Pivotal

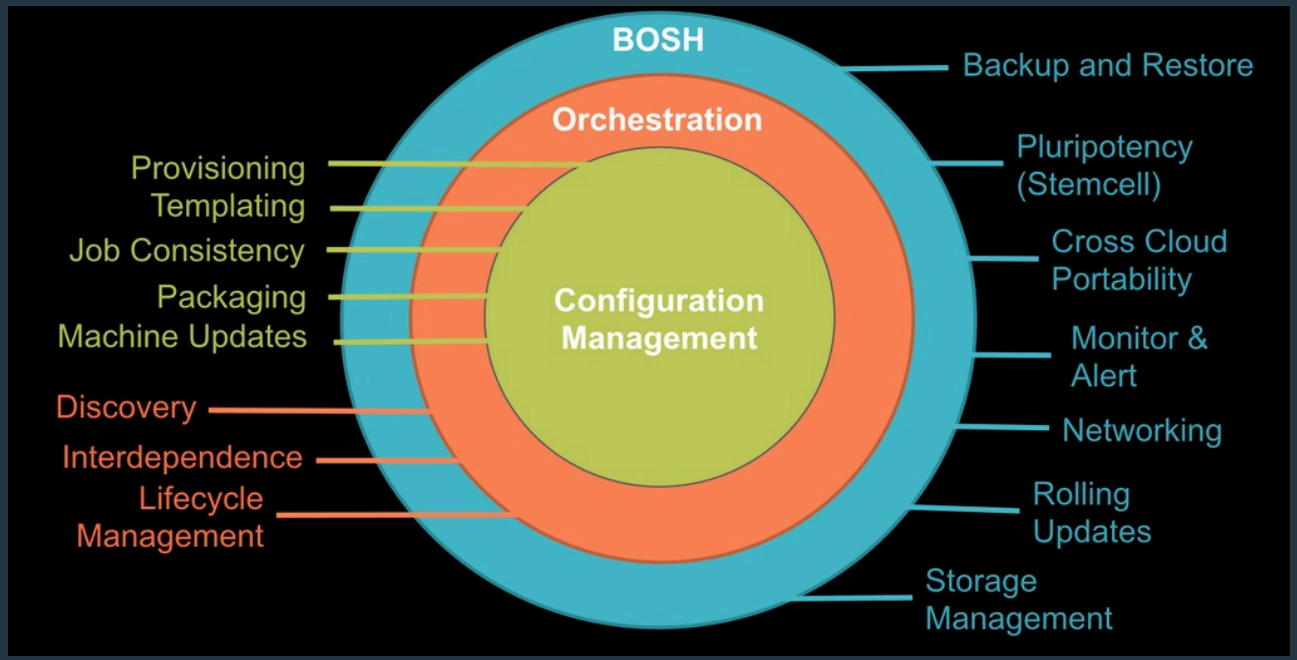
BOSH Fundamentals

Agenda

- BOSH Overview
- BOSH Architecture
- BOSH Deployments and Releases
- BOSH HA and Platform Updates

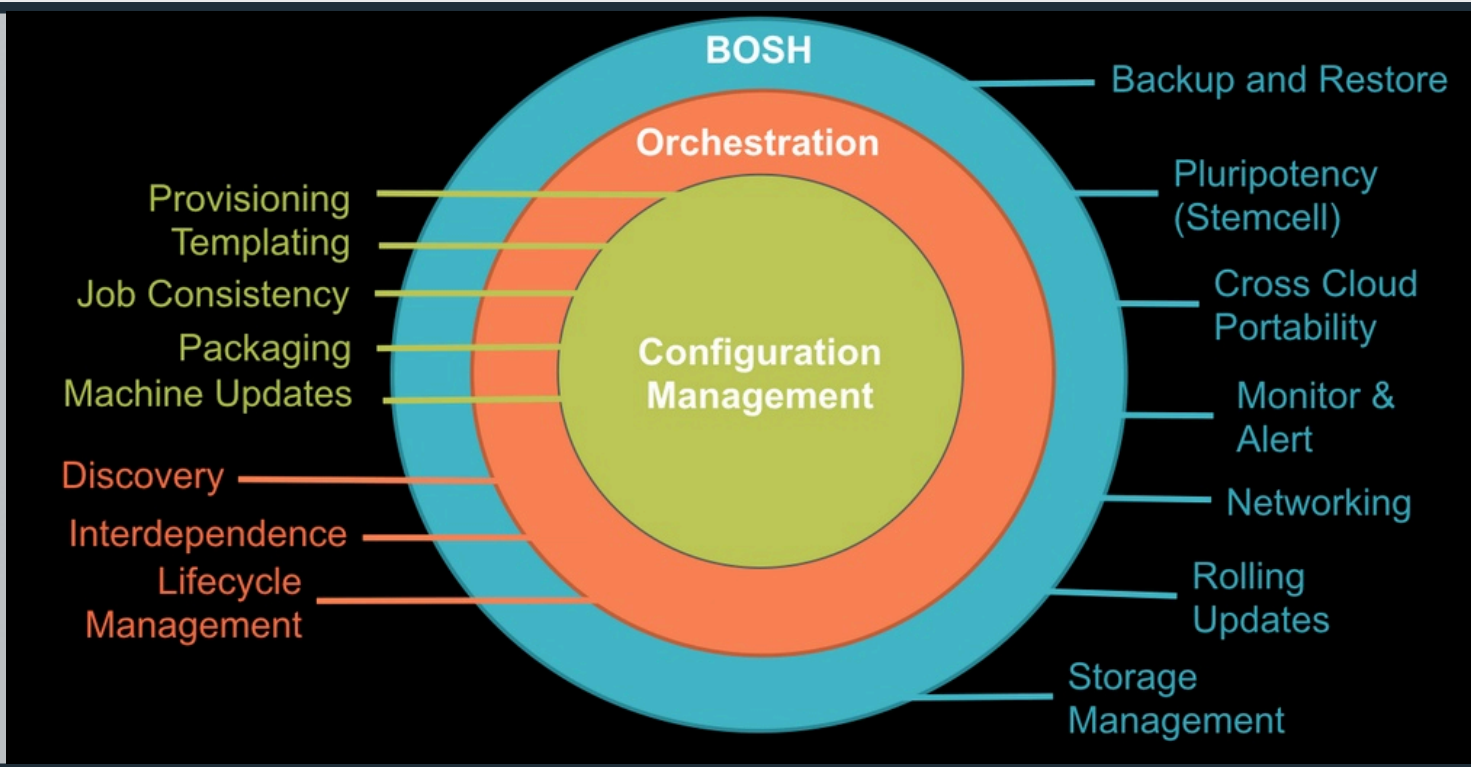
Pivotal

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



Why BOSH?

- Provision services, not machines
- Enables continuous delivery
- Cloud-agnostic view of Platform Ops
- Holistic Toolchain for “rule them all"
- Eliminate bespoke automation on top of config management



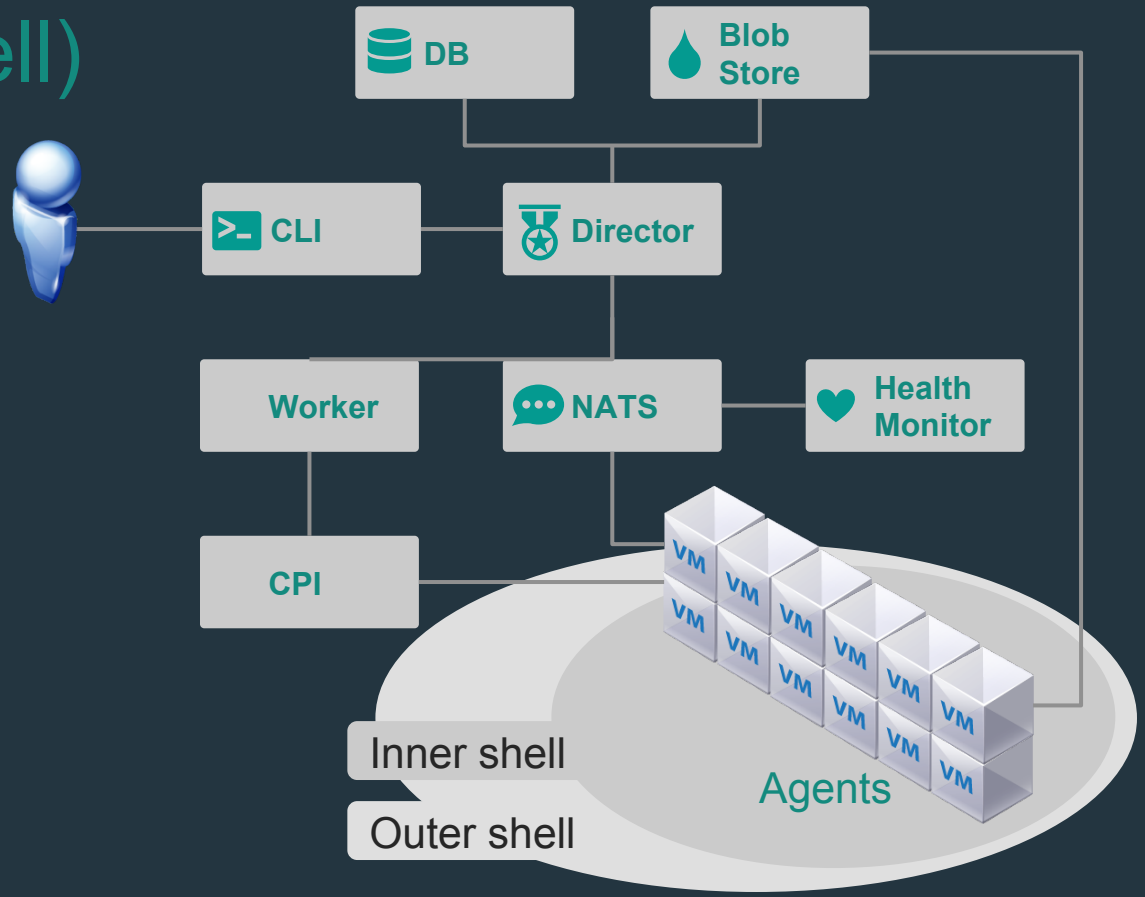
Pivotal

BOSH (Outer Shell) Logical View

Deploys and manages large scale distributed systems. **BOSH** provides the means to go from deployment (i.e., Chef/Puppet) to VM creation and management (i.e., cloud CPI). It includes interfaces for vSphere, vCloud, AWS and OpenStack. Additional CPI can be written for alternative IaaS providers.

Key Elements:

- CLI
- Director
- Blobstore
- Workers
- Message Bus
- Health Monitor
- IaaS CPI
- Agents



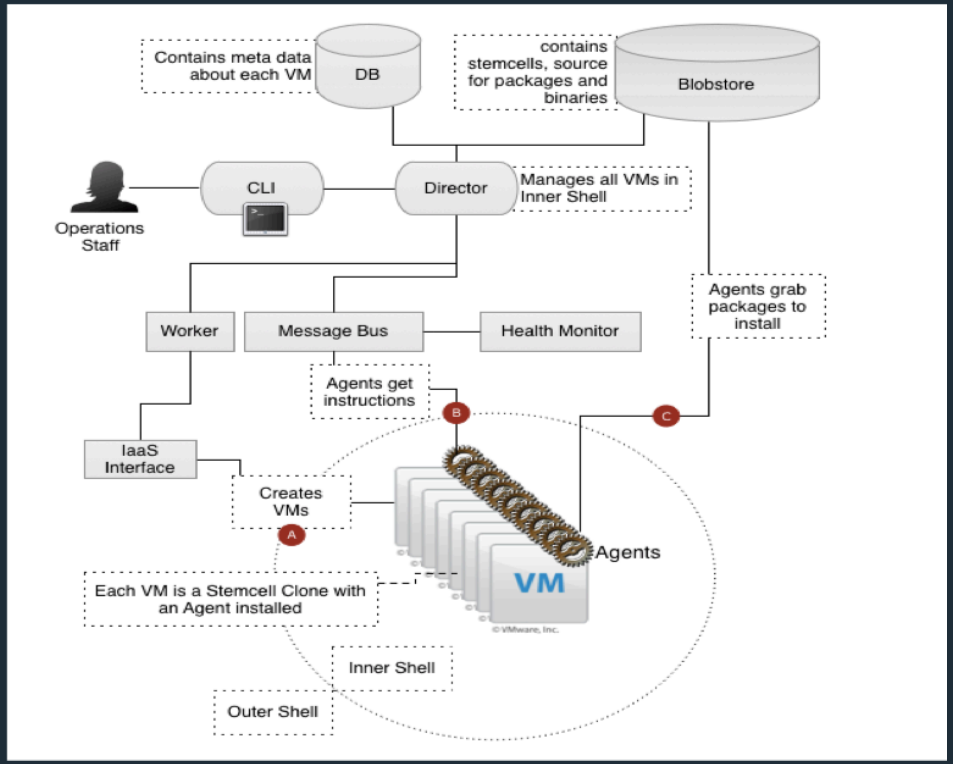
The BOSH Architecture

Very similar to CF architecture itself

Director as analogy to Cloud Controller

Different CPIs exist per IaaS implementation

Workers responsible for executing tasks as dictated by Director



BOSH: Cloud Provider Interface

Stemcell

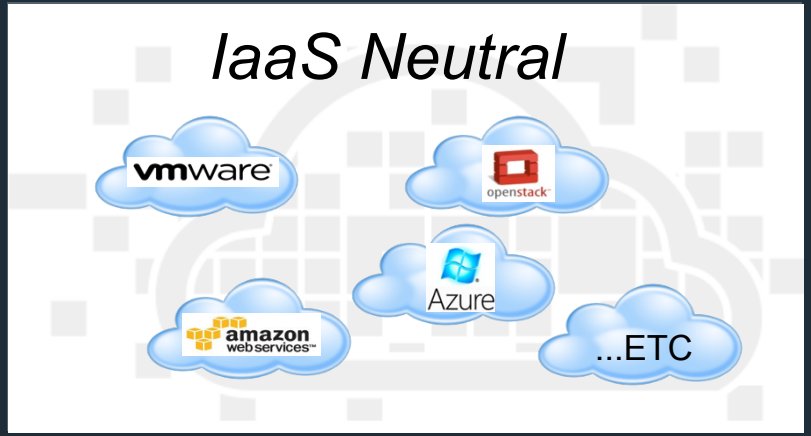
create_stemcell(image, cloud_properties)
delete_stemcell(stemcell_id)

VM

create_vm(agent_id, stemcell_id, resource_pool,
networks, disk_locality, env)
delete_vm(vm_id)
reboot_vm(vm_id)
configure_networks(vm_id, networks)

Disk

create_disk(size, vm_locality)
delete_disk(disk_id)
attach_disk(vm_id, disk_id)
detach_disk(vm_id, disk_id)

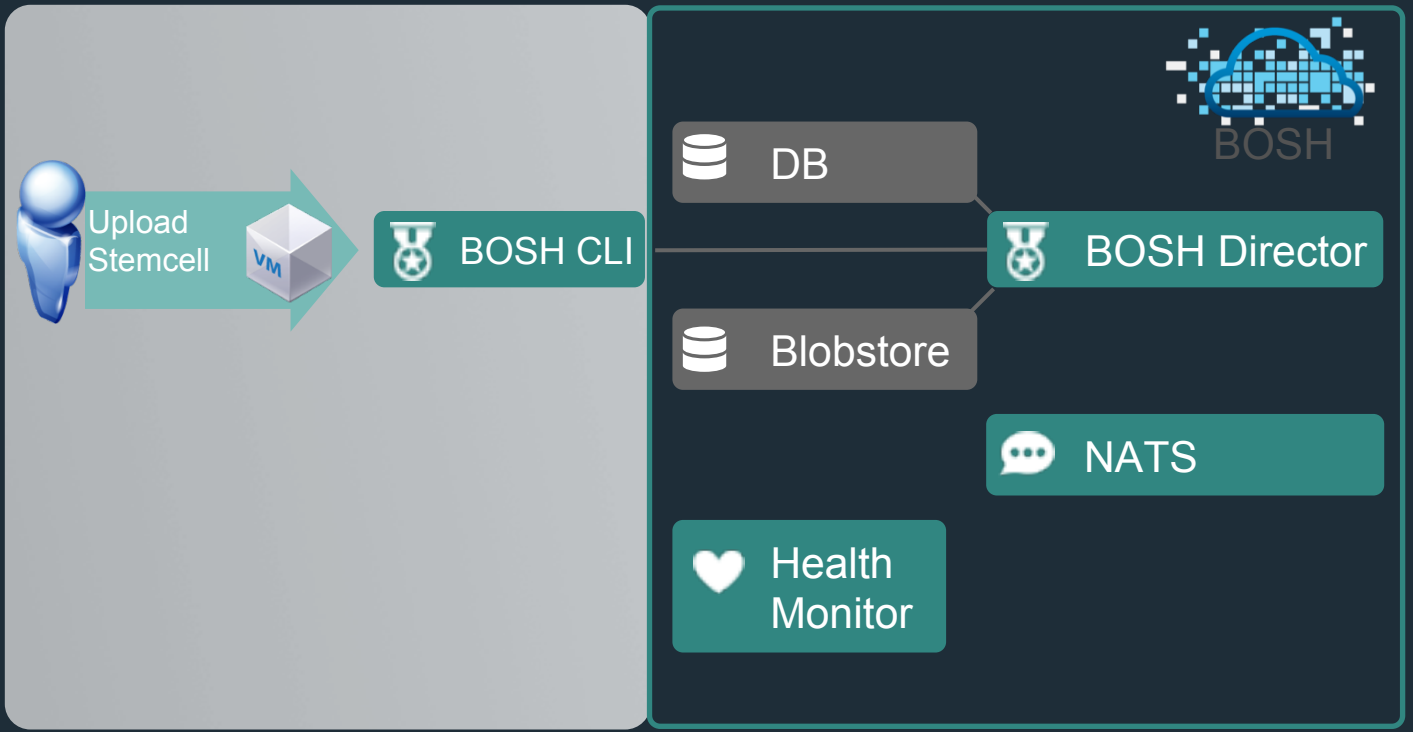


Pivotal

3 Components of a BOSH Deployment

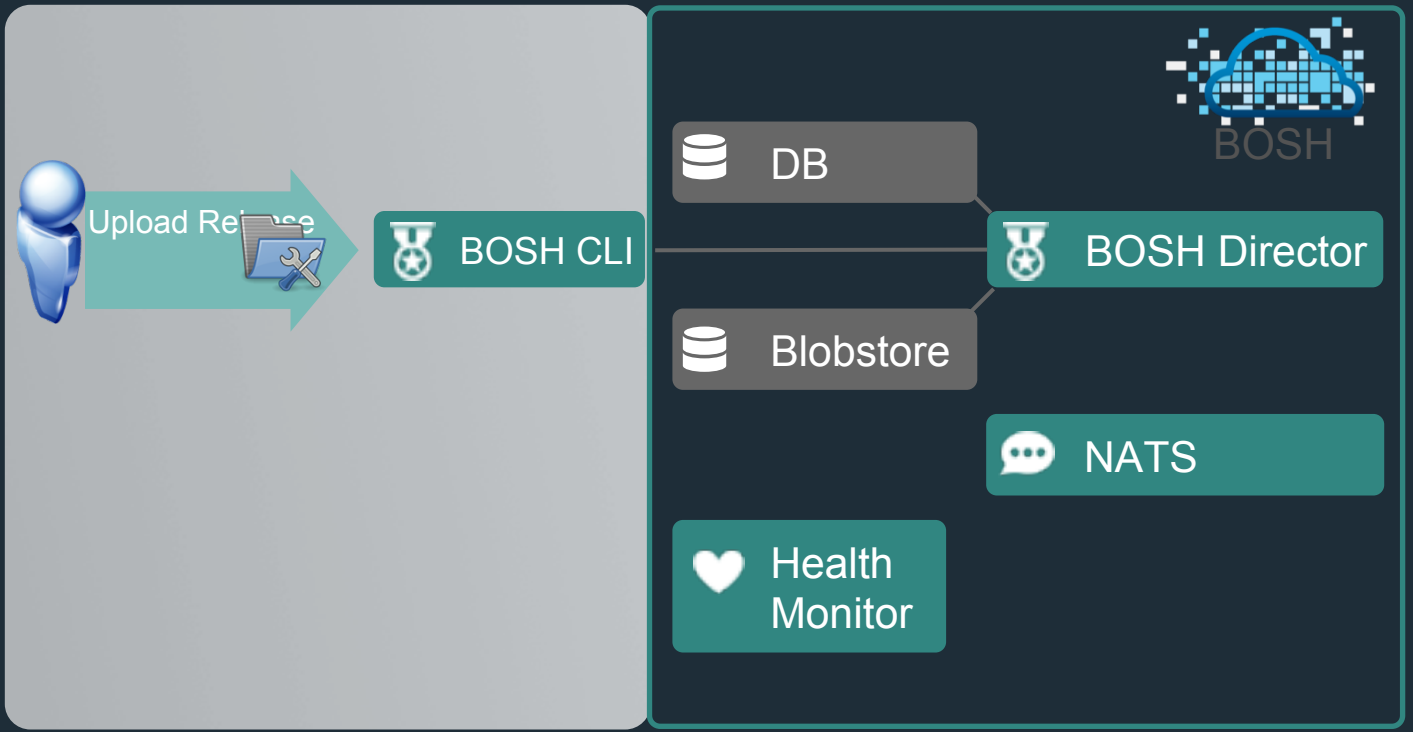
Pivotal

BOSH deployment



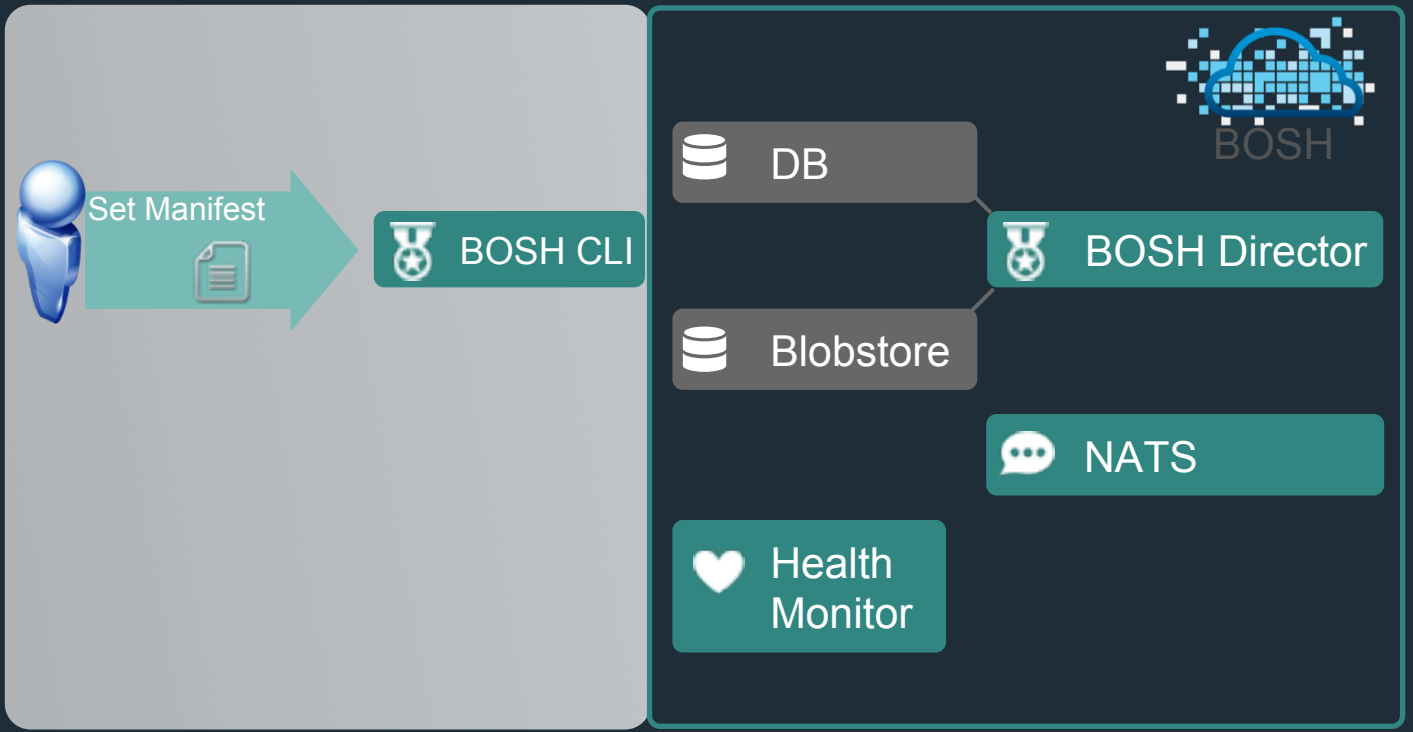
Pivotal

BOSH deployment



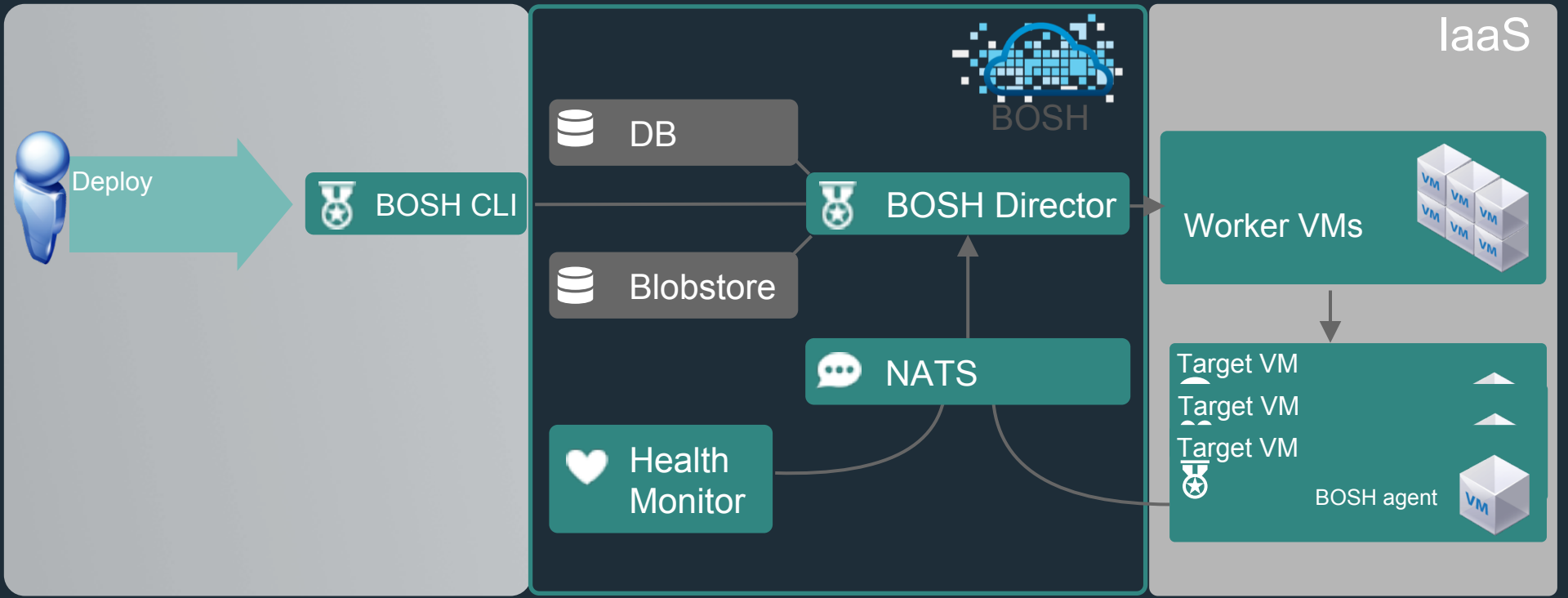
Pivotal

BOSH deployment



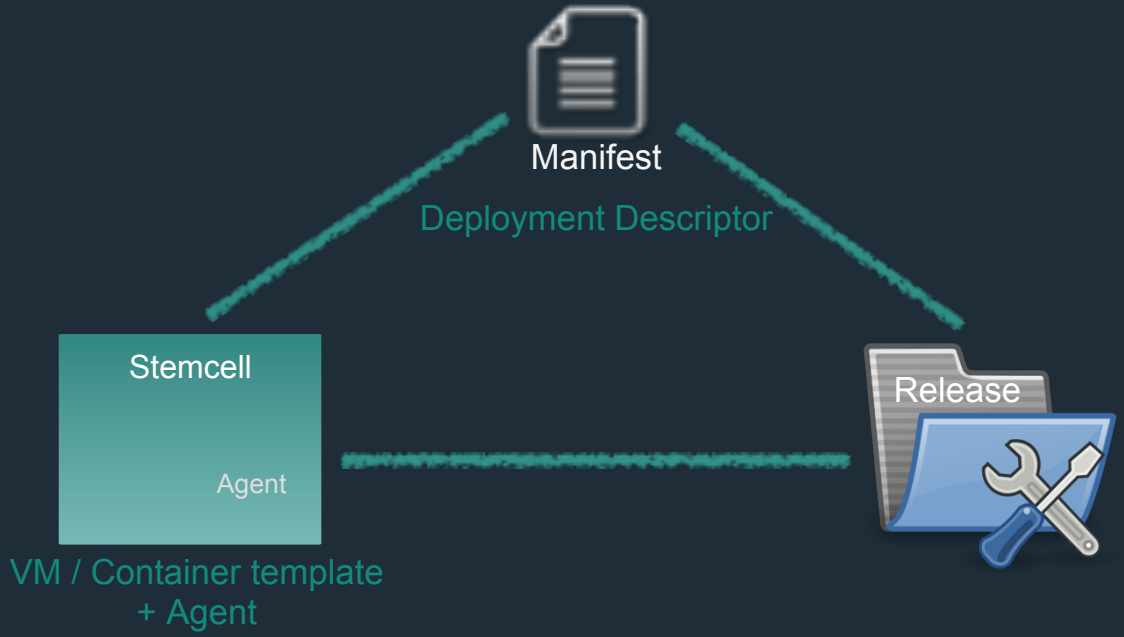
Pivotal

BOSH deployment



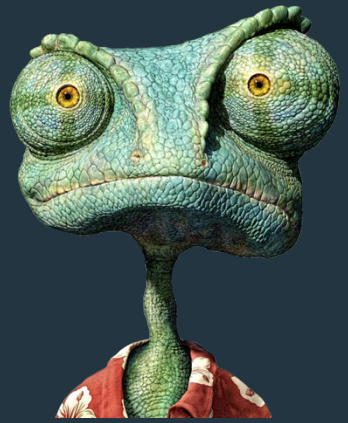
Pivotal

3 Components of a BOSH Deployment



Pivotal

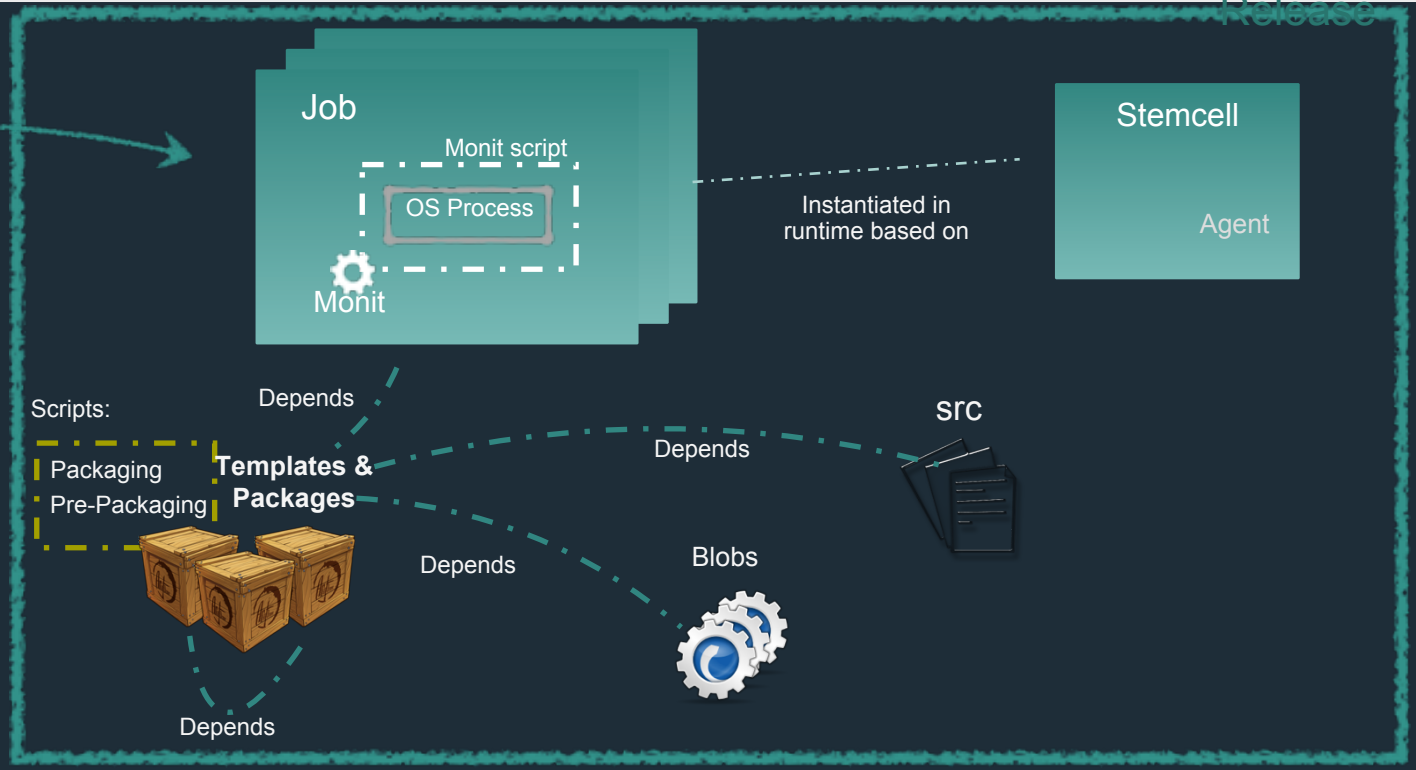
… so what exactly is a BOSH release?



Anatomy of BOSH releases

```
49 jobs:
50 - instances: 1
51   name: postgres
52   networks:
53   - name: default-network
54     static_ips:
55     - 10.244.0.2
56   persistent_disk: 2024
57   release: bosh_intro
58   resource_pool: default-pool
59   template: postgres
60   properties:
61     port: 5637
```

Manifest



Example of Jobs:
- DEA (CF)
- CloudController (CF)
- Service Broker (MySQL & Rabbit)
- Locator (possible GemFire release)

http://docs.cloudfoundry.org/bosh/create-release.html

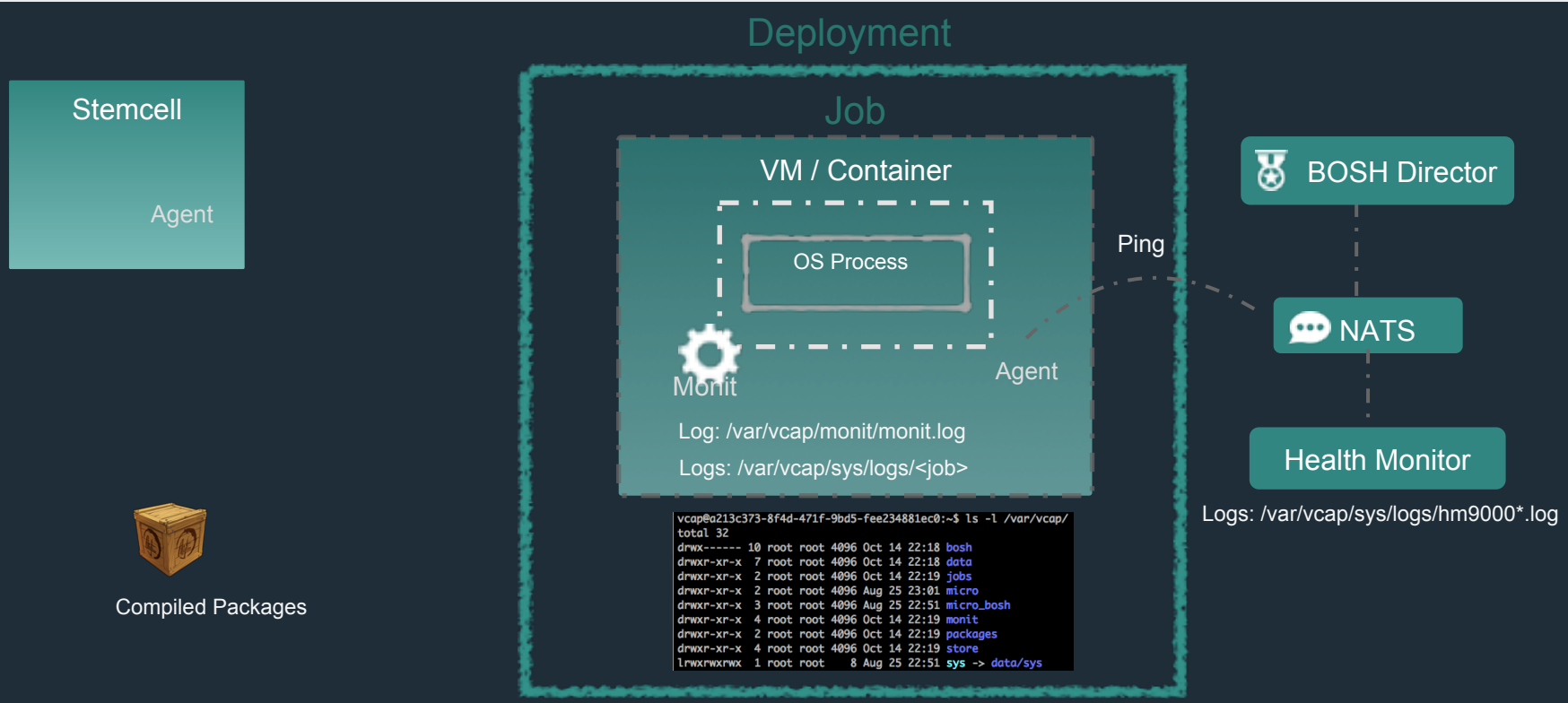
Pivotal

Job Creation

Compiled Packages

Pivotal

Logs



Pivotal

BOSH AND HIGH AVAILABILITY

4 Layers of Built-in High Availability

Application Instance

Platform Processes

Platform VMs

Availability Zones

Pivotal

4 Layers of Built-in High Availability

Application Instance

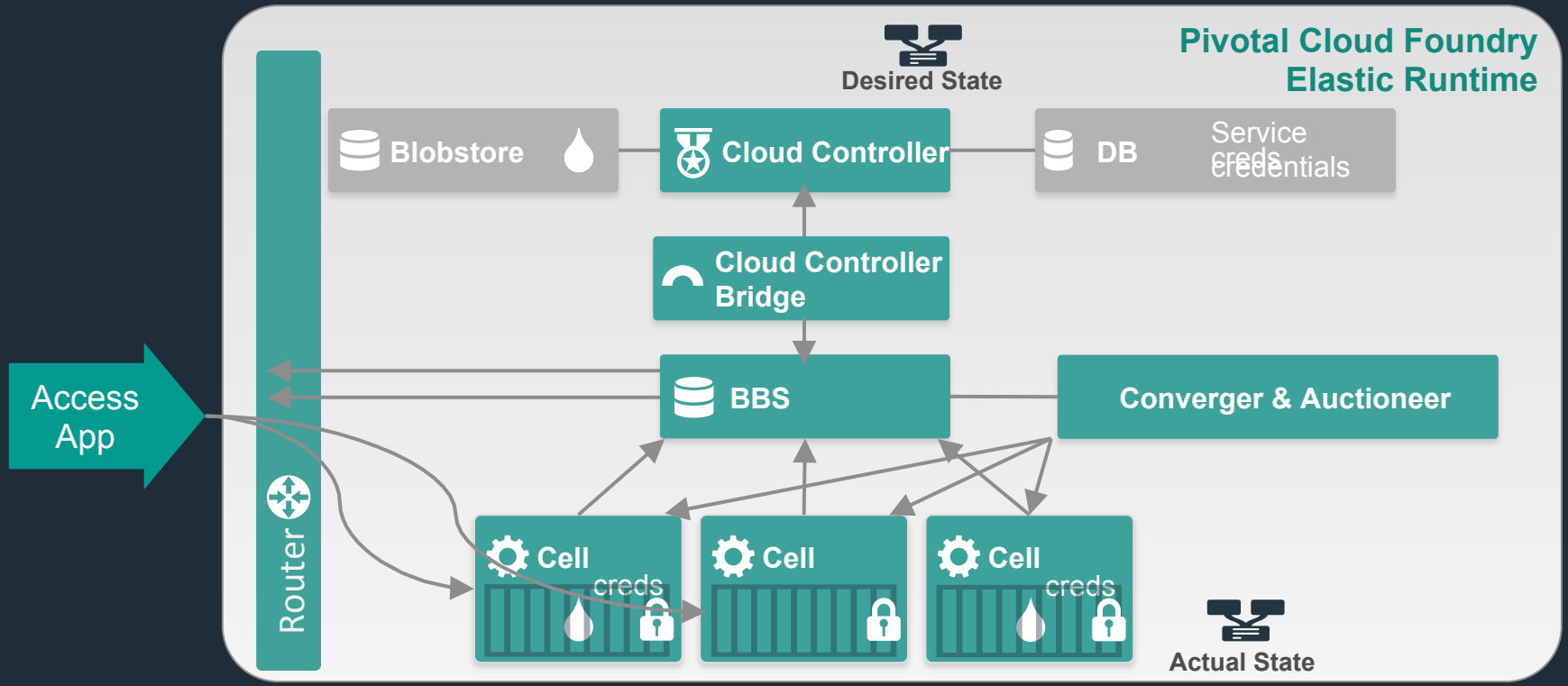
Platform Processes

Platform VMs

Availability Zones

Pivotal

Application Instance HA



Pivotal

4 Layers of Built-in High Availability

Application Instance

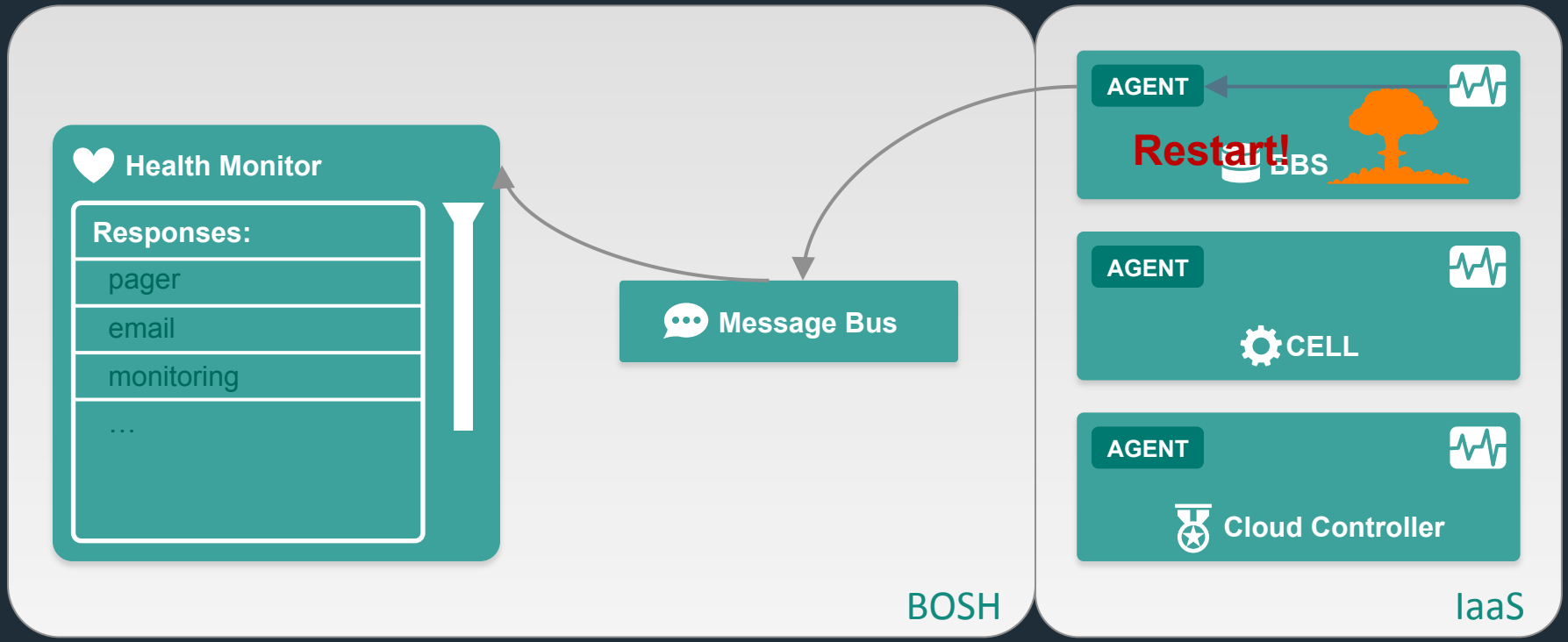
Platform Processes

Platform VMs

Availability Zones

Pivotal

Platform Process HA



Pivotal

4 Layers of Built-in High Availability

Application Instance

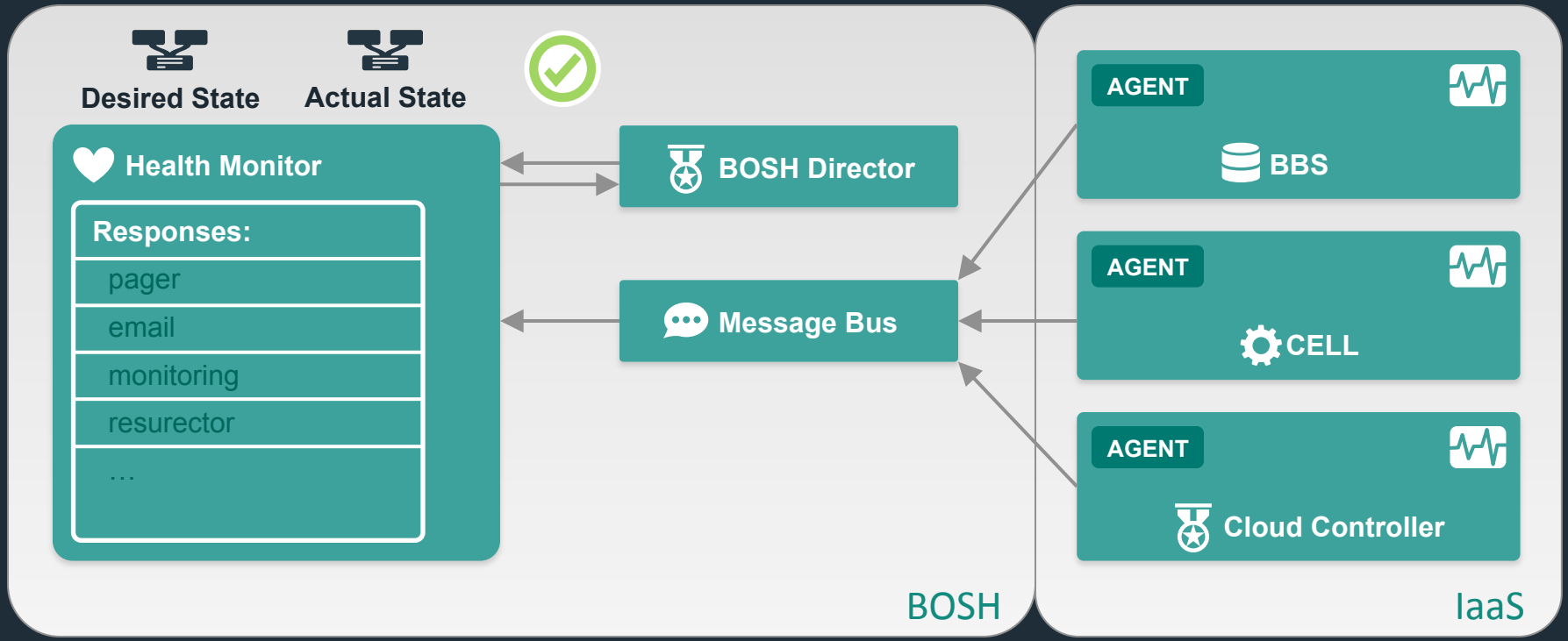
Platform Processes

Platform VMs

Availability Zones

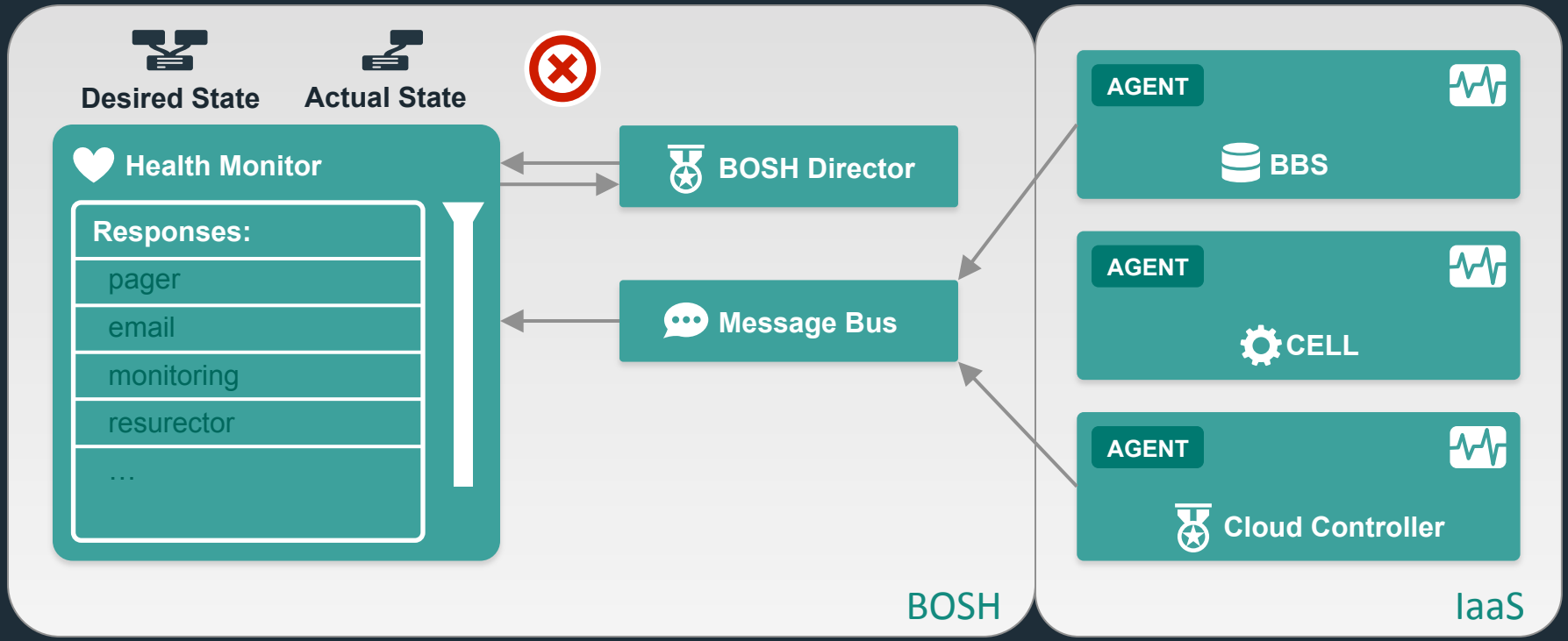
Pivotal

Platform VM HA



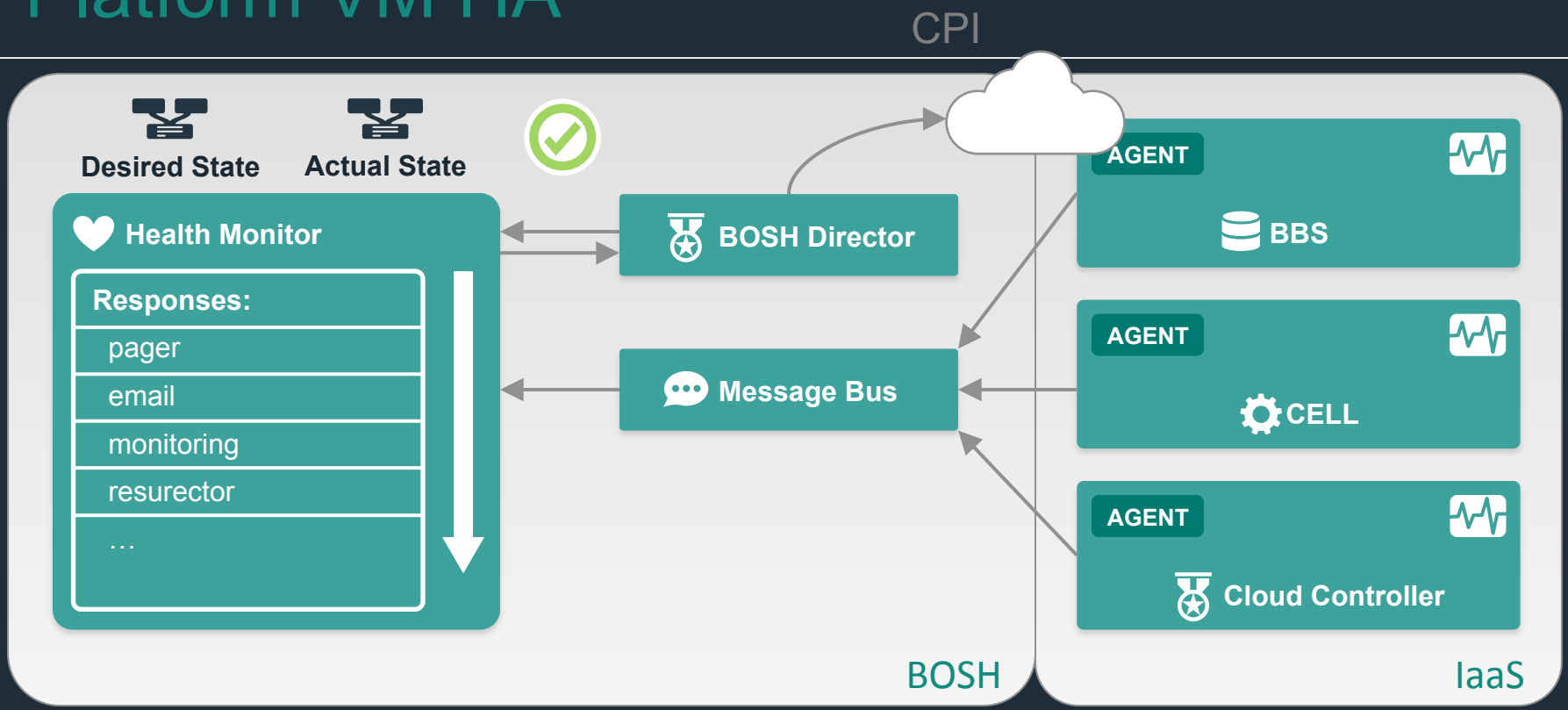
Pivotal

Platform VM HA



Pivotal

Platform VM HA



4 Layers of built-in High Availability

Application Instance

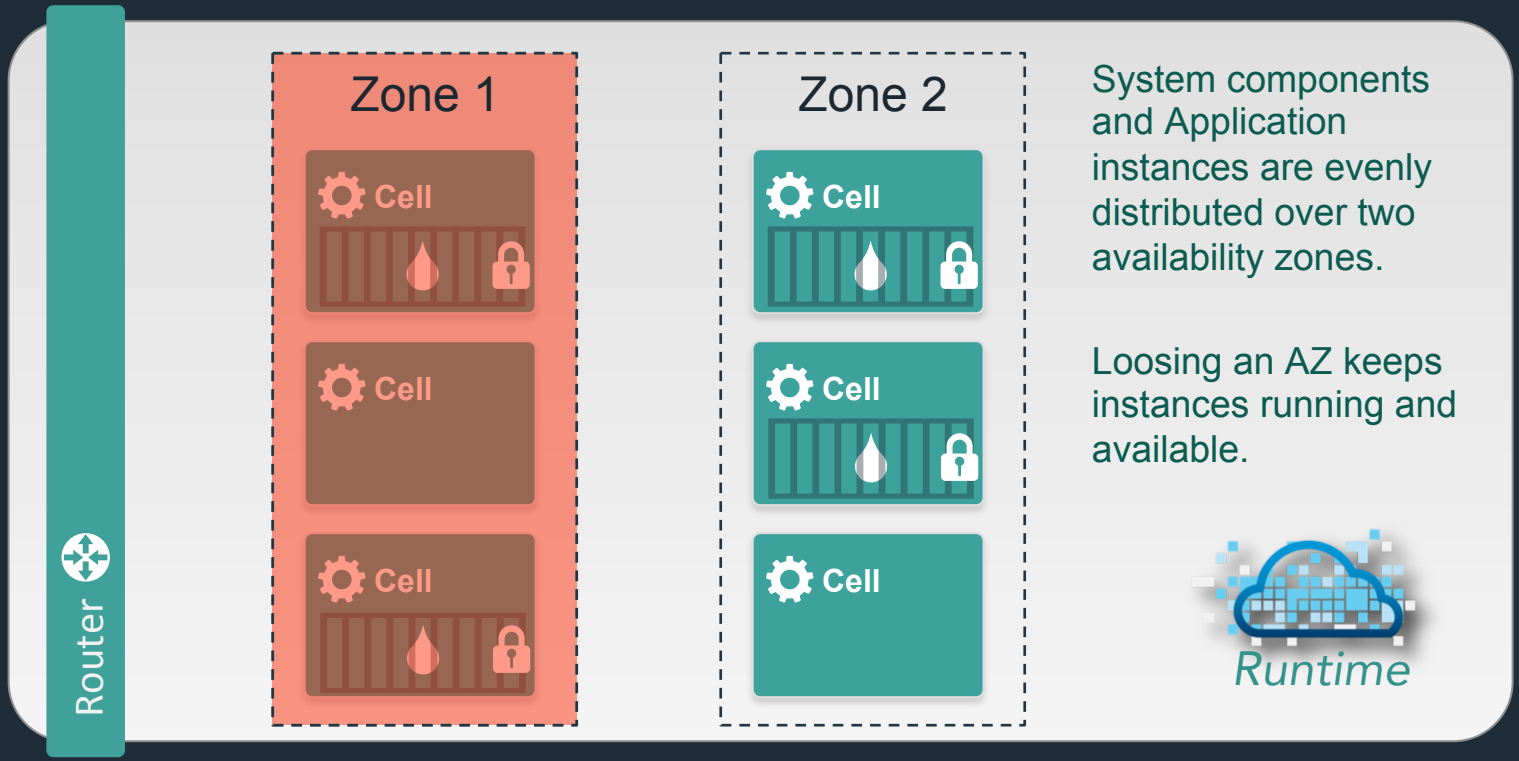
Platform Processes

Platform VMs

Availability Zones

Pivotal

Availability Zones



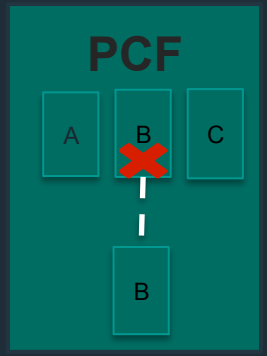
System components and Application instances are evenly distributed over two availability zones.

Loosing an AZ keeps instances running and available.

Runtime

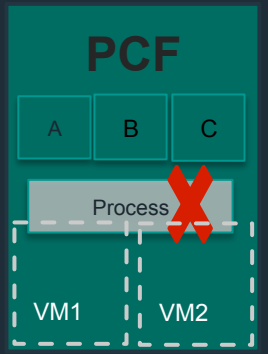
Pivotal

How Pivotal CF enables four layers of HA



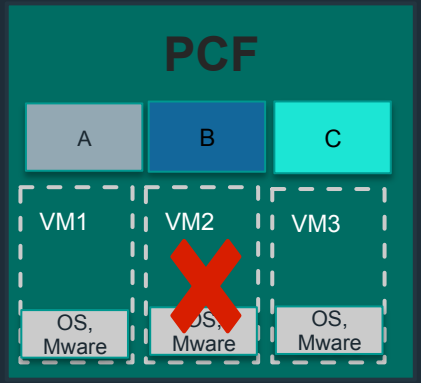
If an **app fails,** PCF reboots app in a new container.

App Fail



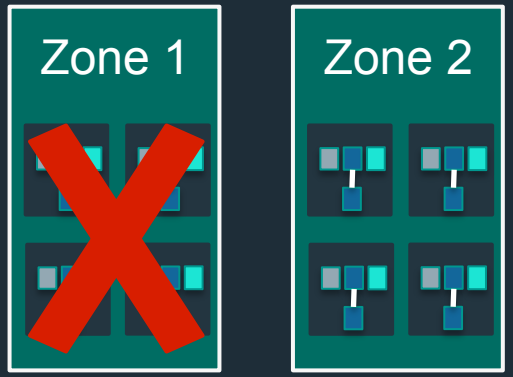
If a **process fails,** PCF restarts the process

Process Fail



If an **OS or network failure occurs** PCF kills the VM and reboots the host in a new Virtual machine.

VM Fail



If **a datacenter rack fails**, PCF ensures applications stay running in multiple availability zones

Rack Fail

Pivotal

BOSH-BASED PLATFORM UPDATES

Canary Deployments

```
update:
  canaries: 1
  canary_watch_time: 30000-300000
  max_errors: 2
  max_in_flight: 1
  update_watch_time: 30000-300000
```

Manifest

Pivotal

How Do Canary Deployments Work?

```
33 update:
34   canaries: 1
35   canary_watch_time: 30000-300000
36   max_errors: 2
37   max_in_flight: 1
38   update_watch_time: 30000-300000
```

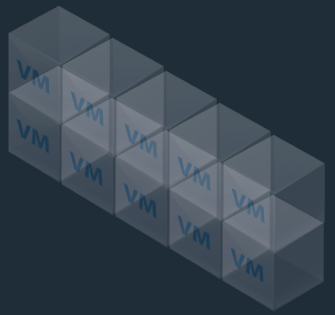
- `canaries` [Integer, required] Number of canary instances. Canary instances are being updated before other instances and any update error for canary instance means the deployment should stop. This prevents a buggy package or job from taking over all job instances, as only canaries will be affected by a problematic code. After canaries are done, other instances of this job will be updated in parallel (respecting `max_in_flight` setting).
- `canary_watch_time` [Range, Integer] How long to wait for canary update to declare job healthy or unhealthy. If Integer is given, director will sleep for that many seconds and check if job is healthy. If Range `lo..hi` is given it will wait for `lo` ms, see if job is healthy, and if it's not it will sleep some more, all up until `hi` ms have passed. If job is still unhealthy it will give up.
- `update_watch_time` [Range Integer]: Semantically no different from `canary_watch_time`, used for regular (non-canary) updates.
- `max_in_flight` [Integer, required] Maximum number of non-canary instance updates that can happen in parallel.

Pivotal

How Do Canary Deployments Work?

```
33 update:
34   canaries: 1
35   canary_watch_time: 30000-300000
36   max_errors: 2
37   max_in_flight: 1
38   update_watch_time: 30000-300000
```

No downtime, atomic rolling update



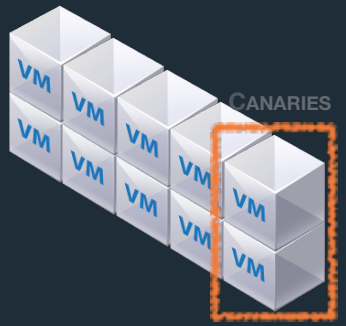
Pivotal

How Do Canary Deployments Work?

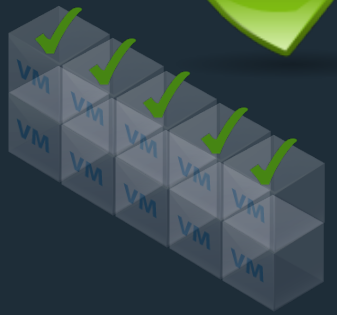
EXAMPLE:

OF CANARIES: 2

MAX IN FLIGHT: 2



v1.0



v1.1

Pivotal

How Do Canary Deployments Work?

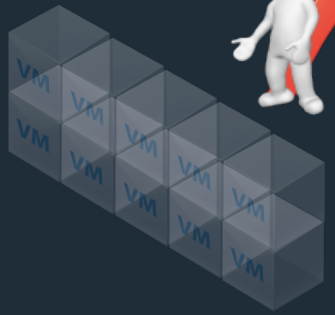
EXAMPLE:

OF CANARIES: 2

MAX IN FLIGHT: 2

v1.1

Once failed, Canary VMs are kept for troubleshooting purposes.

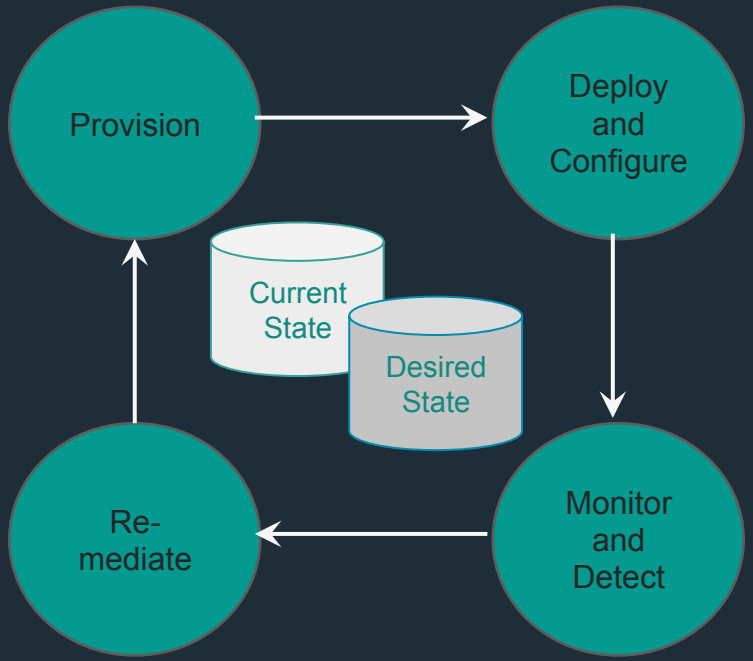


v1.2

Pivotal

BOSH – Purpose-built for “Day 2 Operations”

Consistent, Reliable, Scalable, Secure



- Checks against “desired state to return consistency
- No ad hoc automation burden
- Manage services, not servers
- 4 layers of Self Healing

Pivotal

Pivotal

Open.
Agile.
Cloud-Ready.