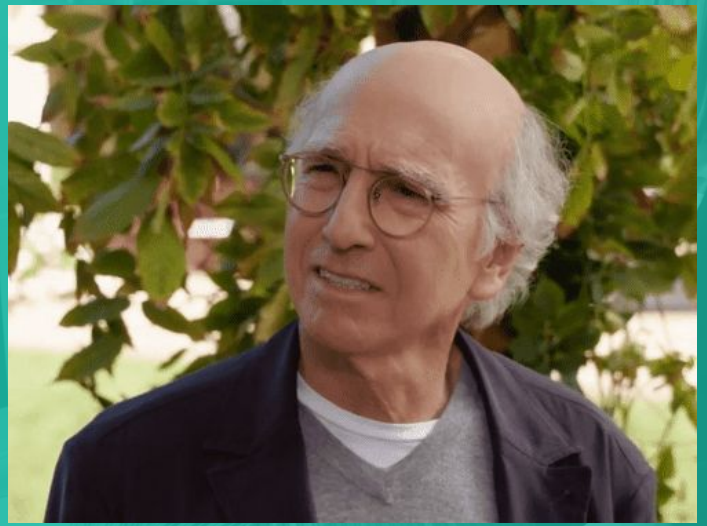
Pivotal

# Spring into Kubernetes



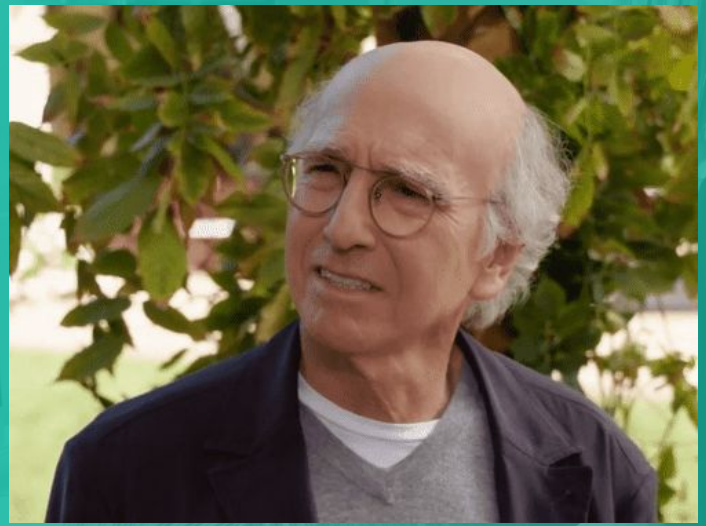
Paul Czarkowski

**pczarkowski@pivotal.io**

Twitter: @pczarkowski

Pivotal

# Spring into Kubernetes

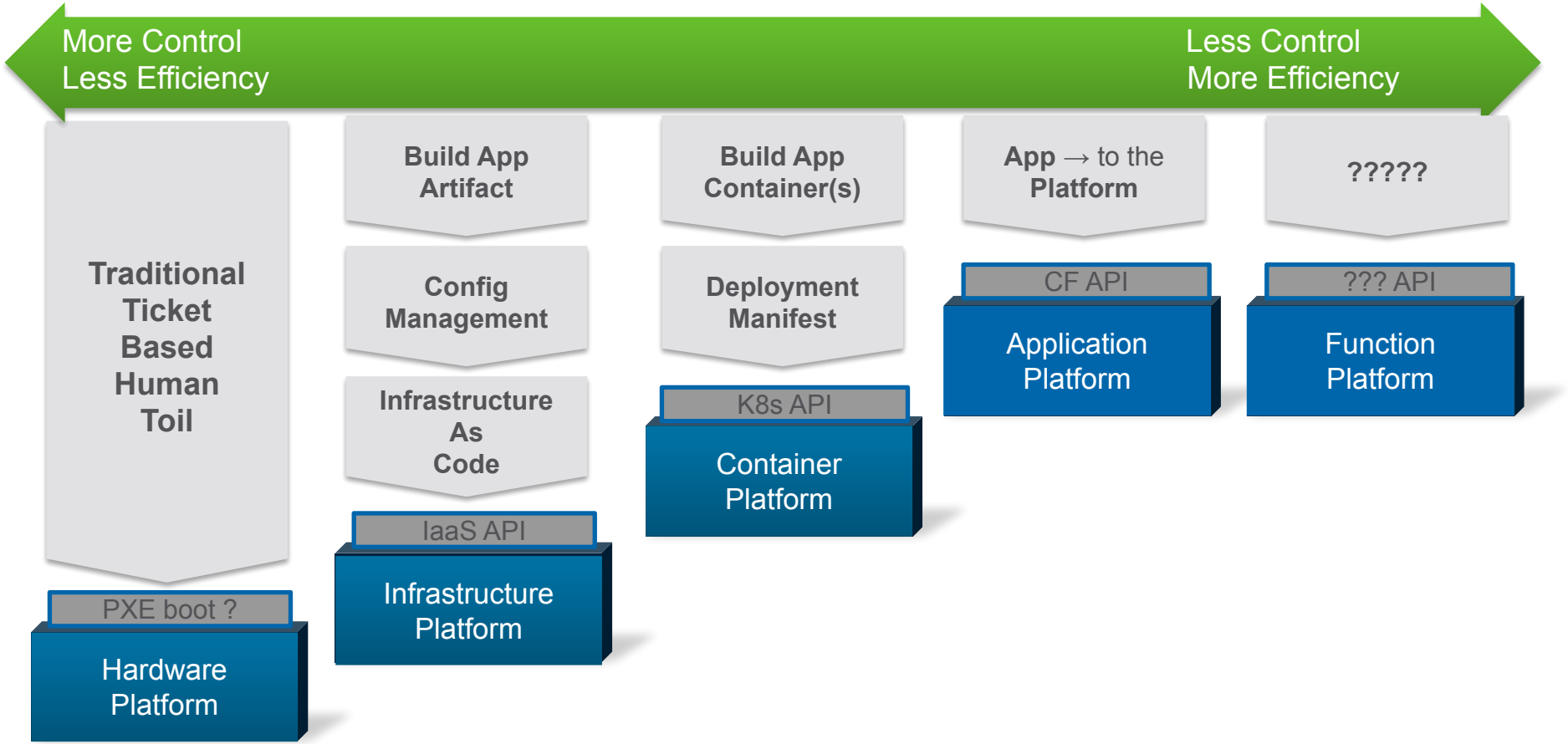


Paul Czarkowski

**pczarkowski@pivotal.io**

Twitter: @pczarkowski

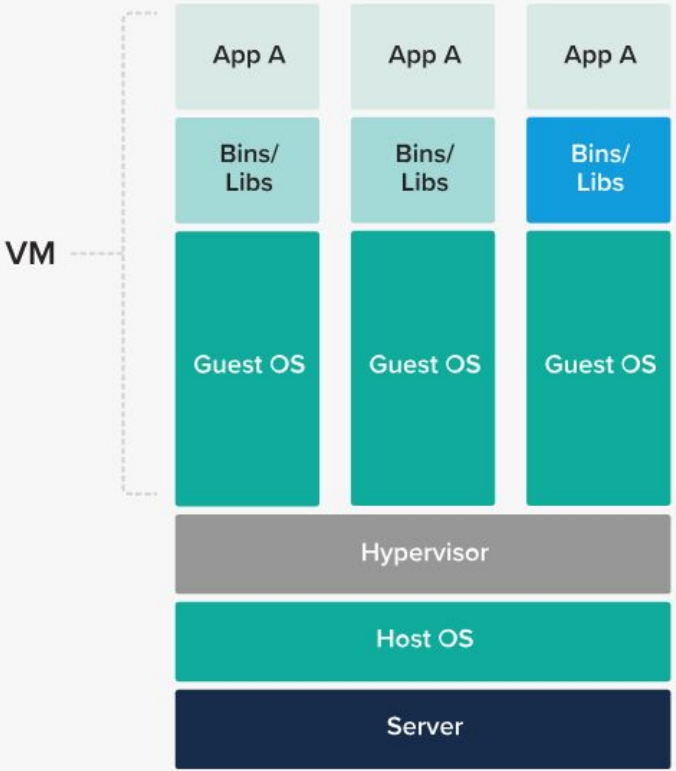

"Kubernetes is named after the greek god of spending money on cloud services" - @QuinnyPig


More Control
Less Efficiency
Less Control
More Efficiency
Traditional Ticket Based Human Toil
PXE boot ?
Hardware Platform
Build App Artifact
Config Management
Infrastructure As Code
IaaS API
Infrastructure Platform
Build App Container(s)
Deployment Manifest
K8s API
Container Platform
App → to the Platform
CF API
Application Platform
?????
??? API
Function Platform

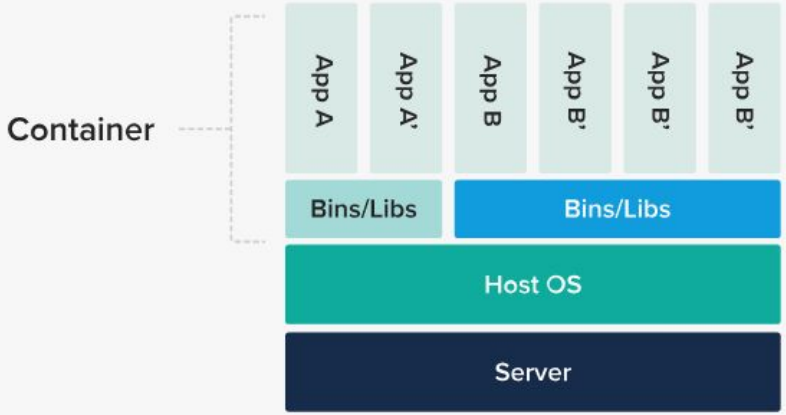

Pivotal

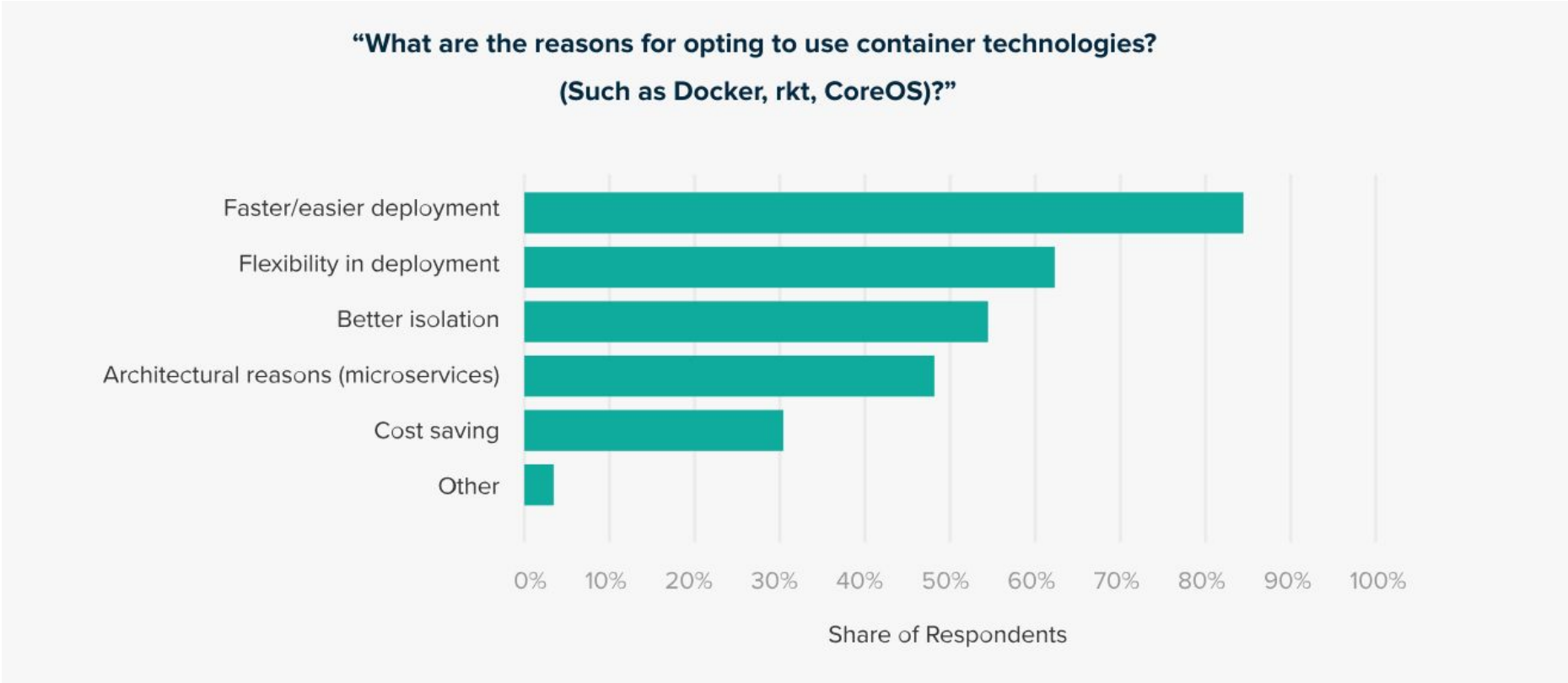

docker

# Containers Isolate and Abstract Resources




- Apps run in a container
- Containers allow multiple apps to run on a single virtual machine (VM)
- Maximizes utilization without OS overhead






*Saurabh Gupta. "Containers and Pivotal Cloud Foundry" 2016.*

```
FROM maven:3.6-jdk-11-slim as BUILD
COPY . /src
WORKDIR /src
RUN mvn install -DskipTests

FROM openjdk:11.0.1-jre-slim-stretch
EXPOSE 8080
WORKDIR /app
ARG JAR=hello-0.0.1-SNAPSHOT.jar

COPY --from=BUILD /src/target/$JAR /app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

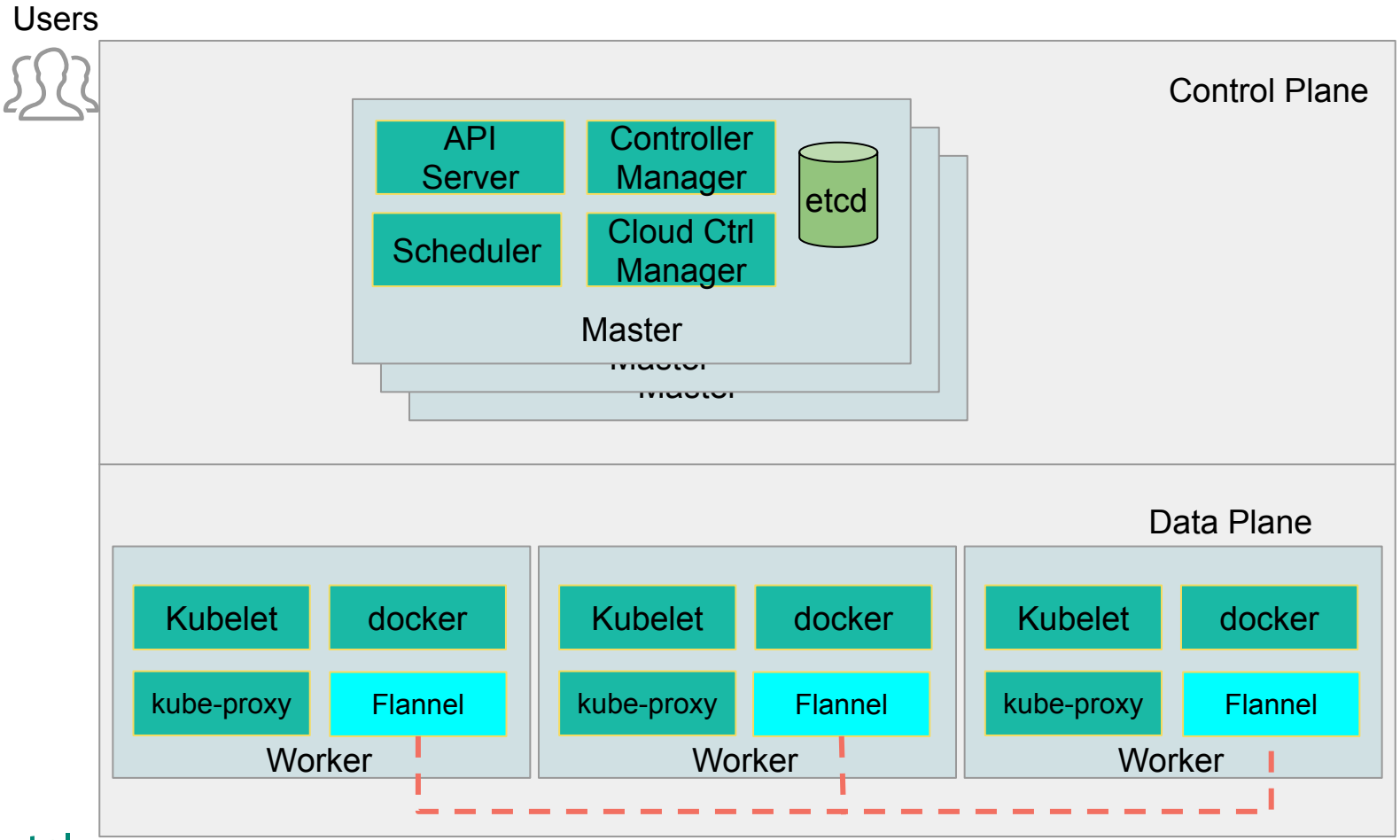

MakeAGIF.com

```
$ docker build -t paulczar/hello .

$ docker push paulczar/hello

$ docker pull paulczar/hello

$ docker run -d -p 8080:8080 paulczar/hello
```

```
<plugin>
  <groupId>com.google.cloud.tools</groupId>
  <artifactId>jib-maven-plugin</artifactId>
  <version>1.6.1</version>
  <configuration>
    <to>
      <image>myimage</image>
    </to>
  </configuration>
</plugin>
```


imgIX
imgIX
imgIX
imgIX


Pivotal


Users
Control Plane
API Server
Controller Manager
etcd
Scheduler
Cloud Ctrl Manager
Master
Data Plane
Kubelet
docker
kube-proxy
Flannel
Worker
Kubelet
docker
kube-proxy
Flannel
Worker
Kubelet
docker
kube-proxy
Flannel
Worker


Controllers


Desired State
Actual State

# Unix Philosophy:

# Do one thing. Do it well.


$ kubectl

# Imperative

```
$ kubectl run hello \
    --image=paulczar/go-hello-world

$ kubectl scale hello \
    --replicas=3

$ kubectl create service clusterip \
    hello --tcp=80:80
```

# Declarative

```
$ kubectl apply -f hello-world.yaml
```

Pivotal

# Declarative

# Vs

# Imperative


Pivotal

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: hello
spec:
  containers:
  - image: paulczar/go-hello-world
    imagePullPolicy: Always
    name: hello
```


Pivotal

- Pods
- Services
- Volumes

# POD


one or more containers that share
a network and storage
Pivotal


the minimum scalable unit
of your application


$ kubectl

```
$ kubectl create deployment hello \
    --image=paulczar/hello
```

Pivotal

- kubectl run created a *deployment* “deployments.apps/hello”

```
NAME                  DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/hello   1         1         1            1           1m
```

- The deployment created a *replicaset* “replicaset.apps/hello-64f6bf9dd4”

```
NAME                              DESIRED   CURRENT   READY     AGE
replicaset.apps/hello-64f6bf9dd4   1         1         1         1m
```

- Which created a *pod* “pod/hello-64f6bf9dd4-tq5dq”

```
NAME                          READY     STATUS              RESTARTS   AGE
pod/hello-64f6bf9dd4-tq5dq   1/1       Running             0          2s
```

```
$ kubectl scale --replicas=3 \
    deployment/hello
```

```
$ kubectl scale --replicas=3 deployment/hello
deployment.extensions/hello scaled

$ kubectl get all
NAME                         READY     STATUS              RESTARTS   AGE
pod/hello-64f6bf9dd4-2bndq   1/1       Running             0          15m
pod/hello-64f6bf9dd4-4kq9l   0/1       ContainerCreating   0          2s
pod/hello-64f6bf9dd4-8lkcs   1/1       Running             0          5s

NAME                    DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/hello   3         3         2            3           16m

NAME                               DESIRED   CURRENT   READY     AGE
replicaset.apps/hello-64f6bf9dd4   3         3         2         16m
```

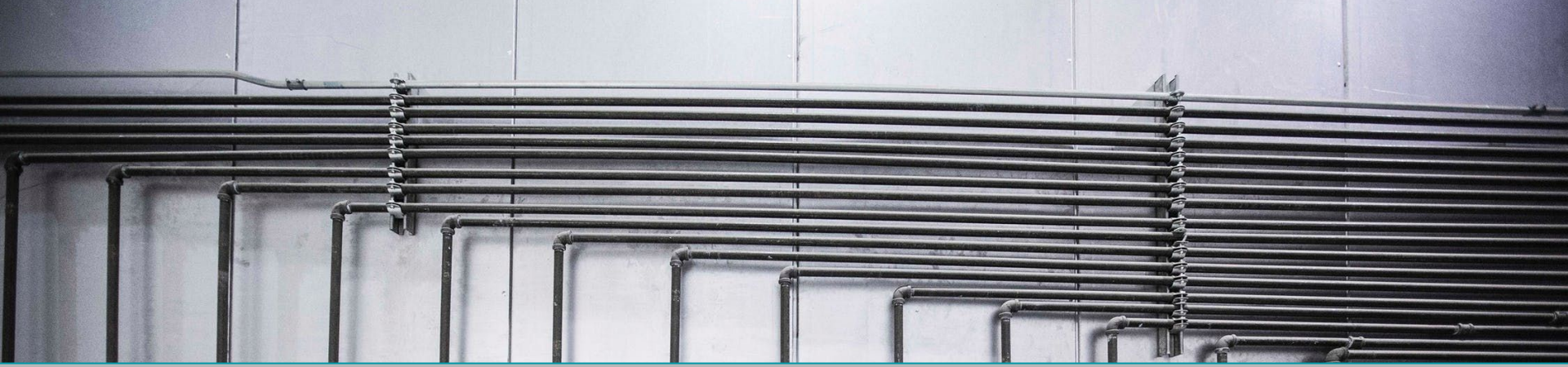
```
$ kubectl set env deployment/hello \
      --env “MESSAGE=Hello Krakow”
```

```
$ kubectl get all

NAME                          READY     STATUS    RESTARTS   AGE
pod/hello-5c75b546c7-4lwnn    1/1       Running   0          1m
pod/hello-5c75b546c7-bwxxq    1/1       Running   0          1m
pod/hello-5c75b546c7-sl2pg    1/1       Running   0          1m

NAME                     DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/hello    3         3         3            3           23m

NAME                                DESIRED   CURRENT   READY     AGE
replicaset.apps/hello-5c75b546c7    3         3         3         1m
replicaset.apps/hello-64f6bf9dd4    0         0         0         23m
```

```
$ kubectl get deployment hello \
      -o yaml
```

```
$ kubectl port-forward deployment/hello 8080
Forwarding from 127.0.0.1:8080 -> 8080

$ curl localhost:8080
<html><head><title>HELLO I LOVE YOU!!!!</title></head><body>HELLO I LOVE
YOU!!!!!</body></html>
```

Pivotal



# Service

```
$ kubectl expose deployment \
    hello --type=LoadBalancer \
      --port 80 --target-port 8080
```

**kubectl expose deployment hello**

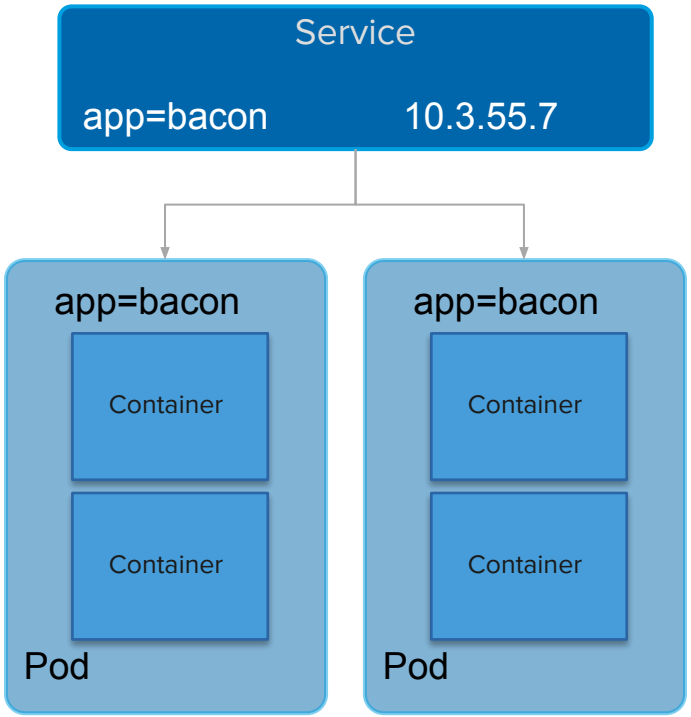
- creates a *service* with a *ClusterIP* that acts as an internal loadbalancer to all *pods* in the “hello” *deployment*

**--type=LoadBalancer**

- Creates a *NodePort*
- Configures a *LoadBalancer* to access the *pods* via the *NodePort*

```
$ kubectl get services
NAME      TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)        AGE
hello     LoadBalancer   10.39.248.123   35.184.17.129   80:30468/TCP   5m

$ curl 35.184.17.129
<html><head><title>HELLO I LOVE YOU!!!!</title></head><body>HELLO I LOVE
YOU!!!!!</body></html>
```

Pivotal

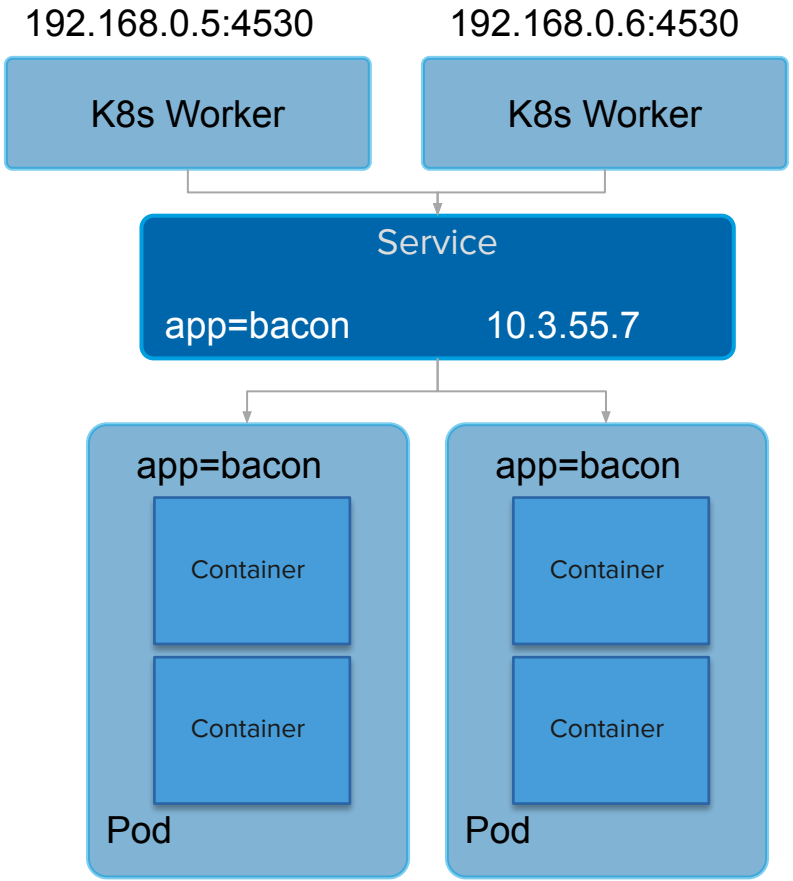



# Service

track Pods based on metadata and provides connectivity and service discovery (DNS, Env variables) for them.

**Type**

ClusterIP (default) exposes service on a **cluster-internal IP**.



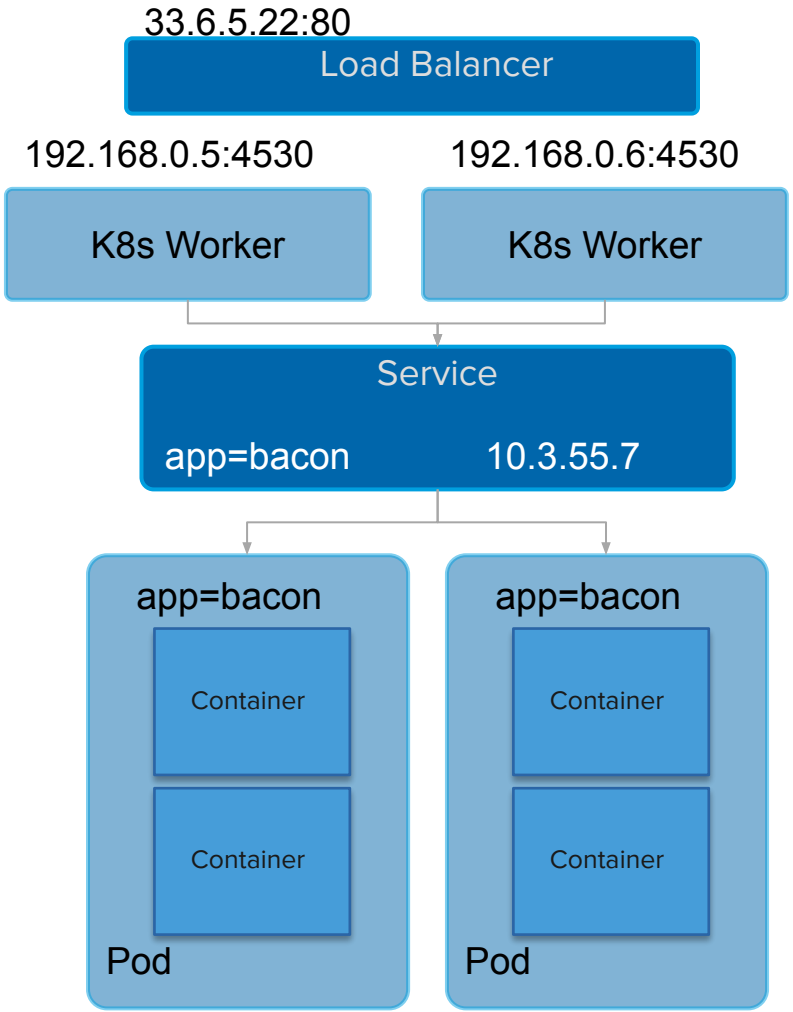

# Service

track Pods based on metadata and provides connectivity and service discovery (DNS, Env variables) for them.

**Type**

NodePort **extends** ClusterIP to expose services on each node's IP via a **static port**.

Pivotal



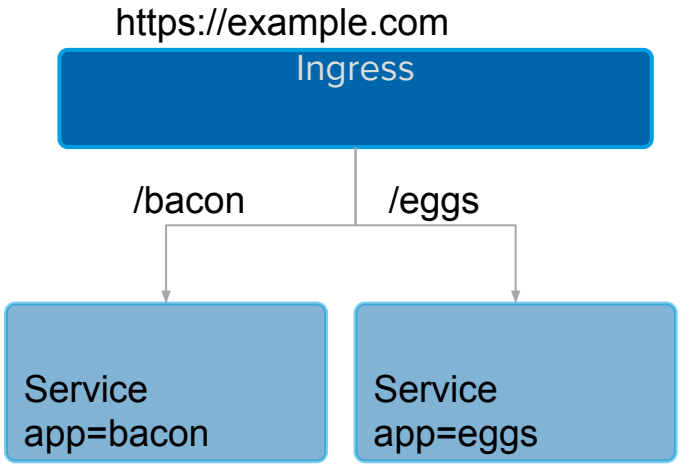

# Service

track Pods based on metadata and provides connectivity and service discovery (DNS, Env variables) for them.

**Type**

LoadBalancer **extends** NodePort to configure a cloud provider's load balancer using the cloud-controller-manager.

Pivotal

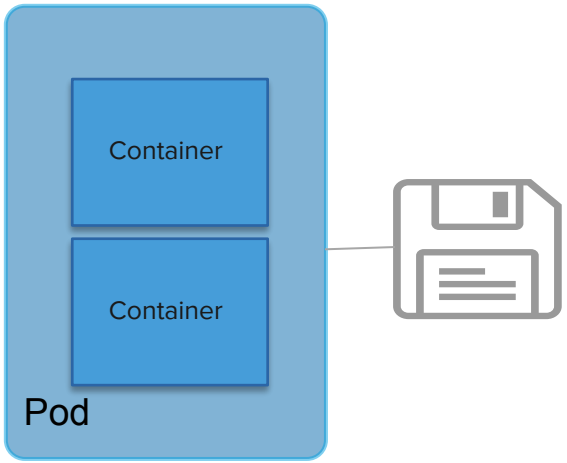



# Ingress

a controller that manages an external entity to provide load balancing, SSL termination and name-based virtual hosting to services based on a set of rules.


# Volume

# Volume




Is [effectively] a **Directory**, possibly with data in it, available to **all containers** in a Pod.

Usually **Shares lifecycle** of a Pod (Created when Pod is created, destroyed when Pod is destroyed).

Persistent Volumes outlive Pods.

Can be mounted from local disk, or from a network storage device such as a EBS volume, iscsi, NFS, etc.

# Config Map / Secret

```
$ kubectl create configmap hello \
      --from-literal=’message=Hello S1T’
```

**kubectl create configmap hello --from-file=index.html**

- creates a *configmap* called *“hello”* containing the contents *index.html*

**$ kubectl get configmap hello -o yaml**

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: hello
data:
  index.html: "<html>\n<head>\n\t<title>Hello to my
friends</title>\n</head>\n<body>\n\tHello
    to my friends\n</body>\n</html>\n\n"
```

**kubectl create secret generic hello --from-file=index.html**

- creates a *secret* called *“hello”* containing a *base64* hash of contents *index.html*

**$ kubectl get secret hello -o yaml**

```
apiVersion: v1
kind: Secret
metadata:
  name: hello
data:
  index.html:
PGh0bWw+CjxoZWFkPgoJPHRpdGxlPkhlbGxvIHRvIG15IGZyaWVuZHM8L3RpdGxlPgo8L2hlYWQ+Cjxib2R5
PgoJSGVsbG8gdG8gbXkgZnJpZW5kcwo8L2JvZHk+CjwvaHRtbD4KCg==
```

Pivotal

## ConfigMaps/Secrets (user-data)

Provides **key-value pairs** to be injected into a pod much like user-data is injected into a Virtual Machine in the cloud.

Allows you to do **last minute configuration** of applications running on Kubernetes such as setting a database host, or a admin password.

ConfigMaps store values as **strings**, Secrets store them as **byte arrays** (serialized as base64 encoded strings).

Secrets are [currently] **not encrypted** by default. This is likely to **change**.

Can be injected as files in a Volume, or as Environment Variables.

# Spring
# Cloud
# Kubernetes

Pivotal

- Discovery
- Ribbon Discovery
- Dynamic Config
- Profile Autoconfiguration
- Istio Awareness
- Pod Health Indicators

# Service Discovery

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-kubernetes</artifactId>
</dependency>
---
@SpringBootApplication
@EnableDiscoveryClient
public class Application {
  public static void main(String[] args) {
    SpringApplication.run(Application.class, args);
  }
}
---
@Autowired
private DiscoveryClient discoveryClient;
```

Pivotal

# spring-cloud-kubernetes-ribbon

```
<dependency>
     <groupId>org.springframework.cloud</groupId>
     <artifactId>spring-cloud-starter-gateway</artifactId>
 </dependency>
 <dependency>
     <groupId>org.springframework.cloud</groupId>
     <artifactId>spring-cloud-starter-kubernetes-ribbon</artifactId>
 </dependency>
```

Pivotal

# spring-cloud-kubernetes-ribbon

```
@SpringBootApplication
@EnableDiscoveryClient
@EnableWebFlux
public class EdgeServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(EdgeServiceApplication.class, args);
    }

    @Bean
    @LoadBalanced
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
}
```

Pivotal

# Native Service Discovery

# spring-cloud-kubernetes-config

```
<dependencies>
     <!-- needed for spring kubernetes config reloads -->
     <dependency>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-actuator</artifactId>
     </dependency>
     <!-- spring cloud kubernetes config server -->
     <dependency>
         <groupId>org.springframework.cloud</groupId>
         <artifactId>spring-cloud-starter-kubernetes-config</artifactId>
     </dependency>
 </dependencies>
```

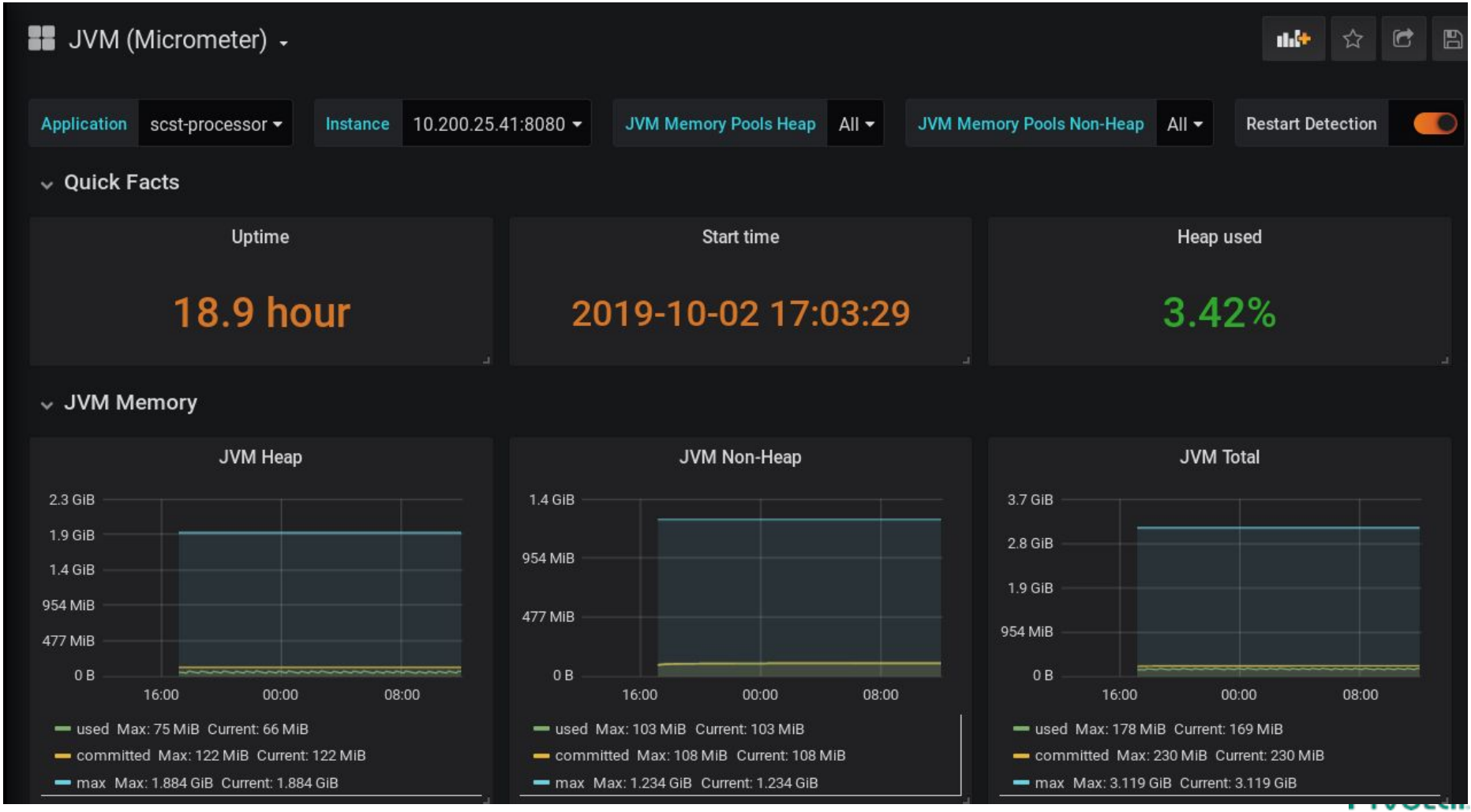
# spring-cloud-kubernetes-config

```
spring:
  application:
    name: cloud-k8s-app
  cloud:
    kubernetes:
      config:
        name: default-name
        namespace: default-namespace
        sources:
         - name: c1
         - namespace: n2
         - namespace: n3
           name: c3
```

Pivotal

# oh and automatic prometheus metrics

```
<dependency>
     <groupId>io.micrometer</groupId>
     <artifactId>micrometer-registry-prometheus</artifactId>
 </dependency>
```


JVM (Micrometer)
Application
scst-processor
Instance
10.200.25.41:8080
JVM Memory Pools Heap
All
JVM Memory Pools Non-Heap
All
Restart Detection
Quick Facts
Uptime
18.9 hour
Start time
2019-10-02 17:03:29
Heap used
3.42%
JVM Memory
JVM Heap
2.3 GiB
1.9 GiB
1.4 GiB
954 MiB
477 MiB
0 B
16:00
00:00
08:00
used Max: 75 MiB Current: 66 MiB
committed Max: 122 MiB Current: 122 MiB
max Max: 1.884 GiB Current: 1.884 GiB
JVM Non-Heap
1.4 GiB
954 MiB
477 MiB
0 B
16:00
00:00
08:00
used Max: 103 MiB Current: 103 MiB
committed Max: 108 MiB Current: 108 MiB
max Max: 1.234 GiB Current: 1.234 GiB
JVM Total
3.7 GiB
2.8 GiB
1.9 GiB
954 MiB
0 B
16:00
00:00
08:00
used Max: 178 MiB Current: 169 MiB
committed Max: 230 MiB Current: 230 MiB
max Max: 3.119 GiB Current: 3.119 GiB

Pivotal

# Transforming How The World Builds Software

@pczarkowski