Elastic Architecture in CloudFoundry and Deploy with OpenStack

Layne Peng
@Layne_Peng
layne.peng@emc.com

Kay Yan
@yankay
kay.yan@emc.com

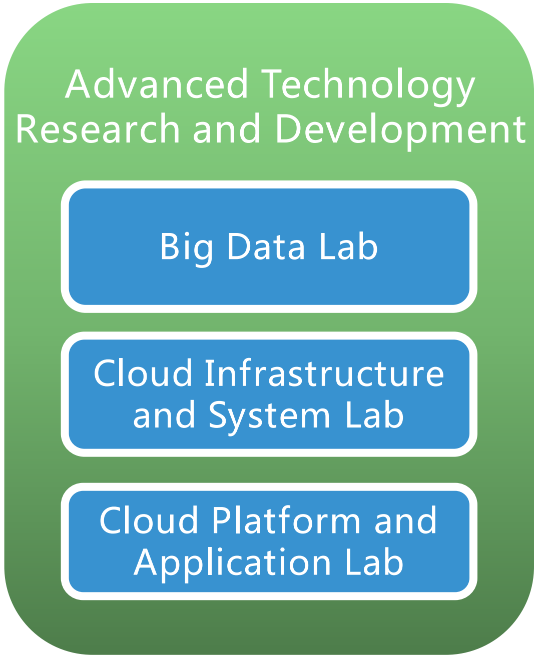
Cloud Platform and Application Lab, EMC Labs China

About Us

Technologist from Cloud Platform and Application, EMC Labs China

- Our work:
  - Research topics related to cloud architecture
- Lab focus areas:
  - PaaS/IaaS
  - NGDC automation management
  - Cloud platform middleware
  - Multi-tenant management

EMC Labs China



Advanced Technology Research and Development

Big Data Lab

Cloud Infrastructure and System Lab

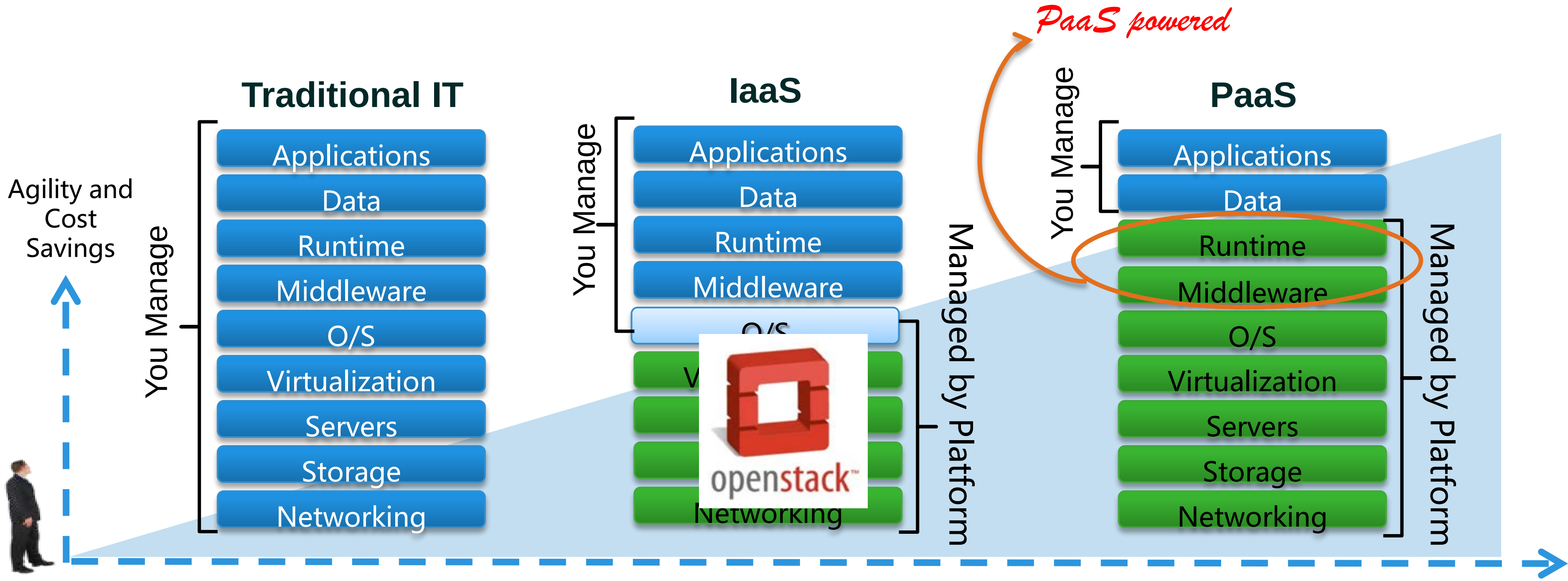
Cloud Platform and Application Lab

Agenda

- Why We Here?
- First Touch CloudFoundry
- Elastic Architecture in CloudFoundry
- Introduce to BOSH
- CPI and OpenStack
- Deploy PaaS with BOSH

Why We Here?

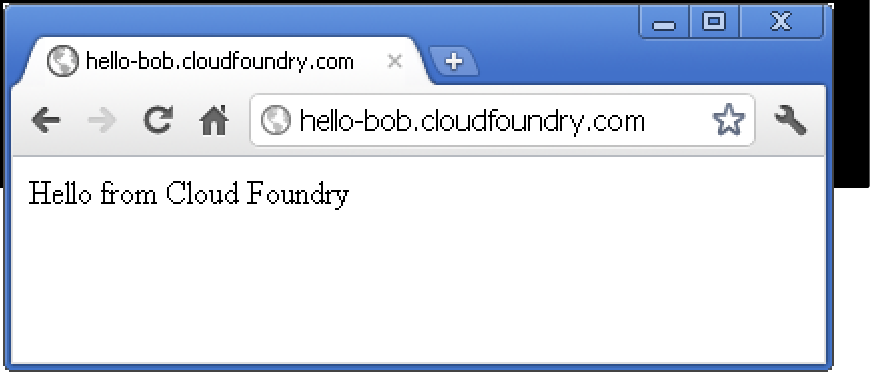
From *Accelerating your Journey to Application Transformation,* EMC World 2012



EMC²

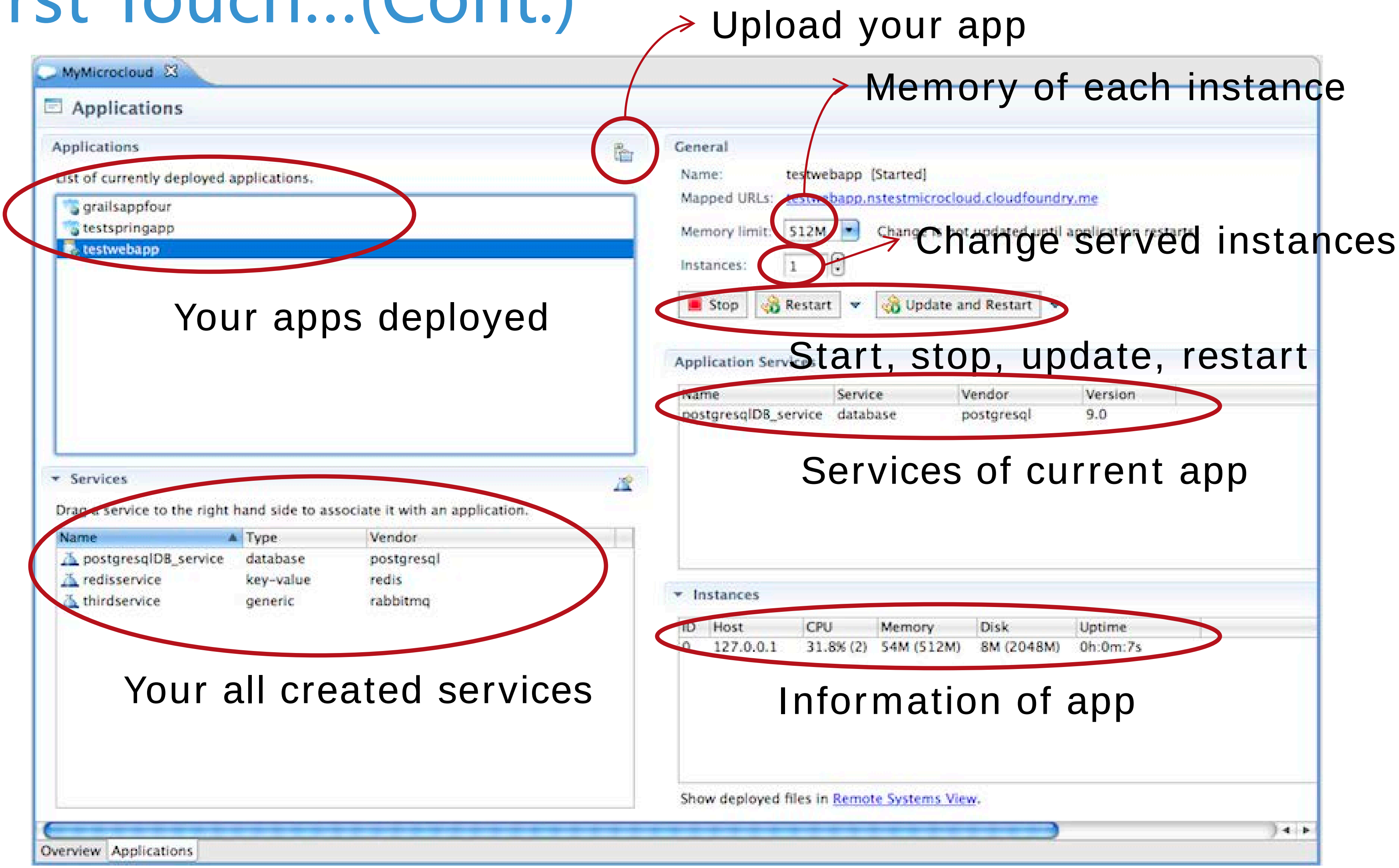
First Touch…

```
prompt> gem install vmc
prompt> vmc target api.cloudfoundry.com
prompt> vmc login
prompt> vmc push

Would you like to deploy from the current directory? [Yn] Yes
   Application Name: hello
   Application Deployed URL: 'hello.cloudfoundry.com'?  hello-bob.cloudfoundry.com
   Detected a Sinatra Application, is this correct? [Yn]  Yes
   Memory Reservation [Default:128M]  (64M, 128M, 256M, 512M or 1G) (Press Enter to take
default)
   Would you like to bind any services to 'hello'? [yN]: No
   Uploading Application:
       Checking for available resources: OK
       Packing application: OK
   Uploading (0K): OK
   Push Status: OK
   Staging Application: OK
   Starting Application: OK
```



Hello from Cloud Foundry

First Touch...(Cont.)



MyMicrocloud

Applications

Applications

List of currently deployed applications.

- grailsappfour
- testspringapp
- testwebapp

Services

Drag a service to the right hand side to associate it with an application.

Name	Type	Vendor
postgresqlDB_service	database	postgresql
redisservice	key-value	redis
thirdservice	generic	rabbitmq

General

Name: testwebapp [Started]

Mapped URLs: testwebapp.nstestmicrocloud.cloudfoundry.me

Memory limit: 512M Changes not updated until application restarts

Instances: 1

Stop | Restart | Update and Restart

Application Services

Name	Service	Vendor	Version
postgresqlDB_service	database	postgresql	9.0

Instances

ID	Host	CPU	Memory	Disk	Uptime
0	127.0.0.1	31.8% (2)	54M (512M)	8M (2048M)	0h:0m:7s

Show deployed files in Remote Systems View.

Overview | Applications

Upload your app

Memory of each instance

Change served instances

Your apps deployed

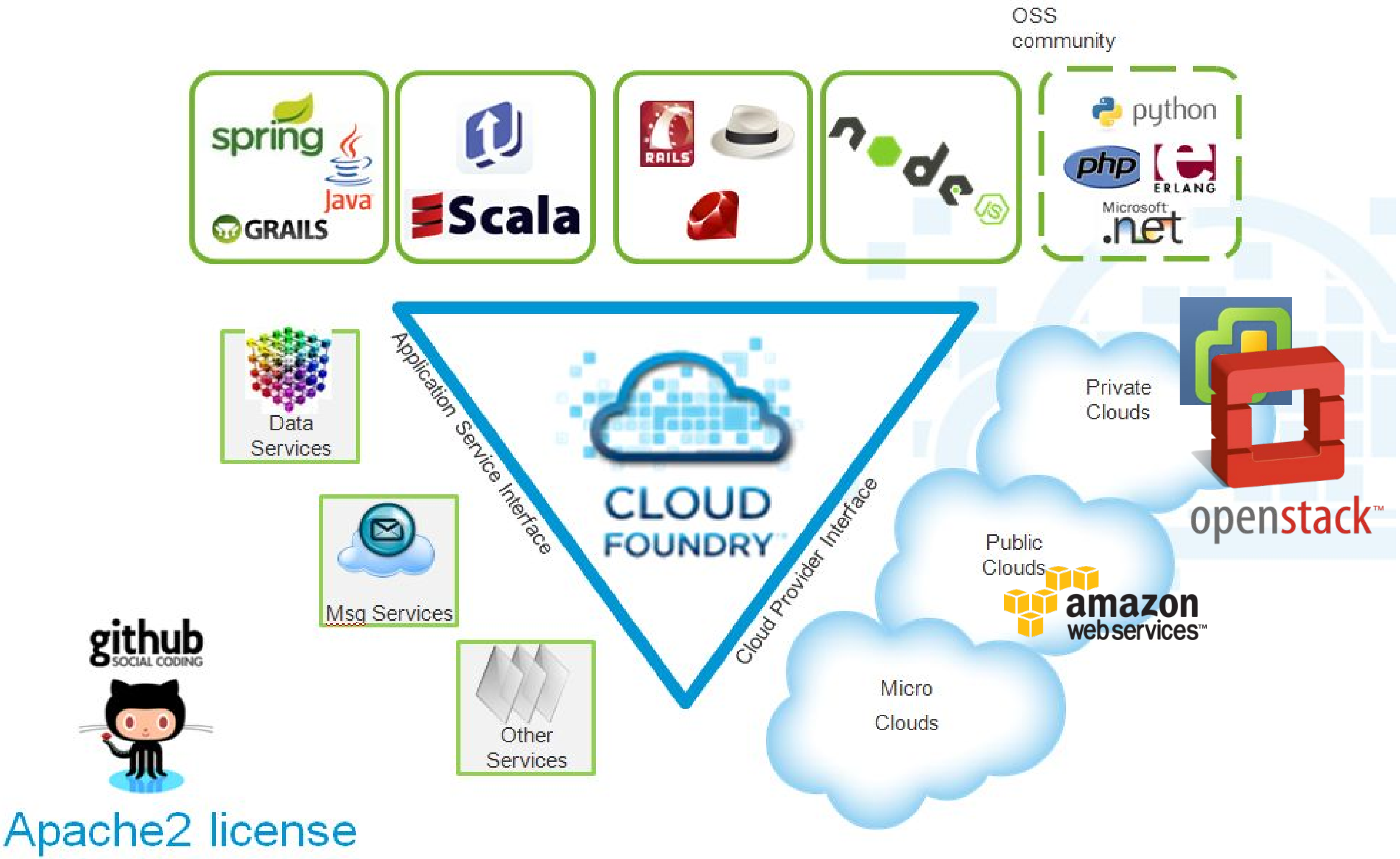
Start, stop, update, restart

Services of current app

Your all created services

Information of app

What CloudFoundry Offer?

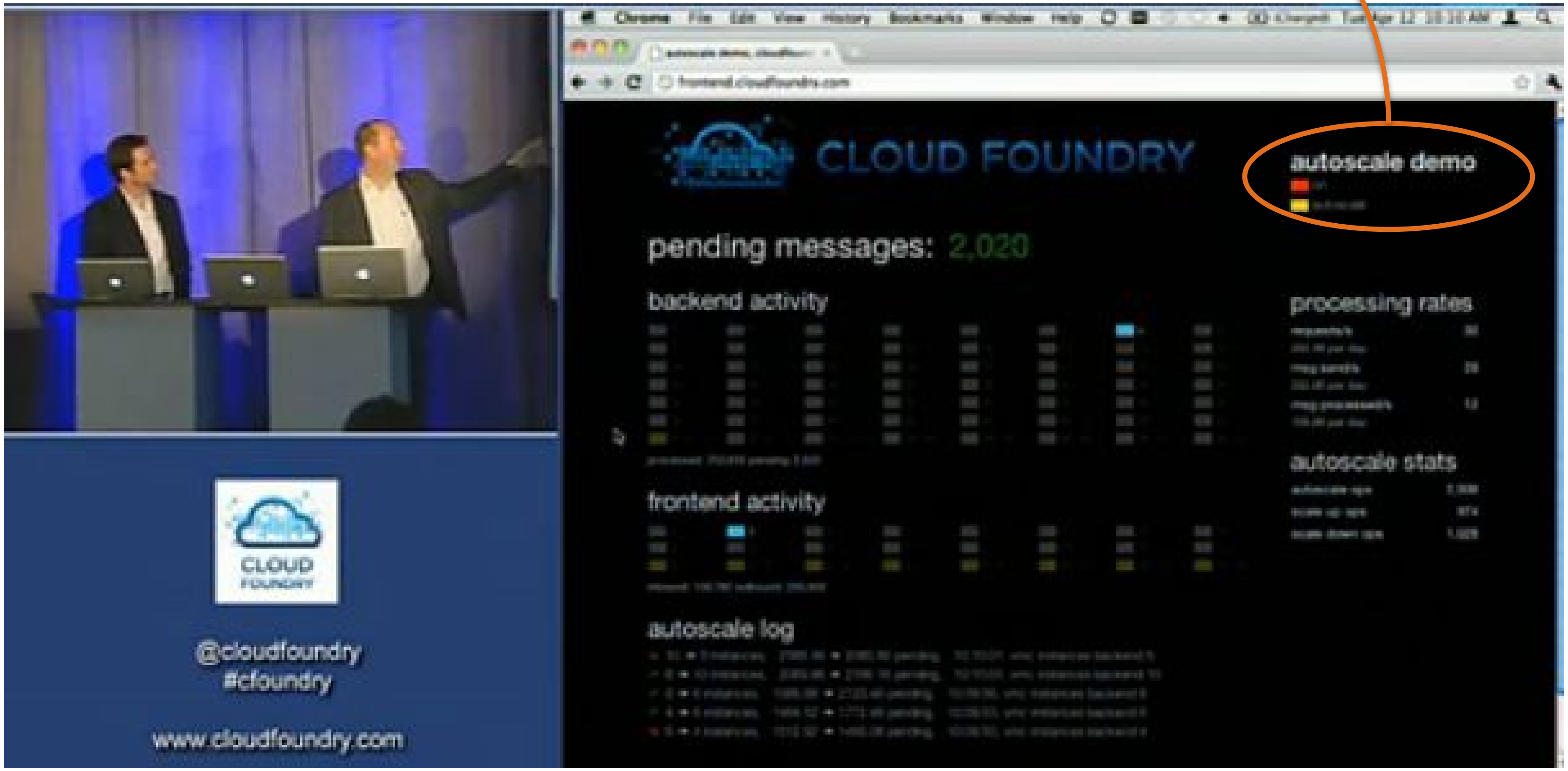


Apache2 license

What CloudFoundry Offer? (Cont.)

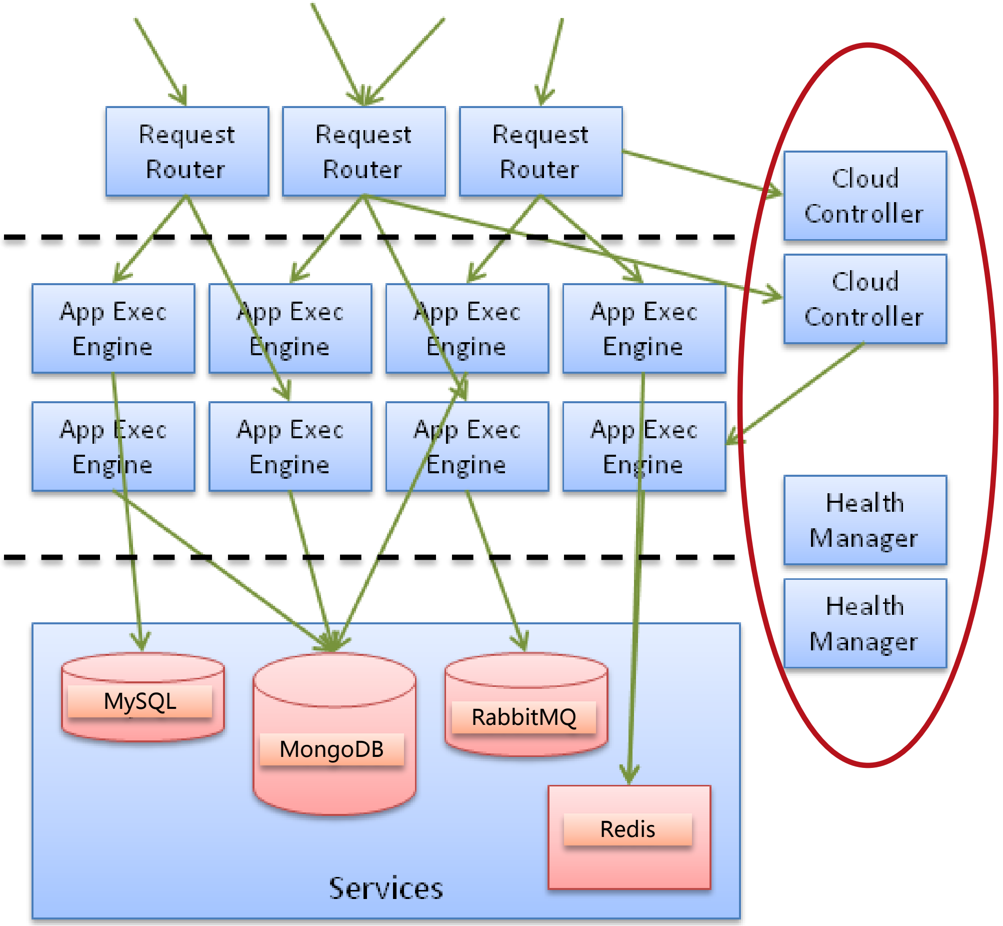
From *Cloud Foundry Launch Event, April, 2011*

PaaS powered



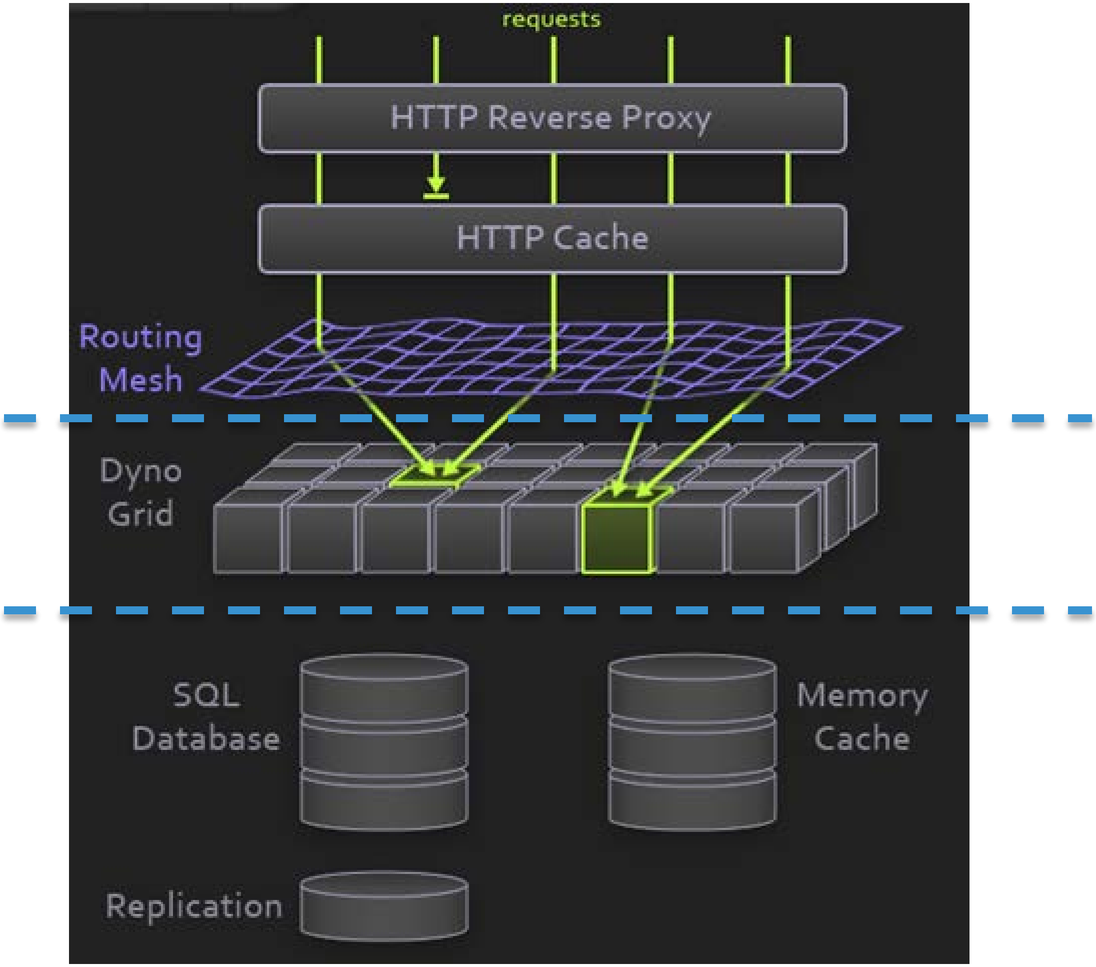
PaaS Architecture Pattern

CLOUD FOUNDRY™



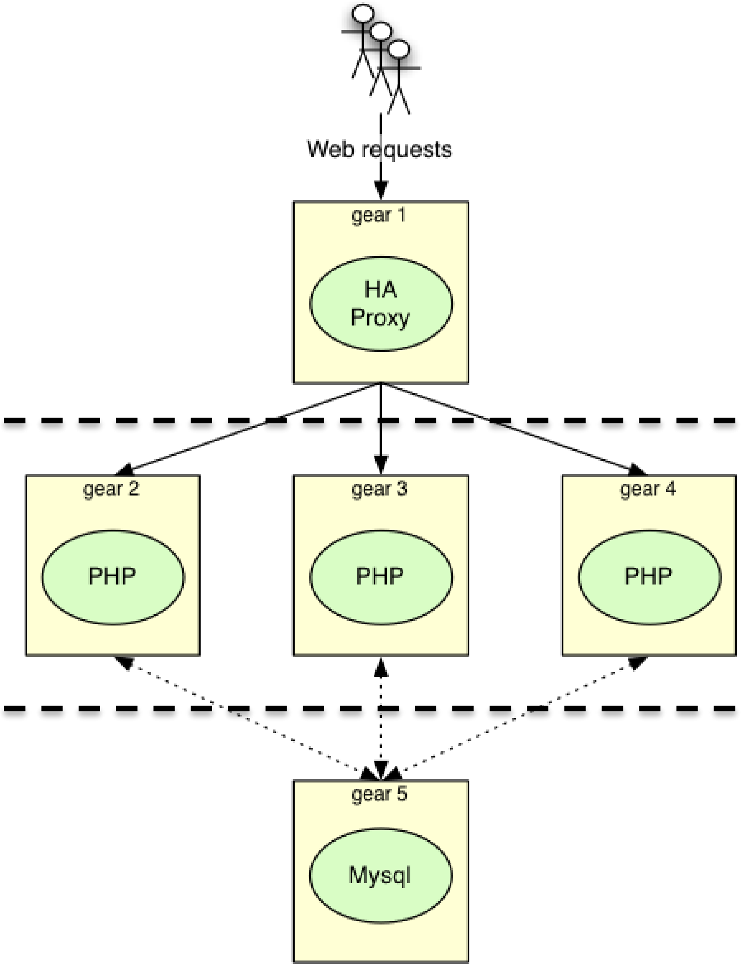
PaaS Architecture Pattern (Cont.)

heroku



PaaS Architecture Pattern (Cont.)

OPENSHIFT



Conclusion

Simplify to three layers:

- Routers for finding right endpoint of Apps
- Nodes of runtime for Apps
- Nodes of services provided by platform, consumed by Apps

The Keys of Design…

- Failover/System Robust
- Scalable
- Resource Recycling

} = Elastic

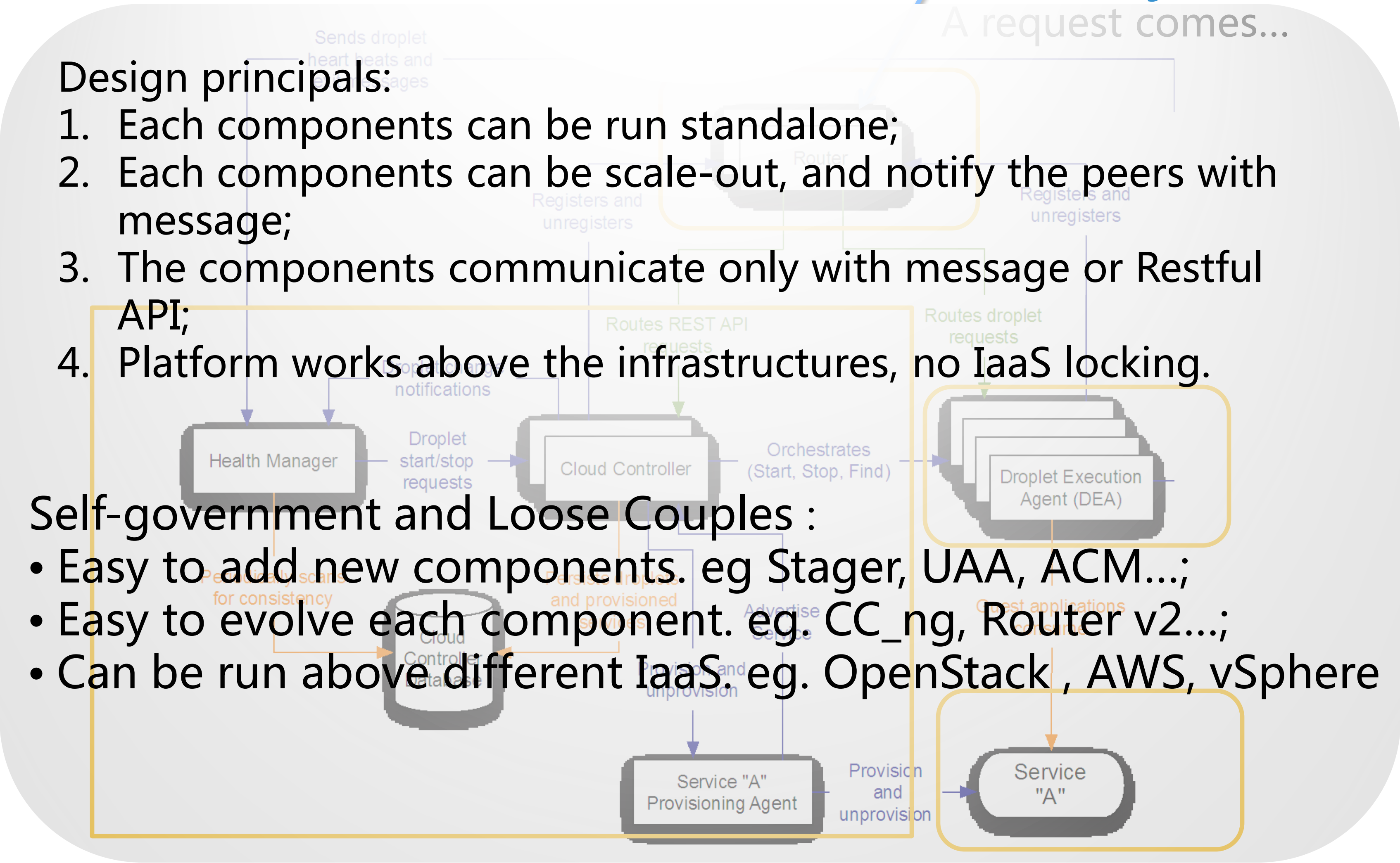
Elastic Architecture in CloudFoundry

Design principals:

1. Each components can be run standalone;
2. Each components can be scale-out, and notify the peers with message;
3. The components communicate only with message or Restful API;
4. Platform works above the infrastructures, no IaaS locking.

Self-government and Loose Couples :

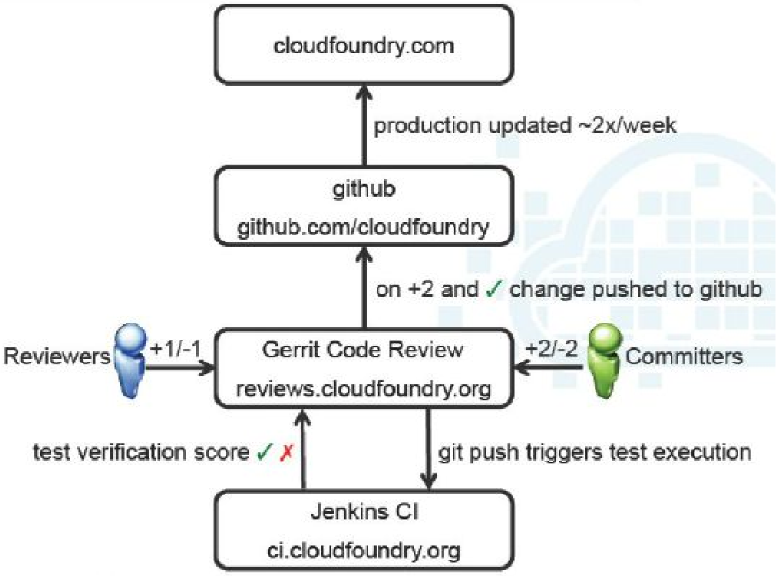
- Easy to add new components. eg Stager, UAA, ACM…;
- Easy to evolve each component. eg. CC_ng, Router v2…;
- Can be run above different IaaS. eg. OpenStack , AWS, vSphere



A request comes…
Sends droplet heart beats and messages
Router
Registers and unregisters
Registers and unregisters
Routes REST API requests
Routes droplet requests
notifications
Health Manager
Droplet start/stop requests
Cloud Controller
Orchestrates (Start, Stop, Find)
Droplet Execution Agent (DEA)
Periodically scans for consistency
Persists droplets and provisioned services
Advertise Service
Guest applications
Cloud Controller Database
Provision and unprovision
Service "A" Provisioning Agent
Provision and unprovision
Service "A"

Open Ecosystem

- Open Dev Proc



github

Apache2 license

- Partners & Communities

ActiveState
Stackato Python Support

PHP Support

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

EMC²

Joyent
node.JS Support

.net Support

Piston CLOUD COMPUTING

新浪云计算

Public PaaS
Ruby Cloud Foundry on SAE

盛大云 Grand Cloud

Public PaaS
Cloud Foundry on Grand Cloud

INGEEK

Public PaaS
Cloud Foundry & Data Director on vSphere/vCloud

中国软件与技术服务股份有限公司

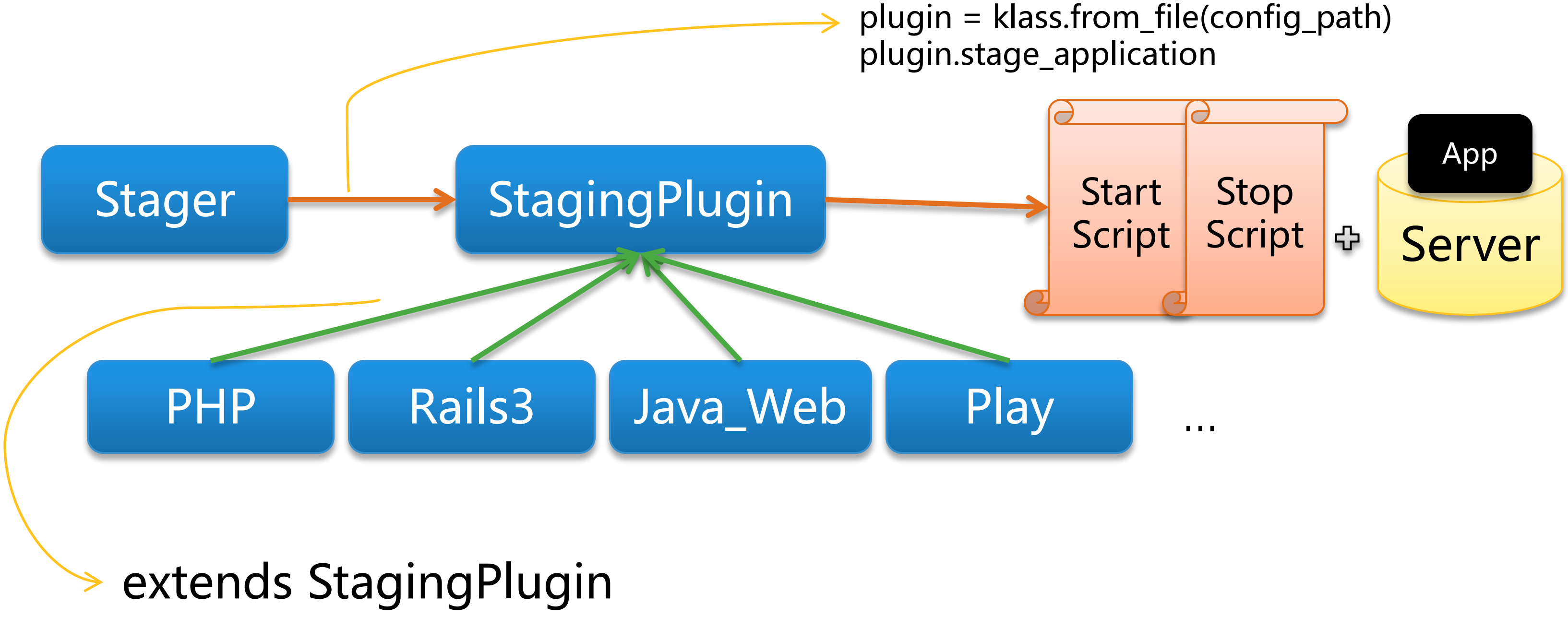
Public/Private/Hybrid PaaS/IaaS
Cloud Foundry on vSphere/vCloud

EMC²

Elastic Runtime Support

Refers to https://github.com/cloudfoundry/vcap-staging

- stager -> vcap-staging



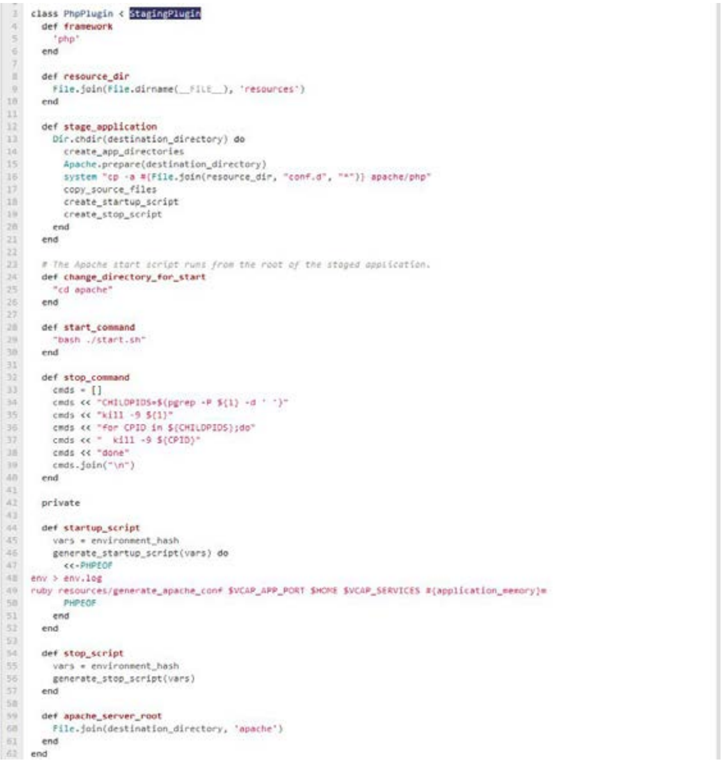
```
klass  =
StagingPlugin.load_plugin_for(plugin_name)
plugin = klass.from_file(config_path)
plugin.stage_application
```

extends StagingPlugin

Elastic Runtime Support (Cont.)

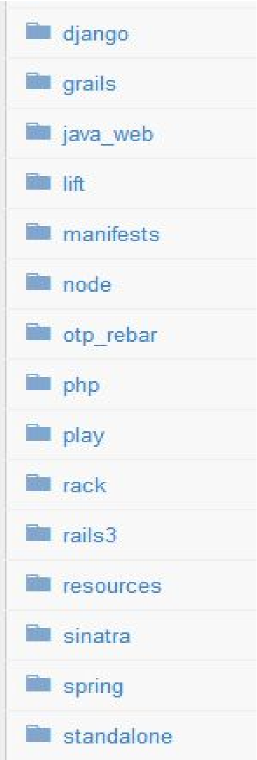
So what we need to do is…

- Extends Class StagingPlugin in Common.rb



```
 3  class PhpPlugin < StagingPlugin
 4    def framework
 5      'php'
 6    end
 7
 8    def resource_dir
 9      File.join(File.dirname(__FILE__), 'resources')
10    end
11
12    def stage_application
13      Dir.chdir(destination_directory) do
14        create_app_directories
15        Apache.prepare(destination_directory)
16        system "cp -a #{File.join(resource_dir, "conf.d", "*")} apache/php"
17        copy_source_files
18        create_startup_script
19        create_stop_script
20      end
21    end
22
23    # The Apache start script runs from the root of the staged application.
24    def change_directory_for_start
25      "cd apache"
26    end
27
28    def start_command
29      "bash ./start.sh"
30    end
31
32    def stop_command
33      cmds = []
34      cmds << "CHILDPIDS=$(pgrep -P ${1} -d ' ')"
35      cmds << "kill -9 ${1}"
36      cmds << "for CPID in ${CHILDPIDS};do"
37      cmds << "  kill -9 ${CPID}"
38      cmds << "done"
39      cmds.join("\n")
40    end
41
42    private
43
44    def startup_script
45      vars = environment_hash
46      generate_startup_script(vars) do
47        <<-PHPEOF
48  env > env.log
49  ruby resources/generate_apache_conf $VCAP_APP_PORT $HOME $VCAP_SERVICES #{application_memory}m
50        PHPEOF
51      end
52    end
53
54    def stop_script
55      vars = environment_hash
56      generate_stop_script(vars)
57    end
58
59    def apache_server_root
60      File.join(destination_directory, 'apache')
61    end
62  end
```

Line3 ~ Line 62, 59 lines of codes to support PHP. ☺

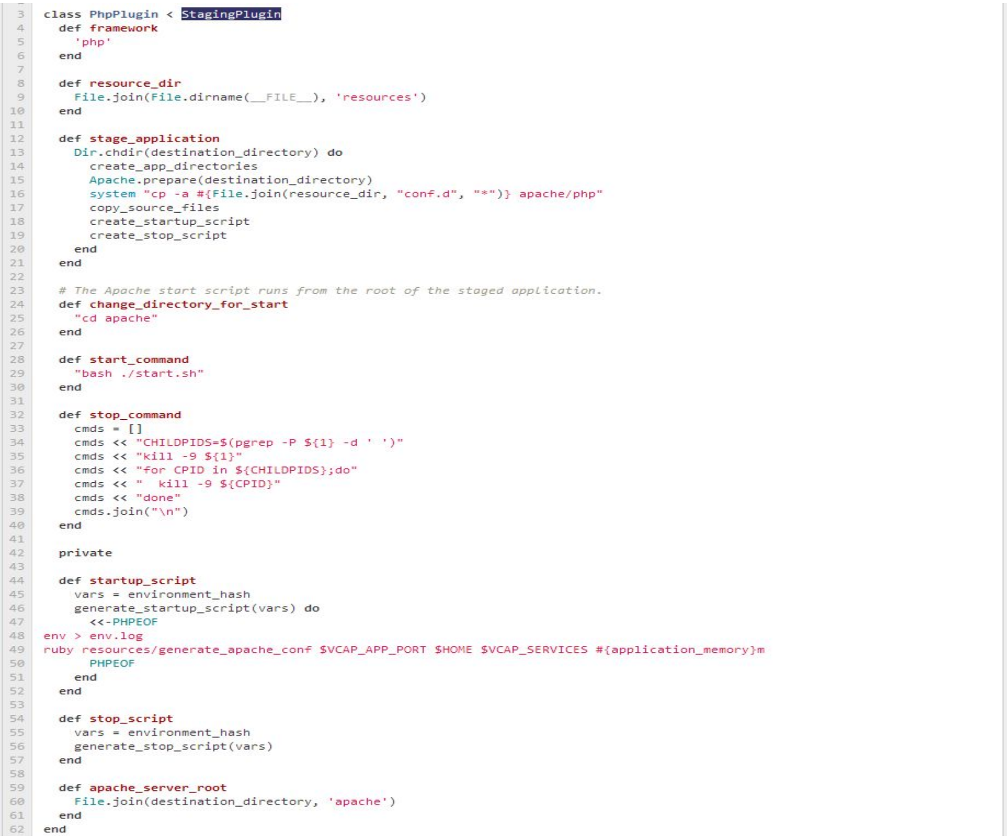


django
grails
java_web
lift
manifests
node
otp_rebar
php
play
rack
rails3
resources
sinatra
spring
standalone

Elastic Runtime Support (Cont.)

Key methods to rewrite:

- stage_application
- start_command
- startup_script
- stop_command
- stop_script



```ruby
class PhpPlugin < StagingPlugin
  def framework
    'php'
  end

  def resource_dir
    File.join(File.dirname(__FILE__), 'resources')
  end

  def stage_application
    Dir.chdir(destination_directory) do
      create_app_directories
      Apache.prepare(destination_directory)
      system "cp -a #{File.join(resource_dir, "conf.d", "*")} apache/php"
      copy_source_files
      create_startup_script
      create_stop_script
    end
  end

  # The Apache start script runs from the root of the staged application.
  def change_directory_for_start
    "cd apache"
  end

  def start_command
    "bash ./start.sh"
  end

  def stop_command
    cmds = []
    cmds << "CHILDPIDS=$(pgrep -P ${1} -d ' ')"
    cmds << "kill -9 ${1}"
    cmds << "for CPID in ${CHILDPIDS};do"
    cmds << "  kill -9 ${CPID}"
    cmds << "done"
    cmds.join("\n")
  end

  private

  def startup_script
    vars = environment_hash
    generate_startup_script(vars) do
      <<-PHPEOF
env > env.log
ruby resources/generate_apache_conf $VCAP_APP_PORT $HOME $VCAP_SERVICES #{application_memory}m
      PHPEOF
    end
  end

  def stop_script
    vars = environment_hash
    generate_stop_script(vars)
  end

  def apache_server_root
    File.join(destination_directory, 'apache')
  end
end
```

Elastic Services Support

Refers to a nice presentation by Nicholas Kushmerick

Cloud Foundry Services in last forum:

- Service advertisement
  - Service Gateway -> Cloud Controller
    - POST /services/v1/offerings
    - DELETE /services/v1/offerings/:label
- Instance management
  - Cloud Controller -> Service Gateway
    - Provision: POST /gateway/v1/configurations
    - Bind: POST /gateway/v1/configurations/:id/handles
    - Unbind: DELETE /gateway/v1/configurations/:id/handles/:handle
    - Unprovision: DELETE /gateway/v1/configurations/:id

Tradeoffs

- Modular Design
- Version Tolerance
- Flexible Runtime/Service
- Elastic Architecture

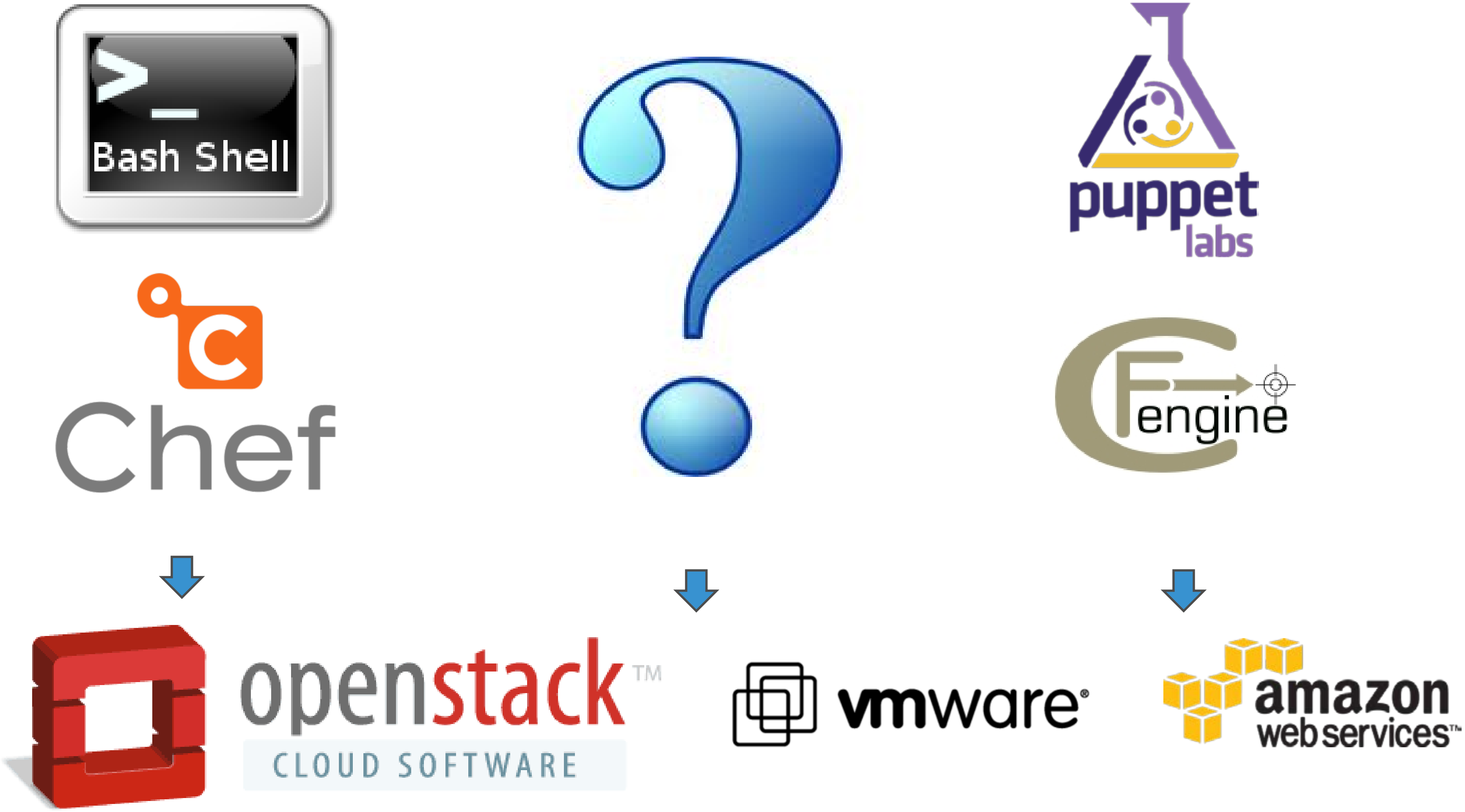
Cause

- Many kinds of nodes
- Many nodes each kind

=

Complex deployment process like other distribution system

How we deployed CloudFoundry?



Practical problem at CloudFoundry.com

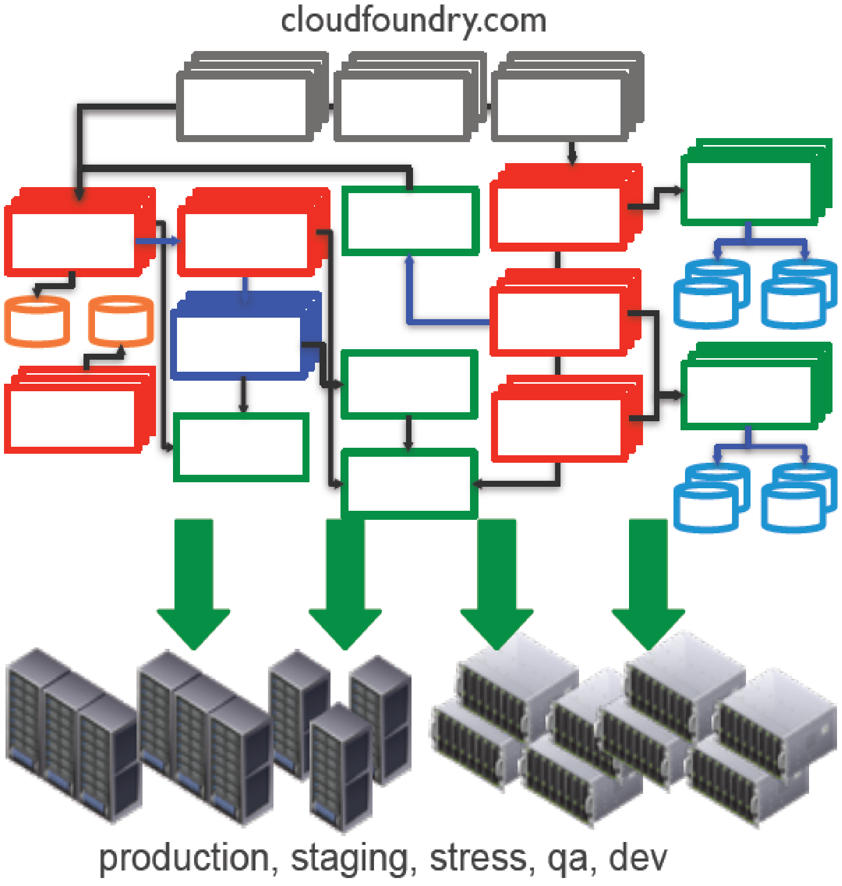
40+ unique node types

75+ unique software packages

500-5,000 VMs

2x/week cf.com updates

Small teams manage many instances



CloudFoundry BOSH

CloudFoundry BOSH is an open source tool-chain for release engineering, deployment, and lifecycle management of large scale distributed services

- Prescriptive way of creating releases and managing systems and services
- It is not a collection of shell scripts, not a pile of Perl

Built to deploy and manage production-class, large scale clusters

Built for DevOps usage and scale by a crack team of veterans

- A project, not a product: command line interface, YAML, etc.

Built from the need to operate cloudfoundry.com

End-to-end management

Generic solution - Any IaaS, Any Service

https://github.com/cloudfoundry/bosh

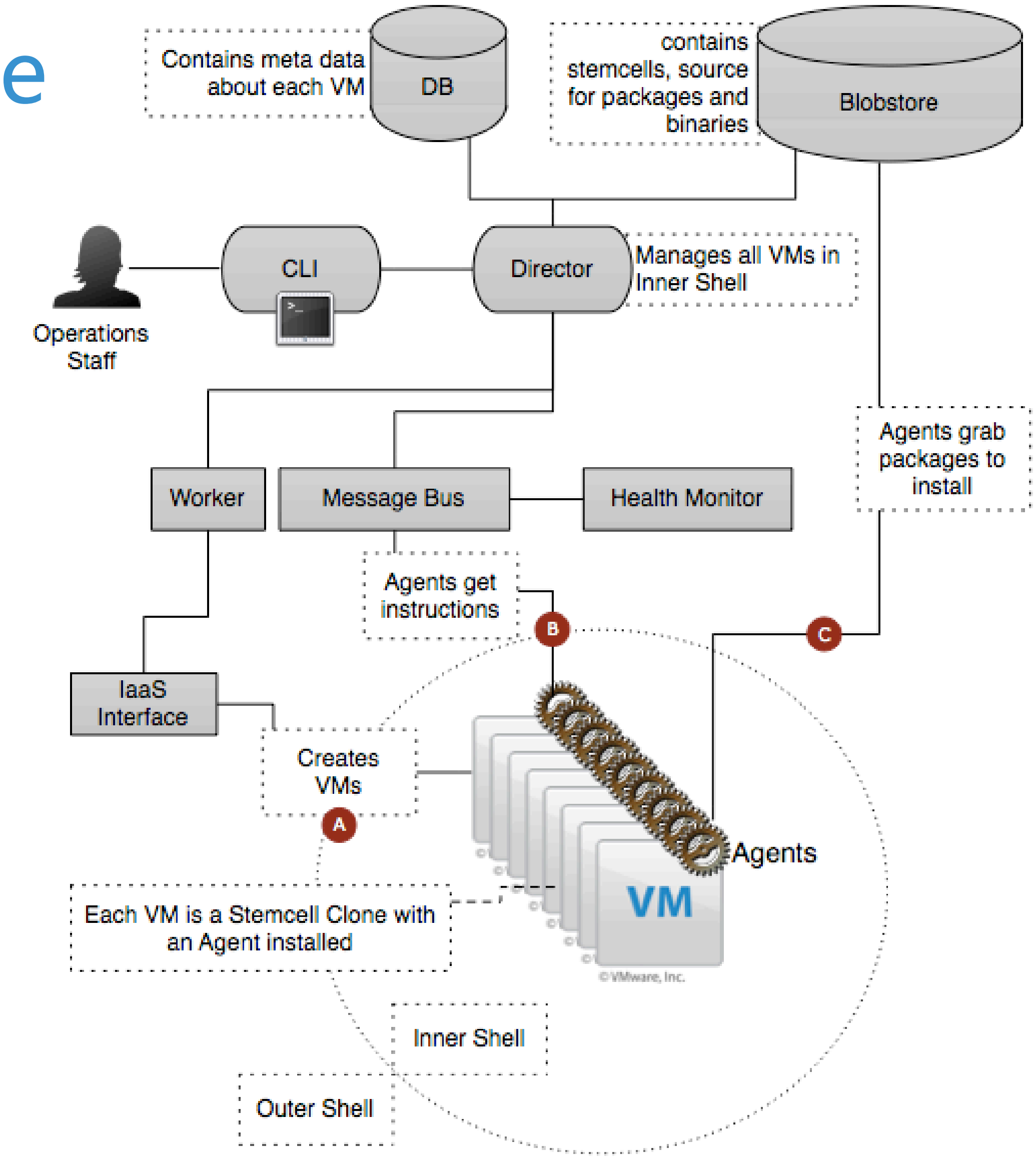
BOSH Architecture

Upload Stemcell

Upload Release

Deploy

- CLI -> Director
- Director -> A
- Agent -> B
- Agent -> C


Contains meta data about each VM
DB
contains stemcells, source for packages and binaries
Blobstore
CLI
Director
Manages all VMs in Inner Shell
Operations Staff
Agents grab packages to install
Worker
Message Bus
Health Monitor
Agents get instructions
B
C
IaaS Interface
Creates VMs
A
Agents
VM
©VMware, Inc.
Each VM is a Stemcell Clone with an Agent installed
Inner Shell
Outer Shell

BOSH Concepts

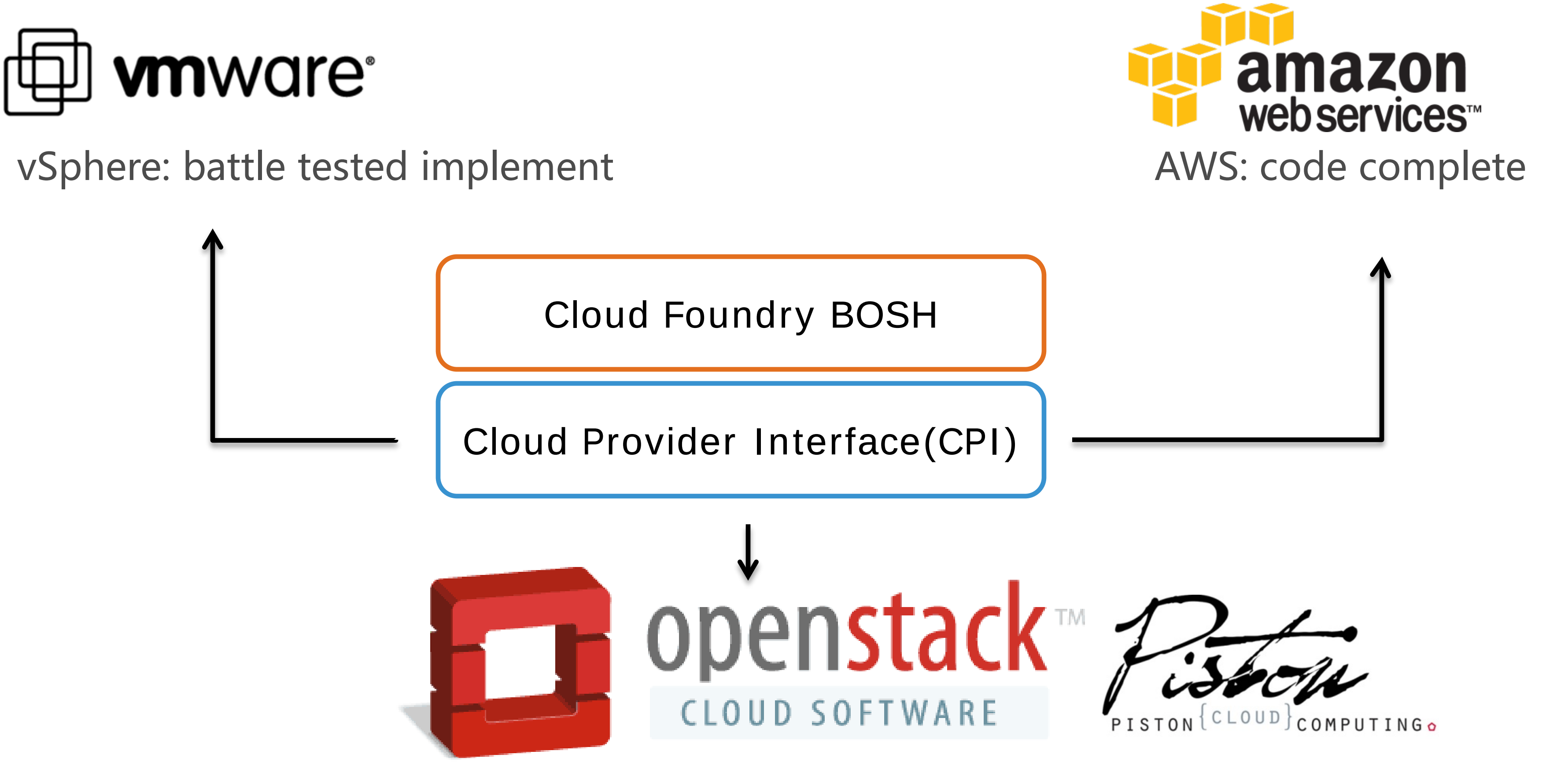
Stemcell
- VM template
- BOSH Agent
- IaaS Plugin

Release
- Jobs

Job
- Packages
- Templates (scripts, confs)
- Monitoring

Package
- Source/blobs
- Dependencies
- Packaging (scripts)

IaaS Neutral



vSphere: battle tested implement

AWS: code complete

Cloud Foundry BOSH

Cloud Provider Interface(CPI)

OpenStack: testable release

https://github.com/piston/openstack-bosh-cpi

EMC²

Cloud Provider Interface

Stemcell

- create_stemcell (image, cloud_properties)
- delete_stemcell (stemcell)

VM

- create_vm (agent_id, stemcell, resource_pool, networks, disk_locality, env)
- delete_vm (vm)
- reboot_vm (vm)
- configure_networks (vm, networks)

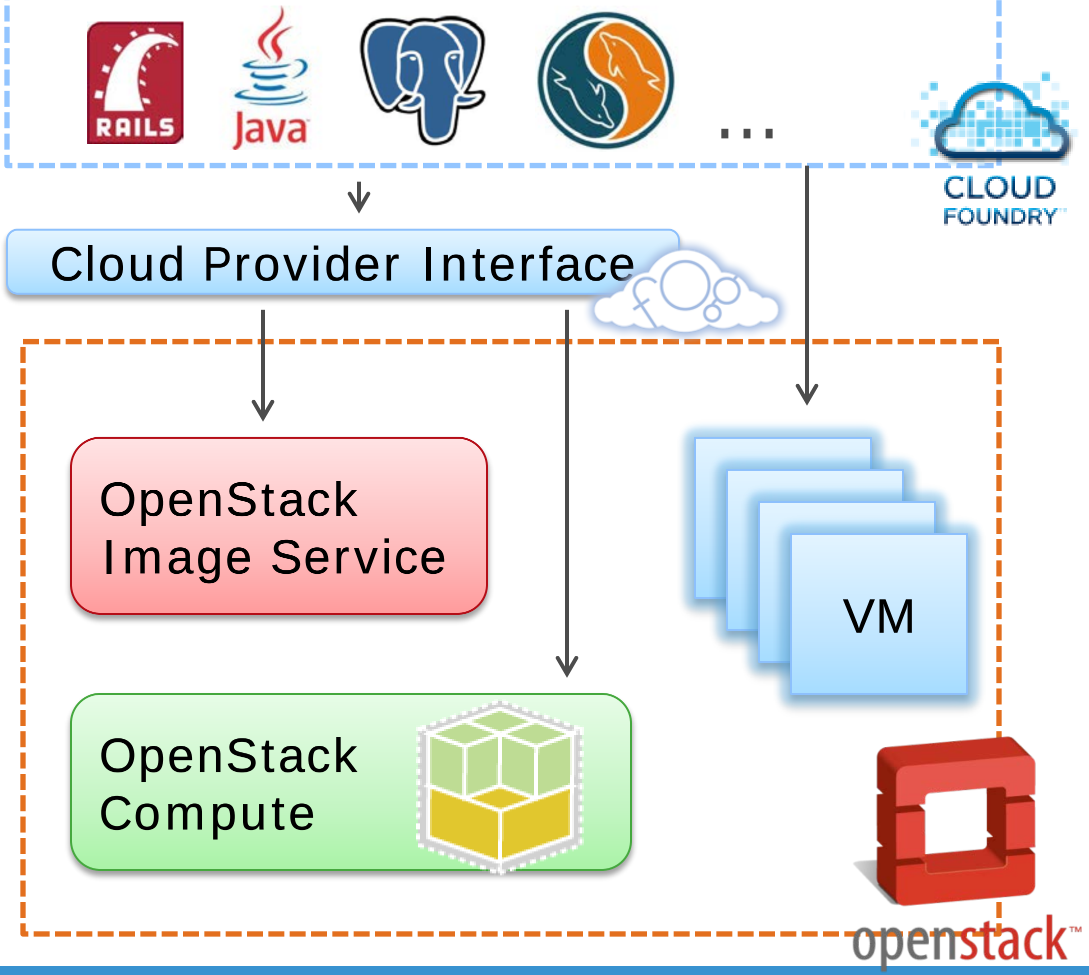
Disk

- create_disk (size, vm_locality)
- delete_disk (disk)
- attach_disk (vm, disk)
- detach_disk (vm, disk)

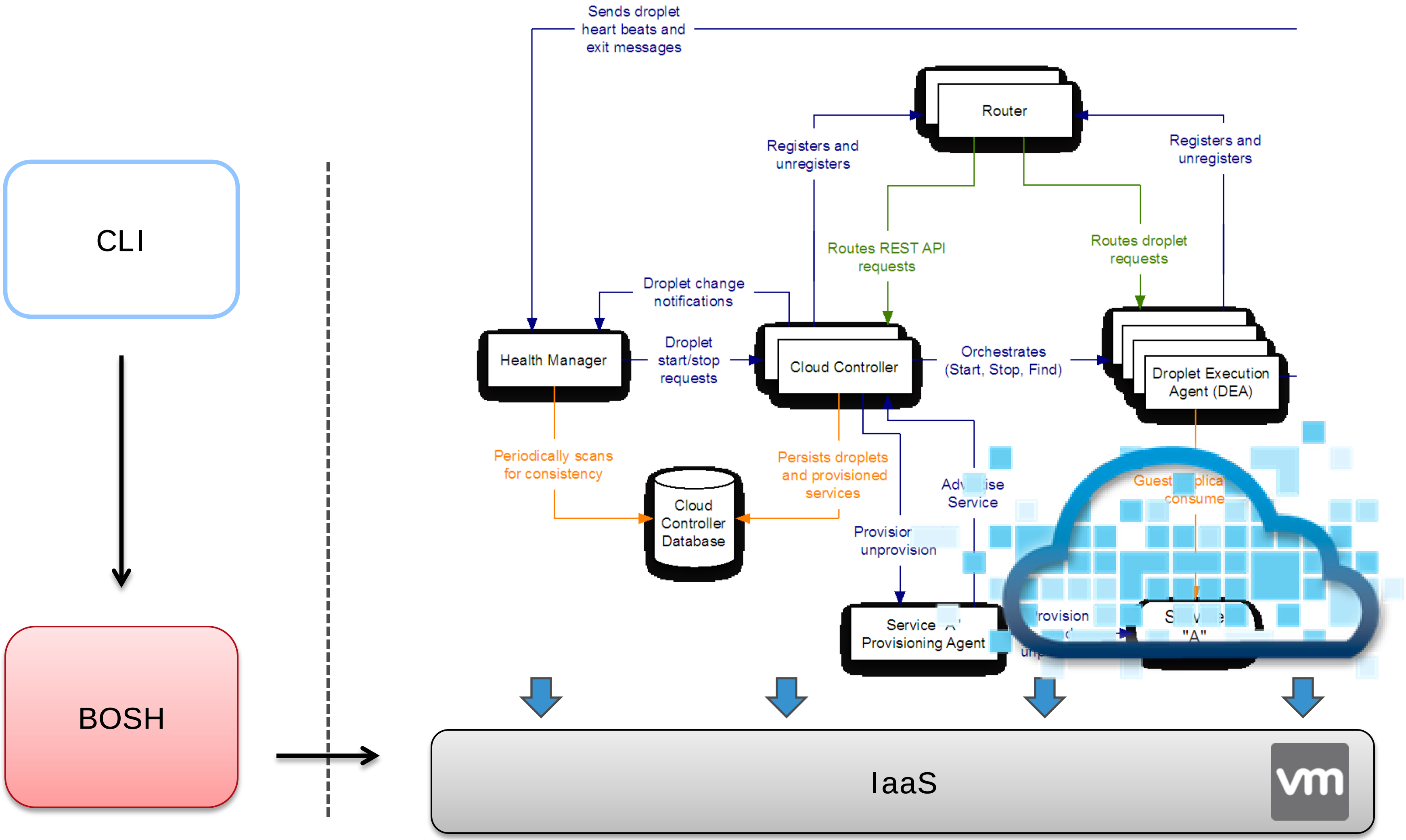
Cloud Provider Interface (Impl.)

For OpenStack

- Stemcell
  - OpenStack Image Service
- VM, Disk & Network
  - OpenStack Compute



Deploy PaaS with BOSH



CLI

BOSH

Sends droplet heart beats and exit messages

Router

Registers and unregisters

Registers and unregisters

Routes REST API requests

Routes droplet requests

Droplet change notifications

Health Manager

Droplet start/stop requests

Cloud Controller

Orchestrates (Start, Stop, Find)

Droplet Execution Agent (DEA)

Periodically scans for consistency

Cloud Controller Database

Persists droplets and provisioned services

Advertise Service

Provision/ unprovision

Service "A" Provisioning Agent

Guest application consume

Provision/ unprovision

Service "A"

IaaS

Demo

- Deploy CloudFoundry using BOSH
  - Upload Stemcell
  - Upload Release
    - bosh create release
    - bosh upload release
  - Write deployment file
  - Deploy
- CloudFoundry HelloWorld
  - Login
  - Push Application

Deployments

Release

Network

Resource pools

Jobs

Properties

Update concurrency

Compilation workers

Cloud properties

Deployments for CloudFoundy

Cloudfoundry.yml

```
name: cloudfoundry

release:
    name: cloudfoundry
    version: 89.1-dev

compilation:
    workers: 4
    network: default
    cloud_properties:
      ram: 1024
      disk: 2048
      cpu: 2

update:
    canaries: 1
    canary_watch_time: 3000-90000
    update_watch_time: 3000-90000
    max_in_flight: 2
    max_errors: 1
```

Deployments for CloudFoundy (Cont.)

Cloudfoundry.yml

```
networks:
 - name: default
  subnets:
  - static:
   - 192.168.2.50 - 192.168.2.89
   range: 192.168.2.0/24
   gateway: 192.168.2.1
   dns:
   - 10.254.174.10
   cloud_properties:
    name: PrivateNetwork
- name: lb
  subnets:
  - static:
   - 192.168.2.90 - 192.168.2.99
   range: 192.168.2.0/24
   gateway: 192.168.2.1
   dns:
   - 10.254.174.10
   cloud_properties:
    name: PrivateNetwork
```

Deployments for CloudFoundy (Cont.)

Cloudfoundry.yml

```
resource_pools:

  - name: infrastructure
    network: default
    size: 29
    stemcell:
      name: bosh-stemcell
      version: 0.6.2
    cloud_properties:
      ram: 256
      disk: 2048
      cpu: 1
    env:
      bosh:
        password:
```

Deployments for CloudFoundy (Cont.)

Cloudfoundry.yml

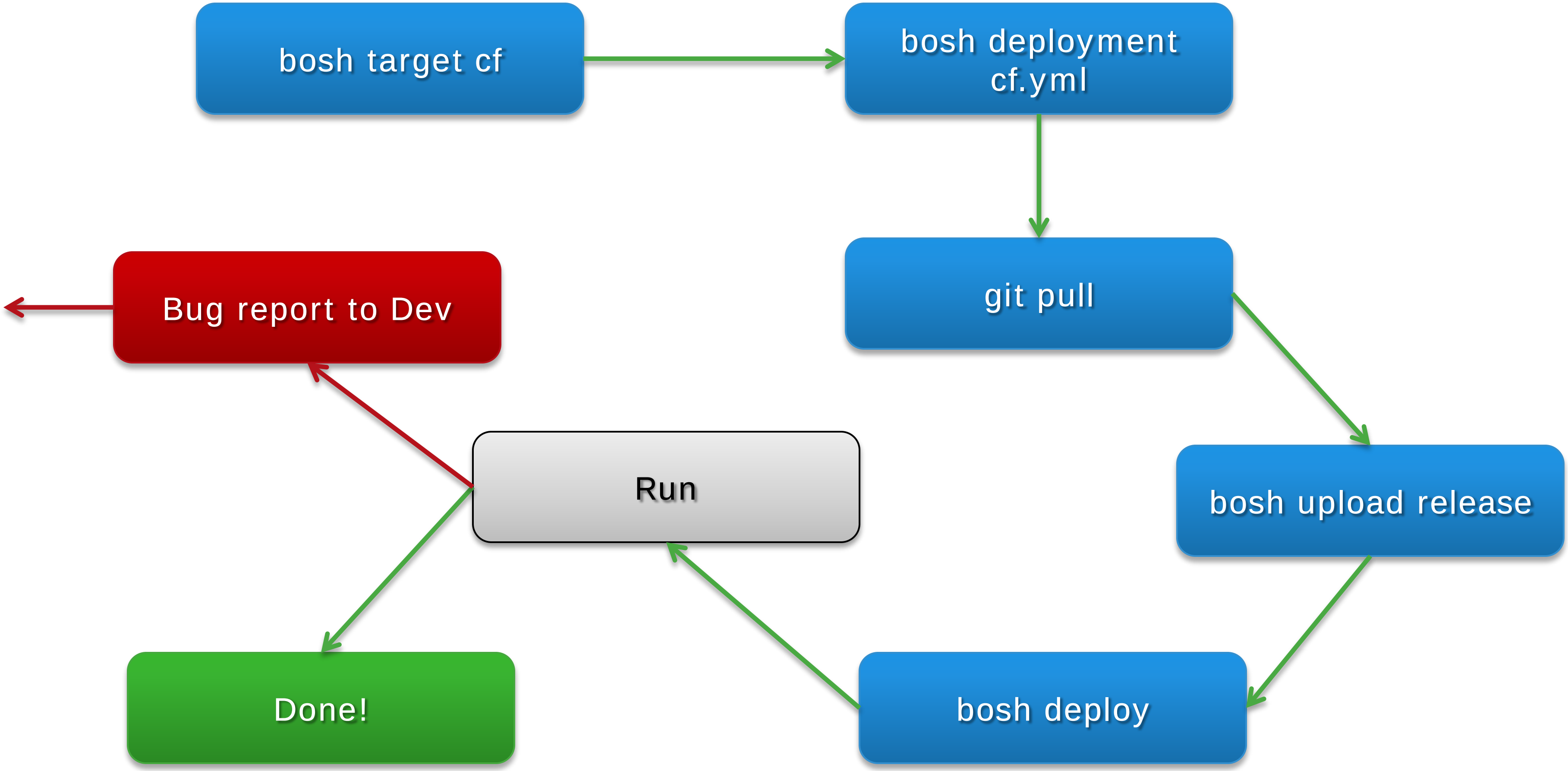
```
jobs:

  - name: cloud_controller
    template: cloud_controller
    instances: 1
    resource_pool: infrastructure
    networks:
    - name: default
      static_ips:
      - 192.168.2.60

  - name: nats
    template: nats
    instances: 1
    resource_pool: infrastructure
    networks:
    - name: default
      static_ips:
      - 192.168.2.52
```

Deployments(CloudFoundy)

Cloudfoundry.yml

```
properties:
    domain: cflocal.com

    env: {}

    networks:
      apps: default
      management: default

    nats:
      user: nats
      password: aaa3ij3122
      address: 192.168.2.52
      port: 4222

router:
    status:
      port: 8080
      user: aaaUxXlS0pc71wVef
      password: aaamaIf9vPV4mJyBe
```

User Case


bosh target cf

bosh deployment cf.yml

git pull

bosh upload release

bosh deploy

Run

Bug report to Dev

Done!

Acknowledgments

VMware China R&D Center

Network & Information Center,
Shanghai Jiao Tong Univ.

CloudFoundry Community
Sina Weibo: @CloudFoundry
http://www.cloudfoundry.org

Piston Community
https://github.com/piston/openstack-bosh-cpi

EMC^{2}®