Kubernetes: What's it do?

Presenter

Eric Paris
Red Hat

Agenda

- What is Kubernetes
- Kubernetes Primitives
- Configuration and Cluster Setup Pointers
- Demo – using kube

What is Kubernetes?

- A declarative language for launching containers.

What is Kubernetes?

- A highly collaborative open source project originally conceived by Google
  - Google has 10+ years experience w/ containerized apps
  - Red Hat has been a member since day 0.
  - Red Hat is the second largest contributing member with many ideas coming from geard
- Sometimes called:
  - kube
  - k8s (that's 'k' + 8 letters + 's')
- Start, stop, update, and manage a cluster of machines running containers in a consistent and maintainable way.

What is Kubernetes?

- Particularly suited for horizontally scaleable, stateless, or 'microservices' application architectures.
  - Does not mean others will not work or are ignored
- Additional functionality to make containers easier to use in a cluster (reachability and discovery).
- Kubernetes does NOT and will not expose all of the 'features' of the docker command line.

Kubernetes Primitives and Key Words

- Master
- Minion/Node
- Pod
- Replication Controller
- Service
- Label
- Namespace

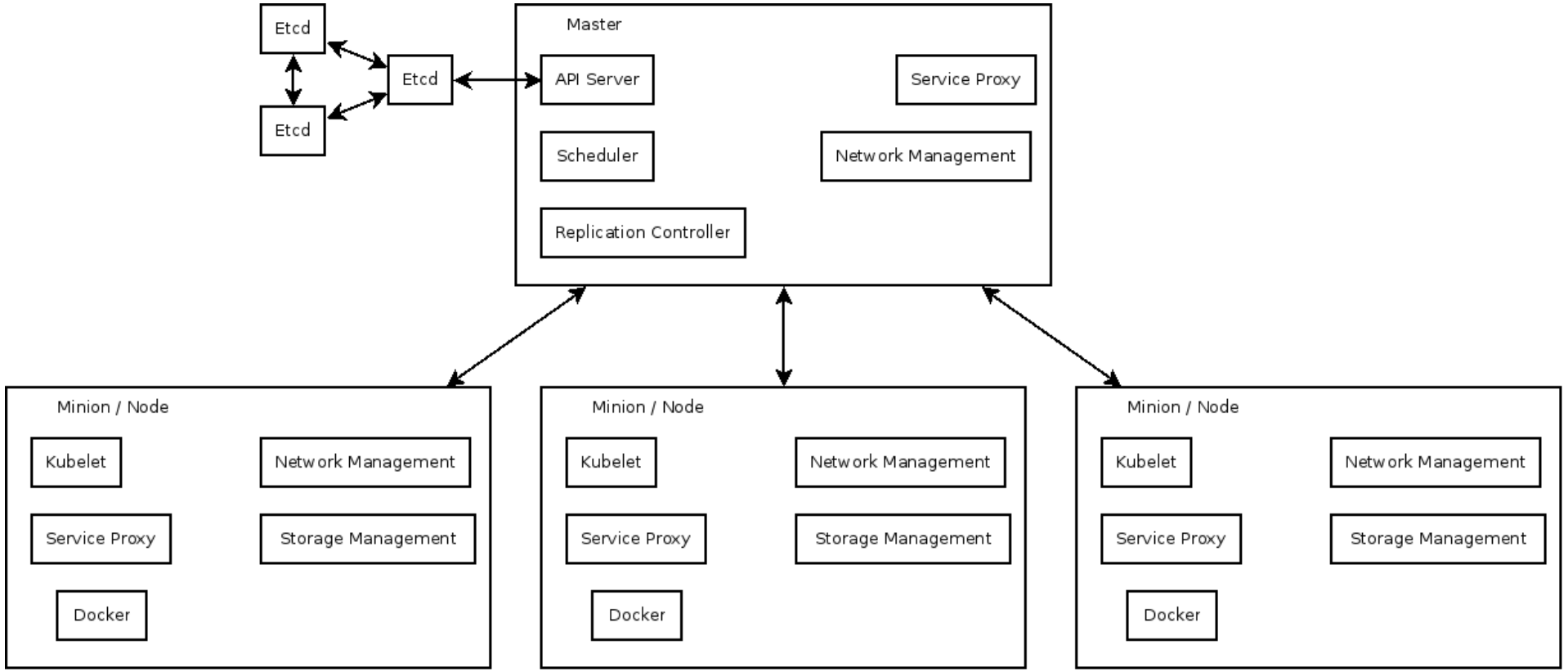
Master

- Typically consists of:
  - kube-apiserver
  - kube-scheduler
  - kube-controller-manager
  - etcd
- Might contain:
  - kube-proxy
  - a network management utility

Minion - Node

- Typically consists of:
  - kubelet
  - kube-proxy
  - cAdvisor
- Might contain:
  - a network management utility
- May be referred to by either name.

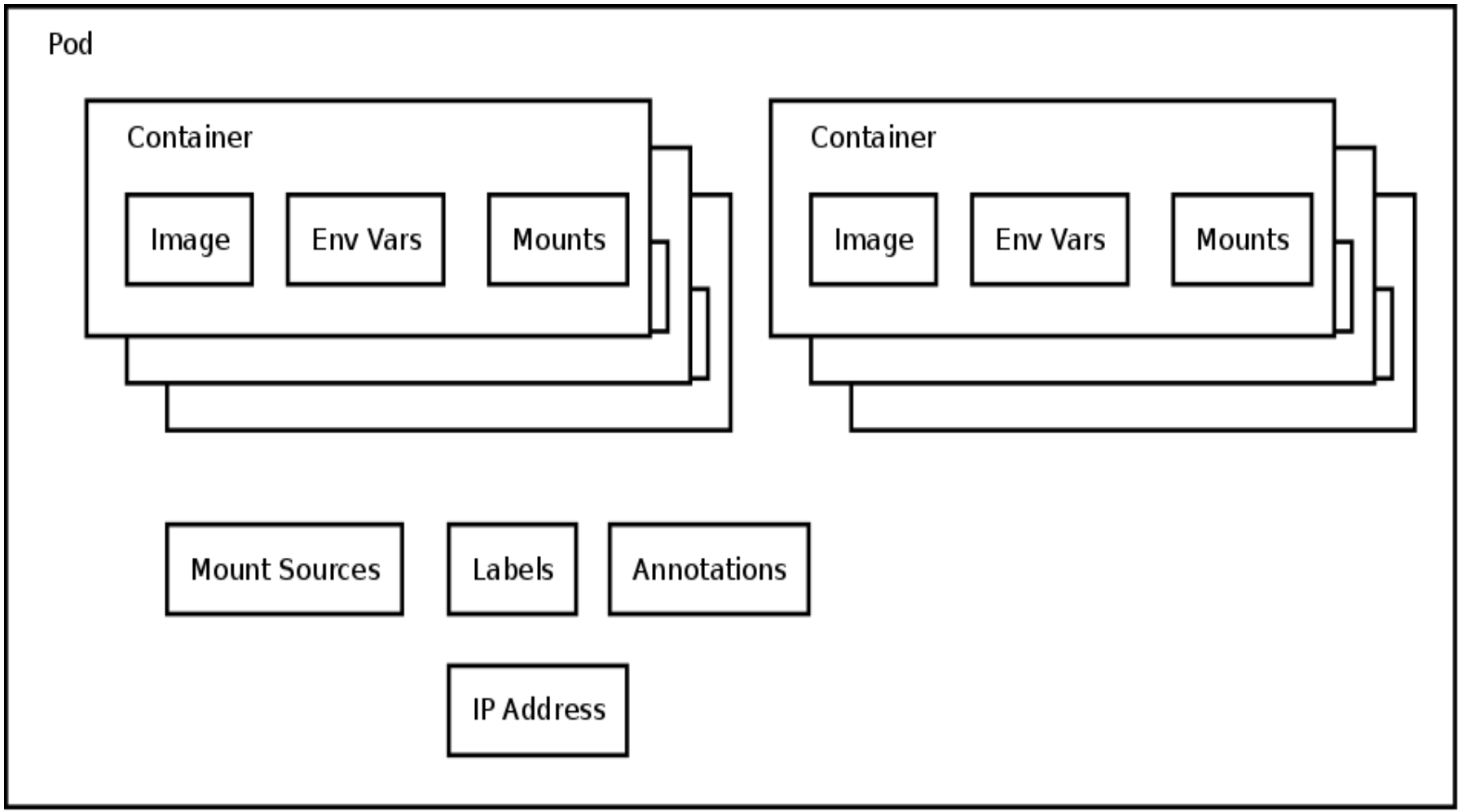
Systems and Binaries



Pod

- Single schedulable unit of work
  - Can not move between machines
  - Can not span machines
- One or more containers
  - Shared network namespace
- Metadata about the container(s)
- Env vars – configuration for the container
- Every pod gets an unique IP
  - Assigned by the container engine, not kube!

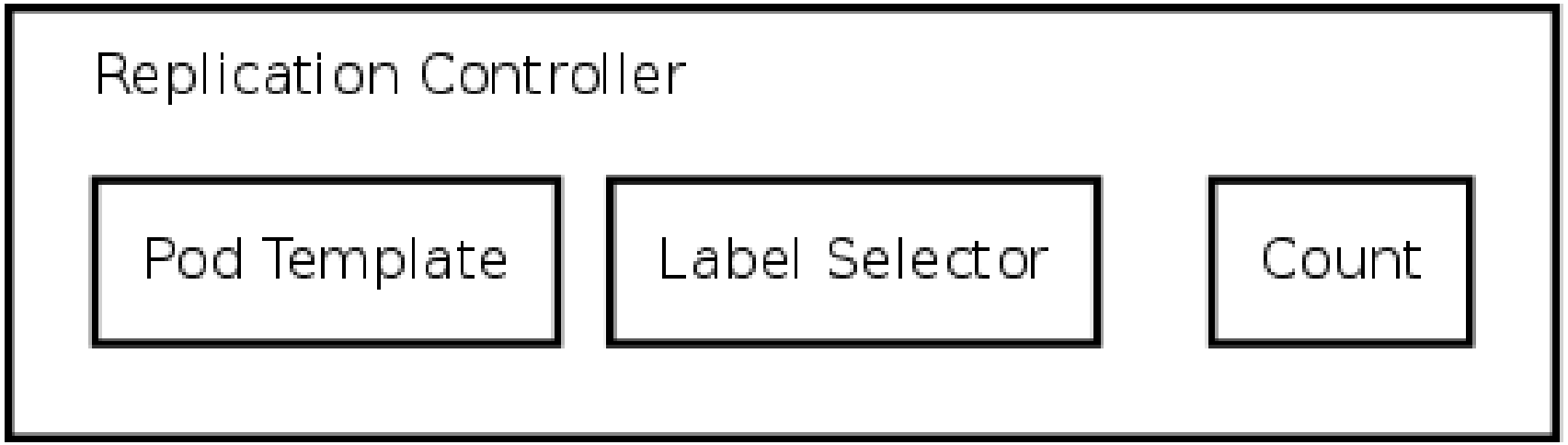
Pod


Pod
Container
Image
Env Vars
Mounts
Container
Image
Env Vars
Mounts
Mount Sources
Labels
Annotations
IP Address

Replication Controller

- Consists of
  - Pod template
  - Count
  - Label Selector
- Kube will try to keep $count copies of pods matching the label selector running
- If too few copies are running the replication controller will start a new pod somewhere in the cluster

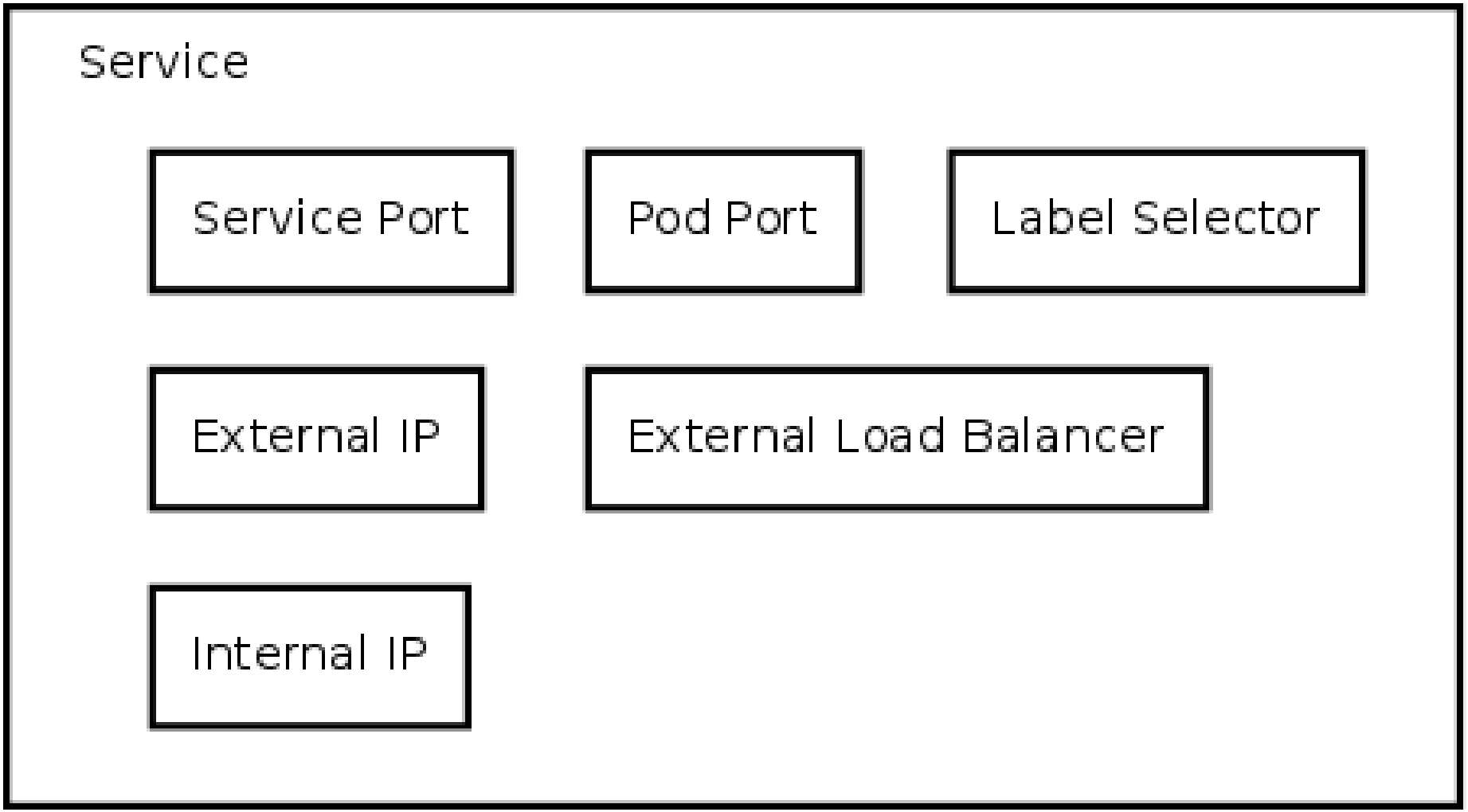
Replication Controller



Services

- How 'stuff' finds pods which could be anywhere
- Define:
  - What port in the container
  - Labels on pods which should respond to this type of request
- Can define:
  - What the 'internal' IP should be
  - What the 'external' IP should be
  - What port the service should listen on

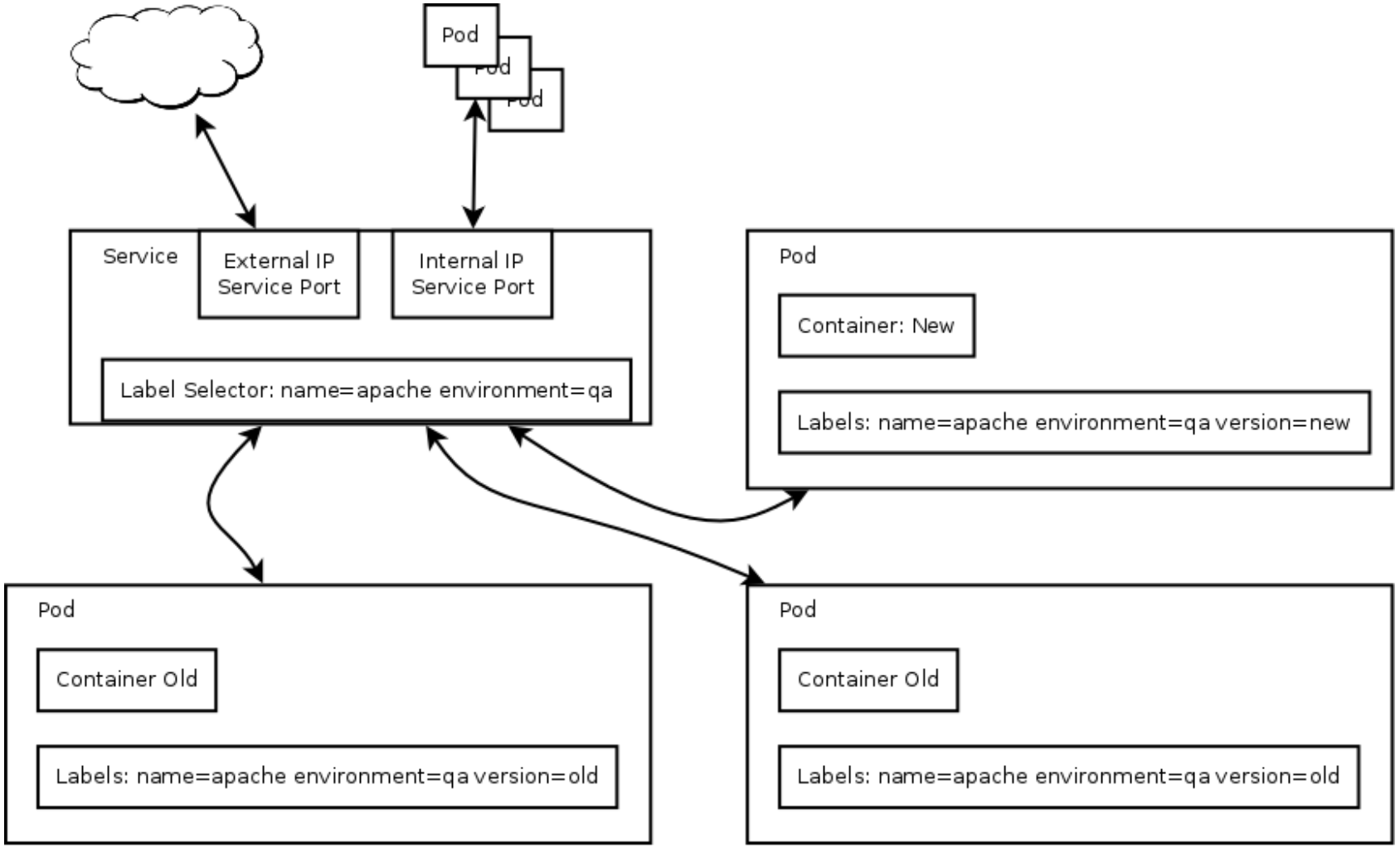
Services


Service
Service Port
Pod Port
Label Selector
External IP
External Load Balancer
Internal IP

Labels

- List of key=value pairs
- Attached to all objects
- Currently used in 2 main places
  - Matching pods to replication controllers
  - Matching pods to services
- Objects can be queried from the API server by label

Services and Labels



Namespace

- Attached to every object
- Pods in ns1 will not get service variable from ns2
- Users with permission to CRUD objects in ns1 may not have permissions to CRUS object in ns2
- The network is not segregated!

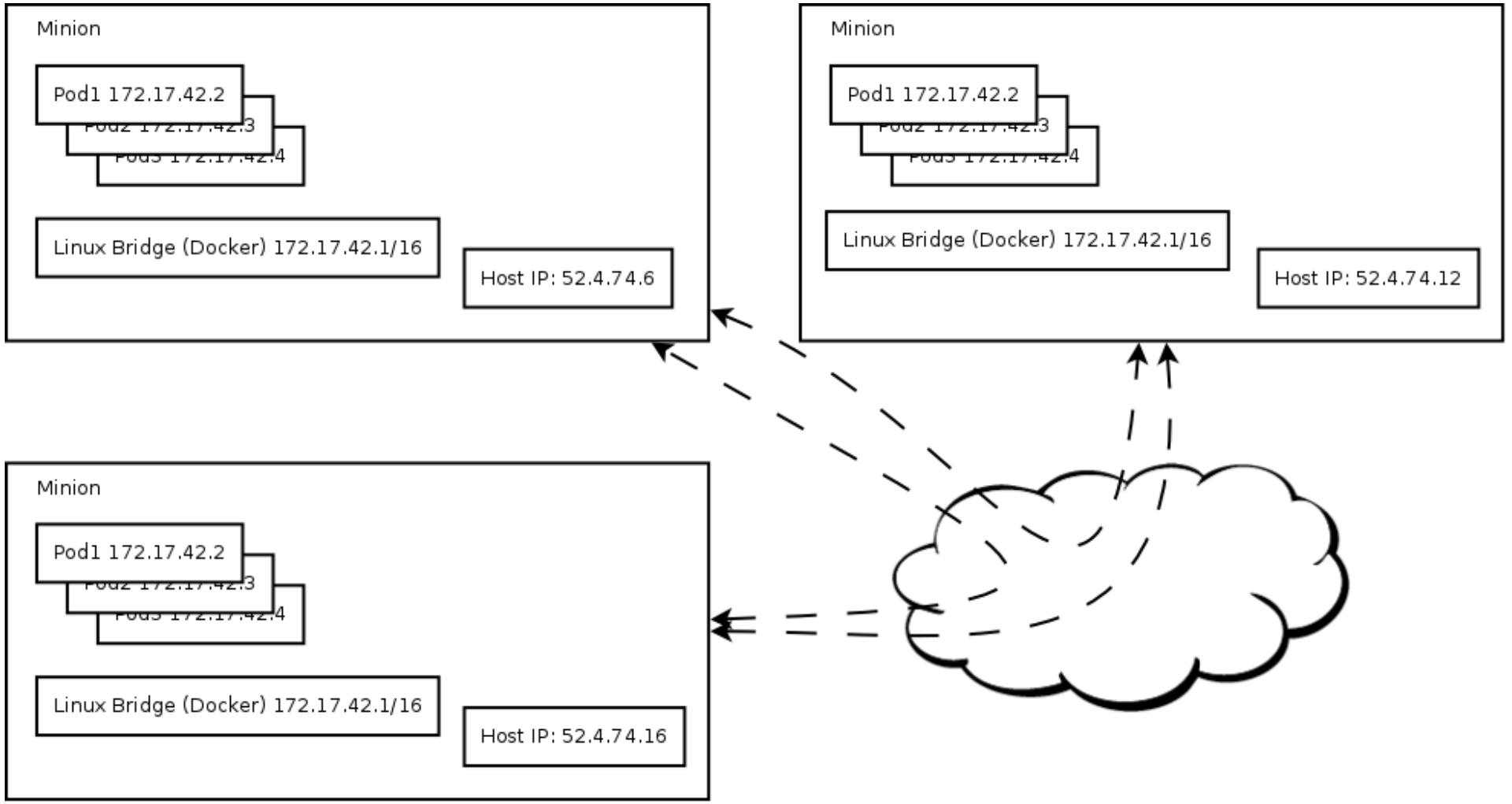
Configuration differences since last talk

- Configuration Changes:
  - Systemd and /etc/kubernetes/ file formatting
  - Kubelet takes: –api_servers=
  - Controller-manager takes –machines=

Networking Setup – In Fedora

- Networking is a docker problem – not kube
  - Kube makes those problems apparent!
  - If any two docker containers on any two hosts can talk over IP, kube will just work.

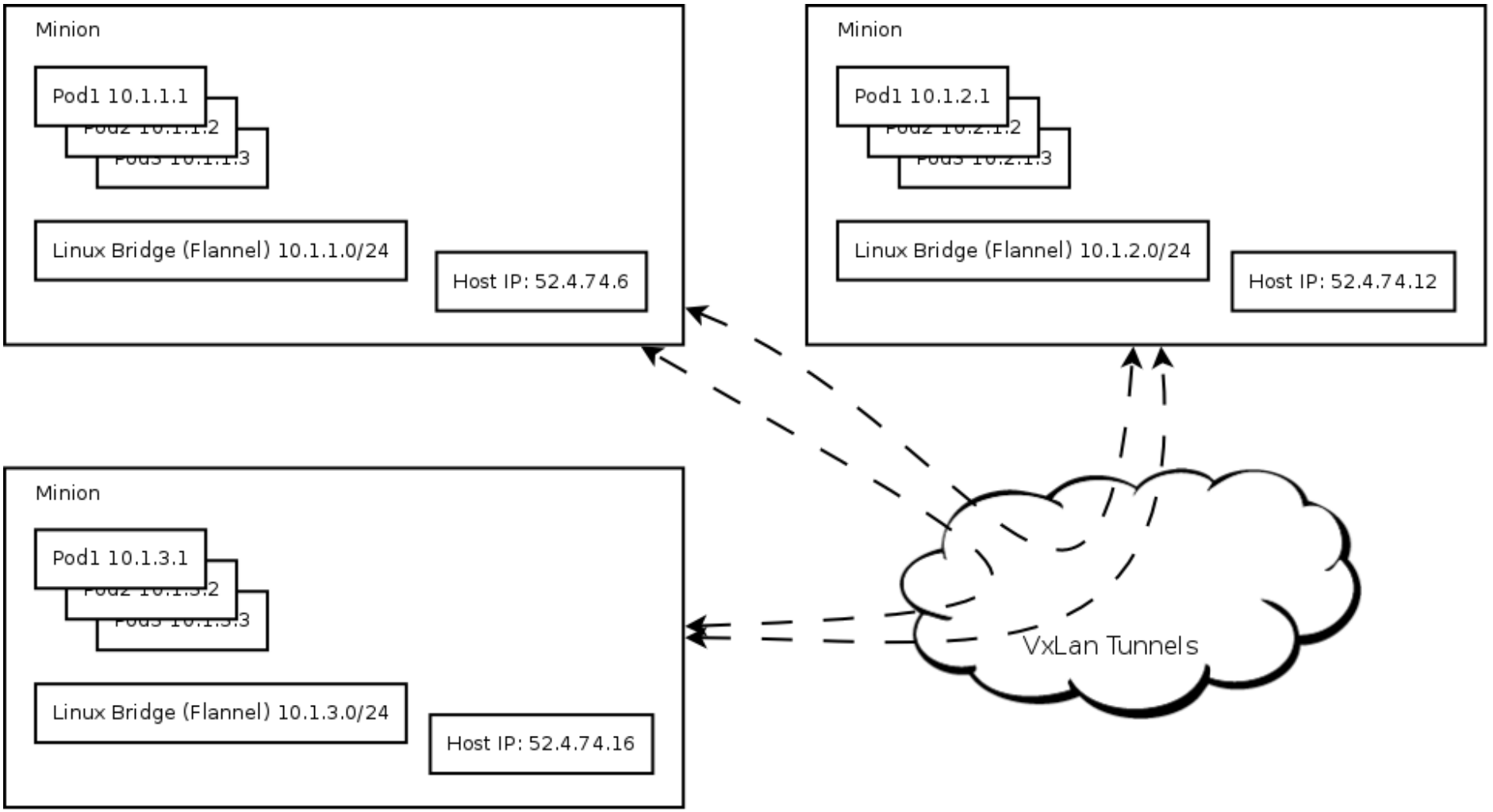
Networking Docker Out Of The Box



Networking Setup – Available in Fedora

- Flannel
  - Super super easy configuration
  - Can create a vxlan overlay network
  - Can configure docker to launch pods in this overlay
  - Pods just work!
- There are many other solutions.
  - This one is easy.

Networking with an overlay network

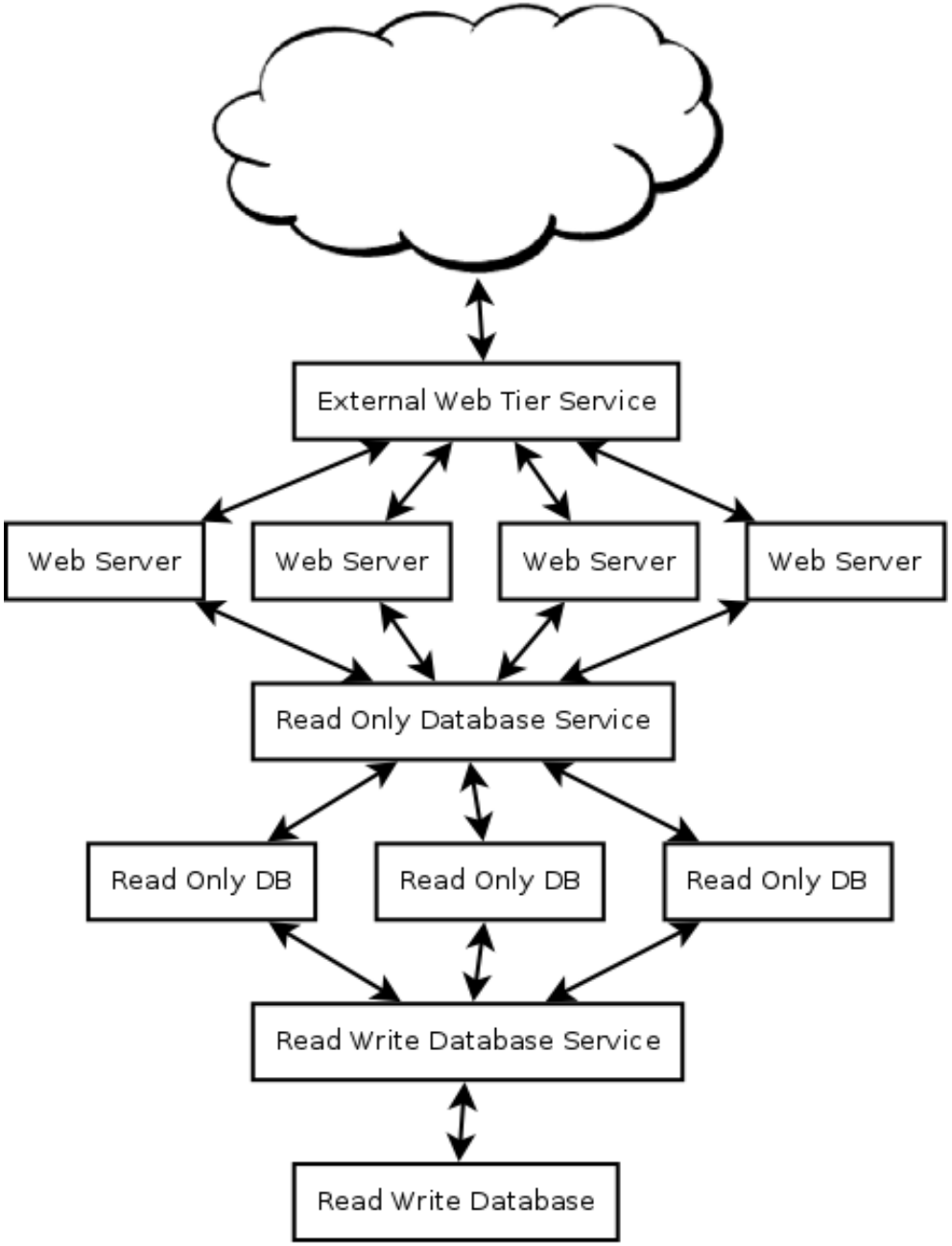


Demo

- Create a multi tier web application
- Show that it works
- Update the web front end with 0 downtime

- How I'm cheating in the demo
  - Cluster is already set up.
  - Containers already created
  - Containers already pulled (docker pull is slow)
  - services and replication controllers already written

Demo


External Web Tier Service
Web Server
Web Server
Web Server
Web Server
Read Only Database Service
Read Only DB
Read Only DB
Read Only DB
Read Write Database Service
Read Write Database