NOVEMBER 2 & 3, 2015 | BERLIN
CLOUD FOUNDRY
SUMMIT
Building of an external CPI for CloudStack

Guillaume Berche & Pierre Oblin
Introduction

• Who we are
• Disclaimer
• Goal of this talk
• Why did we build a [new] CPI for CloudStack?
Outline

• Bosh & CPI intro
• CloudStack CPI design tradeoffs
• Status, future work, learnings
Bosh and CPI overview
Bosh intro
Bosh CPI overview

- external CPI
- cpi drives IaaS & agent
- agent role
- registry
- nats
- => director to cpi api
CPI API V1 overview

- **Stemcell management**
  - `create_stemcell`
  - `delete_stemcell`

- **VM management**
  - `create_vm`
  - `delete_vm`
  - `has_vm`
  - `reboot_vm`
  - `set_vm_metadata`
  - `configure_networks`

- **Disk management**
  - `delete_disk`
  - `has_disk`
  - `attach_disk`
  - `detach_disk`
  - `get_disks`

- **Disk snapshots**
  - `snapshot_disk`
  - `delete_snapshot`
  - `current_vm_id`

---

**Stemcell management**

- `create_stemcell`

Creates a reusable VM image in the IaaS from the stemcell image. It's used later for creating VMs. For example, AWS CPI creates an AMI and returns AMI ID.

See [Stemcell Building](#) for more details.

**Arguments**

- `image_path [String]`: Path to the stemcell image extracted from the stemcell tarball on a local filesystem.
- `cloud_properties [Hash]`: Cloud properties hash extracted from the stemcell tarball.

**Example**

```json
[
  {"image_path": "/tmp/extracted-stemcell-348254vdsn87fr/image",
   "cloud_properties": {
     "name": "bosh-openstack-esxi-ubuntu-trusty-go_agent",
     "version": "2872",
     "infrastructure": "openstack",
     "hypervisor": "esxi",
     "disk": "3072",
     "disk_format": "ovf",
     "container_format": "bare",
     "os_type": "linux",
     "os_distro": "ubuntu",
     "architecture": "x86_64",
     "auto_disk_config": true
   }
  }
]
```

**Returned**

- `stemcell_cid [String]`: Cloud ID of the created stemcell (e.g., stemcells in AWS CPI are made into AMIs so `cid` would be `ami-83fdflf`)
Bosh CPI overview
Bosh agent configuration overview

1. load startup settings provided in stemcell
   i.e. bootstrap settings sources among: meta-data server, config drive, cdrom

2. load bootstrap settings:
   i.e; iaas-specific initial networking & registry lookup URL

3. initial registry config loading
   network configs
   disks

4. dynamic configs through NATs messages
   network reconfig,
   disk resize…
VM instanciation illustrated

- **DIRECTOR**
  - director
  - create_vm

- **CPI**
  - cpi
  - create_vm
  - set user-data
  - feed bosh registry:
    - networking, root+eph disks, bosh nats, blobstore

- **CLOUDSTACK**
  - bosh_registry
  - cloudstack_api
  - vrouter
  - provision_vm
  - set user data in metadata service
  - vm boostraps in dhcp,
    - bosh-agent starts
    - (reading agent.json)
  - query meta-data server:
    - get bosh_registry address from user data
  - gets bootstrap info, ip adress and disks

- **VM**
  - vm
  - bosh_agent
  - reconfigure network static ip
  - mount and partition ephemeral disk
CloudStack CPI design choices
Context and constraints

- Provide flexibility for innovation and optimization while remaining opensource
- Chicken and egg: CPI and stemcell
- Bosh is in active development, incomplete cpi doc
- Lack of public ci infrastructure supporting cpi community contributions
  - corp proxies block us running concourse
  - lack public cloudstack instance
Design tradeoffs

• Where to start: stemcell or CPI?
• Programming language: ruby, go, java?
• CloudStack api client
• Child process invocation model
  • per invocation process or daemon mode?
  • stdin/stdout or REST facade?
• Stemcell to cloudstack vm templates
  • location constraints for bosh-init/director?
  • from volume or HTTP?
  • a light stemcell to kick things off
• CS dynamic metadata URL
  • bosh agent evolution?
  • stemcell iptables workaround?
Status, future work, learnings
Status

• Supported: ~ CF-feature-ready
  – in advanced zone networking: static/dynamic lps
  – vm & disk life cycle
  – configureable disk offering (for eph, root, persistent disks)
  – light stemcell & stemcell upload

• Limitations
  – single persistent disk support (preventing persistent disk resizing)
  – single nic support
  – no yet floating ip/vip
  – no yet disk snapshots
Future short-term work

- Stemcell: Xen (+KVM), Ubuntu+Centos
  - another infrastructure in bosh/stemcell_builder ?
  - patch one of the official stemcell ?
- Private CI (concourse): BATS + CPI lifecycle tests +…
- Local storage for persistent disks
  - snapshot
  - live migration
  - vm placement to existing local storage
- Happy collaborate and committed to support the community: public CI
  - Any public CloudStack instance for public ci ?
- Apply to the CF incubator process
Challenges met & potential bosh improvements

- **documentation**: (PR [docs-bosh/63](https://github.com/bosh-bosh-docs))
  - reference implementations
  - unit tests / integration tests
  - bdd/living doc
  - design docs
  - API developer docs.

- **CPI daemon support in bosh-init** ([Issue CS_CPI/1](https://github.com/bosh-bosh-init/17))

- **bosh-agent cloud-stack style volume support**

- **laas-specific registry impl** ([Issue CS_CPI/1](https://github.com/bosh-bosh-init/17))
  - => use NATS for disk mount support ?

- **bosh-init optim for CPI development**
  - (external blobstore) ([Issue bosh-init/17](https://github.com/bosh-bosh-init/17))
Thanks,

Questions?

@gberche
@gberche-orange
guillaume.berche[at]orange.com

@poblin-orange
pierre.oblin[at]orange.com