

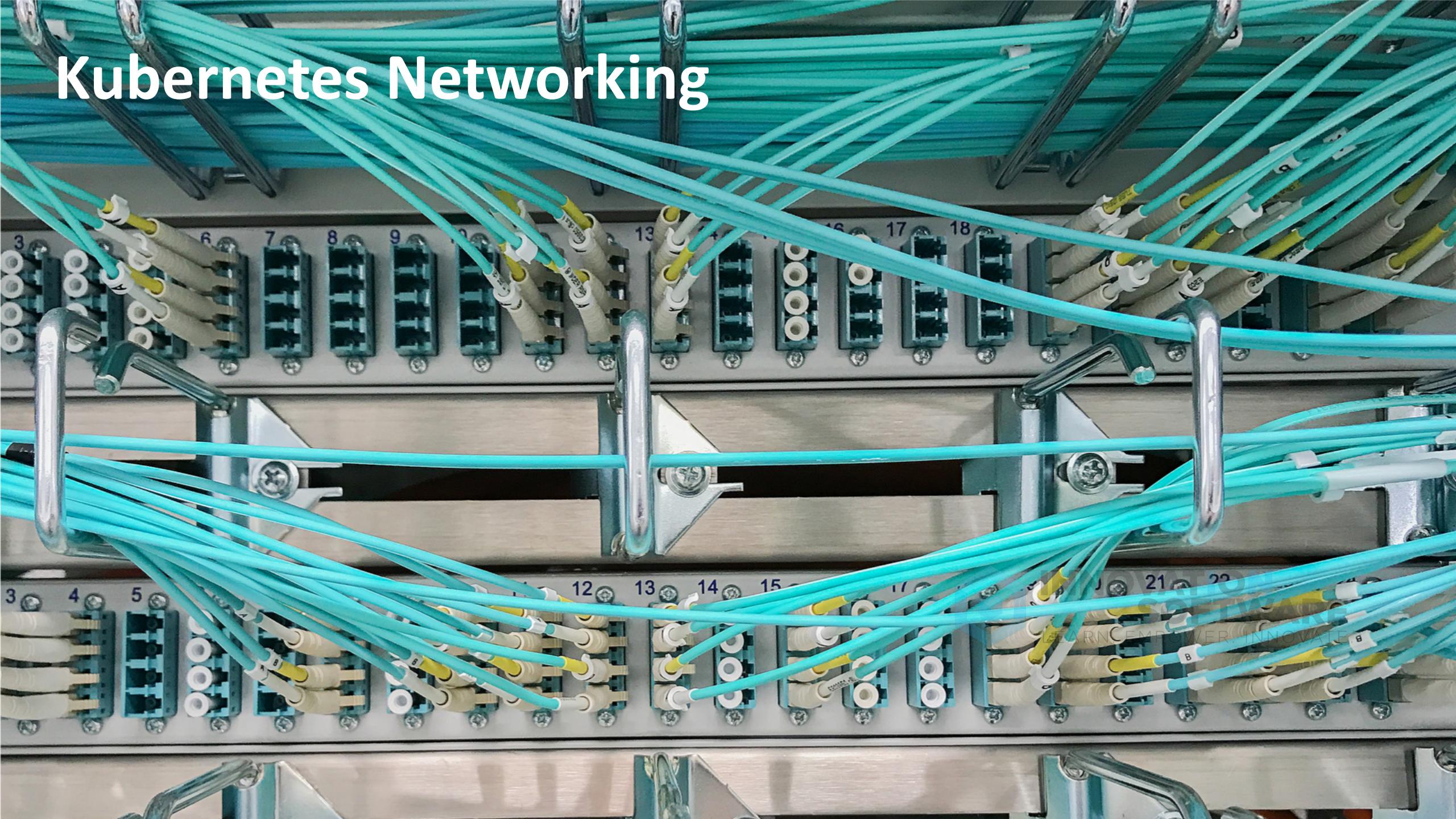


kubernetes



**INNOVATION
SOFTWARE**
LEARN. EMPOWER. INNOVATE.

Kubernetes Networking



Networking Problems and Solutions

1. Highly-coupled container-to-container communications => linux
2. Pod-to-Pod communications => various
3. Pod-to-Service communications => service
4. External-to-Service communications => service or ingress

Kubernetes Networking Model Requirements

- all containers can communicate with all other containers without NAT
- all nodes can communicate with all containers (and vice-versa) without NAT
- the IP that a container sees itself as is the same IP that others see it as

Container to Container Communication

- Containers live inside of pods
- Group of one or more containers that are always co-located and co-scheduled, and run in a shared context
- Containers within a POD share an IP address and port space, and can find each other via localhost (net namespace).
- Communicate with each other using standard inter-process communications (ipc namespace).

Pod to Pod Requirements

- Pods can communicate with all other pods without NAT
- Regardless of which host they land on
- Every pod gets its own IP address
- No need to explicitly create links between pods

Pod Networking Approaches

- **Using an overlay network**

- An overlay network obscures the underlying network architecture from the pod network through traffic encapsulation (for example vxlan).
- Encapsulation reduces performance, though exactly how much depends on your solution.

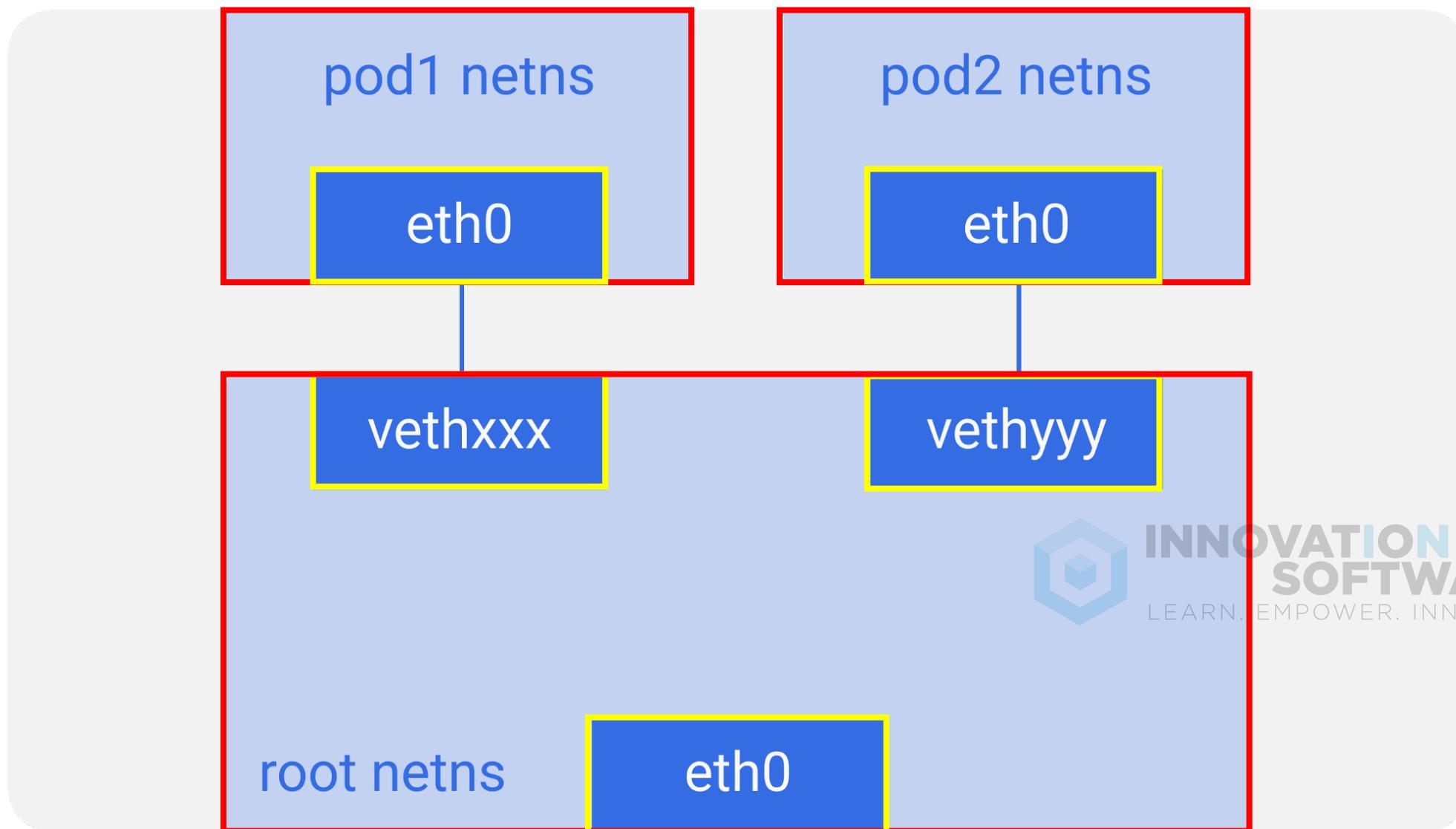
- **Without an overlay network**

- Configure the underlying network fabric (switches, routers, etc.) to be aware of pod IP addresses.
- This does not require the encapsulation provided by an overlay, and so can achieve better performance.

Node Namespace

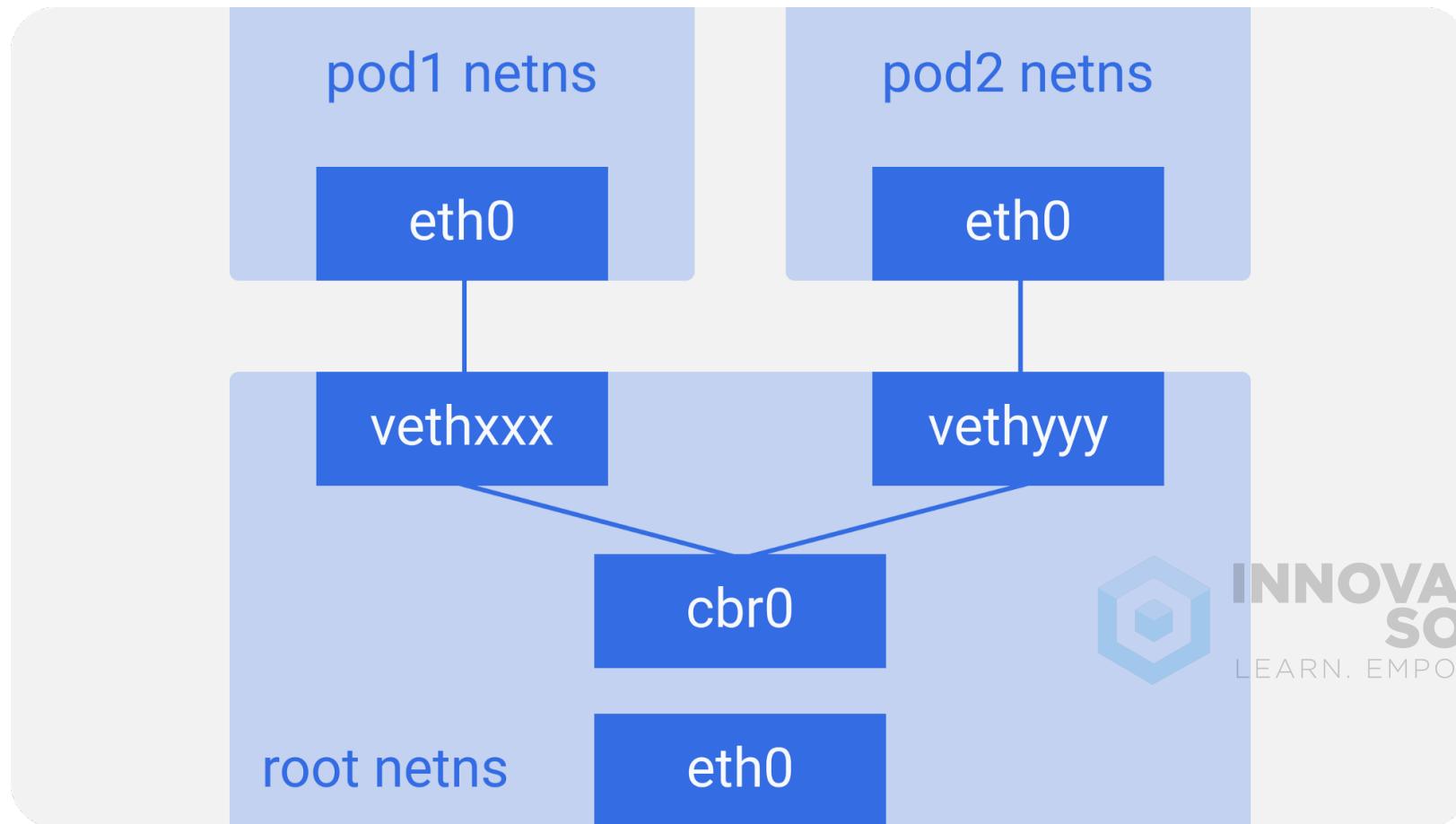


Node Namespaces



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

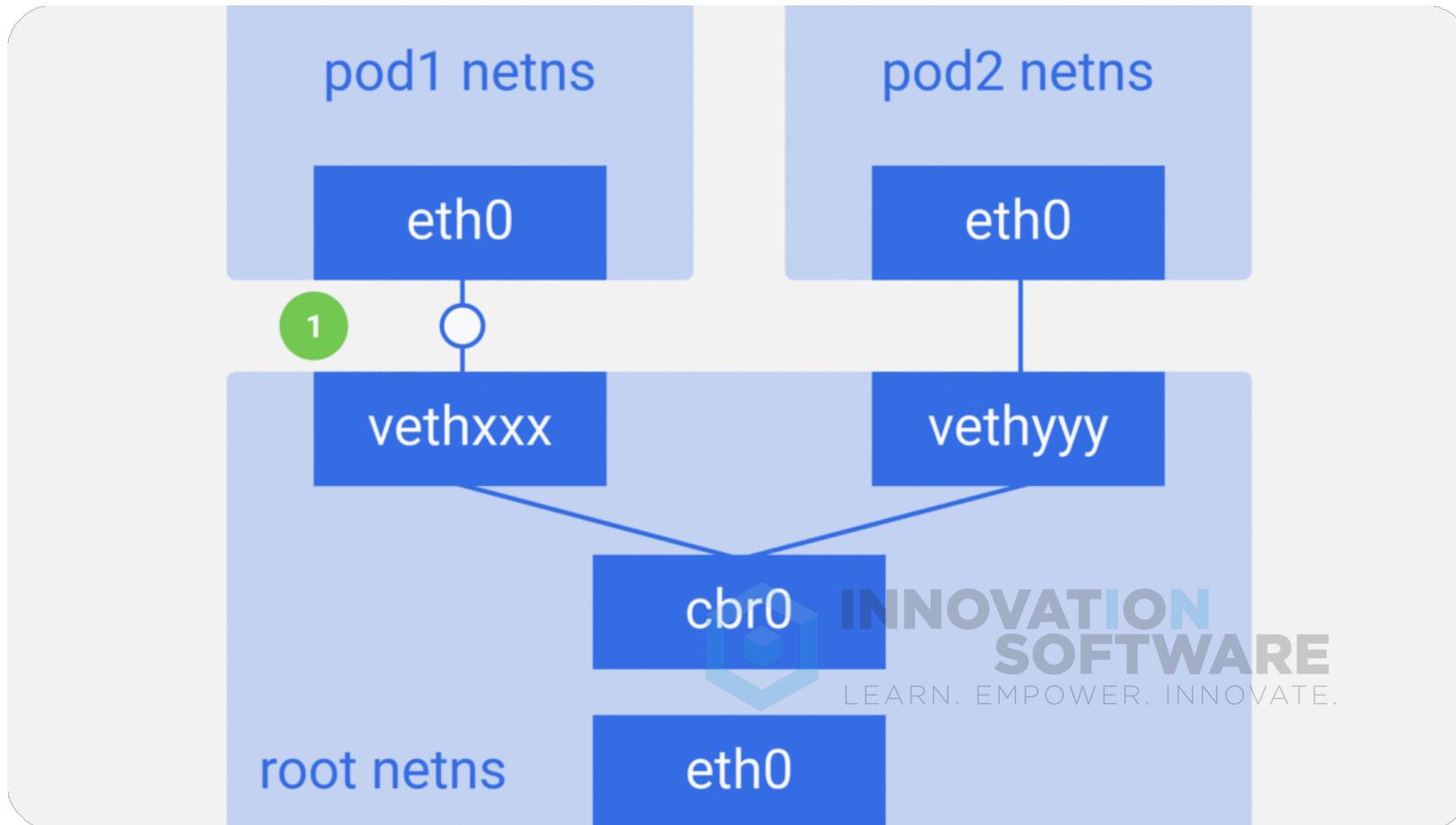
Pod to Host



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

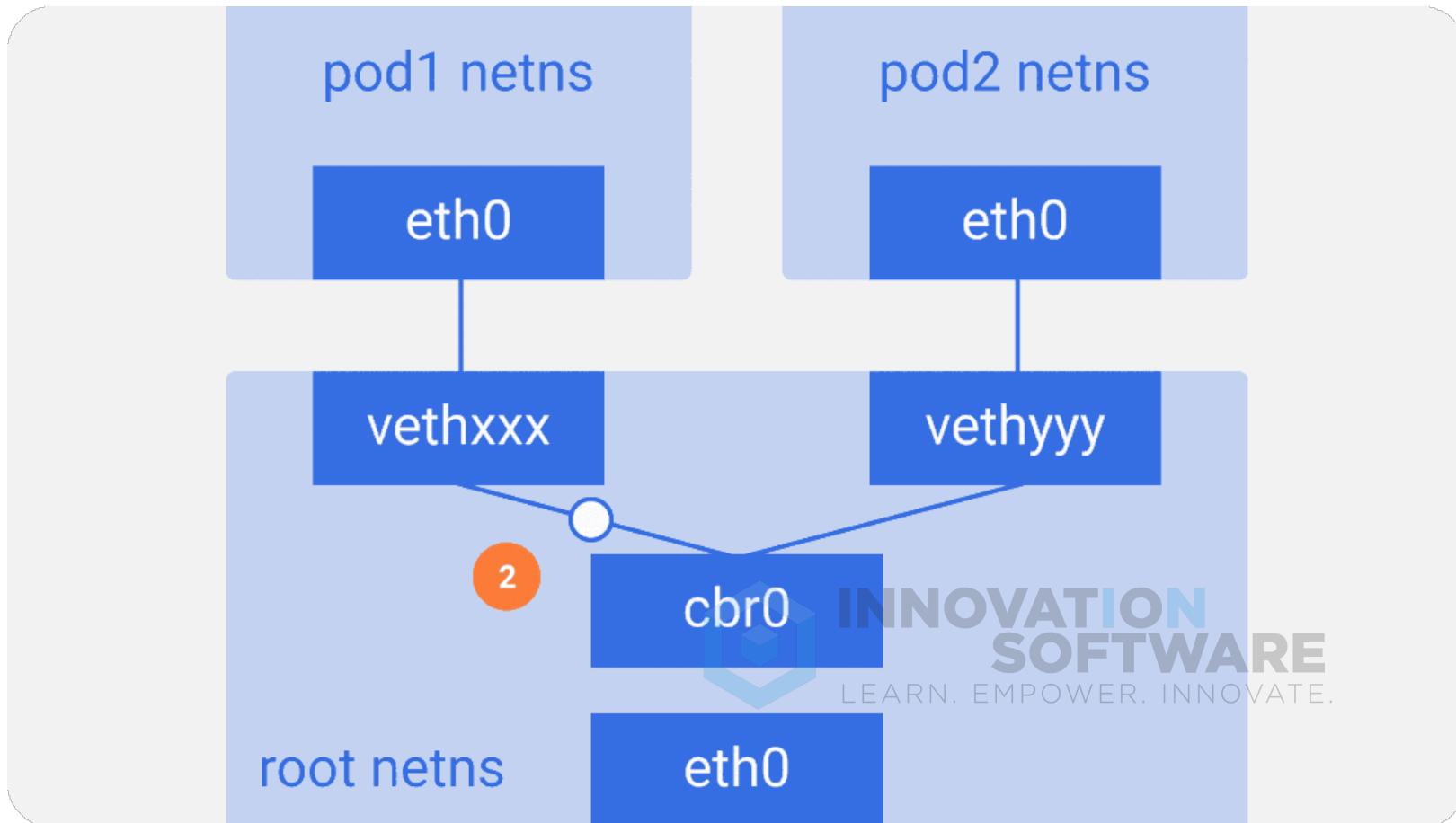
Single Host POD to POD Network

1. Packet leaves pod1's netns at eth0 and enters the root netns at vethxxx.



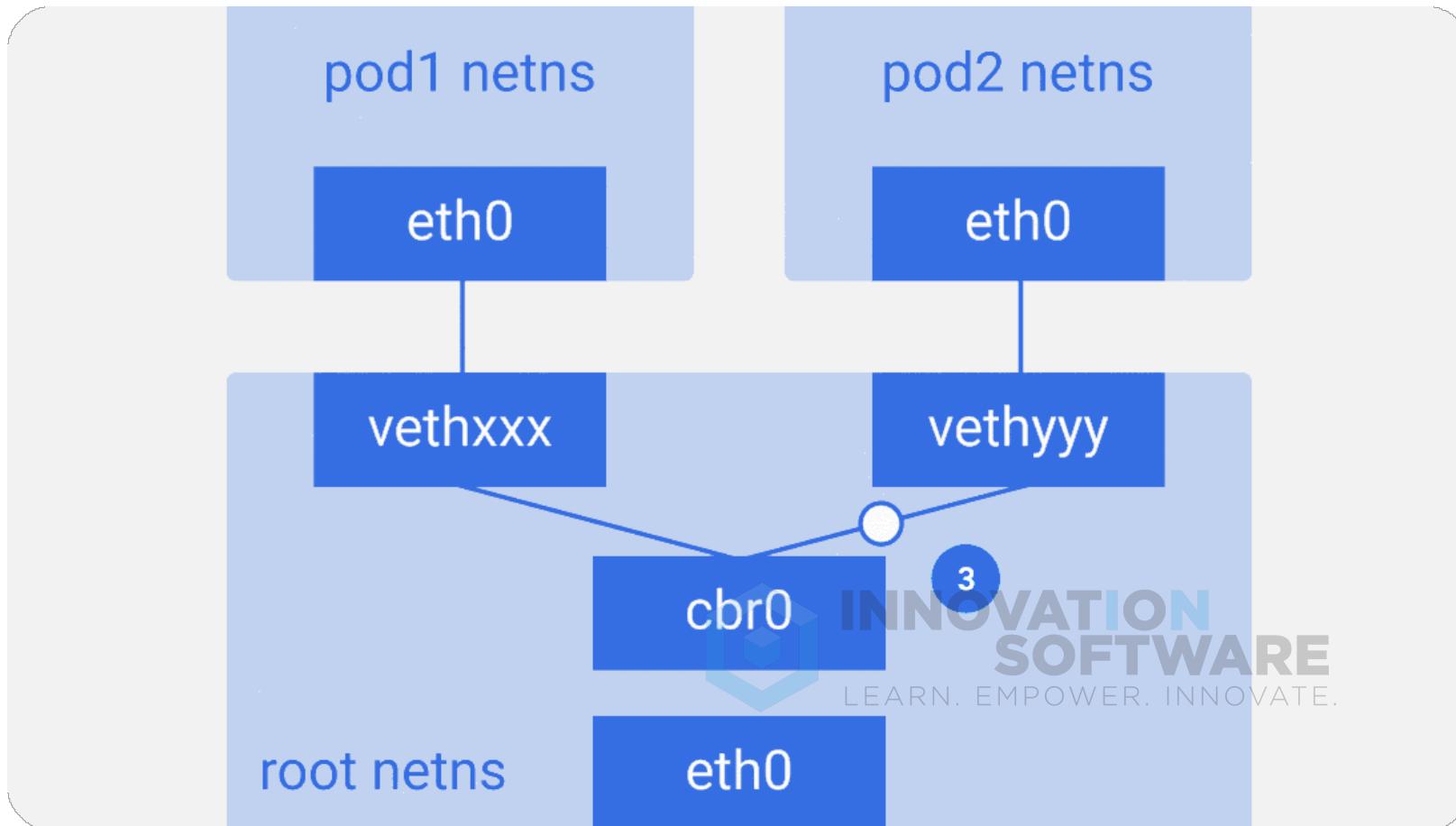
Single Host POD to POD Networking

2. It's passed on to cbr0, which discovers the destination using an ARP request, saying “who has this IP?”



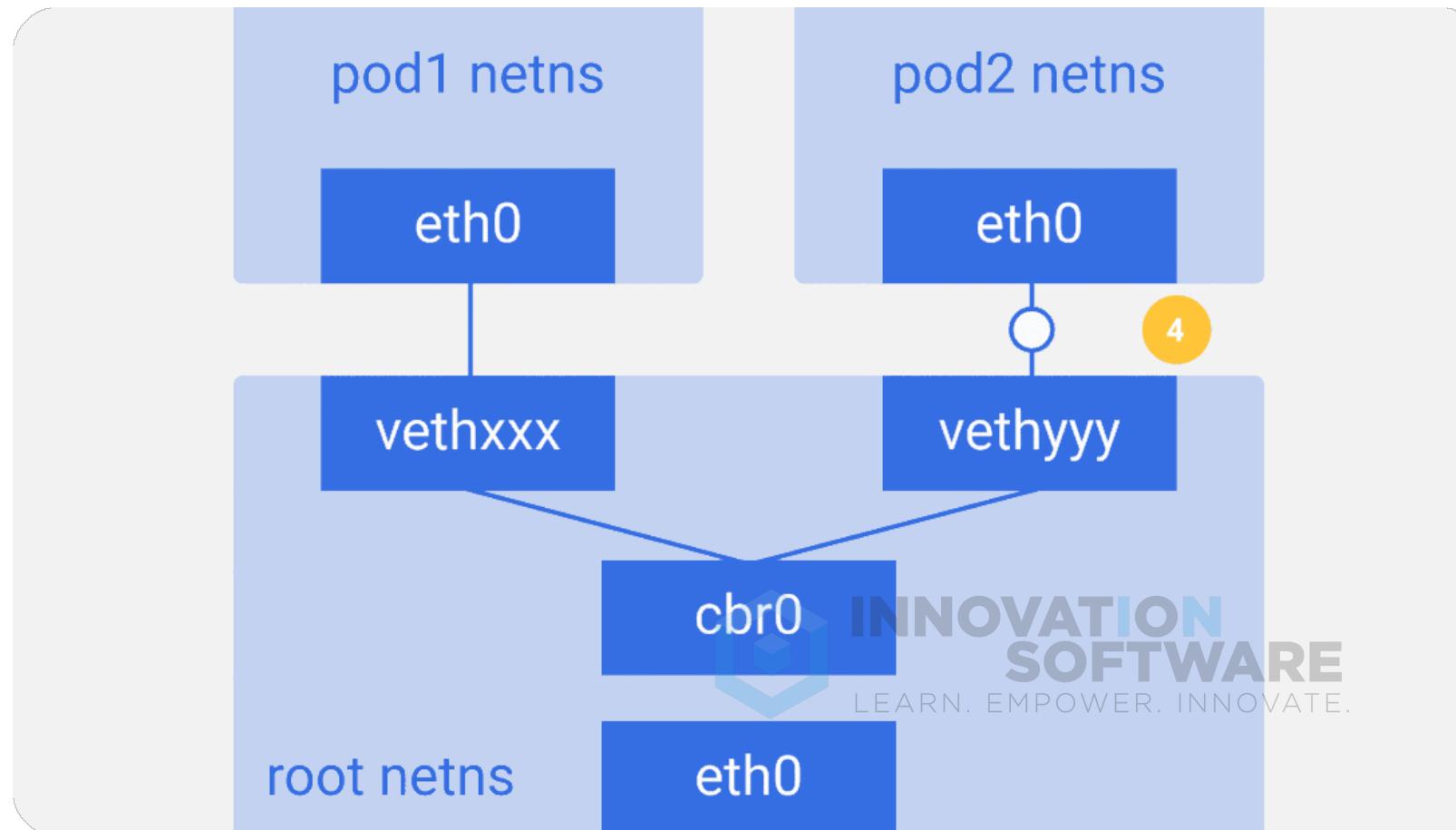
Single Host POD to POD Networking

3. vethyyy says it has that IP, so the bridge knows where to forward the packet.

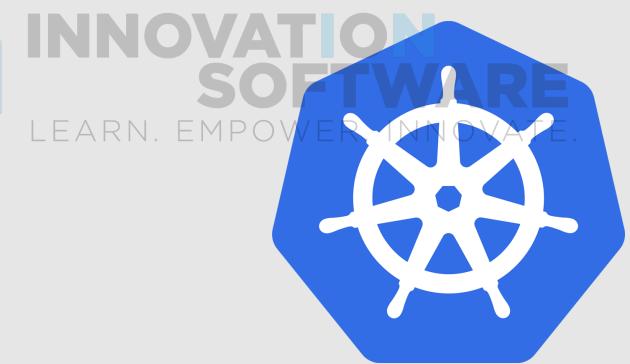


Single Host POD to POD Networking

4. The packet reaches vethyyy, crosses the pipe-pair and reaches pod2's netns.



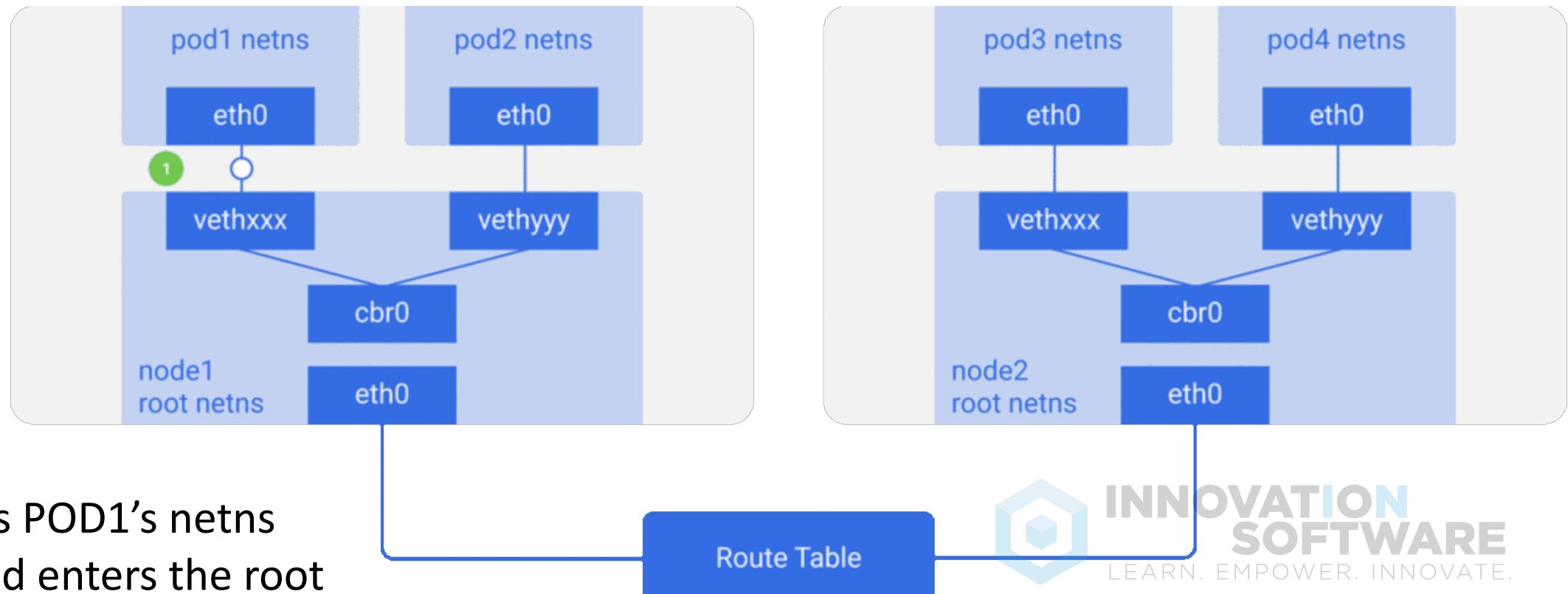
Kubernetes Inter-node Communication



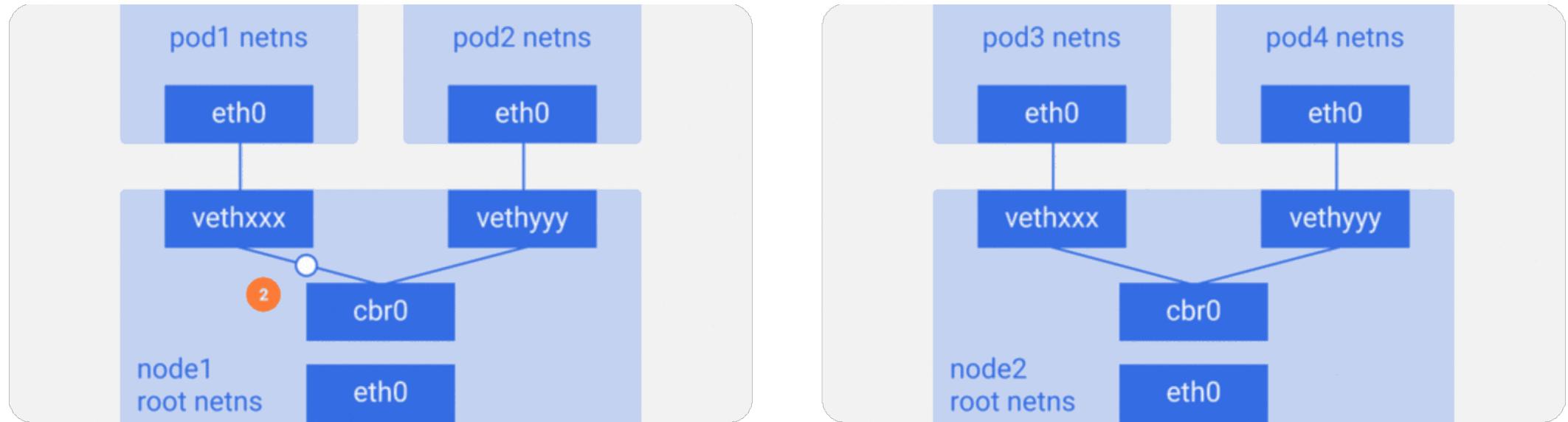
Inter-node Communication

- PODs must be reachable across nodes
- Kubernetes doesn't care how:
 - L2 (ARP across nodes)
 - L3 (IP routing across nodes)
 - Overlay networks

Inter-node Communication



Inter-node Communication



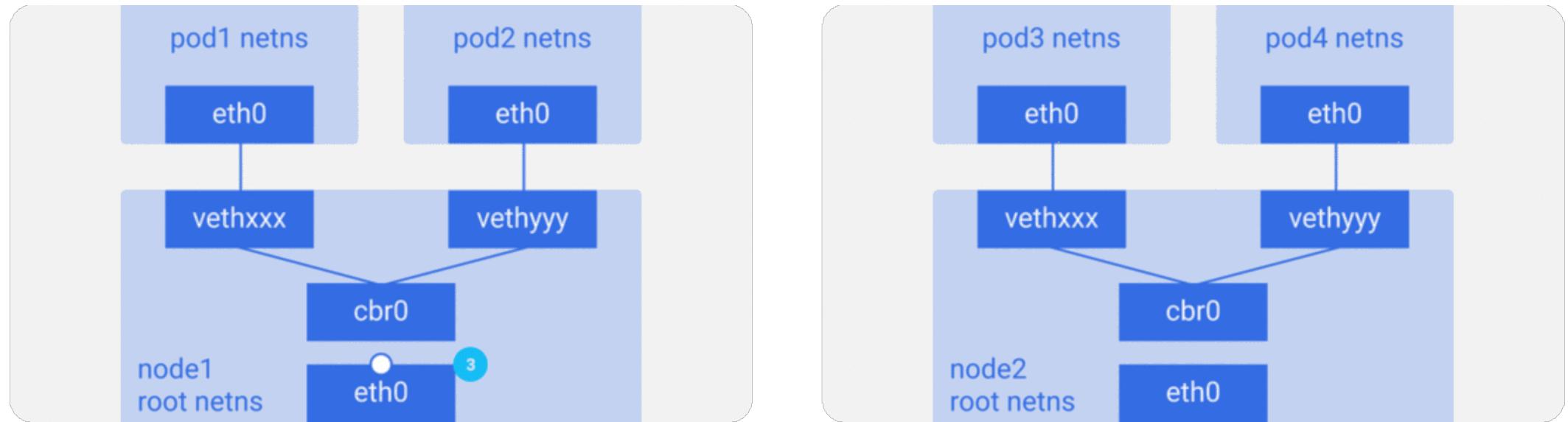
2. It's passed on to cbr0,
which makes the ARP request
to find the destination.

Route Table



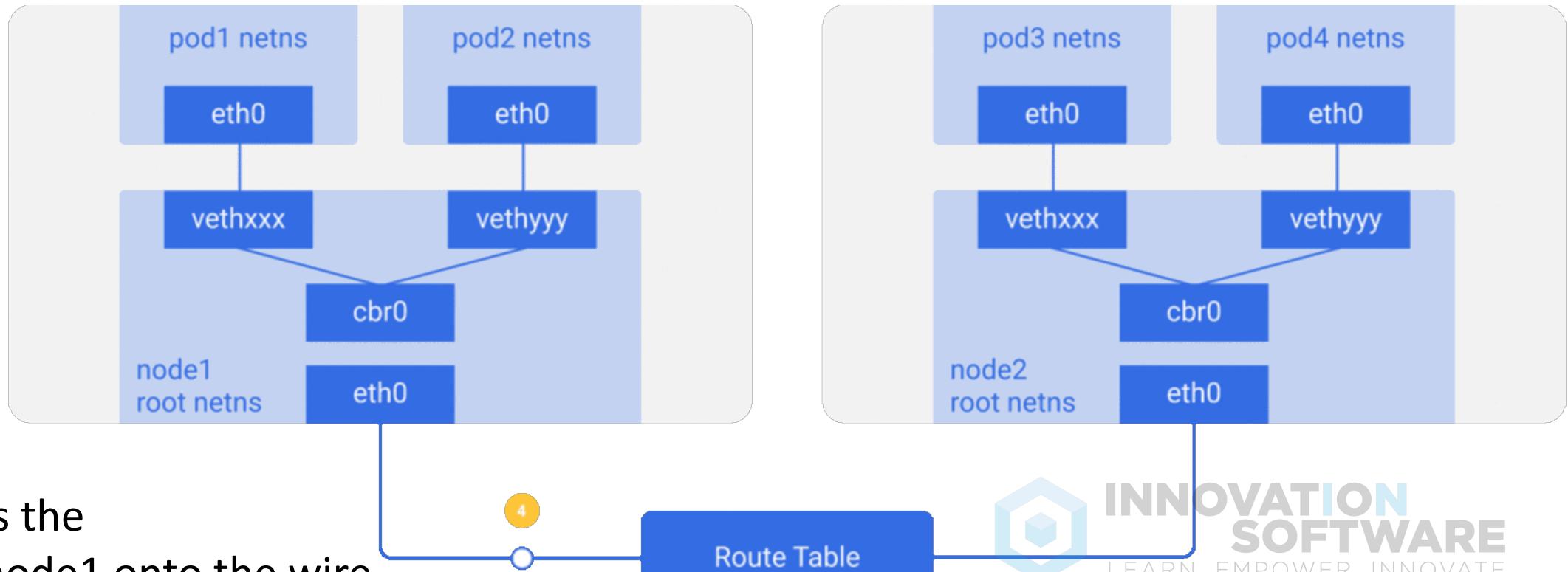
INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Inter-node Communication



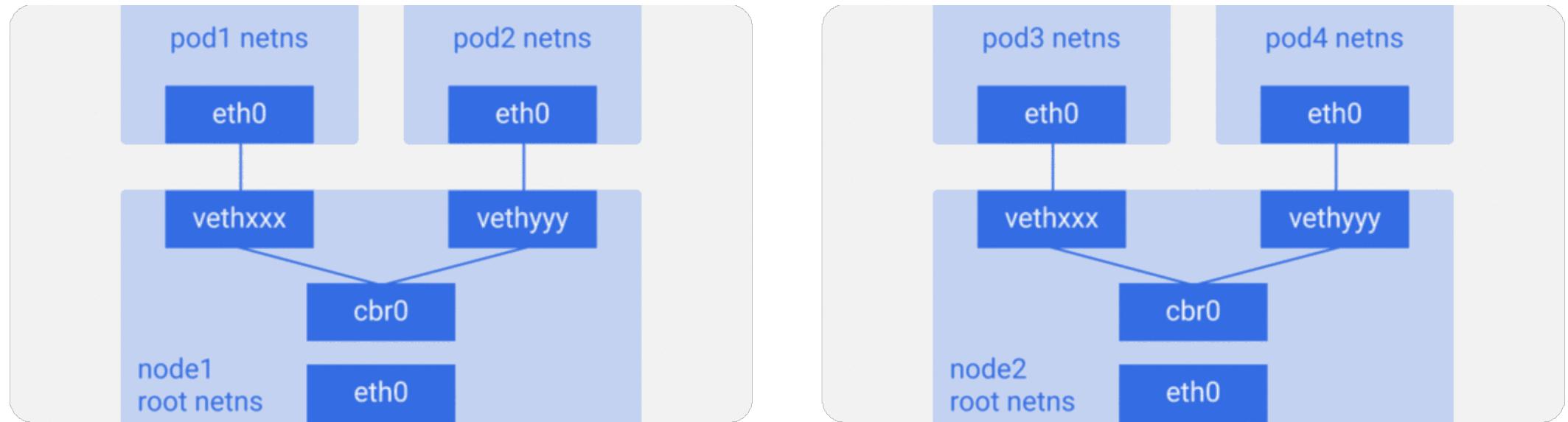
3. It comes out of cbr0 to the main network interface eth0 since nobody on this node has the IP address for POD4.

Inter-node Communication



4. It leaves the machine node1 onto the wire with src=POD1 and dst=POD4.

Inter-node Communication



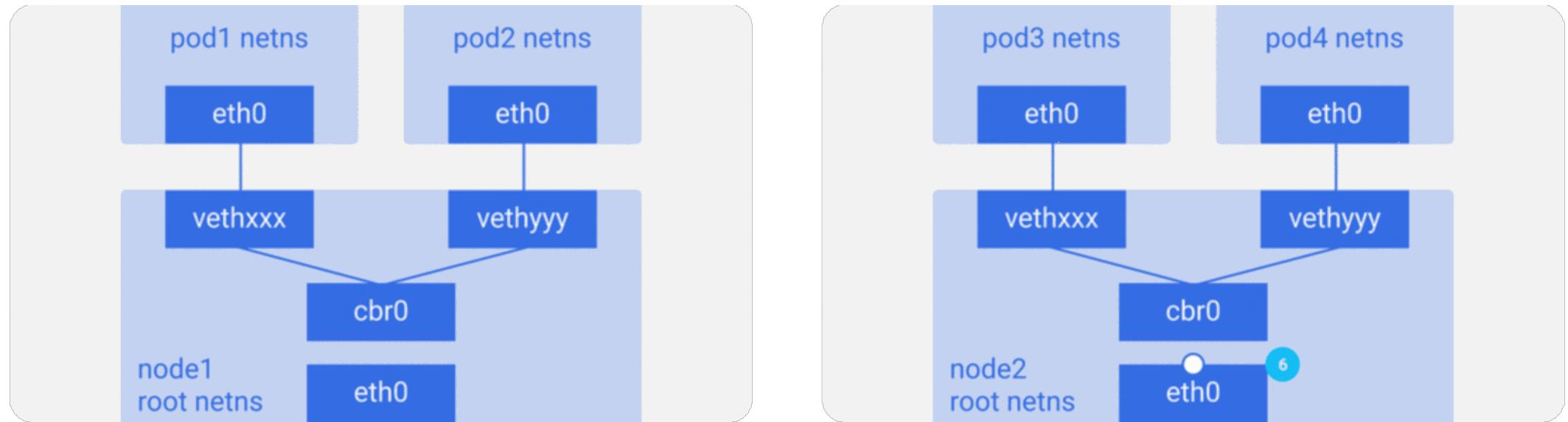
5. The route table has routes setup for each of the node CIDR blocks, and it routes the packet to the node whose CIDR block contains the POD4 IP.

Route Table



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Inter-node Communication



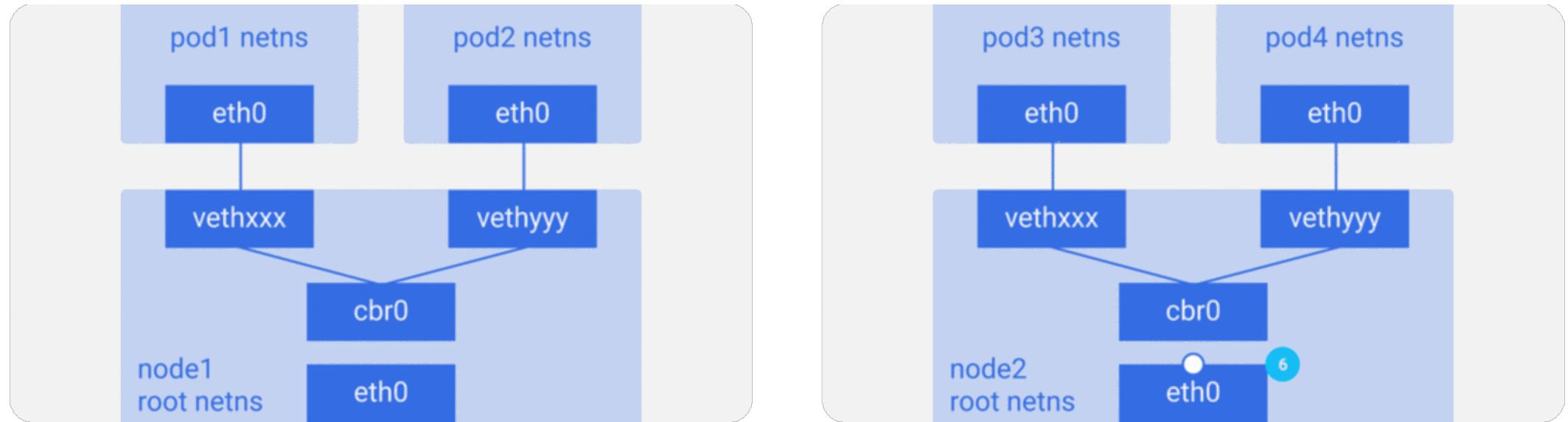
6. Packet arrives at node2 at eth0. POD4 isn't the IP of eth0, the packet is forwarded to cbr0, nodes are configured with IP forwarding enabled.

Route Table



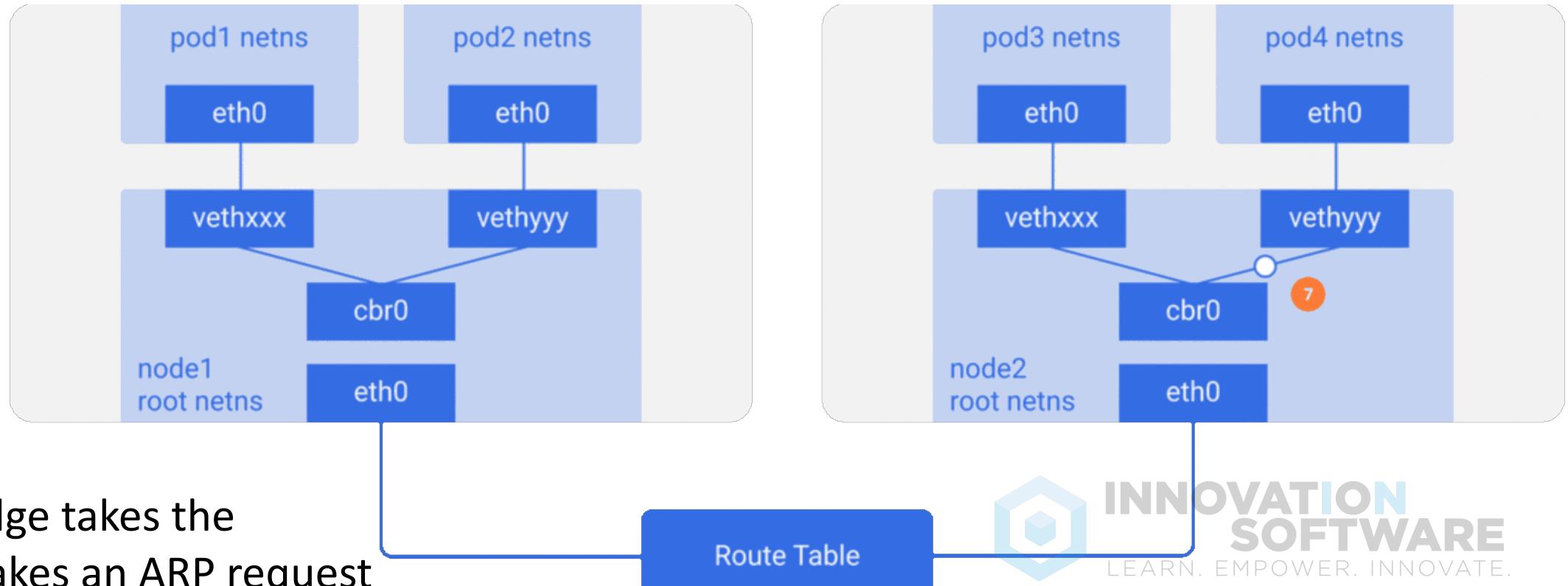
INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Inter-node Communication



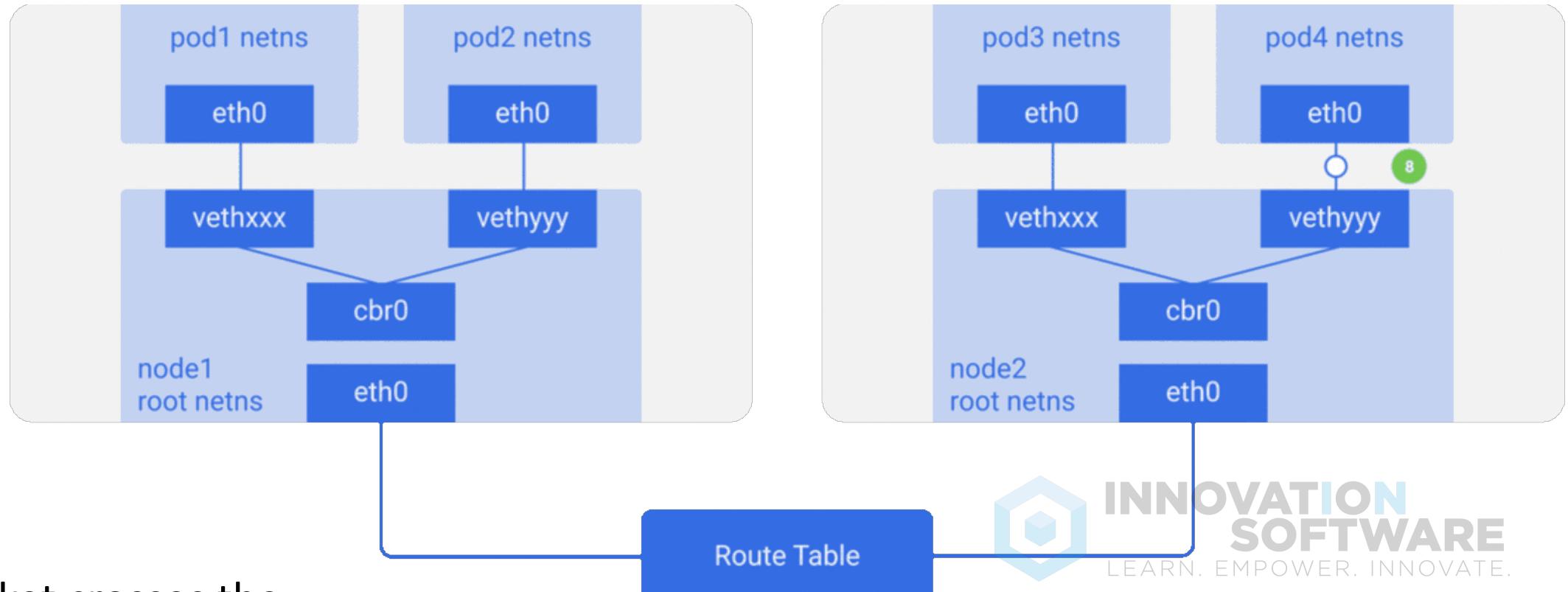
6. The node's routing table is looked up for any routes matching the POD4 IP. It finds `cbr0` as the destination for this node's CIDR block.

Inter-node Communication



**INNOVATION
SOFTWARE**
LEARN. EMPOWER. INNOVATE.

Inter-node Communication



8. The packet crosses the pipe-pair and reaches POD4

Kubernetes Overlay Networking



INNOVATION
SOFTWARE

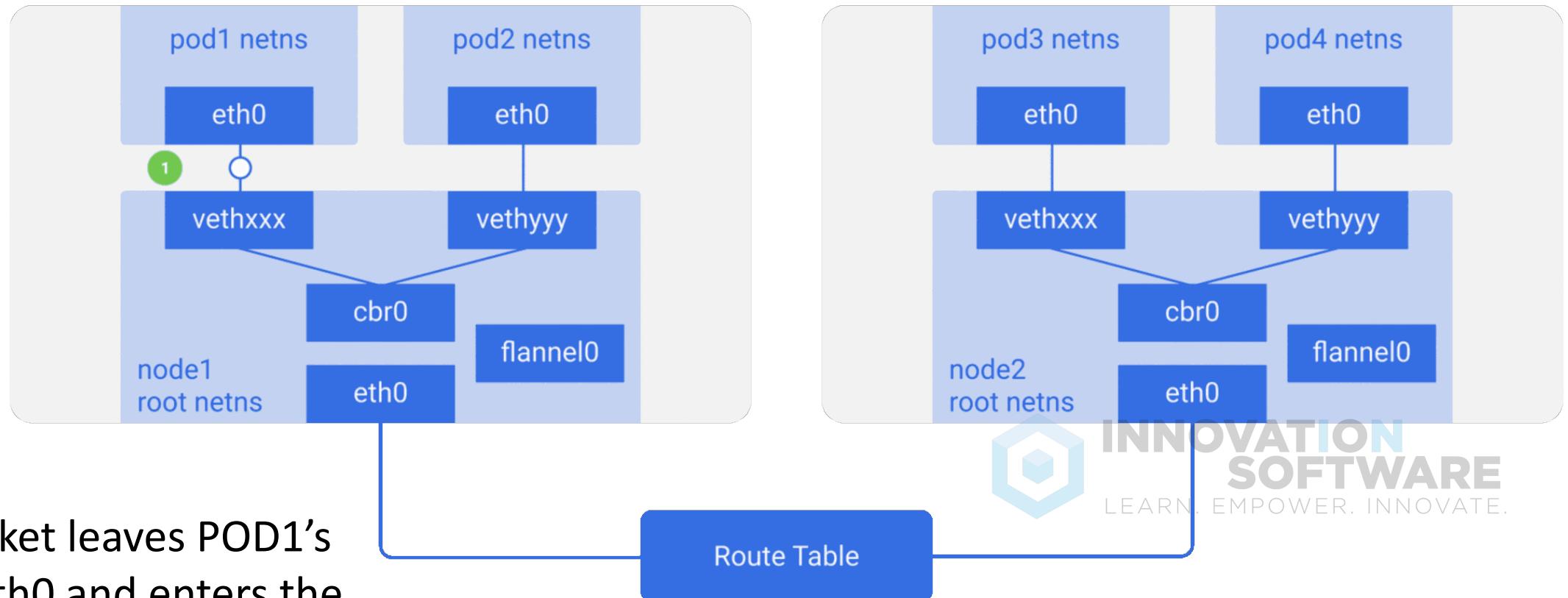
LEARN. EMPOWER. INNOVATE.



Overlay Networking

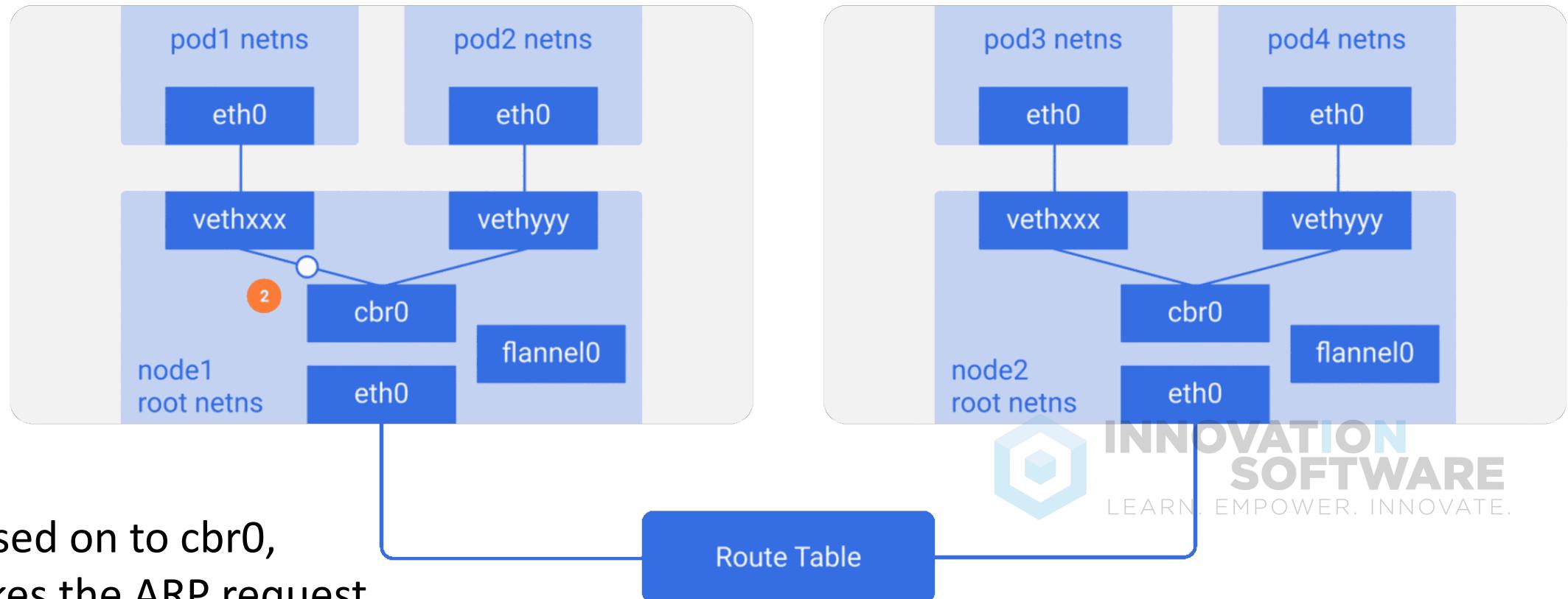
- Not required but very common
- Provide virtual IP space
- Avoid route limits (Public Cloud)
- Encapsulating a packet-in-packet which traverses the native network across nodes.
- Can reduce throughput

Flannel Overlay Networking



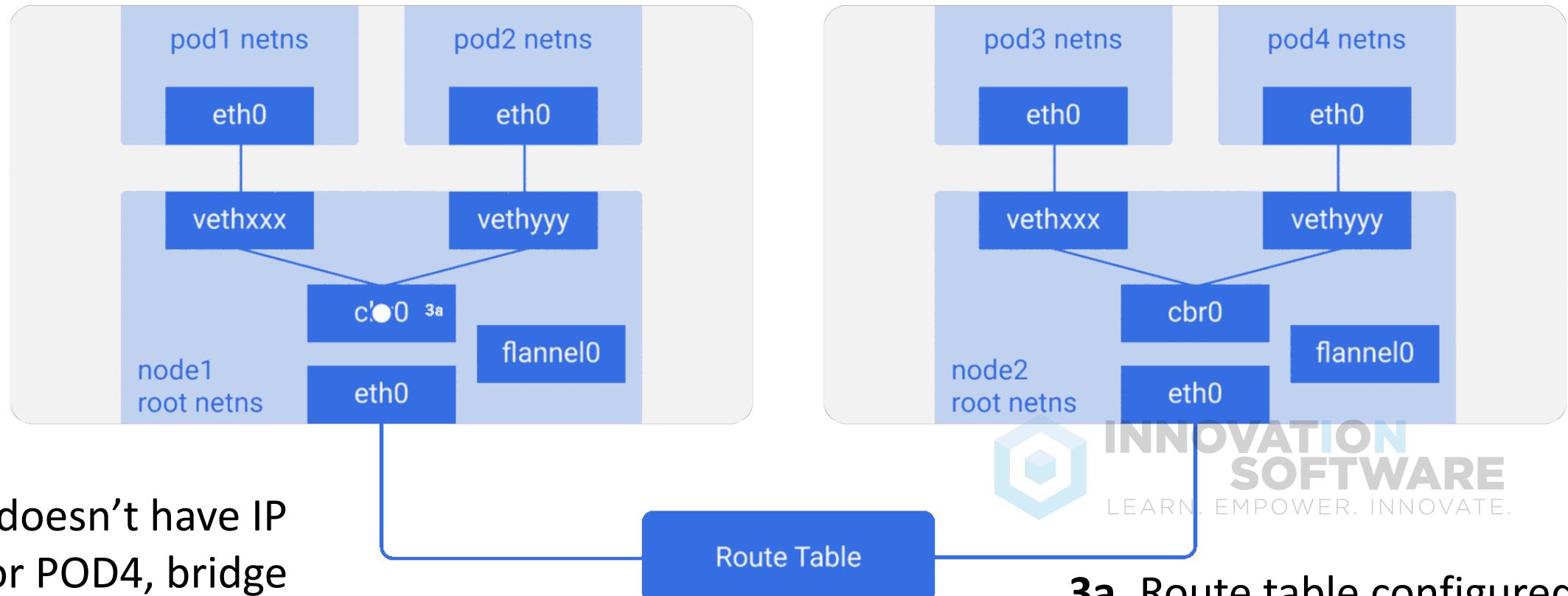
1. The packet leaves POD1's netns at eth0 and enters the root netns at vethxxx.

Flannel Overlay Networking



2. It's passed on to `cbr0`, which makes the ARP request to find the destination.

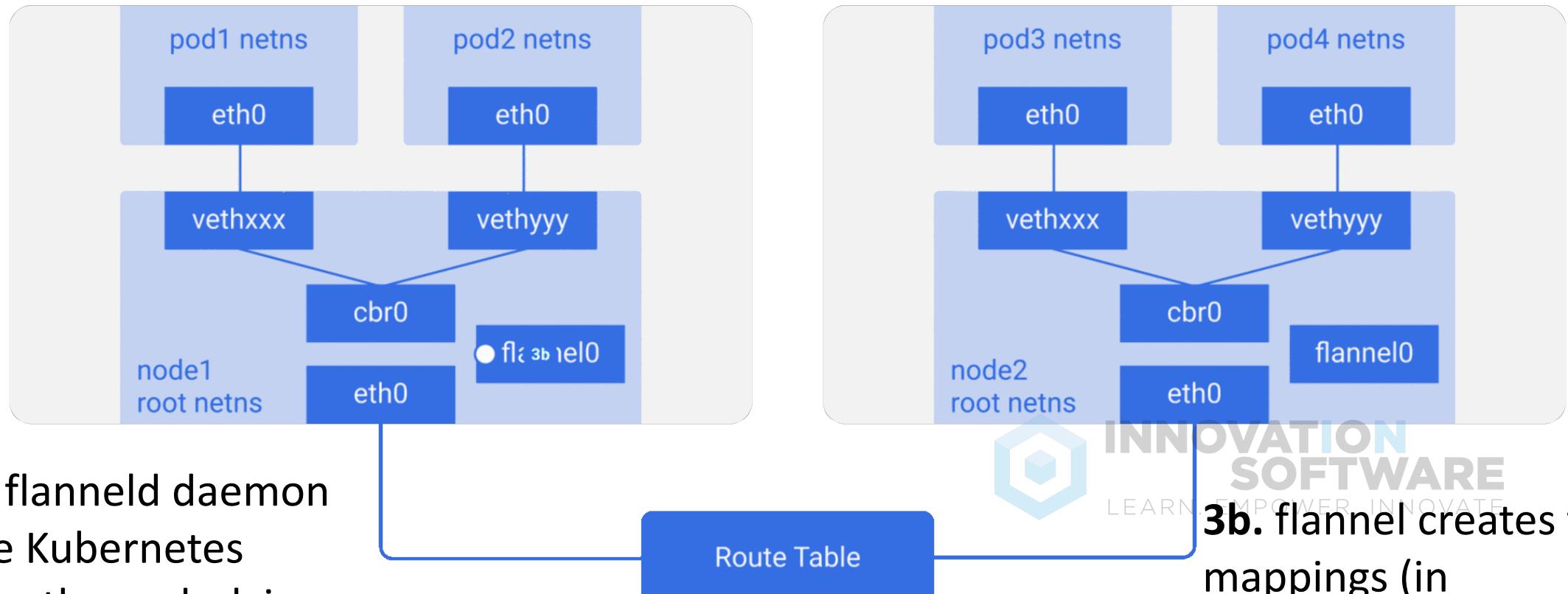
Flannel Overlay Networking



3a. Node doesn't have IP address for POD4, bridge sends to flannel0.

3a. Route table configured with flannel0 as the target for the pod network range .

Flannel Overlay Networking



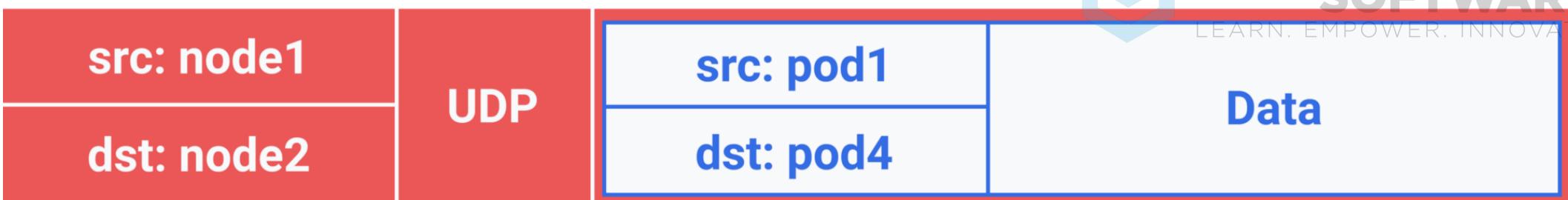
3b. As the flanneld daemon talks to the Kubernetes apiserver or the underlying etcd, it knows about all the pod IPs, what nodes they're on.

INNOVATION SOFTWARE
LEARN. EMPOWER. INNOVATE.
3b. flanneld creates the mappings (in userspace) for pods IPs to node IPs.

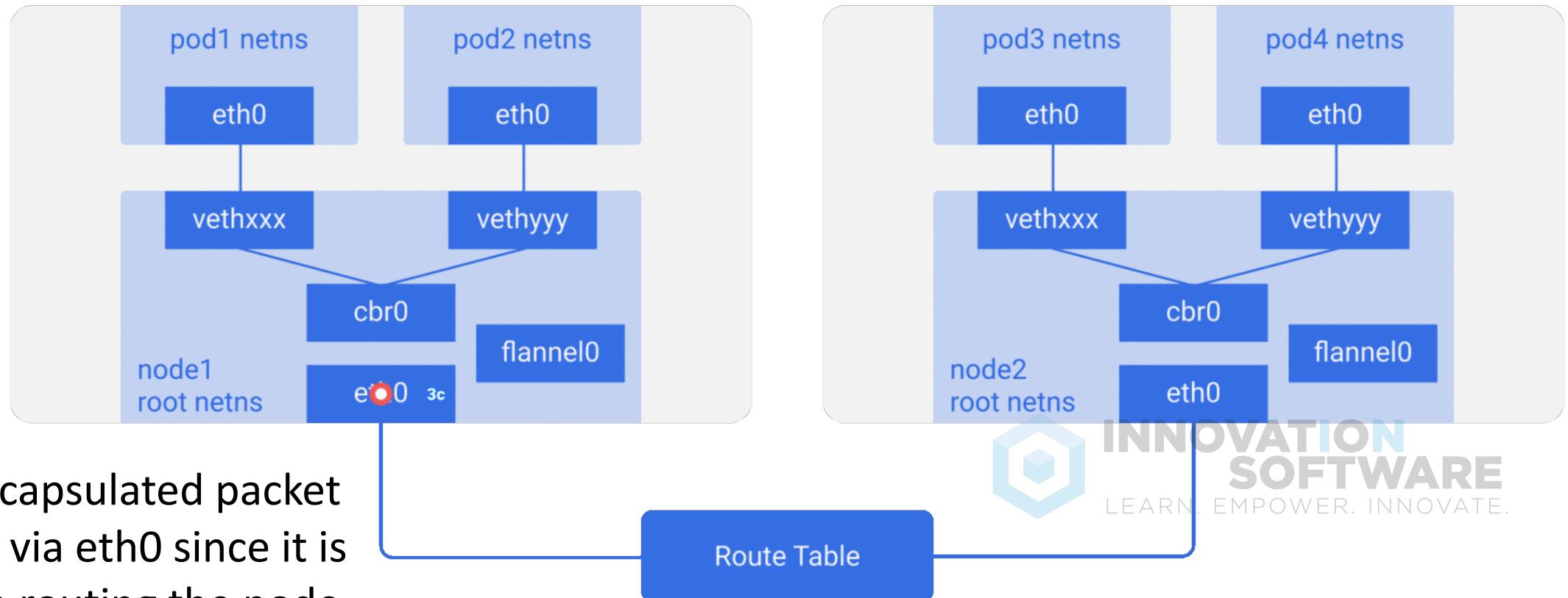
Flannel Overlay Networking



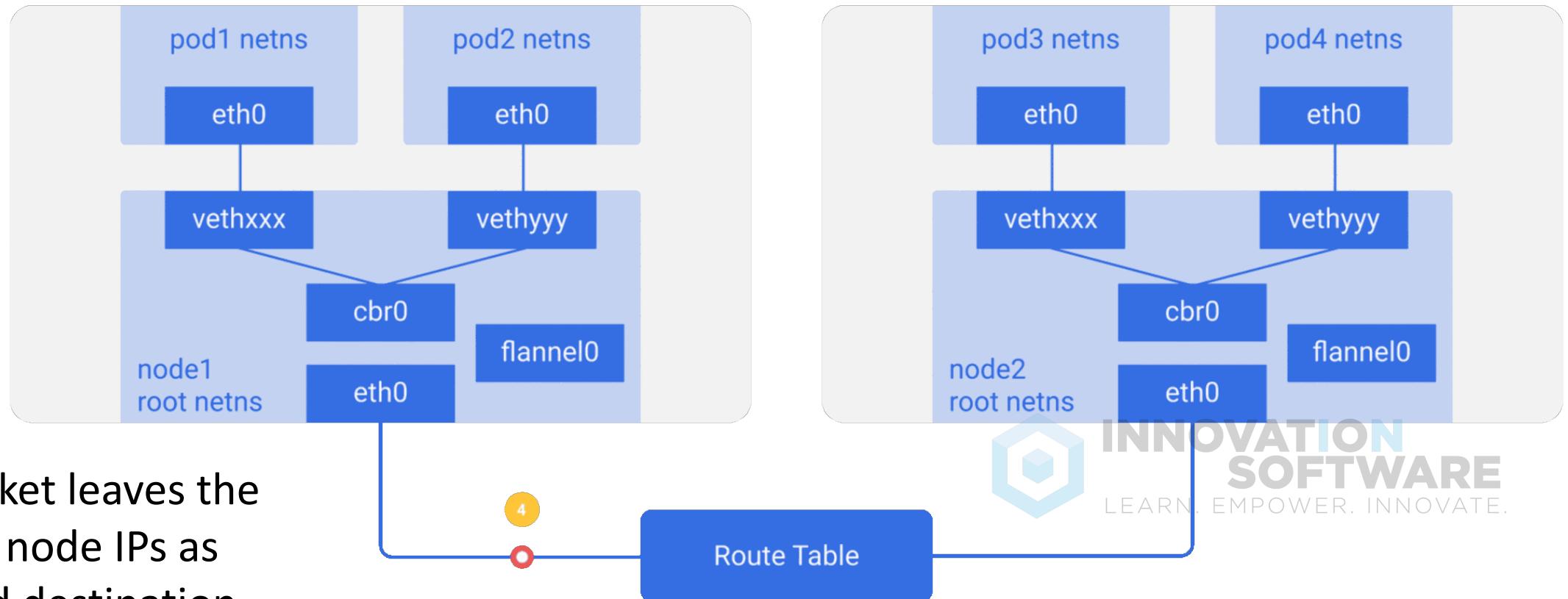
flannel0 takes this packet and wraps it in a UDP packet with extra headers changing the source and destinations IPs to the respective nodes, and sends it to a special vxlan port (generally 8472).



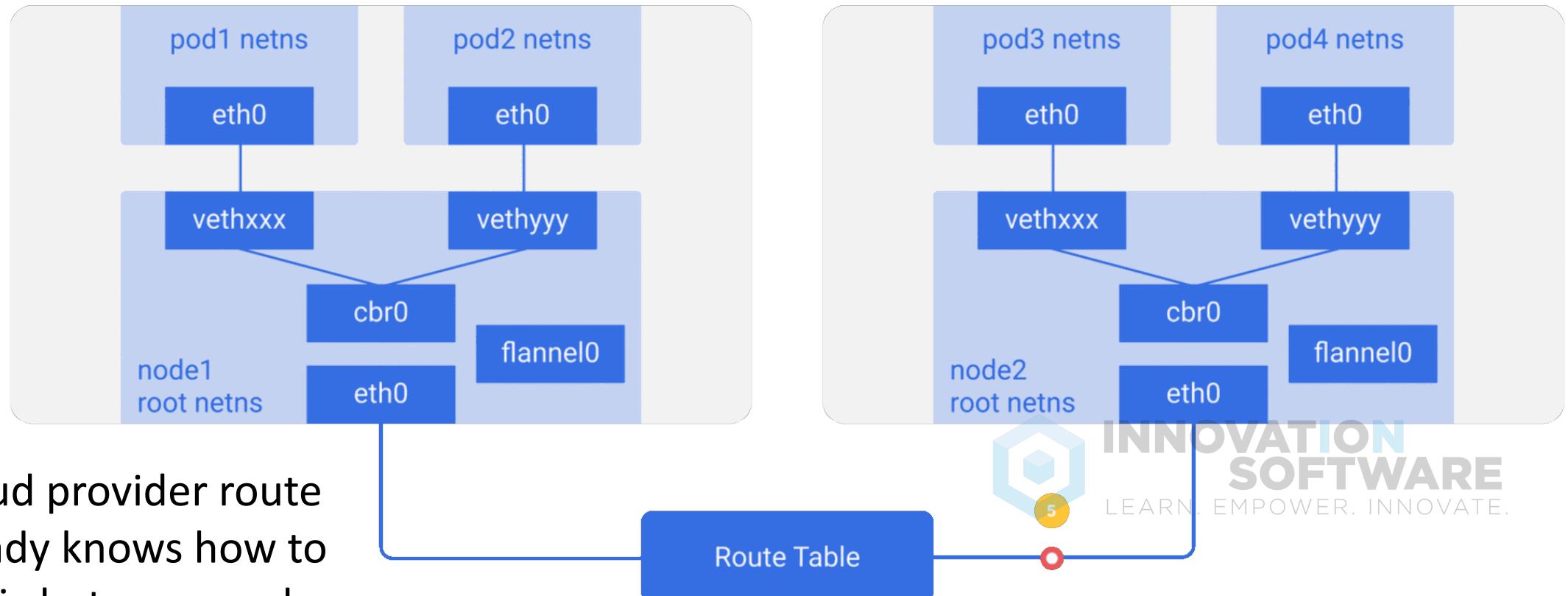
Flannel Overlay Networking



Flannel Overlay Networking

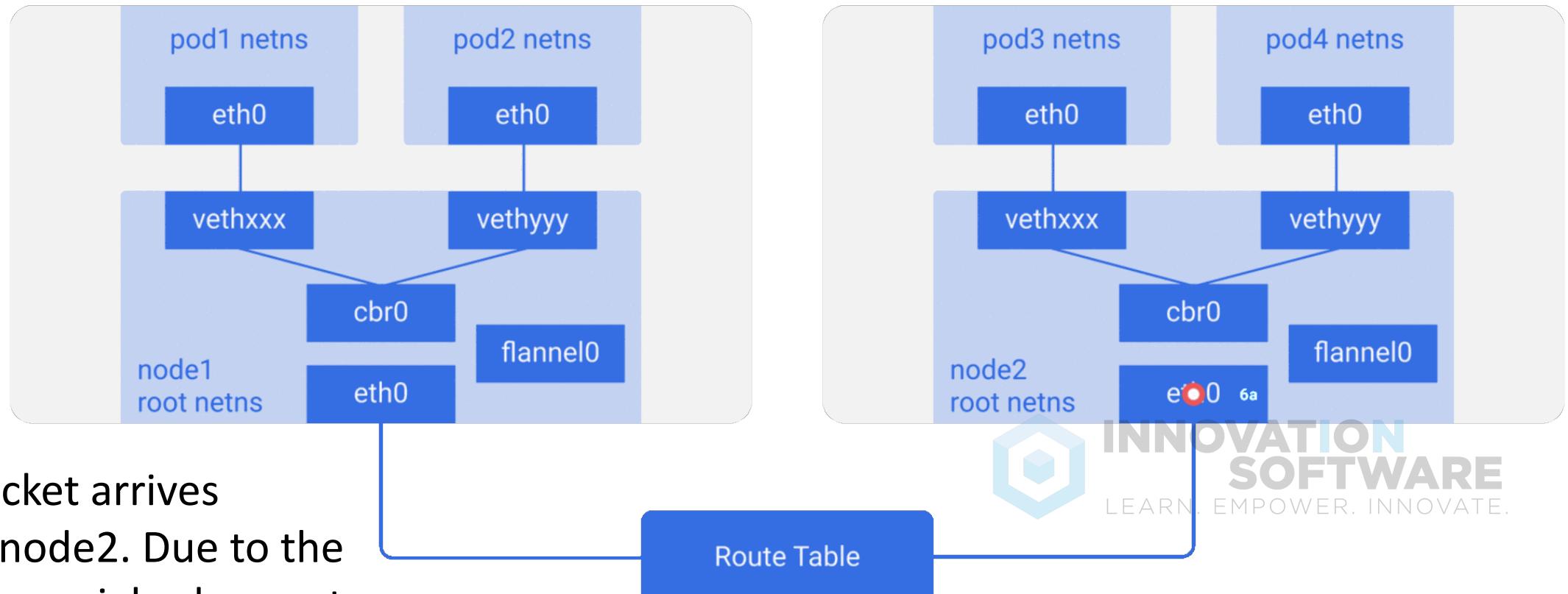


Flannel Overlay Networking



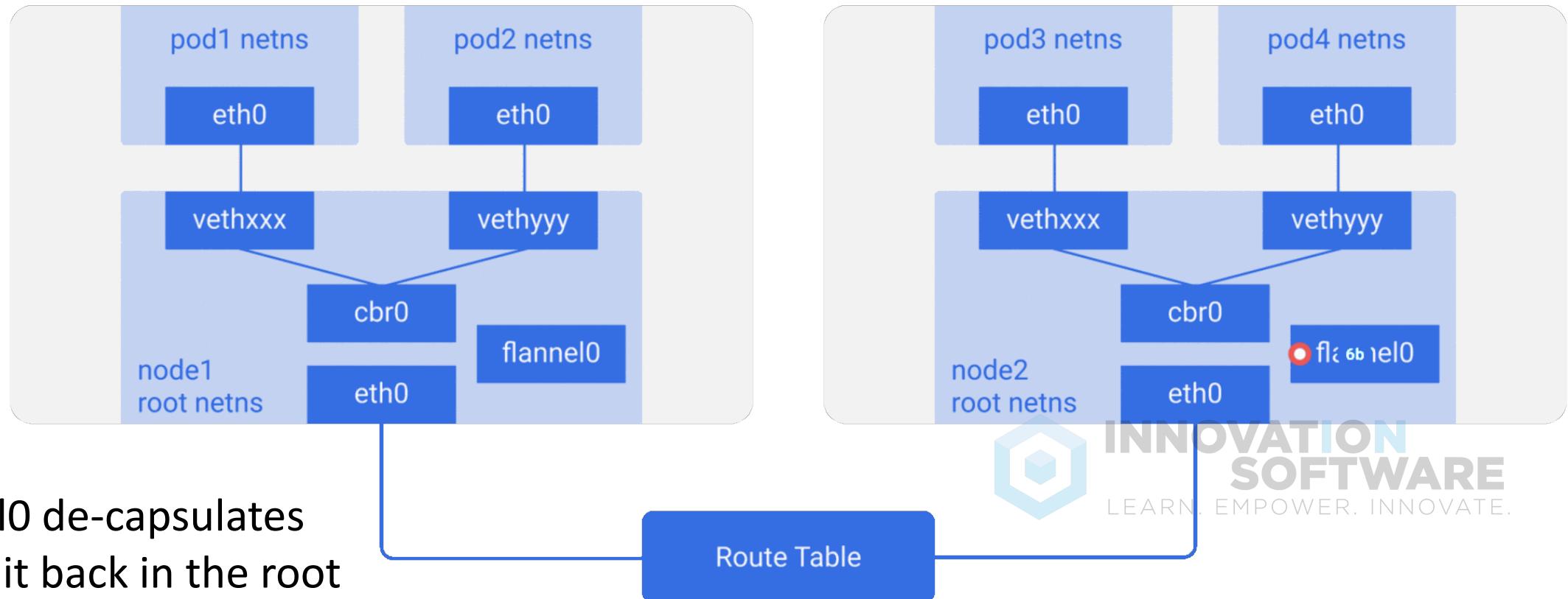
5. The cloud provider route table already knows how to route traffic between nodes, so it sends the packet to destination node2.

Flannel Overlay Networking

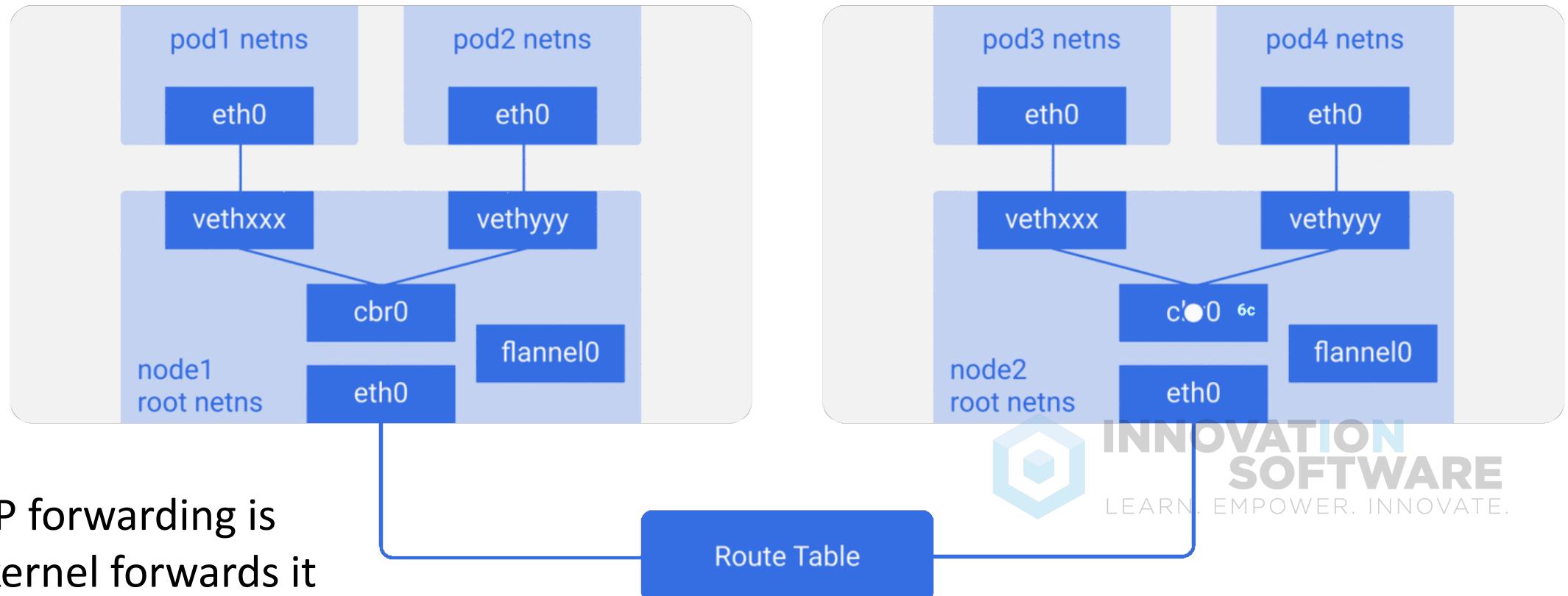


6a. The packet arrives at eth0 of node2. Due to the port being special vxlan port, kernel sends the packet to flannel0 .

Flannel Overlay Networking

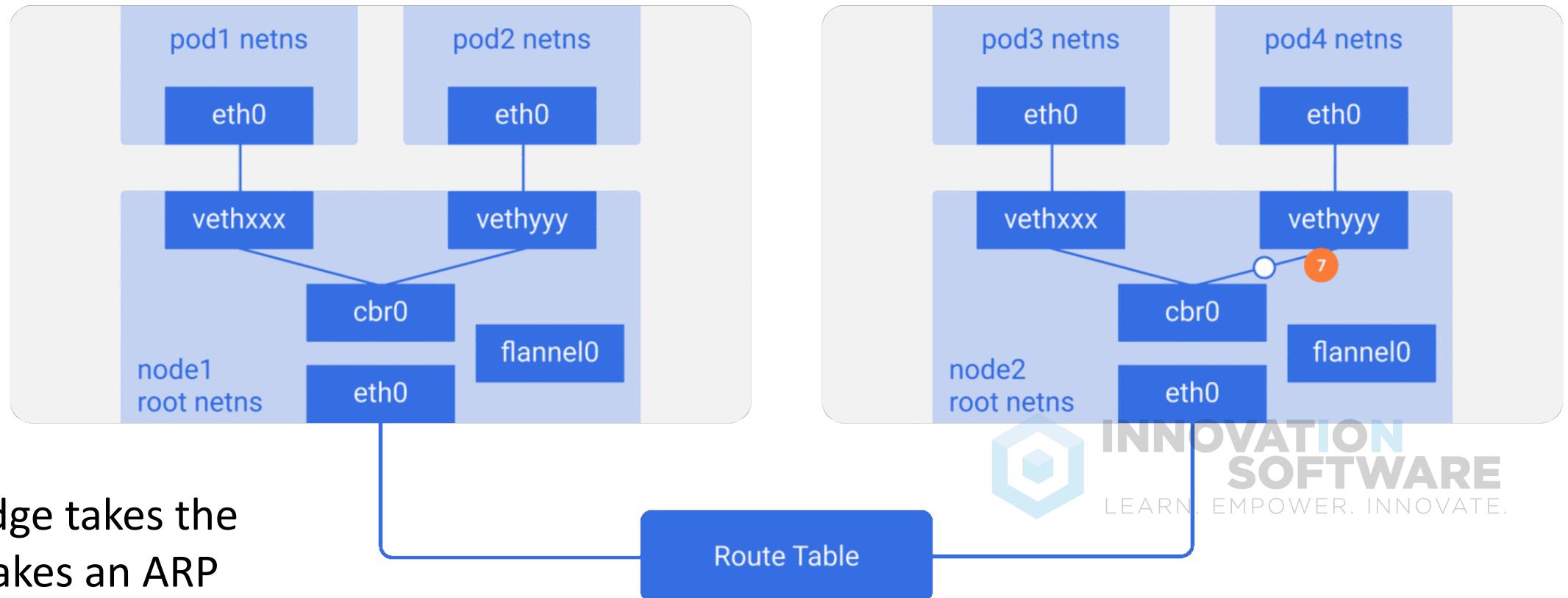


Flannel Overlay Networking



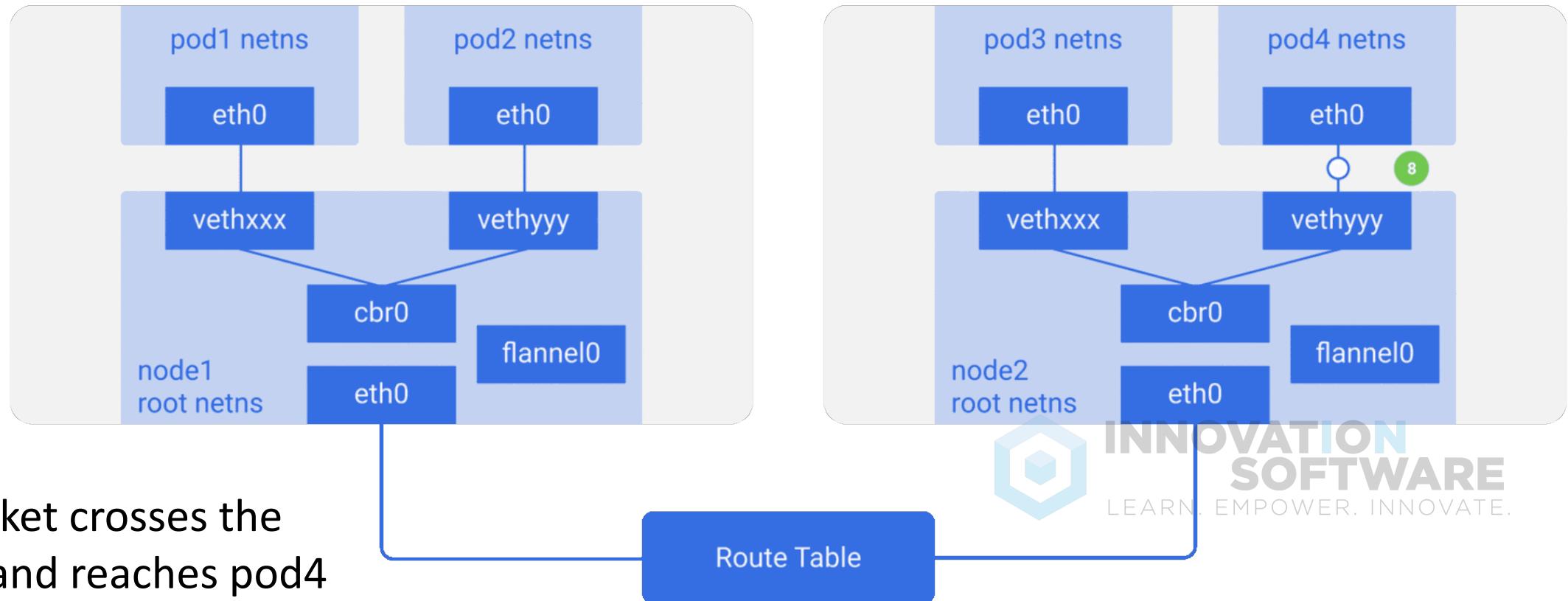
6c. Since IP forwarding is enabled, kernel forwards it to `cbr0` as per the route tables.

Flannel Overlay Networking



7. The bridge takes the packet, makes an ARP request and finds out that the IP belongs to `vethyyy`.

Flannel Overlay Networking



Deployments



What is a Deployment?

Kubernetes controller optimal for stateless applications

- Deployments allow you to declaratively manage pods, including replication
- Deployments support
 - Creating, rolling out, and rolling back changes to homogeneous set of pods
 - Scaling set of pods out and back declaratively
- Deployments include
 - Implicit Replica Set controller to handle pod replicas
 - Template spec of pods to be created and managed – no need to separately create pods
- Deployments used for web applications, mobile back-ends, API's

What if my Application isn't Stateless?

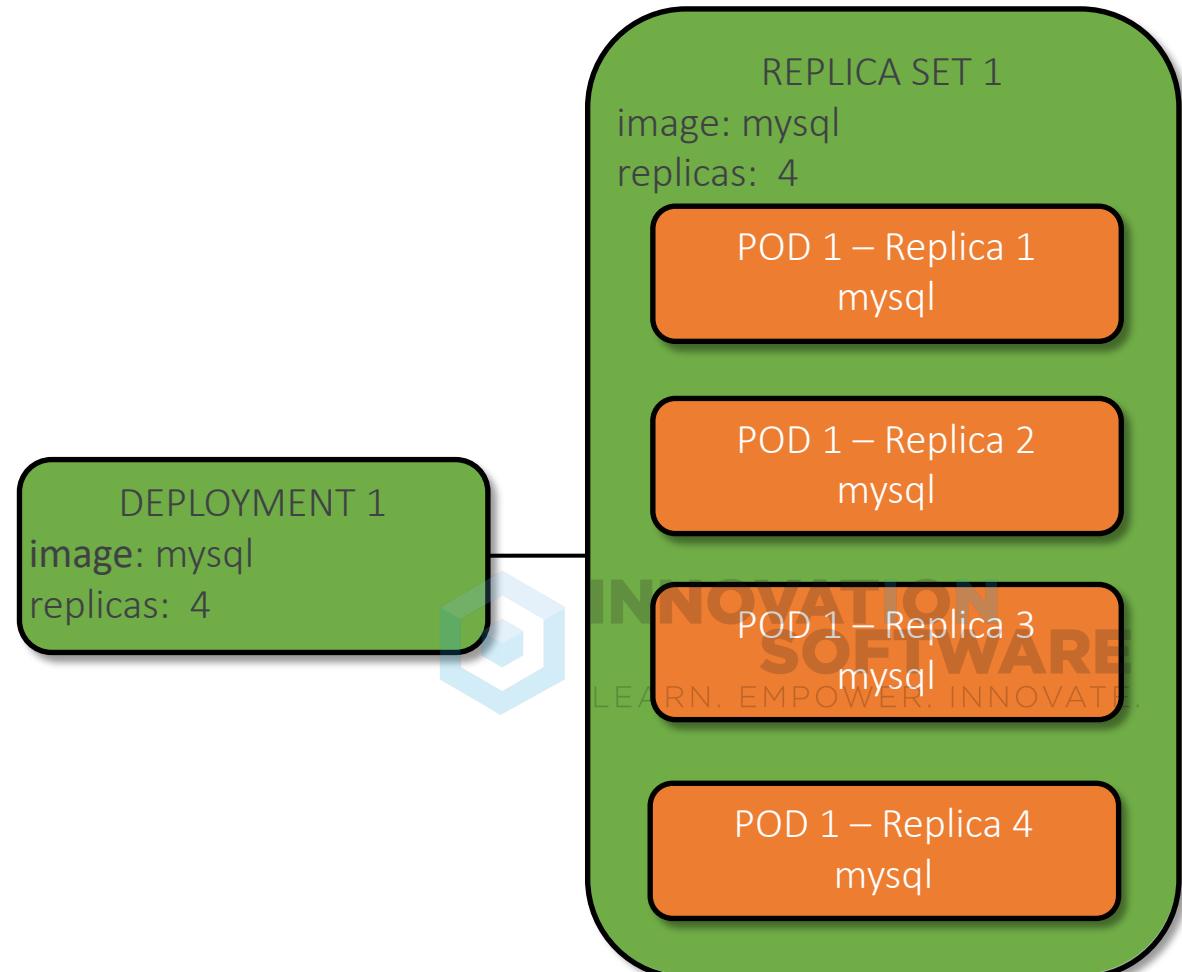
- Kubernetes provides other controller objects for applications that need different deployment schemes
- **StatefulSets** (previously PetSets) control deployment of pods for applications that need more stable deployment contexts
 - Pods in StatefulSets have unique ordinal, stable network identity and stable storage using persistent volumes
 - When pods are deployed, they are created in sequence of ordinals 1..N
 - Pod N must be running and ready before Pod N+1 is deployed
 - When pods are destroyed, they are terminated in reverse sequence N..1
- **DaemonSets** ensure that a replica of a specified pod is running on every node (or every selected node) in the cluster
- **Jobs** manage sets of pods where N must run to successful completion



Deployments Control ReplicaSet Controllers

Definition of how many replicated Pods should exist

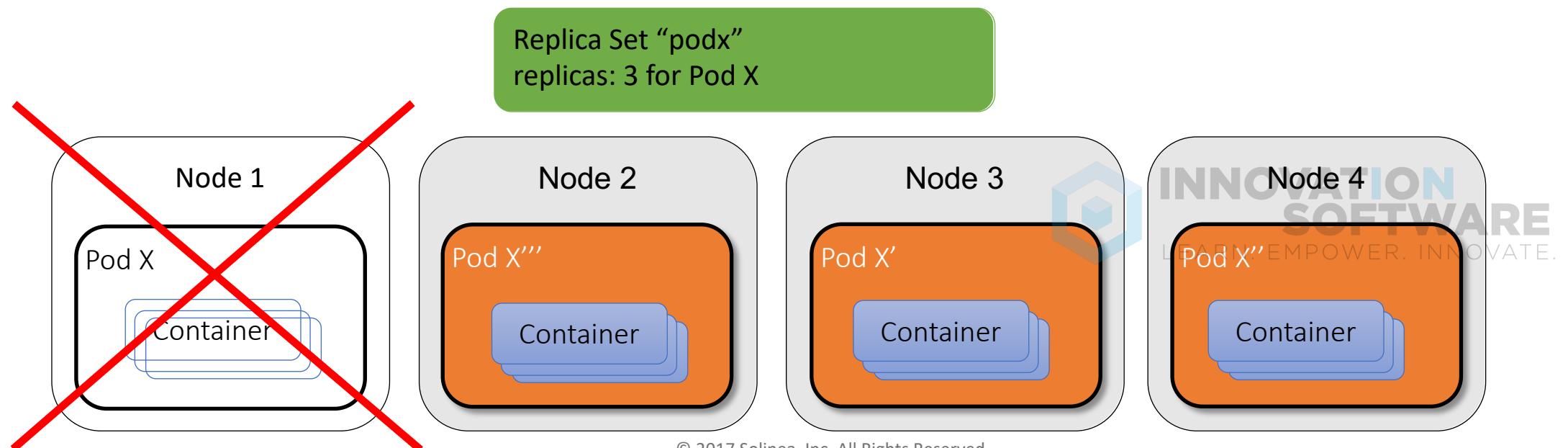
- Deployment creates and manages a Replica Set that manages a set of pods
- Replica count can be adjusted as needed to scale the Replica Set out and back
- Replica Set successor to the ReplicationController object



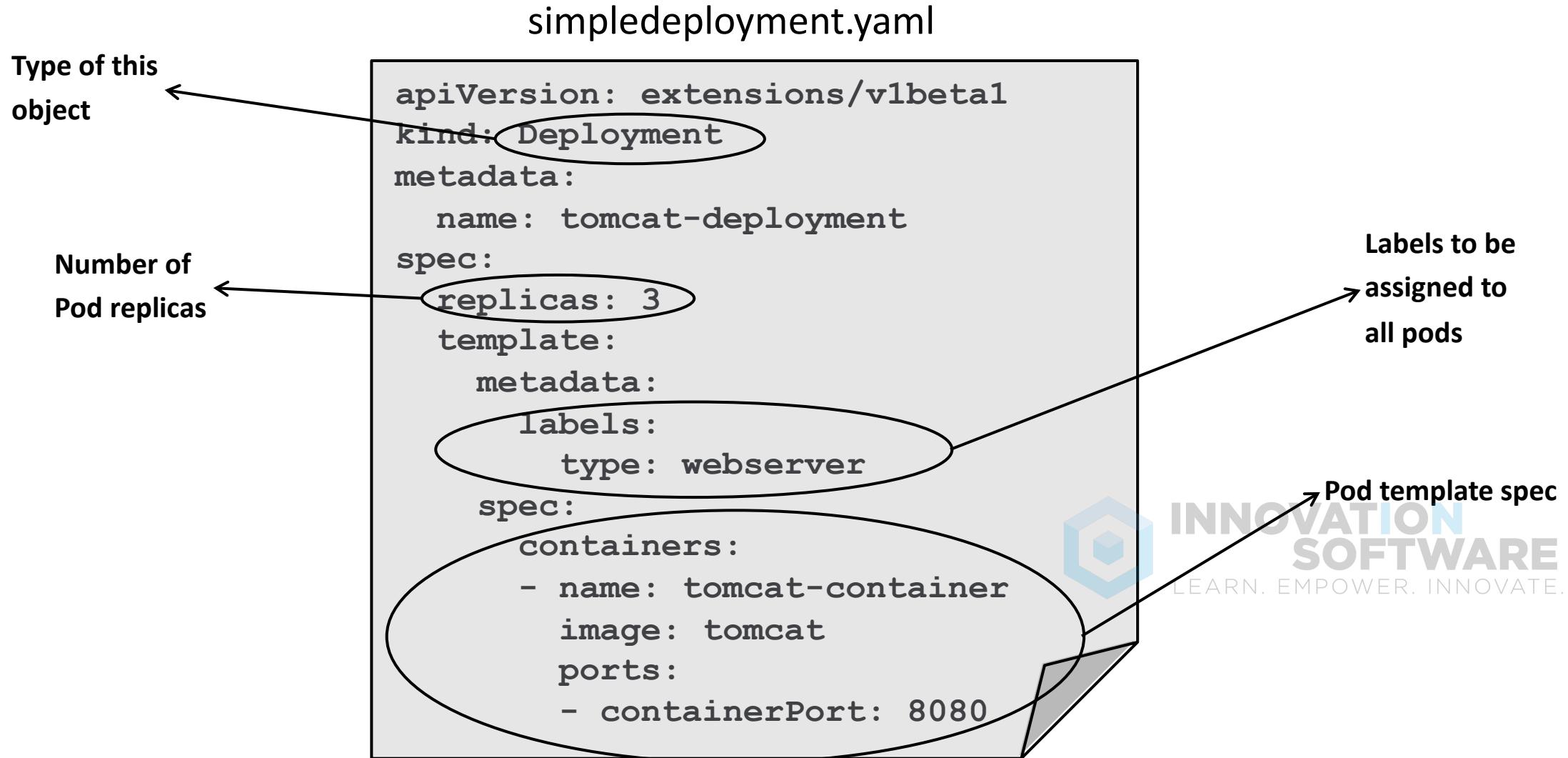
What is a Replica Set?

Provides scaling and high availability

- Replica count can be changed to provide scaling on demand as needed
- If the node hosting a pod fails, the Kubernetes cluster will recreate the pod elsewhere to achieve the target number of replicas

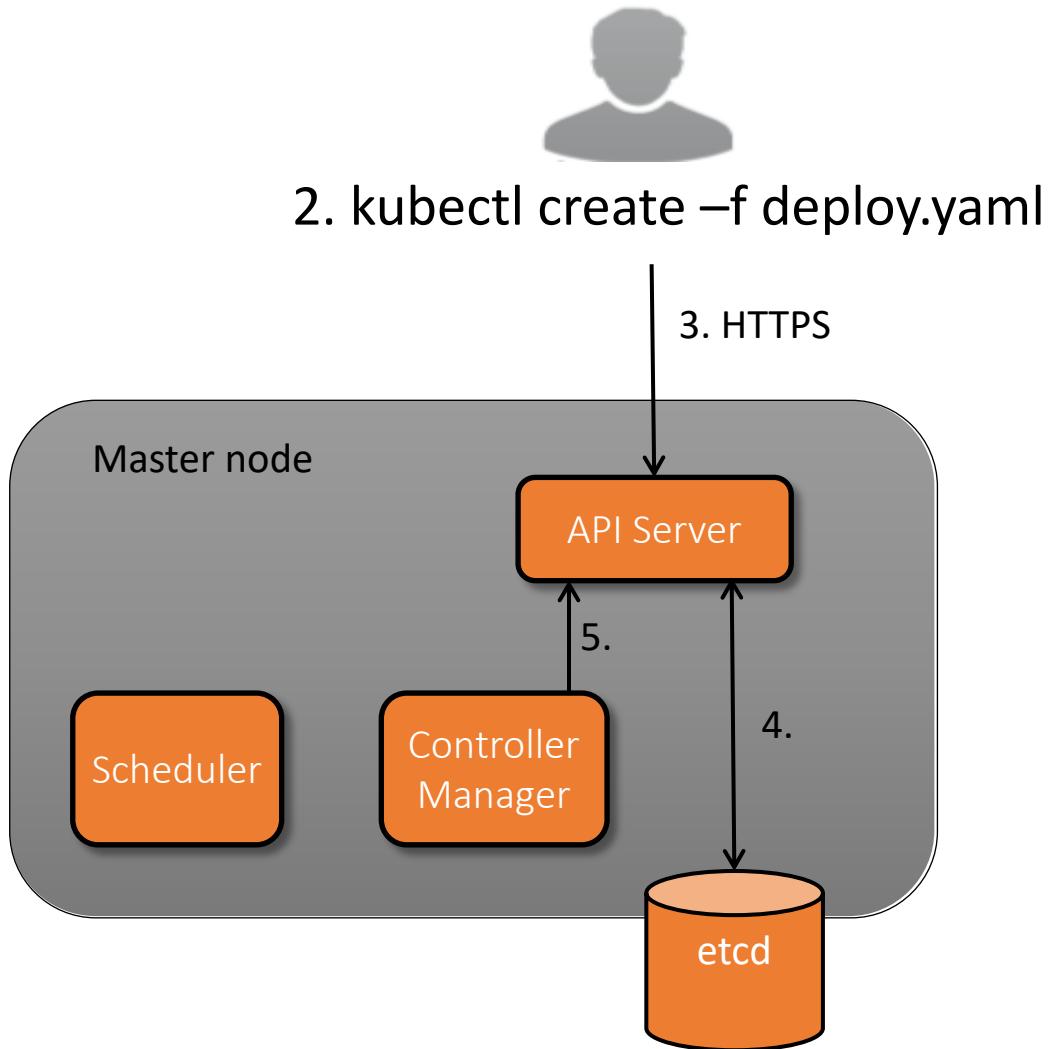


Examining a Deployment Manifest File



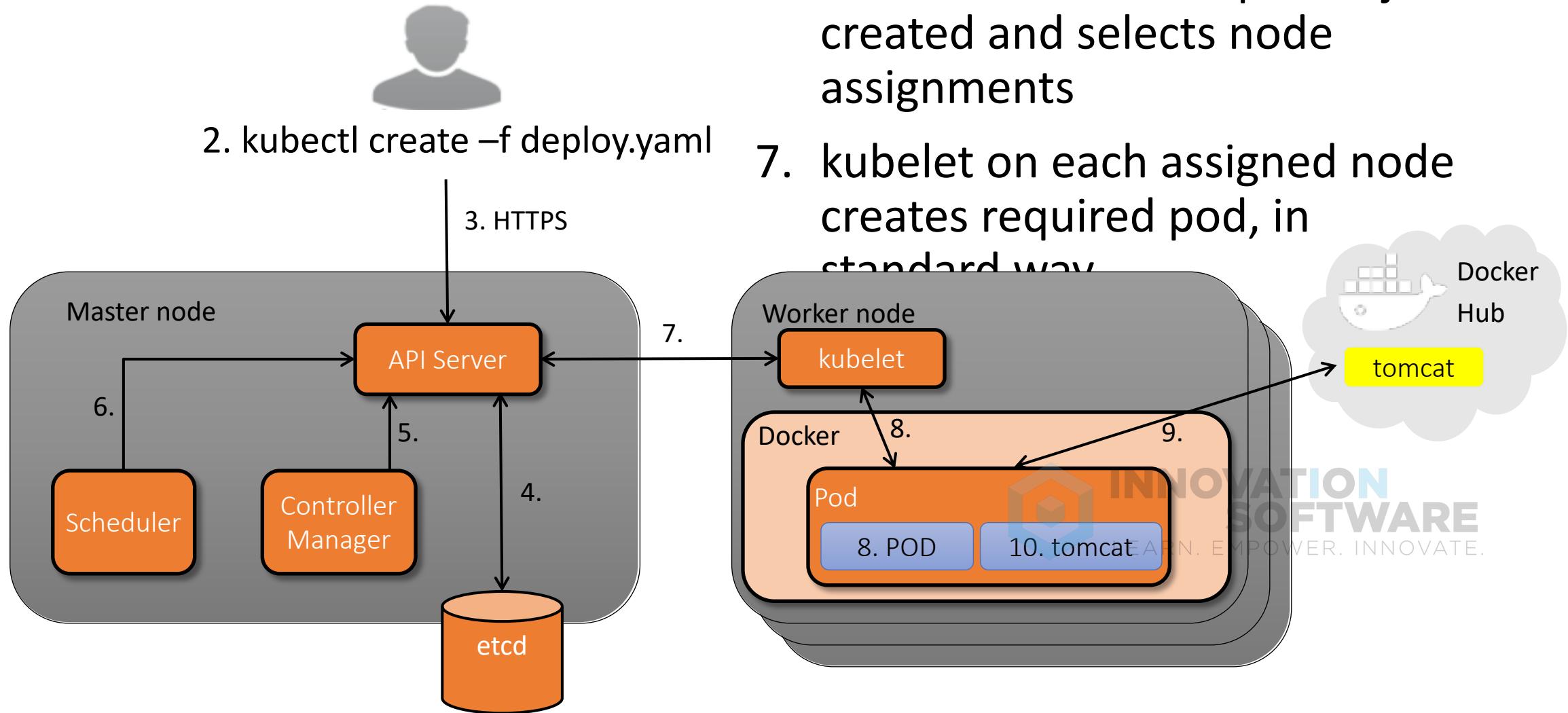
INNOVATION SOFTWARE
LEARN. EMPOWER. INNOVATE.

Deployment Creation Process



1. User writes a deployment manifest file
2. User requests creation of deployment from manifest via CLI
3. CLI tool marshals parameters into K8s RESTful API request (HTTP POST)
4. kube-apiserver creates new deployment object record in etcd, and new Replica Set object
5. kube-controller-manager sees new Replica Set and
 - Evaluates state of existing vs. required replicas
 - Submits pod creation requests to API to create required number of replicas

Deployment Creation Process



INNOVATION
SOFTWARE
EARN. EMPOWER. INNOVATE.

Deployments Management



INNOVATION
SOFTWARE

LEARN. EMPOWER. INNOVATE.



Deployments Control ReplicaSets

The deployment creates a ReplicaSet that controls pod creation

Deployment name defined in the yaml	Number of replicas defined in the spec	Number of existing pods	Number of Pods that match the Deployment config	Actual number of Pods running
\$ kubectl get deployments NAME tomcat-deployment	DESIRED CURRENT	3 3	UP-TO-DATE AVAILABLE	3

\$ kubectl get rs	NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
	tomcat-deployment-1699985759	3	3	3	3	2m



INNOVATION
SOFTWARE
POWER. INNOVATE.

Replica Set Details

Replica Set name is <Deployment name>-<pod template hash value>

```
$ kubectl describe replicaset tomcat-deployment-1699985759
Name:          tomcat-deployment-1699985759
Namespace:     default
Image(s):      tomcat
Selector:      pod-template-hash=1699985759,type=webserver
Labels:        pod-template hash-1699985759
               type=webserver
Replicas:      3 current / 3 desired
Pods Status:   3 Running / 0 Waiting / 0 Succeeded / 0 Failed
...
```

Selector uses labels
from pod template
and template hash



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Pod Naming

Pod name is <Replica Set name>-<pod unique random string>

```
$ kubectl get pods
```

NAME	READY	STATUS
tomcat-deployment-1699985759-h11sz	0/1	ContainerCreating
tomcat-deployment-1699985759-ka13j	0/1	ContainerCreating
tomcat-deployment-1699985759-u03b5	1/1	Running

```
$ kubectl get pods
```

NAME	READY	STATUS
tomcat-deployment-1699985759-h11sz	1/1	Running
tomcat-deployment-1699985759-ka13j	1/1	Running
tomcat-deployment-1699985759-u03b5	1/1	Running

Pod name based on
Replica Set name

Random string to
differentiate Pods

Status of Pods



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Modifying a Deployment to Trigger a Rollout

Multiple ways to change a Deployment configuration

- Change the manifest file and apply it via *kubectl apply*
- Change specific Deployment attribute via *kubectl set*
- Edit the Deployment config in the cluster via *kubectl edit*
- **Any changes to the Pod template will trigger a rollout of the Deployment to create and replicate new Pods with the new template**
 - This means any changes – even things like pod labels!

Pausing a Deployment

Pause a Deployment to temporarily halt rollout of updated pods

```
$ kubectl set image deployment/tomcat-deployment tomcat-container=tomcat:8.5.5;  
kubectl rollout pause deployment/tomcat-deployment  
deployment "tomcat-deployment" image updated  
deployment "nginx-deployment" paused
```

```
$ kubectl rollout resume deployment/tomcat  
deployment "tomcat-deployment" resumed
```

```
kubectl rollout status deployment/tomcat-deployment  
deployment "tomcat-deployment" successfully rolled out
```



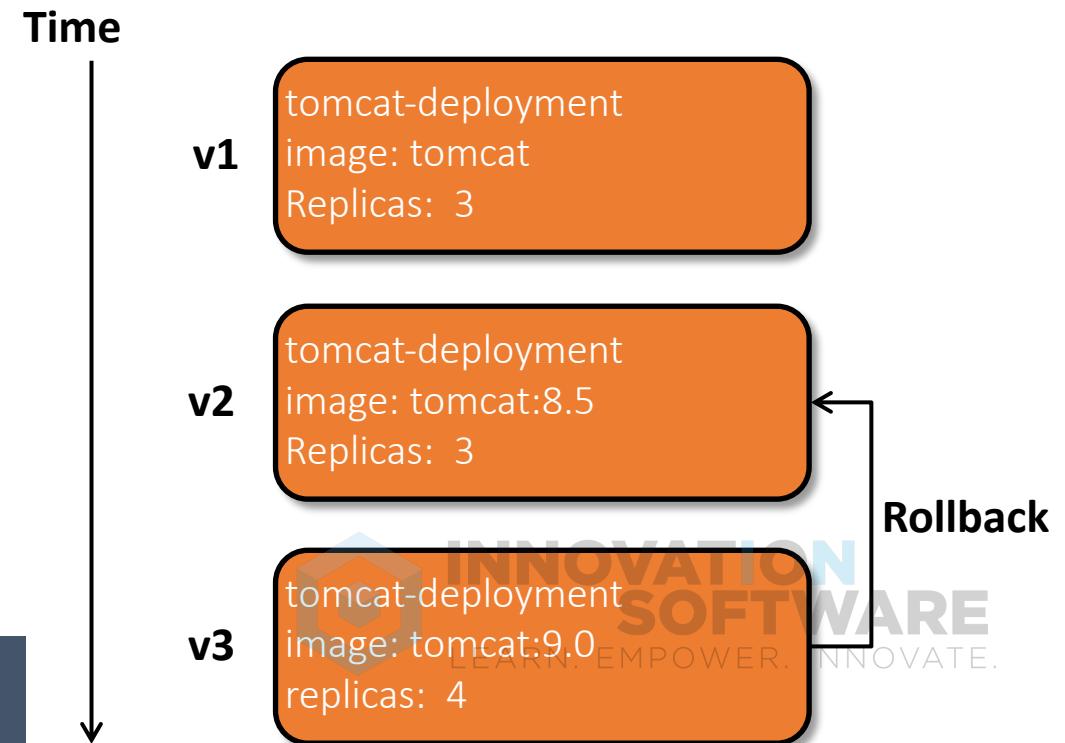
INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Checking Deployment Rollout History

Kubernetes tracks revisions made to a Deployment

- Users can query history of a deployment and see how many versions existed, and what change was made
 - Need to use *-record* flag on kubectl to record details of change commands
 - History allows you to roll back state of a Deployment to a previous version

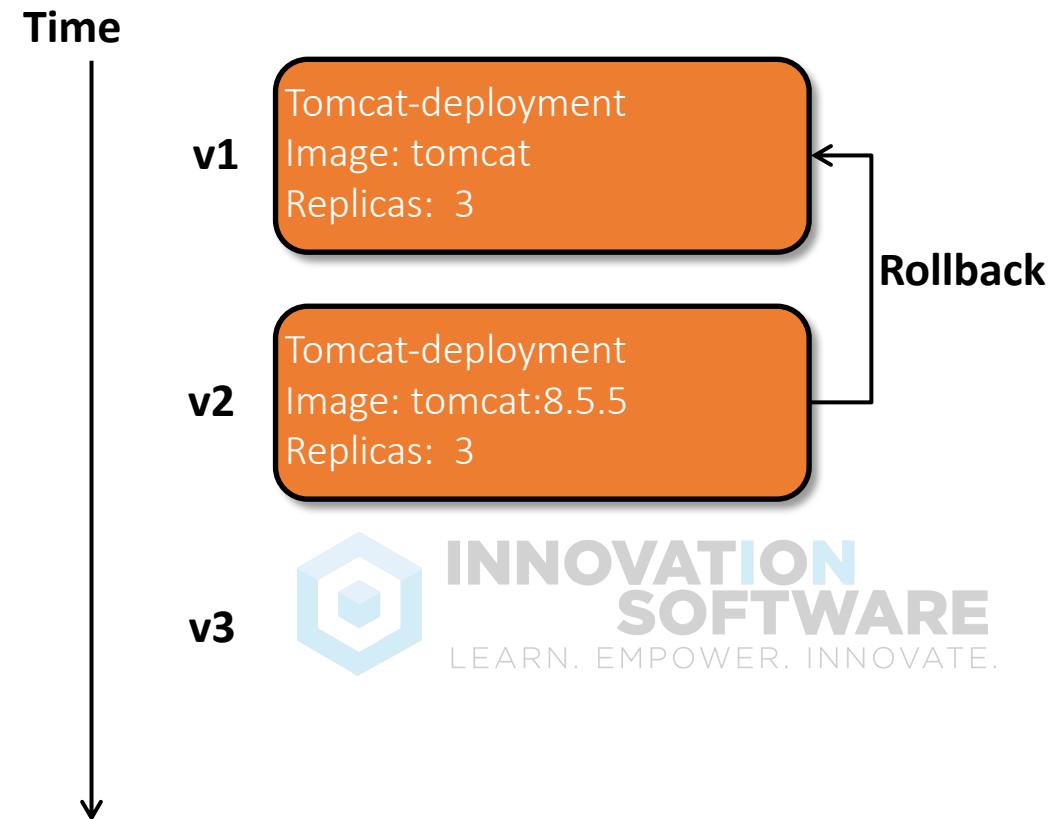
```
$ kubectl rollout history deploy/<deployment>
```



Rolling back a Deployment

Undoing Deployment changes via rollback operation

- You can undo the last change or roll back to a specific version
- The revision order will be changed to reflect the change
 - In this example, revision v1 will become the new revision v3
 - Version numbers increase monotonically
- Revision state stored in the corresponding Replica Sets

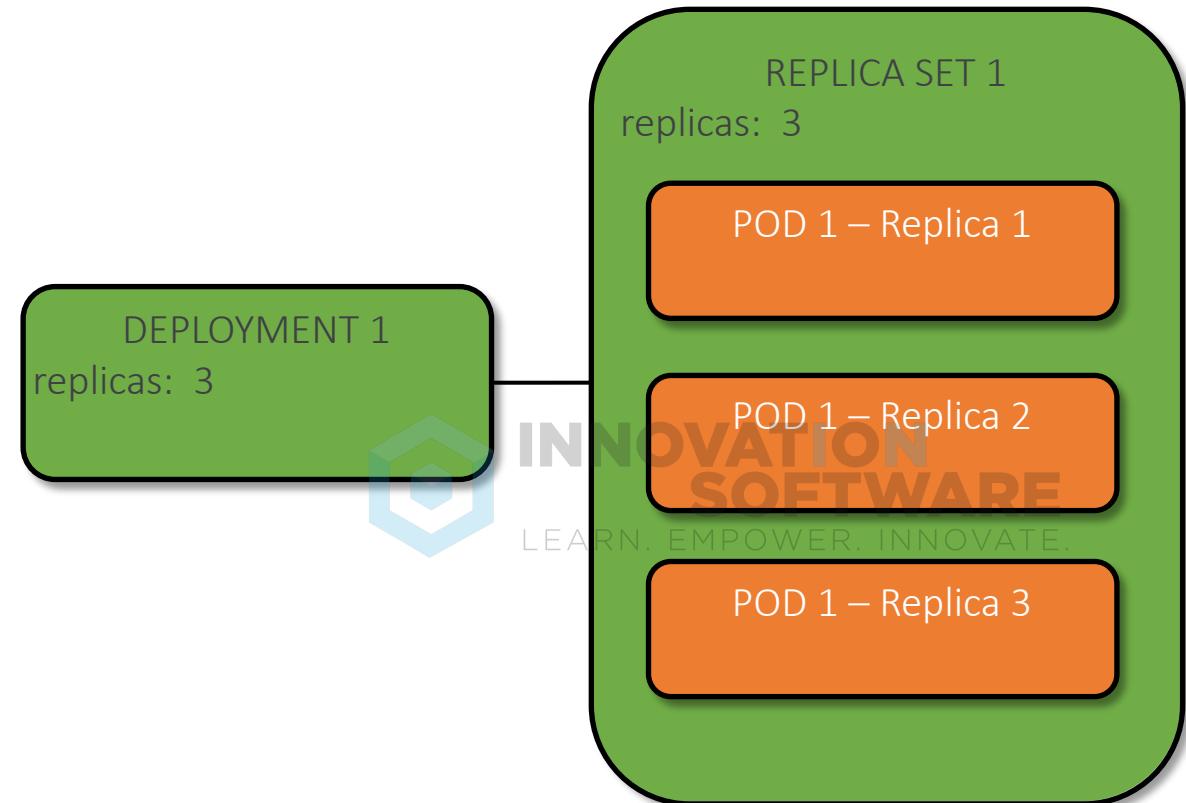


**INNOVATION
SOFTWARE**
LEARN. EMPOWER. INNOVATE.

Deleting a Deployment

Deleting a Deployment will delete its Replica Set and all its Pods

- By default, when Deployment is deleted, its Replica Sets are deleted
- Deletion of Replica Set cascades to deletion of pods managed by the rs
- Any Replica Sets reflecting previous versions of the Deployment will also be deleted



Scaling Deployments



INNOVATION
SOFTWARE

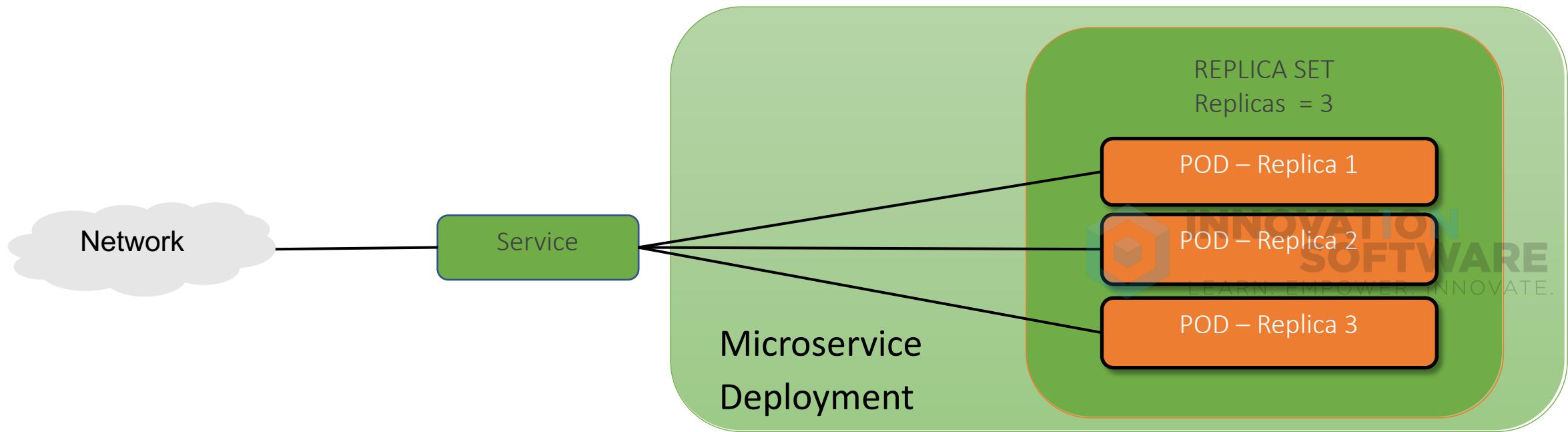
LEARN. EMPOWER. INNOVATE.



Kubernetes Deployments for Microservices

Kubernetes is perfect for Microservices management

- Applications built as services or collections of microservices easily modeled using Kubernetes Deployment and Service resources
- Key property is that application or component can be scaled horizontally



Application Scaling Strategies

Several ways to achieve scalability

- **Pre-defined application scale:** application deployed with a predefined, static number of resources => not the Kubernetes way
- **Manual scaling:** deployment scale reconfiguration driven by operator
- **Auto-scaling:** automatic scaling of resources based on a defined trigger (number of hits, CPU usage, etc)

Application Scaling Strategies

Manual scaling

- K8s supports manual scaling of Deployment to adjust replica count
- Operator just needs to adjust declaration of how many replicas are desired, and K8s system will manage to that number
 - Pod creation and placement completely automatic
 - Pods automatically re-created if nodes fail

```
$ kubectl scale --replicas=2 deployment/tomcat
```

```
$ kubectl scale --replicas=8 deployment/tomcat
```

```
$ kubectl scale --replicas=4 deployment/tomcat
```

```
$ kubectl edit deployment/tomcat
```



INNOVATION
SOFTWARE

LEARN. EMPOWER. INNOVATE.

```
$ kubectl apply -f simpledeployment.yaml
```

Application Scaling Strategies

Auto-scaling in Kubernetes

- Kubernetes has a controller object for automatic scaling of Deployments (or Replica Sets or ReplicationControllers)
- HorizontalPodAutoscaler is control loop to adjust scale of pod set depending on one or more metrics, evaluated at configurable interval (default 30s)
 - Requires Heapster to be deployed on the cluster to provide metric data
 - Metrics include CPU and RAM utilization for pods
- Typical scenario: increase Deployment replica count when average Pod CPU utilization is above threshold value for specified period of time
- Note: application must support horizontal scaling!

Automatic Scaling of a Deployment

Creating a HorizontalPodAutoscaler resource

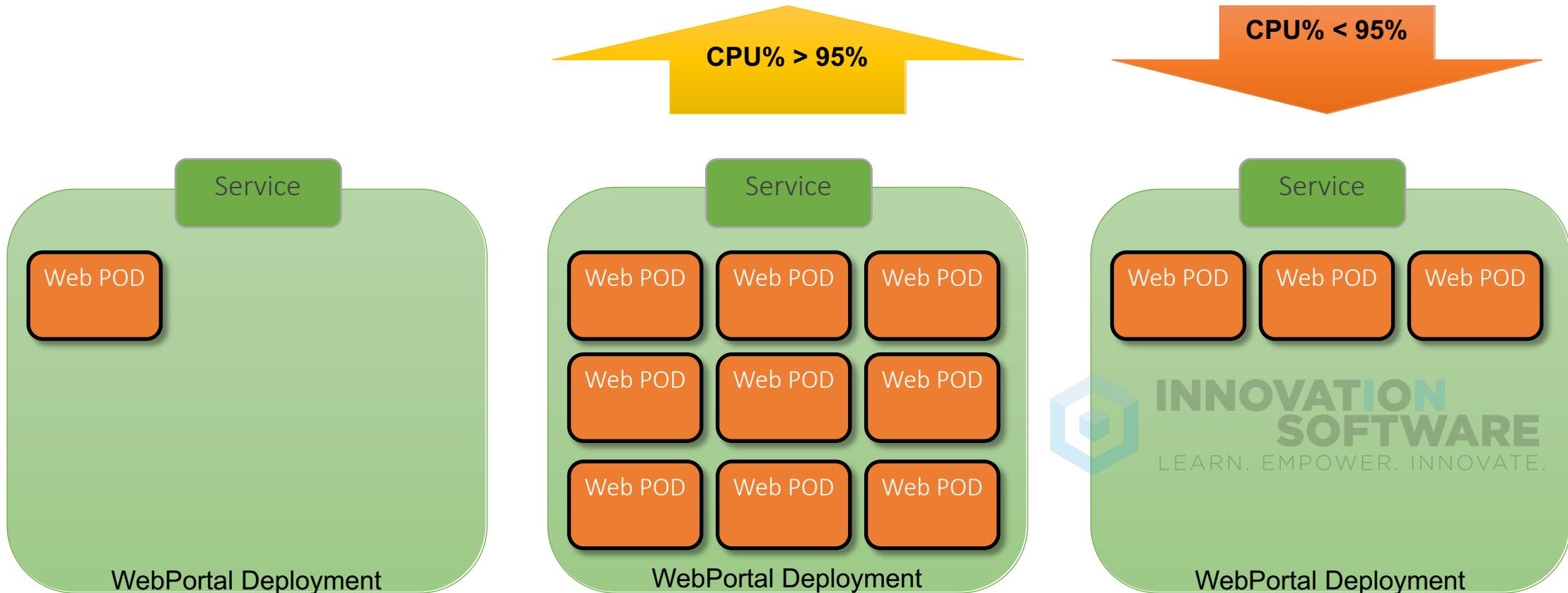
- The hpa is created similarly to any other Kubernetes resource
 - Use a manifest file for source tracking
 - Can create directly with *kubectl*
- 30 seconds is the default for considering the threshold
- Scaling is metrics-driven, either resource metrics from Heapster or custom metrics API

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: tomcat-autoscaler
spec:
  maxReplicas: 10
  minReplicas: 2
  scaleTargetRef:
    kind: Deployment
    name: tomcat-deployment
  targetCPUUtilizationPercentage: 75
```

simplescaler.yaml

Auto-Scaling of a Deployment

HorizontalPodAutoscaler with a 95% CPU usage threshold



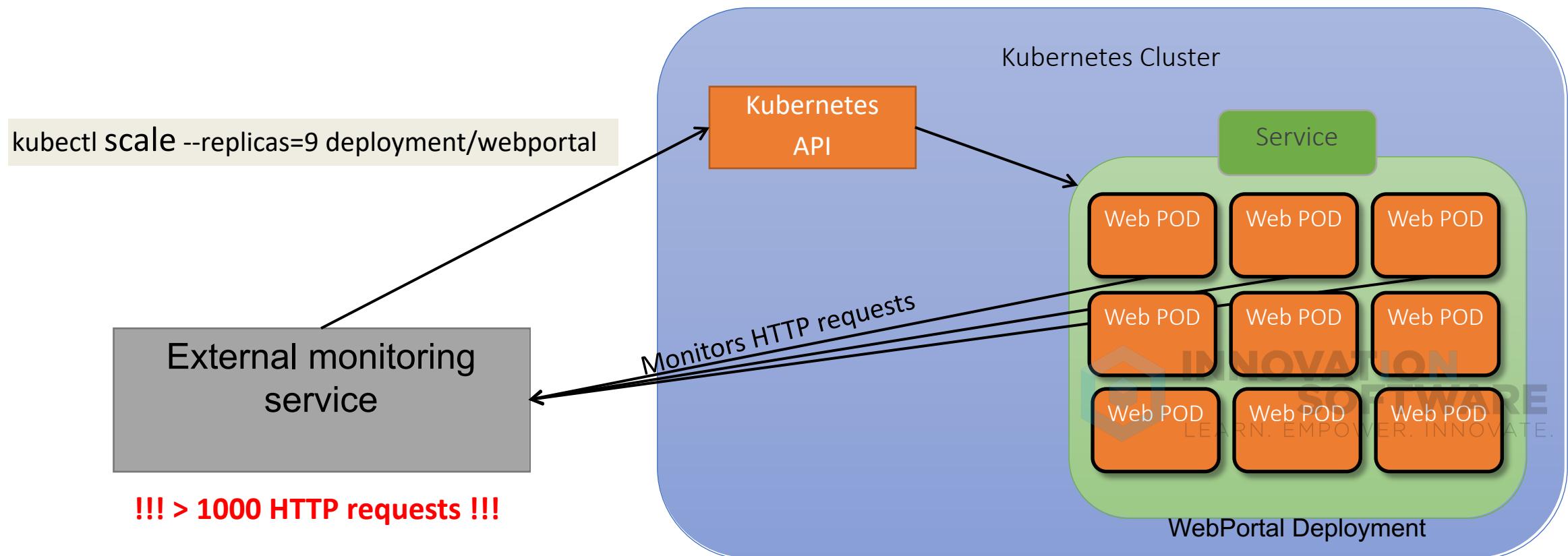
Application Scaling Strategies

Externally-driven auto-scaling

- Kubernetes API can be used to change Deployment replica counts remotely
- Any monitoring application that can send REST calls (e.g. Nagios) could be used to drive automatic scaling externally
- Very flexible approach, but more complex to implement
- **Needs robust testing before going into production**

Application Scaling Strategies

Auto-scaling driven by external system





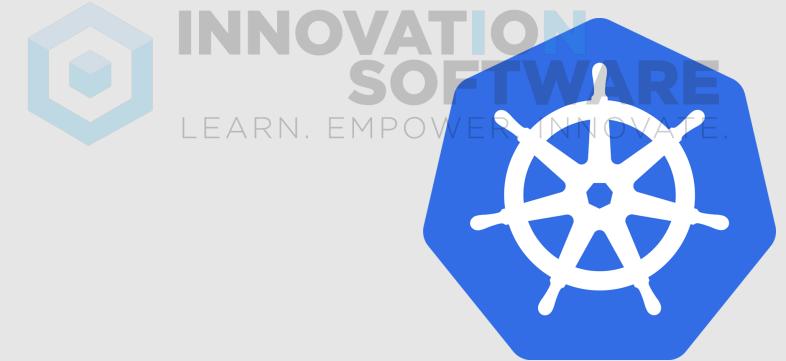
Questions



INNOVATION
SOFTWARE

LEARN. DISCOVER. INNOVATE.

Deployment Strategies Overview



What is a Deployment Strategy?

Approaches to manage risks on updating Deployments

- On each Deployment update/change, all pods in the deployment will be deleted and recreated
- Recreation process can have service impacts, especially for large Deployments
- A Deployment strategy defines how this rebuild process is done, to minimize downtime due to application failures or malfunctions

Types of Deployment Strategies

Kubernetes supports two basic strategies, but users can also leverage multiple Deployments when applying changes

- Strategies for single Deployments
 - Recreate
 - RollingUpdate
- Strategic approaches using two Deployments with a Service
 - Canary deployments
 - Blue/Green deployments



Each approach has a specific behavior and advantages/disadvantages.

Deployment Strategy: Recreate



INNOVATION
SOFTWARE

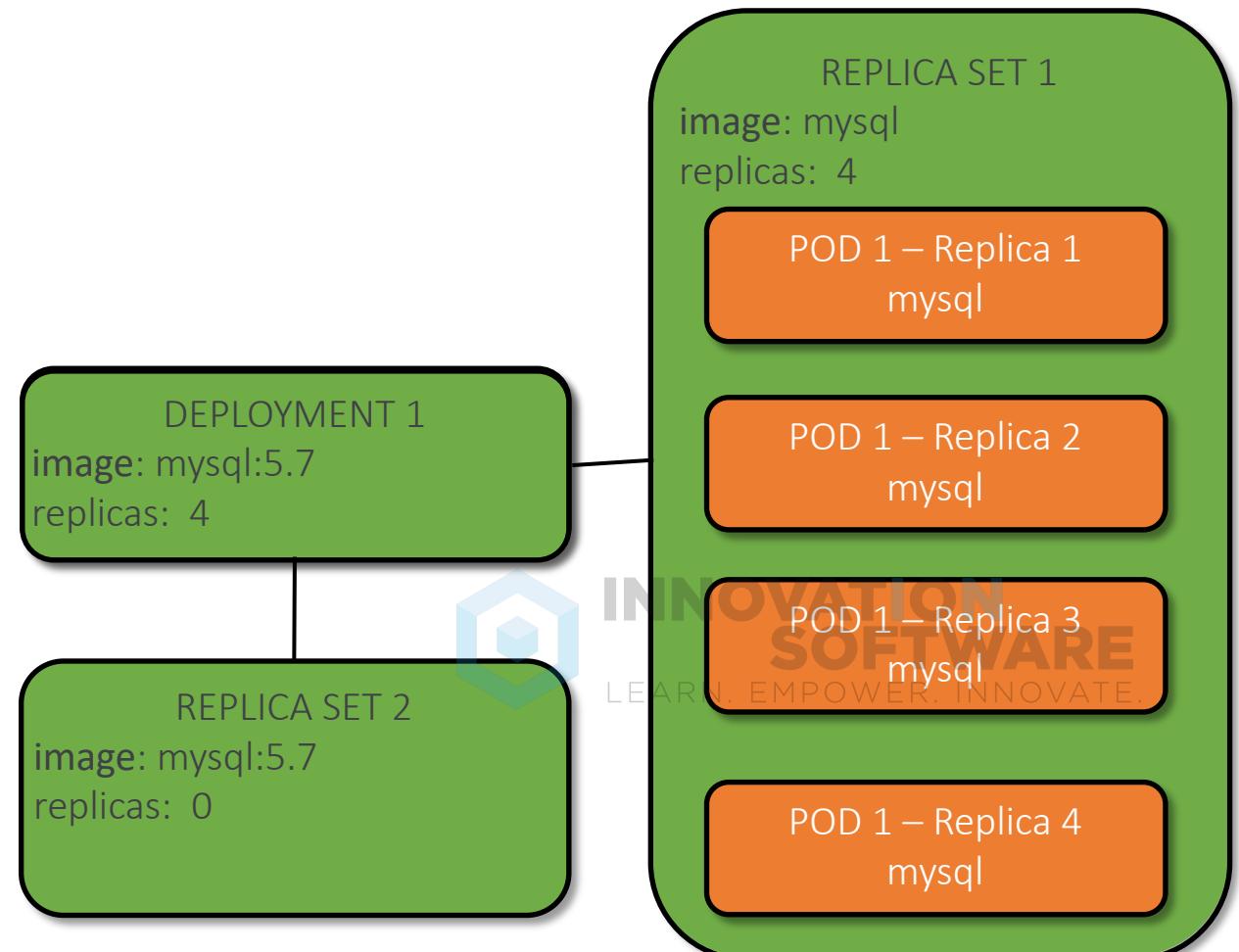
LEARN. EMPOWER. INNOVATE.



Deployment Strategy: Recreate

Simplest strategy for deployments

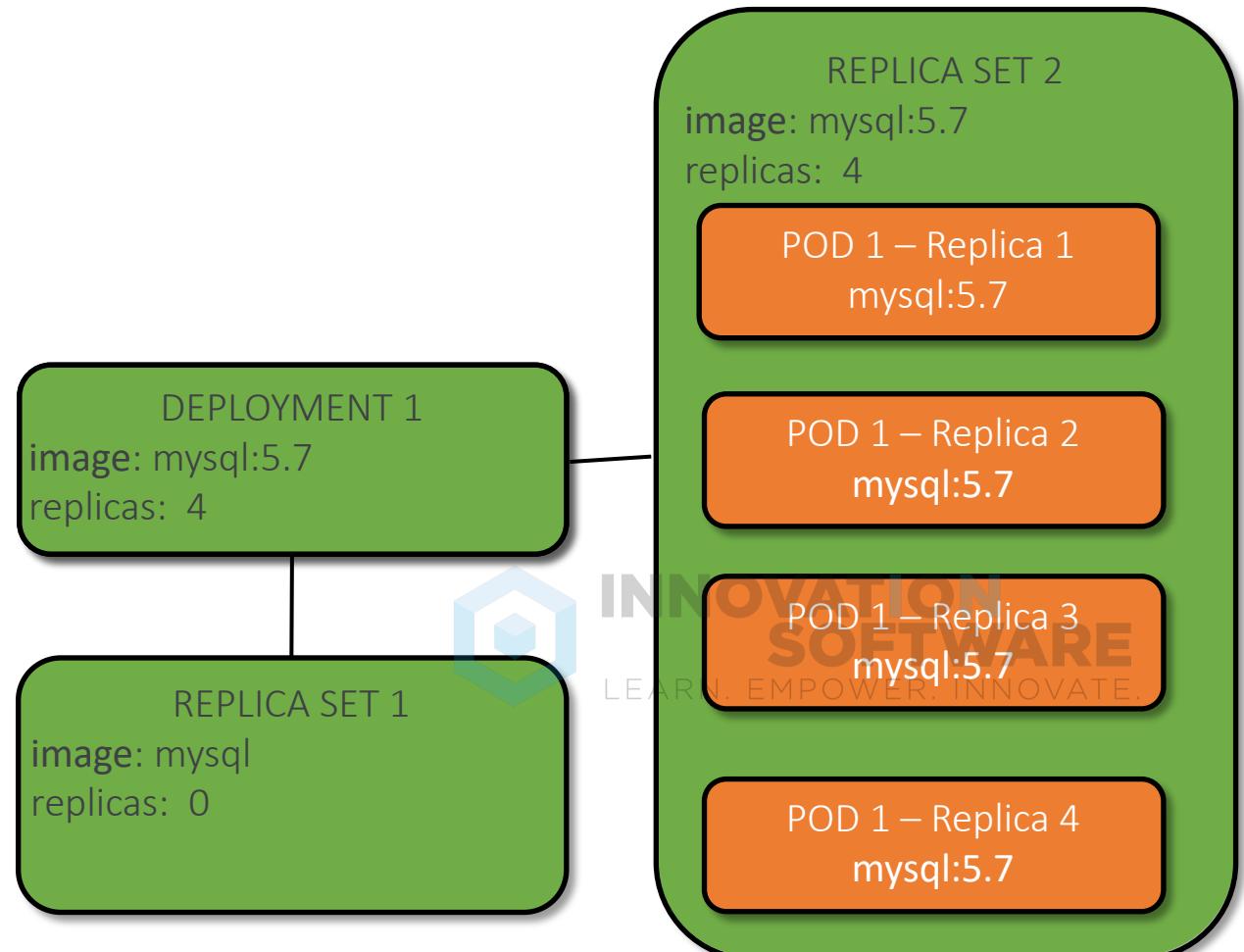
- When a change is made to a Deployment's spec, all Pods are removed and then recreated
 - Old Replica Set pods are killed
 - Then, new Replica Set starts pods
- May lead to downtime during the process while new pods are started



Deployment Strategy: Recreate

Simplest strategy for deployment updates

- When a change is made to a Deployment's template, all Pods are removed and then recreated
 - Old Replica Set pods are killed
 - New Replica Set starts pods
- May cause downtime due to delay between old pods terminating and new pods becoming available



Deployment Strategy: Recreate

Strategies are defined in the spec of a Deployment

- **strategy** parameter in Deployment spec sets the strategy to be used for updates
- If no parameter value is set, the default is **RollingUpdate**

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: tomcat-deployment
spec:
  replicas: 3
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        type: webserver
    spec:
      containers:
        - name: tomcat-container
          image: tomcat
          ports:
            - containerPort: 8080
```



**INNOVATION
SOFTWARE**
LEARN. EMPOWER. INNOVATE.

Deployment Strategy: RollingUpdate



INNOVATION
SOFTWARE

LEARN. EMPOWER. INNOVATE.



Deployment Strategy: Rolling Update

***RollingUpdate** is **DEFAULT** strategy for Deployments*

- When a change is made to the Deployment, the old Replica Set pods are scaled down as new pods are created by the new Replica Set
- A minimum number of running Pods is specified, so the Deployment will never be totally out of Pods to respond to service requests
- During the update process, the requested replica count may be temporarily exceeded

Deployment Strategy: RollingUpdate

Configure parameters to control the update process

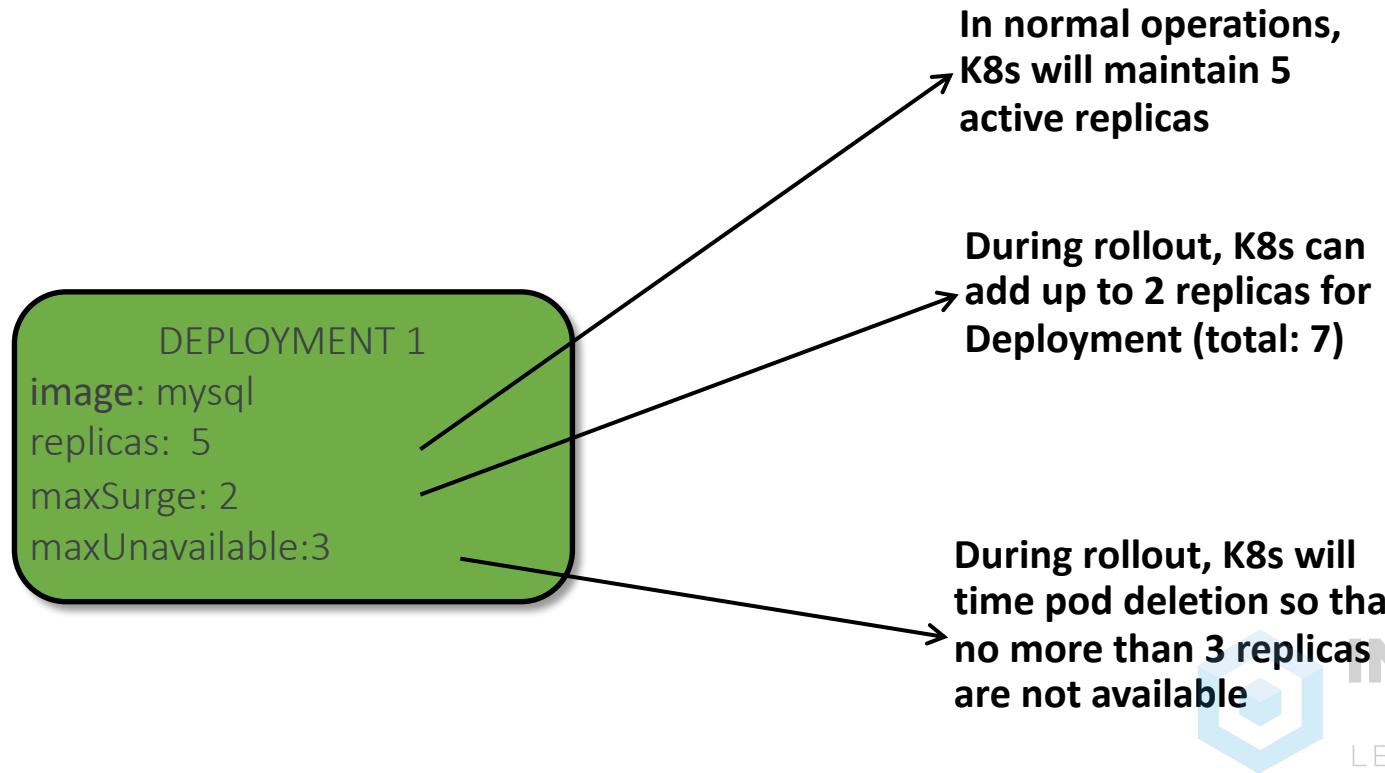
```
...  
  metadata:  
    name: tomcat-deployment  
  spec:  
    replicas: 3  
    strategy:  
      type: RollingUpdate  
      rollingUpdate:  
        maxSurge: 25%  
        maxUnavailable:10%  
    template:  
      metadata:  
        labels:  
          type: webserver  
...  
...
```

- **maxSurge**: number or percentage of additional Pods that can be created exceeding the replica count during update
 - Default value of 25%
- **maxUnavailable**: number of Pods that can be unavailable during the update
 - Default value of 25%



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Deployment Strategy: RollingUpdate

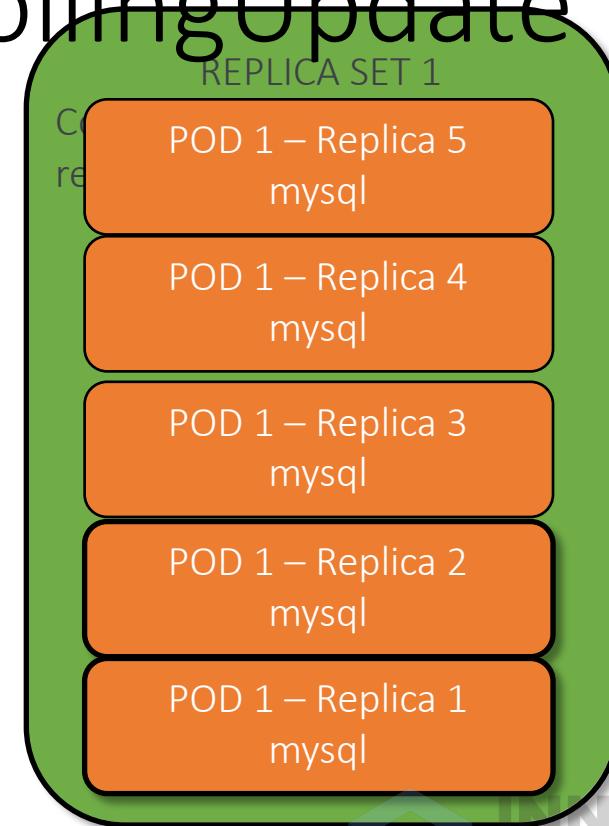


INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Deployment Strategy: RollingUpdate

Initial State

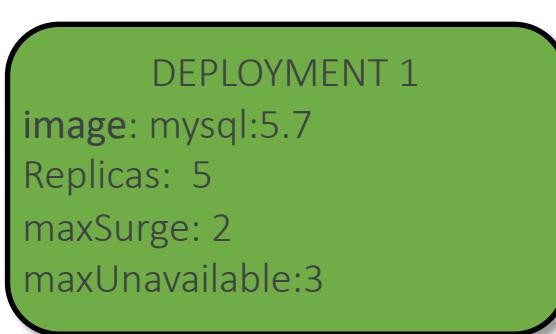
DEPLOYMENT 1
image: mysql
Replicas: 5
maxSurge: 2
maxUnavailable:3



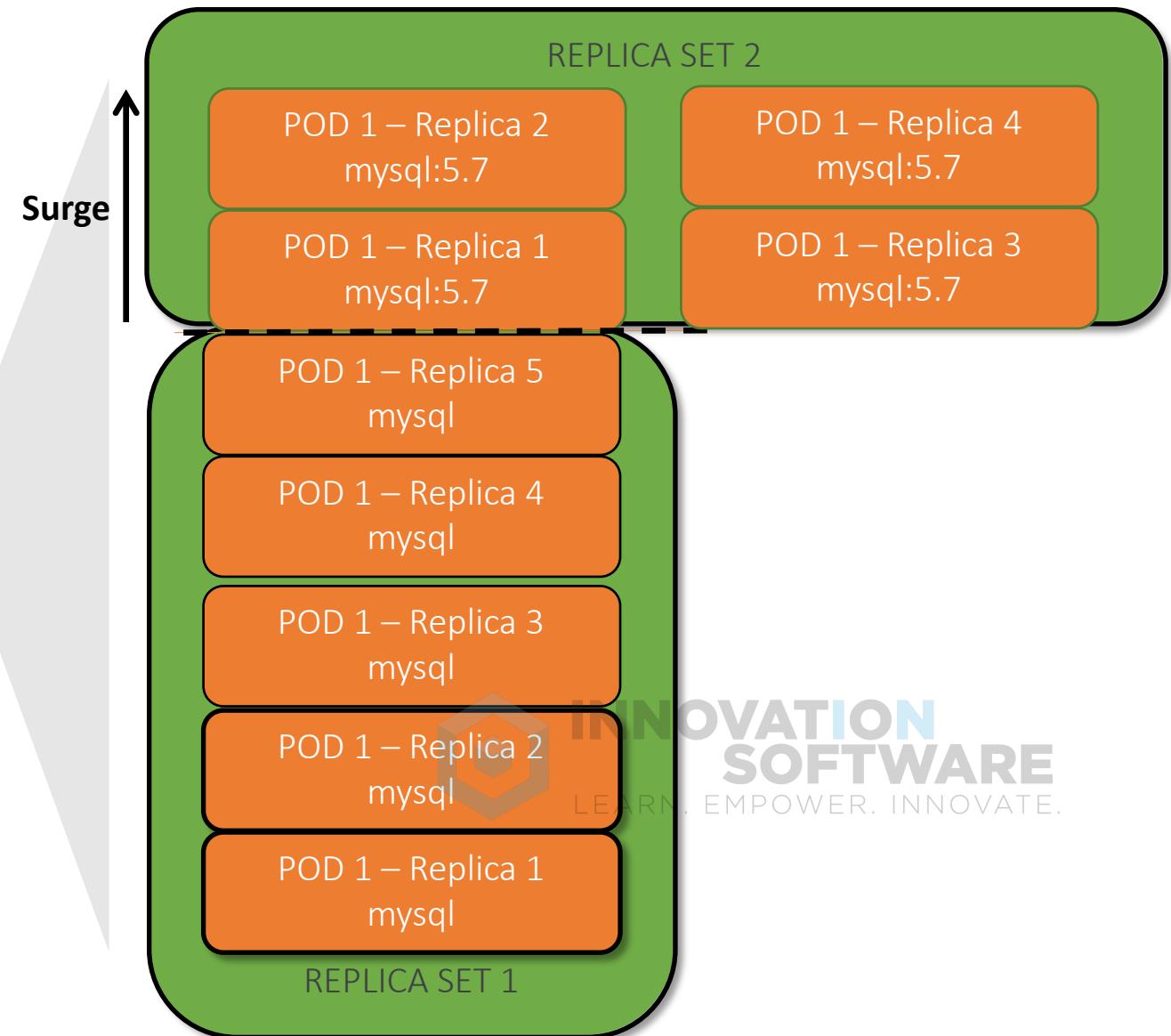
```
$ vi simpledeployment.yaml
...
  image: mysql:5.7
...
$ kubectl apply -f simpledeployment.yaml
```

Deployment Strategy: RollingUpdate

Rollout in progress

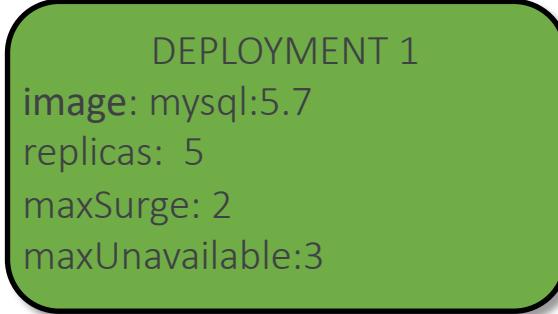


- Initial surge of new pods on new Replica Set
- Original Replica Set scaled back as new RS scaled out

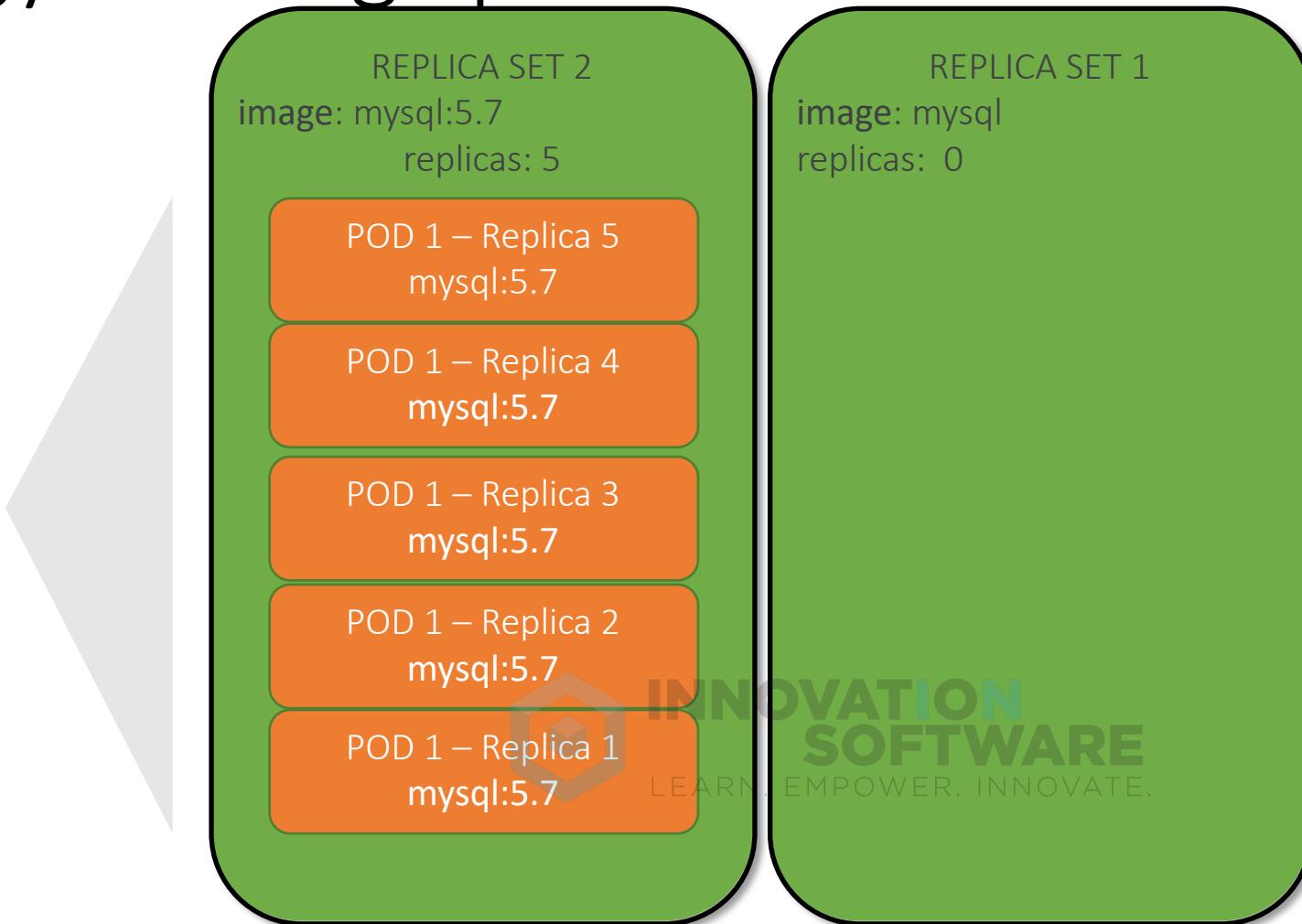


Deployment Strategy: RollingUpdate

Rollout complete



- By default, old, inactive Replica Set saved – previous version of the Deployment



Updating Using Multiple Deployments



INNOVATION
SOFTWARE

LEARN. EMPOWER. INNOVATE.



RollingUpdate using Multiple Deployments

Controlled testing of new versions in production

- Assume an application running as a Deployment, exposed as a Service
- To apply a new application version in production, a second Deployment can be used using labels in common with the first Deployment
 - Canary deployment allows for limited testing of new version in production
 - Blue/green deployment

Strategic Approach: Canary Deployment

Controlled testing of the update on production

- Consider a Service selecting pods from a Deployment of application pods
- In a canary deployment, a second Deployment (Canary Deployment) is created with pods for the new version, with labels matching the Service's selector
- Service directs some requests to pods on the Canary, allowing testing of changes in production
- If a malfunction is detected, it will only impact a small portion of the Pods and can be undone.

Strategic Approach: Canary Deployment

DEPLOYMENT 1
image: tomcat
replicas: 3
type=webserver
channel=production



POD 1 – Replica 1
tomcat

POD 1 – Replica 2
tomcat

POD 1 – Replica 3
tomcat

SERVICE
selector:
type=webserver



DEPLOYMENT 2
image: tomcat:8.5.5
replicas: 1
type=webserver
channel=canary

POD 1 – Replica 1
tomcat:8.5.5



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Strategic Approach: Canary Deployment

Decisions after running the canary Deployment in production

- If the application error rate is not increasing and Canary Deployment is stable:
 - The main Deployment can be updated to the newer version (using Rolling Update for example) and then the Canary can be discarded; OR
 - The Canary can be scaled up and reconfigured and the old Deployment can be discarded.
- If the test results in failure, the Canary deployment can be deleted

Strategic Approach: Blue/Green Deployment

Complete environment switch from one version to another

- With a Blue/Green deployment, you create a new full-scale Deployment in addition to the current production Deployment
- Reconfiguring the pod label selector on the application's Service allows choice of directing requests to old Deployment or new Deployment
- Similar to effect of Replace strategy without application downtime

Strategic Approach: Blue/Green Deployment

DEPLOYMENT 1

```
image: tomcat  
replicas: 3  
type=webserver  
color=blue
```

DEPLOYMENT 2

```
image: tomcat:8.5.5  
replicas: 3  
type=webserver  
color=green
```



SERVICE

```
selector:  
type: webserver  
color=blue
```



INNOVATION SOFTWARE
LEARN. EMPOWER. INNOVATE.

Strategic Approach: Blue/Green Deployment

DEPLOYMENT 1

image: tomcat
replicas: 3
type: webserver
color=blue

DEPLOYMENT 2

image: tomcat:8.5.5
replicas: 3
type: webserver
color=green



SERVICE

selector:
type: webserver
color=green



Kubernetes Services



INNOVATION
SOFTWARE
LEARN. LEADER. INNOVATE.

Services

- Persistent way to access logical set of PODs
- Set of PODs (usually) determined by a Label Selector
- Kubernetes-native apps: Kubernetes offers a simple Endpoints API that is updated whenever the set of Pods in a Service changes.
- Non-native apps: Kubernetes offers a virtual-IP-based bridge to Services which redirects to the backend Pods.

Service Definition

```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
```



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Types of Services

- ClusterIP
 - NodePort
 - LoadBalancer
 - Headless
-
- Ingress is not a service

ClusterIP

- Virtual IP that every node can route to

NodePort

- Same port open on every node (default range: 30000-32767)



LoadBalancer

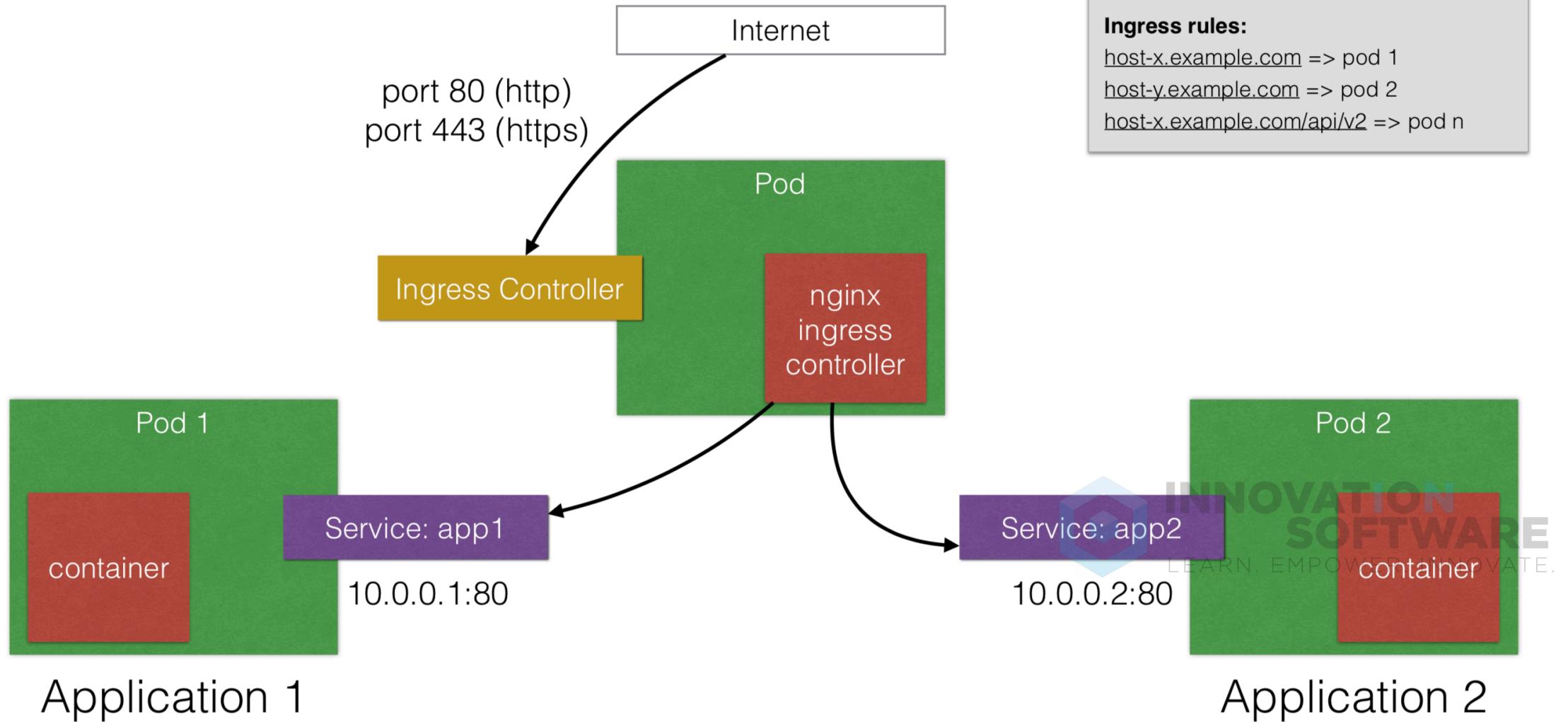
- External L4 network load balancer
- Requires cloud / IaaS coordination



Ingress

- L7 (instead of L4)
- HTTP (instead of tcp/udp)
- Ingress can route paths differently:
 - /api/app1 -> app1 pods
 - /api/app2 -> app2 pods
- Requires ingress provider (might be cloud, might be in-cluster)

Ingress



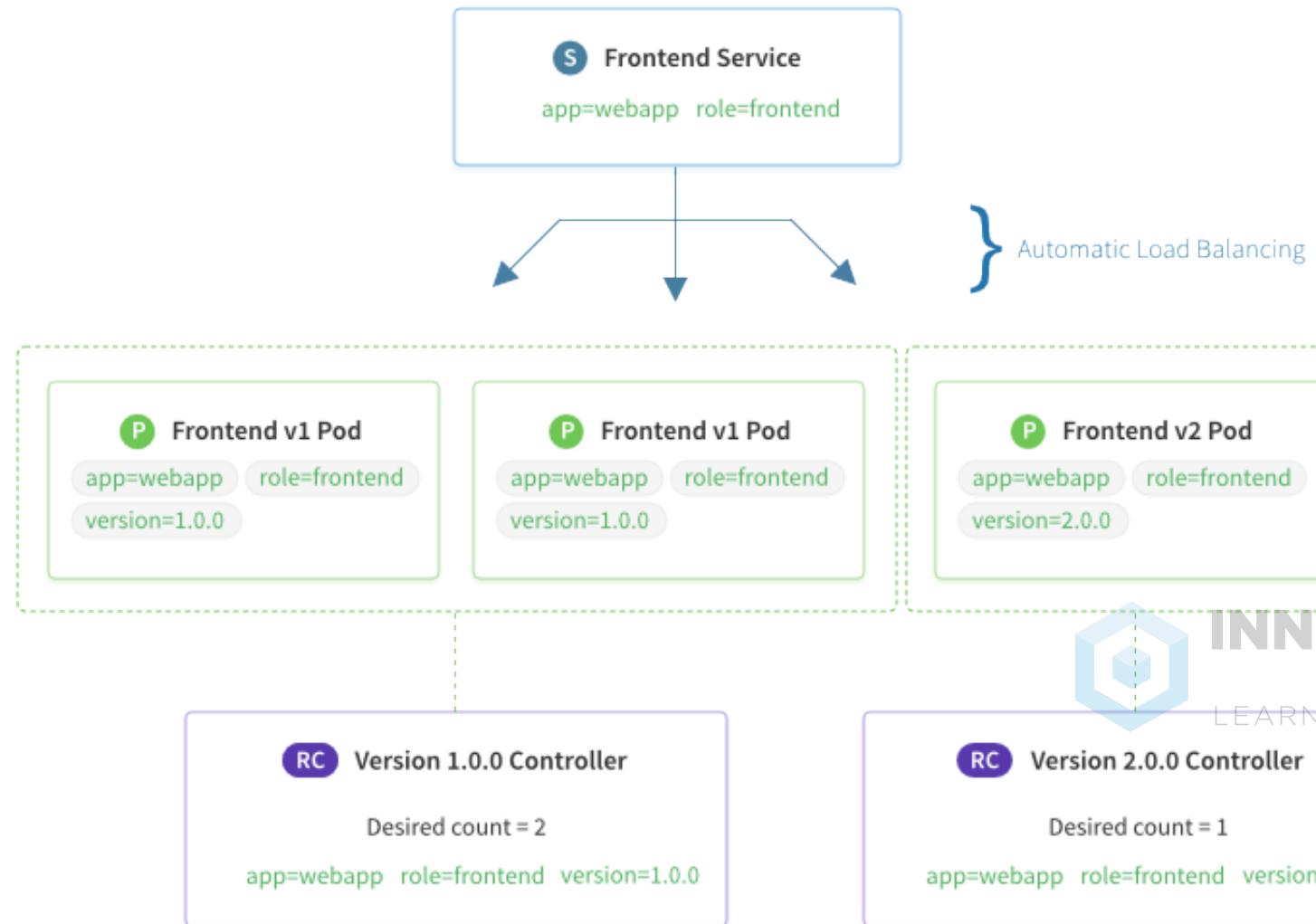
Ingress Definition

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: helloworld-rules
spec:
  rules:
    - host: helloworld-v1.example.com
      http:
        paths:
          - path: /
            backend:
              serviceName: helloworld-v1
              servicePort: 80
    - host: helloworld-v2.example.com
      http:
        paths:
          - path: /
            backend:
              serviceName: helloworld-v2
              servicePort: 80
```



**INNOVATION
SOFTWARE**
LEARN. EMPOWER. INNOVATE.

Services



**INNOVATION
SOFTWARE**

LEARN. EMPOWER. INNOVATE.

Services

- Services can map incoming port to any targetPort
- By default the targetPort will be set to the same value as the port field.
- NOTE: targetPort can be a string, referring to the name of a port in the backend Pods.
- Services support TCP and UDP for protocols. The default is TCP.



Services mapping string

Deployment Specification

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 8080
              name: http-web
              protocol: TCP
```



LEARN. EMPOWER. INNOVATE.

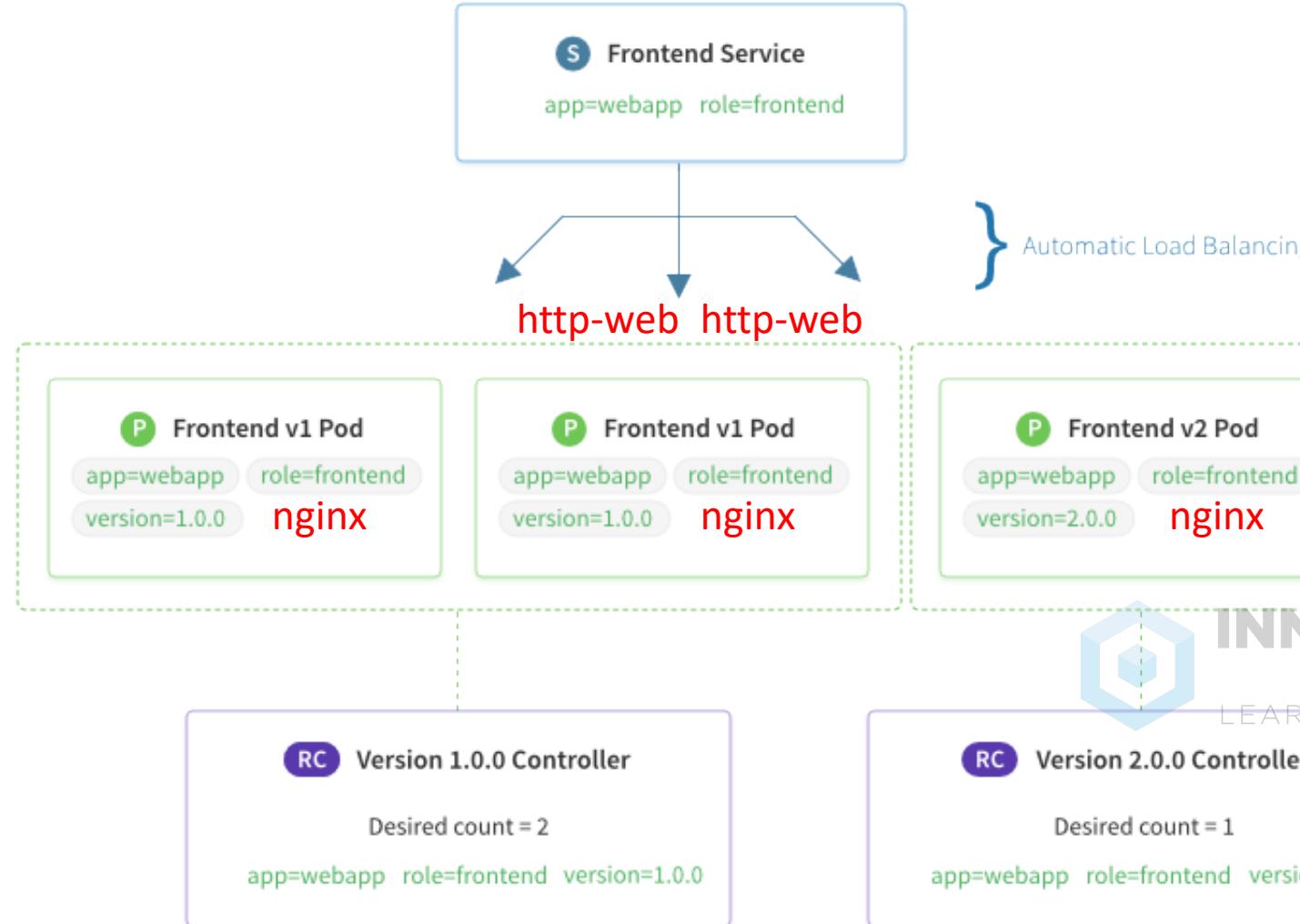
Services targetPort mapped to string

Service specification

```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    name: http-web
    port: 80
    protocol: TCP
    targetPort: http-web
```



Services diagram mapping string



INNOVATION
SOFTWARE

LEARN. EMPOWER. INNOVATE.

Services without selectors

- Why would you want a Service that's not mapped to a selector?
- Use an external database cluster in production, but in test you use your own databases.
- Point your service to a service in another namespace or on another cluster.
- Migrating your workload to Kubernetes and some of your backends run outside of Kubernetes
- More flexibility



Kubernetes DNS



INNOVATION
SOFTWARE

LEARN. EMPOWER. INNOVATE.



Kubernetes DNS

- Kubernetes Schedules a DNS POD, which sits in “Pending” until a network is created.
- Objects that have DNS names:
 - Everything
- DNS enables PODs in different namespaces to communicate easily

Kubernetes DNS

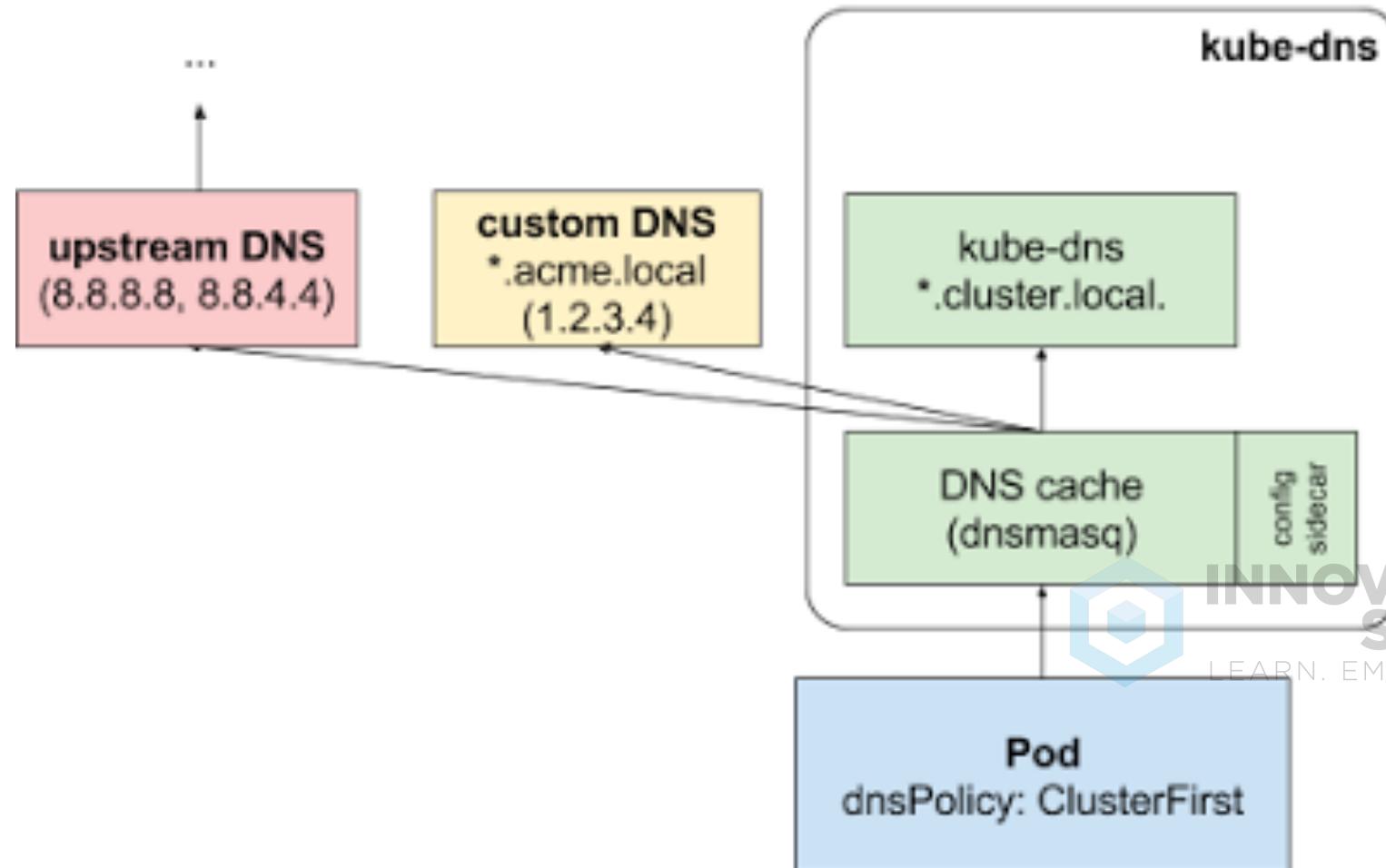
```
apiVersion: v1
kind: Service
metadata:
  name: default-subdomain
spec:
  selector:
    app: MyApp
ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 9376
  - name: https
    protocol: TCP
    port: 443
    targetPort: 9377
```

```
apiVersion: v1
kind: Pod
metadata:
  name: busybox1
  labels:
    name: busybox
spec:
  hostname: busybox-1
  subdomain: default-subdomain
  containers:
    - image: busybox
      command:
        - sleep
        - "3600"
      name: busybox
```



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE

Kubernetes DNS



Kubernetes DNS

- Use a ConfigMap to specify:
 - custom stub domains
 - upstream nameservers

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kube-dns
  namespace: kube-system
data:
  stubDomains: |
    {"acme.local": ["1.2.3.4"]}
  upstreamNameservers: |
    ["8.8.8.8", "8.8.4.4"]
```



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Debugging DNS

- Simple POD to use for testing
 - YAML file to deploy busybox

```
apiVersion: v1
kind: Pod
metadata:
  name: busybox
  namespace: default
spec:
  containers:
    - name: busybox
      image: busybox
      command:
        - sleep
        - "3600"
      imagePullPolicy: IfNotPresent
      restartPolicy: Always
```



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Debugging DNS

- Simple POD to use for testing
 - YAML file to deploy busybox

```
kubectl exec -ti busybox – nslookup kubernetes.default  
Server: 10.96.0.10  
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local  
Name: kubernetes.default  
Address 1: 10.96.0.1 kubernetes.default.svc.cluster.local
```

Debugging DNS

- If nslookup fails check local DNS config first!
- Check the resolv.conf

```
kubectl exec busybox cat /etc/resolv.conf  
nameserver 10.96.0.10  
search default.svc.cluster.local svc.cluster.local cluster.local ec2.internaloptions  
ndots:5
```

Debugging DNS

- Problems with kube-dns add-on or associated Services:

```
kubectl exec -ti busybox - nslookup kubernetes.default
```

```
Server: 10.0.0.10
```

```
Address 1: 10.0.0.10
```

```
nslookup: can't resolve 'kubernetes.default'
```

Debugging DNS

- Check to make sure DNS pod is running
- Check for errors in kube-dns PODs logs

```
kubectl get pods -n kube-system -l k8s-app=kube-dns
```

NAME	READY	STATUS	RESTARTS	AGE
kube-dns-6f4fd4bdf-pr659	3/3	Running	0	6d

Service Discovery: Env Var

Service Discovery also possible via environment variable:

```
> kubectl run -it --restart=Never --rm busybox --image=busybox env  
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin  
HOSTNAME=busybox  
TERM=xterm  
SIMPLE_API_PORT_80_TCP_PROTO=tcp  
NGINX_DEPLOYMENT_SERVICE_HOST=10.55.246.243  
NGINX_DEPLOYMENT_PORT_80_TCP_PORT=80  
NGINX_SERVICE_PORT=80  
SIMPLE_API_PORT=tcp://10.55.241.109:80  
KUBERNETES_PORT_443_TCP=tcp://10.55.240.1:443  
NGINX_SERVICE_HOST=10.55.255.228  
KUBERNETES_SERVICE_HOST=10.55.240.1  
SIMPLE_API_PORT_80_TCP_PORT=80  
NGINX_PORT=tcp://10.55.255.228:80  
SIMPLE_API_SERVICE_PORT=80  
NGINX_DEPLOYMENT_SERVICE_PORT=80  
NGINX_DEPLOYMENT_PORT_80_TCP_PROTO=tcp  
KUBERNETES_SERVICE_PORT=443  
KUBERNETES_PORT_443_TCP_PROTO=tcp  
NGINX_PORT_80_TCP=tcp://10.55.255.228:80  
SIMPLE_API_PORT_80_TCP=tcp://10.55.241.109:80  
KUBERNETES_PORT_443_TCP_PORT=443  
SIMPLE_API_SERVICE_HOST=10.55.241.109
```



Helm: Package management



INNOVATION
SOFTWARE

LEARN. EMPOWER. INNOVATE



Helm sample application

The screenshot shows a web browser window titled "Guestbook" at the URL "dev.frontend.minikube.local/". The page displays a guestbook application with the title "Guestbook : "MyPopRock Festival 2.0"" and the Globomantics logo.

The main content area shows a table of messages:

Name	Message
John	Very nice show !!

Below the table, there is a "Filter" input field and pagination controls: "Items per page: 5" and "1 - 1 of 1".

At the bottom of the page, there is a form titled "Leave your feedback!" with fields for "Your name *" (Jane) and "Your questbook message" (Congrats to Globomantics DevOps!). A "Leave message" button is also present.

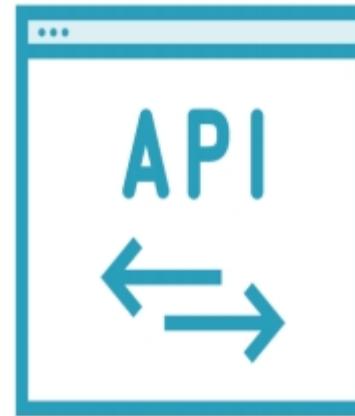
INNOVATION SOFTWARE
LEARN. EMPOWER. INNOVATE.

Sample application components



Frontend

- ✓ ConfigMap
- ✓ Pod
- ✓ Service
- ✓ Ingress



Backend API

- ✓ Secret
- ✓ Pod
- ✓ Service



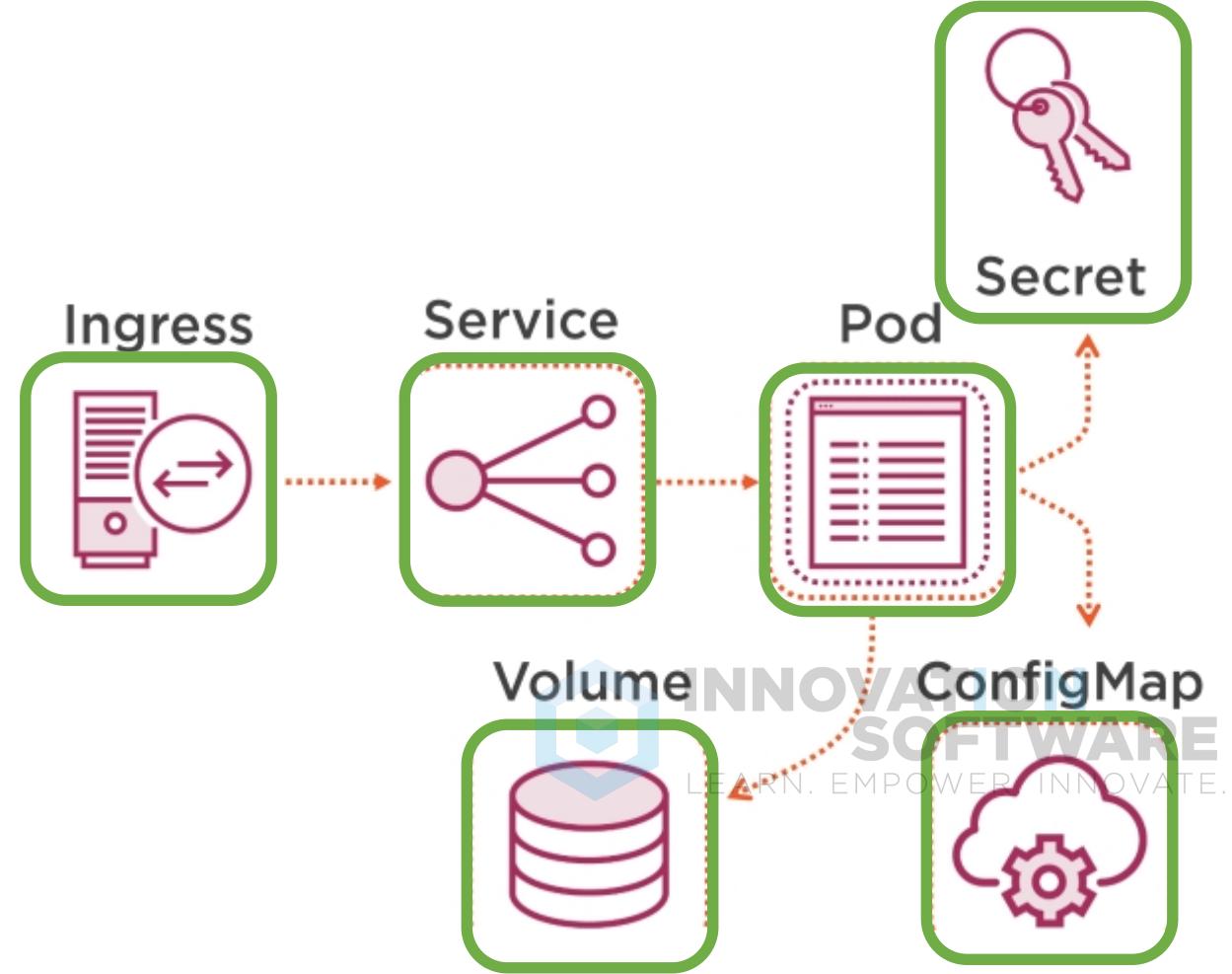
Database

- ✓ Secret
- ✓ PV
- ✓ PVC
- ✓ Pod
- ✓ Service



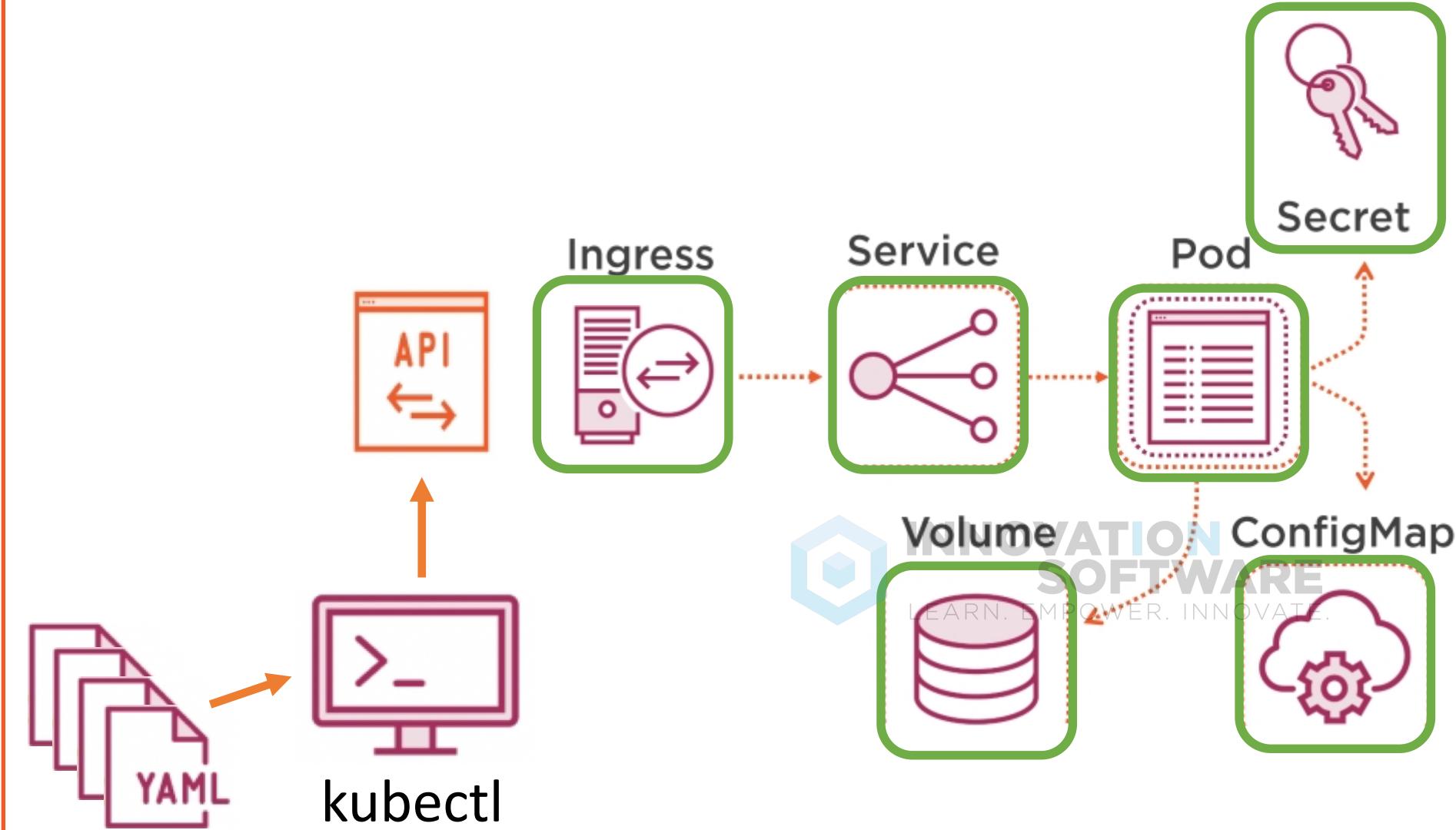
Native Kubernetes way

- Application
- Container
- Pod
- Service
- Ingress
- ConfigMap
- Secrets
- Volumes: PV, PVC, Storage



Native Kubernetes way

- Limitations
 - Packaging
 - Versioning



Guestbook: version 1



- ConfigMap
- Pod
- Service
- Ingress



Frontend



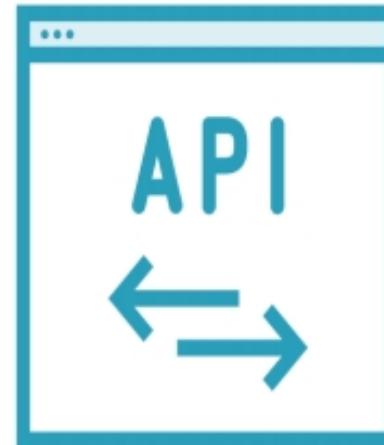
Guestbook: version 2



- ConfigMap
- Pod
- Service
- Ingress



Frontend



Backend



- Secret
- Pod
- Service



- Secret
- PV
- PVC
- Pod
- Service

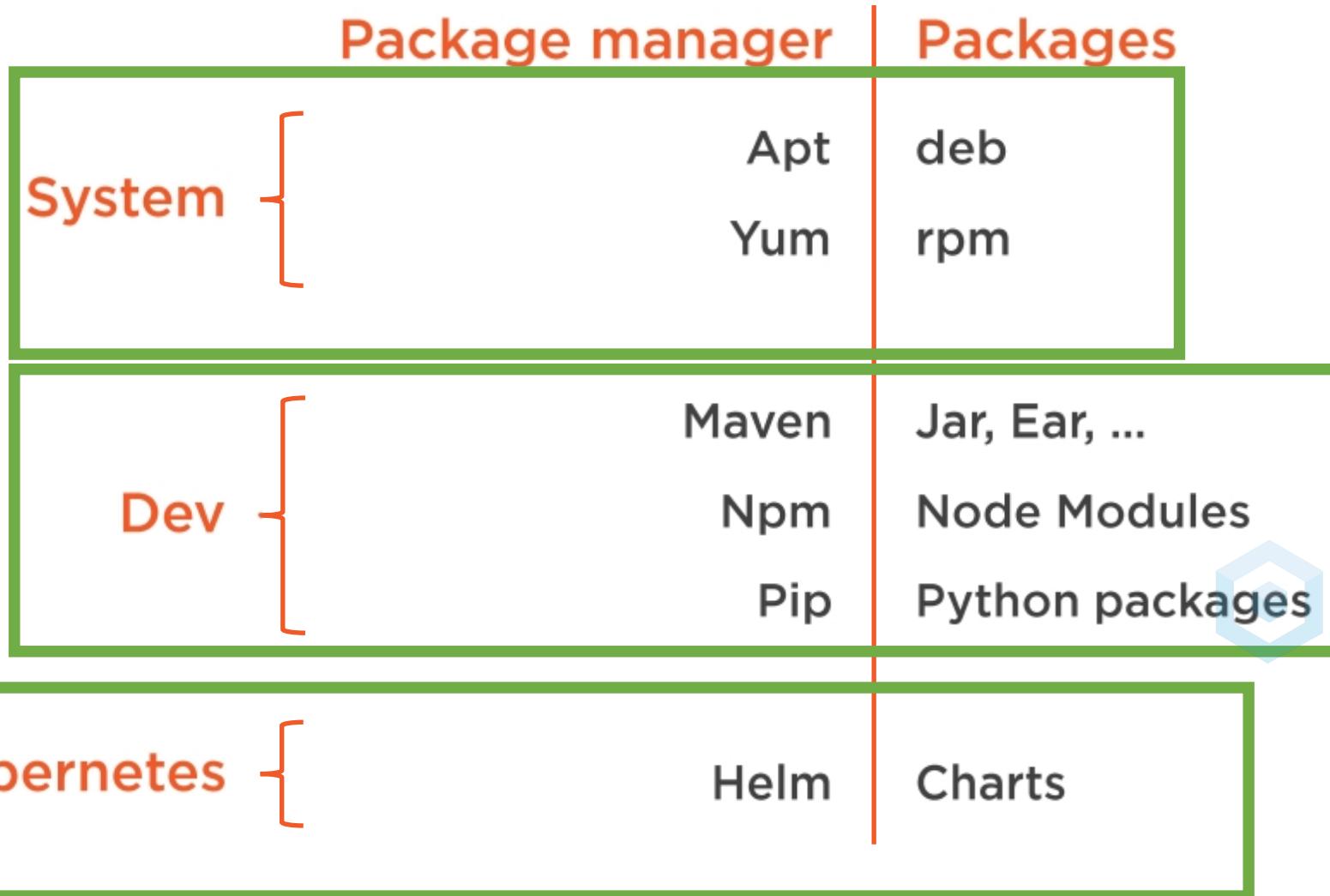
Painful to manage



**INNOVATION
SOFTWARE**

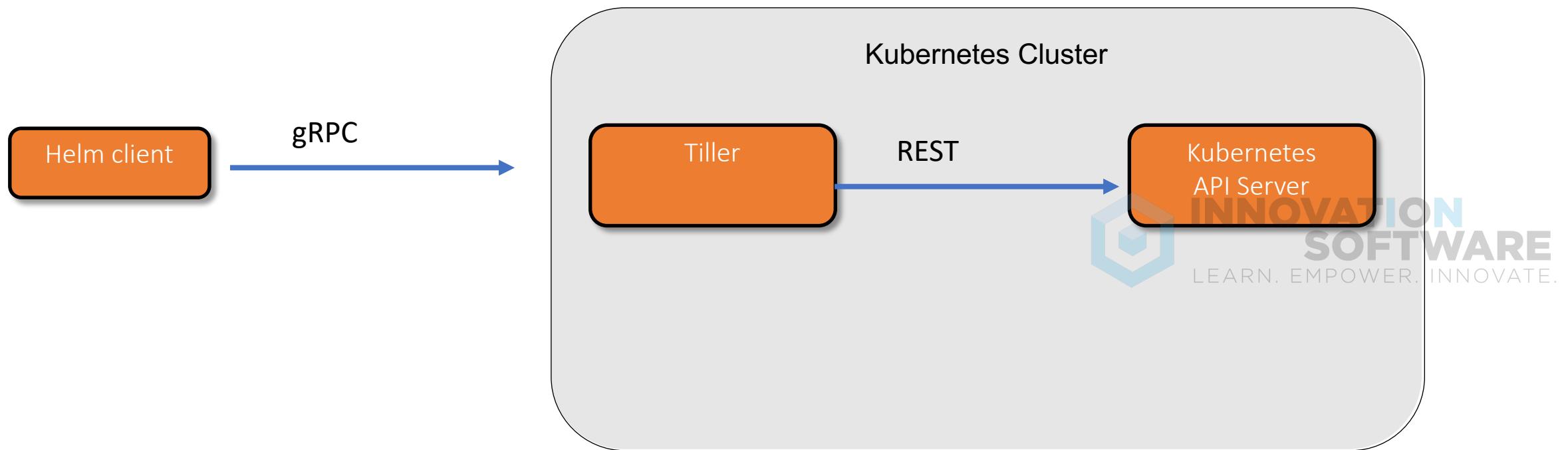
LEARN. EMPOWER. INNOVATE.

Helm features



Helm architecture

- Helm client
 - command-line
 - Interacts with Tiller
 - Local chart development
- Tiller
 - In-cluster
 - Listens to Helm client
 - Interacts with Kube API
 - Manages lifecycle



Helm features



Charts



Templates



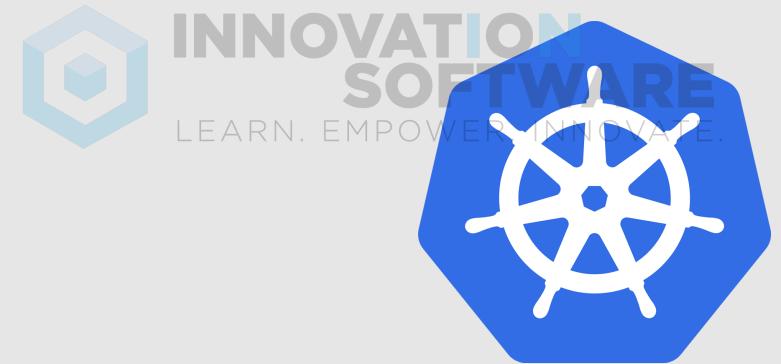
Dependencies



INNOVATION
REPOSITORIES
SOFTWARE
LEARN. EMPOWER. INNOVATE.



Helm Chart Structure



Helm Chart structure



- nginx-demo
 - Chart.yaml
- Chart properties
 - name
 - version
 - more..

Helm Chart structure



- **nginx-demo**
 - Chart.yaml
 - README.md
- **Document chart**
 - Overrides
 - Maintainer
 - Instructions

Helm Chart structure



- nginx-demo
 - Chart.yaml
 - README.md
- templates
 - deployment.yaml
 - ingress.yaml
 - service.yaml
- Kubernetes object definitions
 - customizable YAML templates

Helm Chart structure



- nginx-demo
 - Chart.yaml
 - README.md
- templates
 - deployment.yaml
 - ingress.yaml
 - service.yaml
- values.yaml
- Kubernetes object definitions
 - customizable YAML templates
 - Provide default values.

Helm Chart structure



- nginx-demo
 - Chart.yaml
 - README.md
- templates
 - deployment.yaml
 - ingress.yaml
 - service.yaml
 - NOTES.txt
- values.yaml
- Provide helpful output after installation
 - How to access application
 - Create user/pass

Helm Chart structure



- nginx-demo
 - Chart.yaml
 - README.md
 - templates
 - deployment.yaml
 - ingress.yaml
 - service.yaml
 - NOTES.txt
 - tests
 - test-connection.yaml
 - values.yaml
- You can create tests to confirm Chart works as expected.

Helm Chart structure

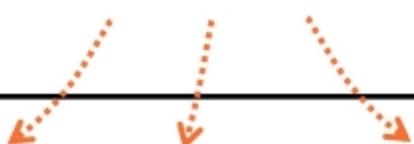


- nginx-demo
 - Chart.yaml
 - README.md
 - requirements.yaml
 - templates
 - deployment.yaml
 - ingress.yaml
 - service.yaml
 - NOTES.txt
 - tests
 - test-connection.yaml
 - values.yaml
- Define sub-charts and dependencies

Helm Chart.yaml

Chart.yaml

```
apiVersion: v1
appVersion: "1.0"
description: A Helm chart for Kubernetes
name: nginx-demo
version: 0.1.0
```



Major, Minor, Patch (SemVer 2.0)



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

- App you are installing

Helm Chart.yaml

Chart.yaml

```
apiVersion: v1
appVersion: "1.0"
description: A Helm chart for Kubernetes
name: nginx-demo
version: 0.1.0
```



Major, Minor, Patch (SemVer 2.0)



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

- Version of Helm chart

Common Helm commands

Action	Command
Install a Release	<code>helm install [chart]</code>
Upgrade a Release revision	<code>helm upgrade [release] [chart]</code>
Rollback to a Release revision	<code>helm rollback [release] [revision]</code>
Print Release history	<code>helm history [release]</code>
Display Release status	<code>helm status [release]</code>
Show details of a release	<code>helm get [release]</code>
Uninstall a Release	<code>helm delete [release]</code>
List Releases	<code>helm list</code>

Deploy applications with Helm

- Install Helm
- Create Helm chart
- Deploy application
- Modify Chart
- Upgrade application
- Rollback application



Liveness & Readiness Probes

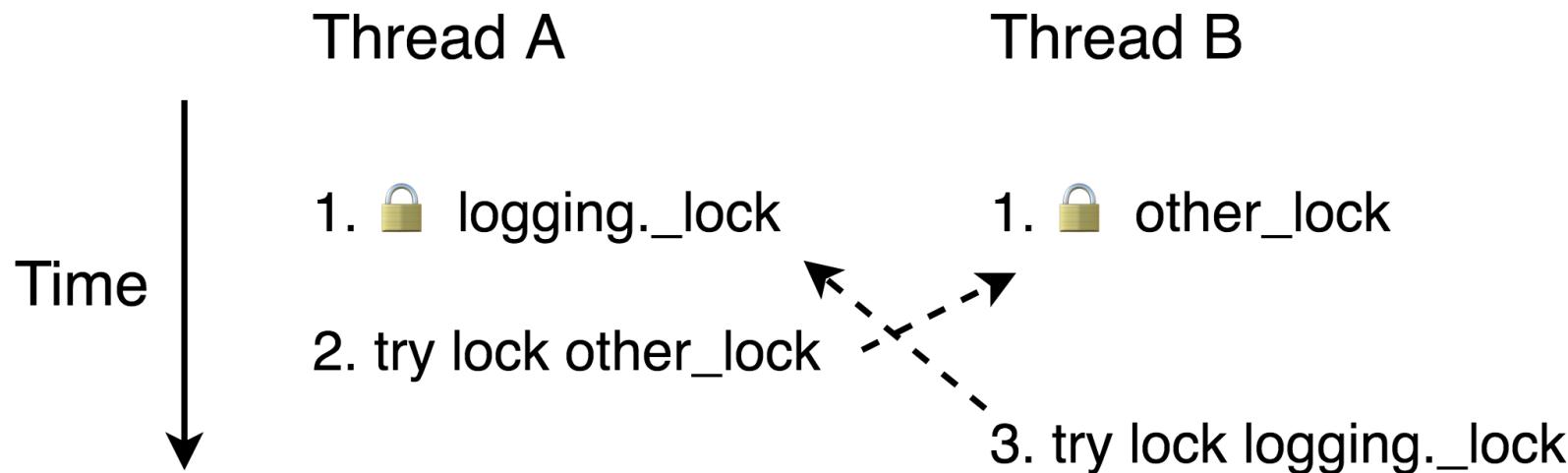


INNOVATION
SOFTWARE

LEARN. EMPOWER. INNOVATE.



Health Checks: deadlock



Health Checks

- Simple process health checks are enabled by default.
- Kubelet checks to see if container is responding and if not recreates it.

Go deadlock example code

```
lockOne := sync.Mutex{}  
lockTwo := sync.Mutex{}  
go func() {  
    lockOne.Lock();  
    lockTwo.Lock();  
    ...  
}()  
lockTwo.Lock();  
lockOne.Lock();
```

Health Checks

Kubernetes provides 3 types of application health checks

- HTTP: Kubelet calls a web hook and looks for return status between 200 and 399 for success.
- Container Exec: Kubelet will execute a command inside your container, exit status=0 is a success
- TCP Socket: Kubelet will attempt to open a socket to your container. If it establishes a connection, the container is considered healthy

In all cases, if the Kubelet discovers a failure the container is restarted.

Health Checks

Kubernetes refers to its application health checks as:

- Liveness Probe: Check to confirm application is running as expected
- Readiness Probe: Check to confirm application has completed startup configuration

Health Checks

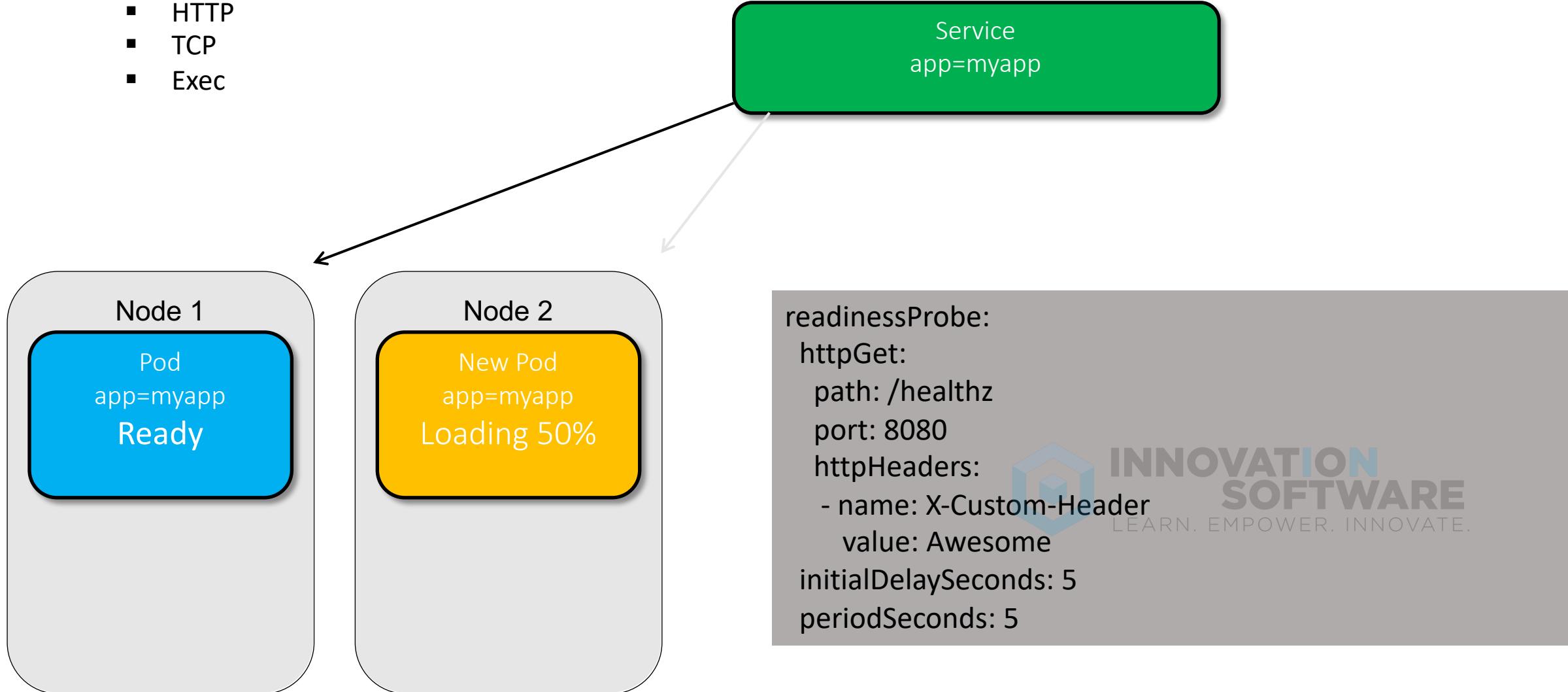
- Execute command to check if application is responding
- Set delay in seconds
 - How long before performing first check
- Set period in seconds.
 - How often it should run

livenessProbe example

```
livenessProbe:  
  exec:  
    command:  
      - cat  
      - /tmp/healthy  
  initialDelaySeconds: 5  
  periodSeconds: 5
```

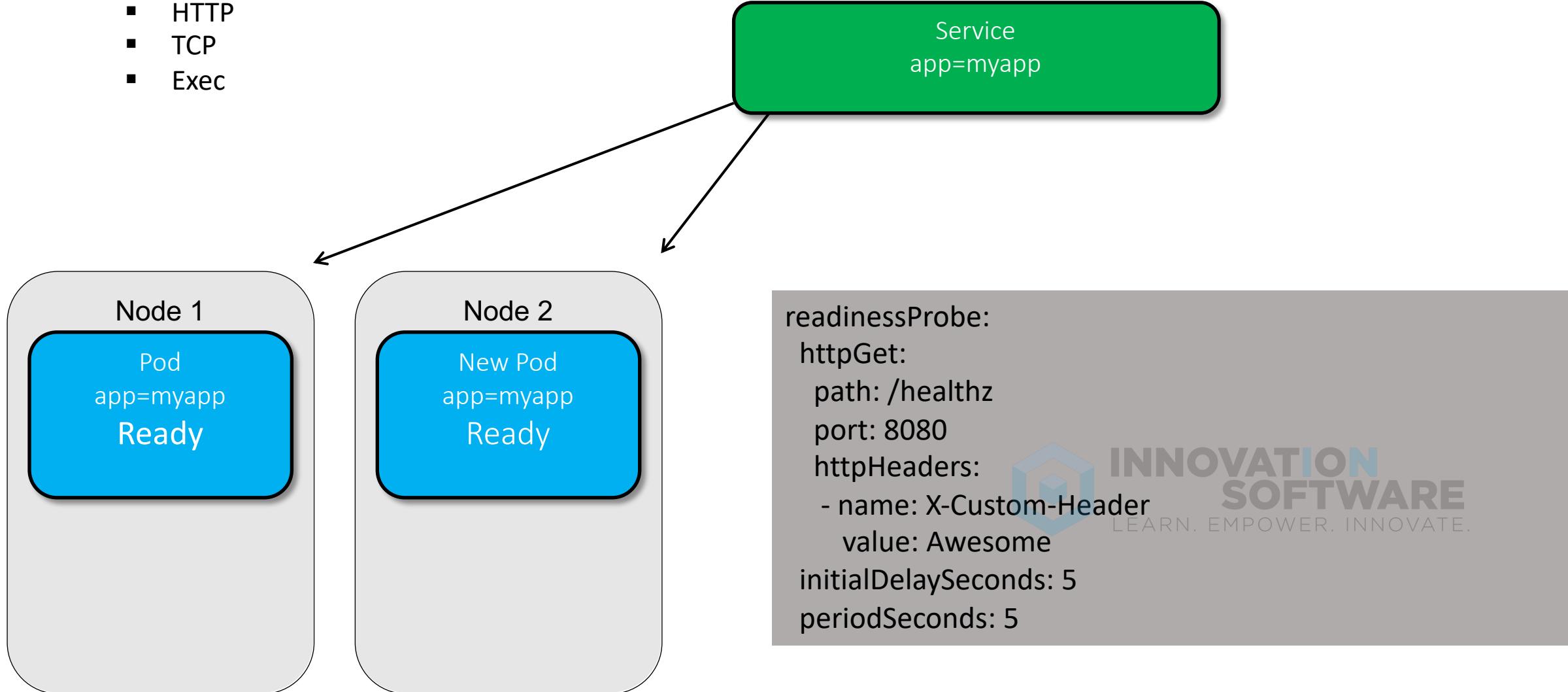
Health Checks: livenessProbe

- Do not send traffic to Pod until passes.
 - HTTP
 - TCP
 - Exec



Health Checks: livenessProbe

- Do not send traffic to Pod until passes.
 - HTTP
 - TCP
 - Exec



Health Checks

- Same as livenessProbe but traffic won't be sent to it if probe fails.

readinessProbe example

```
readinessProbe:  
  httpGet:  
    path: /healthz  
    port: 8080  
  httpHeaders:  
    - name: X-Custom-Header  
      value: Awesome  
  initialDelaySeconds: 5  
  periodSeconds: 5
```



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Health Check Lab

- Create POD with liveness check
- Verify check is working as expected
- Create POD with HTTP liveness check
- Confirm HTTP liveness probes are working correctly
- Create deployment with liveness and readiness probe
- Confirm checks are working

Role Based Access Controls



INNOVATION
SOFTWARE

LEARN. EMPOWER. INNOVATE.



RBAC

- Considered stable in Kubernetes 1.8 and later
- To enable start apiserver with
 - `--authorization-mode=RBAC`
 - This allows for more fine-grained control of access permissions

RBAC Roles

- Role used to grant access to resources in a single namespace

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: pod-reader
rules:
  - apiGroups: [""] # "" indicates the core API group
    resources: ["pods"]
    verbs: ["get", "watch", "list"]
```



**INNOVATION
SOFTWARE**
LEARN. EMPOWER. INNOVATE.

RBAC ClusterRole

- ClusterRole used to grant access to resources that are cluster specific, not namespace specific

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  # "namespace" omitted since ClusterRoles are not namespaced
  name: secret-reader
rules:
- apiGroups: []
  resources: ["secrets"]
  verbs: ["get", "watch", "list"]
```



RBAC RoleBinding

- Grant permissions defined in a role to user or set of users

```
# This role binding allows "jane" to read pods in the "default" namespace.
```

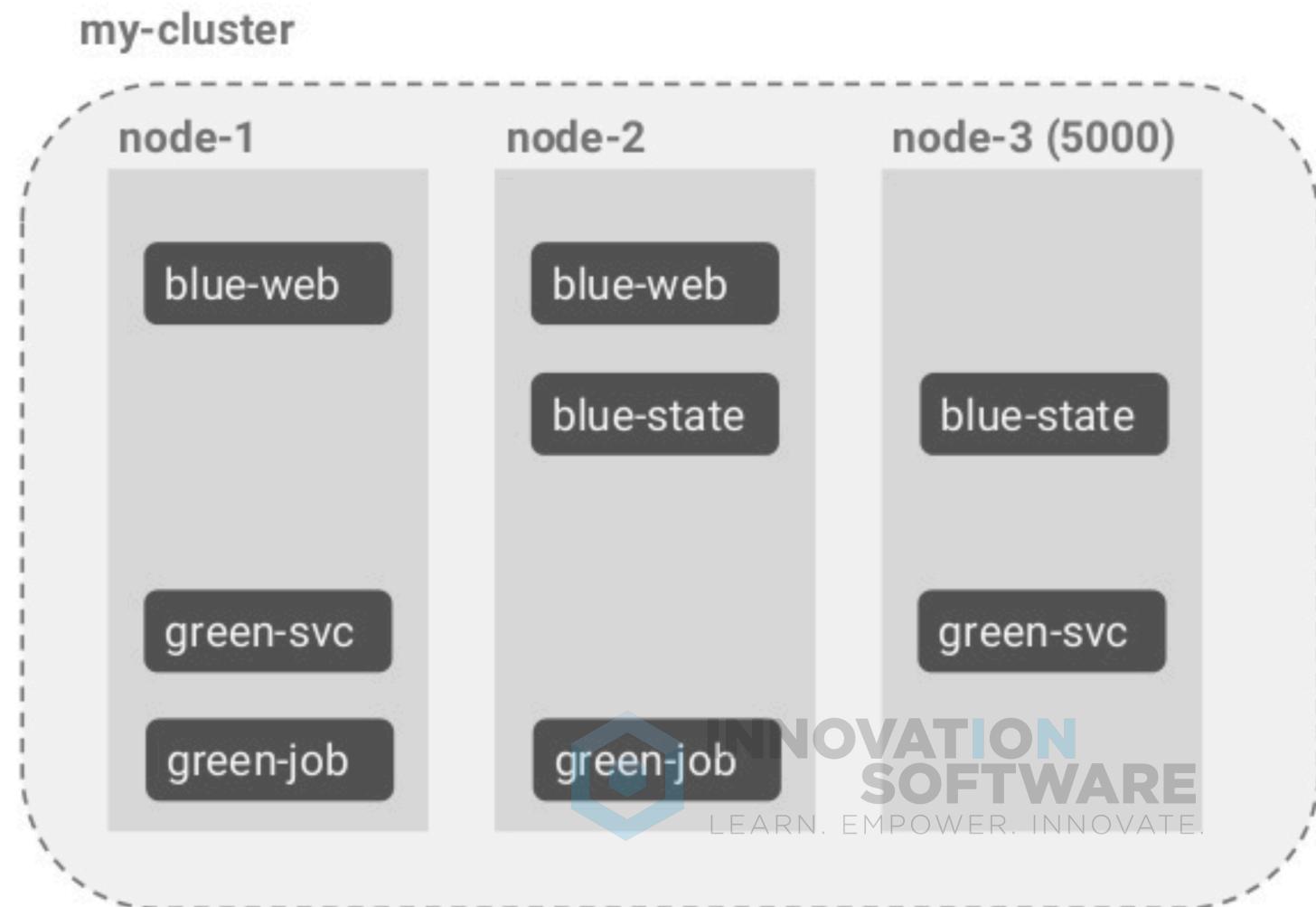
```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: jane
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

Without fine-grained Access:

- Authorization at cluster level
- All pods have same authorization

Without controlled scheduling:

- Lack flexibility for multi-workload

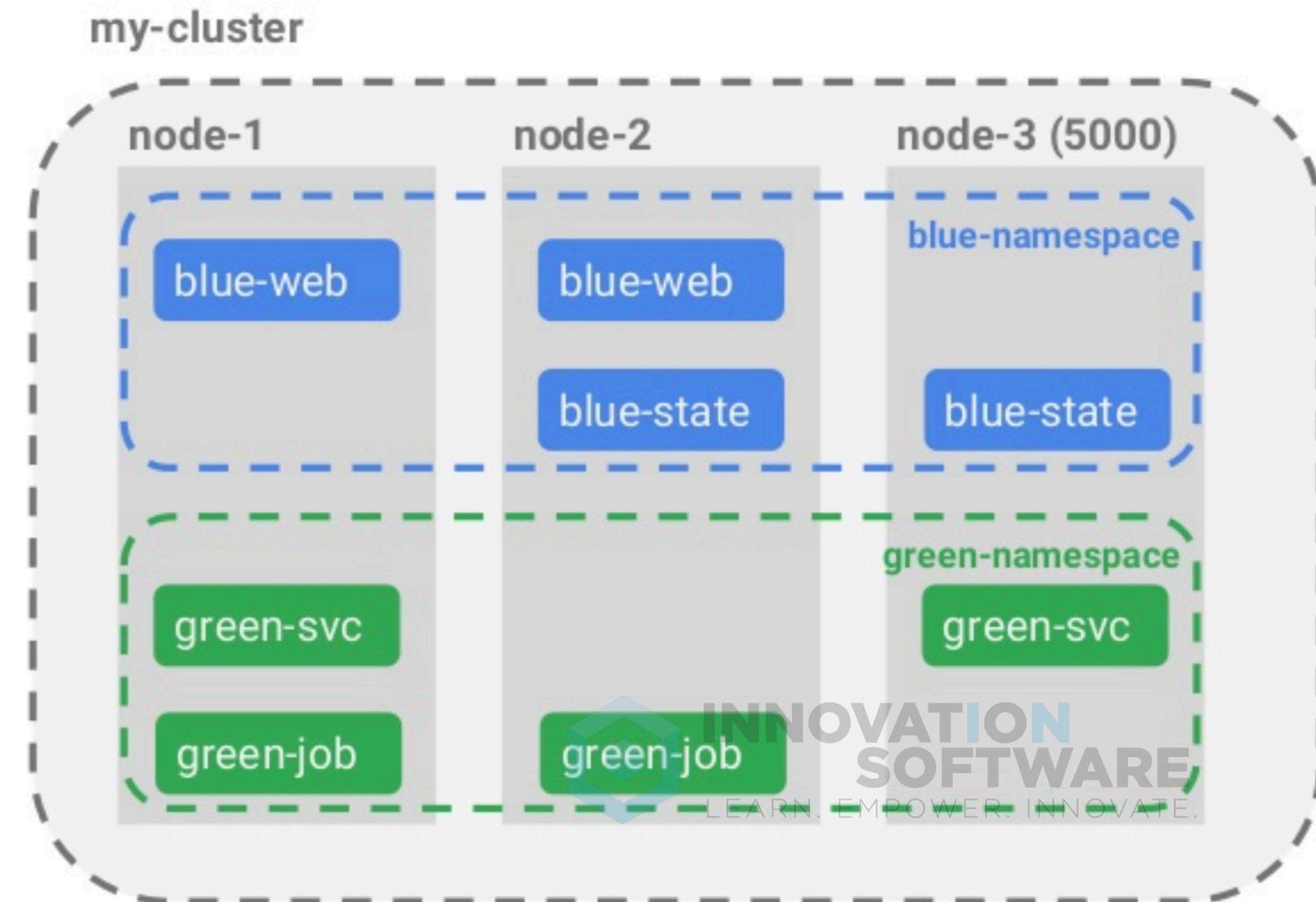


Introducing RBAC:

- Per-namespace/ resource, role, action

Examples:

- **Alice** can list **Eng** services, but not **HR**
- **Bob** can create Pods in **Test** namespace, but not in **Prod**
- **Scheduler** can read **Pods** but not **Secrets**



RBAC lab

- Create user
- Generate SSL credentials for new user
- Create RBAC Role and RoleBinding
- Assign correct permissions to user
- Confirm RBAC is working as expected



Deployments lab

- Deploy Redis master
 - Deployment/Service
- Deploy Redis slaves
 - Deployment/Service
- Deploy Frontend
 - Deployment/Service

****NOTE:**** Investigate YAML files to see how DNS is being used between services.



Kubernetes API & Security

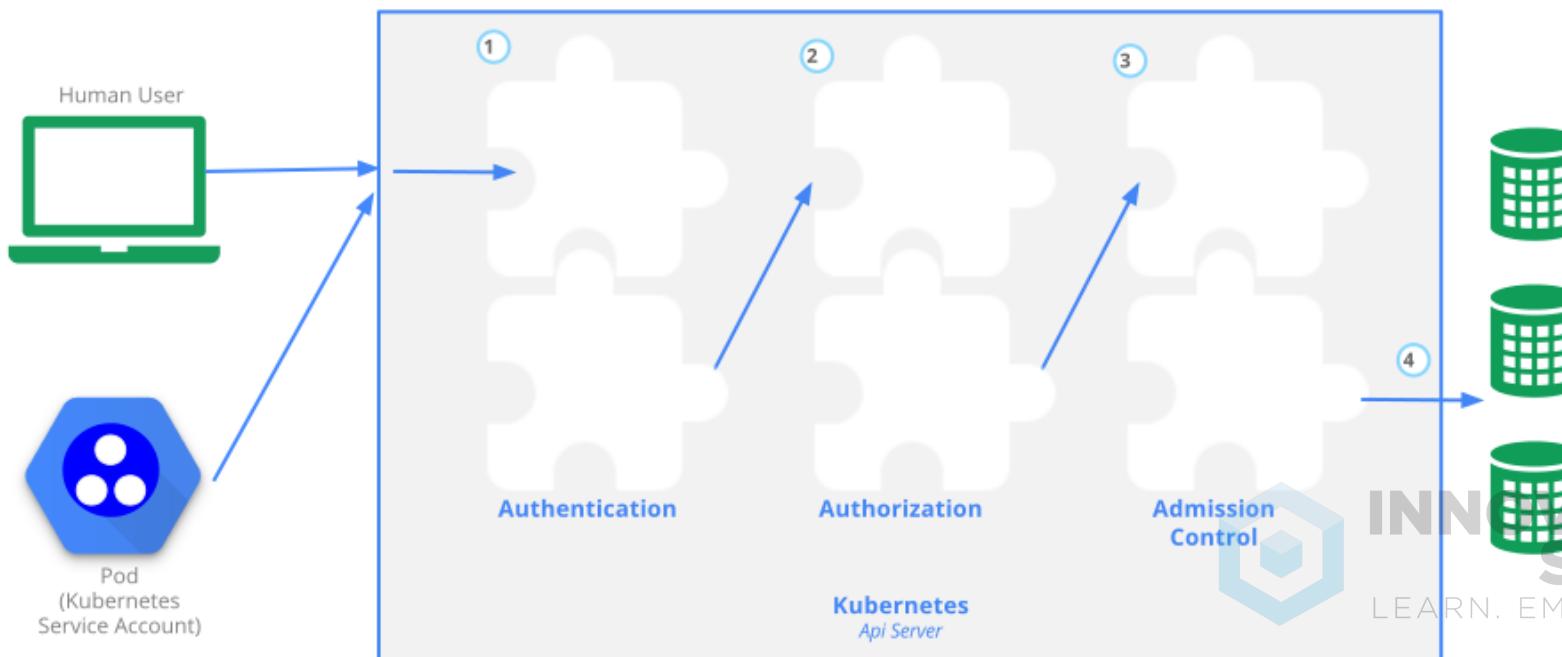


INNOVATION
SOFTWARE

LEARN. EMPOWER. INNOVATE.



API Authorization Flow



INNOVATION SOFTWARE



LEARN. EMPOWER. INNOVATE.

API Authentication

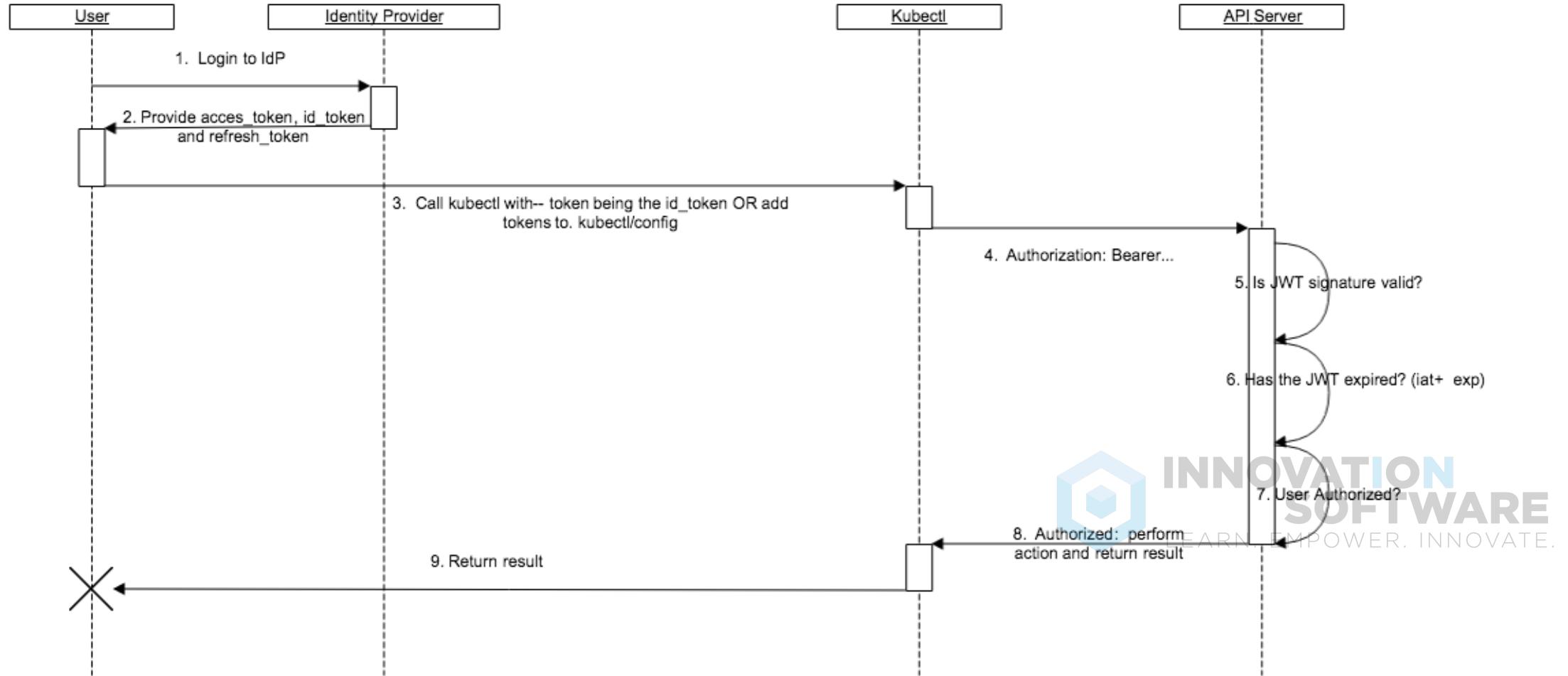
- In a typical Kubernetes cluster, the API serves on port 443
- Typically **\$USER/.kube/config** contains the root certificate
- If request cannot be authenticated it's rejected with HTTP status code 401
- Kubernetes can use “usernames” for access control decisions and in request logging, it does not have a user object nor does it store usernames or other information about users in its object store.

API Authentication

- types: user, service account
- users are managed by independent service, not kubernetes cluster
- service accounts are managed by kubernetes api
- Authentication modules include Client Certificates, Password, and Plain Tokens, Bootstrap Tokens, and JWT Tokens (used for service accounts).



API Auth with Identity Provider



API Authorization

Request must include:

- username
- action
- object affected by action

API Authorization

- Example of policy for Bob

```
{  
    "apiVersion": "abac.authorization.kubernetes.io/v1beta1",  
    "kind": "Policy",  
    "spec": {  
        "user": "bob",  
        "namespace": "projectCaribou",  
        "resource": "pods",  
        "readonly": true  
    }  
}
```

API Authorization

This request will be authorized for Bob

```
{  
  "apiVersion": "authorization.k8s.io/v1beta1",  
  "kind": "SubjectAccessReview",  
  "spec": {  
    "resourceAttributes": {  
      "namespace": "projectCaribou",  
      "verb": "get",  
      "group": "unicorn.example.org",  
      "resource": "pods"  
    }  
  }  
}
```

API Access (kubectl proxy)

- kubectl proxy creates a tunnel from local machine to API endpoint



Admission Controllers

- Additional custom security modules which may reject the request
- Must be enabled on the api server when kube-apiserver is started
- Examples:
 - AlwaysPullImages
 - Security enhancement (checks image access every time new Pod created)
 - NameSpaceAutoProvision
 - Priority
 - Checks priorityClassName field, fails if not listed
 - many more!

Additional info: <https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/>



Monitoring



Monitoring

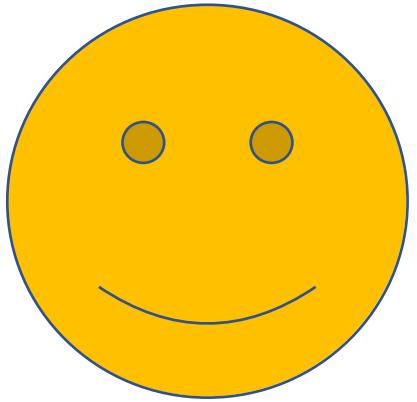
Why monitor?



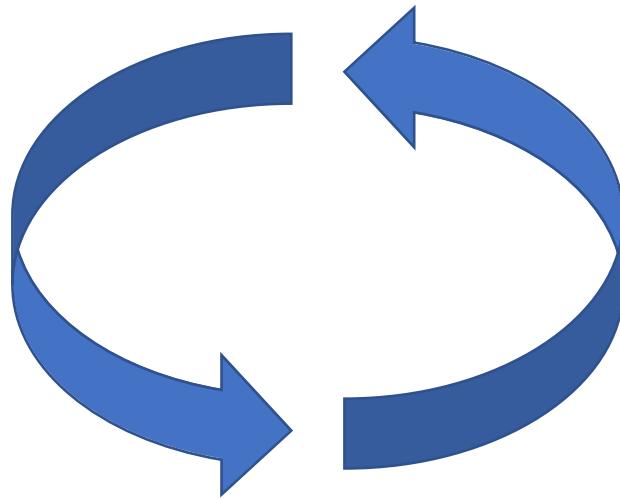


ON SOFTWARE
LEARN. EMPOWER. INNOVATE.

Monitoring



Quality Assurance



Continuous Improvement



Monitoring

Not about collecting data

- Why vs How



Monitoring

Why monitor?

- Know when things go wrong
- Debug and learn from issues
- Historical view/Trends
- Drive technical/business decisions
- Feed into other systems/processes (QA, Security, automation)

Monitoring Challenges

- Monitoring tools are limited
 - Technically
 - Conceptually
- Tools don't scale well and are difficult to manage
- Operational practices don't align with business needs

Monitoring Challenges

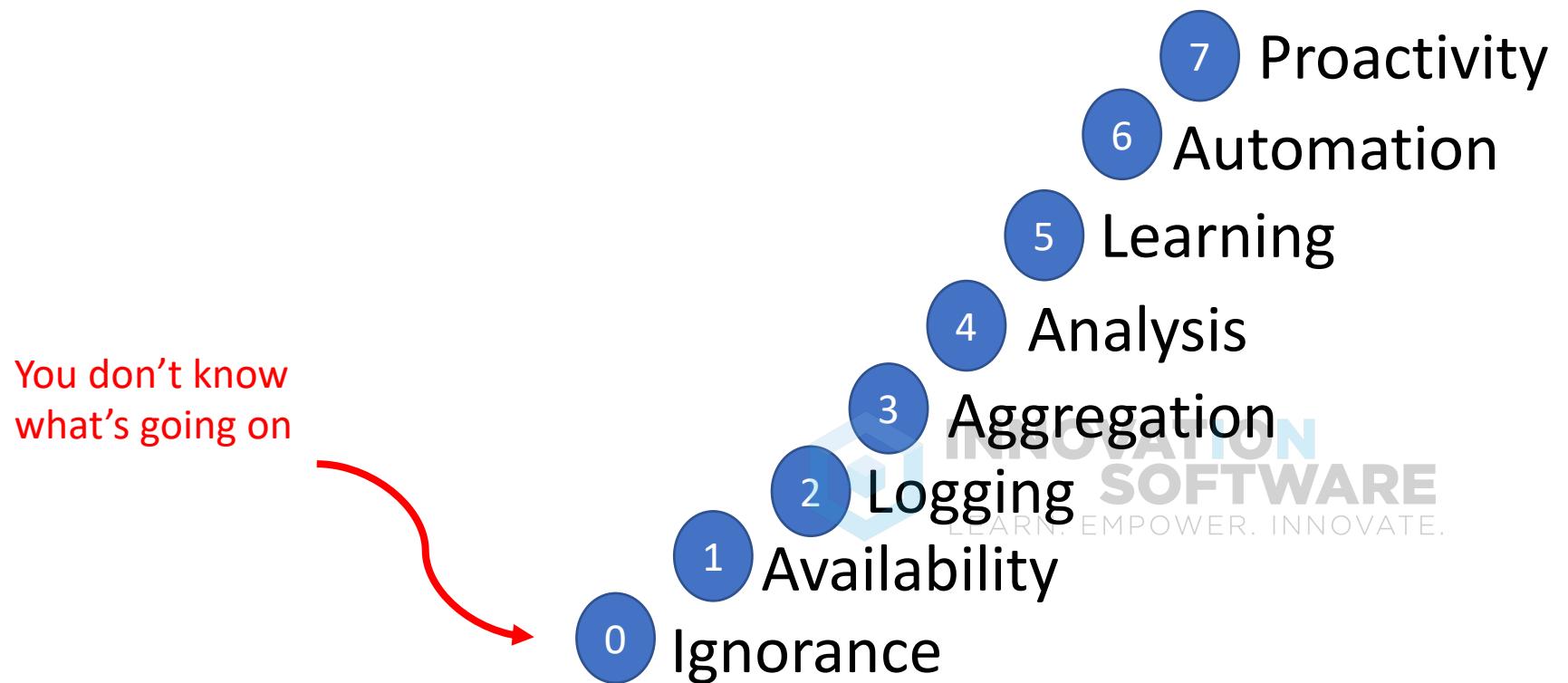
Example:

Customers care about increased latency and it's in your SLAs.
Monitoring only supports alerting on individual machines performance
and not the entire clusters.

Result: Engineers get a bunch of non critical/false-positive alerts and
are woken up for non-issues. This leads to fatigue and ignoring alerts.



Monitoring



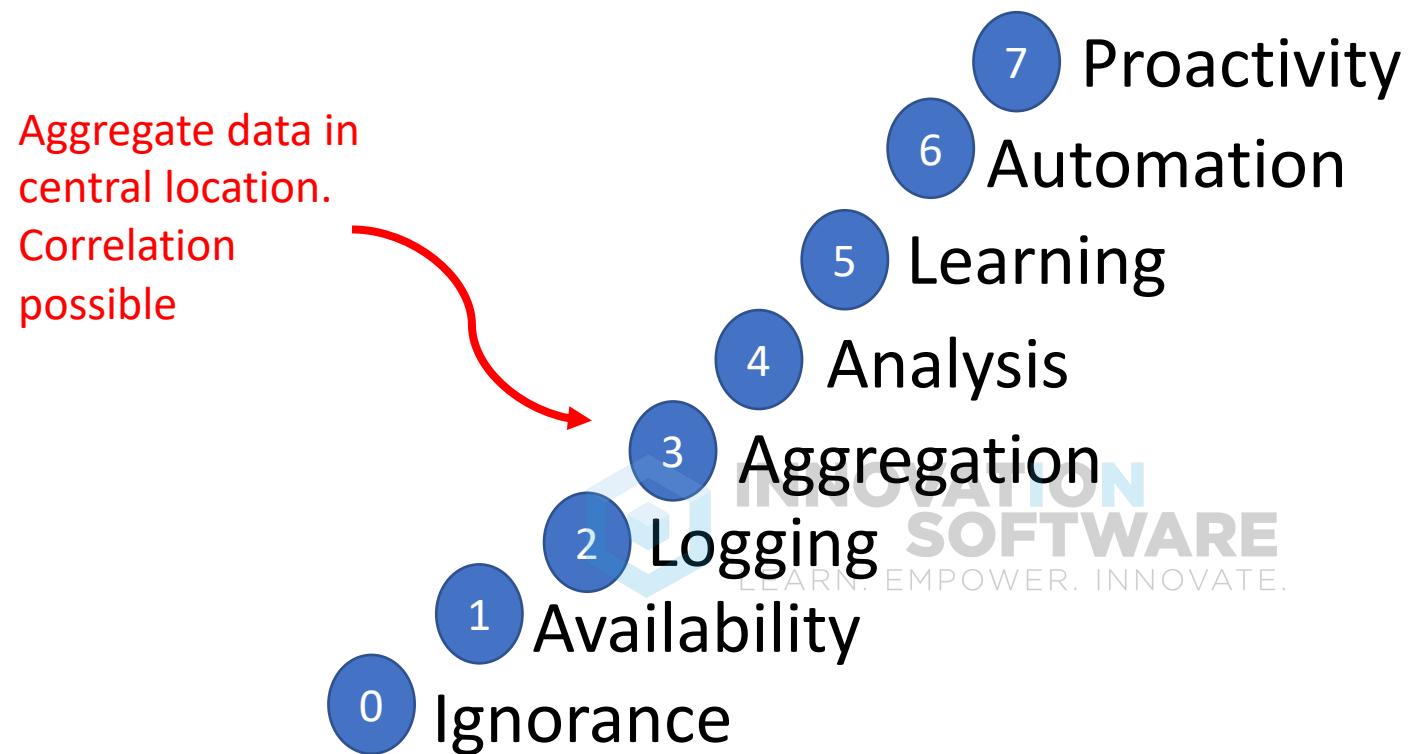
Monitoring



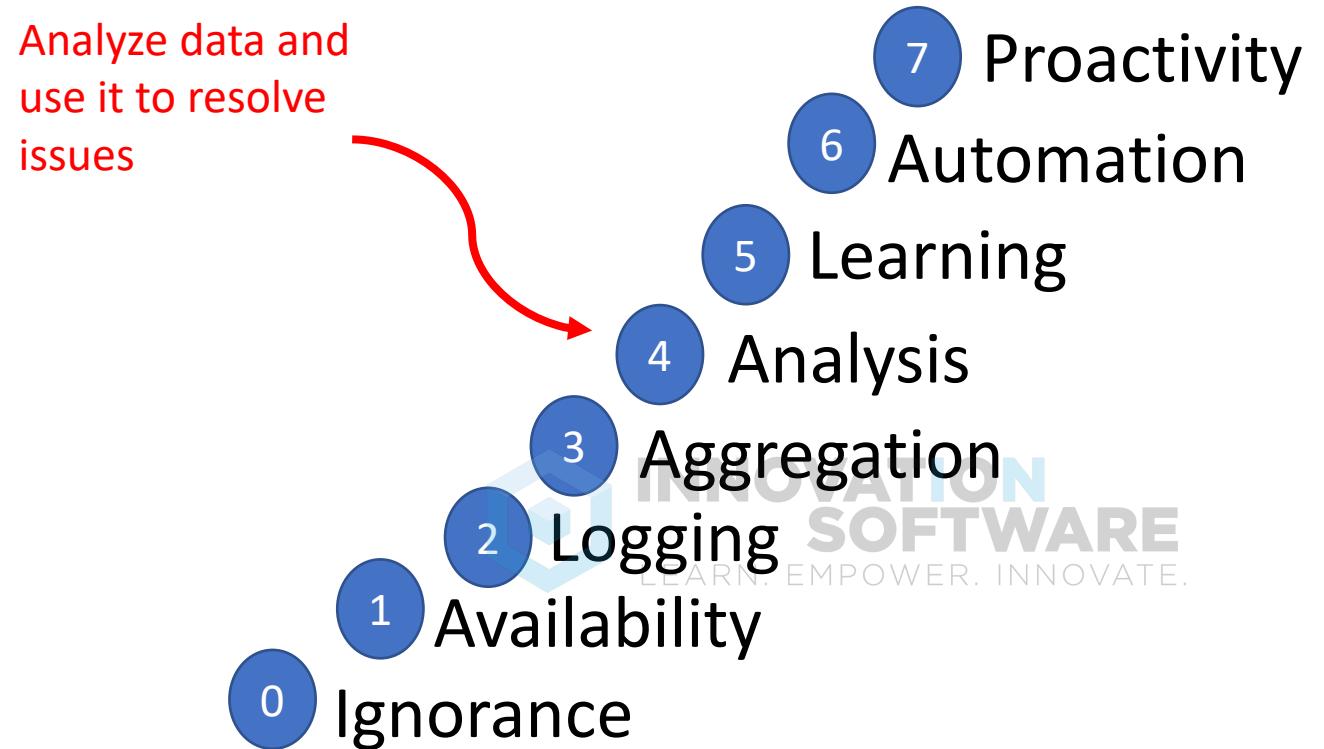
Monitoring



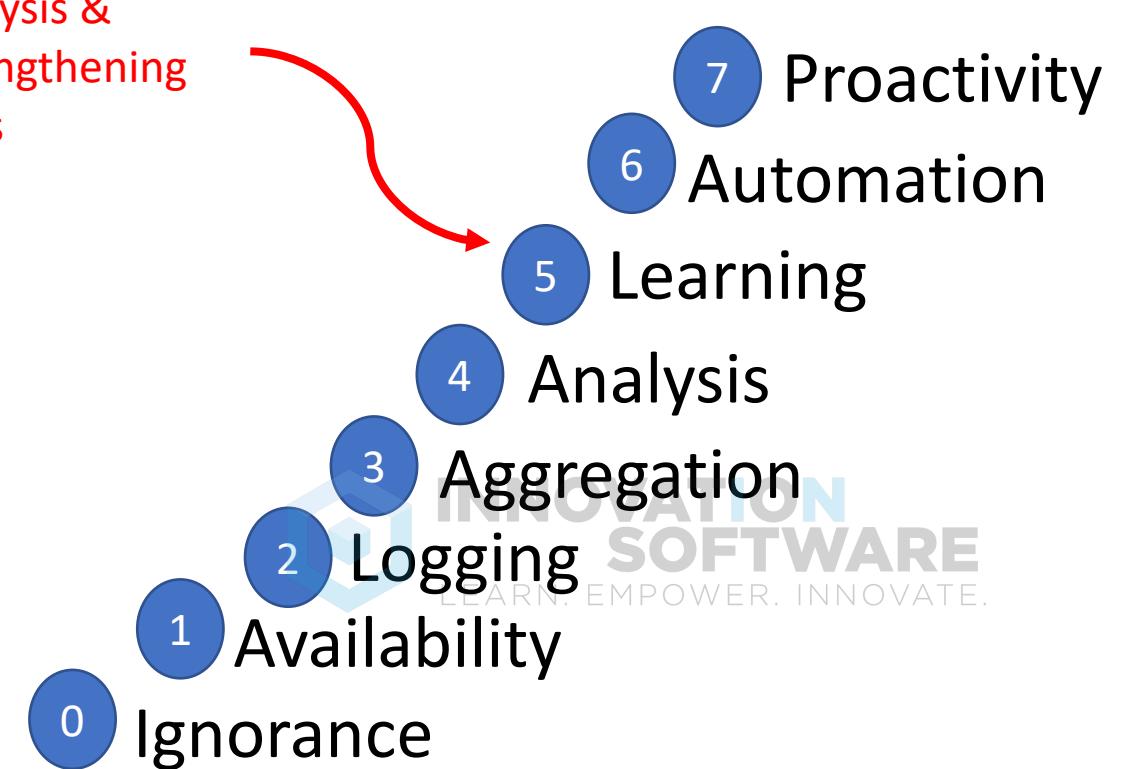
Monitoring



Monitoring

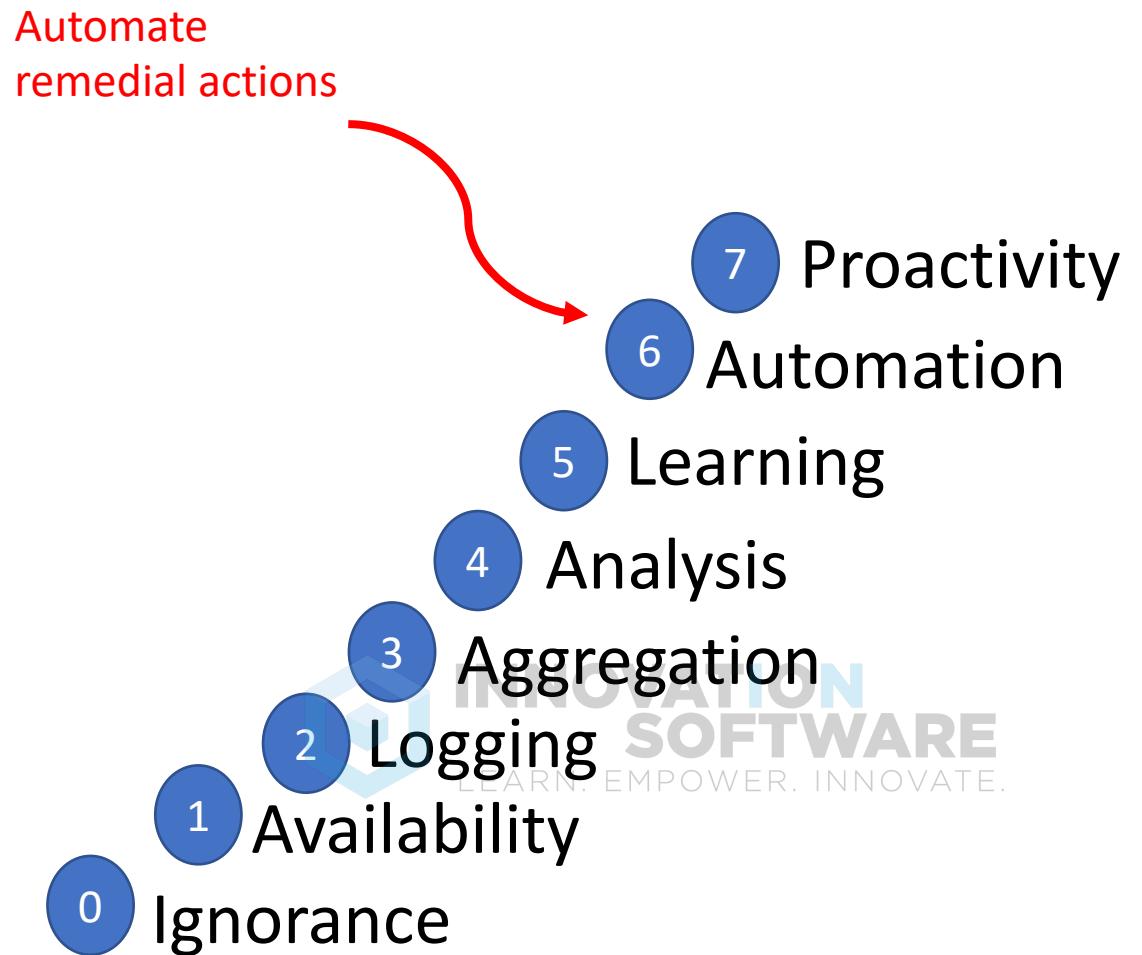


Monitoring

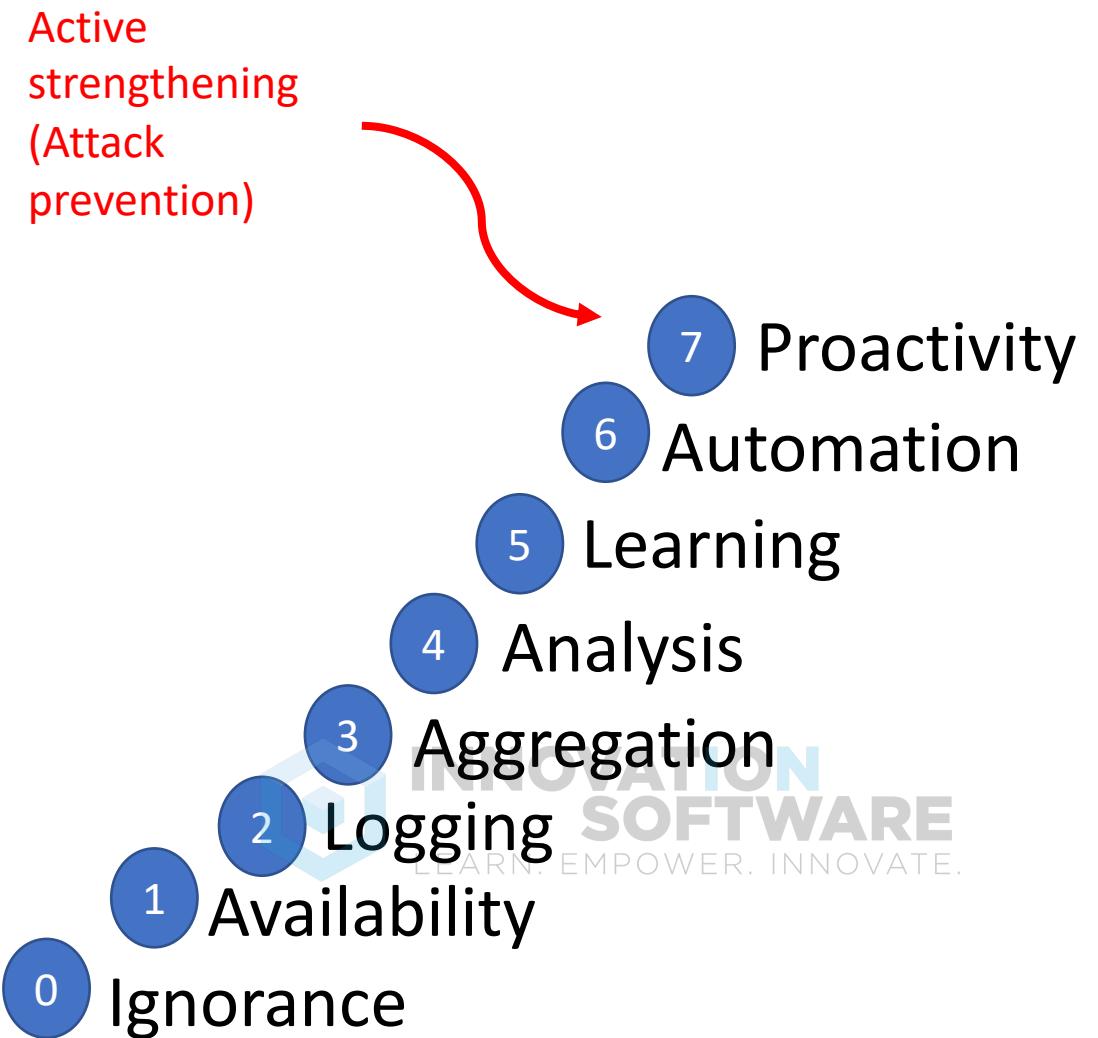


Root cause
analysis &
strengthening
fixes

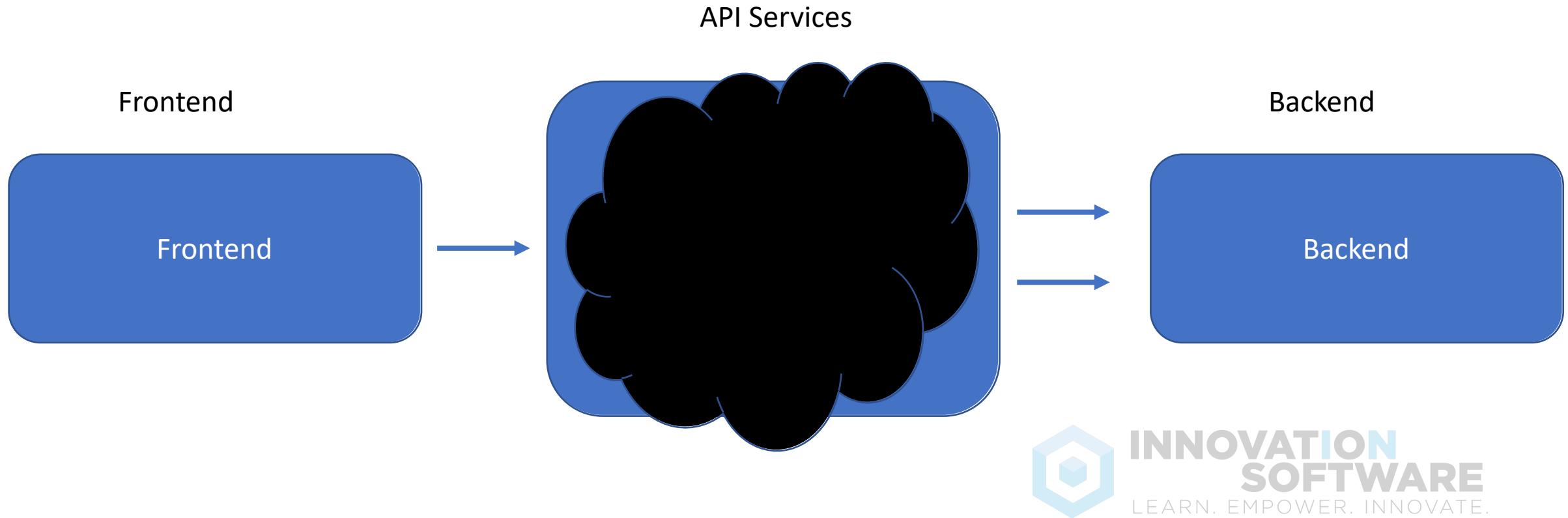
Monitoring



Monitoring



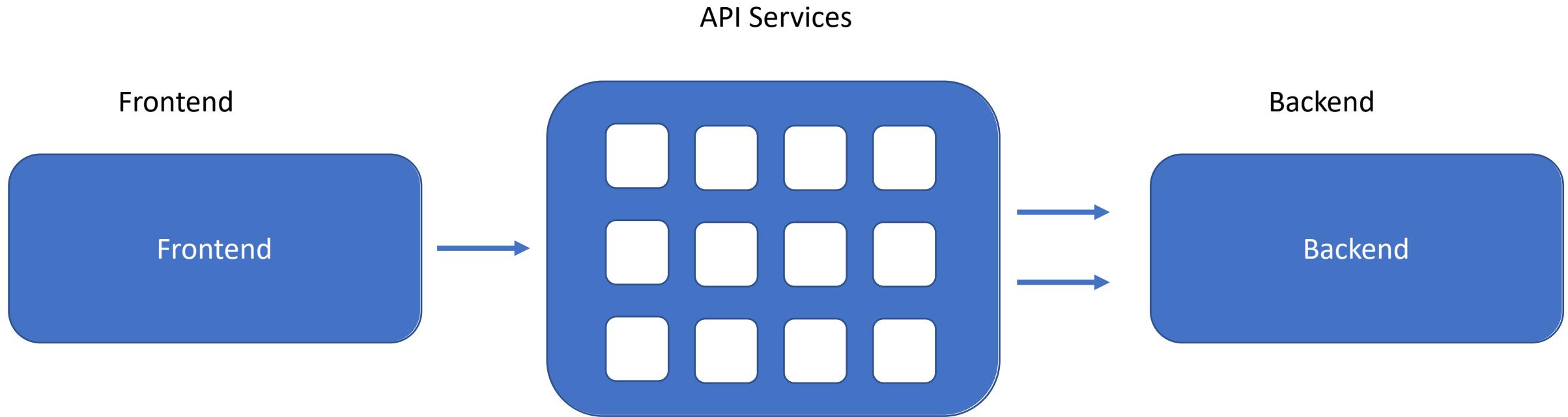
Monitoring



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

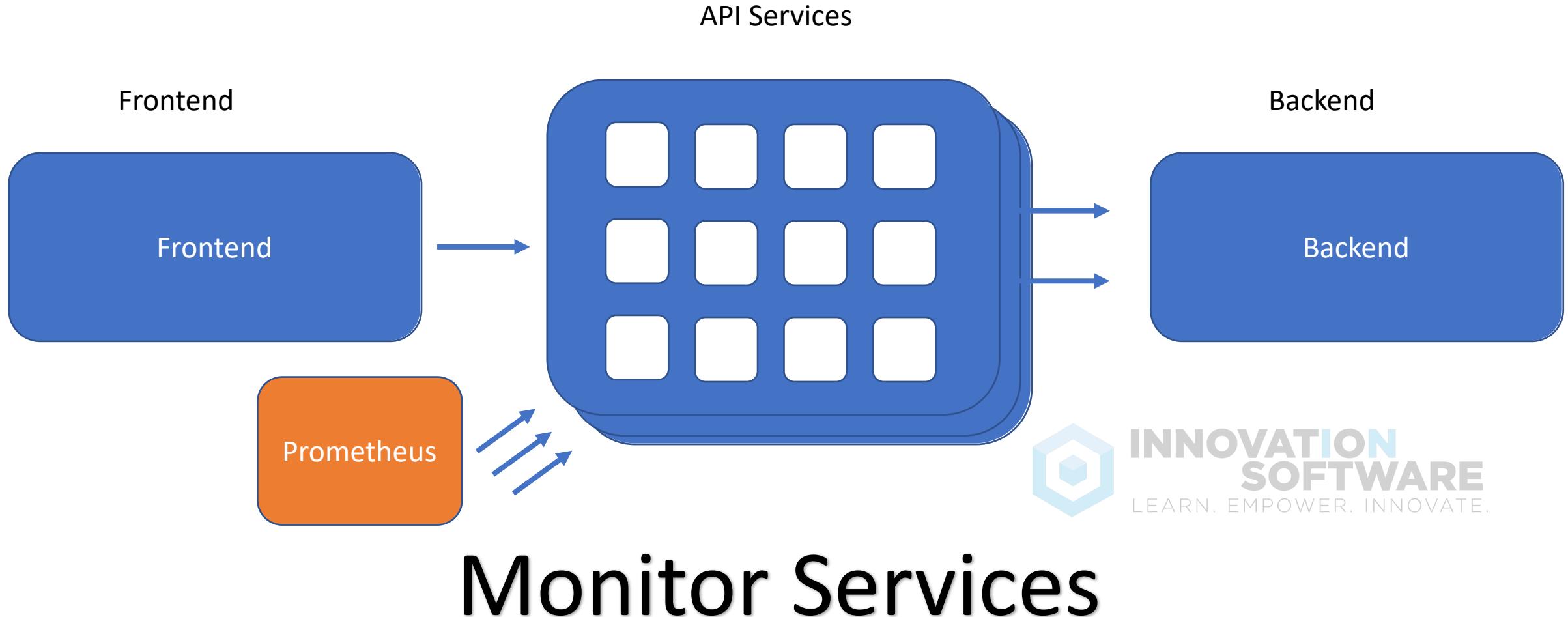
Limited visibility

Monitoring



Monitor Internals

Monitoring



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Monitoring Prometheus

The screenshot shows the Prometheus web interface. At the top, there is a navigation bar with links for Prometheus, Alerts, Graph, Status, and Help. Below the navigation bar is a search bar labeled "Expression (press Shift+Enter for newlines)". To the right of the search bar are two buttons: "Execute" and "- insert metric at cursor -". Below the search bar, there are two tabs: "Graph" (which is selected) and "Console". Under the "Graph" tab, there is a table with two columns: "Element" and "Value". The table contains the text "no data". At the bottom left, there is a blue button labeled "Add Graph".



Monitoring

Prometheus

Prometheus Alerts Graph Status ▾ Help

Targets

kubernetes-apiservers (1/1 up)				
Endpoint	State	Labels	Last Scrape	Error
https://192.168.64.2:8443/metrics	UP	instance="192.168.64.2:8443"	49.262s ago	

kubernetes-cadvisor (1/1 up)				
Endpoint	State	Labels	Last Scrape	Error
https://kubernetes.default.svc:443/api/v1/nodes/minikube/proxy/metrics/cadvisor	UP	beta_kubernetes_io_arch="amd64" beta_kubernetes_io_os="linux" instance="minikube" kubernetes_io_hostname="minikube"	46.023s ago	

kubernetes-nodes (1/1 up)				
Endpoint	State	Labels	Last Scrape	Error
https://kubernetes.default.svc:443/api/v1/nodes/minikube/proxy/metrics	UP	beta_kubernetes_io_arch="amd64" beta_kubernetes_io_os="linux" instance="minikube" kubernetes_io_hostname="minikube"	8.864s ago	

kubernetes-pods (1/1 up)				
Endpoint	State	Labels	Last Scrape	Error
http://172.17.0.3:9090/metrics	UP	instance="172.17.0.3:9090" kubernetes_namespace="monitoring" kubernetes_pod_name="prometheus-5947dcc755-97ajd" name="prometheus" pod_template_hash="1503877311"	8.719s ago	



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Monitoring

Grafana

The screenshot shows the Grafana Home Dashboard. At the top, there's a navigation bar with a gear icon, a "Home" dropdown, and a settings gear. On the right, there are zoom controls ("Zoom Out", "Last 6 hours") and a refresh button.

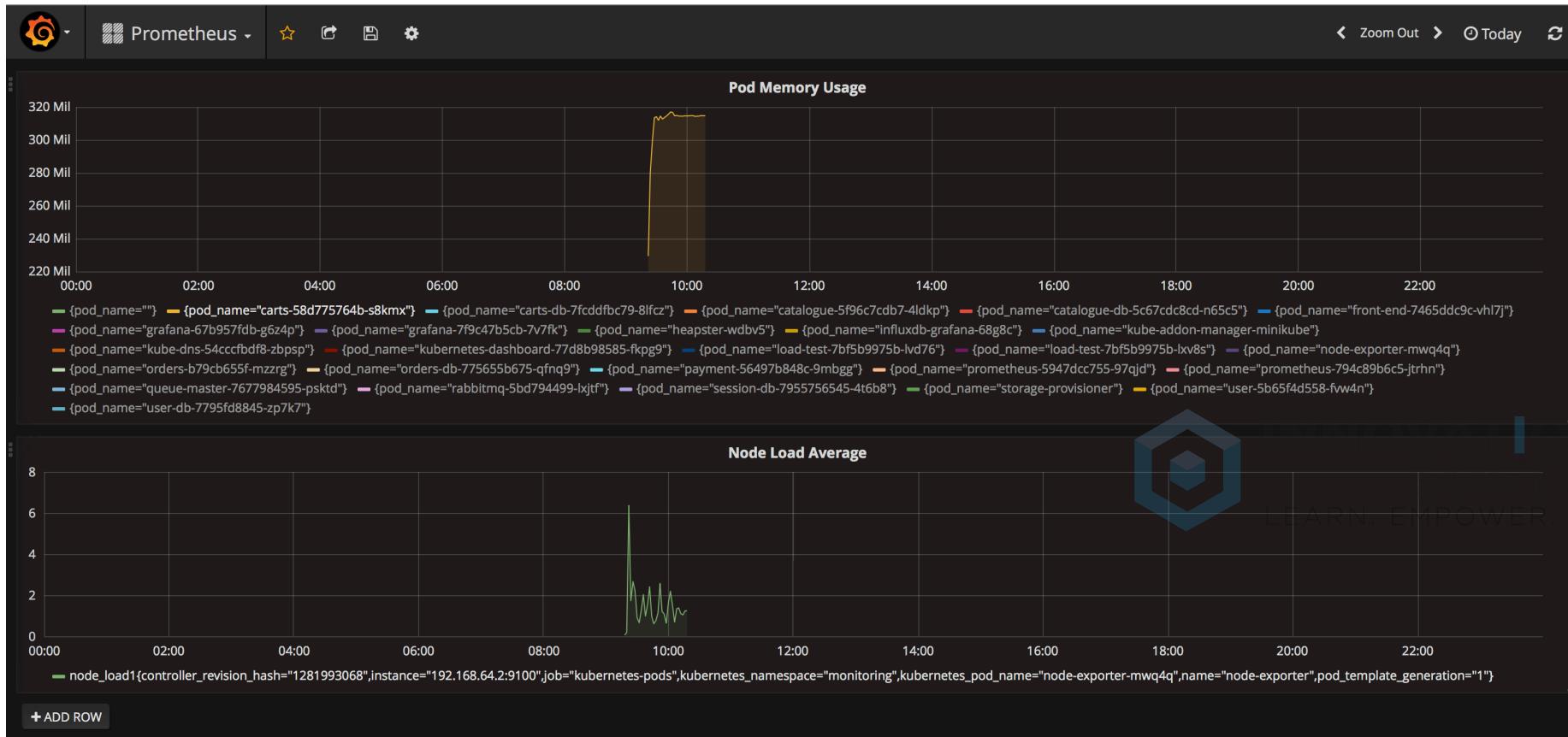
The main title is "Home Dashboard". Below it, a "Getting Started with Grafana" section provides five steps:

- Install Grafana (checkmark)
- Create your first data-source (checkmark)
- Create your first dashboard (checkmark)
- Add Users (green button)
- Install apps & plugins (link)

On the left, there are sections for "Starred dashboards" (Prometheus) and "Recently viewed dashboards" (Prometheus). On the right, there are sections for "Installed Apps" (None installed, link), "Installed Panels" (None installed, link), and "Installed Datasources" (None installed, link). A large "LEARN. EMPOWER. INNOVATE" banner is visible at the bottom right.

Monitoring

Grafana



N
WARE
INNOVATE.

Monitoring

Grafana

The screenshot shows the Grafana Alerting interface. At the top, there is a navigation bar with a gear icon and the text "Alerting". Below the navigation bar, the title "Alert List" is displayed in a bold, italicized font. To the right of the title are two buttons: "Configure notifications" and "How to add an alert". Below the title, there are two buttons: "Filter by state" and "All". The main area is currently empty, indicating no alerts are present.



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Monitoring

Grafana

The screenshot shows the Grafana interface for managing data sources. At the top, there's a navigation bar with a logo, a dropdown for 'Data Sources', and a green button labeled '+ Add data source'. Below this, the main area is titled 'Data Sources' and features a single entry for 'PROMETHEUS'. This entry includes a icon of a flame, the name 'prometheus', a status indicator 'default', and the URL 'http://prometheus:9090'. To the right of the data source list are two small icons: a grid and a list. In the bottom right corner of the main area, there's a watermark or logo for 'INNOVATION SOFTWARE' with the tagline 'LEARN. EMPOWER. INNOVATE.'

PROMETHEUS

prometheus

default

http://prometheus:9090

+ Add data source

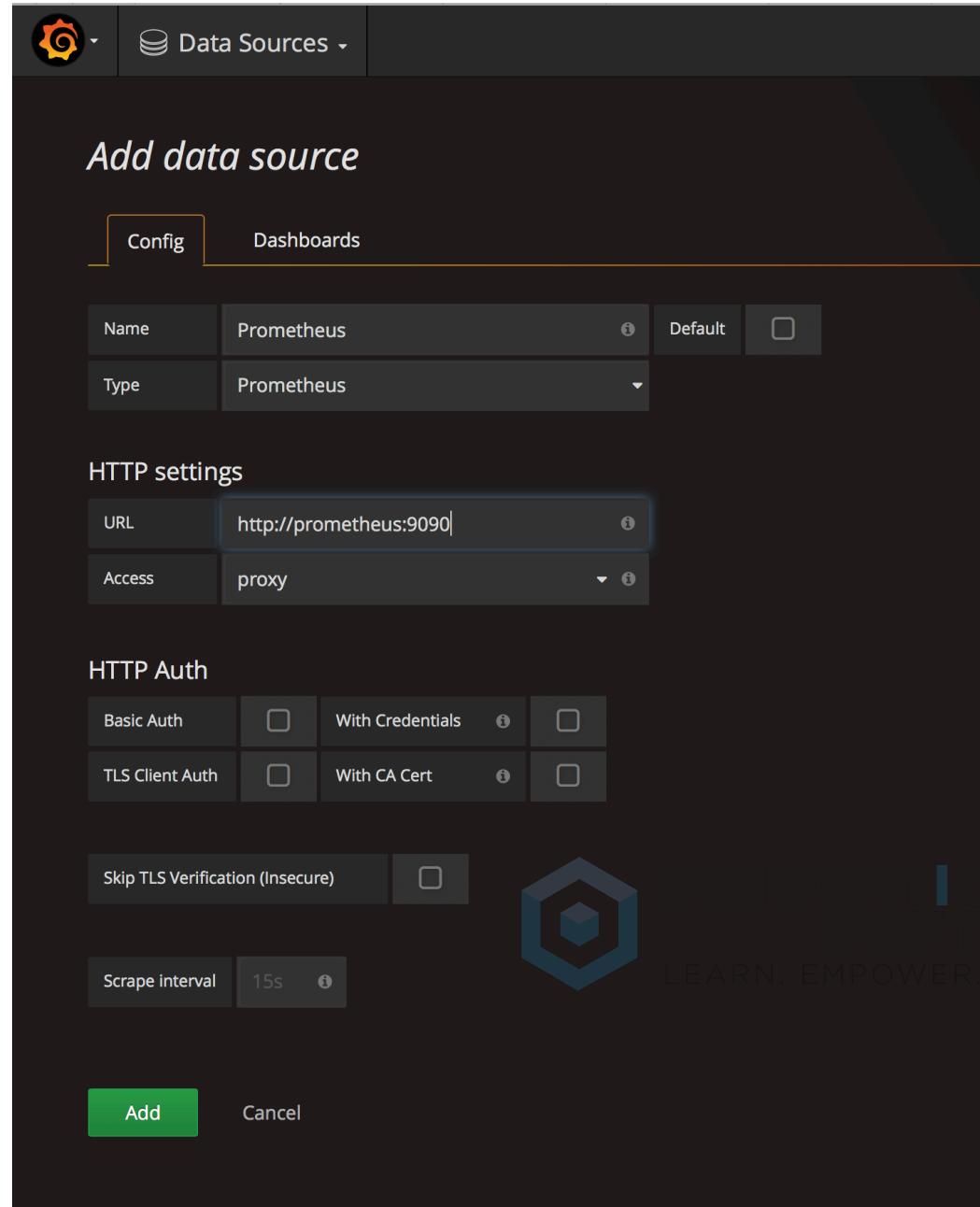
INNOVATION SOFTWARE
LEARN. EMPOWER. INNOVATE.

Monitoring

Grafana

&

Prometheus



INNWARE
LEARN. EMPOWER. INNOVATE.

Monitoring

Add graph

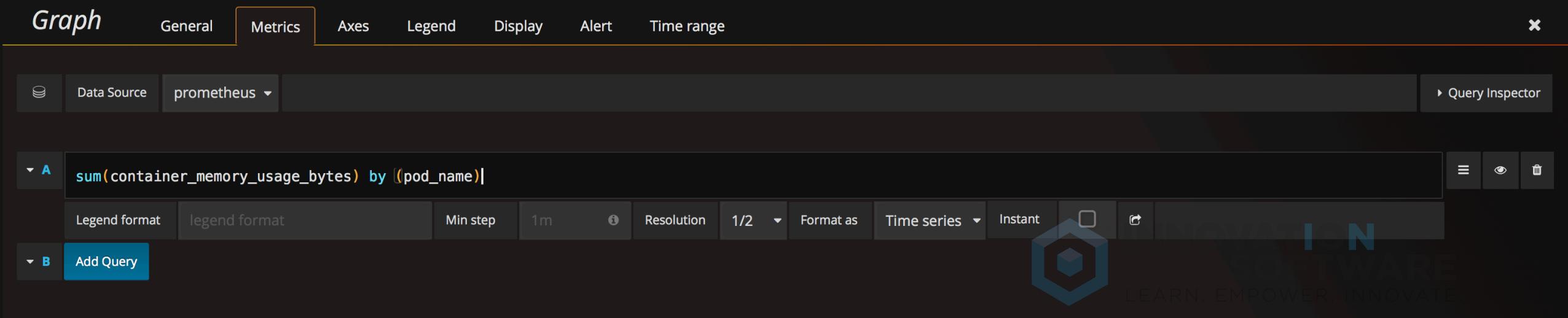
Graph General Metrics Axes Legend Display Alert Time range x

Data Source prometheus ▶ Query Inspector

▼ A sum(container_memory_usage_bytes) by (pod_name)|☰ ⚡ 🗑

Legend format legend format Min step 1m ⓘ Resolution 1/2 Format as Time series Instant

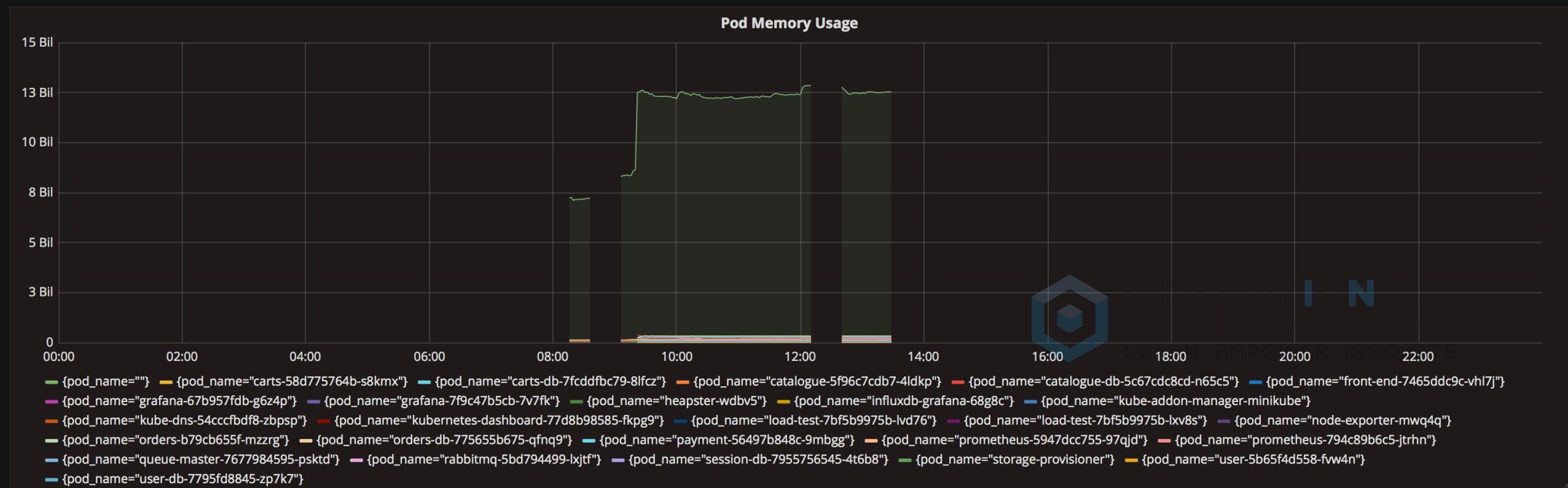
▼ B Add Query



The screenshot shows the Grafana interface for creating a new dashboard. The top navigation bar has tabs for Graph, General, Metrics, Axes, Legend, Display, Alert, and Time range, with Metrics currently selected. Below the tabs is a data source dropdown set to 'prometheus'. The main area contains a query editor with a query A: 'sum(container_memory_usage_bytes) by (pod_name)'. The editor includes controls for legend format, min step (set to 1m), resolution (set to 1/2), format as (set to Time series), and instant mode. A 'Add Query' button is visible at the bottom left. The background features a dark theme with a hexagonal icon and text for 'IN SOFTWARE LEARN. EMPOWER. INNOVATE'.

Monitoring

Grafana graph



Kubernetes Secrets



INNOVATION
SOFTWARE

LEARN. EMPOWER. INNOVATE.



Application Secrets

What secrets do applications have?

- Database credentials
- API credentials & endpoints (Twitter, Facebook etc.)
- Infrastructure API credentials (Google, Azure, AWS)
- Private keys (TLS, SSH)
- Many more!

Application Secrets

It is a bad idea to include these secrets in your code.

- Accidentally push up to GitHub with your code
- Push into your file storage and forget about
- Etc.



Application Secrets

There are bots crawling GitHub searching for secrets

Real life example:

Dev put keys out on GitHub, woke up next morning with a ton of emails and missed calls from Amazon

- 140 instances running under Dev's account.
- \$2,375 worth of Bitcoin mining

Create Secret

- Designed to hold all kinds of sensitive information
- Can be used by Pods (filesystem & environment variables) and the underlying kubelet when pulling images

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  password: mmyWfoidfluL==
  username: NyhdOKwB
```



Pod Secret

```
kind: Pod
metadata:
  name: secret-env-pod
spec:
  containers:
    - name: mycontainer
      image: redis
      env:
        - name: SECRET_USERNAME
          valueFrom:
            secretKeyRef:
              name: mysecret
              key: username
```



Volume Secret

```
spec:  
  containers  
    - name: mycontainer  
      image: redis  
      volumeMounts:  
        - name: "secrets"  
          mountPath: "/etc/my-secrets"  
          readOnly: true  
  volumes:  
    - name: "secrets"  
      secret:  
        secretName: "mysecret"
```



Secrets Lab

- Create secrets file
- Deploy application that uses secrets
- Decrypt secrets



Kubernetes Init Containers



INNOVATION
SOFTWARE

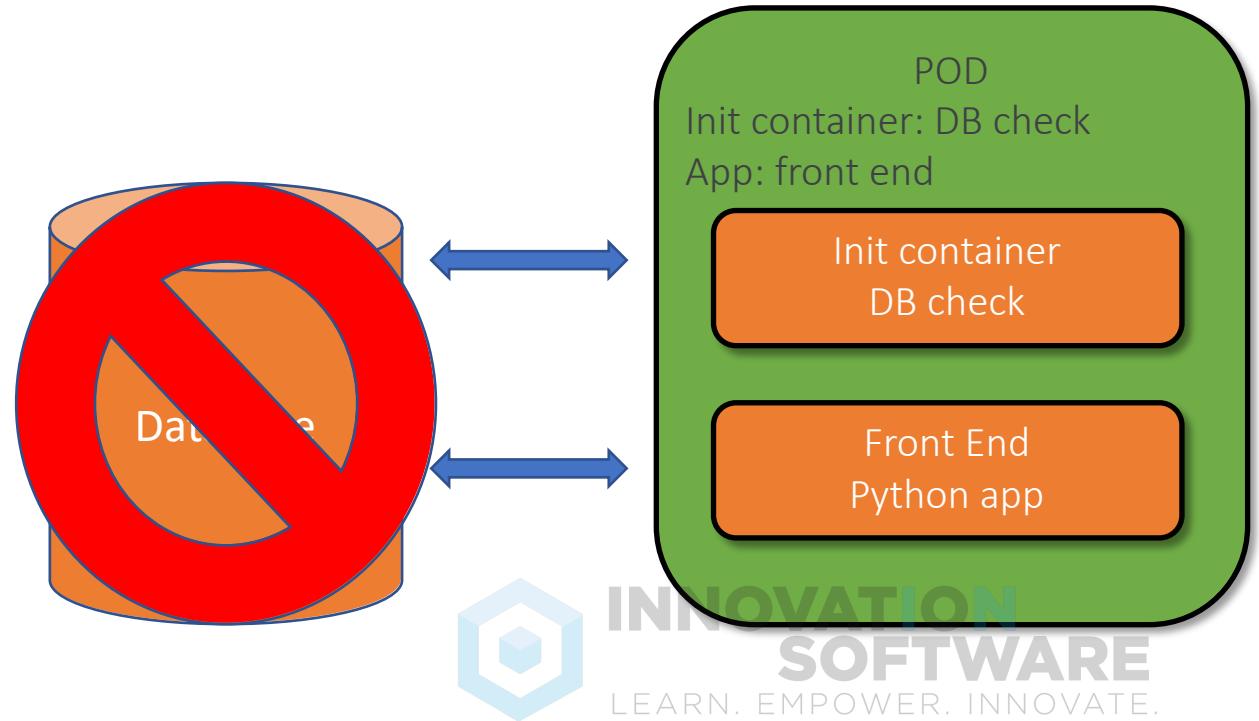
LEARN. EMPOWER. INNOVATE.



Init containers

Application orchestration

- Init container runs first to completion
 - If failure, POD is restarted until successful.



Init containers

- Support same features as application containers.
 - Resource limits
 - Volumes
 - Security settings
 - Etc.
- Do not support readiness probes since they must run to completion before POD can be “ready”



Kubernetes 1.5 syntax

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
  annotations:
    pod.beta.kubernetes.io/init-containers: '[{"name": "init-myservice", "image": "busybox", "command": ["sh", "-c", "until nslookup myservice; do echo waiting for myservice; sleep 2; done;"]}, {"name": "init-mydb", "image": "busybox", "command": ["sh", "-c", "until nslookup mydb; do echo waiting for mydb; sleep 2; done;"]}]'
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo The app is running! && sleep 3600']
```



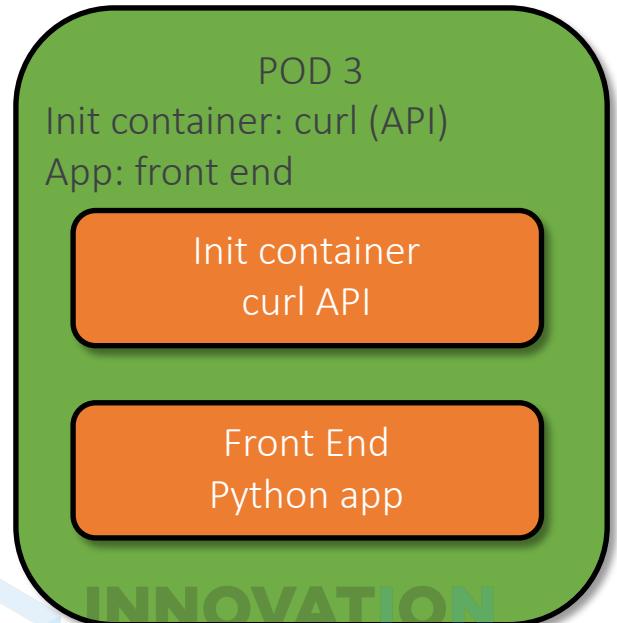
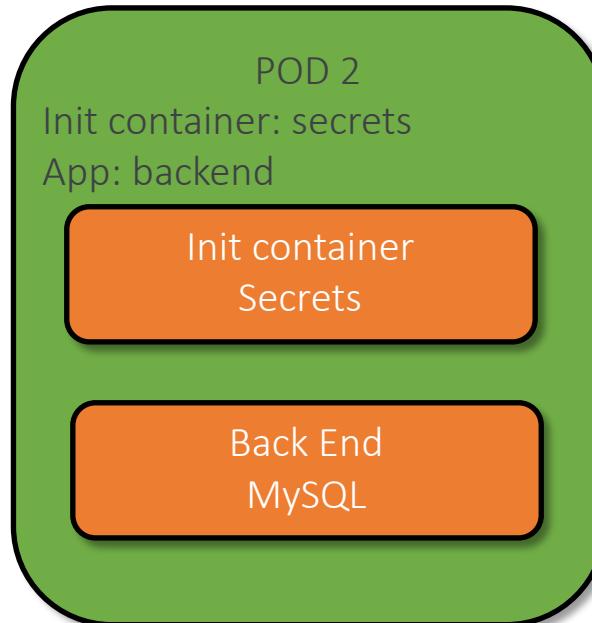
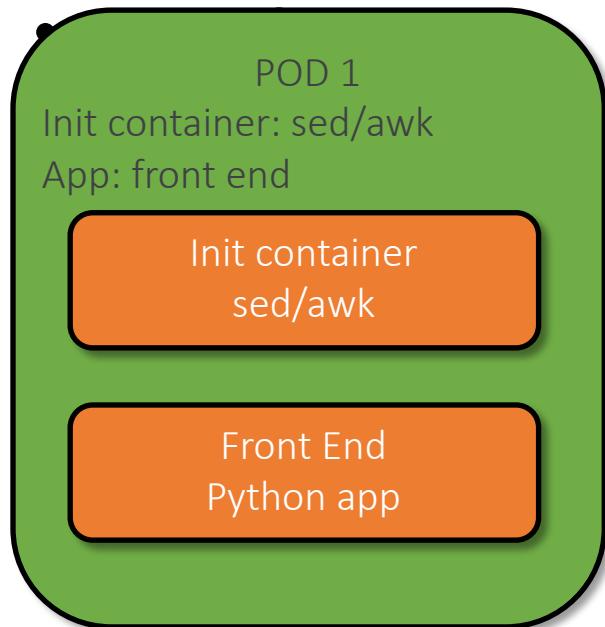
Kubernetes 1.6+ syntax

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo The app is running! && sleep 3600']
  initContainers:
  - name: init-myservice
    image: busybox
    command: ['sh', '-c', 'until nslookup myservice; do echo waiting for myservice; sleep 2; done;']
  - name: init-mydb
    image: busybox
    command: ['sh', '-c', 'until nslookup mydb; do echo waiting for mydb; sleep 2; done;']
```



LEARN EMPOWER INNOVATE

Init container use-cases



INNOVATION
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Init containers Lab

- Deploy POD with init containers that wait for services to come online
- Investigate POD status
- Create services
- Confirm init containers executed and POD was started
- Create POD with init container that fetched webpage for nginx web server.

Local Kubernetes Cluster



INNOVATION
SOFTWARE

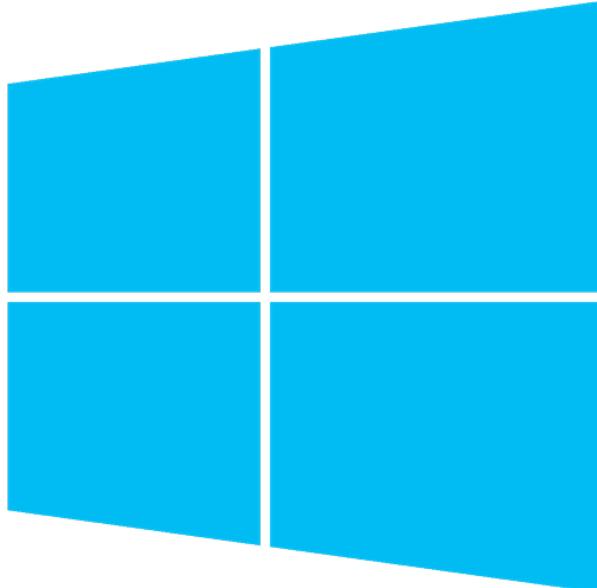
LEARN. EMPOWER. INNOVATE.



Minikube: Run Kubernetes All-In-One

Run single-node Kubernetes cluster on local machine

- Minikube runs on all 3 operating systems
 - Windows, Mac, and Linux



- VirtualBox
- Hyper-V

- VirtualBox
- Xhyve
- Vmware Fusion

- VirtualBox
- KVM

Minikube: Supported Features

DNS

NodePorts

Ingress

Load Balancer

Container Runtime

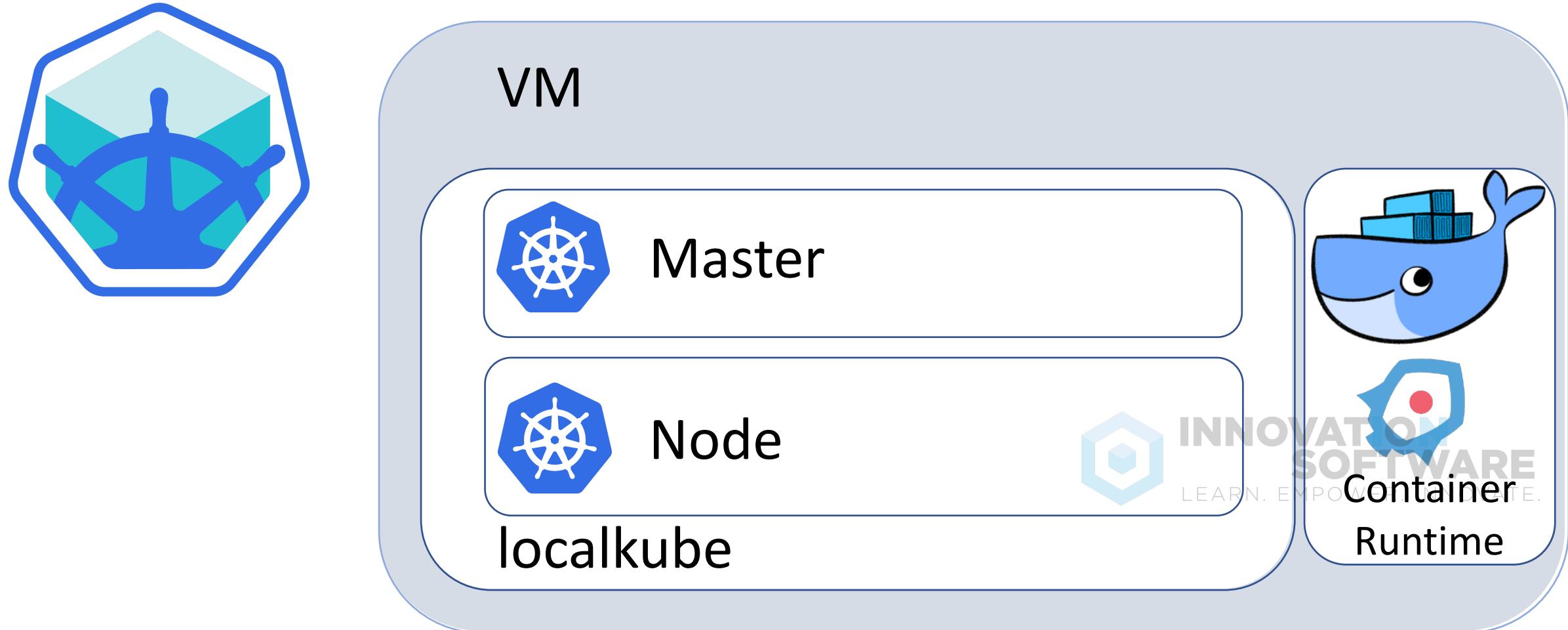
Persistent Volumes

ConfigMaps
Secrets



INNOVATION Dashboard
SOFTWARE
LEARN. EMPOWER. INNOVATE.

Minikube: Architecture



Capstone Lab

- Create ConfigMap that contains value of “success”
- Create secret with a passphrase (i.e. “KubernetesRocks”)
- Deploy web application:
 - Serves webpage with contents of ConfigMap
 - Secret is available at /secrets/passphrase.txt
- Create service for web application
 - Served on port 8081
 - Available externally

