VMware Pivotal Labs

Tanzu Platform Automation

As a Software

Oct 2020
MinSeok Kim

Agenda

1. Why Automate?
2. What to Automate?
3. How to Automate?
4. Demo
5. Lessons Learned

vmware®

Why Automate?
What to Automate?
How to Automate?
Demo
Lessons Learned

Customer Adoption

Fortune 500 customers

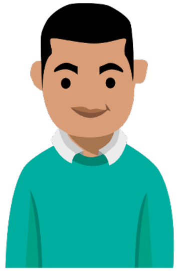
How do I code, test, get feedback, and iterate faster and safer?

Apps Engineer

T-Mobile increased frequency of changes by **20x** (versus 3-6 months to deliver new features)

How do I enable a path to production that’s faster and more resilient?

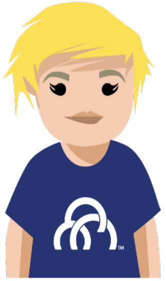
DevOps Engineer



Liberty Mutual has 2,500 daily builds and deploys **1,000 times a day** to production

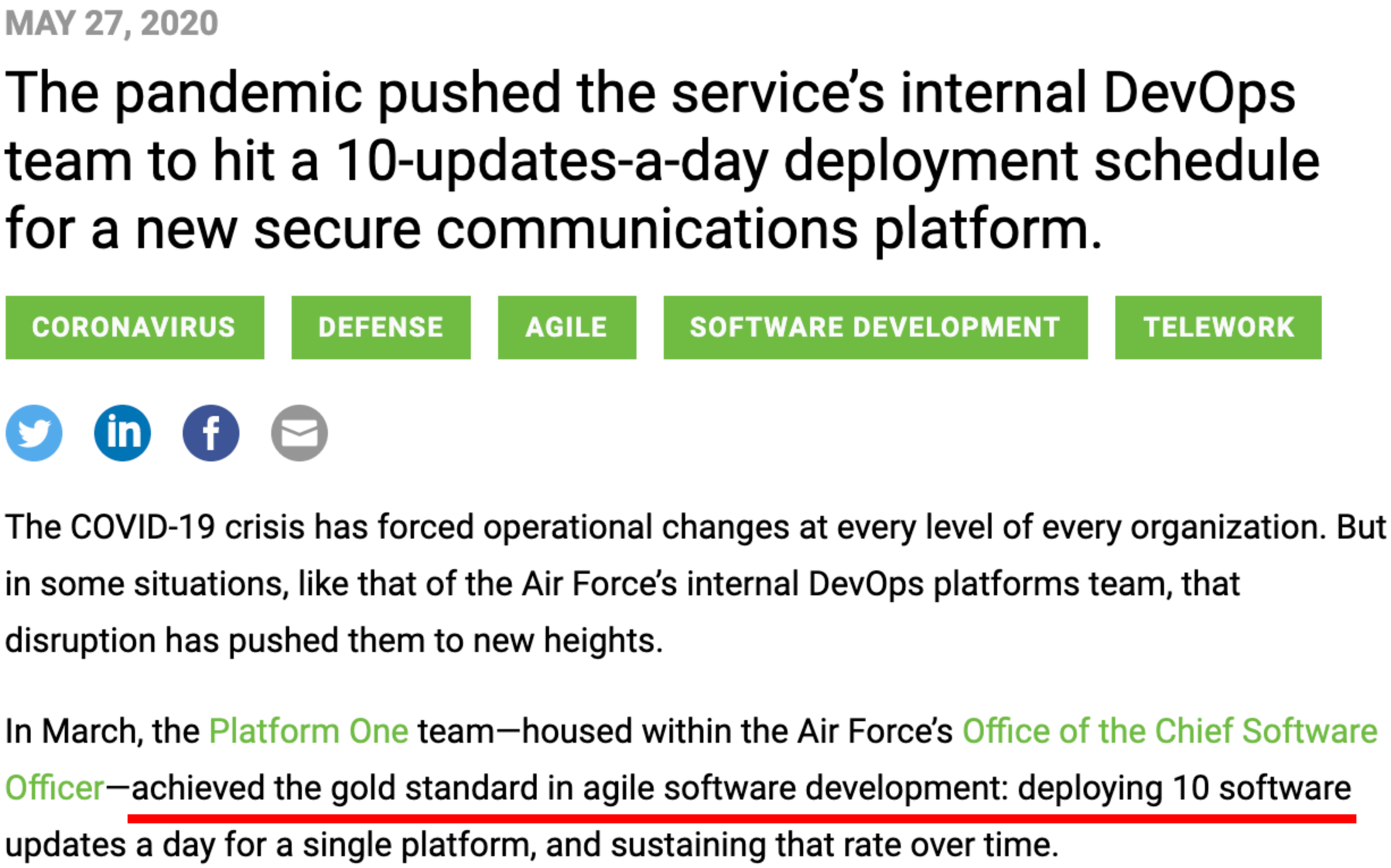
How do I keep my multiple foundations stable and secure?

Platform Engineer



Express Scripts improved patching frequency by **89%** from 45 days to 5 days

USAF Platform One Team go more Agile by Covid-19 hit



MAY 27, 2020

The pandemic pushed the service's internal DevOps team to hit a 10-updates-a-day deployment schedule for a new secure communications platform.

CORONAVIRUS DEFENSE AGILE SOFTWARE DEVELOPMENT TELEWORK

The COVID-19 crisis has forced operational changes at every level of every organization. But in some situations, like that of the Air Force's internal DevOps platforms team, that disruption has pushed them to new heights.

In March, the Platform One team—housed within the Air Force's Office of the Chief Software Officer—achieved the gold standard in agile software development: deploying 10 software updates a day for a single platform, and sustaining that rate over time.

https://www.nextgov.com/emerging-tech/2020/05/air-forces-platform-one-team-thought-it-was-agile-then-covid-19-hit/165676/

Paradigm shift with Cloud Native Era

Traditional

1. Small number of Big System
2. Dedicated system
3. Reliable Infra
4. Large changes (monthly, yearly)
5. Silo team & ticket based service

Cloud Native

1. Many small system (distributed)
2. Disposable system
3. Fragile infra
4. Small change more Frequently (Days, weeks)
5. Devops & self services & Cross Functional Team

Why automate?

- Larger Scale
- Frequent Changes
- Human Error
- Confidence

Why Automate?

What to Automate?

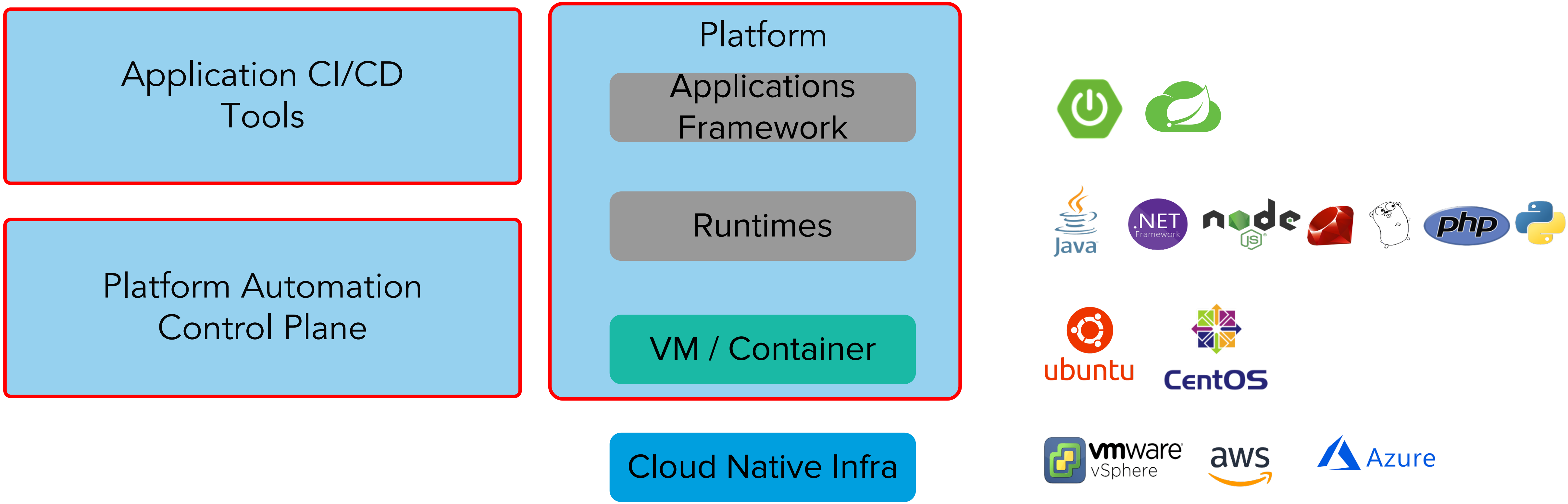
How to Automate?

Demo

Lessons Learned

Cloud Native Terminalogy

Cloud Native App & Platform & cloud Infra



Application CI/CD Tools

Platform Automation Control Plane

Platform

Applications Framework

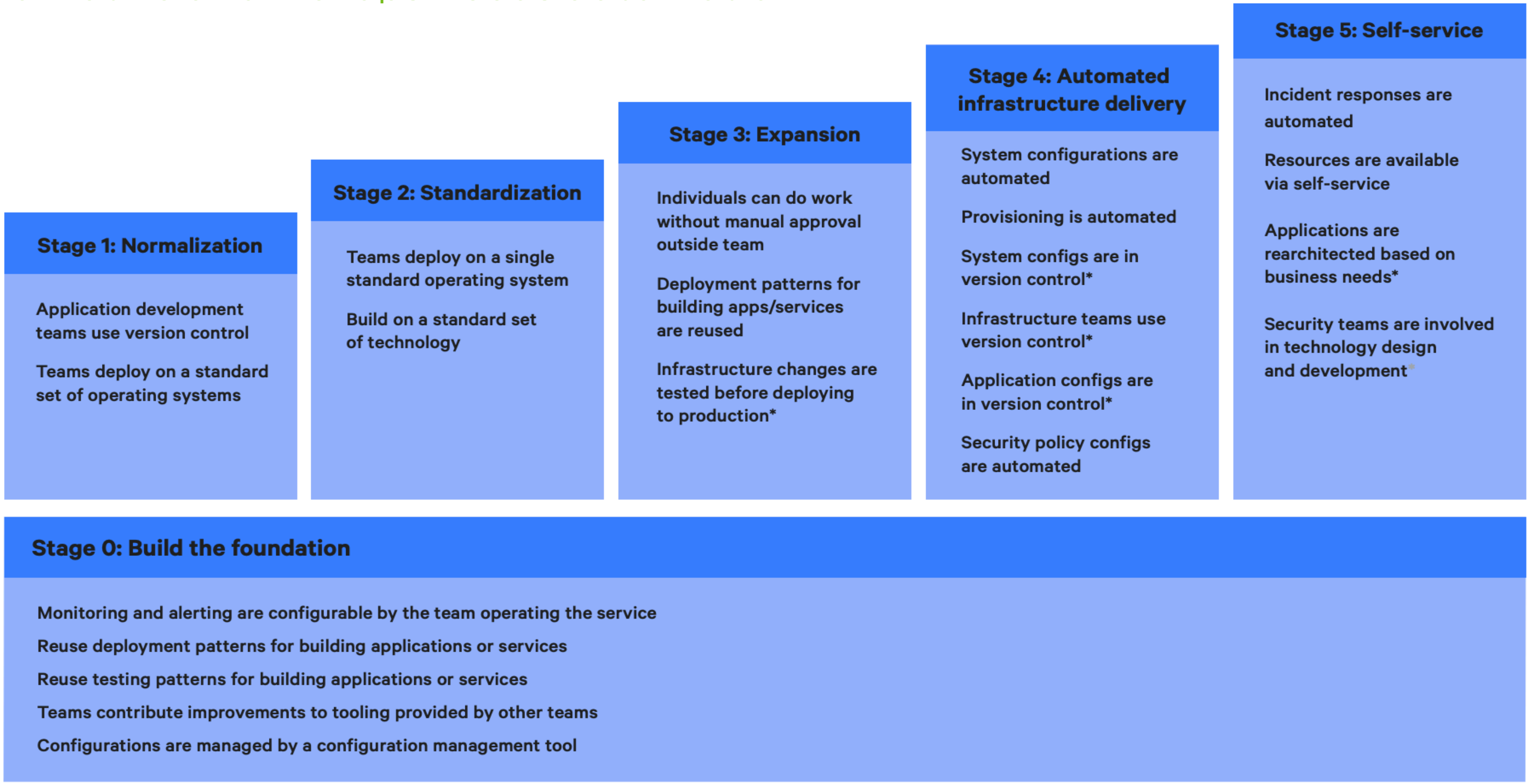
Runtimes

VM / Container

Cloud Native Infra

5 stages of DevOps Evolution

Cloud Native and Devops needs automation



Stage 1: Normalization

Application development teams use version control

Teams deploy on a standard set of operating systems

Stage 2: Standardization

Teams deploy on a single standard operating system

Build on a standard set of technology

Stage 3: Expansion

Individuals can do work without manual approval outside team

Deployment patterns for building apps/services are reused

Infrastructure changes are tested before deploying to production*

Stage 4: Automated infrastructure delivery

System configurations are automated

Provisioning is automated

System configs are in version control*

Infrastructure teams use version control*

Application configs are in version control*

Security policy configs are automated

Stage 5: Self-service

Incident responses are automated

Resources are available via self-service

Applications are rearchitected based on business needs*

Security teams are involved in technology design and development*

Stage 0: Build the foundation

Monitoring and alerting are configurable by the team operating the service

Reuse deployment patterns for building applications or services

Reuse testing patterns for building applications or services

Teams contribute improvements to tooling provided by other teams

Configurations are managed by a configuration management tool

* These practices are highly correlated with the stage

Puppet’s State of DevOps 2019 report

Typical PLATFORM Automation Areas

Cloud
Native
Platform

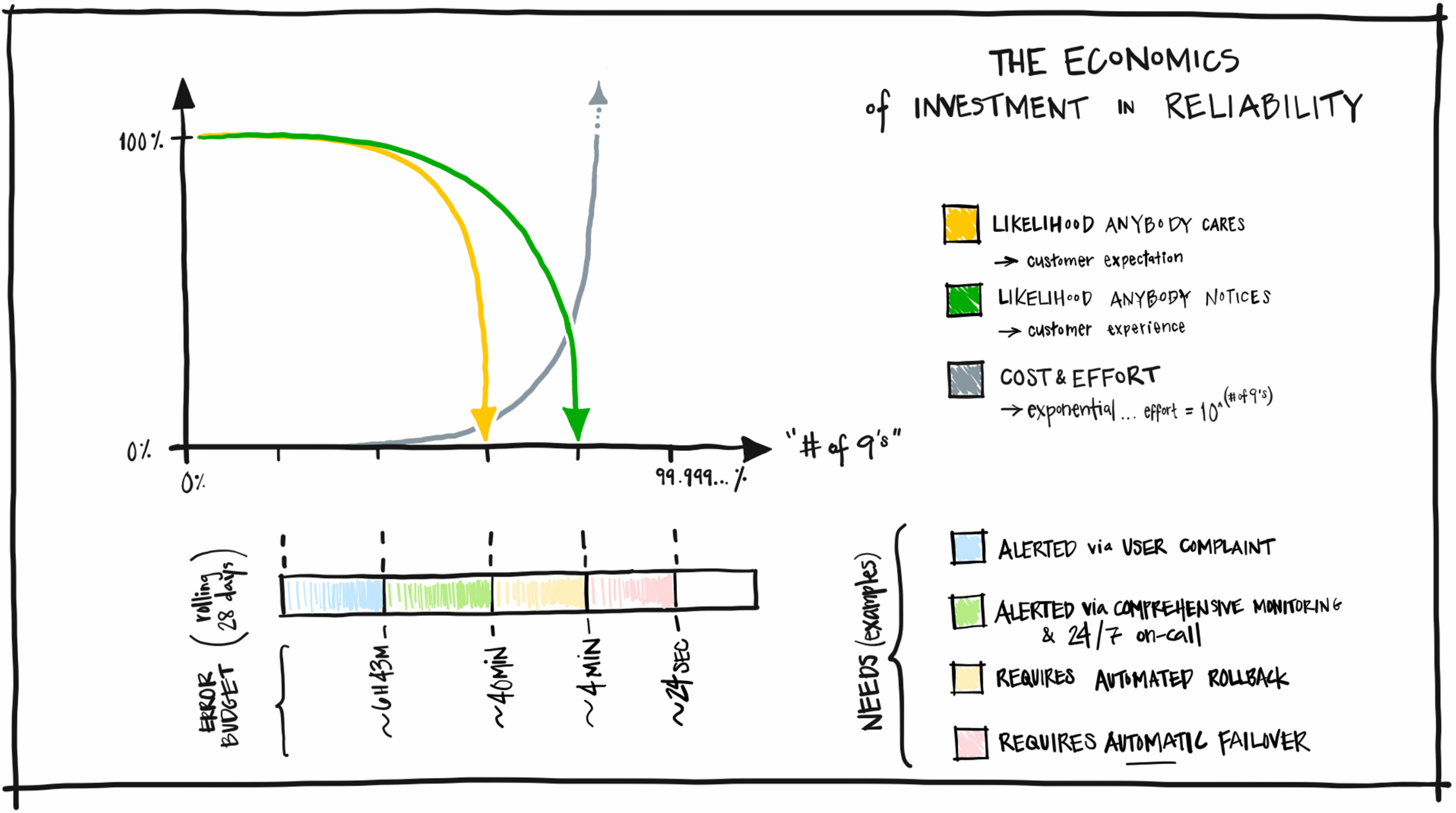
is

API based
Application

1. Provisioning/ Patching Platform(include security patch)
2. Platform Configuration with version control
3. Monitoring and Alerting
4. Scaling
5. Automatic Failover and Failback
6. Self-service(Devops)

SRE practice /Error budget

SLO	Error Budget (per 28 Days)
99%	< 7 hours
99.5%	< 3.5 hours
99.9%	~ 40 minutes
99.95%	~ 20 minutes
99.99%	~ 4 minutes
99.999%	~ 24 seconds



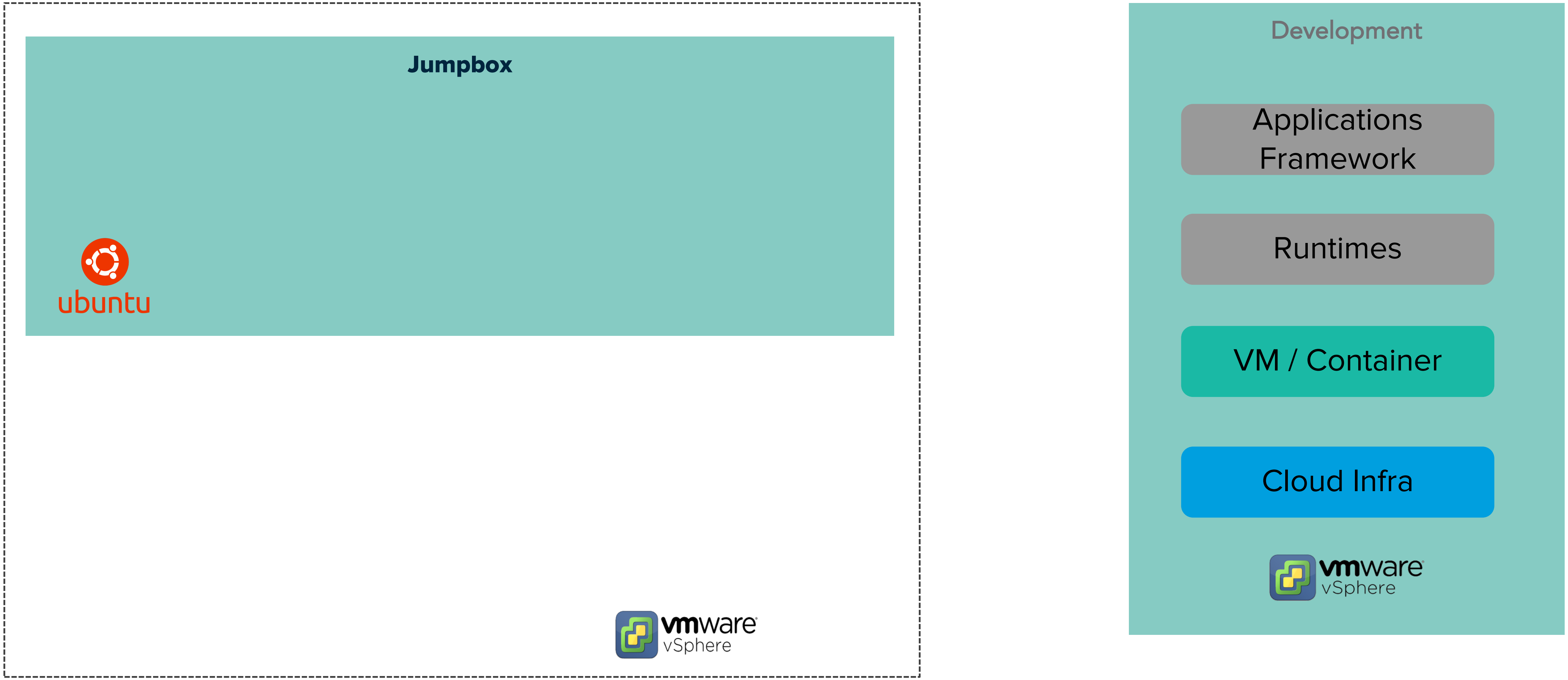
Why Automate?
What to Automate?
How to Automate?
Demo
Lessons Learned

Designing Platform Automation Architeture

Basic

Platform Automation Control Plane

Target Platform



Jumpbox

ubuntu

vmware vSphere

Development

Applications Framework

Runtimes

VM / Container

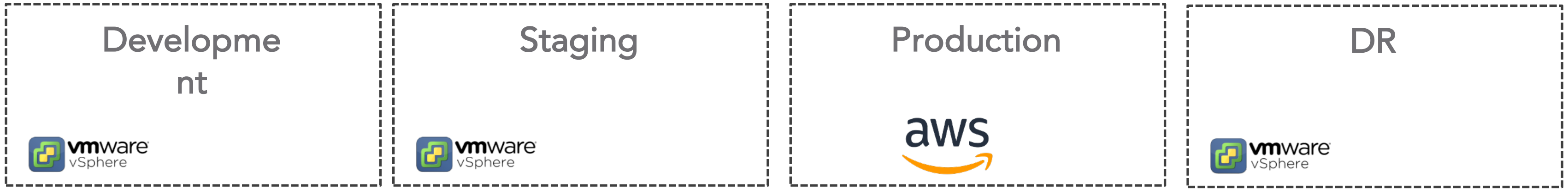
Cloud Infra

vmware vSphere

Typical Platform Operation Problems

1. Human error
2. Slow at scale
3. Slow to fix, Need human education
4. System Parity problem

1. Consistent config at scale
2. scalable
3. Easy and faster to fix



Development

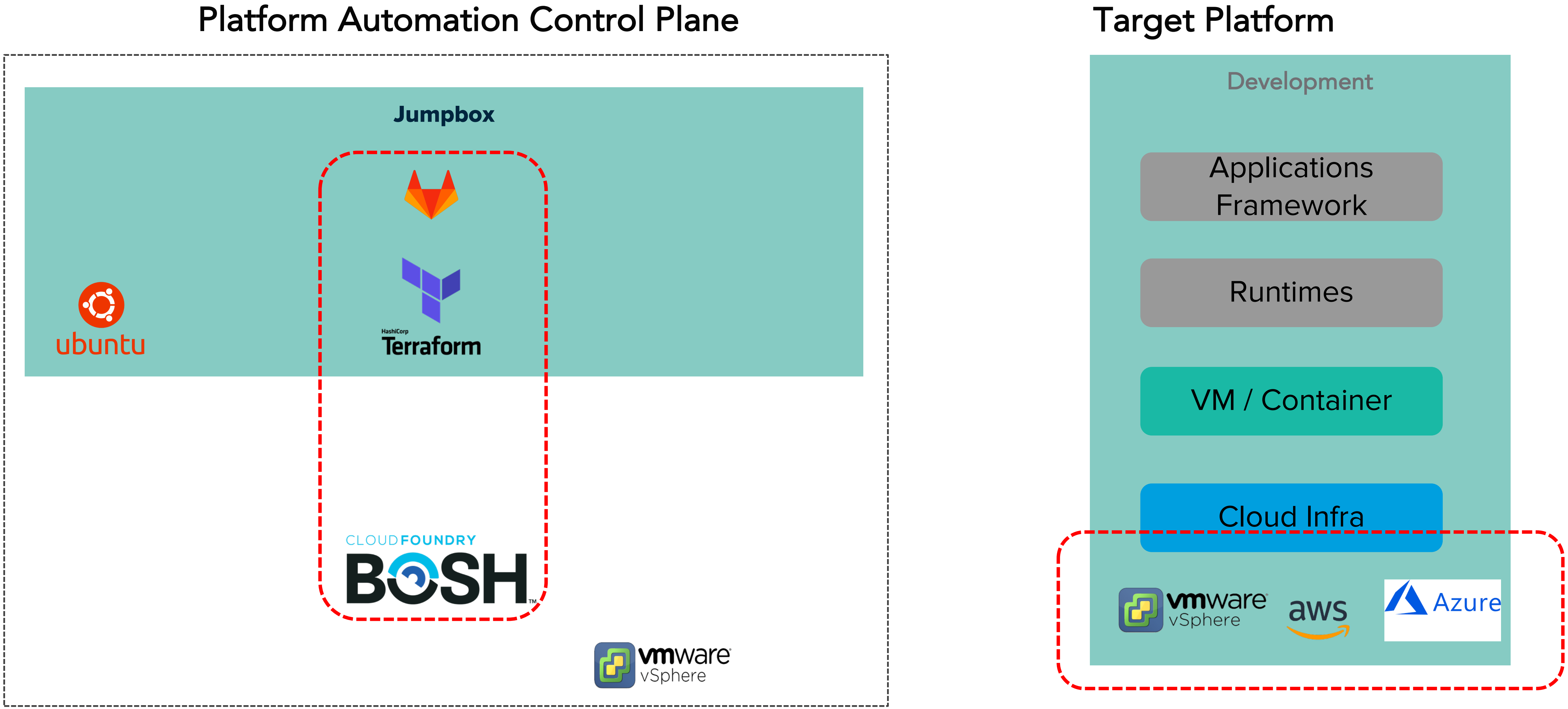
Staging

Production

DR

Designing Platform Automation Architeture

Adopting tools, multi cloud



Target Platform Requirements

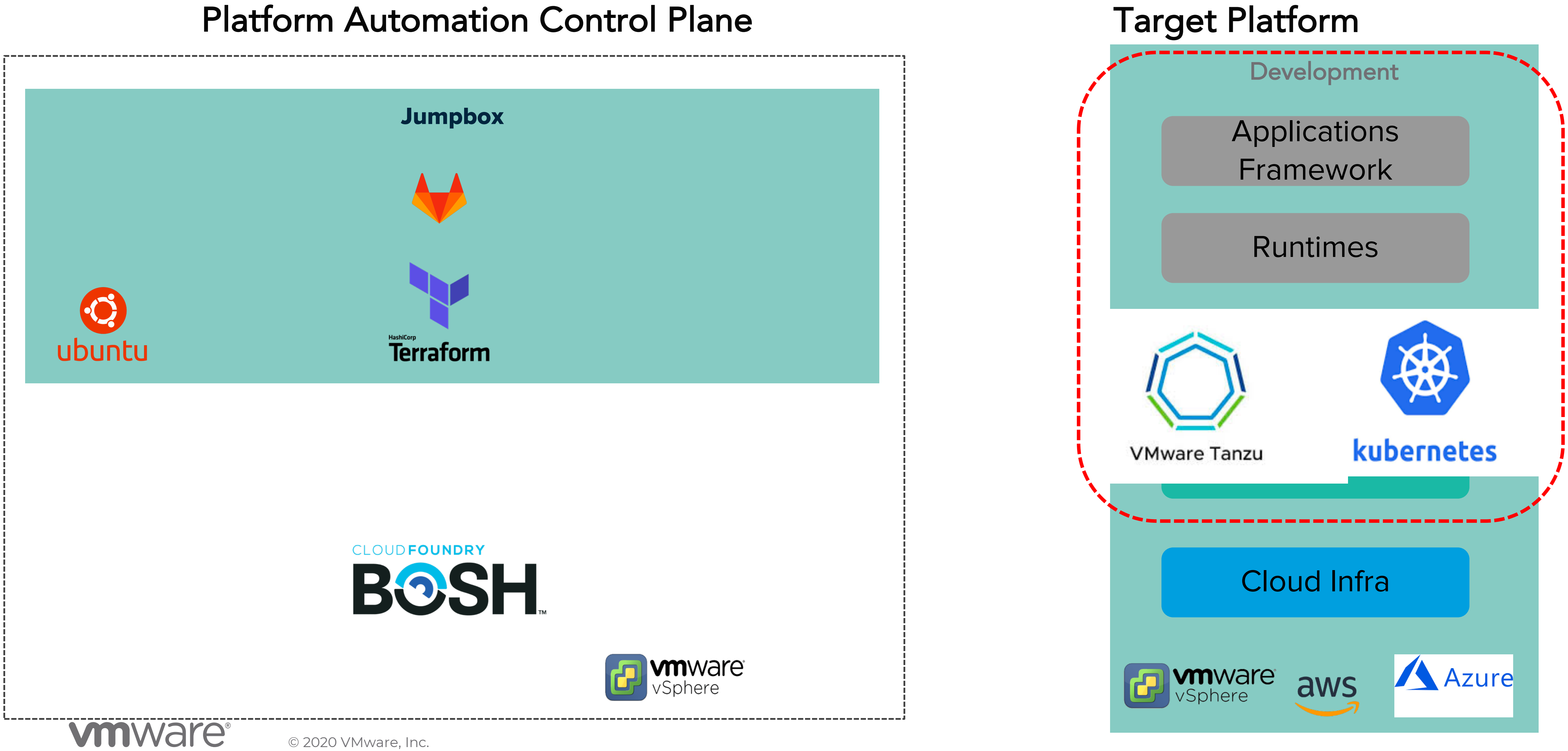
1. Automatic Platform Rolling Upgrade
2. Monitoring
3. Scaling
4. Automatic Platform Failover Handling And healing
5. Automatic App Failover Handling
6. API based Platform

Cloud Native Platform

Cloud Native App

Designing Platform Automation Architeture

Adopting Cloud Native Platform



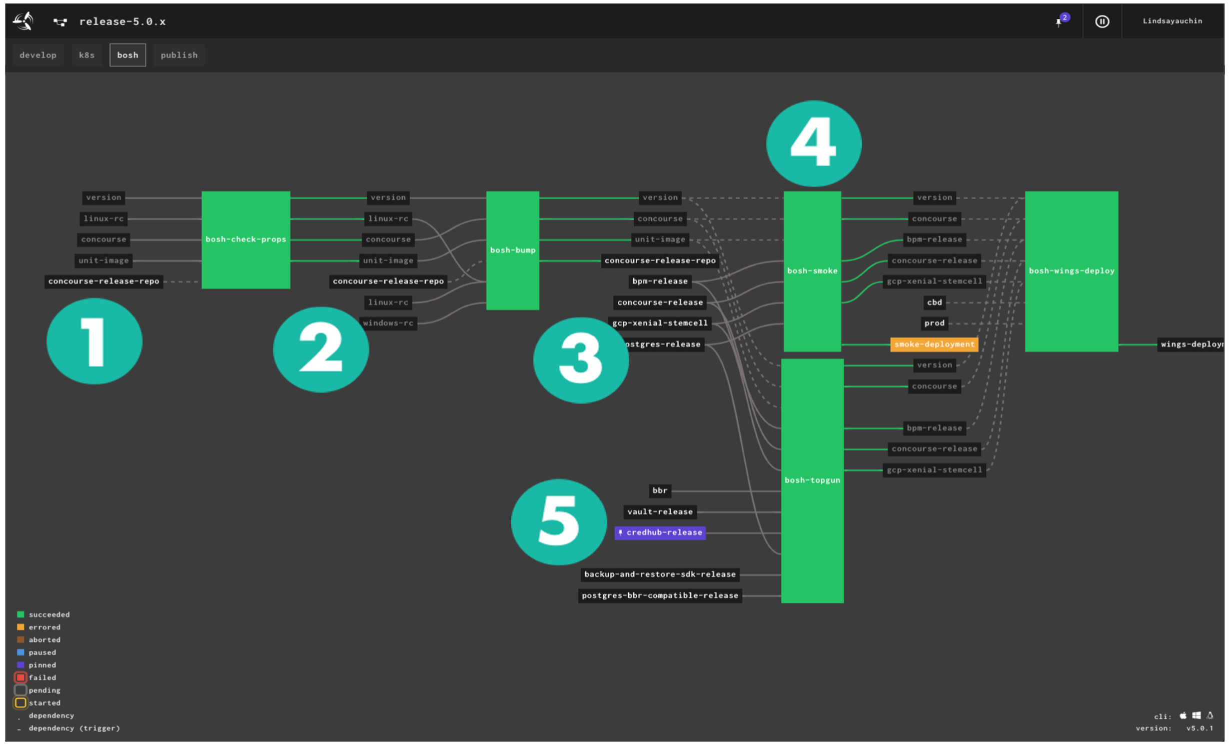
Platform Automation Tool Problems

1. Difficult to control and trace input/output versioning
2. Conflict dependency and config on runtime
3. Workload Scalability
4. slow time to recover pipeline, difficult to automate.
5. Pipeline migration cost to adopt multi-cloud
6. Exception handling
7. Automatic Platform Failover Handling

Concourse – new automation tools

Concourse moves to realize the conceptual delivery model in visual pipelines!

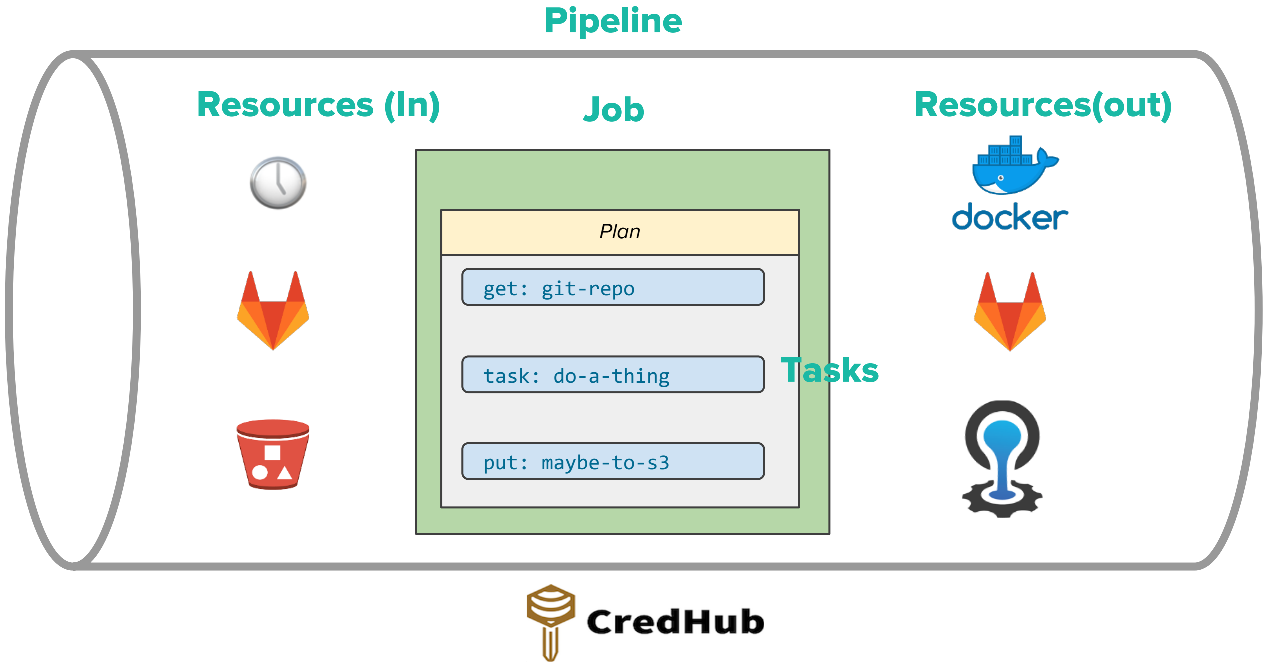
1. Versioned input, versioned output
2. All pipeline by code
3. Containerized runtime
4. Scalable CI system
5. Strong Abstractions and fewer things to learn



https://ci.concourse-ci.org/teams/main/pipelines/concourse

Concourse Concepts

1. Strong Abstractions and fewer things to learn

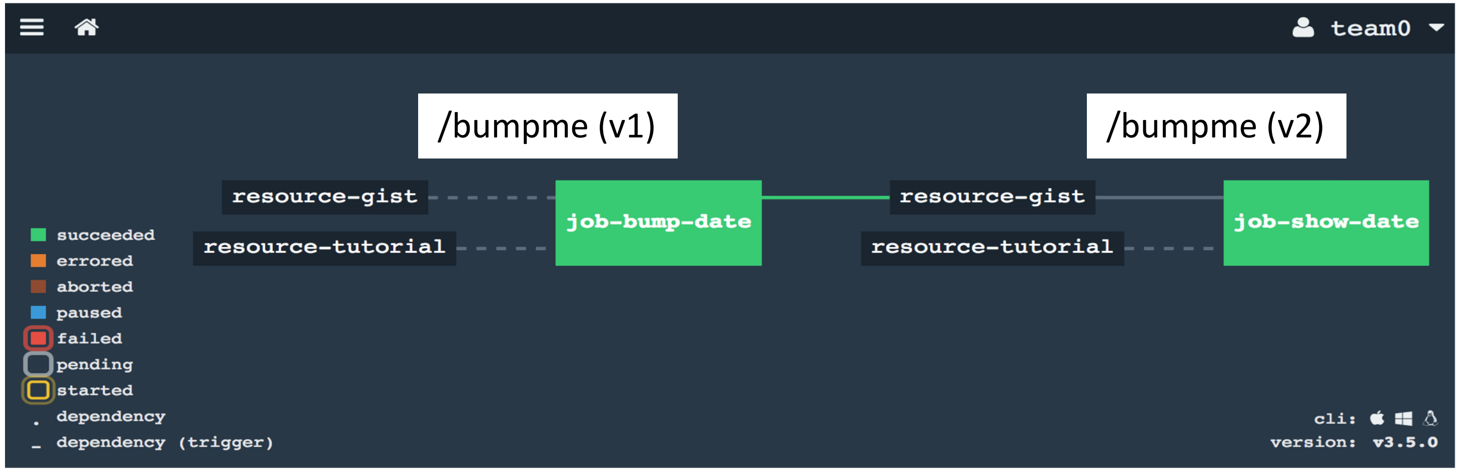


```
get: git-repo
task: do-a-thing
put: maybe-to-s3
```

100+ OSS resources to external systems and tool

Concourse Concepts

1. Versioned input, versioned output
2. All pipeline by code (Infra as a code)



Resource-gist

[< https://gist.github.com/test/resource>](https://gist.github.com/test/resource)

/bumpme (v3)
(by manual push)

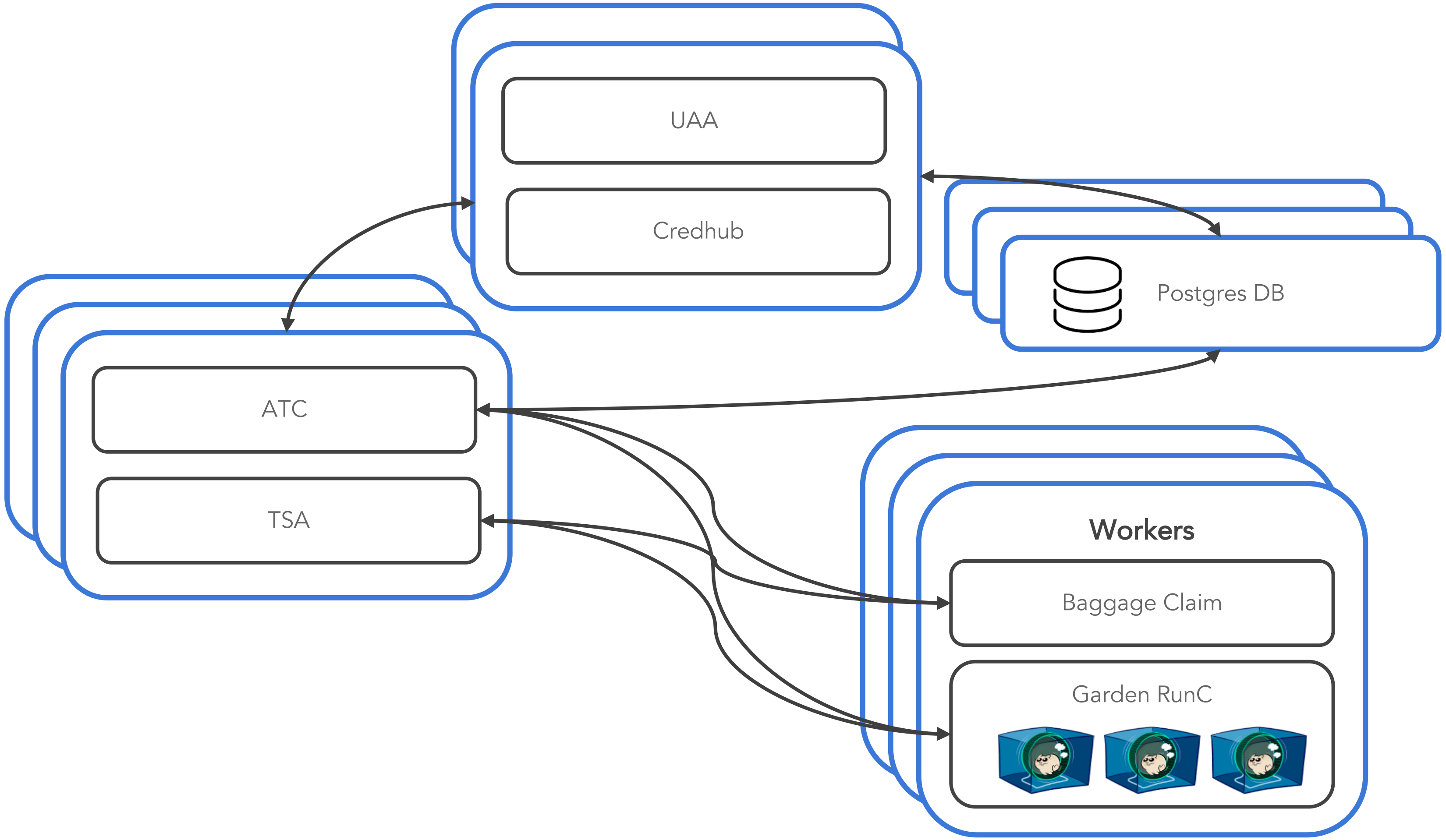
/bumpme (v2)
(by job-bump-date)

/bumpme (v1)

```
- name: job-show-date
  plan:
  - get: resource-tutorial
  - get: resource-gist
    passed: [job-bump-date]
    trigger: true
  - task: show-date
    config:
      platform: linux
      image_resource:
        type: docker-image
        source: {repository: busybox}
      inputs:
        - name: resource-gist
      run:
        path: cat
        args: [resource-gist/bumpme]
```

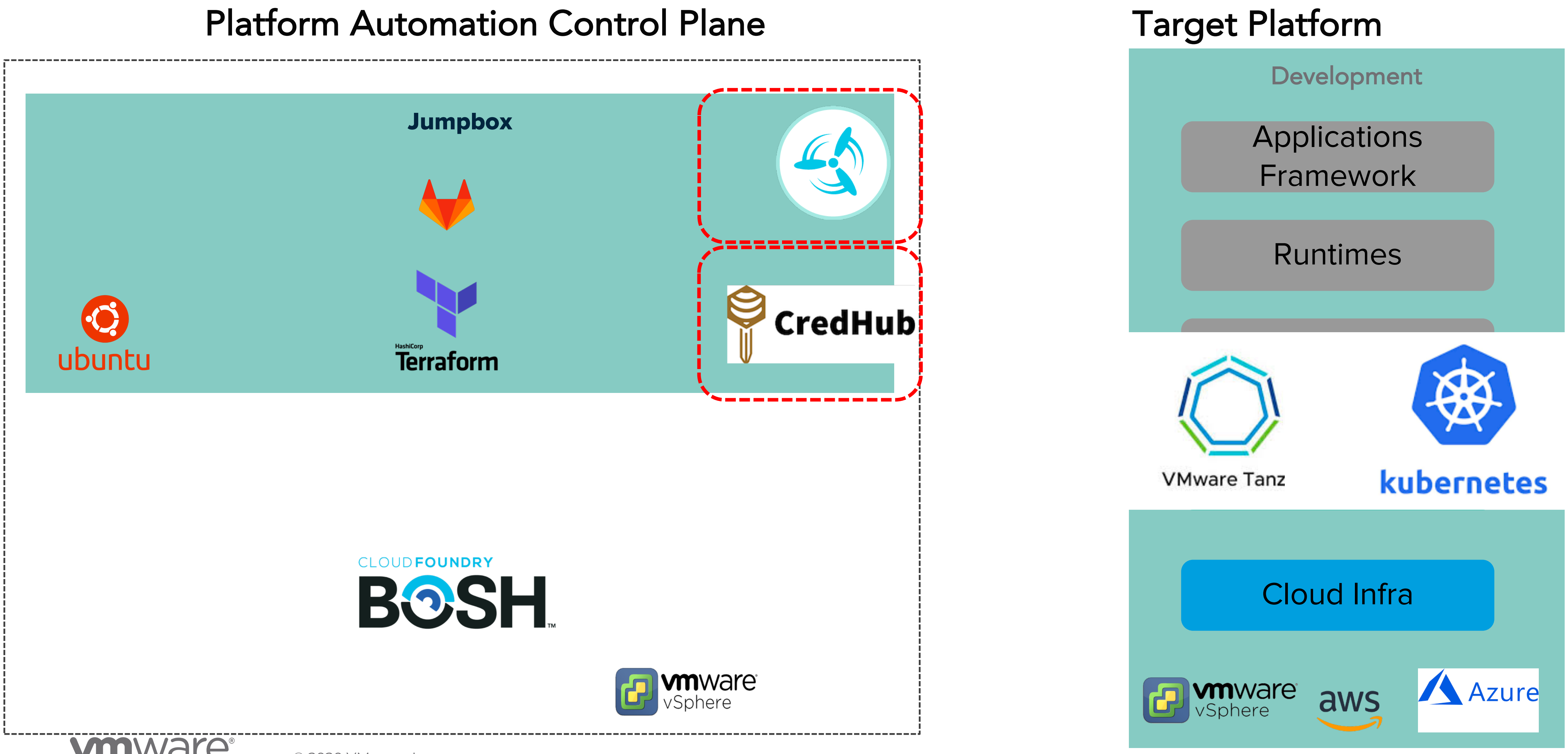
Concourse Concepts

1. Containerized runtime
2. Scalable CI system



Designing Platform Automation Architeture

Concourse



Platform Automation Control Plane

Jumpbox

Target Platform

Development

Applications Framework

Runtimes

Cloud Infra

Platform Automation Toolkit

Platform Automation for PCF provides the **building blocks** for teams to create a **repeatable and reusable automated pipeline** for **upgrading and installing** their **PCF** foundations.

“***perpetual upgrade machine***”

Platform Automation Toolkit

Composable, Reusable Tasks

apply-changes

apply-director-changes

assign-stemcell

configure-authentication

configure-director

configure-ldap-authentication

configure-product

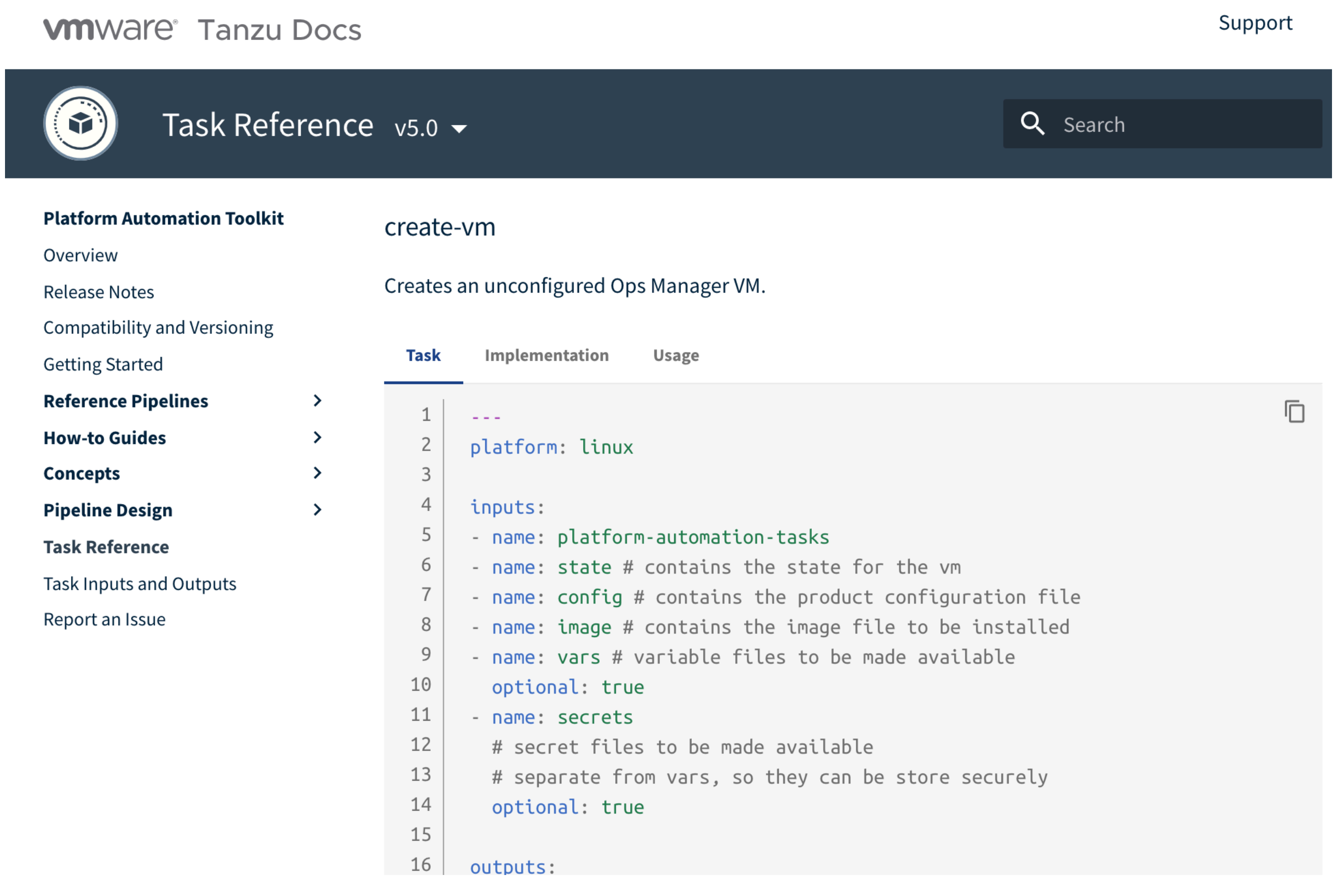
configure-saml-authentication

create-vm

credhub-interpolate

delete-installation

delete-vm



create-vm

Creates an unconfigured Ops Manager VM.

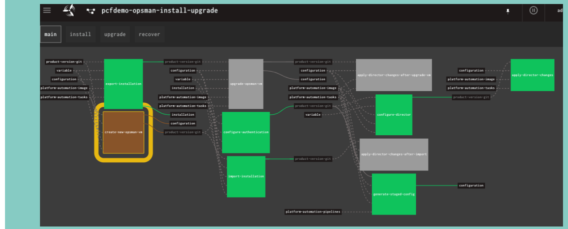
```yaml
---
platform: linux

inputs:
- name: platform-automation-tasks
- name: state # contains the state for the vm
- name: config # contains the product configuration file
- name: image # contains the image file to be installed
- name: vars # variable files to be made available
  optional: true
- name: secrets
  # secret files to be made available
  # separate from vars, so they can be store securely
  optional: true

outputs:
```

Designing Platform Automation Architeture (Final)

Infra as a software: Eventually Consistency, Idempotent

Platform Automation Control Plane

Platform Automation Tool



Platform Automation ToolKit

HashiCorp Terraform

Version Control

Object Storage

MINIO

Security

CredHub

Docker Registry

Jumpbox

ubuntu

CLOUD FOUNDRY BOSH™

vmware vSphere aws Azure

Target Platform

Sandbox Platform

vmware vSphere aws Azure

Dev Platform

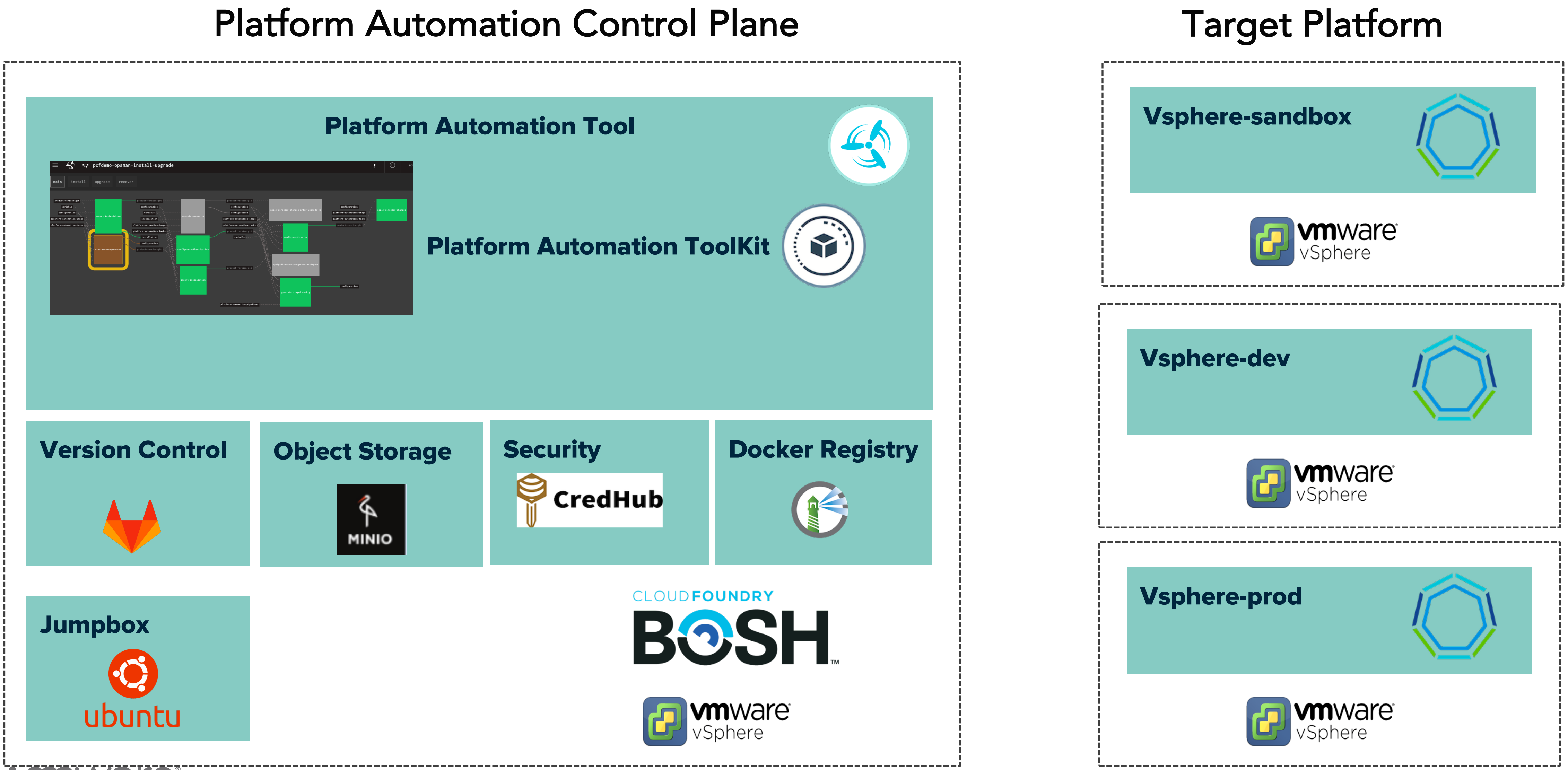
vmware vSphere aws Azure

Prod Platform

vmware vSphere aws Azure

Why Automate?
What to Automate?
How to Automate?
Demo
Lessons Learned

Demo - Platform Automation Architecture



Platform Automation Control Plane

Platform Automation Tool

Platform Automation ToolKit

Version Control

Object Storage

Security

Docker Registry

Jumpbox

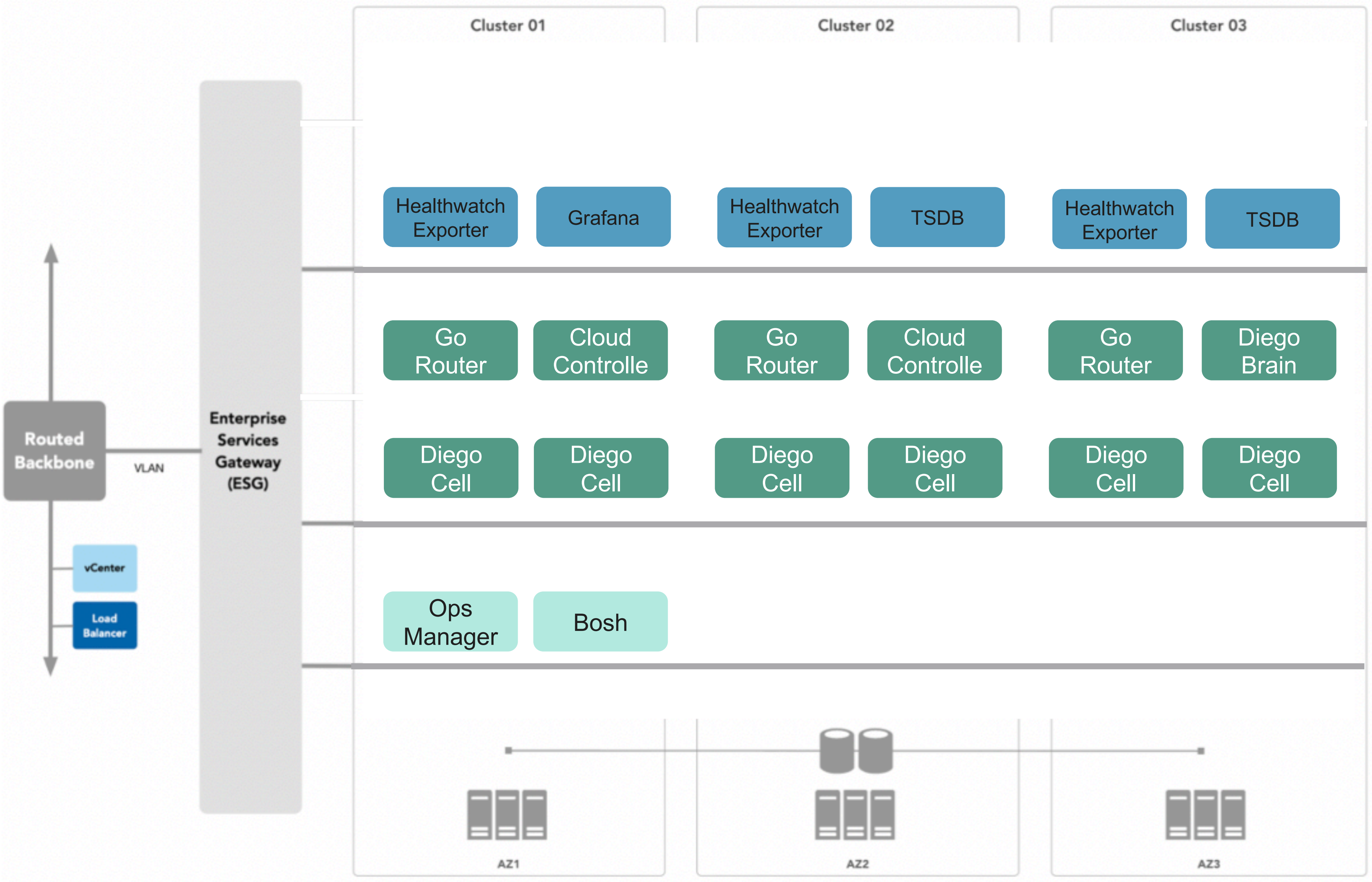
Target Platform

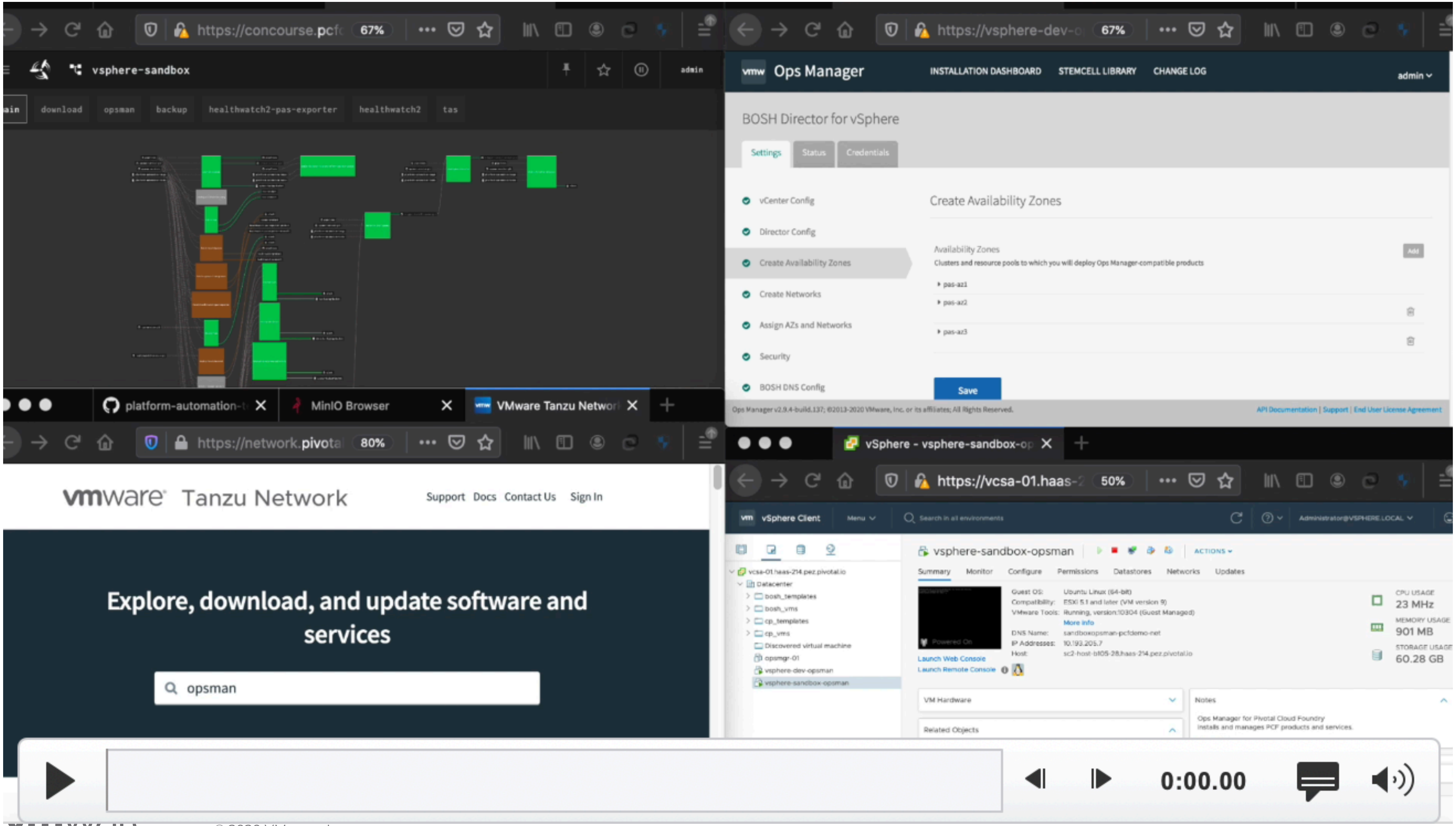
Vsphere-sandbox

Vsphere-dev

Vsphere-prod

Demo - Target Platform Architeture(Tanzu Application Service)





Ops Manager INSTALLATION DASHBOARD STEMCELL LIBRARY CHANGE LOG admin

BOSH Director for vSphere

Settings Status Credentials

- vCenter Config
- Director Config
- Create Availability Zones
- Create Networks
- Assign AZs and Networks
- Security
- BOSH DNS Config

Create Availability Zones

Availability Zones Add

Clusters and resource pools to which you will deploy Ops Manager-compatible products

- pas-az1
- pas-az2
- pas-az3

Save

Ops Manager v2.9.4-build.137; ©2013-2020 VMware, Inc. or its affiliates; All Rights Reserved. API Documentation | Support | End User License Agreement

vmware Tanzu Network Support Docs Contact Us Sign In

Explore, download, and update software and services

opsman

vSphere Client Menu Search in all environments Administrator@VSPHERE.LOCAL

- vcsa-01.haas-214.pez.pivotal.io
  - Datacenter
    - bosh_templates
    - bosh_vms
    - cp_templates
    - cp_vms
    - Discovered virtual machine
    - opsmgr-01
    - vsphere-dev-opsman
    - vsphere-sandbox-opsman

vsphere-sandbox-opsman ACTIONS

Summary Monitor Configure Permissions Datastores Networks Updates

Powered On

Guest OS: Ubuntu Linux (64-bit)
Compatibility: ESXi 5.1 and later (VM version 9)
VMware Tools: Running, version:10304 (Guest Managed)
More info
DNS Name: sandboxopsman-pcfdemo-net
IP Addresses: 10.193.205.7
Host: sc2-host-b105-28.haas-214.pez.pivotal.io

Launch Web Console
Launch Remote Console

CPU USAGE 23 MHz
MEMORY USAGE 901 MB
STORAGE USAGE 60.28 GB

VM Hardware

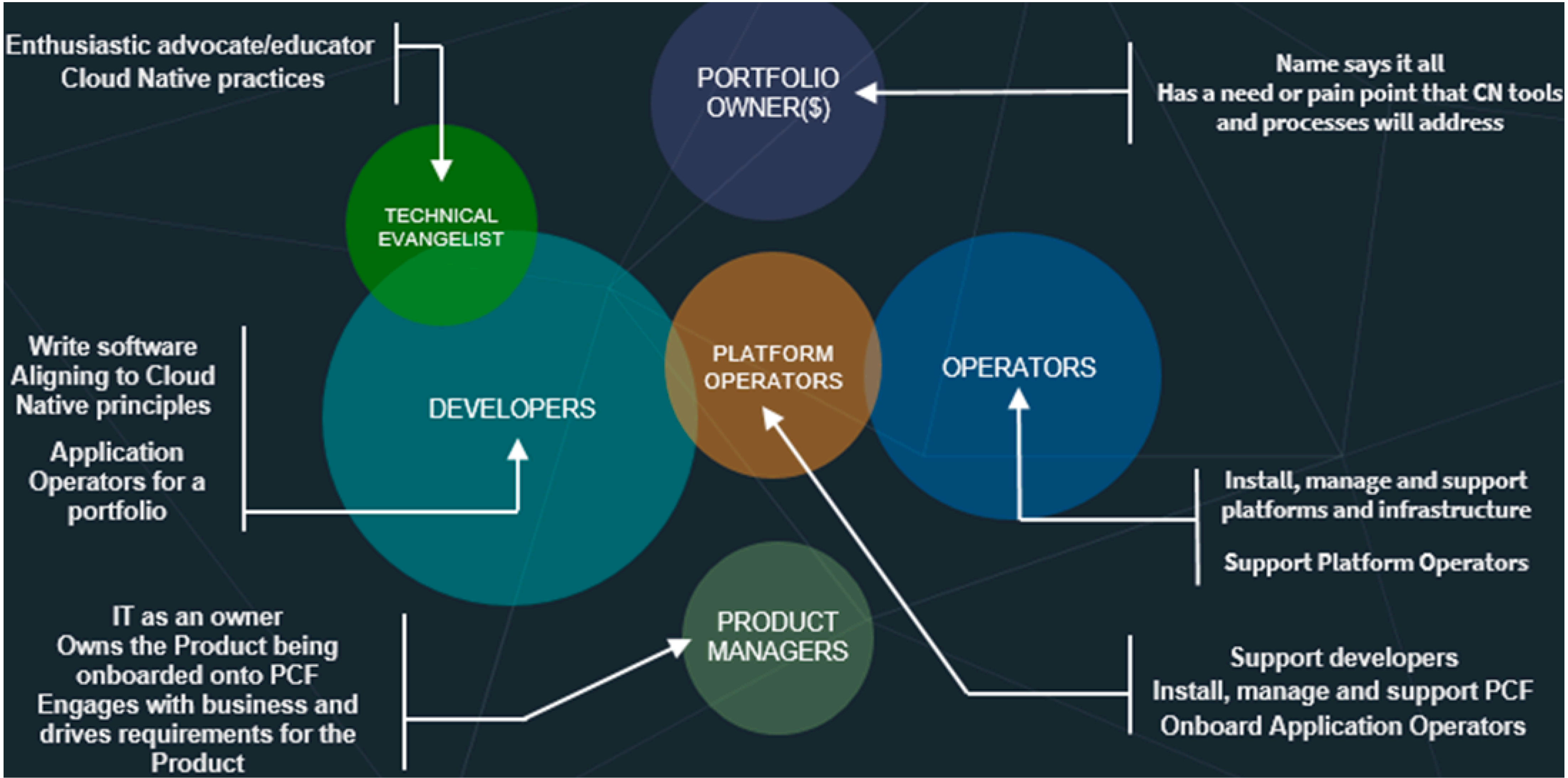
Related Objects

Notes
Ops Manager for Pivotal Cloud Foundry
installs and manages PCF products and services.

© 2020 VMware, Inc.

Why Automate?
What to Automate?
How to Automate?
Demo
Lessons Learned

Cloud Native Platform Team



Enthusiastic advocate/educator
Cloud Native practices

Name says it all
Has a need or pain point that CN tools and processes will address

Write software
Aligning to Cloud Native principles

Application Operators for a portfolio

Install, manage and support platforms and infrastructure

Support Platform Operators

IT as an owner
Owns the Product being onboarded onto PCF
Engages with business and drives requirements for the Product

Support developers
Install, manage and support PCF
Onboard Application Operators

Lessons Learned

1. **Set your Platform automation Roadmap for** your platform journey.
2. Adopt **Cloud Native Platform**
3. Design your platform automation architecture **using cloud agnostic tools**
4. Automation requires **Dedicated team** to manage.