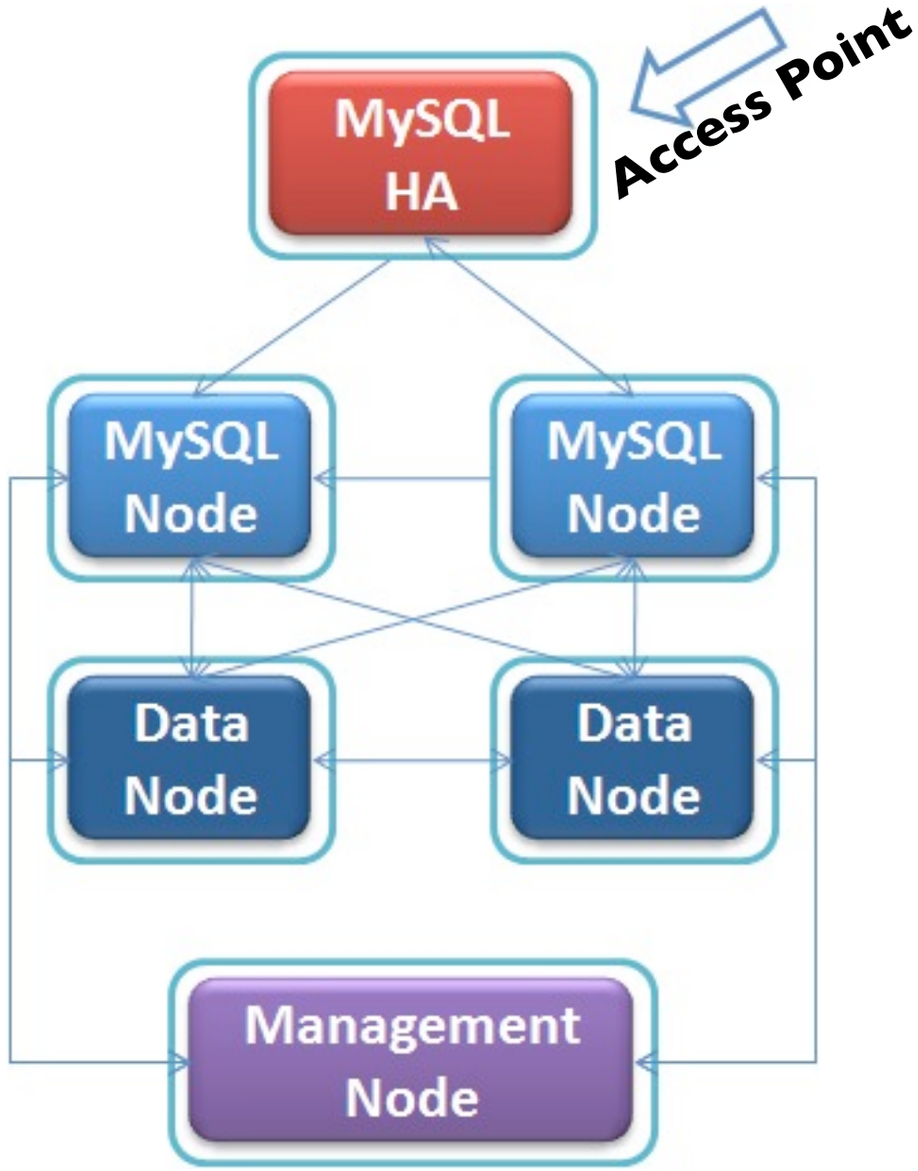
***IaaS**: Infrastructure as a Service, e.g EC2, Eucalyptus, Nimbus, Open Nebula?

Deploy This Cluster is NOT Easy



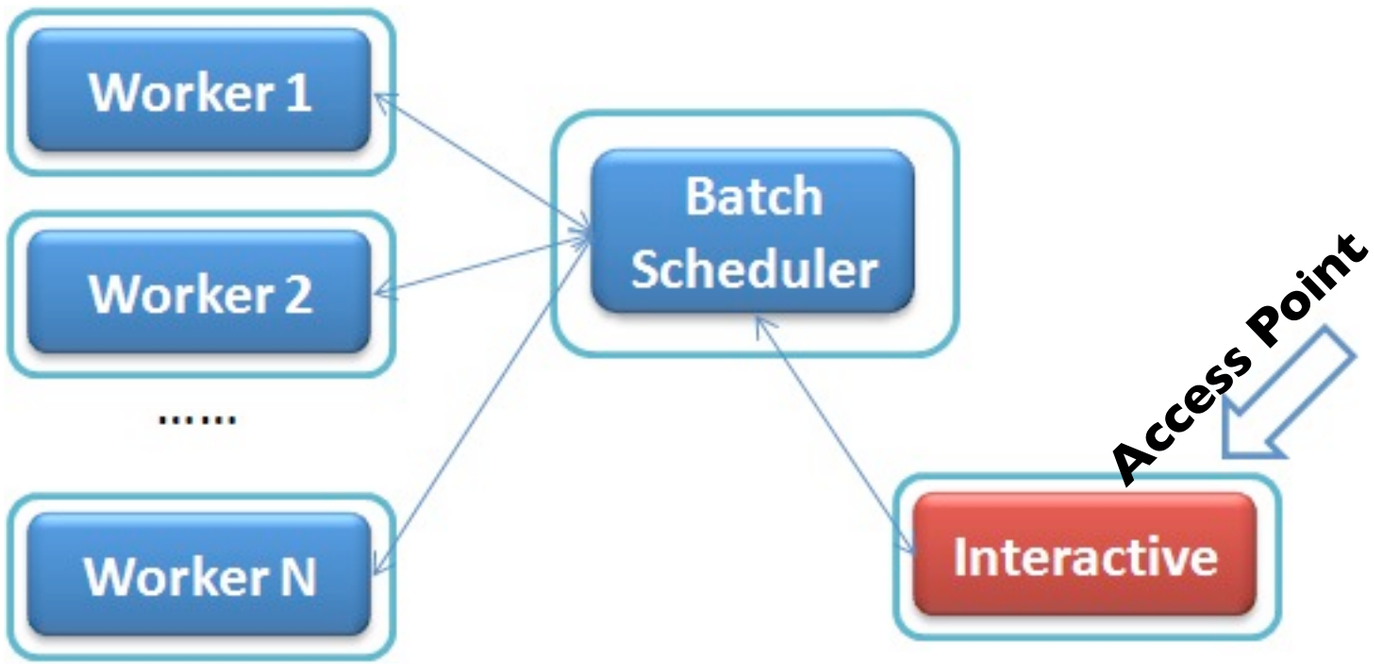
Load Balanced
Web Server Cluster

Deploy This Cluster is NOT Easy



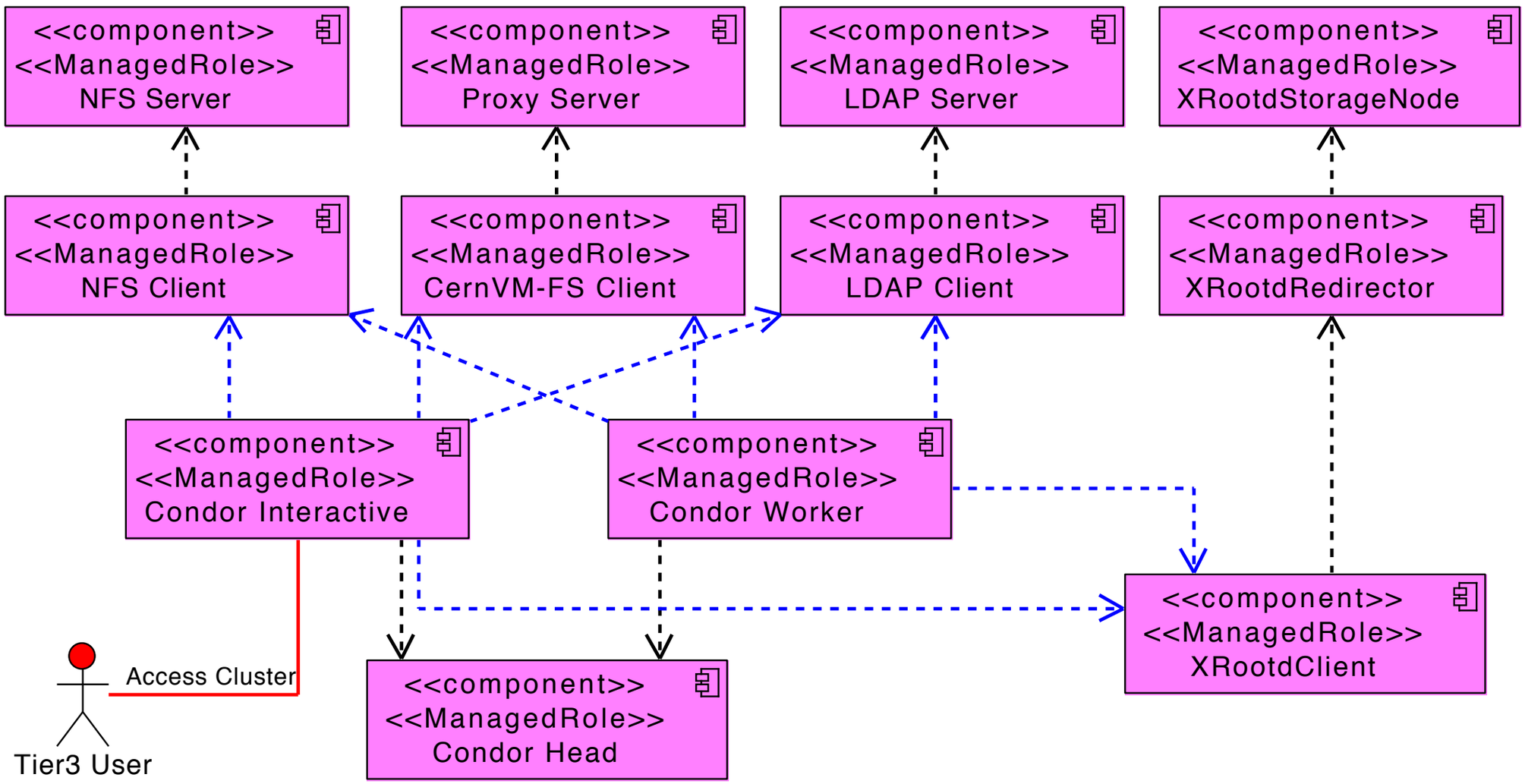
MySQL Cluster

Deploy This Cluster is NOT Easy



Auto-Scaling Batch Cluster

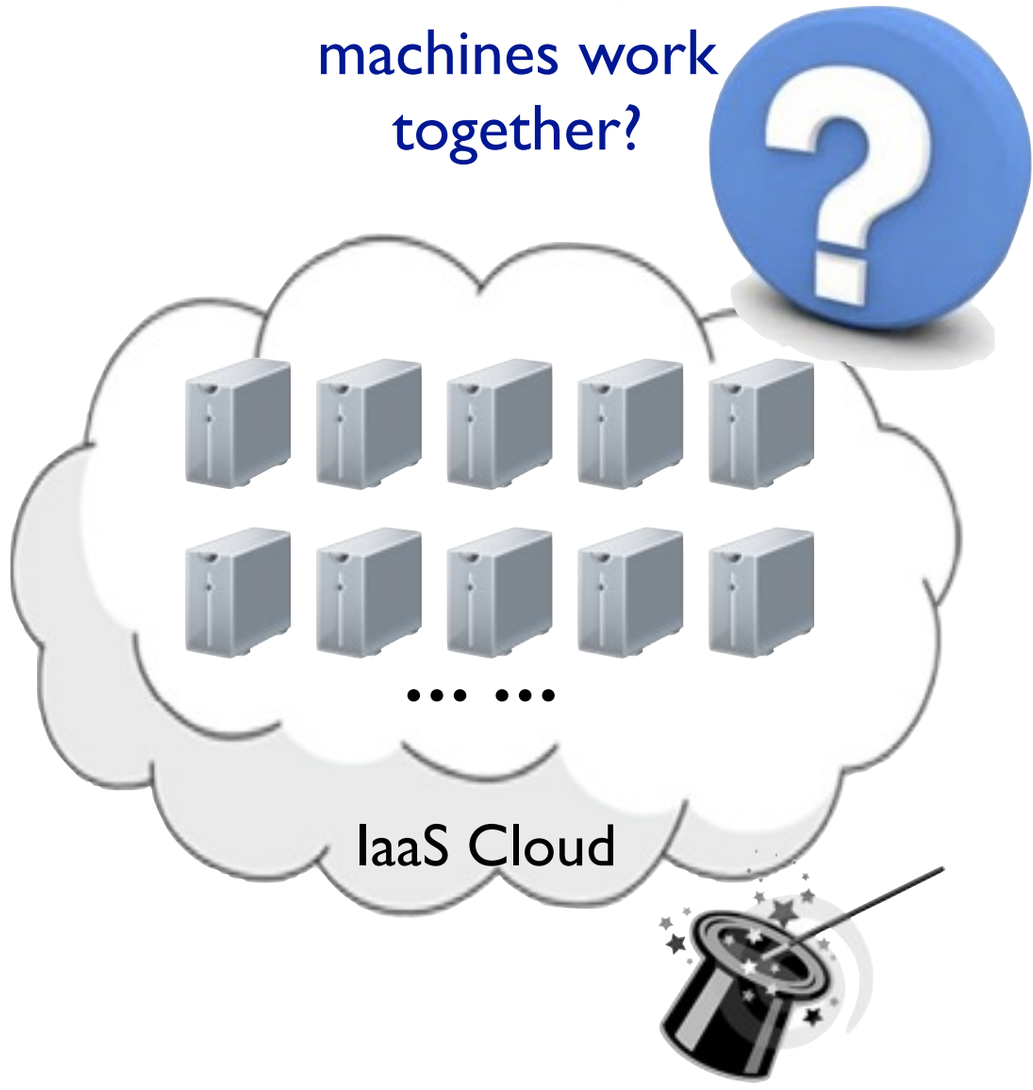
ATLAS Tier3 Cluster



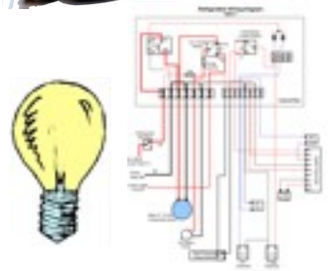
Really Good?

- I would like the machines to work as a complex cluster. How to configure them?
- I'm not an IT professional, I don't know how to do it
- Even if I know how to configure, it's too much work to login to 100 machines and do it by hand
- How can I scale the size of the cluster on-demand?

Yes, I can easily get 100 machines.
But how can the 100 machines work together?



An Analogy

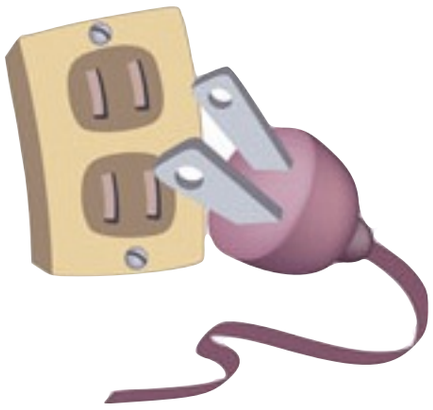


Parts in a Fridge

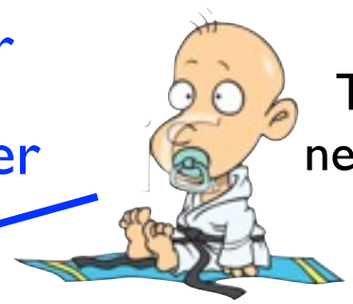
Manufacture

Fridge

Plug and Use



Substances	A Cluster	A Fridge
	Various of Apps and Services	Parts in a Fridge
	Computing Power from Cloud	Electricity
Actors	Cloud Provider	Electricity Provider
	Cluster Designer	Fridge Manufacturer
	Cluster Manager/Deployer	Fridge User
Actions	Design a Cluster	Manufacture a Fridge (Need expert)
	Deploy a Cluster	Plug the Fridge to the Wall (Simple)



The guy who needs to deploy a Cluster

Virtual Cluster Appliance (VCA)

BERKELEY LAB

Virtual Appliance

Pre-installed pre-configured OS and Application Stack That performs certain tasks

What if my tasks need more than one VM to perform?

Combine more than one services/ applications + Define Multiple Virtual Machines to hold these services/applications + How they work together

Virtual Cluster Appliance

Virtual Cluster Appliance = { **Multiple Virtual Machines** + **Multiple Services/Applications** + **Their Relationships**

Lifecycle of a VCA

✦ VCAs are Designed by Experts:

✦ Select base image and figure out how to Install and Configure all the software and services

✦ Define how the different VMs and services interact during deployment so that they can work together once deployed

✦ Can use ***any script (bash, puppet, cfengine, etc)*** to describe the above building process

✦ Virtual Cluster Appliances are Packaged into a set of Scripts and specifications in an SVN Repository

- ✦ Easy to ship/copy
- ✦ Versioning control makes production releases/bug fixes much easier.

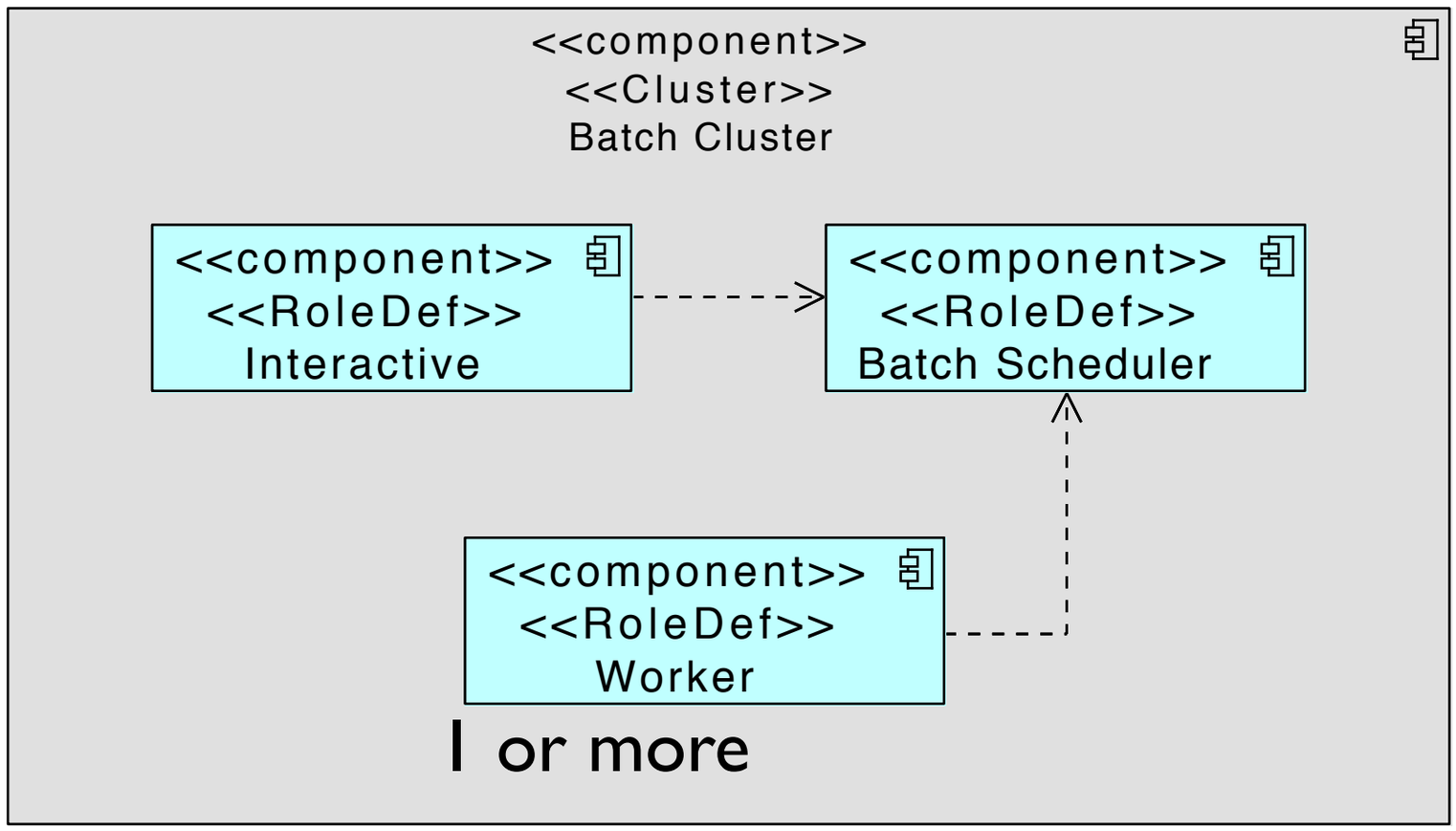
✦ To deploy a VCA, one need a tool to:

- ✦ Control the lifetime of the VMs inside the cluster
- ✦ Configure the VMs to work together

✦ **CloudCRV (Cluster-Roles-VMs) is a tool to do this**

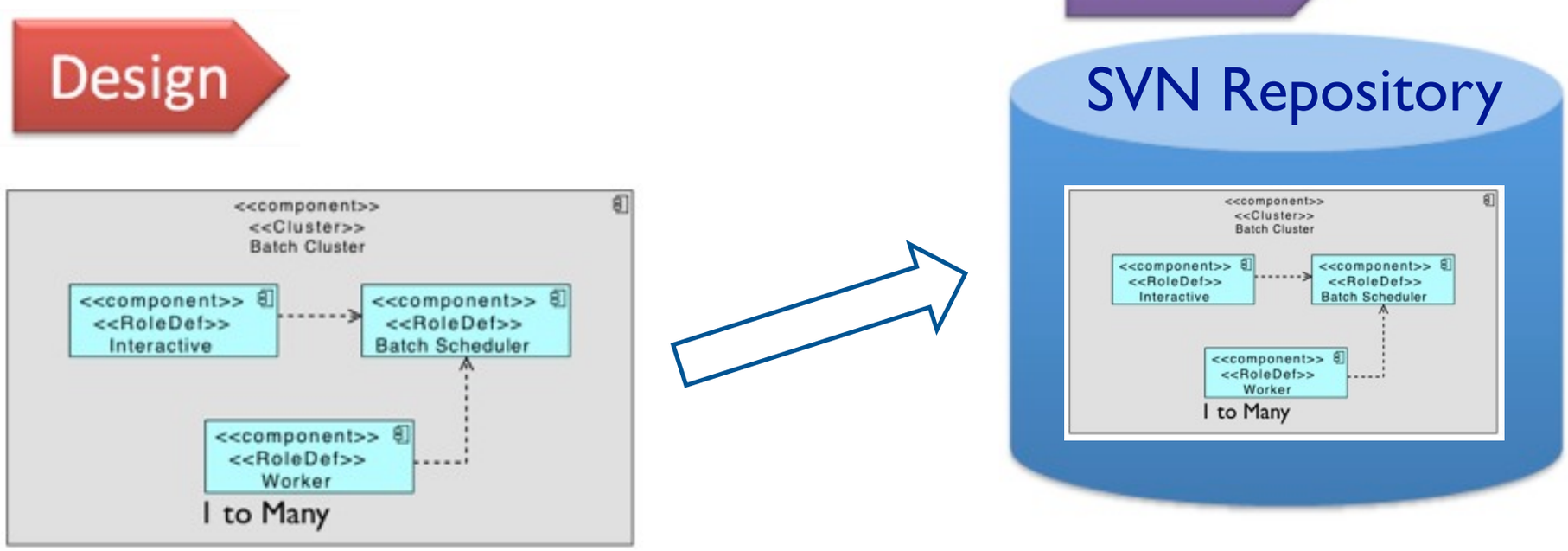
Example - The Batch System VCA

Design



- Each RoleDef represents certain set of functionalities
- RoleDefs are interdependent, e.g. Worker depend on Scheduler
- Designing this cluster is the job of a Cluster Designer

Design to Distribute



- Puppet and Python Scripts are used to describe the individual RoleDefs and how they work together
- All scripts are put into an SVN Repository

Deploying a VCA

Deploy

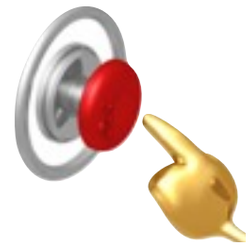
Cluster Manager

Start Up CloudCRV

CloudCRV

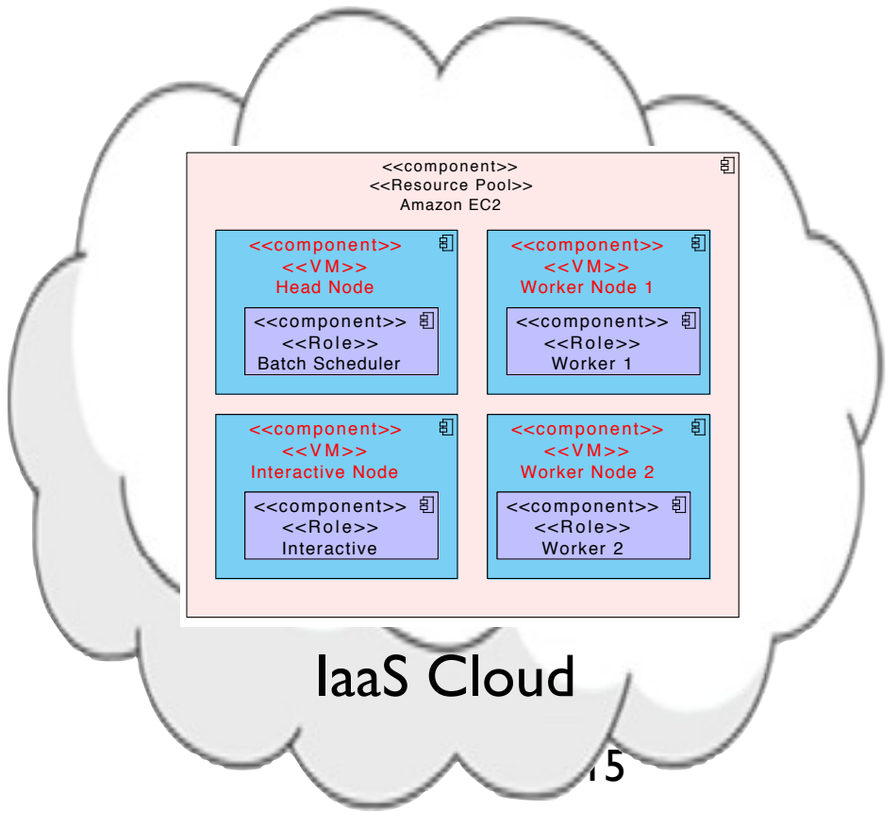
Specify a Virtual Cluster Appliance

Set basic Parameters Eg. How many workers?

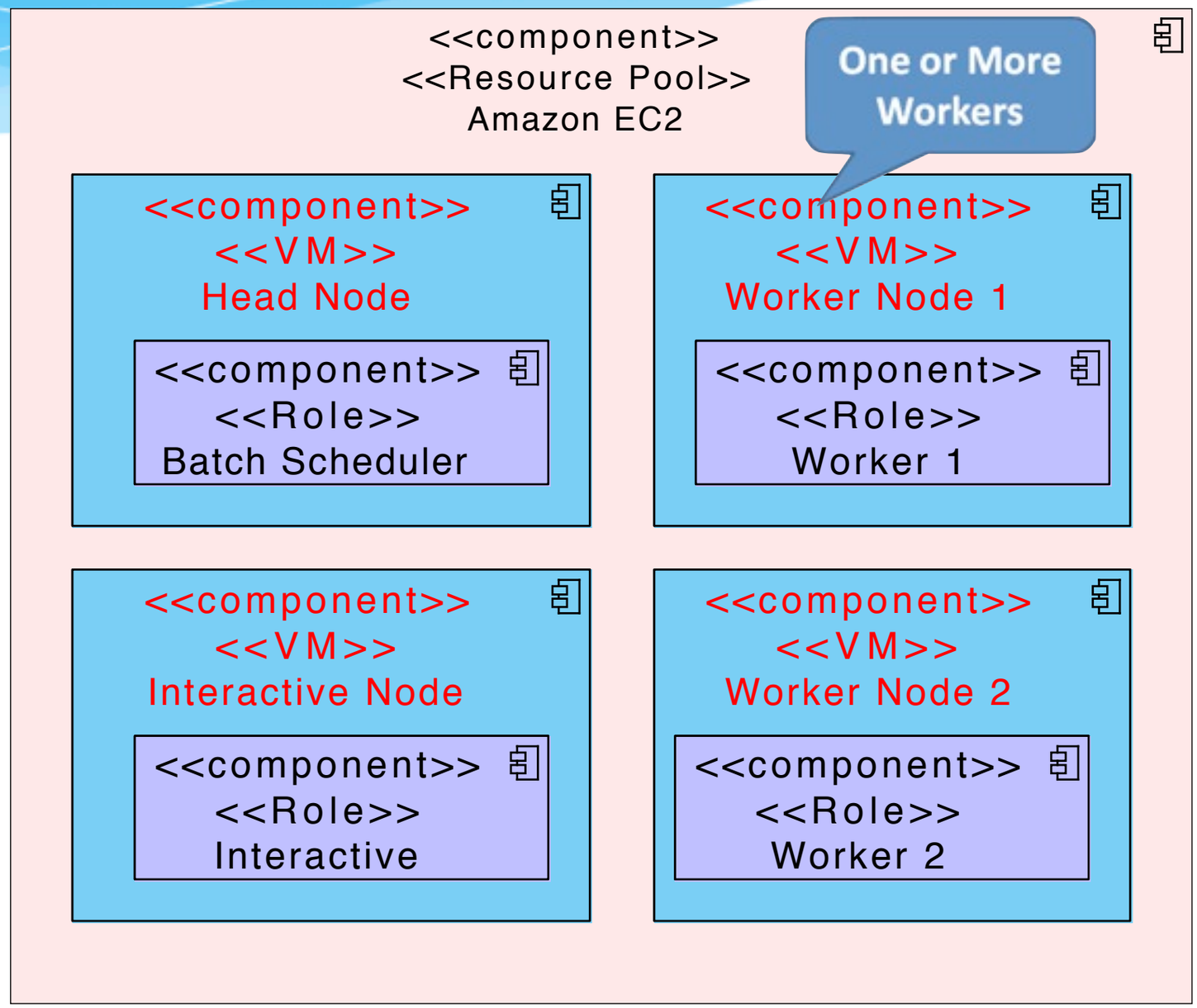


Click the **Deploy** Button

- Allocate Resources
- Configure Services

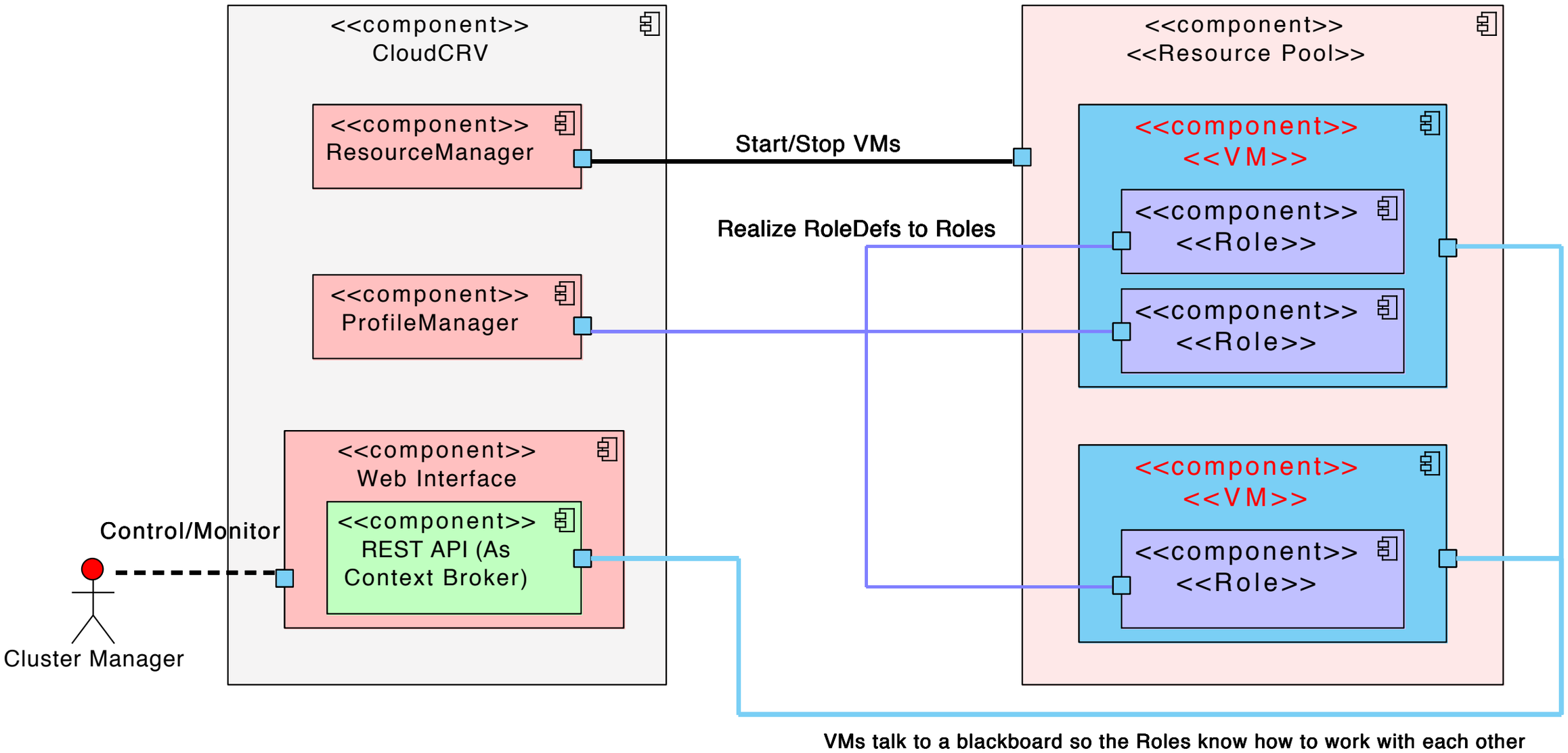


The Deployed Batch Cluster VCA



- We define 1 resource pool - Amazon EC2
- Each RoleDef is realized to one or more Roles that sit on multiple VMs
- The CloudCRV and Puppet will configure all the Roles to work together

Components of CloudCRV



Roles Interact with each other via broker

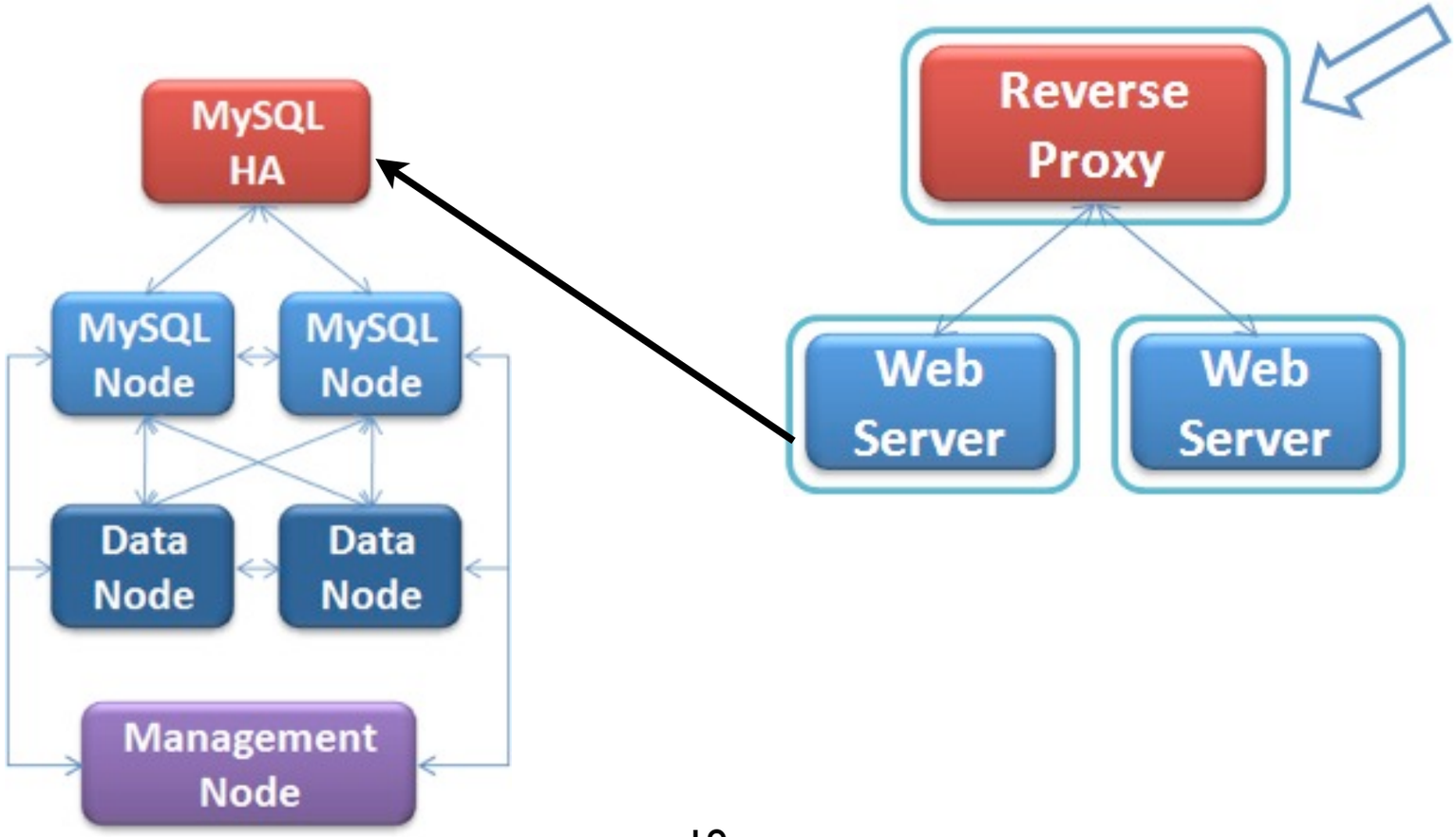
- **ResourceManager**: Allocate/Deallocate resource in the resource pools (Turn on/off VMs)
- **ProfileManager**: Realize a RoleDef to a Role that's sitting on a VM
- **Web Interface**:
  - Let the Cluster Manager Control the deployment of the Cluster and Monitor the status
  - Provide a media for VMs (Roles) to talk with each other so that the all VMs/Roles can work together as a cluster

Not only on IaaS Cloud

- We showed an example on IaaS
- CloudCRV can not only deploy to EC2 like clouds. It also support other resource pools, e.g. libvirt based VMs, raw hardware via gPXE
- To add support for a new "resource pool", simply extend the interface and implement 3 functions.

Reusability

- VCAs are defined as module-based scripts
  - Designers can easily reuse components in other VCAs or combine several VCA to a new one
  - Much faster development cycle
- E.g. Adding High Availability MySQL to Web Server cluster is simple



Security

- SVN Repository/Scripts can prevent unauthorized use
- CloudCRV manages its own CA, all web service are HTTPS based
  - Manger UI enforce user/pass authentication
  - REST API enforce PKI based authentication - each VM has its own Cert.
- Puppet network traffic is encrypted and authenticated with PKI, using the Certs provided by CloudCRV

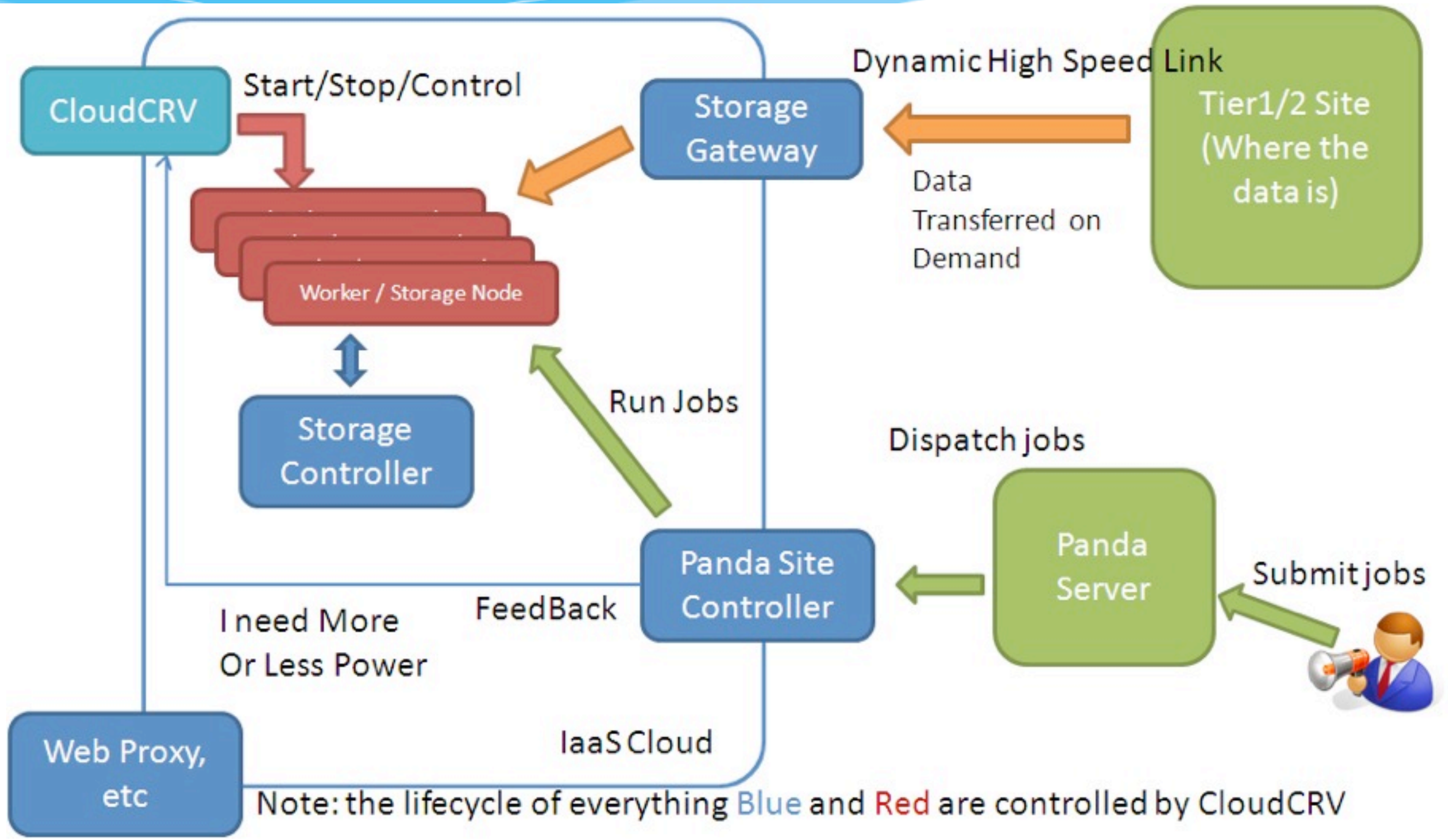
Summary

- IaaS Cloud only gives you computers, but not functional clusters.
  - A Virtual Cluster Appliance is an expert-designed cluster that can be easily deployed by non-expert users
- CloudCRV deploys a Virtual Cluster Appliance to IaaS Cloud (and other platforms)
- More Info:
  - Project: http://code.google.com/p/cloudcrv/
  - Poster: http://indico.cern.ch/getFile.py/access?contribId=26&sessionId=5&resId=0&materialId=poster&confId=92498

Want to try it?

- For early adopters, give us your requirements, we can provide you a VCA to start with.

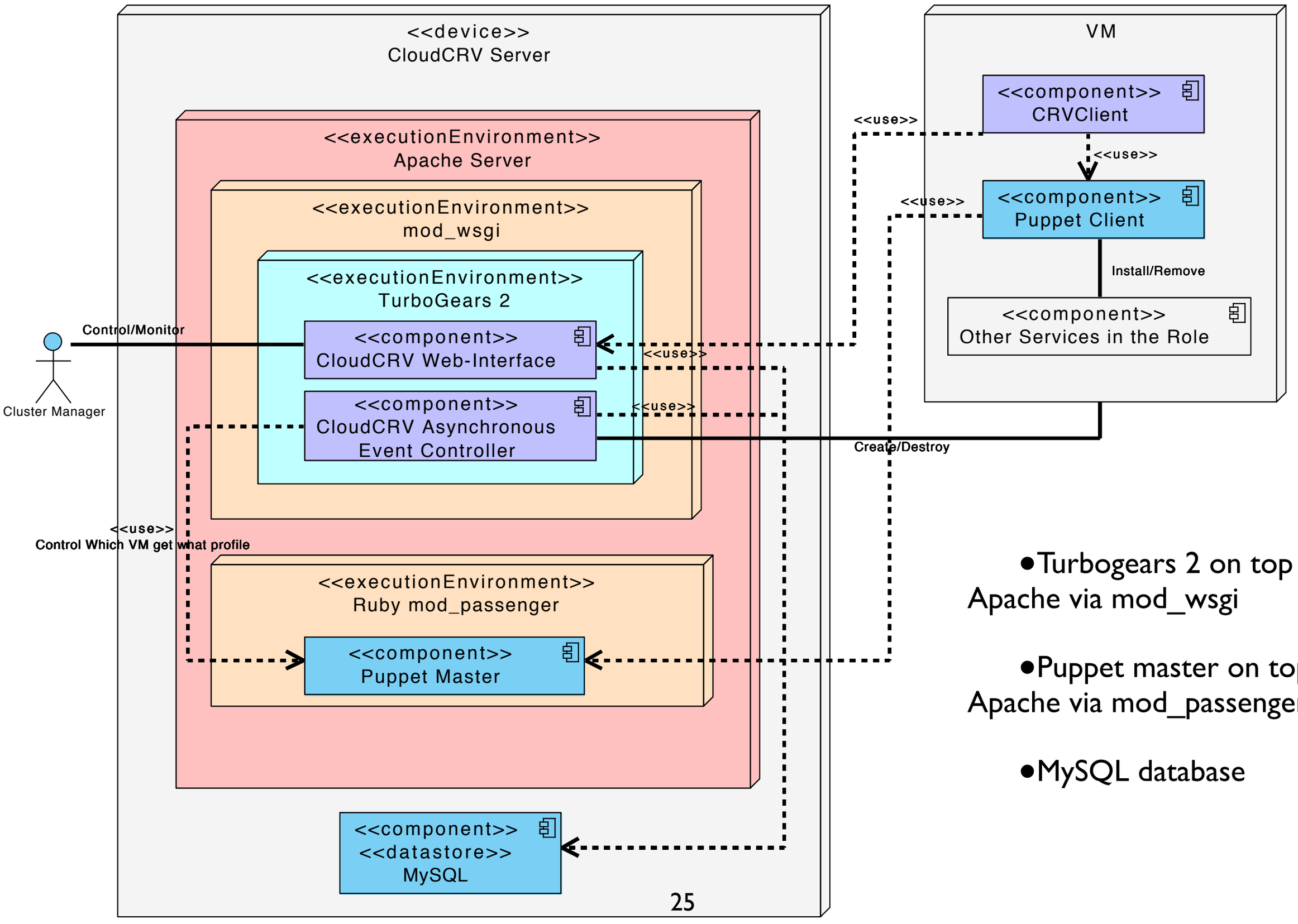
Auto-Scale ATLAS Cluster on Magellan



- Scale on-demand
- Storage on Worker (HDFS)
- Panda Based
- High Speed Data Link

Backup

The CloudCRV Server and Clients



- Turbogears 2 on top of Apache via mod_wsgi
- Puppet master on top of Apache via mod_passenger
- MySQL database

Puppet

- Configuration Management system
- Use a language to describe a service:

```
class ssh {
    package { ssh: ensure => installed }
    file { sshd_config:
        name => "/etc/ssh/sshd_config",
        owner => root,
        group => root,
        source => "puppet://server/apps/ssh/sshd_config",
        after => Package[ssh]
    }
    service { sshd:
        ensure => running,
        subscribe => [Package[ssh], File[sshd_config]]
    }
}
```

How to Design a Cluster

- 1. Know how to setup individual services
- 2. Write a script for each role (preferable Puppet scripts)
- 3. One Python script to define the relations of the roles/vms in a cluster
- 4. Put everything to an SVN repository

Define a Role with Puppet Script

Ganglia Server
(gmetad, web-front)

```
define at3_ganglia_srv($webaddr="", $roleid="") {
   #Get some attributes from its dependents
   $clientlist=get_dependent_attrlist($webaddr,$roleid,"GangliaClientAddr")
   #Install necessary RPMs
   package { ["ganglia", "ganglia-web", "ganglia-gmetad","httpd"]:
     ensure => installed,    }
   #Create configuration file
   file { "/etc/gmetad.conf": ... ...  notify => Service["gmetad"],    }
   #Start Service
   service { ["gmetad", "httpd"]:  ensure => running,  enable => true,  ... ...
     require => [Package ["httpd", "ganglia", "ganglia-web", "ganglia-gmetad"],File["/etc/
gmetad.conf"]]   } }
   #Publish its IP so that its dependents know where the server is
   set_role_attr($webaddr,$roleid,"GangliaSrvAddr","$ipaddress")
}

define at3_ganglia_daemon($webaddr="", $roleid="") {
   #Figure out where the Ganglia Server is
   $gangliasrvaddr=get_provider_attr($webaddr,$roleid,"GangliaSrvAddr")
   #Install RPMs
   package { ["ganglia", "ganglia-gmond"]:     ensure => installed  }
   #Config files
   file { "/etc/gmond.conf":... ...      notify => Service["gmond"],   }
   #Start Service
   service { [ "gmond"]: ensure => running, enable => true, ... ...
     require => [Package["ganglia", "ganglia-gmond"], File["/etc/gmond.conf"]] } }
   #Publish its IP so that server can authenticate it
   set_role_attr($webaddr,$roleid,"GangliaClientAddr","$ipaddress")
}
```

Ganglia Client
(gmond)

Python Part

BERKELEY LAB

```
#Create Resource Pool
magellan_euca=EucaVMMaker("magellan_euca")
cpc=CentOS5_Puppet_ClientMaker(name="RHEL5_Puppet",doc="CRVClient for RHEL5 Clients using
    Puppet",templatedir="/usr/local/cloudcrv/cloudcrv/crvclient")
ppm=Puppet_ProfileMaker(name="Puppet",doc="Profile Maker using Puppet")
rp=RP("publicIP","Resource Pool with Publicly Addressed VMs, Puppet Profiles and RHEL5
Clients",profilemaker=ppm,vmmaker=acs_euca,clientmaker=cpc)

#Role Definitions
rdGangliaSrv=RoleDef('GangliaSrv','Ganglia Server (MetaDaemon and WebFront)','at3_ganglia_srv',"puppet")
rdGangliaClient=RoleDef('GangliaClient','Ganglia Client (Monitoring Daemon)','at3_ganglia_daemon',"puppet")

#remote dependency
rdGangliaClient.addDependOn(rdGangliaSrv)

rdProxy=RoleDef('Proxy',"Proxy Server","at3_proxy","puppet")

#Local dependency
rdProxy.addDependOn(rdGangliaClient,local=True)

#the cluster
cl=Cluster("tier3")

#roles
rGangliaSrv=cl.addRole(name="rGangliaSrv", roledef=rdGangliaSrv, vm=rp.newVM("vmGangliaSrv"), enabled=True)
rProxy=cl.addRole(name="rProxy", roledef=rdProxy, vm=rpprivate.newVM("vmProxy"), enabled=True)
... ...
```