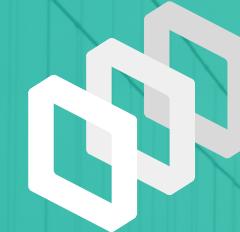




Pivotal Container Service Overview and Deep Dive



Pongsak Khamdee

Platform Architect, Pivotal

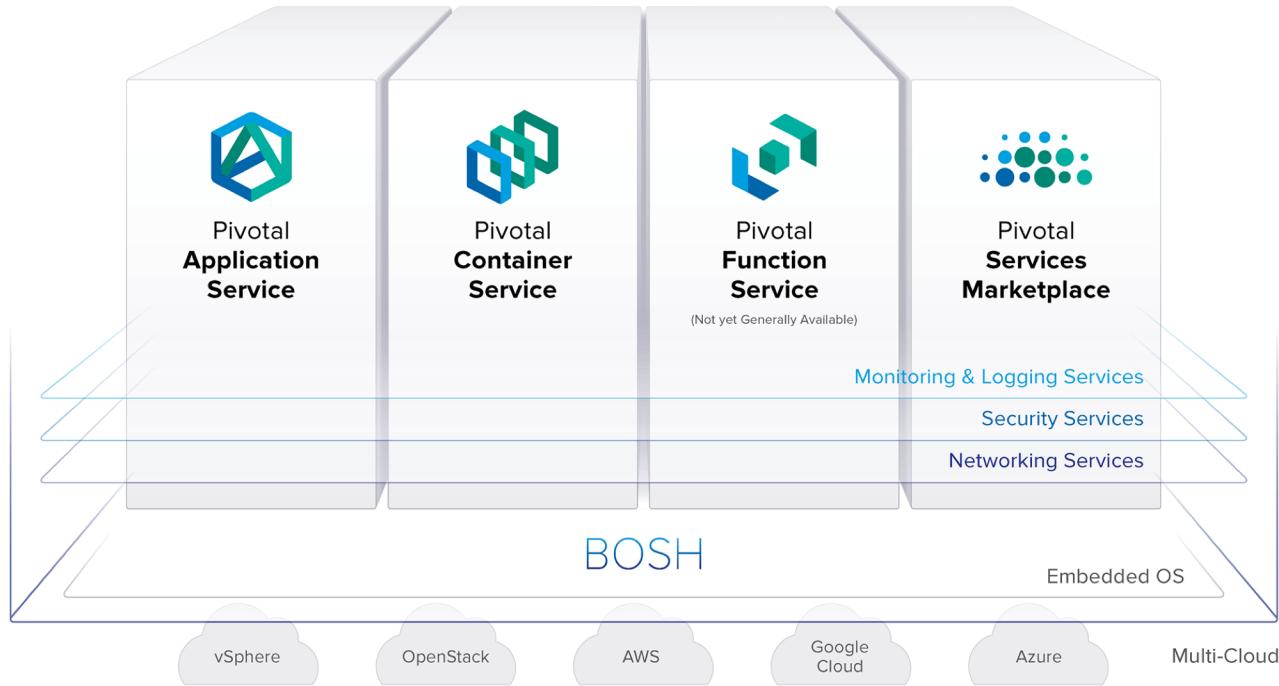
pkhamdee@pivotal.io

July 2019

Agenda

- Why PKS?
- What is PKS
- Deep Dive
- Reference Architectures
- Workloads





Main Capabilities

Why PKS?

Platform Team Delivering Real Value



Developer Productivity

- Accelerate feedback loops by improving delivery velocity
- Focus on applications, not infrastructure
- Give developers the tools and frameworks to build resilient apps



Operational Efficiency

- Employ 500:1 developer to operator ratio
- Perform zero-downtime upgrades
- Runs the same way on every public/private cloud



Comprehensive Security

- Adopt a defense-in-depth approach
- Continuously update platforms to limit threat impact
- Apply the 3 R's → repair, repave, rotate



High Availability

- Run platforms that stay online under all circumstances
- Scale up and down, in and out, through automation
- Deploy multi-cloud resilience patterns

What Workloads are Realizing these Benefits?



MICROSERVICES



Some MONOLITHIC
APPLICATIONS

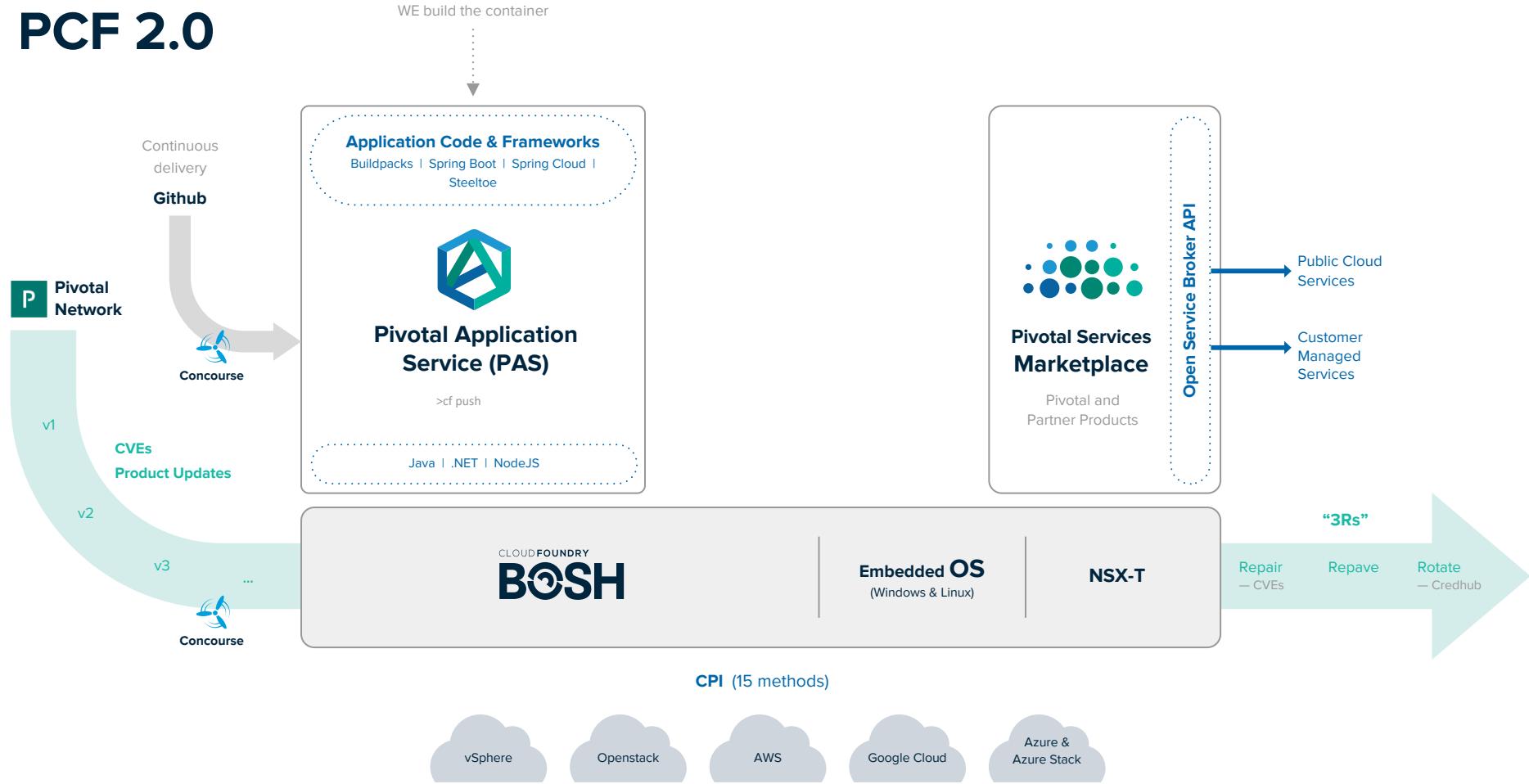


Some DATA SERVICES



.NET APPLICATIONS

PCF 2.0



Pivotal Application Service (PAS): A Runtime for Apps



Increase speed and deploy code to production thousands of times per month. Use PAS to run Java, .NET, and Node apps.

Best runtime for Spring and Spring Boot — Spring's microservice patterns—and Spring Boot's executable jars—are ready-made for PAS.

Turnkey microservices operations and security — Spring Cloud Services brings microservices best practices to PAS. It includes Config Server, Service Registry, and Circuit Breaker Dashboard.

A native Windows and .NET experience — Use PAS to run new apps built with .NET Core. Run your legacy .NET Framework apps on PAS too, using the .NET Hosted Web Core buildpack. Push applications to containers running on Windows Server 2016.

Built for apps — PAS has everything to need to run apps. Buildpacks manage runtime dependencies; metrics, logging, and scaling are done for you. Multitenancy, and blue/green deployment patterns are built-in. Extend apps with a rich service catalog.

Container-ready — PAS supports the OCI format for Docker images. Run platform-built and developer-built containers.

Can we realize these benefits for other workloads too?



MICROSERVICES



Some MONOLITHIC APPLICATIONS



Some DATA SERVICES



.NET APPLICATIONS



CONTAINERS



COTS



More MONOLITHIC APPLICATIONS



Stateful or MICROSERVICES Clusters

Can we realize these benefits for other workloads too?



MICROSERVICES



Some MONOLITHIC APPLICATIONS



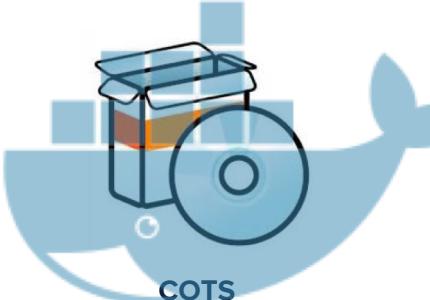
Some DATA SERVICES



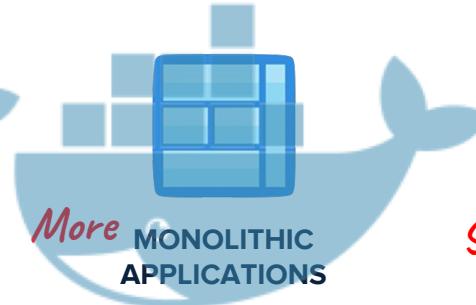
.NET APPLICATIONS



CONTAINERS



COTS

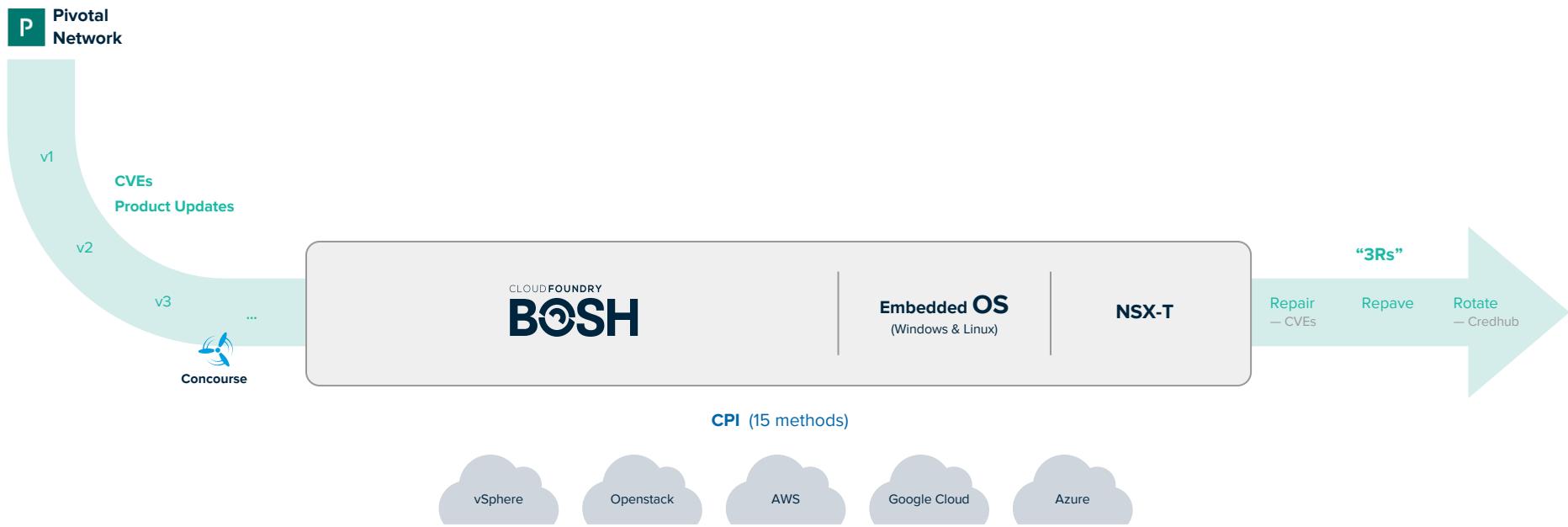


More MONOLITHIC APPLICATIONS

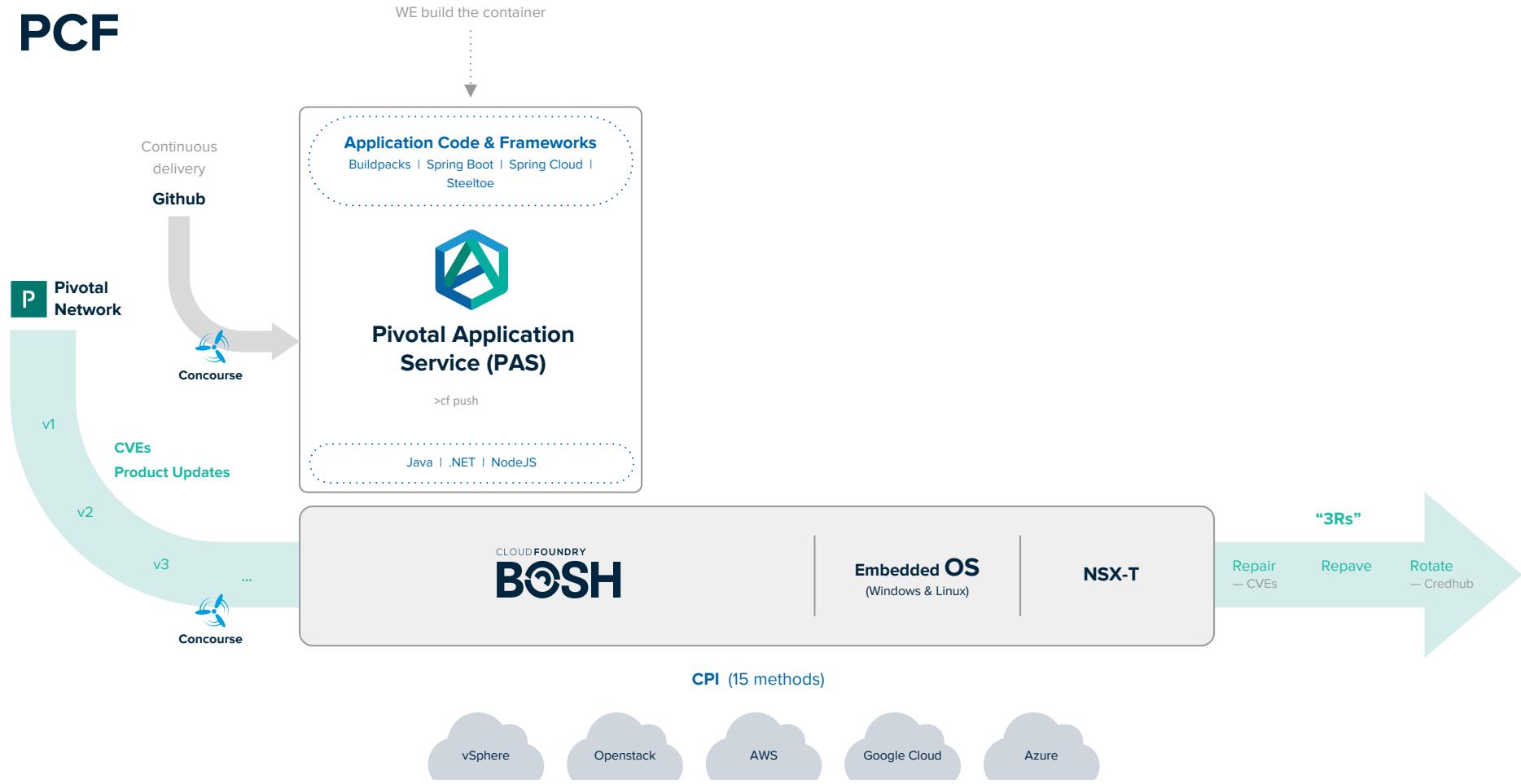


Stateful
or
MICROSERVICES
Clusters

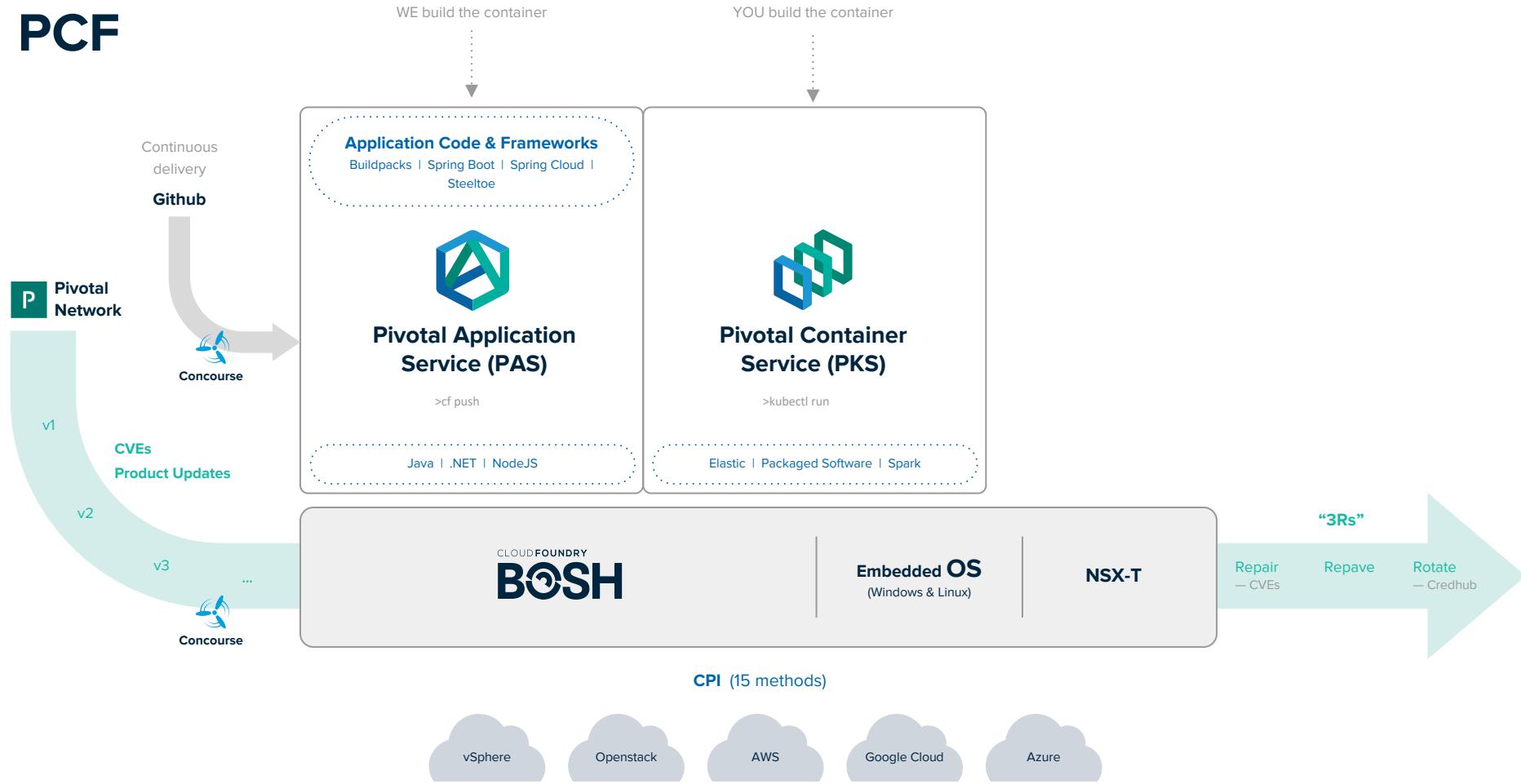
PCF



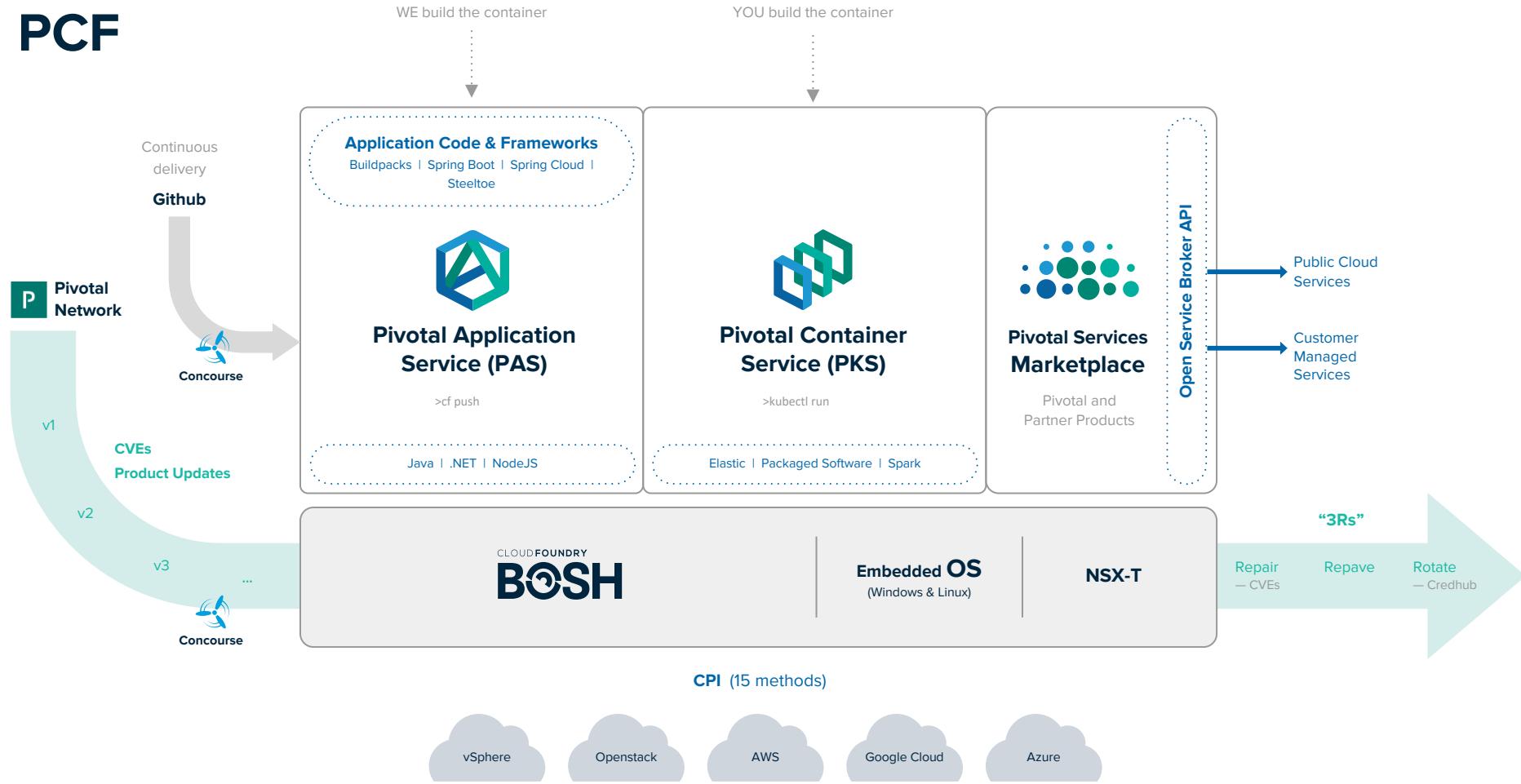
PCF



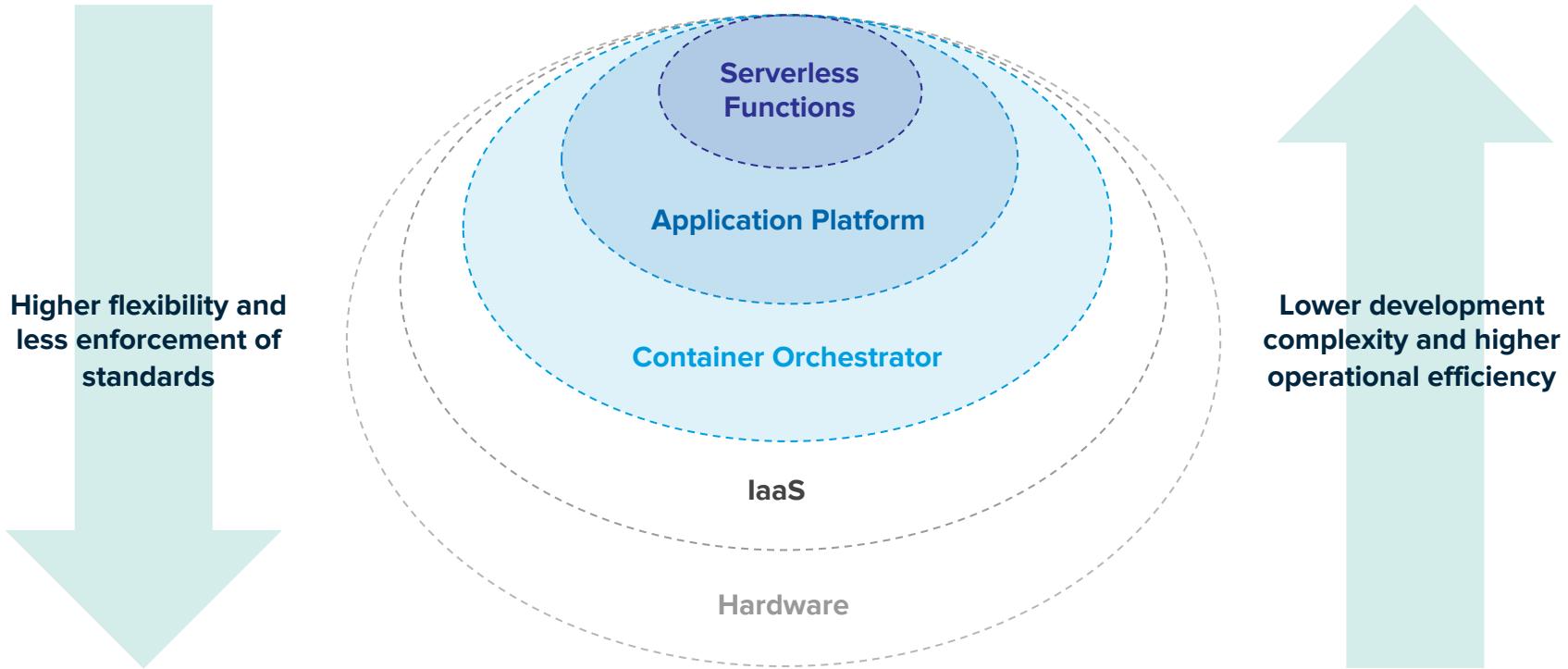
PCF



PCF



Choose the right tool for the job



Strategic goal (portfolio optimization): Push as many workloads as technically feasible to the top of the platform hierarchy

Kubernetes



Kubernetes is an open-source platform designed to automate deploying, scaling, and operating **application containers**.

With Kubernetes, you are able to quickly and efficiently respond to customer demand:

- Deploy your applications quickly and predictably.
- Scale your applications on the fly.
- Roll out new features seamlessly.
- Limit hardware usage to required resources only.
- Manage your applications like cattle instead of pets.

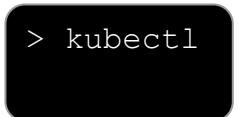
Kubernetes is a Runtime for Containerized Workloads



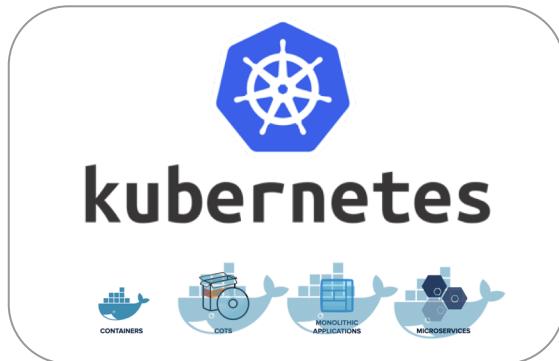
App User



Dev / Apps



Kubernetes Dashboard



IT / Ops

Compute

Storage

Networking

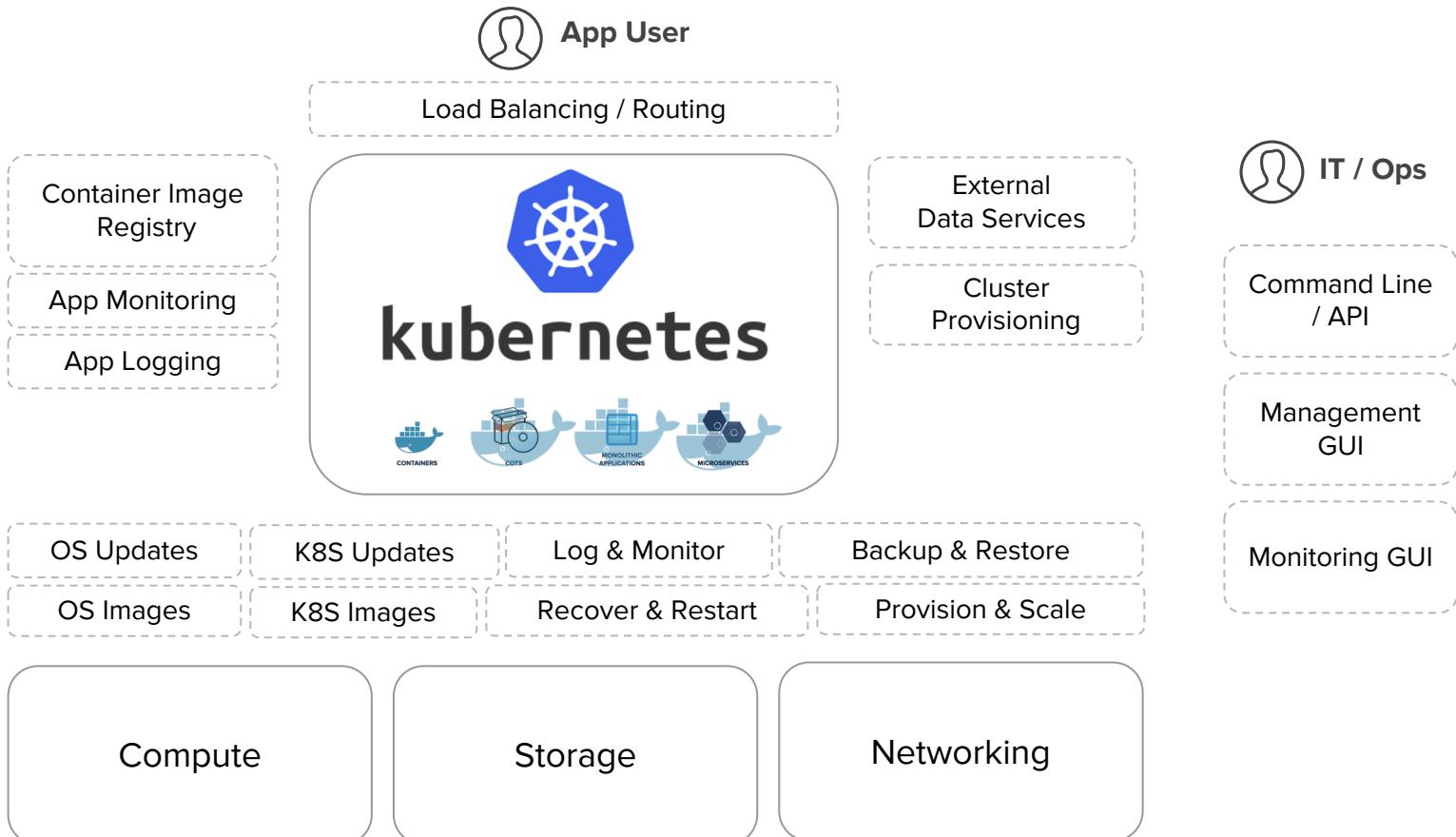
...but Kubernetes alone is not enough for enterprises



> kubectl



Kubernetes Dashboard



...but Kubernetes alone is not enough for enterprises

Kubernetes is a platform for building platforms. It's a better place to start; not the endgame.

7:04 PM - 27 Nov 2017

152 Retweets 462 Likes

9 152 462

Pivotal

Pivotal Container Service (PKS) provides what's missing

Pivotal

vmware

Google



> kubectl



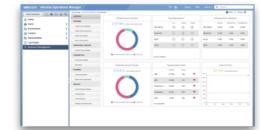
Kubernetes Dashboard



> pks



Operations Manager



vRealize Operations*

on any Cloud

Pivotal

vmware

Google

Dev / Apps

> kubectl



Kubernetes Dashboard

App User

vmware
NSX



kubernetes



WAVEFRONT

vRealize
LogInsight



PKS Control Plane

IT / Ops

> pks



Operations Manager

CLOUD FOUNDRY
BOSH

f flannel

vmware
NSX

vmware
vSphere

Azure

Google Cloud Platform

aws

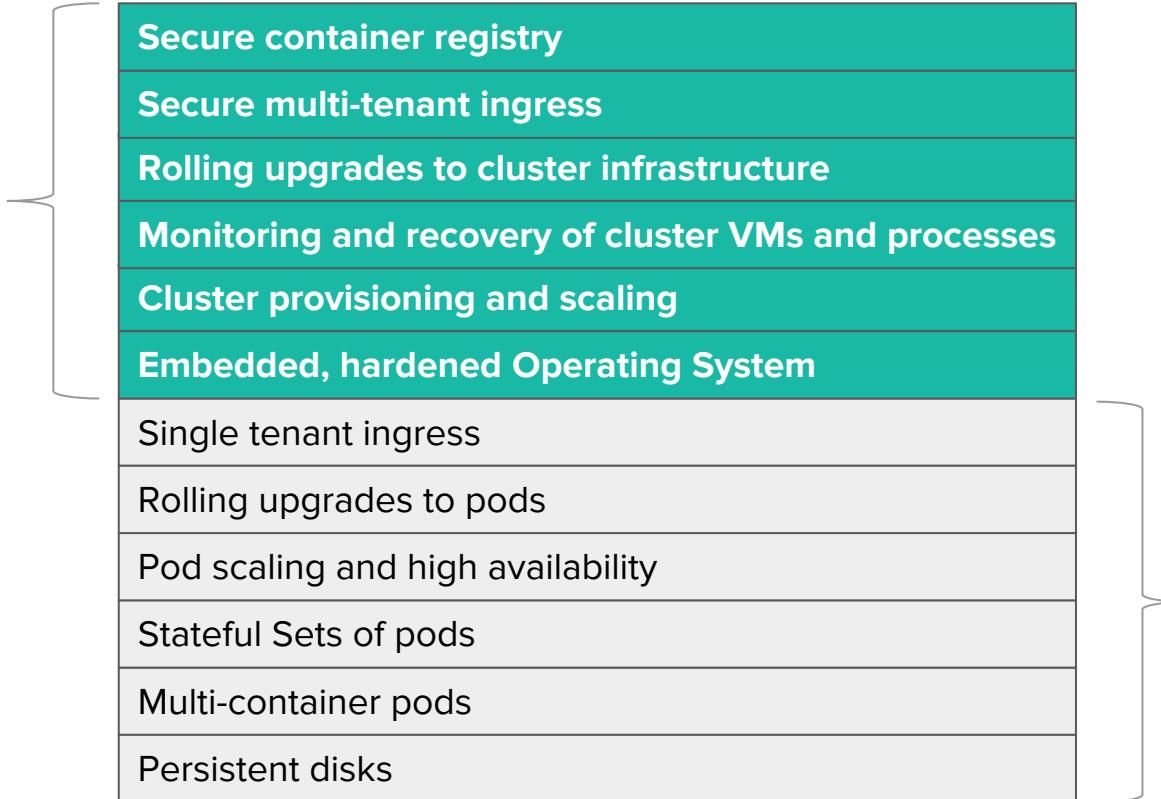
openstack.



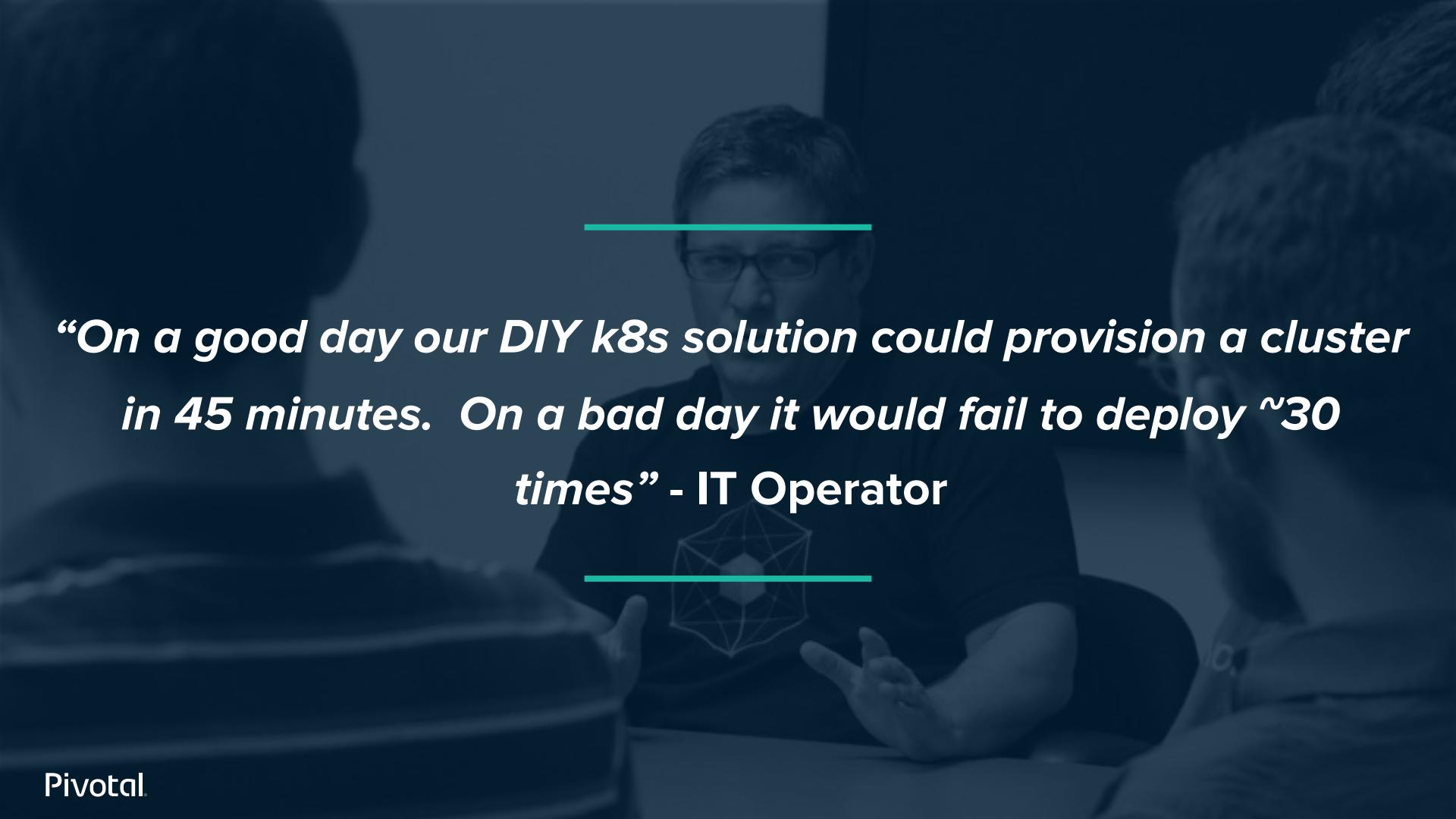
vRealize Operations*

What PKS adds to Kubernetes

PKS value-added features



Built into
Kubernetes



“On a good day our DIY k8s solution could provision a cluster in 45 minutes. On a bad day it would fail to deploy ~30 times” - IT Operator

Three Questions



Upgrades

How many times does K8s release per year?

Answer: Major releases quarterly; minor releases/patches frequently

PIVOTAL'S OPINION

- Automated upgrades on-demand with BOSH
- Automatic patching with Concourse pipelines
- Self-healing nodes on failure



Multi-tenancy

How many clusters are you going to need?

Answer: More than one required to provide true multi-tenancy

PIVOTAL'S OPINION

- Self-service, on-demand provisioning of clusters
- Pre-defined T-shirt size clusters
- Scale clusters up and down



Network

Is your network ready? (How mature is your SDN?)

Answer: Probably not; if you require tickets/manual process for network or firewall rules, you aren't ready

PIVOTAL'S OPINION

- Microsegmentation with NSX-T
- Automated IP allocation and load balancer provisioning
- Monitoring & troubleshooting with familiar VMware tooling

Main Capabilities

What is PKS?



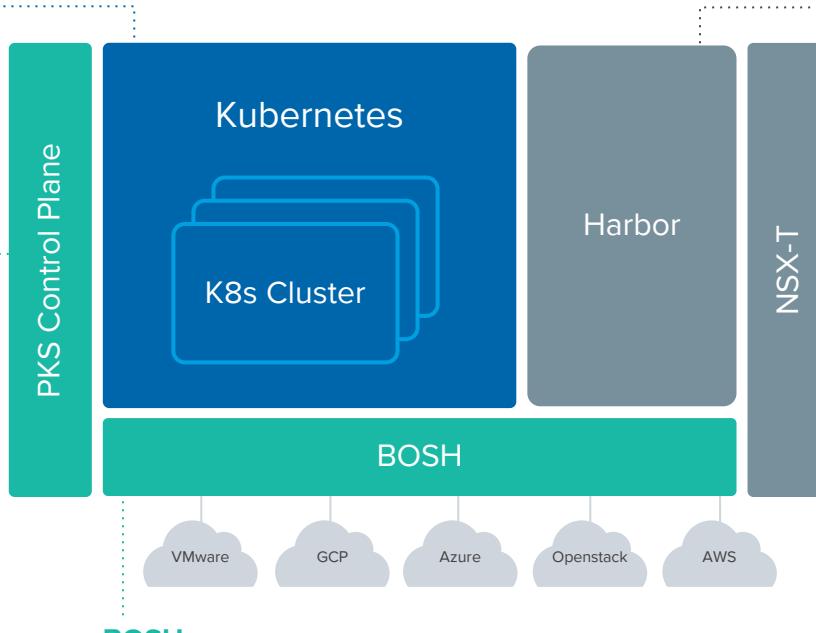
Enterprise-Grade Kubernetes

Built with open-source Kubernetes

Constant compatibility with the latest stable release Kubernetes —no proprietary extensions.

PKS Control Plane

Use the PKS CLI and API to create, operate, and scale your clusters.



BOSH

Reliable and consistent operational experience for any cloud.

Harbor

An enterprise-class container registry. Includes vulnerability scanning, identity management, and more.

NSX-T

Network management, security, and load balancing out-of-the-box with VMware NSX-T. Multi-cloud, multi-hypervisor.

Main Capability #1

BOSH



BOSH is an open source tool for release engineering, deployment, lifecycle management, and monitoring of distributed systems.

mark lucovsky (@marklucovsky) Follow

BOSH: Designed and built by @vadimspivak and @skaar, influenced by Google's borg. BOSH == borg++ (r+1=s, g+1=h).

Cloud Foundry @cloudfoundry
BOSH is the powerful orchestration tool that lies at the heart of #CloudFoundry.
ow.ly/4ne4iU

RETWEETS 12 LIKES 20

7:11 AM - 7 May 2016

1 12 20



Package



Provision



Deploy



Monitor



Upgrade



- Packaging w/ embedded OS
- Server provisioning on any IaaS
- Software deployment across AZs



- Health monitoring (server & processes)
- Self-healing w/ Resurrector
- Storage management
- Rolling upgrades with canaries
- Easy scaling of clusters
- Repeatability and Consistency

A dark, semi-transparent background image of two people, a man and a woman, looking at a screen. The screen displays a complex network graph with many nodes and connections. A horizontal teal line is positioned above the text.

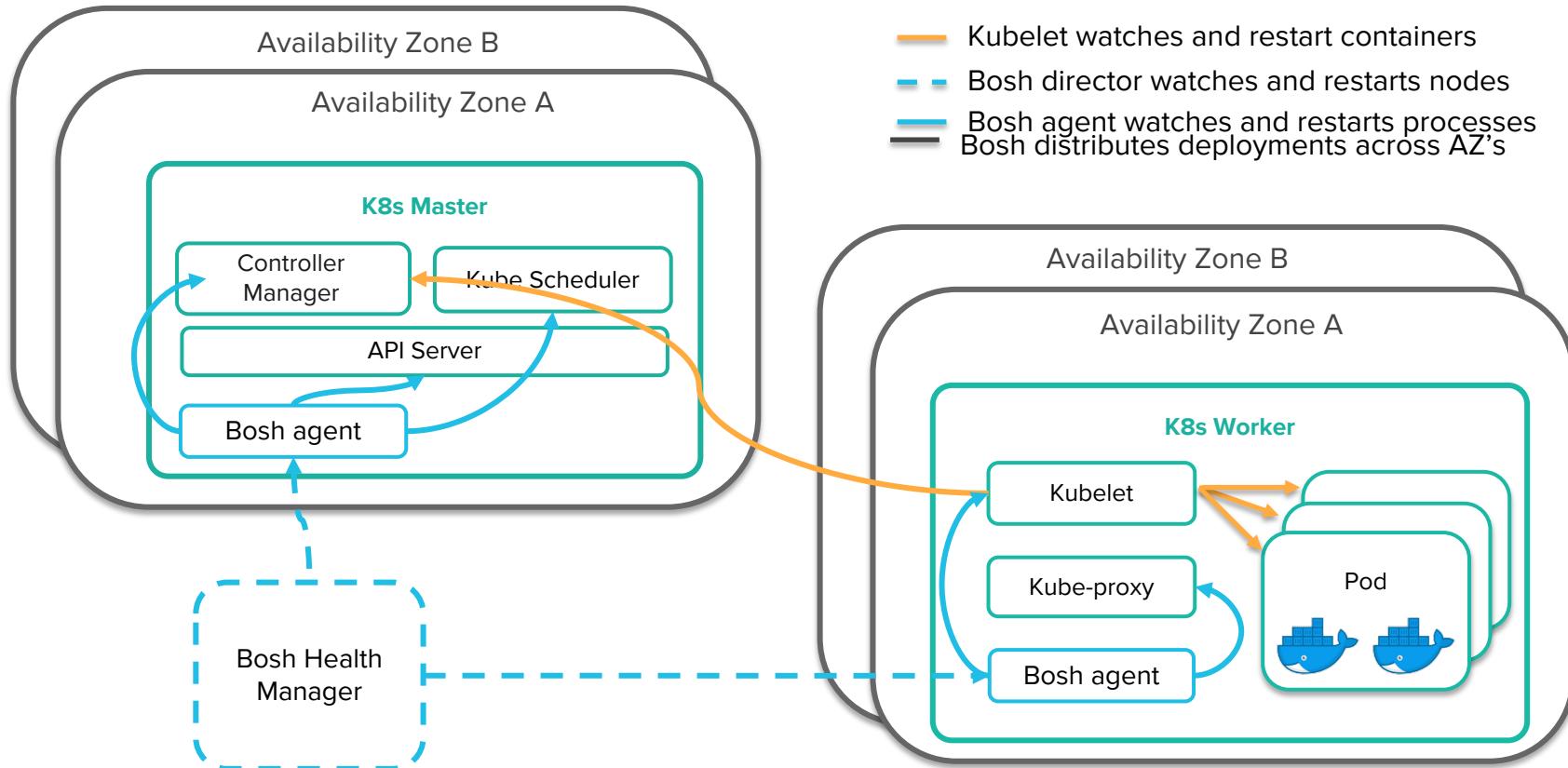
PKS does for your Kubernetes

what

Kubernetes does for your apps



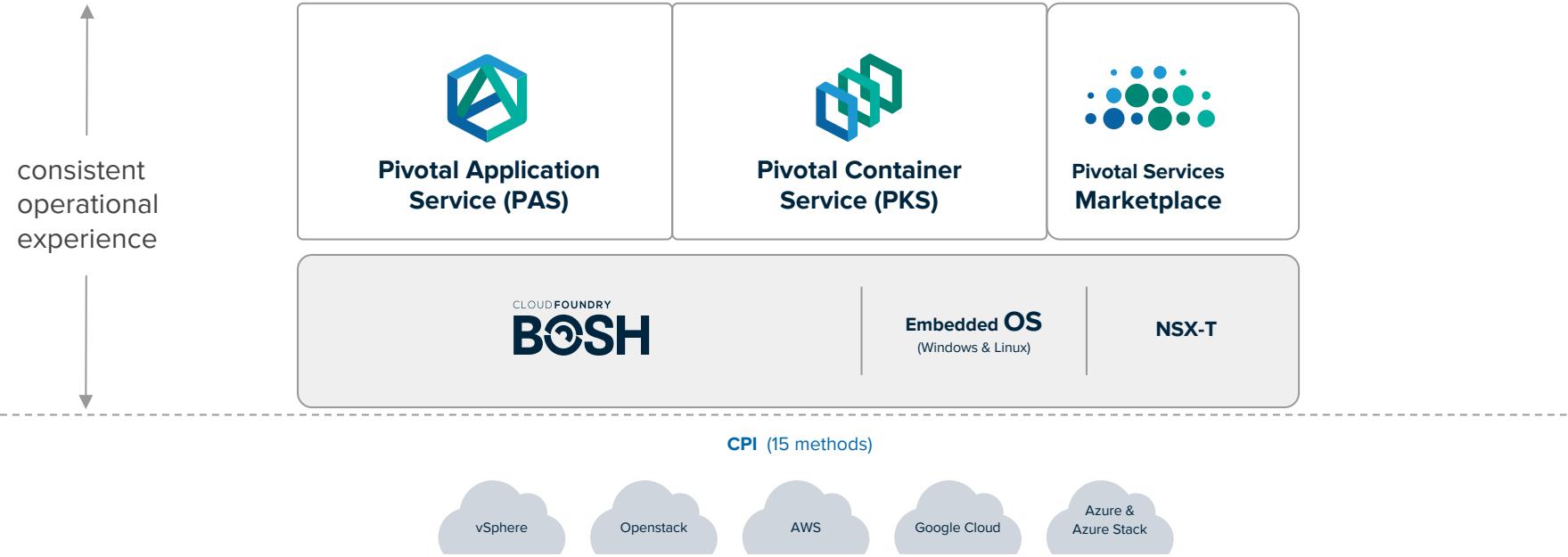
PKS Health Management



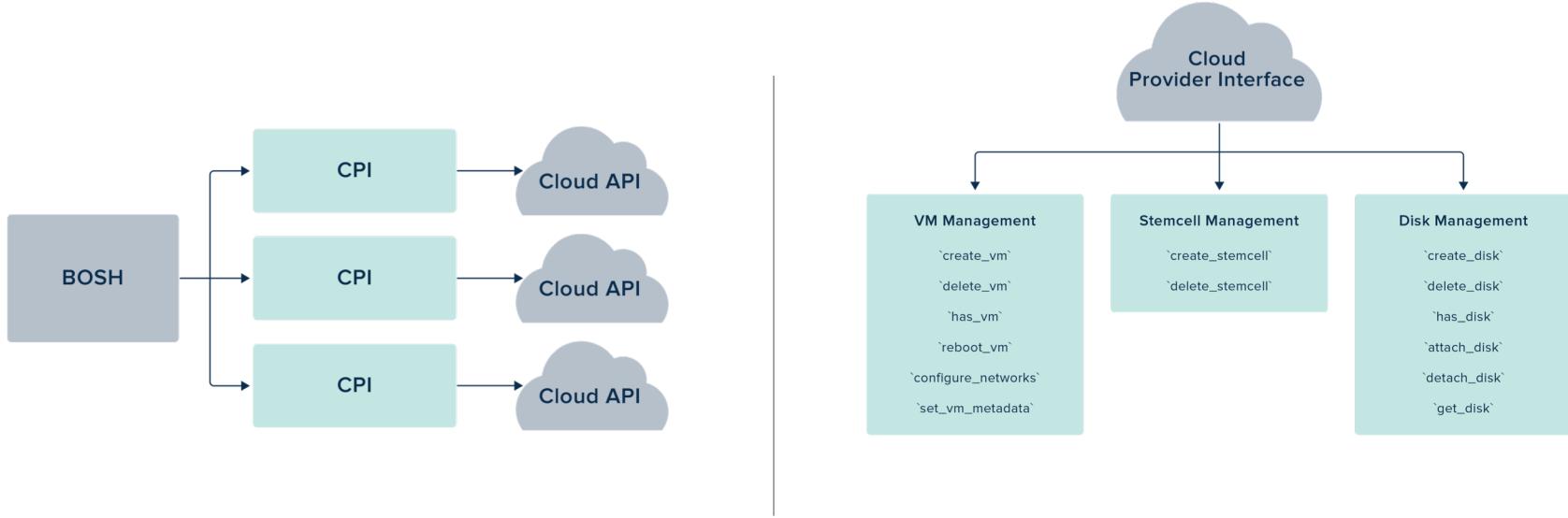
Main Capability #2

Multi Cloud

Multicloud



Multi-Cloud with BOSH + CPI



Main Capability #3

Multi Cluster

Tenancy



Joe Beda

@jbeda

Follow



Replying to @andreisavu @foolusion

what they are doing is what I would call "soft multitenant". Share network for fast cross team rpc.

2:29 AM - 8 Dec 2016 from Seattle, WA

Multi-tenancy models remain weak in
Kubernetes alone

We provide solutions for this today

Two models supported

Multi-tenant clusters

- Leverage Kubernetes namespaces

Limitations with Kubernetes alone

- Noisy neighbors (workloads can affect other tenants)
- Share the same network
- Share DNS
- Shared Configuration
- ...

We add

- Network microsegmentation with NSX-T
 - Eliminating “Share the same network”

No other “on prem” solution has this!!!

Multi (Single-tenant) clusters

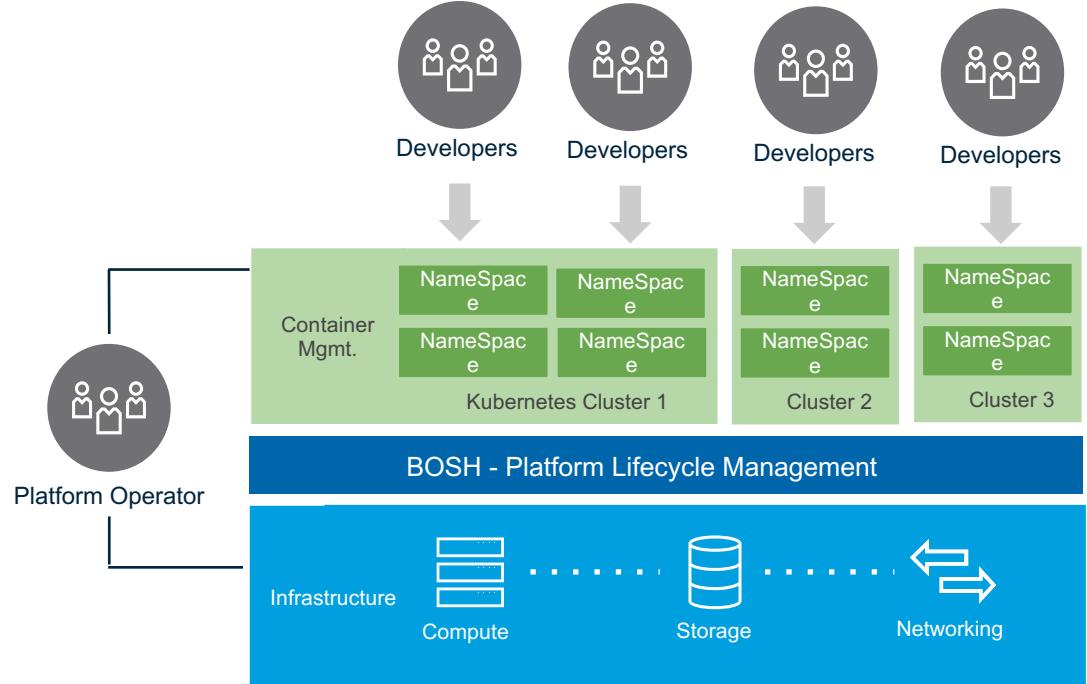
It is having an API for creation and management that enables this!!!

- Every tenant gets their own cluster

Addresses limitations

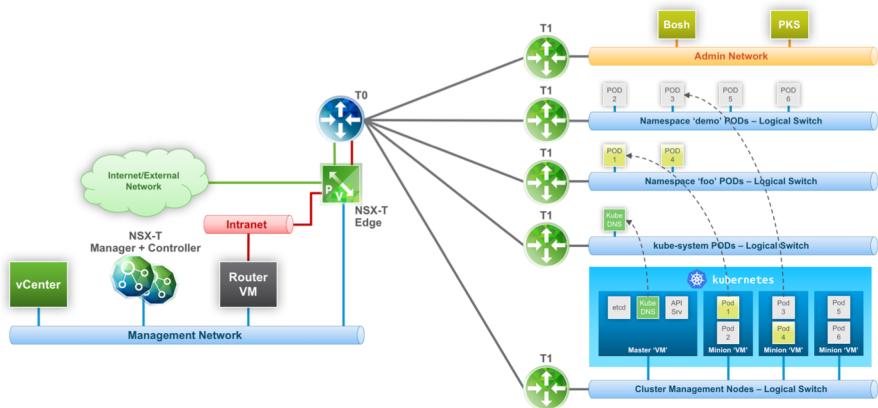
- Single tenant worker VMs (depend on the hypervisor to ensure host is properly shared)
- Every cluster has own network segment
- Every cluster has own DNS
- Every cluster has own configuration
- ...

Flexible Multi-Tenancy



Main Capability #4

NSX-T



Unified VM to Container Networking

On-demand network virtualization

Microsegmentation

Full Network Visibility

Enterprise Support

Pod-Level Container Networking

Load Balancing

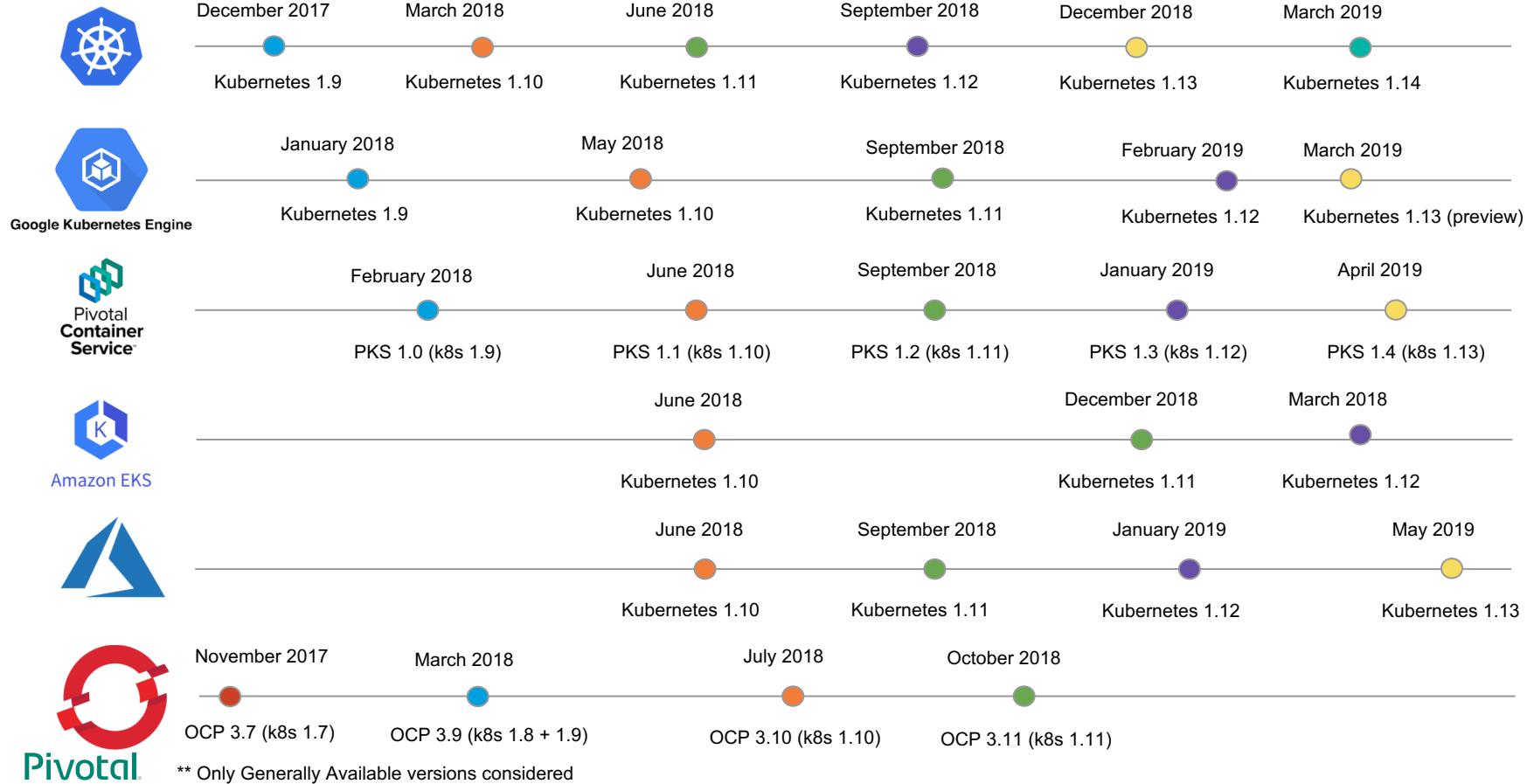
Network Security policies

Tenant - level isolation

Unique logical switch per K8s namespace

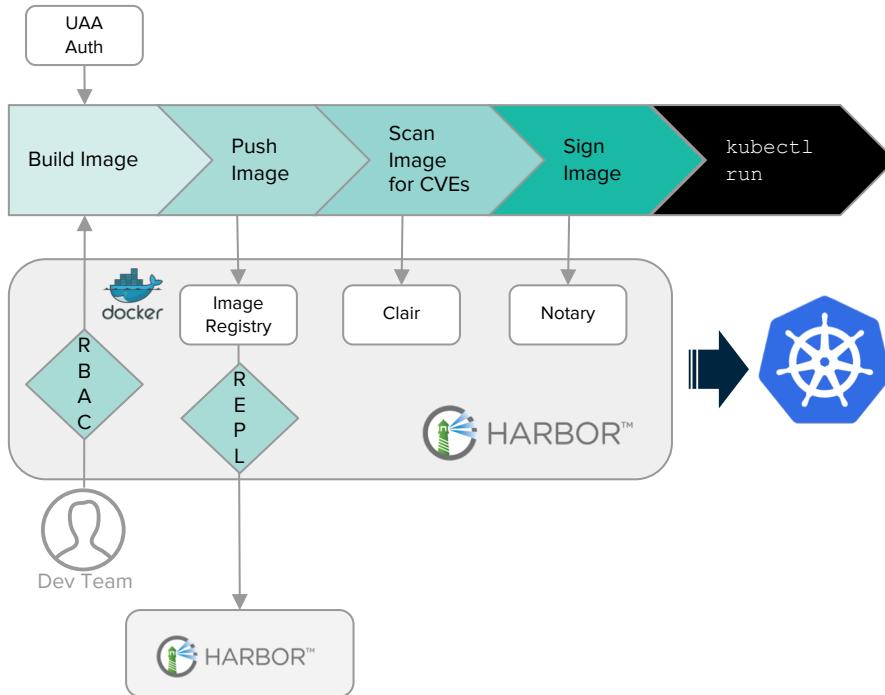
And...

Keeping the pace with k8s / giving Devs the latest





An enterprise-class registry server for Docker images



Role-Based Access Control (RBAC)

LDAP/AD Integration

Image Vulnerability Scanning (Clair)

Notary Image Signing

Policy-Based Image Replication

Graphical User Portal & RESTful API

Image Deletion & Garbage Collection

Auditing

Container Registry Vulnerability scanning

Vulnerability	Severity	Package	Current version	Fixed in version
> CVE-2016-1252	high	apt	1.0.1ubuntu2.6	1.0.1ubuntu2.17
> CVE-2016-5011	low	util-linux	2.20.1-5.1ubuntu20.3	
Description: The parse_dos_extended function in partitions/dos.c in the libblkid library in util-linux allows physically proximate attackers to cause a denial of service (memory consumption) via a crafted MSDOS partition table with an extended partition boot record at zero offset.				
> CVE-2014-9114	low	util-linux	2.20.1-5.1ubuntu20.3	
> CVE-2013-0157	low	util-linux	2.20.1-5.1ubuntu20.3	
> CVE-2017-6350	low	vim	2:7.4.052-1ubuntu3	
> CVE-2017-5953	low	vim	2:7.4.052-1ubuntu3	
> CVE-2017-6349	low	vim	2:7.4.052-1ubuntu3	
> CVE-2016-1248	medium	vim	2:7.4.052-1ubuntu3	2:7.4.052-1ubuntu3.1
> CVE-2017-9525	low	cron	3.0pl1-124ubuntu2	
> CVE-2017-10685	medium	ncurses	5.9+20140118-1ubuntu1	
> CVE-2016-7543	medium	bash	4.3-7ubuntu1.5	4.3-7ubuntu1.7

Container Registry Image Signing

The screenshot shows the vSphere Integrated Containers interface. The left sidebar is titled 'PROJECT default-project' and includes sections for Home, Library (with 'Project repositories' selected), and Infrastructure. The main area is titled 'Project Repositories' and displays a table of Docker images. The table has two tabs: 'Name' (selected) and 'Tag'. The 'Name' tab shows three images: 'default-project/redis' (1 tag, 0 pulls), 'default-project/ubuntu' (1 tag, 20 pulls), and 'default-project/demo-busybox' (2 tags). The 'Tag' tab shows two tags for 'default-project/demo-busybox': '1.0' (vulnerability: green, Signed: red, Author: 6/24/2016, Creation Time: 7:23 AM) and 'signed' (vulnerability: green, Signed: green, Author: 11/1/2015, Creation Time: 6:22 AM). A blue box highlights the 'signed' tag entry.

Name	Tags	Pulls
default-project/redis	1	0
default-project/ubuntu	1	20
default-project/demo-busybox	2	0

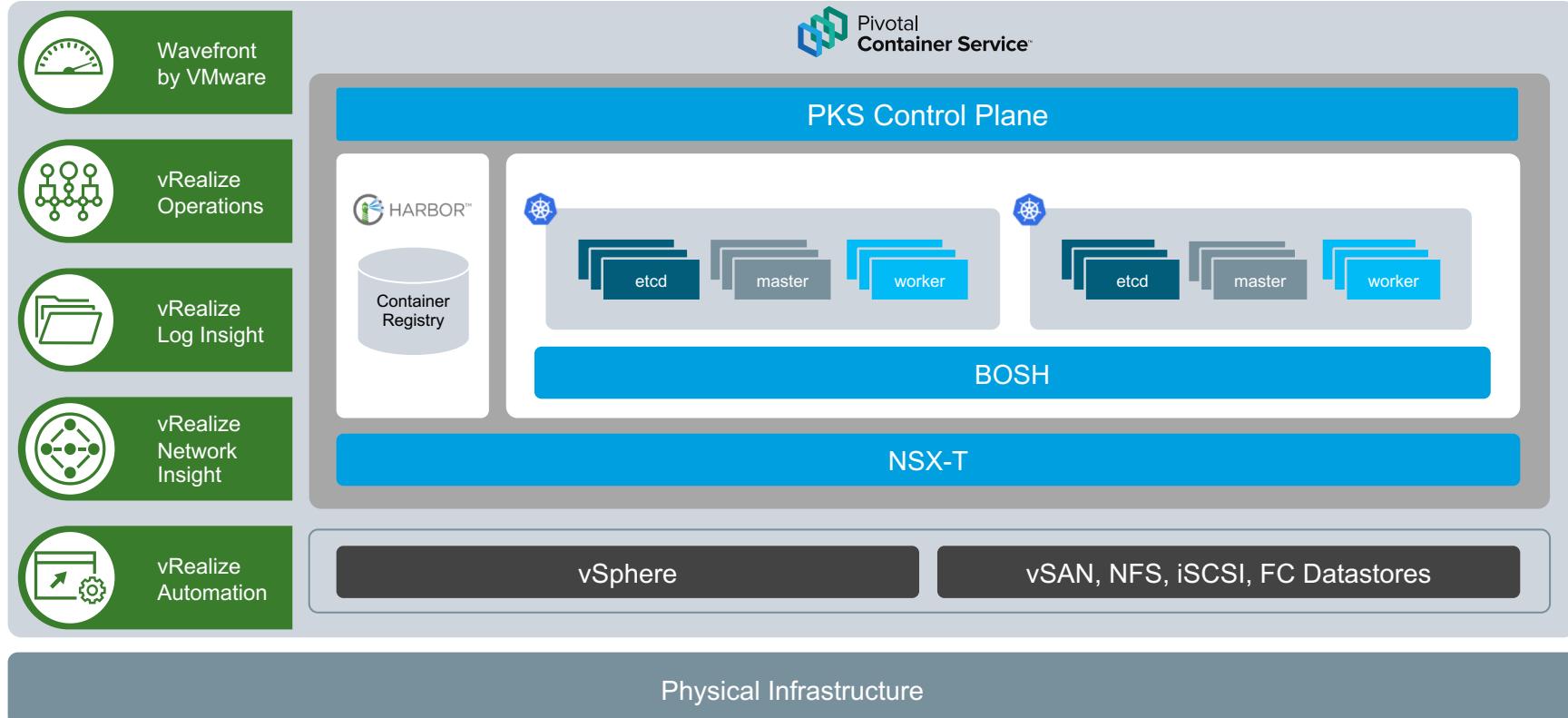
Tag	Pull Command	vulnerability	Signed	Author	Creation Time
1.0	docker pull 10.160.247.138/default-project/demo-busybox:1.0	green	✗	6/24/2016, 7:23 AM	
signed	docker pull 10.160.247.138/default-project/demo-busybox:signed	green	✓	11/1/2015, 6:22 AM	

```
root@jt-dev:/var/log/harbor/2017-08-09# export DOCKER_CONTENT_TRUST=0
root@jt-dev:/var/log/harbor/2017-08-09# docker pull 10.160.247.138/default-project/demo-busybox:1.0
Error response from daemon: unknown: The image is not signed in Notary.
root@jt-dev:/var/log/harbor/2017-08-09#
```



Pivotal
Container Service™

Integrations w/ VMware



Deep Dive

A Day in Life with PKS



Platform Operator (Alana)

- Install/Manage PKS
- Configure cluster plans
- Apply a patch / update
- Onboard Cluster Owner via RBAC
- Operate Bosh

Cluster Owner (Cody)

- Create a cluster
- Scale a cluster
- Create Network Policy
- Onboard App Dev via RBAC
- ...

App Dev (Naomi)

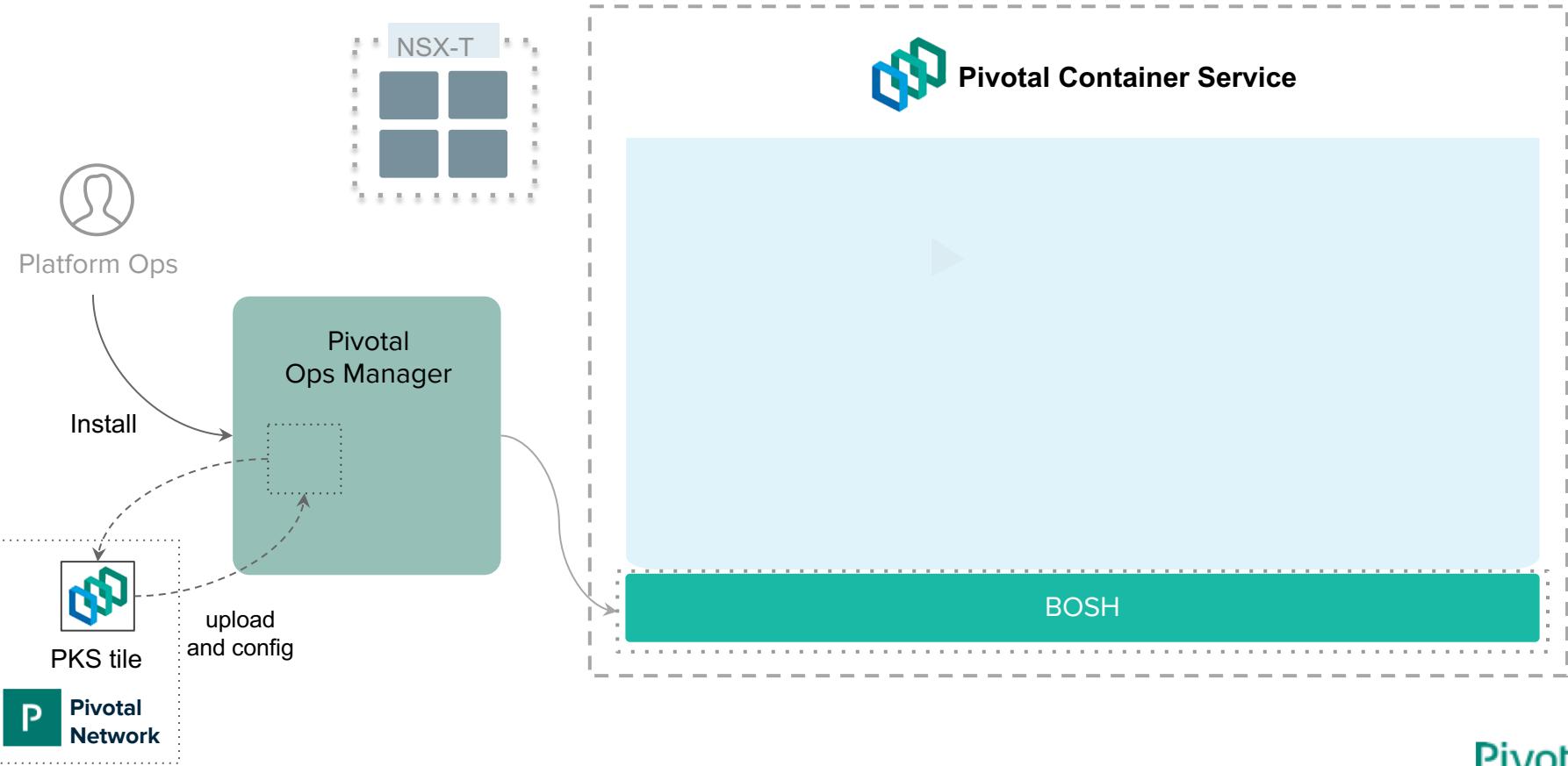
- Deploy an app
- Expose app with service type: LoadBalancer
- Expose app with Ingress
- ...



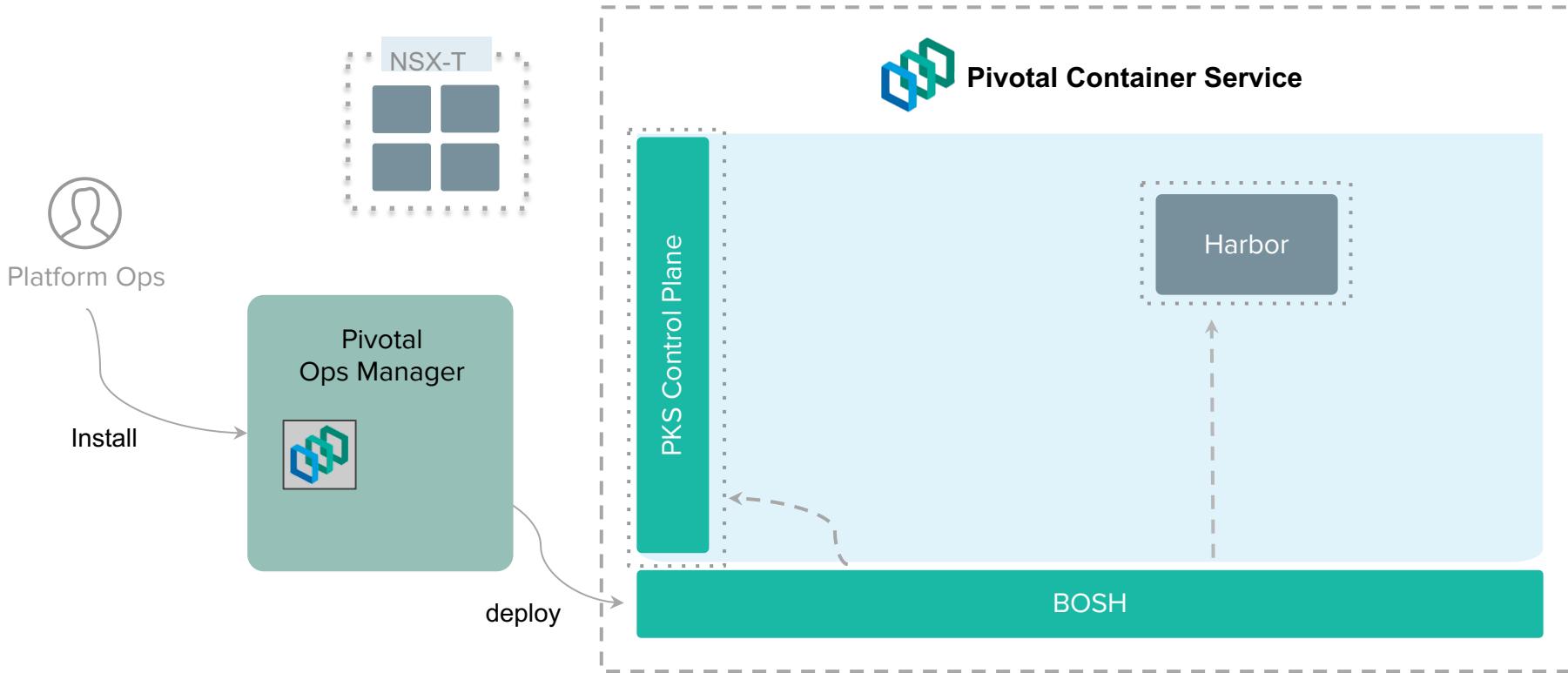
Automation

- Health Management (server & process)
- Network Automation

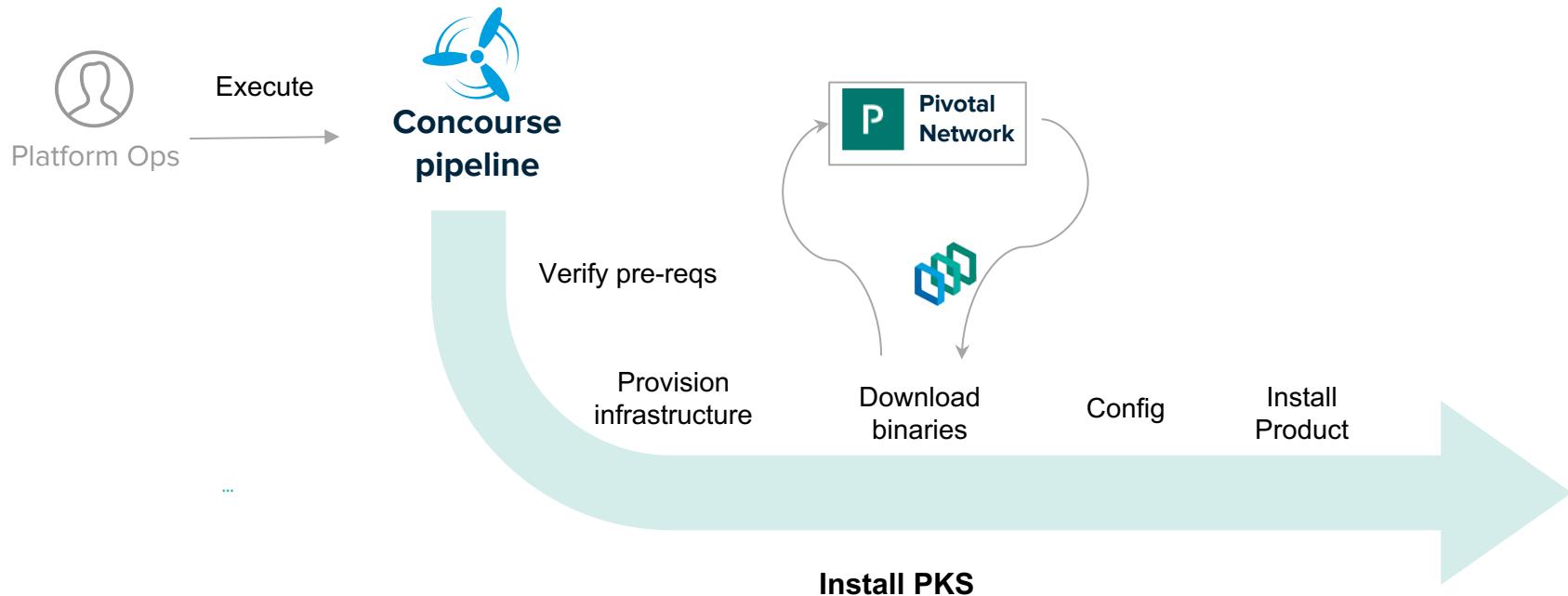
Installing PKS

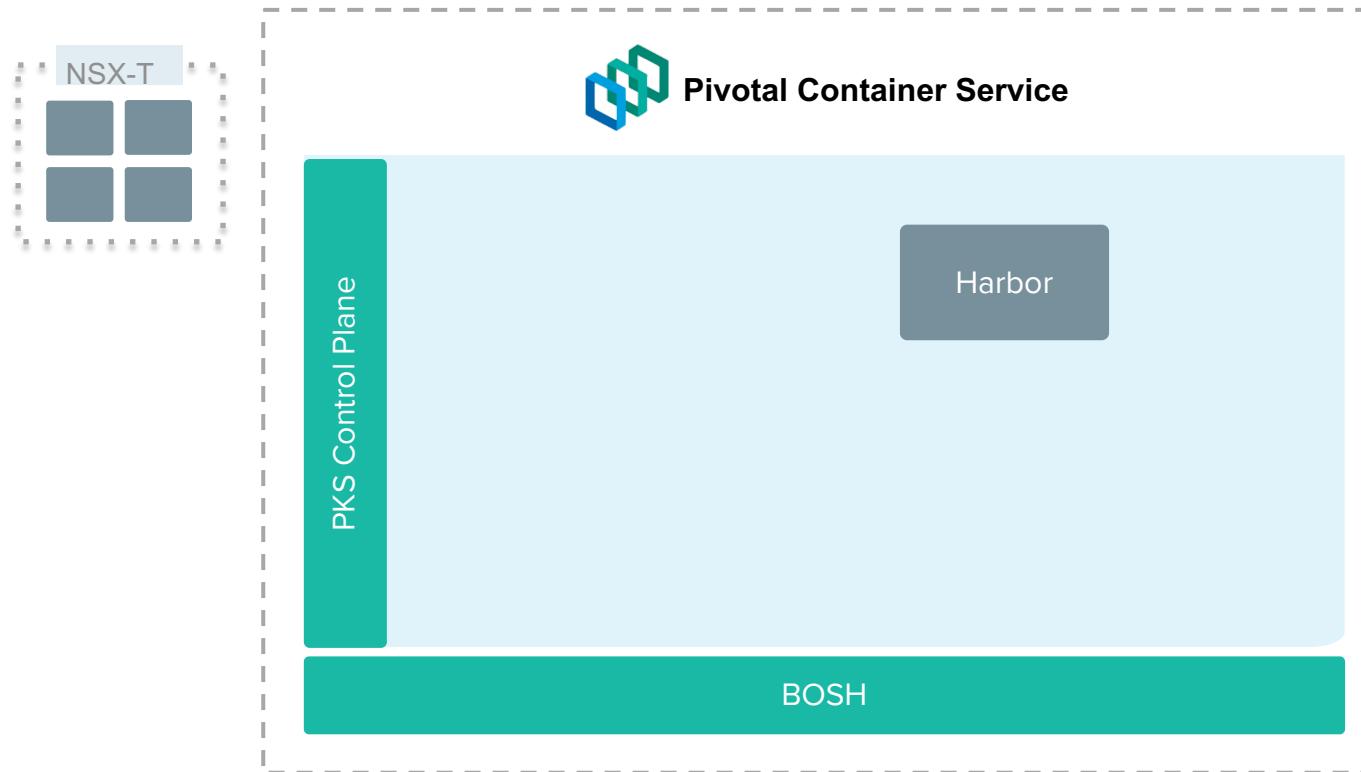


Installing PKS



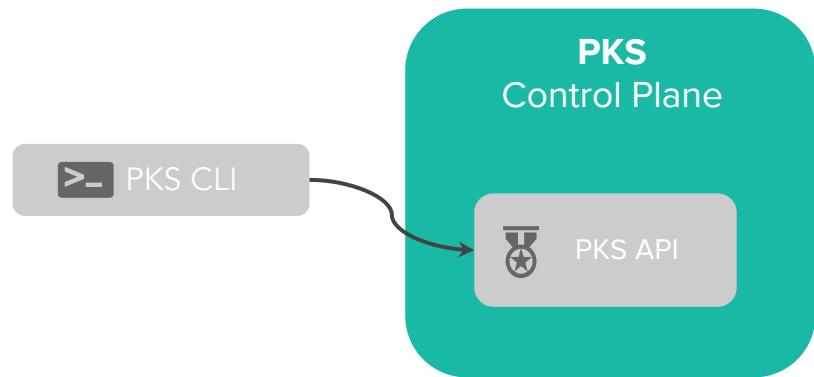
... Or



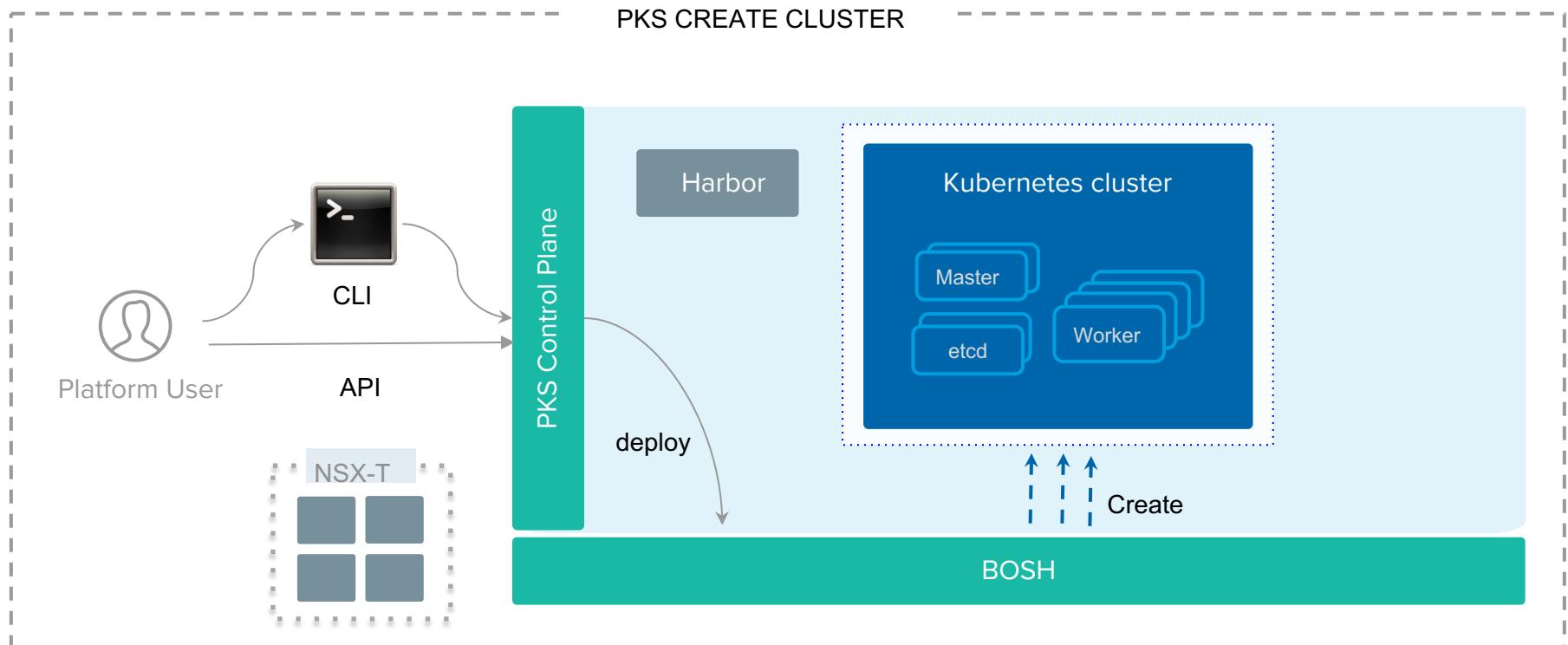


PKS User Interaction

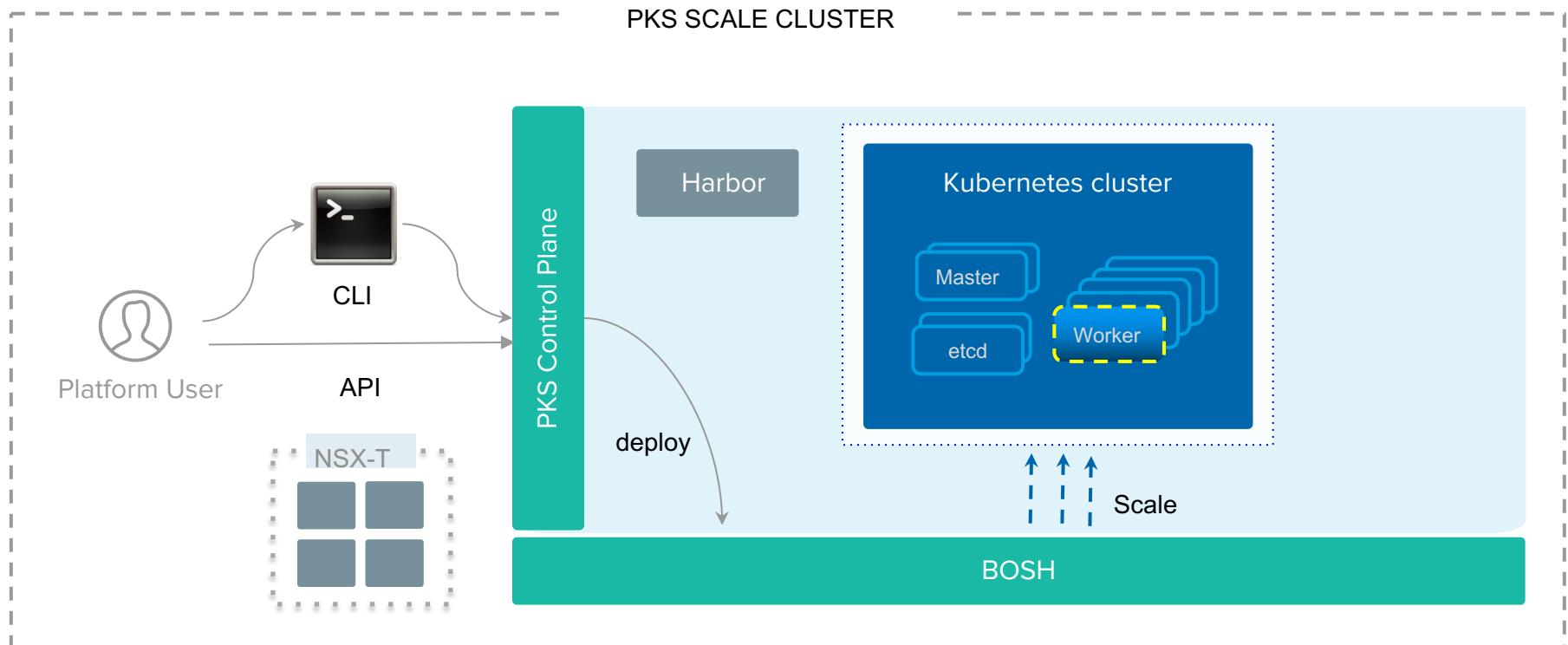
- The PKS Management VM runs the PKS API together with the Broker, UAA and a MySQL DB.
- The PKS API orchestrates the initial kubernetes cluster deployments and scaling of those clusters.
- A single PKS VM can manage hundreds of Kubernetes cluster.
- The PKS CLI is a single binary that can be installed on a Mac, Windows, or Linux to drive the PKS API.



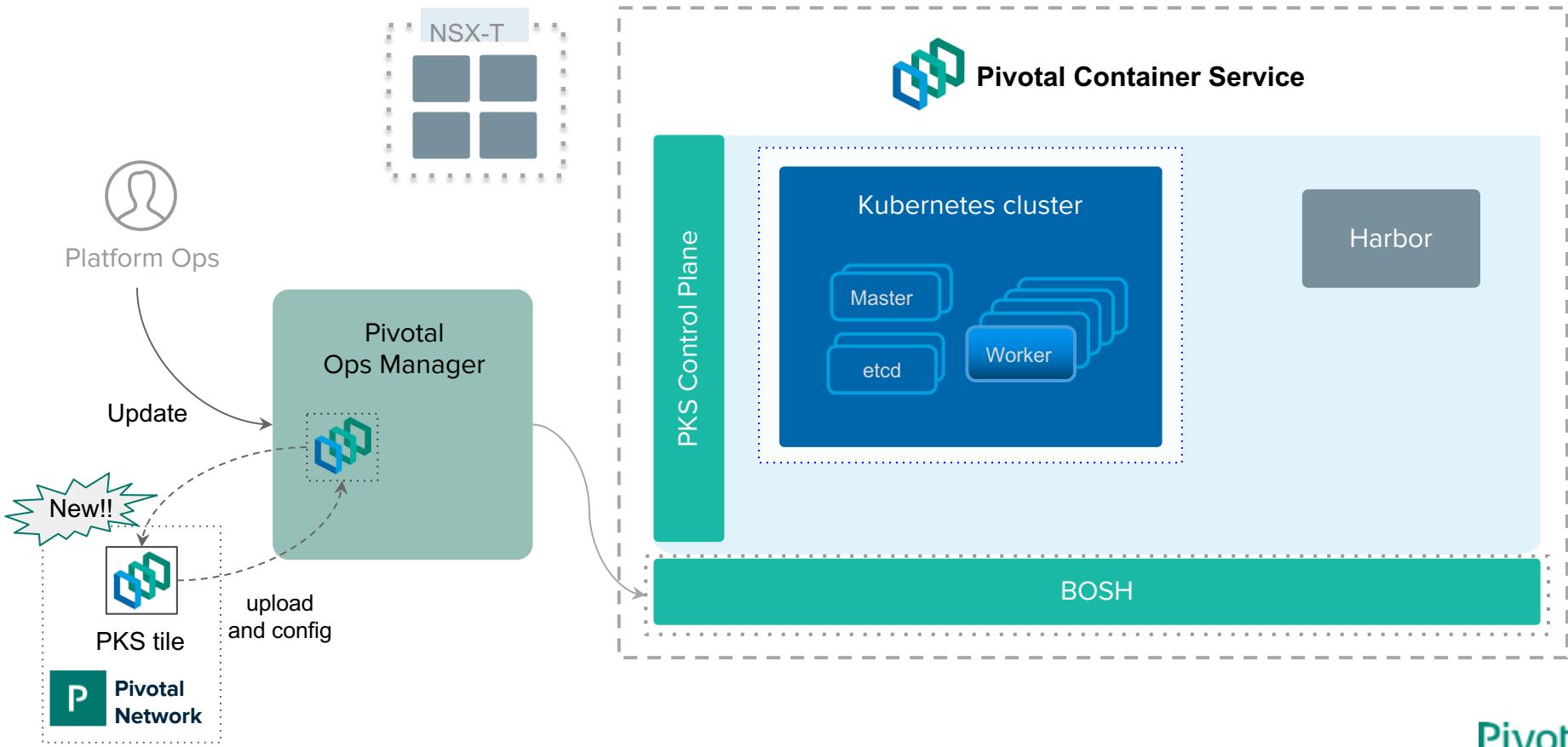
Creating a new K8s Cluster



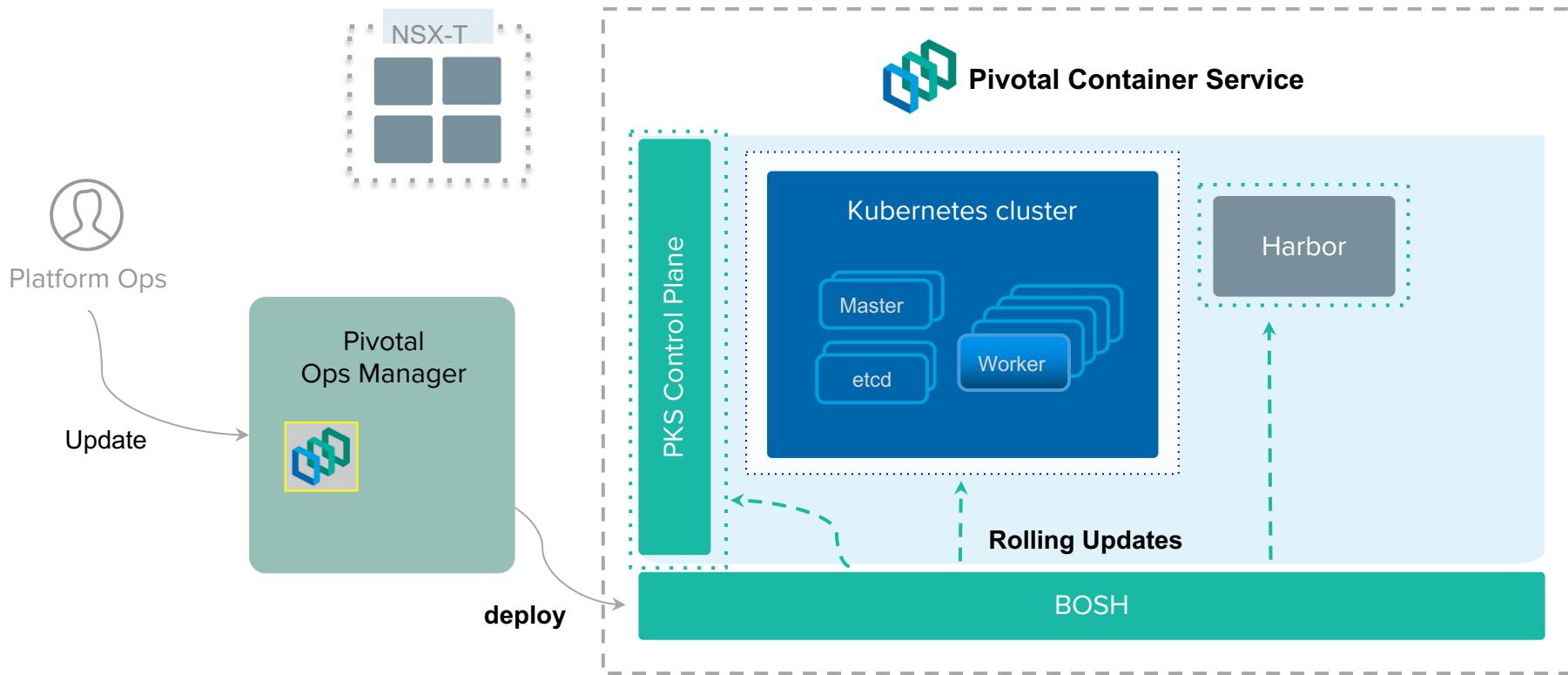
Scaling a Kubernetes Cluster



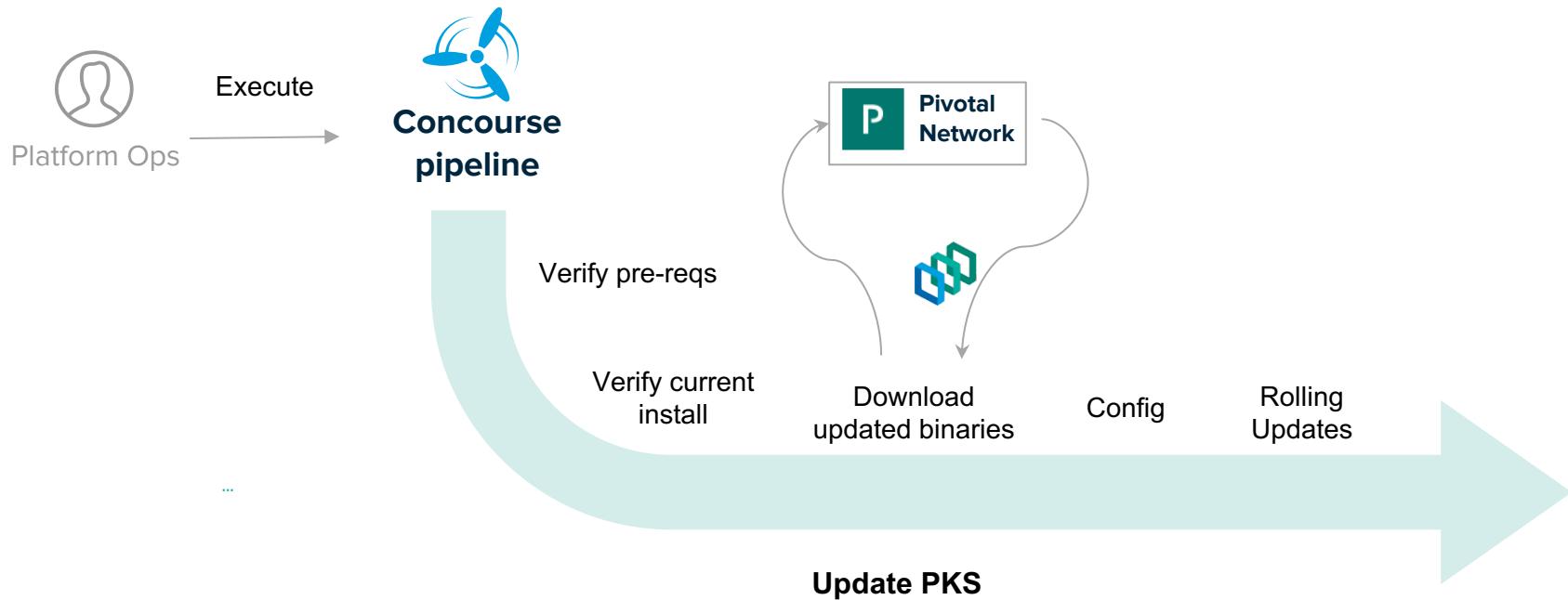
Platform Ops updates PKS



Platform Ops updates PKS

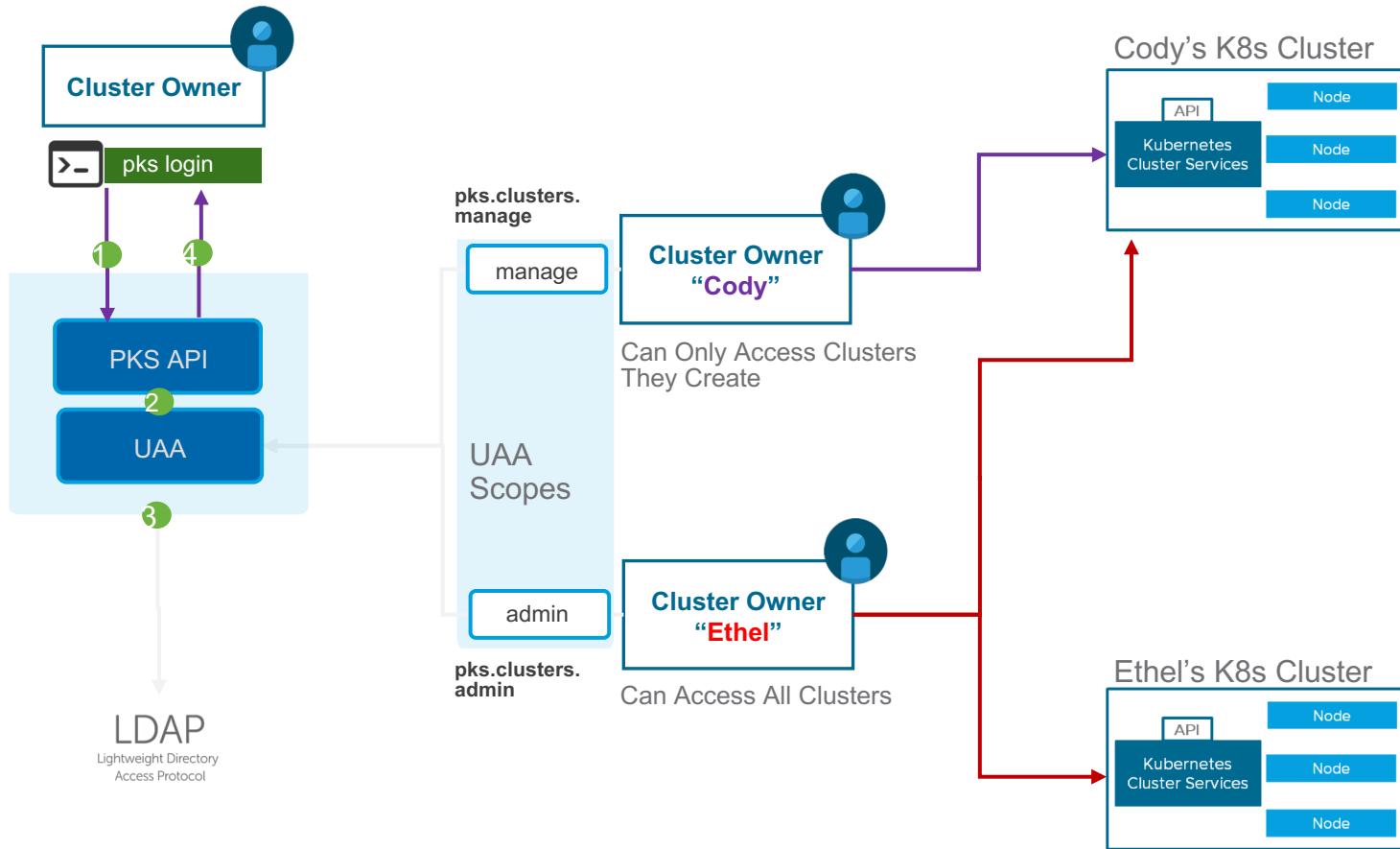


... Or

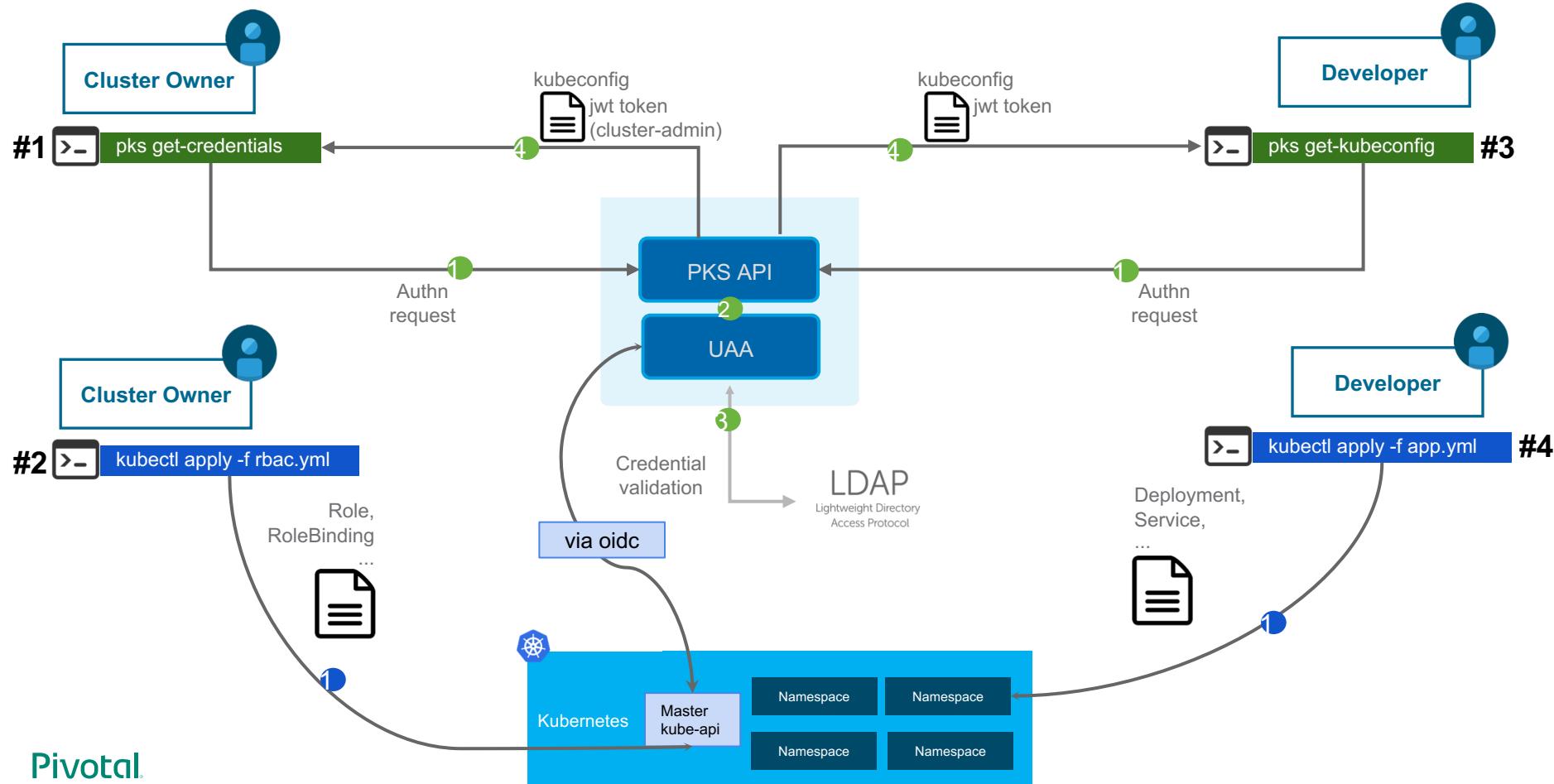


PKS Control Plane - Centralized Access Management

- Identity Backed by Enterprise Standards (LDAP/AD)
- PKS RBAC Controls who can create, scale, and update K8s clusters on demand



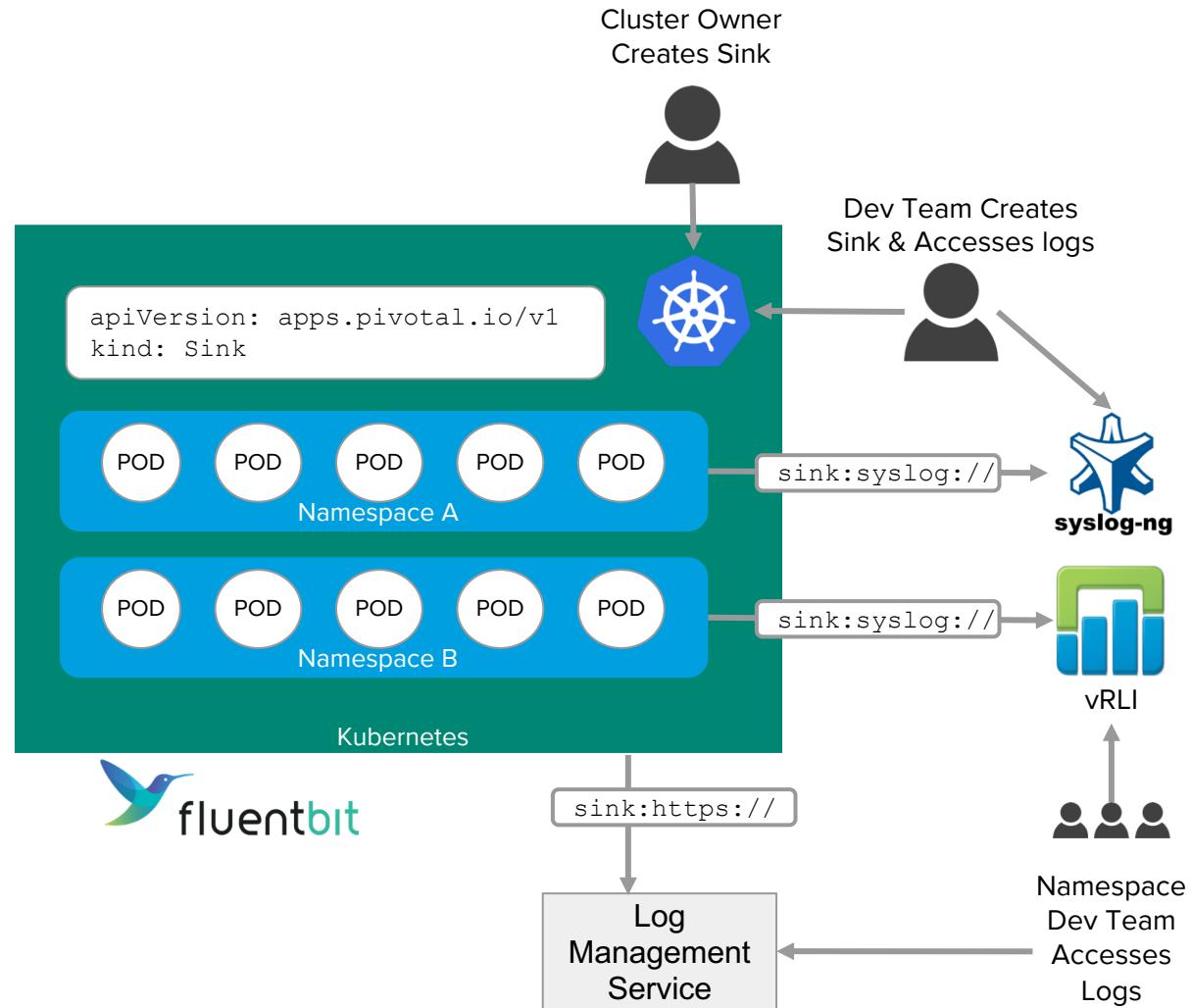
K8s Cluster - Access Authentication/Authorization



Dev Productivity: Log Sinks

Application Logging by Cluster or Namespace

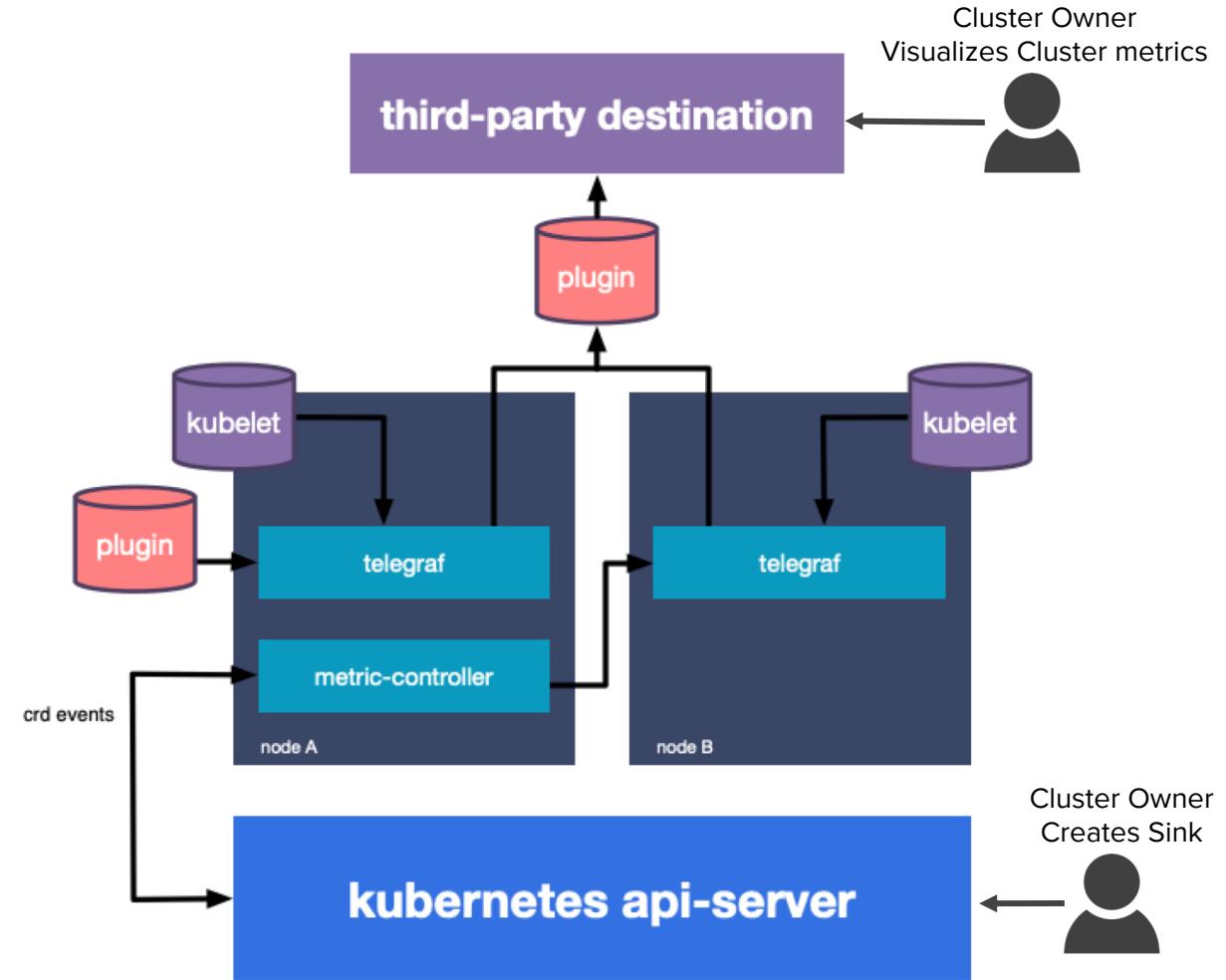
- Syslog endpoints
- Webhook endpoints



Operations and Observability: Metrics Sinks

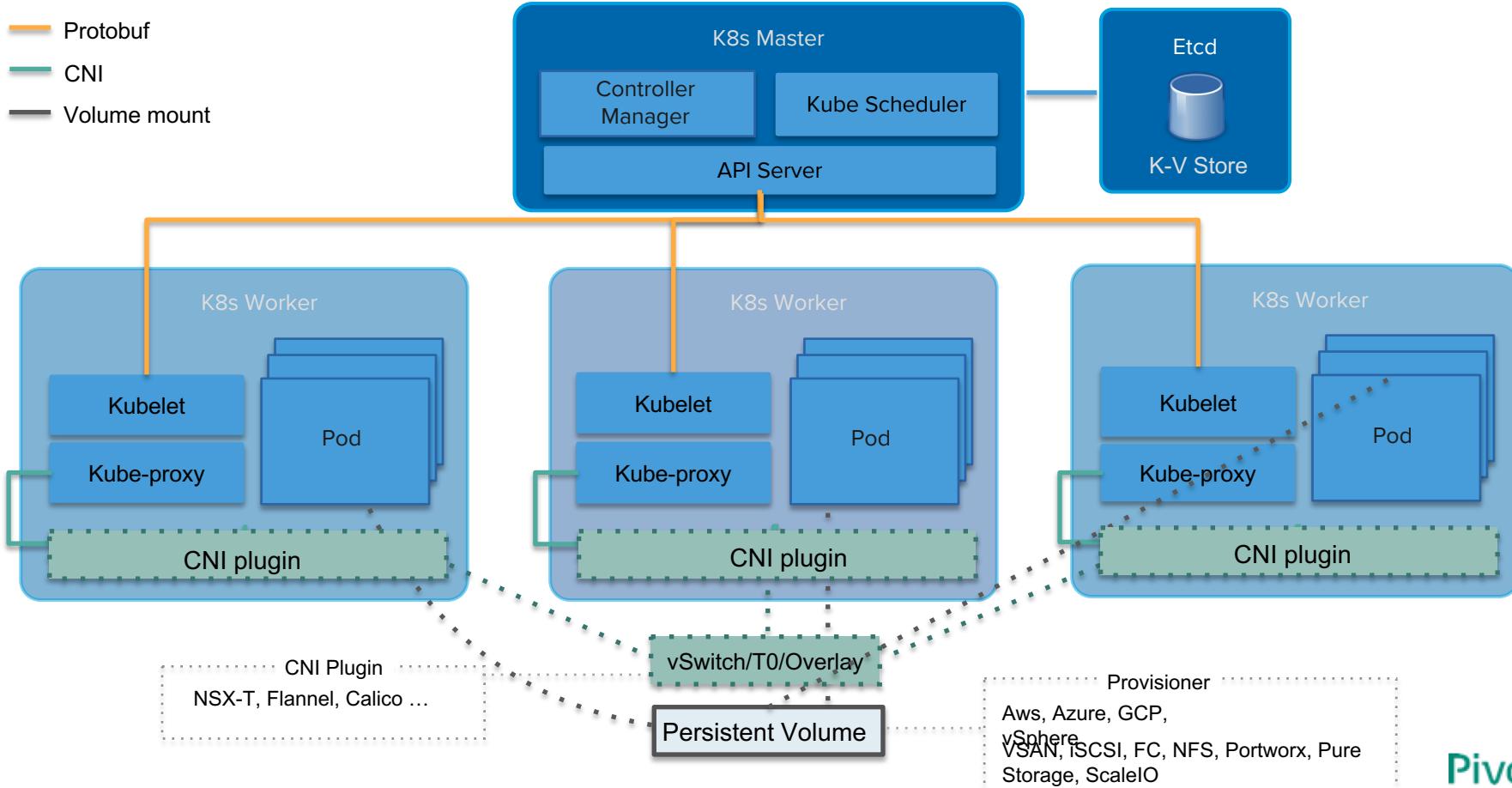
Metric Sink collects and writes metrics from a cluster to specified outputs

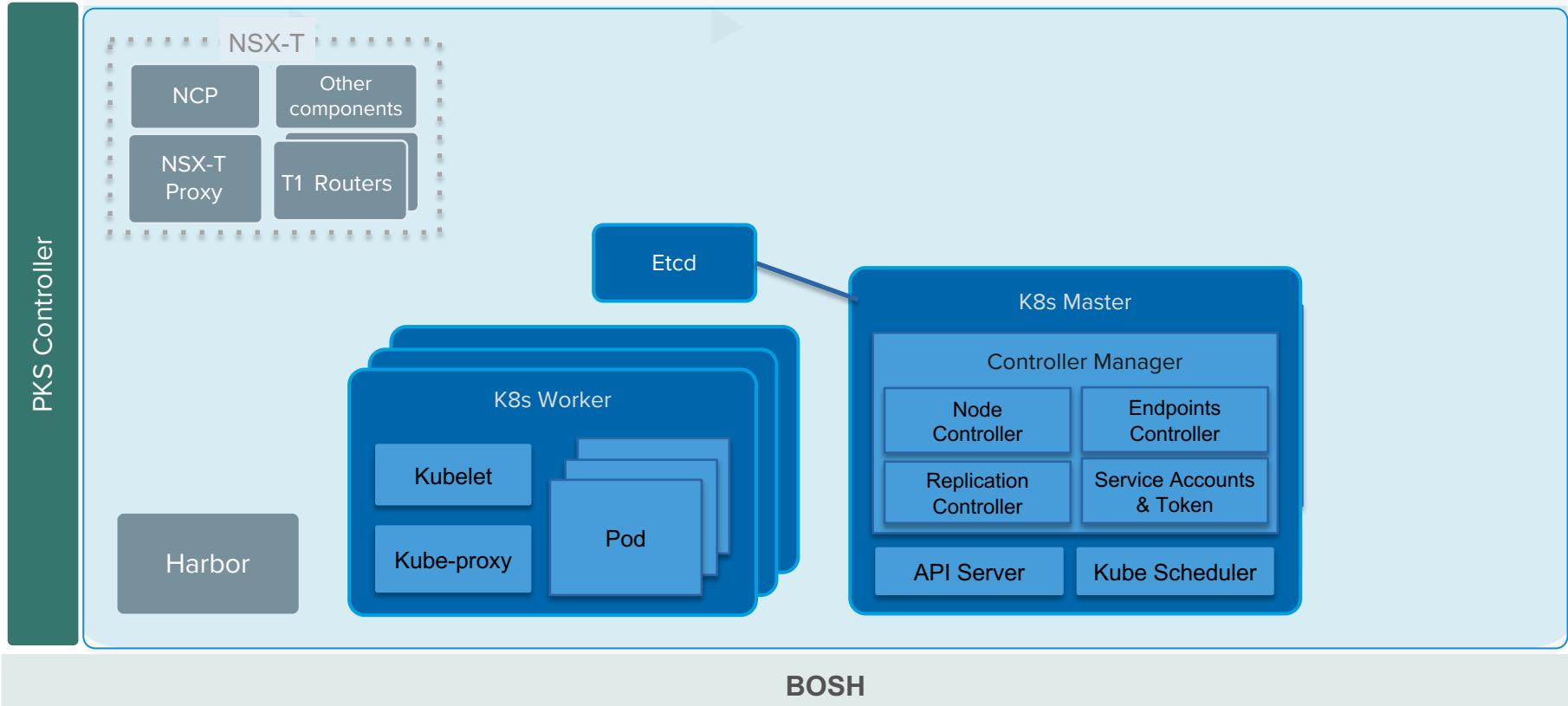
Supports Telegraf input/output plugins



Kubernetes high-level components

- Protobuf
- CNI
- Volume mount

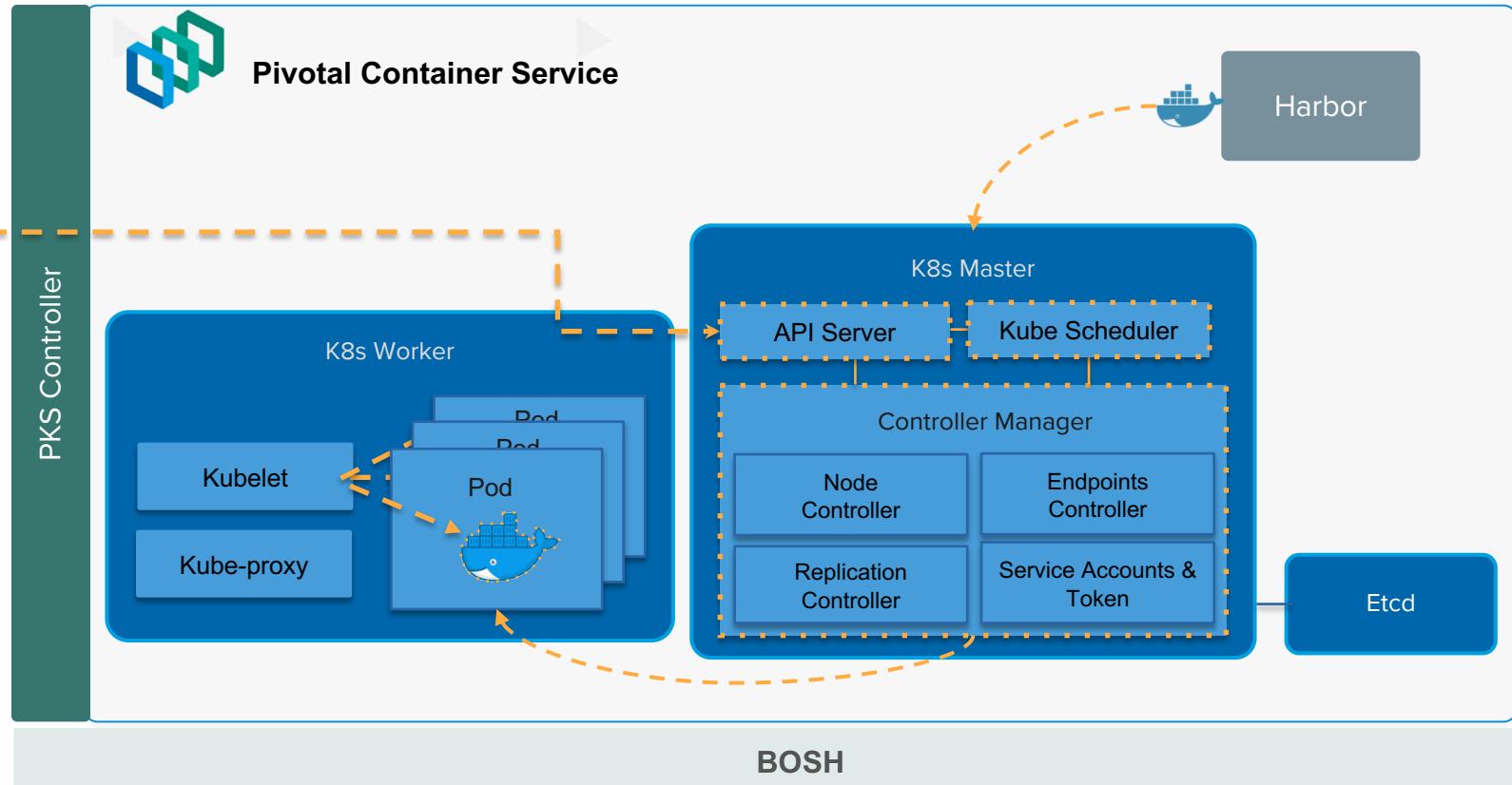




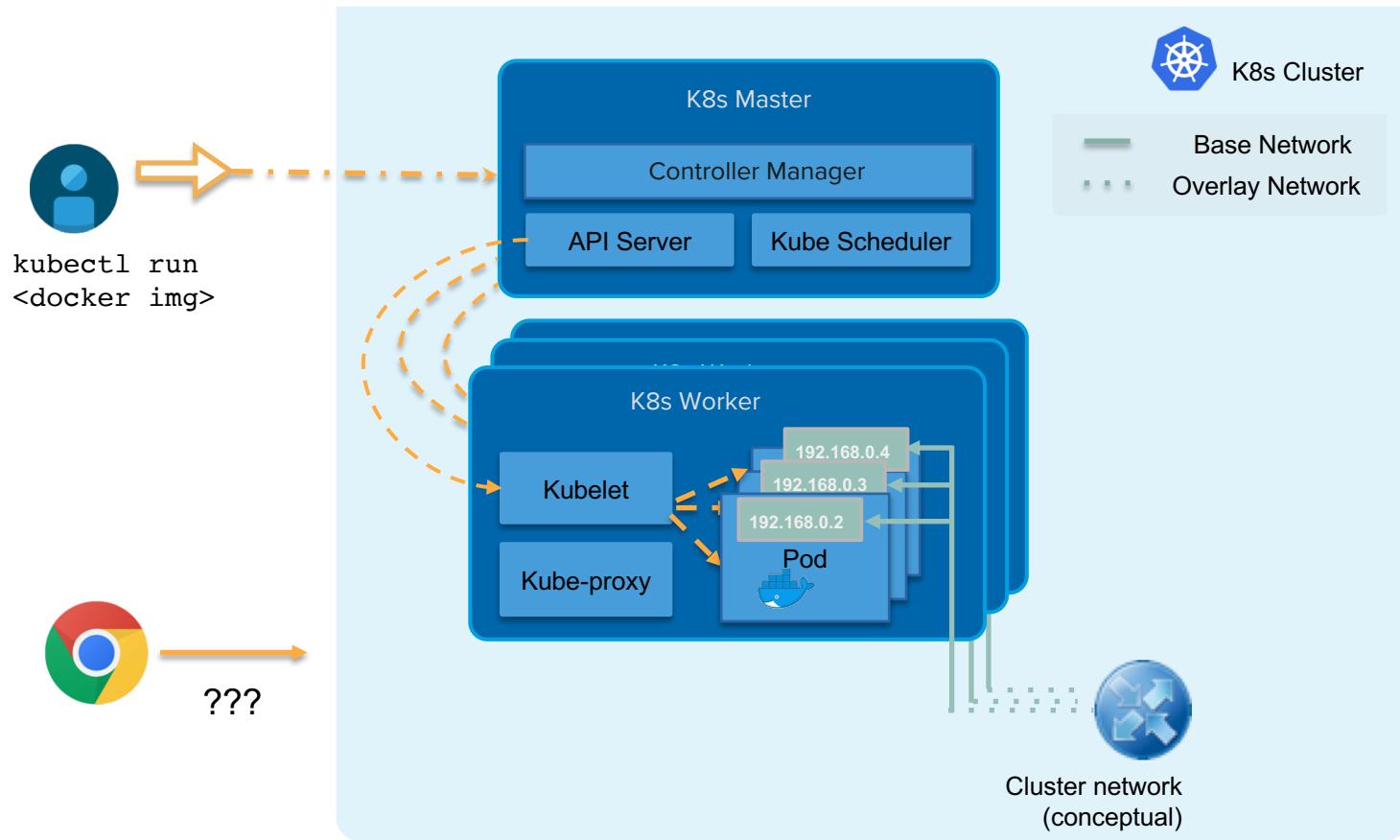
 Included component, but optional usage

  OSS Kubo

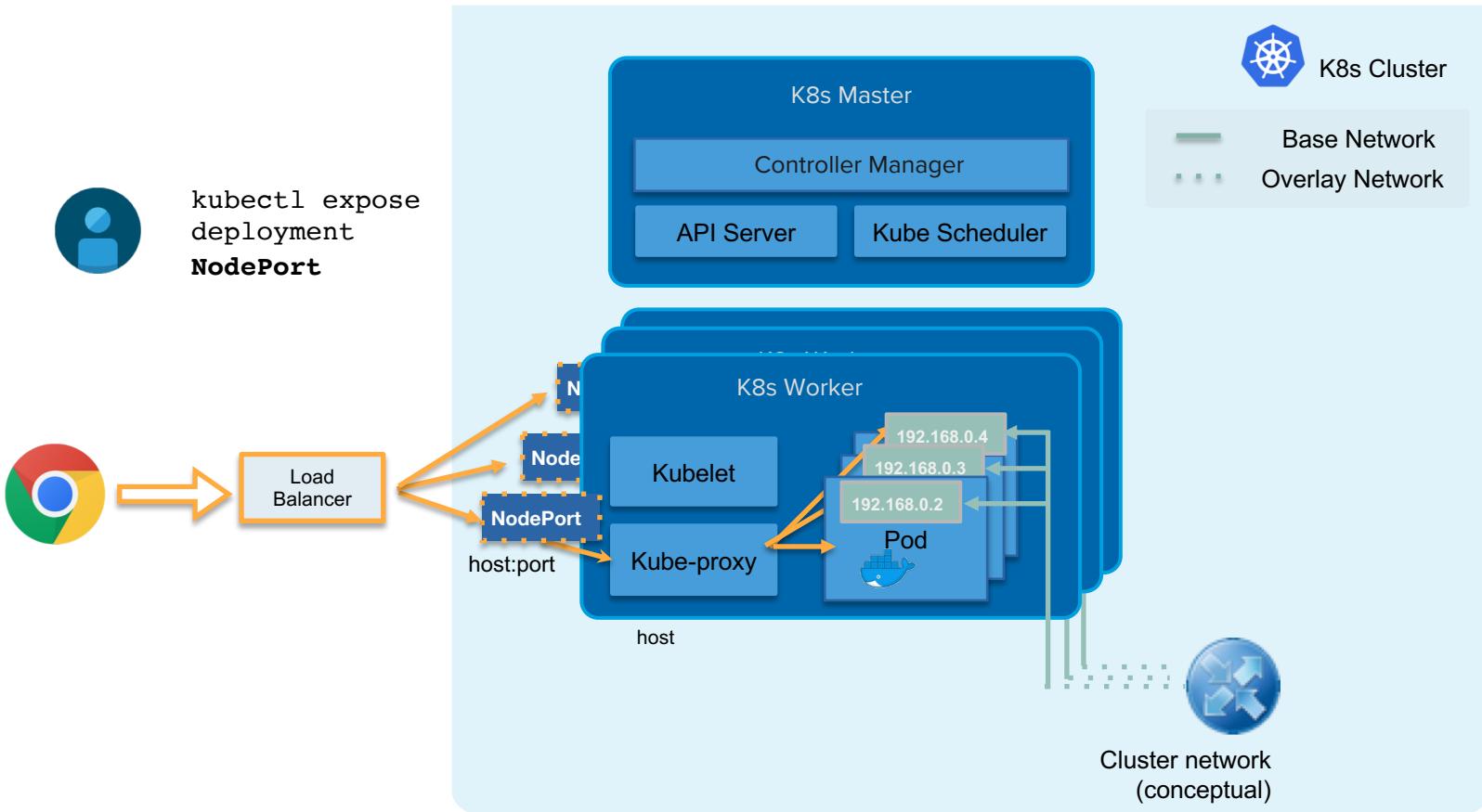
Pushing a Container



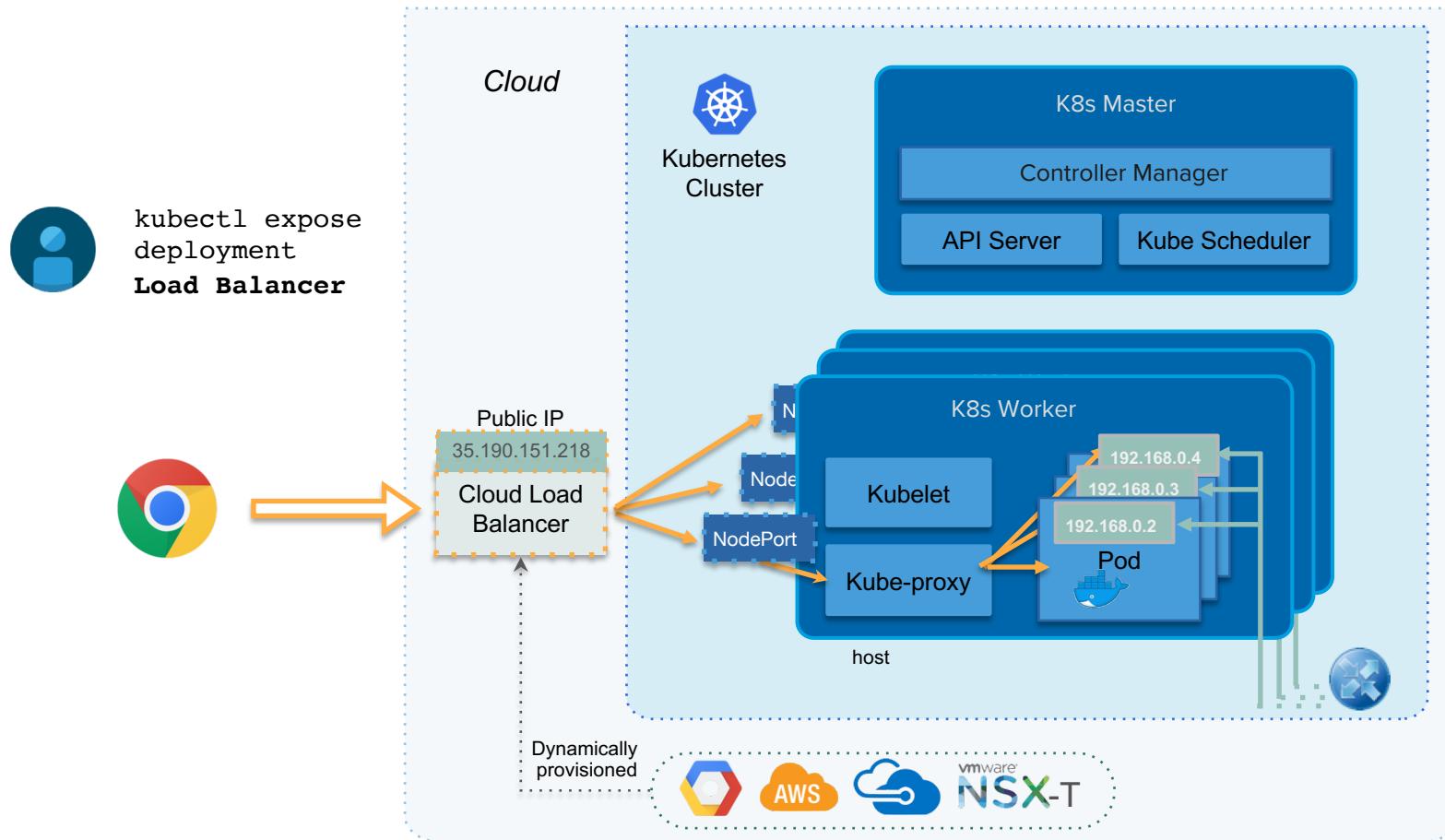
Services and Routing



Services: NodePort



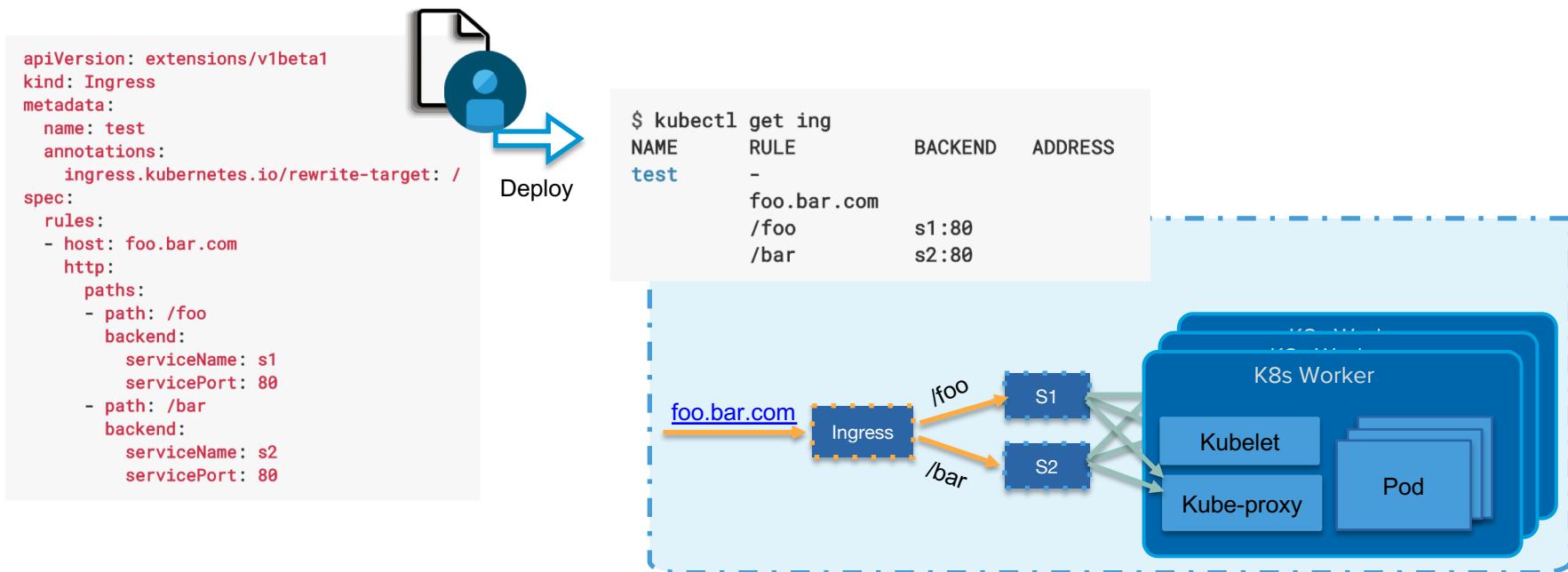
Services: Load Balancer



Services: Ingress

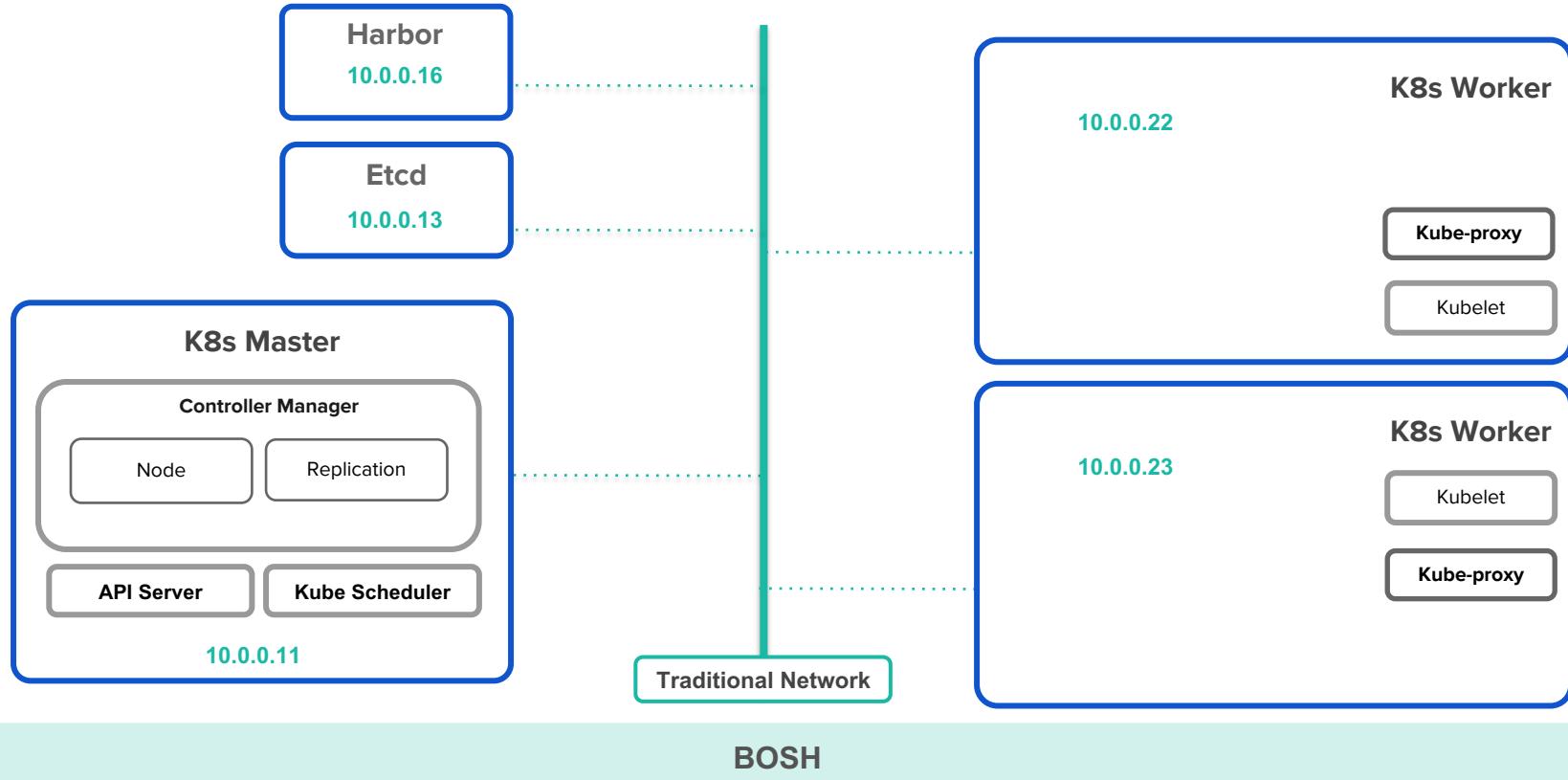
An API object that manages external access to the services in a cluster, typically HTTP.

Ingress can provide **load balancing, SSL termination and name-based virtual hosting**.



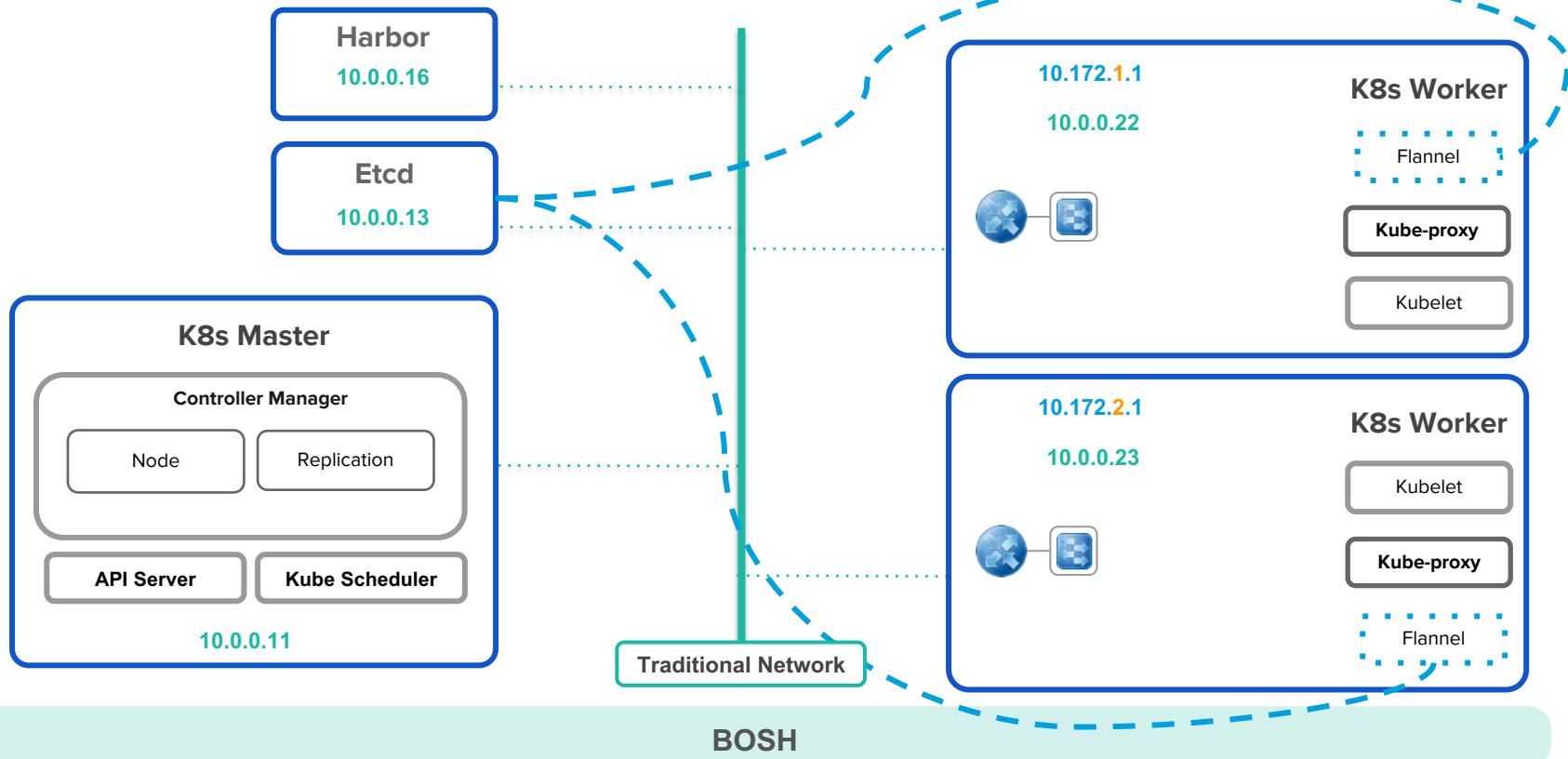


Pivotal Container Service
Networking





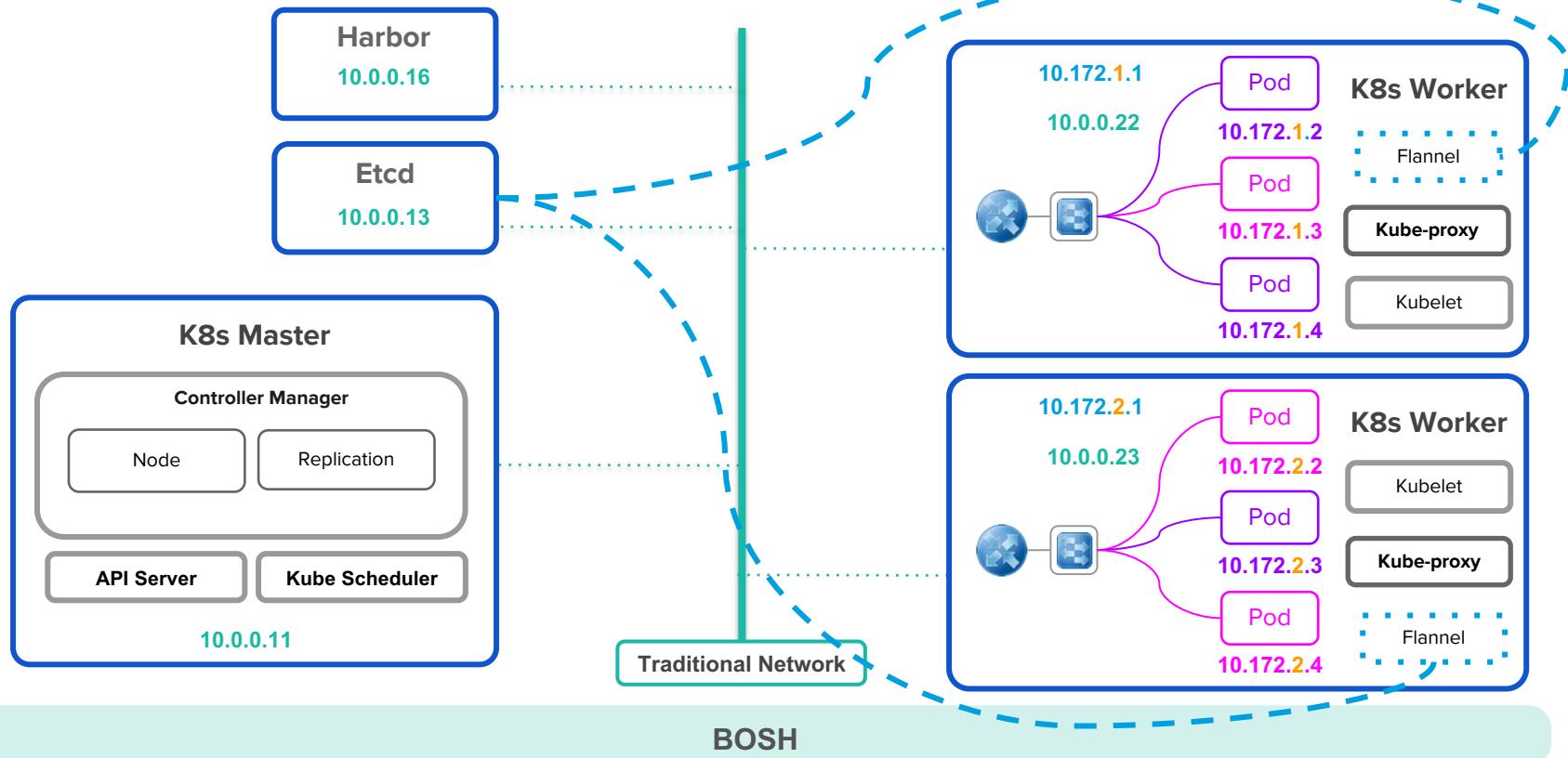
Pivotal Container Service Networking





Pivotal Container Service
Networking

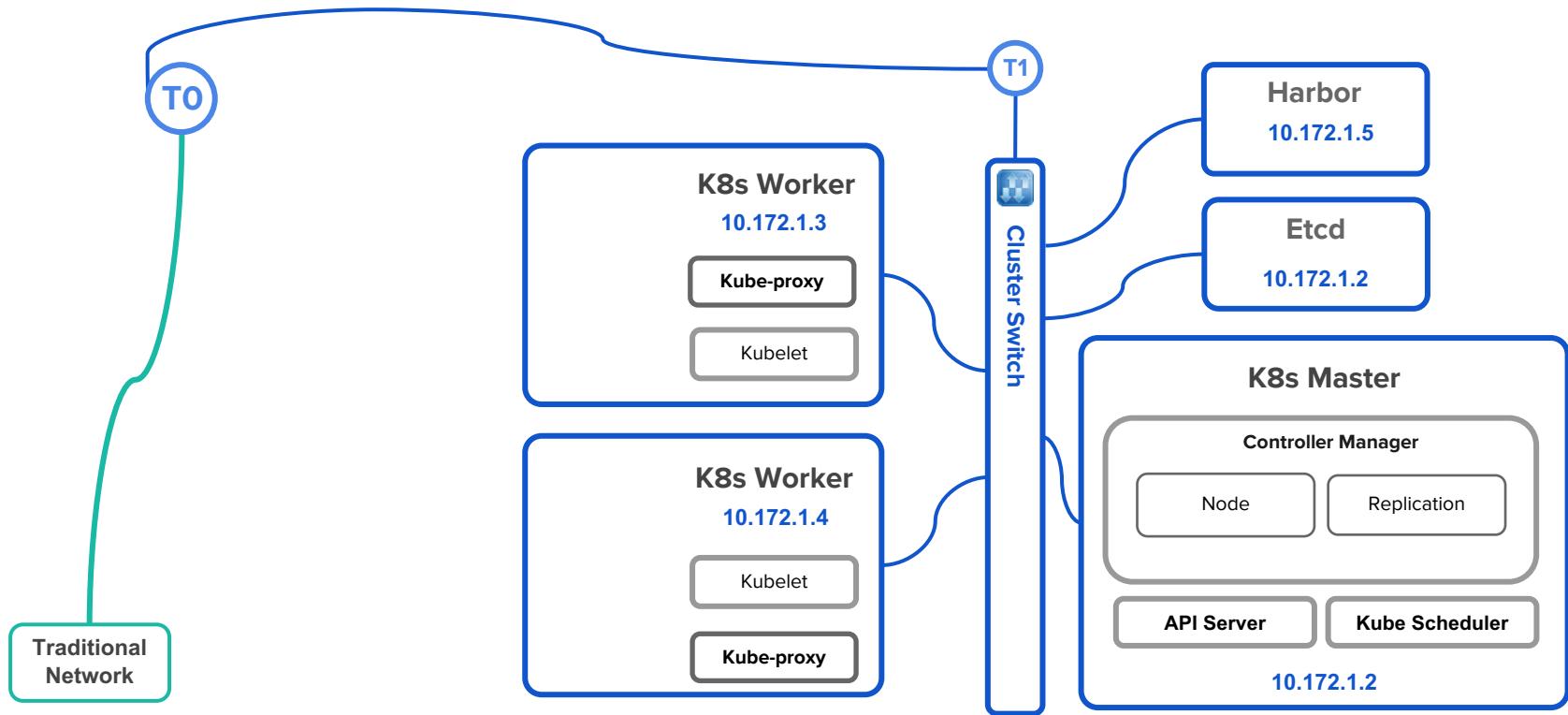
flannel

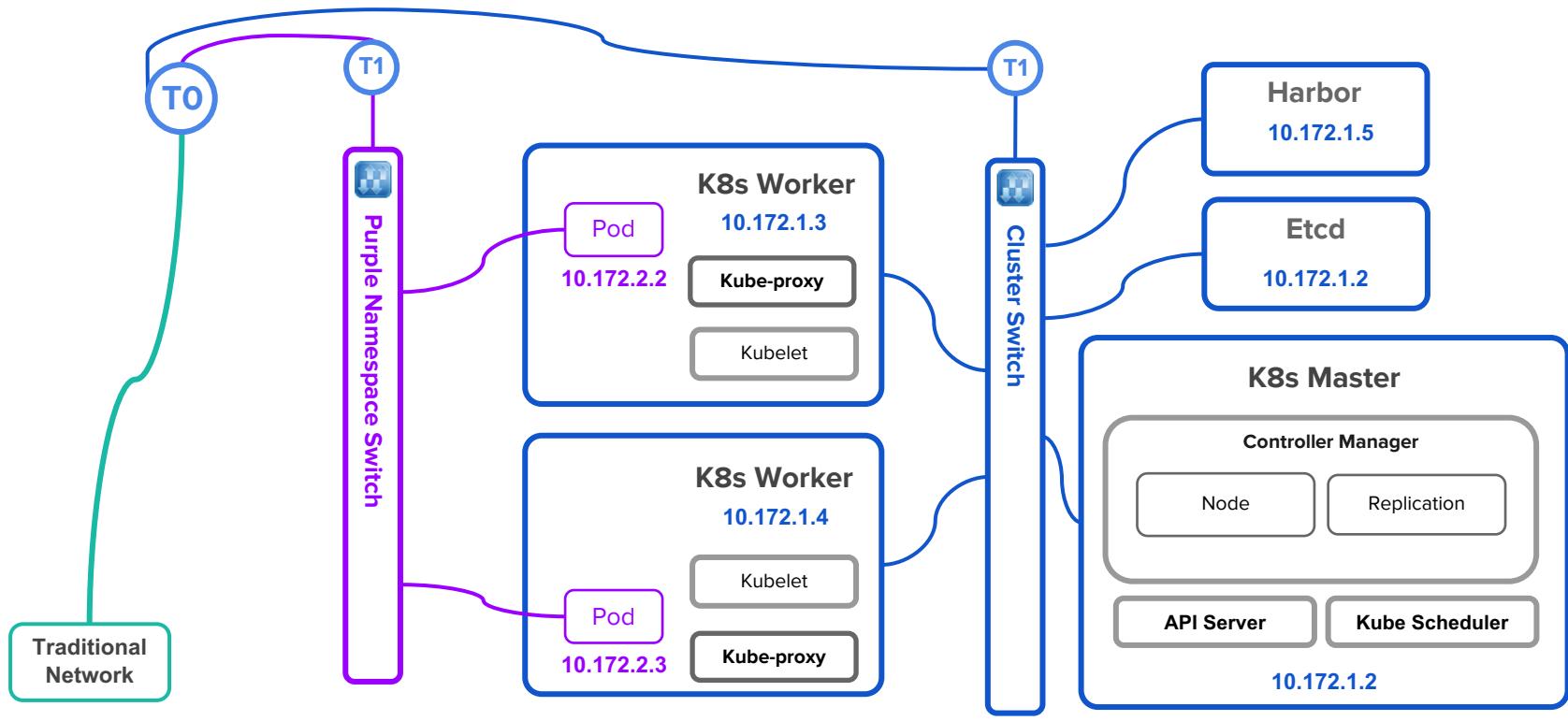


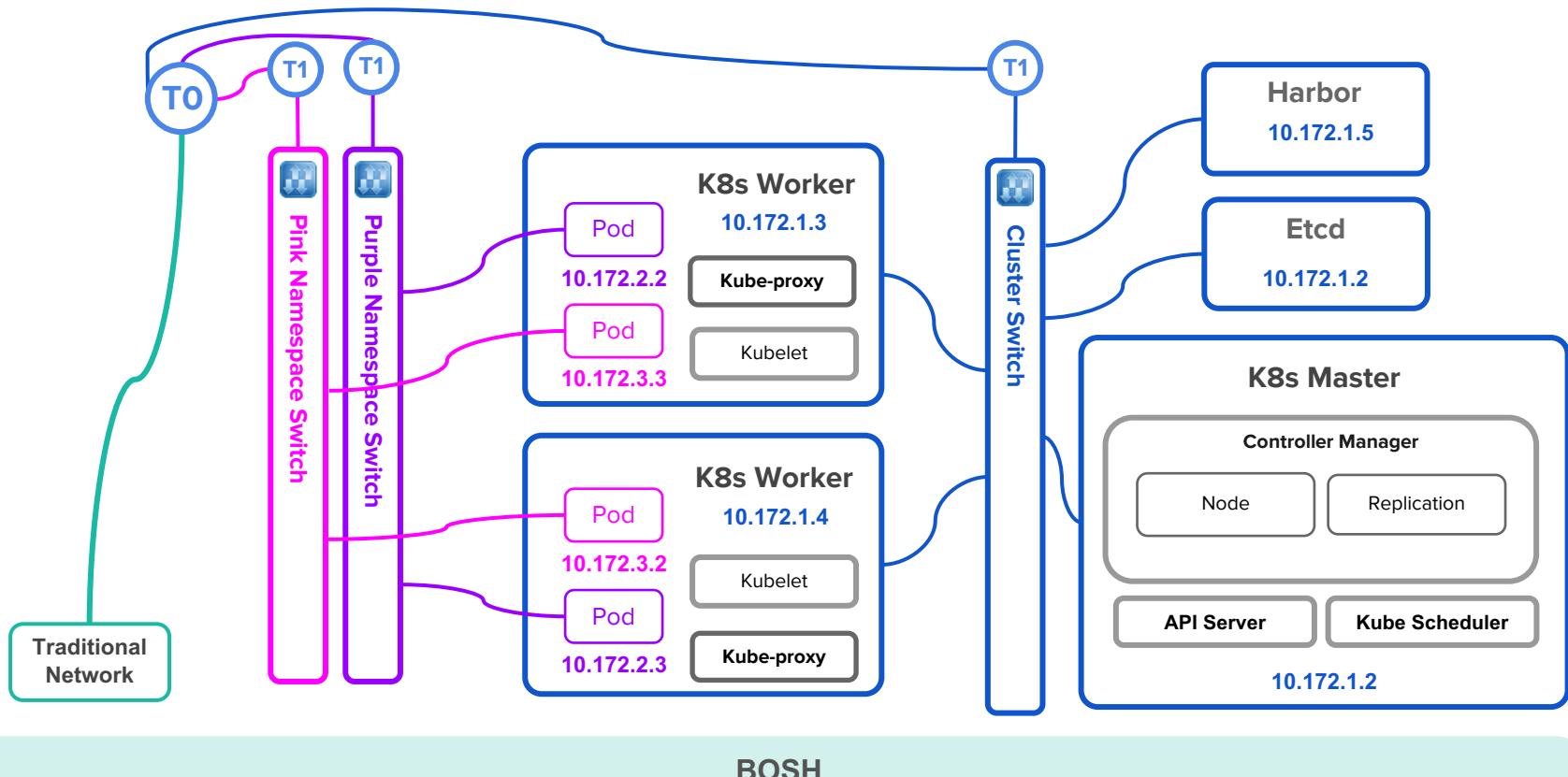


Pivotal Container Service
Networking

vmware
NSX-T

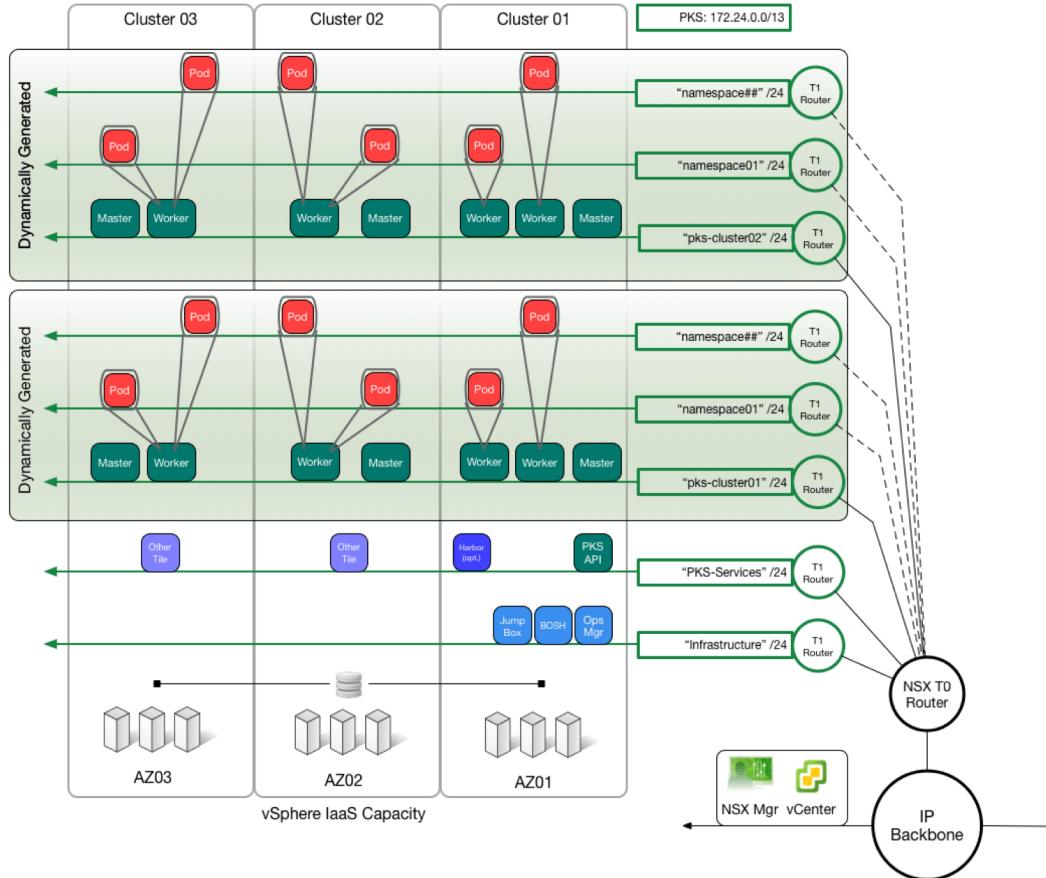




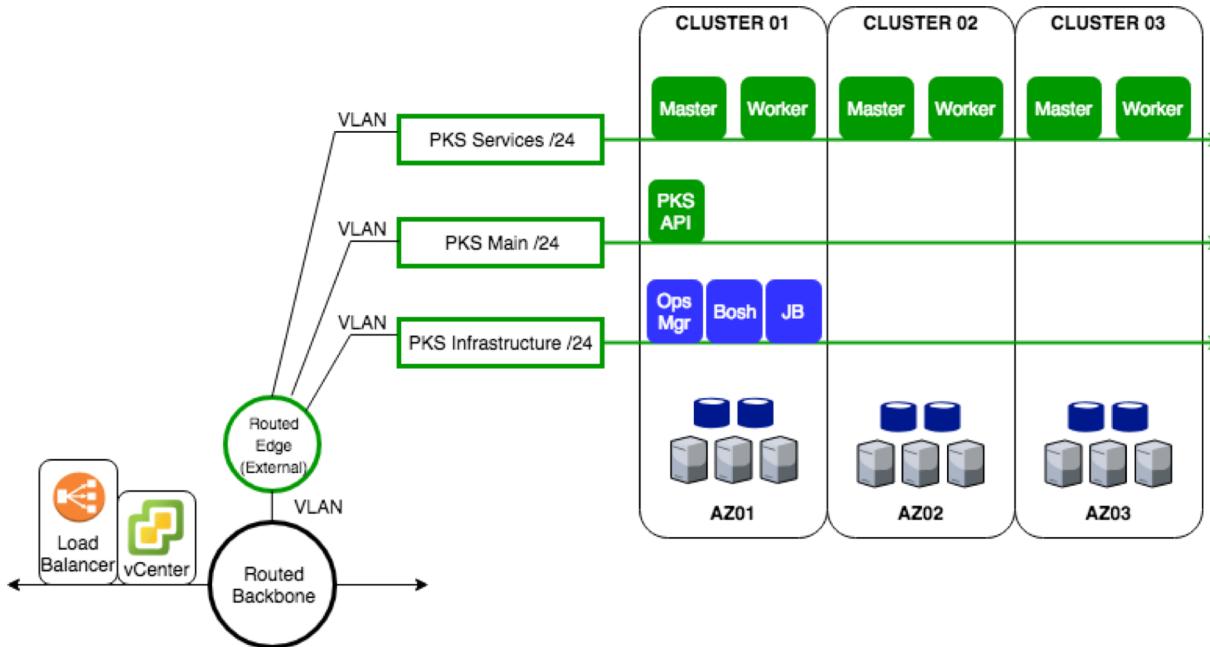


Reference Architectures

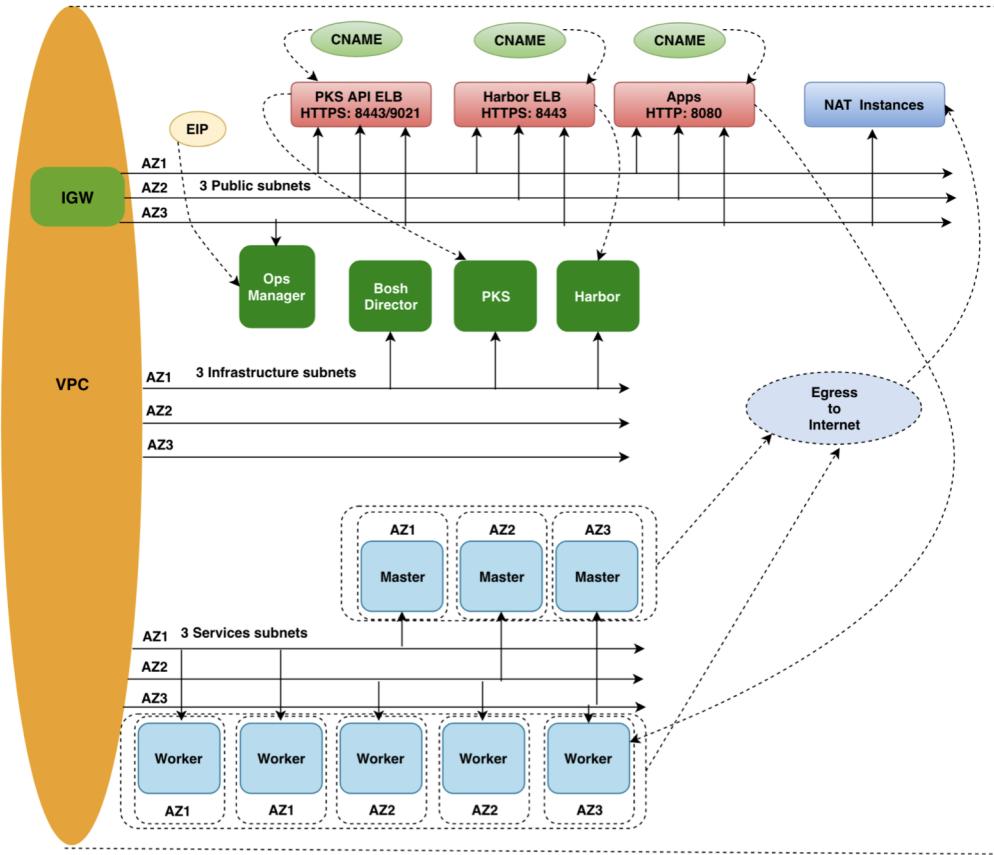
Reference Architecture vSphere w/ NSX-T



Reference Architecture vSphere w/o NSX-T

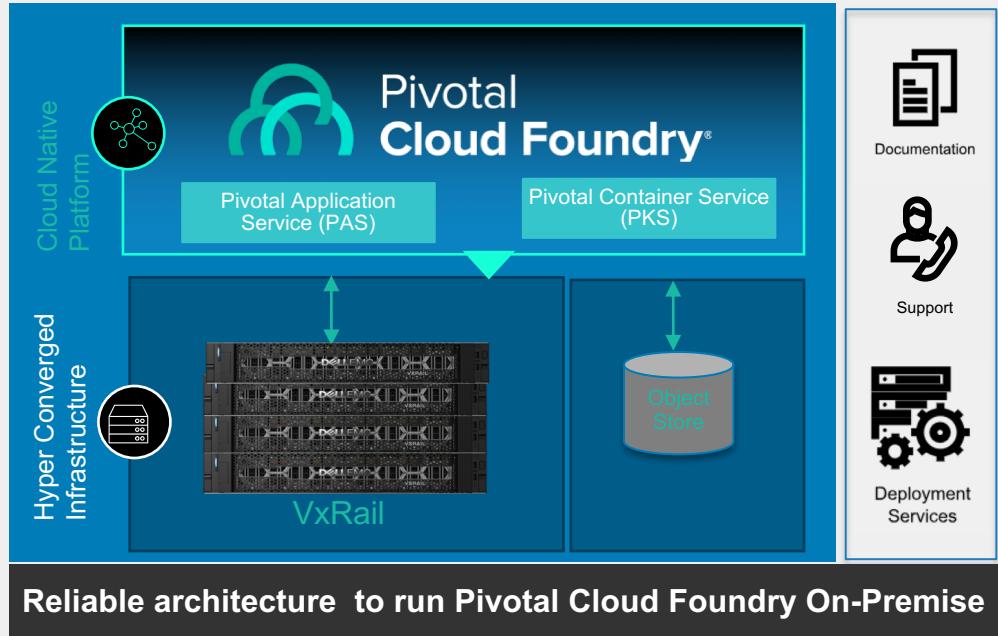


Reference Architecture AWS



Pivotal Ready Architecture

- Pivotal Cloud Foundry tested and **validated** on top of VxRail.
- Hyper-converged Infrastructure which provides robustness, ability to **scale** out and **ease of** lifecycle management
- Documentation and tools enables the **design** and **deploy** of the architecture **in a resilient and reliable way**.
- Product Support.
- Optional **service** can be leveraged by customers to deploy and manage the lifecycle of the Platform.



Deeper Dive

Workload Placement

What workloads should run on PKS?

Stateful workloads

**MongoDB, PostgreSQL,
Cassandra, Spark, Elastic Search**

Commercial (packaged) software

Microsoft SQL Server

Software distributed via Helm chart

Apache Kafka

Apps using non-standard port behavior

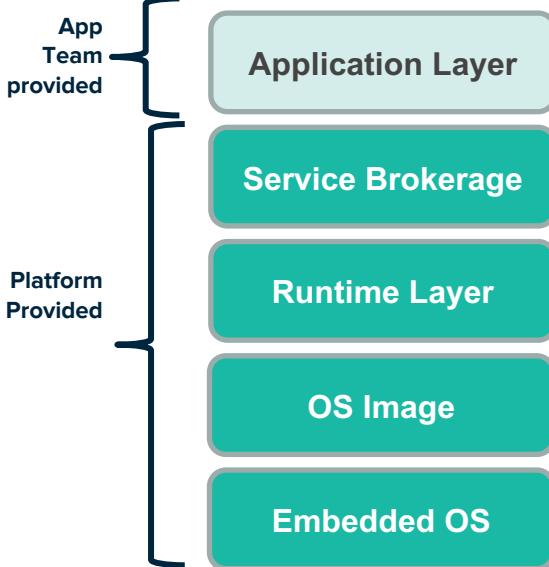
**Custom-built C++ app listening on
3 ports**

Legacy, zero-factor, apps

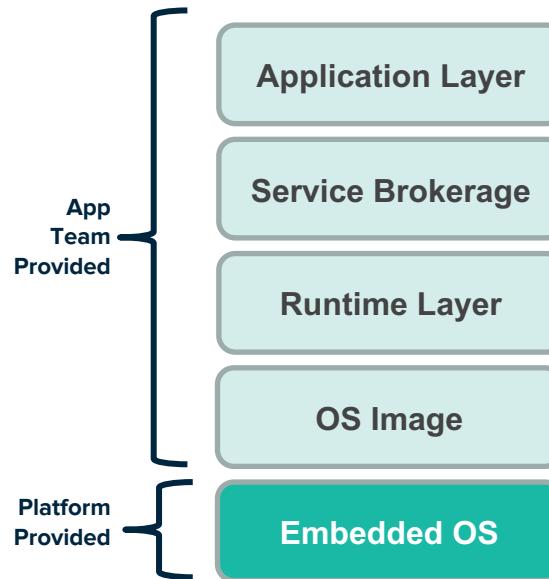
Decide what you want to “own” from security perspective



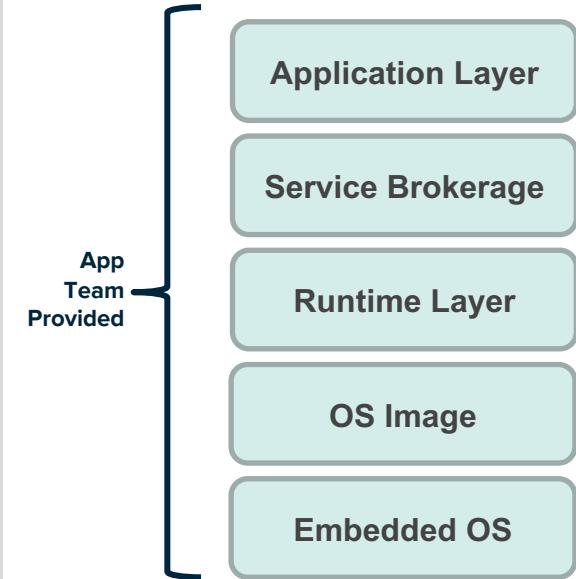
Pivotal
Application Service™



Pivotal
Container Service™



DIY k8s or container stack



Comparing Spring Boot app deployment processes

Spring Boot app deployed to **PKS**

- Compile Spring Boot app
- Choose base Docker image
- Author Dockerfile w/ app entrypoint
- Build Docker image
- Upload image to container registry
- Decide on JVM tuning parameters to use when starting pods
- Create kubernetes deployment config
- Use CI/CD tool or kubectl to apply kubernetes configuration and deploy pods
- Create service to expose pod for users
- Profit!

Snowflake container images
Governance enforced by corporate process

Spring Boot app deployed to **PAS**

- Compile Spring Boot app
- Create manifest.yml to describe the app
- Use CI/CD tool or cf push to deploy
- Profit!

Standardized container images
Governance enforced by the platform

Spring Boot k8s deployment

Step 1. Compile Spring Boot App



Step 2c. Update App entrypoint

```
FROM php:5.6-apache
Run alpine rebase
# Install the PHP extensions we need
RUN apt-get update && apt-get install -y libpng2-dev \
&& docker-php-ext-configure gd --with-png-dir=/usr/include/libpng2 \
&& docker-php-ext-install gd
RUN docker-php-ext-install mysqli
```



```
VOLUME /var/www/html
```



```
ENV WORDPRESS_VERSION 4.2.2
ENV WORDPRESS_UPGRADE_VERSION 4.2.2
ENV WORDPRESS_SHA1 2e7a707e7154ed4a0a0d9f793a8a7e0221
```



```
# A cronjob periodically checks if a newer version is available
RUN curl -o wordpress.tar.gz "https://wordpress.org/latest.tar.gz" && tar -xzf wordpress.tar.gz && mv wordpress/* /var/www/html && rm wordpress.tar.gz && chown -R www-data:www-data /var/www/html/wordpress
```



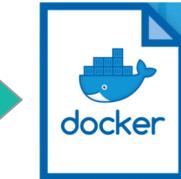
```
COPY docker-entrypoint.sh /entrypoint.sh
```

Step 2. Author Dockerfile



ubuntu
CentOS
alpine

Step 2a. Choose Base Docker image
Step 2b. Add required binaries

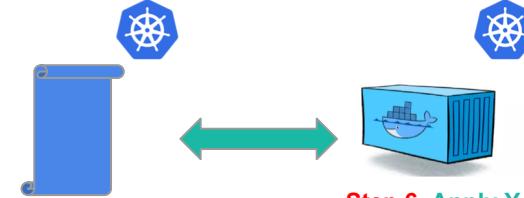


Step 3. Build Docker image

Step 8. User access app



Step 7. Create Service & expose POD to user



Step 6. Apply YAML and deploy POD in cluster



```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: my-nginx-webserver
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: my-nginx-webserver
    spec:
      containers:
        - name: my-nginx-webserver
          image: nginx:alpine
          ports:
            - containerPort: 80
          volumeMounts:
            - name: hostvol
              mountPath: /usr/share/nginx/html
          volumes:
            - name: hostvol
              hostPath:
                path: /home/docker/my_yol
```

Step 5a. Provide JVM tuning parameters

Step 5b. Other config & env variables

Step 4a. docker push

Step 4b. Sign image

Step 4c. Scan image for vulnerabilities



CLI

Step 4. Upload image to Container Registry

Step 5c. Reference registry image to run

Spring Boot PKS deployment

Step 1. Compile Spring Boot App



Step 2c. Update App entrypoint

```
FROM php:5.6-apache  
RUN apt-get update && apt-get install -y libpng2-dev  
&& docker-php-ext-configure gd --with-png-dir=/usr/include/libpng2  
&& docker-php-ext-install gdal  
&& docker-php-ext-install mysqli
```



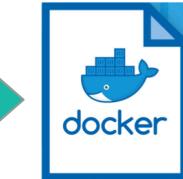
```
VOLUME /var/www/html  
  
ENV WORDPRESS_VERSION 4.2.2  
ENV WORDPRESS_UPSTREAM_VERSION 4.2.2  
ENV WORDPRESS_SHA1 2a7d07e1515ed4a0a050f7935a8a76221  
  
# A few extra variables included in .wp-config.php file to this plugin  
ENV curl -o wordpress.tar.gz SL https://wordpress.org  
ENV curl -o wp-content.tar.gz SL https://wp-content.org  
ENV curl -o wp-content-theme.tar.gz SL https://wp-content-theme.org  
&& rm -rf /var/www/html/*  
&& rm -rf /var/www/html/wp-content/*  
&& rm -rf /var/www/html/wp-content-theme/*  
  
COPY docker-entrypoint.sh /entrypoint.sh
```

Step 2. Author Dockerfile



ubuntu
CentOS
alpine

- Step 2a. Choose Base Docker image
Step 2b. Add required binaries



Step 3. Build Docker image

Step 8. User access app



CLI

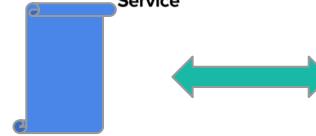


Pivotal
Container
Service



Pivotal
Container
Service

Step 7. Create Service & expose POD to user



Step 6. Apply YAML and deploy POD in cluster



```
apiVersion: extensions/v1beta1  
kind: Deployment  
metadata:  
  name: my-nginx-webserver  
spec:  
  replicas: 3  
  template:  
    metadata:  
      labels:  
        app: my-nginx-webserver  
    spec:  
      containers:  
      - name: my-nginx-webserver  
        image: nginx:alpine  
        ports:  
        - containerPort: 80  
      volumes:  
      - name: hostvol  
        hostPath:  
          path: /home/docker/my-yol
```

Step 5. Author K8s Deployment YAML

Step 5a. Provide JVM tuning parameters
Step 5b. Other config & env variables

Step 5c. Reference registry image to run



Step 4. Upload image to Harbor

Spring Boot PAS deployment

Step 1. Compile App



Step 2a. Update App details

```
FROM php:5.6-apache
RUN alpine curl
# install the PHP extensions we need
RUN apt-get update && apt-get install -y libpng2-dev \
    && docker-php-ext-configure gd --with-png-dir=/usr \
    && docker-php-ext-install gd
RUN docker-php-ext-install mysqli

VOLUME /var/www/html

# The WORDPRESS_VERSION 4.2.2
# The WORDPRESS_UPSTREAM_VERSION 4.2.2
# The WORDPRESS_SHA1 dfa7a7e07e15edea0a0f793a1a7a2111

# upstream partially includes /wordpress, so this gives us
# curl -o wordpress.tar.gz -S https://wordpress.org/
# curl -o "WORDPRESS_UPSTREAM.tar.gz" https://wordpress.org/
# tar -xzf WORDPRESS_UPSTREAM.tar.gz -C /var/www/html/
# rm wordpress.tar.gz
# rm WORDPRESS_UPSTREAM.tar.gz
# chown -R www-data:www-data /var/www/html/wordpress

COPY docker-entrypoint.sh /entrypoint.sh
```

Step 2. Author Manifest file



Pivotal Application Service



Step 3. User access app

Comparing Spring Boot app deployment processes

Deploying a Microservice onto Kubernetes (Minikube)

```
kubectl create -f src/main/kubernetes/configmap.yaml
kubectl create secret generic mysecret --dry-run=client --from-literal=dbusername=secret --from-literal=dbpassword=secret --from-literal=dbpassword-test=secret --from-literal=dbpassword-test-test=secret
[minikube] ~ % kubectl get secrets mysecret -o yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
spec:
  data:
    activeprofiler: k8s
    myparameter: my-parameter-value
    youhostis: OnMinikube
    vcapapplication:
      {
        "application_name": "pcfspringboot-demo"
      }

```

config manifest

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: myconfigmap
data:
  activeprofiler: k8s
  myparameter: my-parameter-value
  youhostis: OnMinikube
  vcapapplication: {
    "application_name": "pcfspringboot-demo"
  }
```

service manifest

```
apiVersion: v1
kind: Service
metadata:
  name: myservice
spec:
  ports:
    type: NodePort
    port: 8080 # expose this service on port 8080
    protocol: TCP
    targetPort: 8080
    selector:
      app: springboot-demo
```

missing a browser friendly route for the outside world to use...

database manifest

```
apiVersion: v1
kind: Service
metadata:
  name: mysql
spec:
  ports:
    - port: 3306
      protocol: TCP
      selector:
        app: mysql
  ...
  annotations:
    extensions/v1beta1.sinceDeployment:
      minReadySeconds: 60
      progressDeadlineSeconds: 300
      selector:
        replicas:
          - app: springboot-demo
        strategy:
          rollingUpdate:
            maxSurge: 10%
            maxUnavailable: 0%
          type: RollingUpdate
      template:
        metadata:
          annotations:
            scheduler.alpha.kubernetes.io/change-cause: "Deploying springboot-demo to v0.0.1-SNAPSHOT"
        labels:
          app: springboot-demo
          version: "0.0.1-SNAPSHOT"
        spec:
          containers:
            - name: springboot-demo
              image: benvulcock/pcfspringboot-demo:0.0.1-SNAPSHOT
              imagePullPolicy: IfNotPresent
              ports:
                containerPort: 8080
              protocol: TCP
              livenessProbe:
                httpGet:
                  path: /health
                  port: 8080
                initialDelaySeconds: 10
                timeoutSeconds: 10
                periodSeconds: 60
                failureThreshold: 3
              readinessProbe:
                httpGet:
                  path: /health
                  port: 8080
                initialDelaySeconds: 30
                periodSeconds: 10

```

you need to create and maintain the database and other resources your application needs

in this case a MySQL database (singleton, not HA) is being added for testing purposes

app manifest

```
apiVersion: extensions/v1beta1
sinceDeployment:
  metadata:
    name: mydeployment
    labels:
      app: springboot-demo
    spec:
      replicas: 1
      minReadySeconds: 60
      progressDeadlineSeconds: 300
      selector:
        replicas:
          - app: springboot-demo
        strategy:
          rollingUpdate:
            maxSurge: 10%
            maxUnavailable: 0%
          type: RollingUpdate
      template:
        metadata:
          annotations:
            scheduler.alpha.kubernetes.io/change-cause: "Deploying springboot-demo to v0.0.1-SNAPSHOT"
        labels:
          app: springboot-demo
          version: "0.0.1-SNAPSHOT"
        spec:
          containers:
            - name: springboot-demo
              image: benvulcock/pcfspringboot-demo:0.0.1-SNAPSHOT
              imagePullPolicy: IfNotPresent
              ports:
                containerPort: 8080
              protocol: TCP
              livenessProbe:
                httpGet:
                  path: /health
                  port: 8080
                initialDelaySeconds: 10
                timeoutSeconds: 10
                periodSeconds: 60
                failureThreshold: 3
              readinessProbe:
                httpGet:
                  path: /health
                  port: 8080
                initialDelaySeconds: 30
                periodSeconds: 10

```

Takes about 2hrs to configure and run an app

Takes about 2hrs to configure and run an app

Deploying a Microservice onto PCF

```
cf create-service p-config-server standard config -c config-server-setup.json
cf create-service cleardb spark mysql
cf create-service cloudant lemur rabbit
cf create-service p-service-registry standard registry
cf create-service rediscloud gumb redis
cf create-service p-circuit-breaker-dashboard standard breaker
```

app manifest

```
applications:
  - name: pcf-springboot-demo
    timeout: 180
    memory: 1G
    disk_quota: 1G
    path: build/libs/pcf-springboot-demo-0.0.1-SNAPSHOT.jar
  services:
    - mysql
    - rabbit
    - config
    - registry
    - redis
    - breaker
```

cf push
cf scale -i 3

Pivotal Cloud Foundry

connection details and creds are generated automatically

no built in management of offer secrets as yet

no blue/green as standard

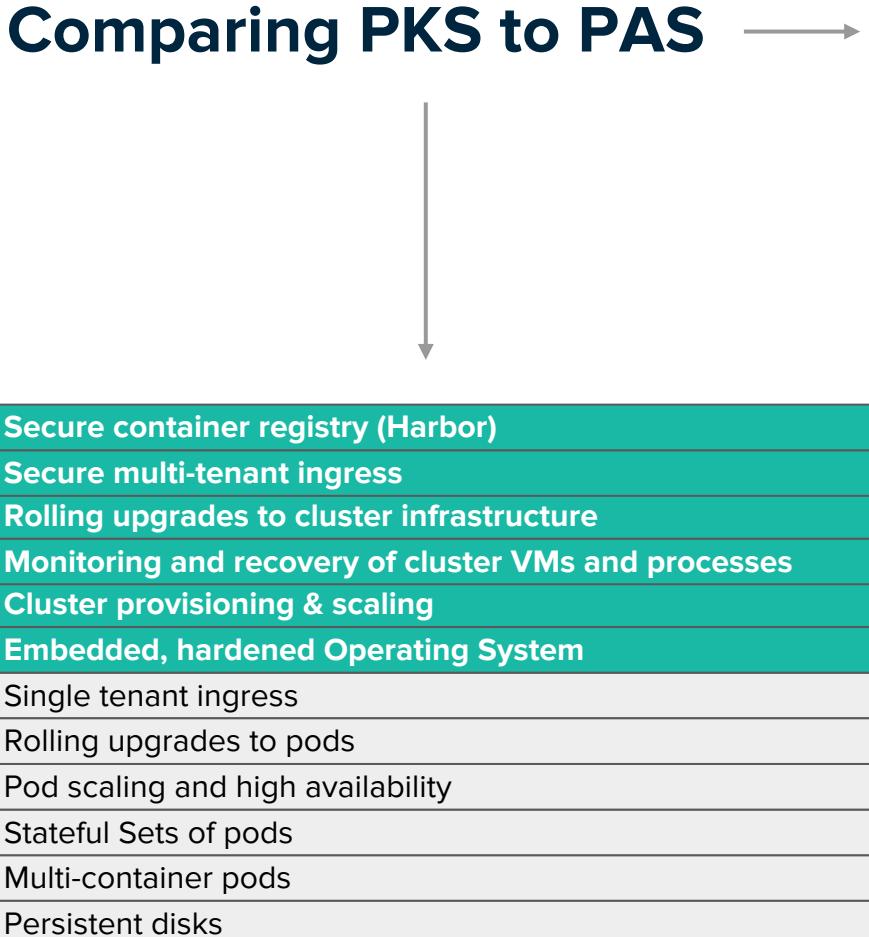
Takes about 5 mins to configure and run an app

Takes about 5 mins to configure and run an app

connection details and creds are generated automatically

no built in management of offer secrets as yet

Comparing PKS to PAS



Secure container registry (Harbor)	Native .NET and Windows support
Secure multi-tenant ingress	Built-in managed services for RabbitMQ, MySQL, PCC
Rolling upgrades to cluster infrastructure	Spring Cloud Services
Monitoring and recovery of cluster VMs and processes	Multi-tenancy through Orgs, Spaces, and Quotas
Cluster provisioning & scaling	IdM components for OAuth, SAML, LDAP
Embedded, hardened Operating System	Marketplace and secure service bindings
Single tenant ingress	Built in load balancing with flexible route bindings
Rolling upgrades to pods	Metrics and aggregated logging
Pod scaling and high availability	Droplets automatically created by buildpacks
Stateful Sets of pods	Interceptable routes with Route Services
Multi-container pods	Secure container registry (Harbor or Blob Store)
Persistent disks	Secure multi-tenant ingress (Go Router)
	Rolling upgrades to cluster infrastructure
	Monitoring and recovery of cluster VMs and processes
	Cluster provisioning and scaling
	Embedded, hardened Operating System
	Single tenant ingress (Go Router + Isolation Segments)
	Rolling upgrades to containers (blue/green deploy)
	Container scaling and high availability
-	-
-	-
	Persistent disks (Volume Services)



Pivotal®

Transforming How The World Builds Software