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
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*
First Touch...

```bash
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 (OK): OK
Push Status: OK
Staging Application: OK
Starting Application: OK

Hello from Cloud Foundry
```
First Touch...(Cont.)

- Upload your app
- Memory of each instance
- Change served instances
- Services of current app
- Start, stop, update, restart
- Information of app

Your apps deployed
Your all created services
What CloudFoundry Offer?

Apache2 license
What CloudFoundry Offer? (Cont.)

From *Cloud Foundry Launch Event*, April, 2011
PaaS Architecture Pattern

- Request Router
- Request Router
- Request Router
- App Exec Engine
- App Exec Engine
- App Exec Engine
- App Exec Engine
- App Exec Engine
- App Exec Engine
- App Exec Engine
- App Exec Engine
- MySQL
- MongoDB
- RabbitMQ
- Redis
- Cloud Controller
- Cloud Controller
- Health Manager
- Health Manager
PaaS Architecture Pattern (Cont.)

heroku
PaaS Architecture Pattern (Cont.)
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

\[\rightarrow\] 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
Open Ecosystem

• Open Dev Proc

- cloudfoundry.com
  - production updated ~2x/week
- github
  - github.com/cloudfoundry
  - on +2 and ✓ change pushed to github
  - test verification score ✓
- Committers
- Jenkins CI
  - ci.cloudfoundry.org

• Partners & Communities

- ActiveState
  - Stackato, Python Support
- af
  - PHP Support
- Joyent
  - node.js Support
- PISTON CLOUD COMPUTING
- EMC²

Public PaaS
- Ruby Cloud Foundry on SAE

Public PaaS
- Cloud Foundry on Grand Cloud

Public PaaS
- Cloud Foundry & Data Director on vSphere/vCloud

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

Stager

StagingPlugin

PHP  Rails3  Java_Web  Play  Server

extends StagingPlugin

Start Script  Stop Script

App

...
Elastic Runtime Support (Cont.)

So what we need to do is...

- Extends Class StagingPlugin in Common.rb

Line 3 ~ Line 62, 59 lines of codes to support PHP.

😊
Elastic Runtime Support (Cont.)

Key methods to rewrite:

- `stage_application`
- `start_command`
- `startup_script`
- `stop_command`
- `stop_script`
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/offering
    - DELETE /services/v1/offering/: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
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
- Cloud Foundry BOSH
- Cloud Provider Interface (CPI)
- AWS: code complete
- OpenStack: testable release

https://github.com/piston/openstack-bosh-cpi
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
  - IaaS
  - BOSH
  - Cloud Controller
    - Cloud Controller Database
    - Health Manager
      - Sends droplet heart beats and exit messages
    - Cloud Controller
      - Routes REST API requests
      - Routes droplet requests
      - Orchestrates (Start, Stop, Find)
      - Registers and unregisters
      - Persist droplets and provisioned services
    - Droplet Execution Agent (DEA)
      - Registers and unregisters
      - Routes droplet requests
    - Droplet Provisioning Agent
      - Provision and unprovision
      - Advertise Service
    - Service
      - Guest replica consume
      - Cloud
        - "A"
  - IaaS
    - vm
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 CloudFoundry

Cloudfoundry.yml

```yaml
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 (CloudFoundry)

Cloudfoundry.yml

properties:
  domain: cflocal.com

env: {}

networks:
  apps: default
  management: default

nats:
  user: nats
  password: aad3ij3122
  address: 192.168.2.52
  port: 4222

router:
  status:
    port: 8080
    user: aadUxXls0pc71wVe
    password: aadamaIf9vPV4mJyBe
User Case

1. Run `bosh target cf`
2. Run `bosh deployment cf.yml`
3. Run `git pull`
4. Run `bosh upload release`
5. Run `bosh deploy`
6. Done!
7. Bug report to Dev
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/open
stack-bosh-cpi