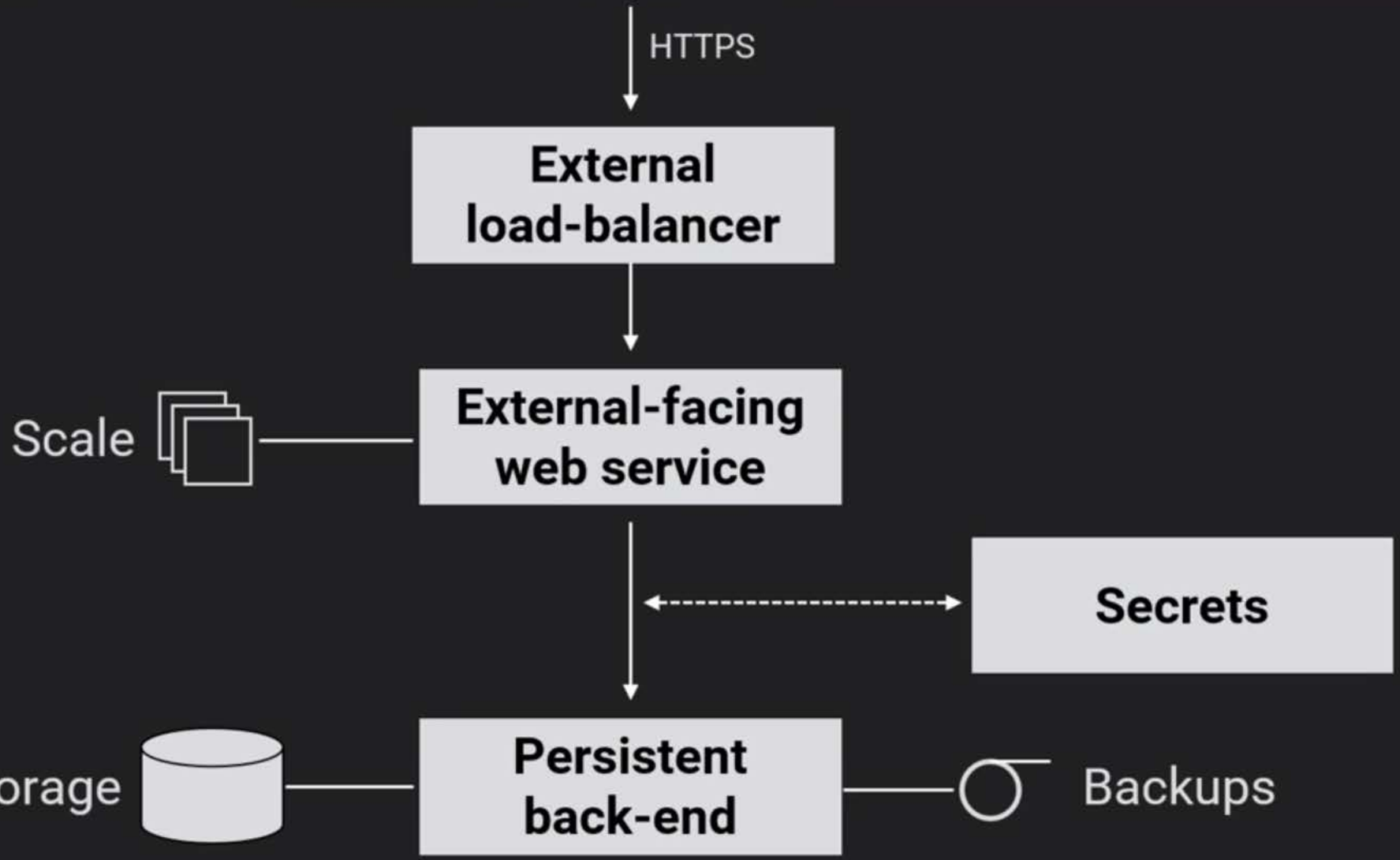
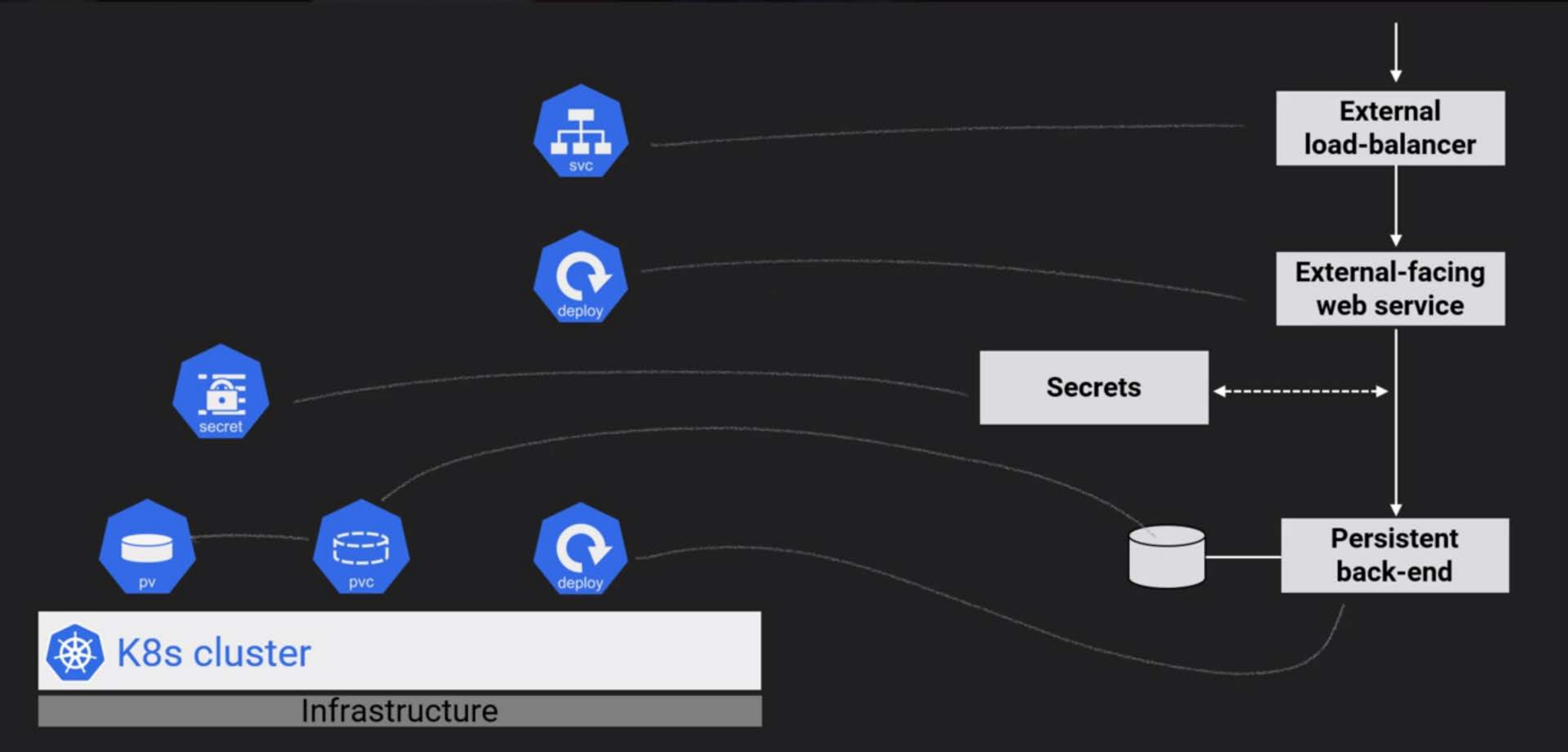
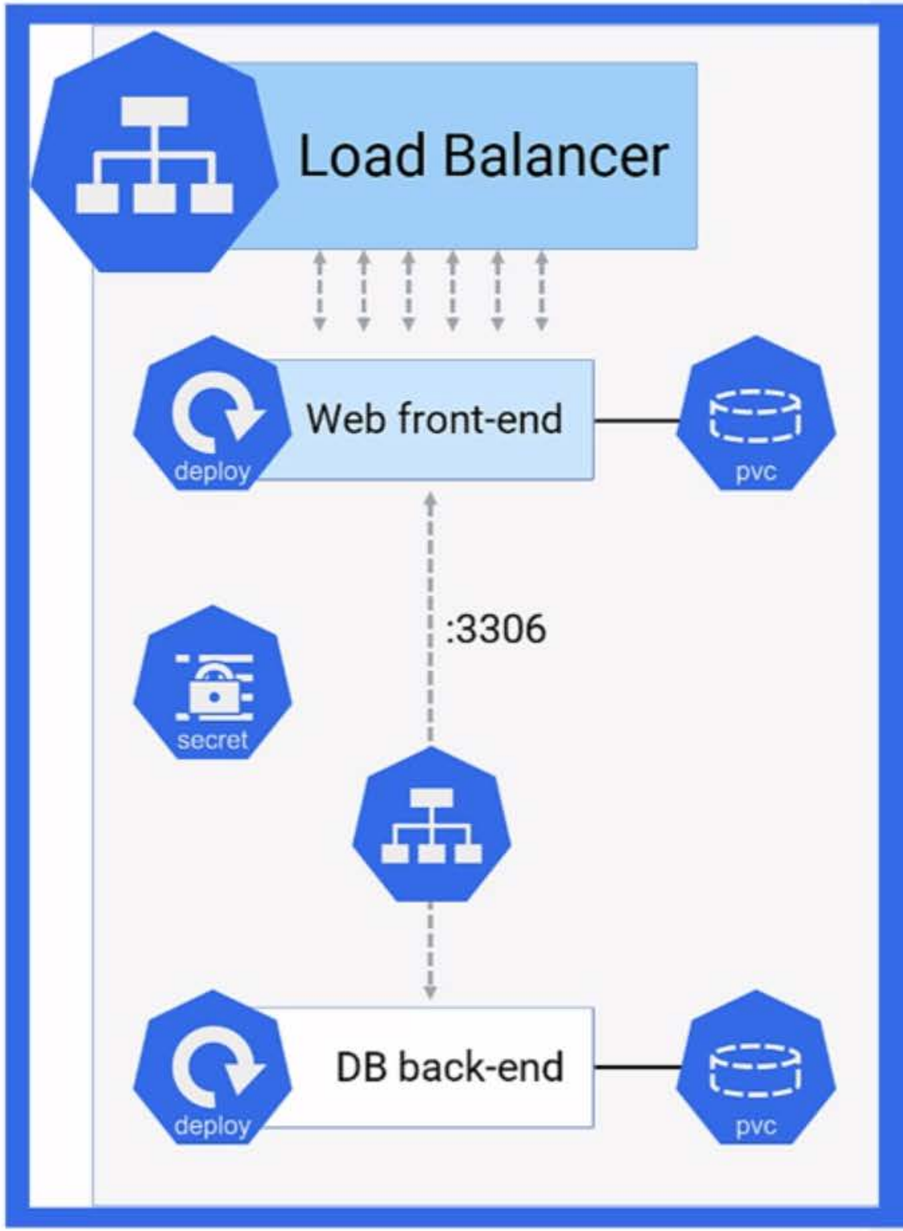
Kubernetes App Theory



Kubernetes App Theory



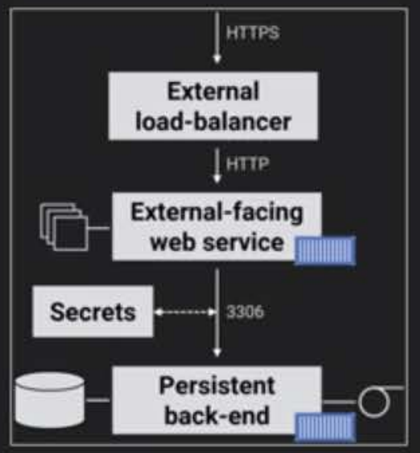
```yaml
apiVersion: v1
kind: Service
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  ports:
    - port: 80
  selector:
    app: wordpress
    tier: frontend
  type: LoadBalancer
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
```



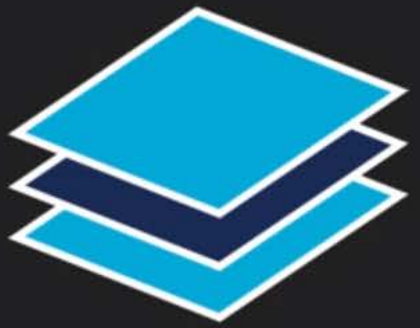
Recap

Ideas

Requirements




```
package main
import (
    "fmt"
    "reflect"
)
func main() {…
```



Kubernetes Networking Basics

House Rules

All Nodes can talk

All Pods can talk

(No NAT)

Every Pod gets its own IP

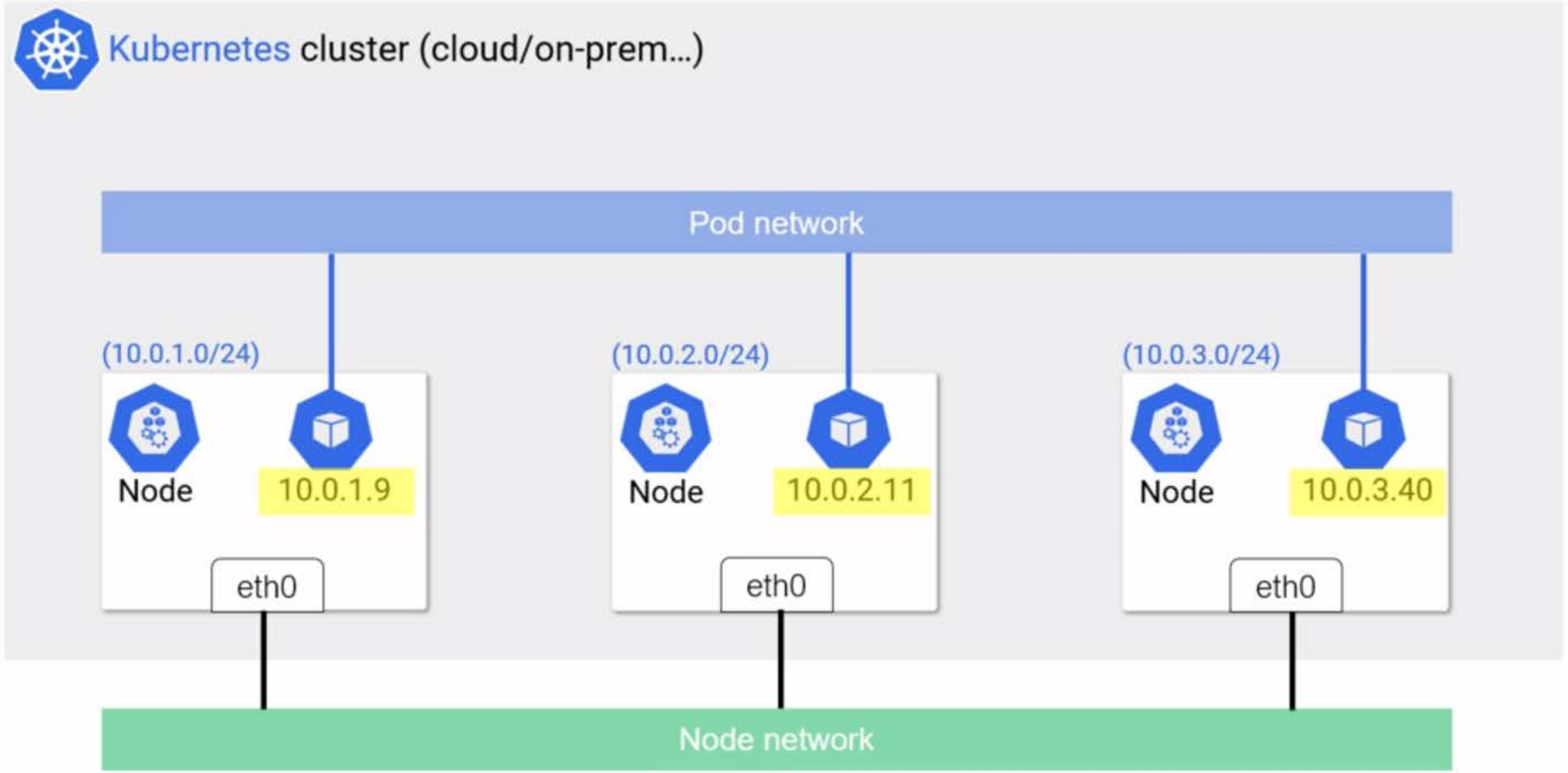
Kubernetes Networking Basics

Pod network

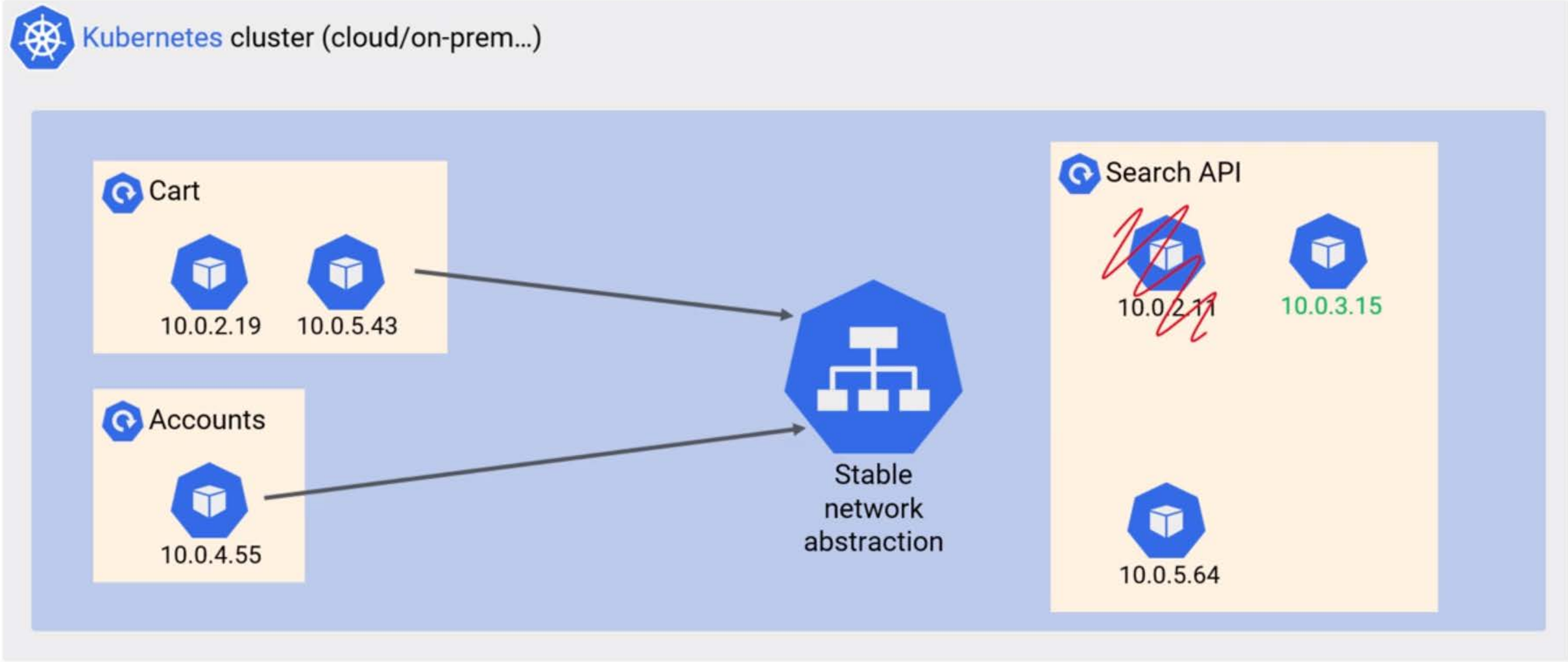
CNI plugin
- Big flat network

Node network

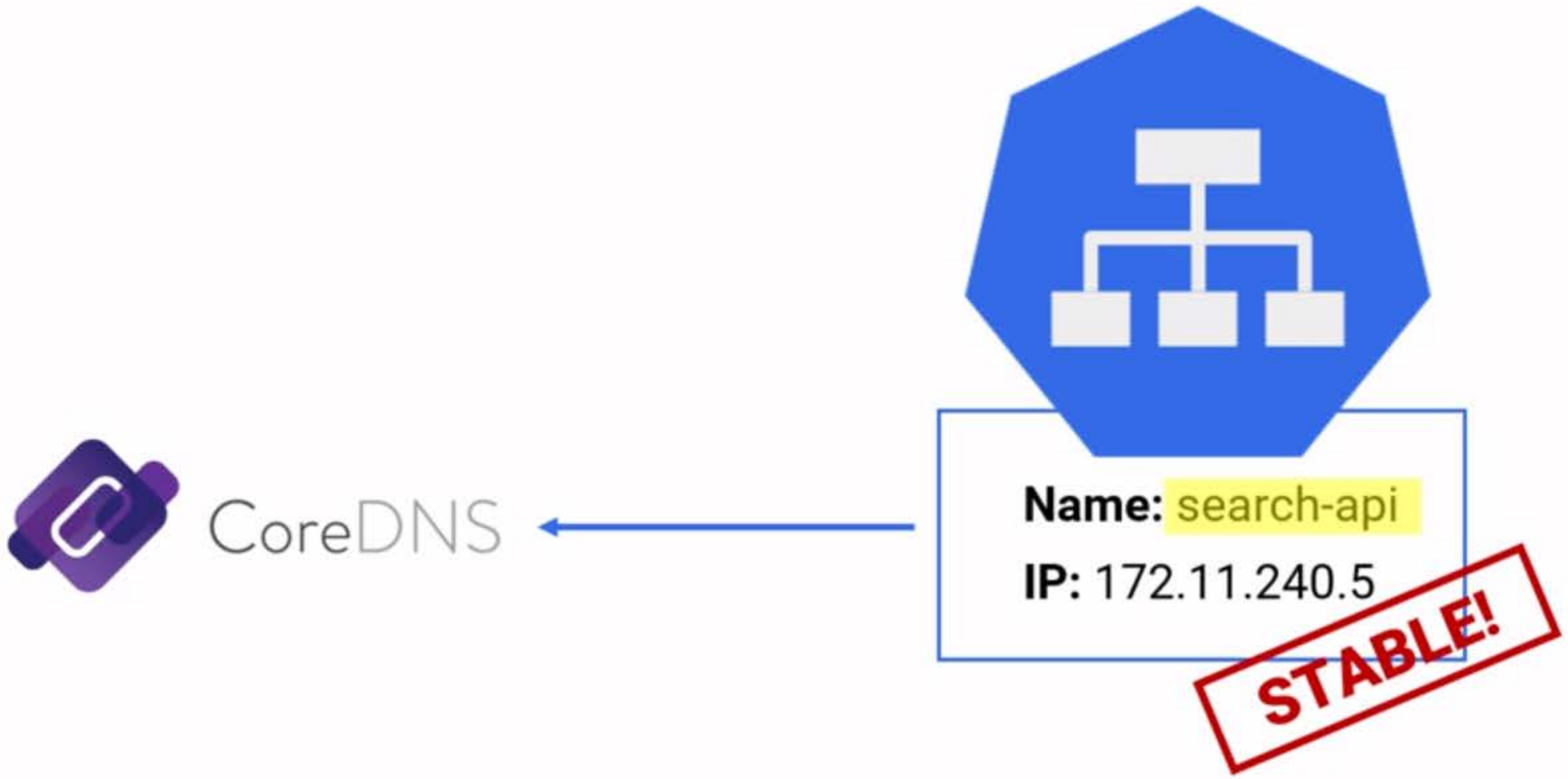
- 443 (HTTPS)
- Pod networking



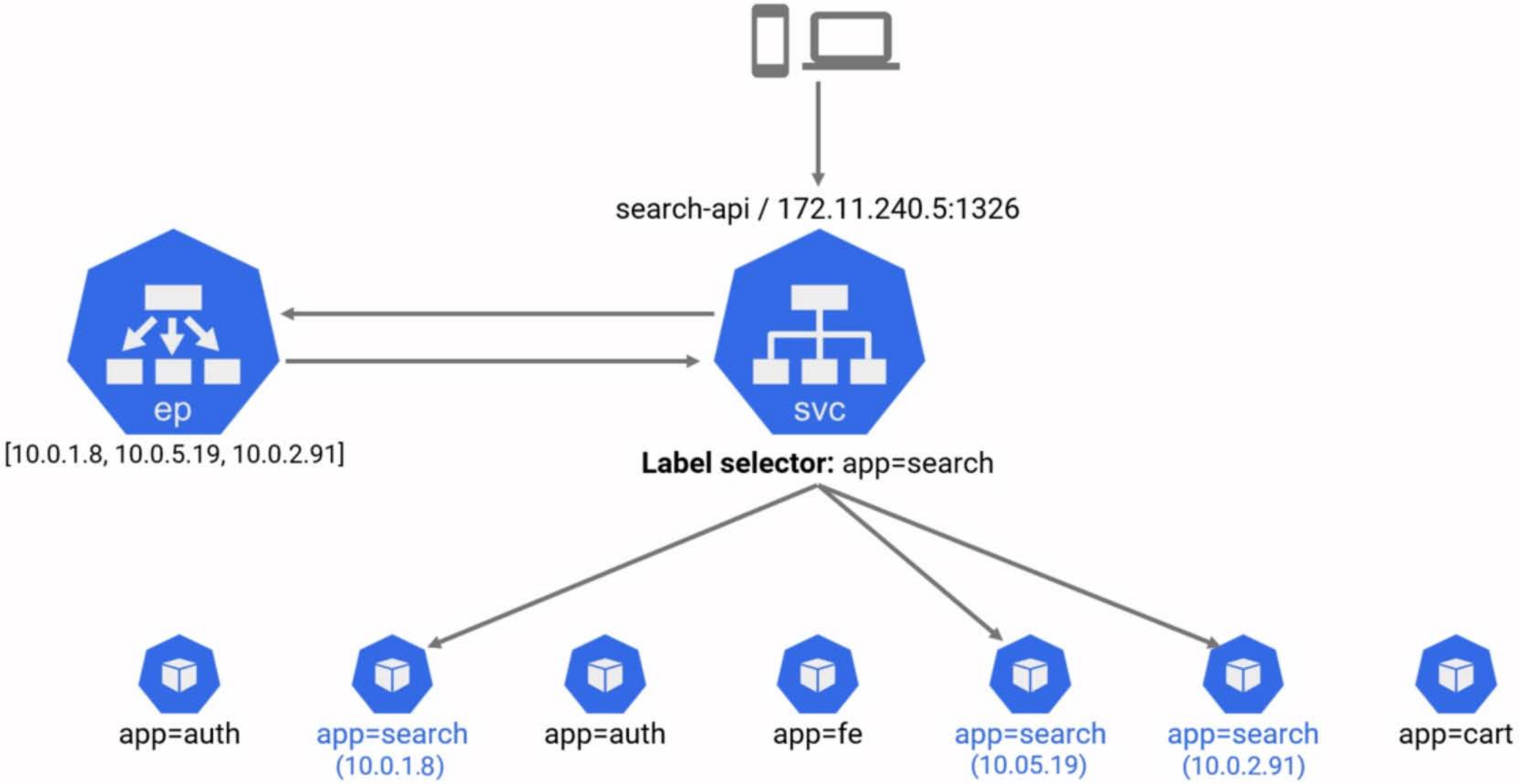
Kubernetes Services



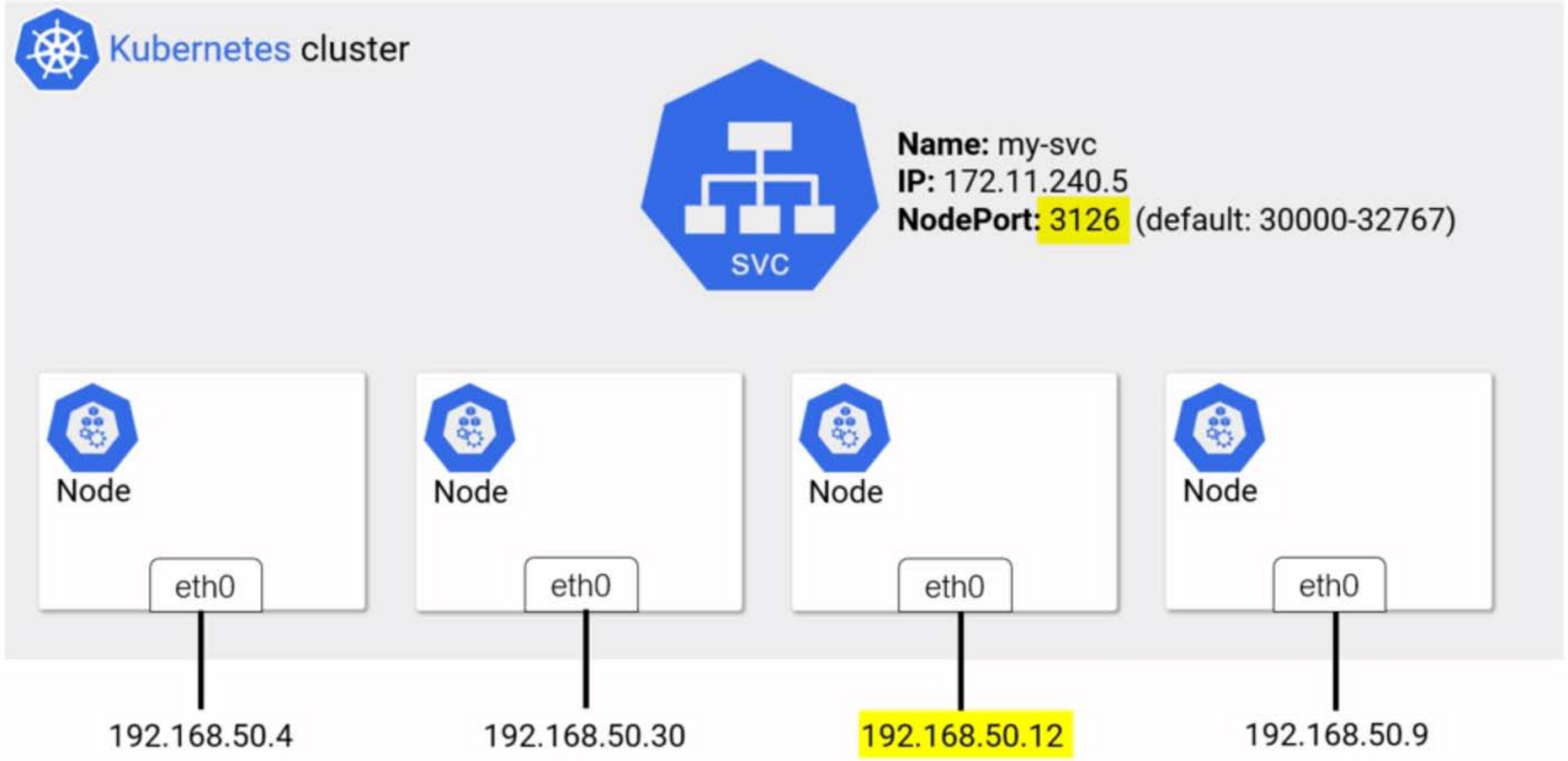
Kubernetes Services



Kubernetes Services



Kubernetes Networking Basics

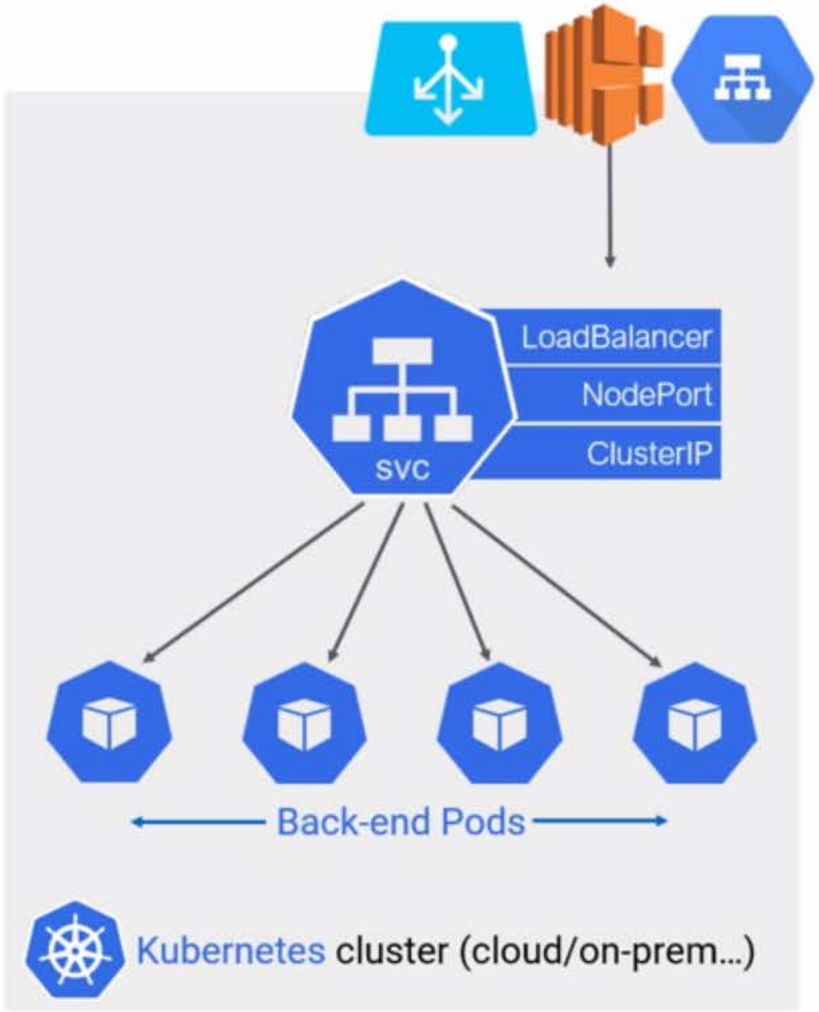


Kubernetes Services

LoadBalancer:
- Integrates with public cloud platform

NodePort:
- Gets cluster-wide **port**
- Also accessible from outside of cluster

ClusterIP (default):
- Gets own IP
- Only accessible from within cluster



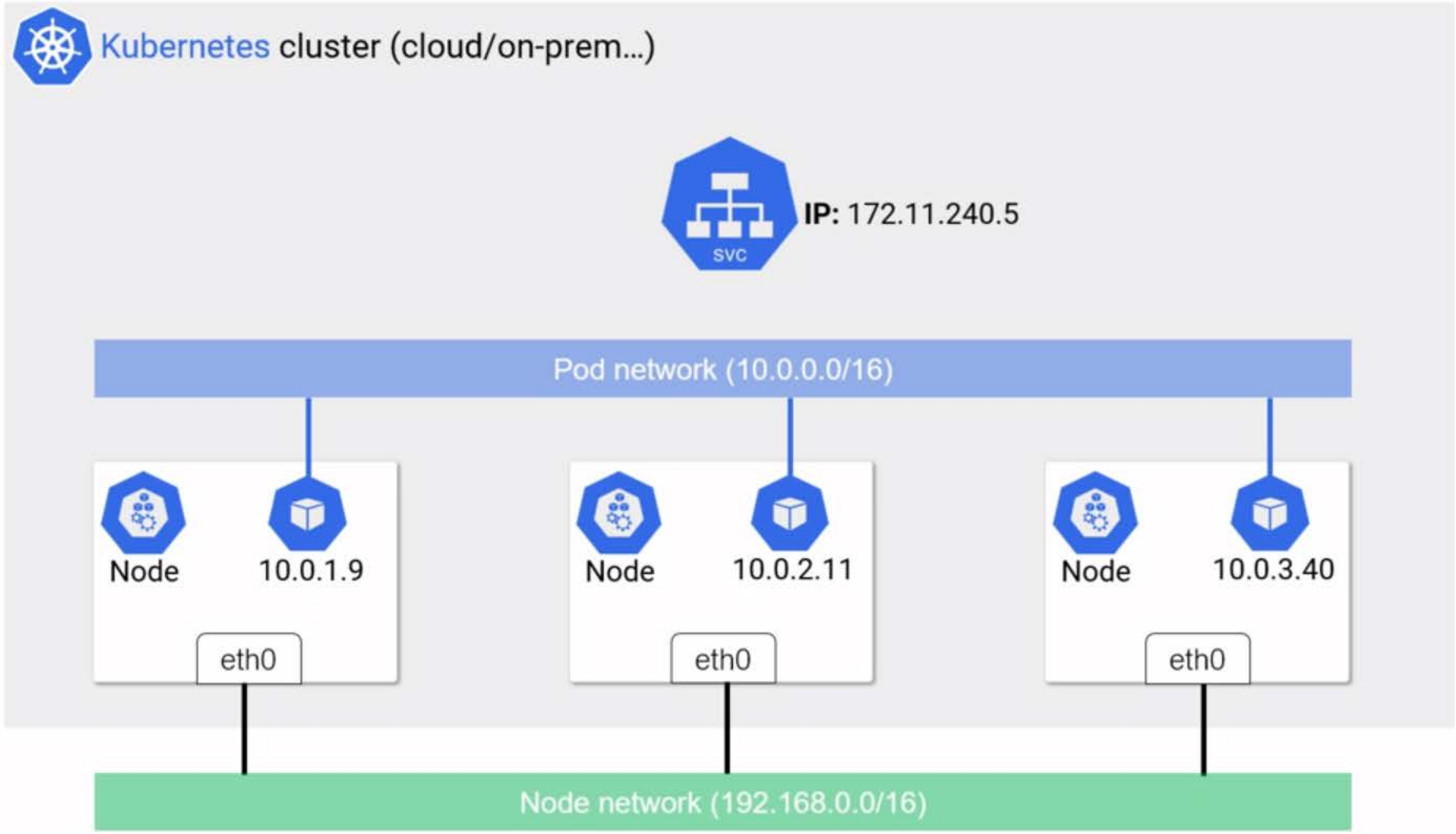
The Service Network

Pod network
- 10.0.0.0/16

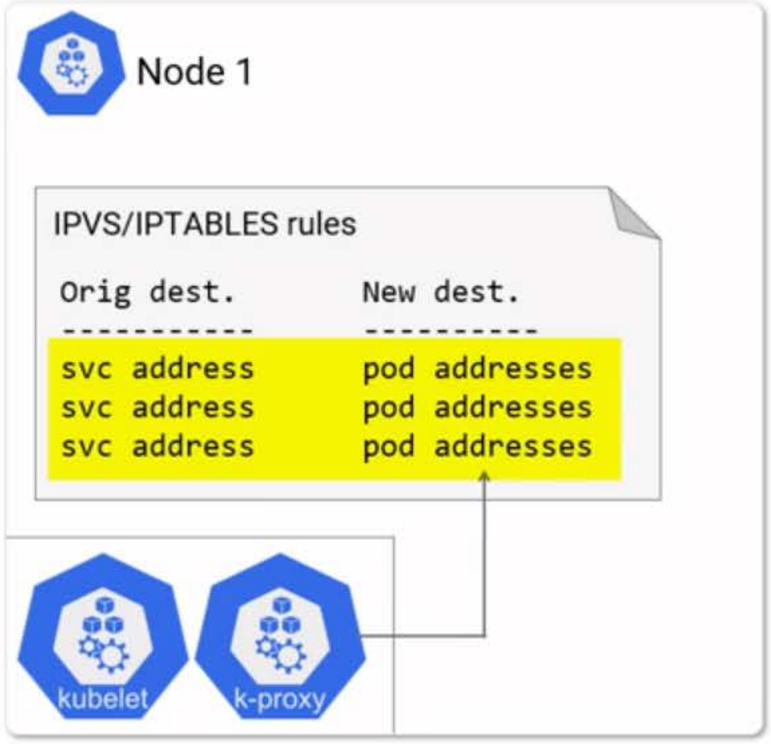
Node network
- 192.168.0.0/16

Service network
- 172.11.11.0/24

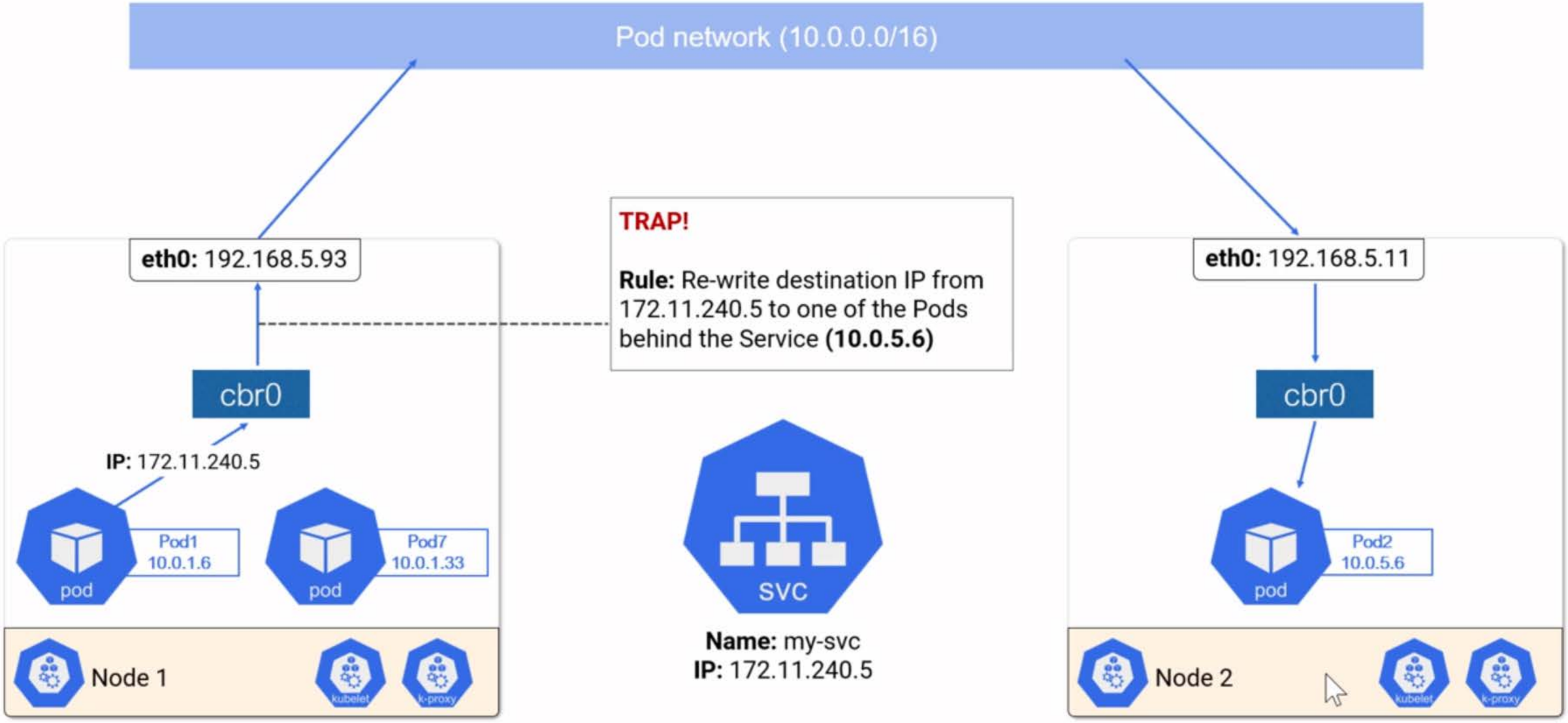
* Example network addresses for maximum clarity. Not the defaults that Kubernetes uses.



The Service Network



The Service Network



The Service Network

Kube-proxy IPTABLES Mode

- Default since Kubernetes 1.2
- Doesn't scale well
- Not really designed for load balancing

Kube-proxy IPVS Mode

- Stable (GA) since Kubernetes 1.11
- Uses Linux kernel IP Virtual Server
- Native Layer-4 load balancer
- Supports more algorithms

Recap

Node Network

All nodes need to be able to talk
- Kubelet ⟷ API Server
- Pod network ports

Not implemented by Kubernetes

Pod Network

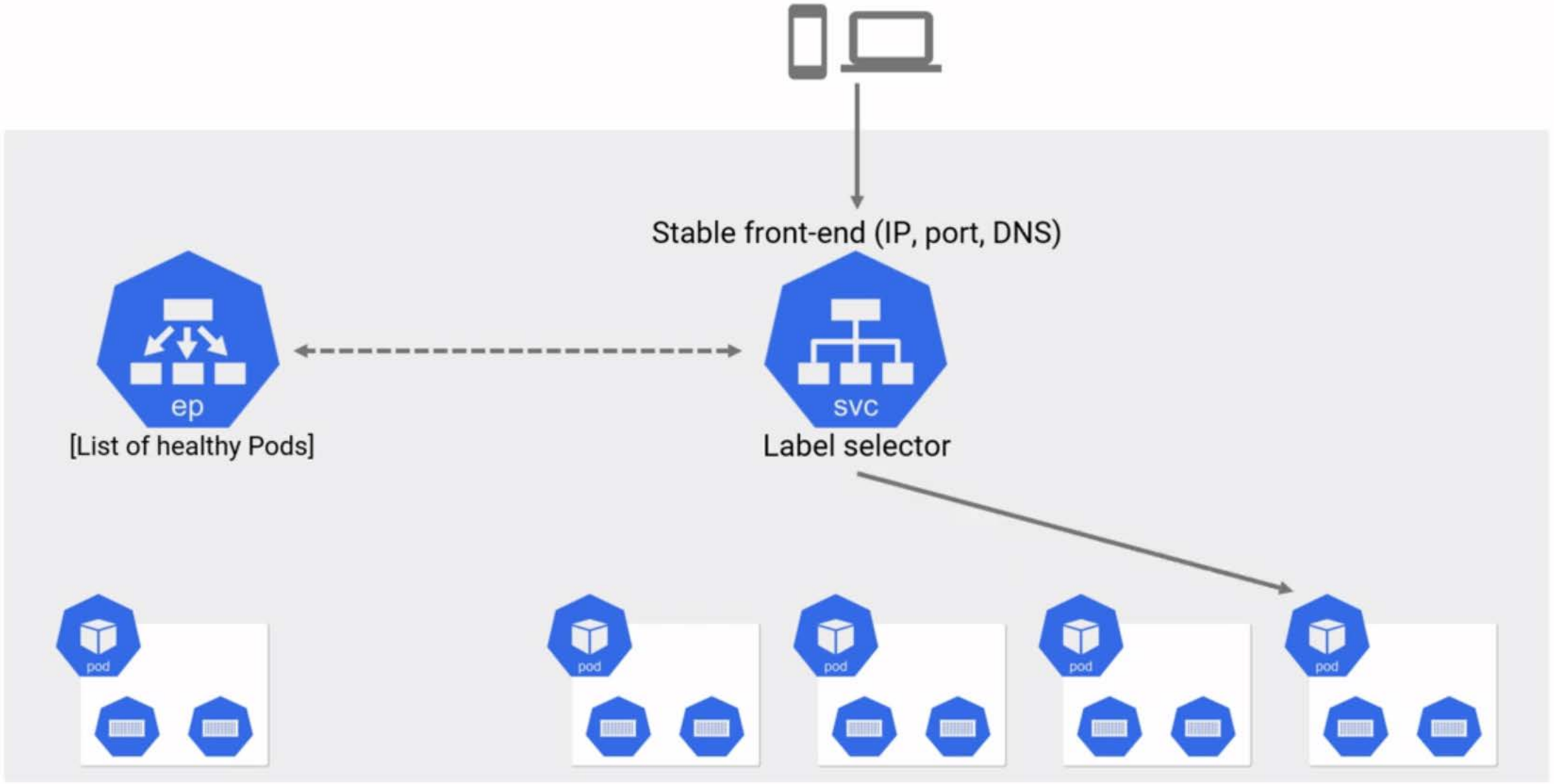
Implemented via CNI plugins

Big & flat

IP-per-Pod

All Pods can communicate

Recap


Stable front-end (IP, port, DNS)
ep
[List of healthy Pods]
svc
Label selector
pod
pod
pod
pod
pod

Recap

The Service Network

(It's not really a network)

```
$ kubectl describe svc nginx-service
Name:              my-svc
...
Type:              ClusterIP
IP:                10.11.240.5
Port:              http    3080/TCP
Endpoints:         10.244.0.44:8080,10.244.1.39:8080
Session Affinity:  None


$ ipvsadm -ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.11.240.5:3080  rr
  -> 10.244.0.44:8080             Masq    1      0          0
  -> 10.244.1.39:8080             Masq    1      0          0
```

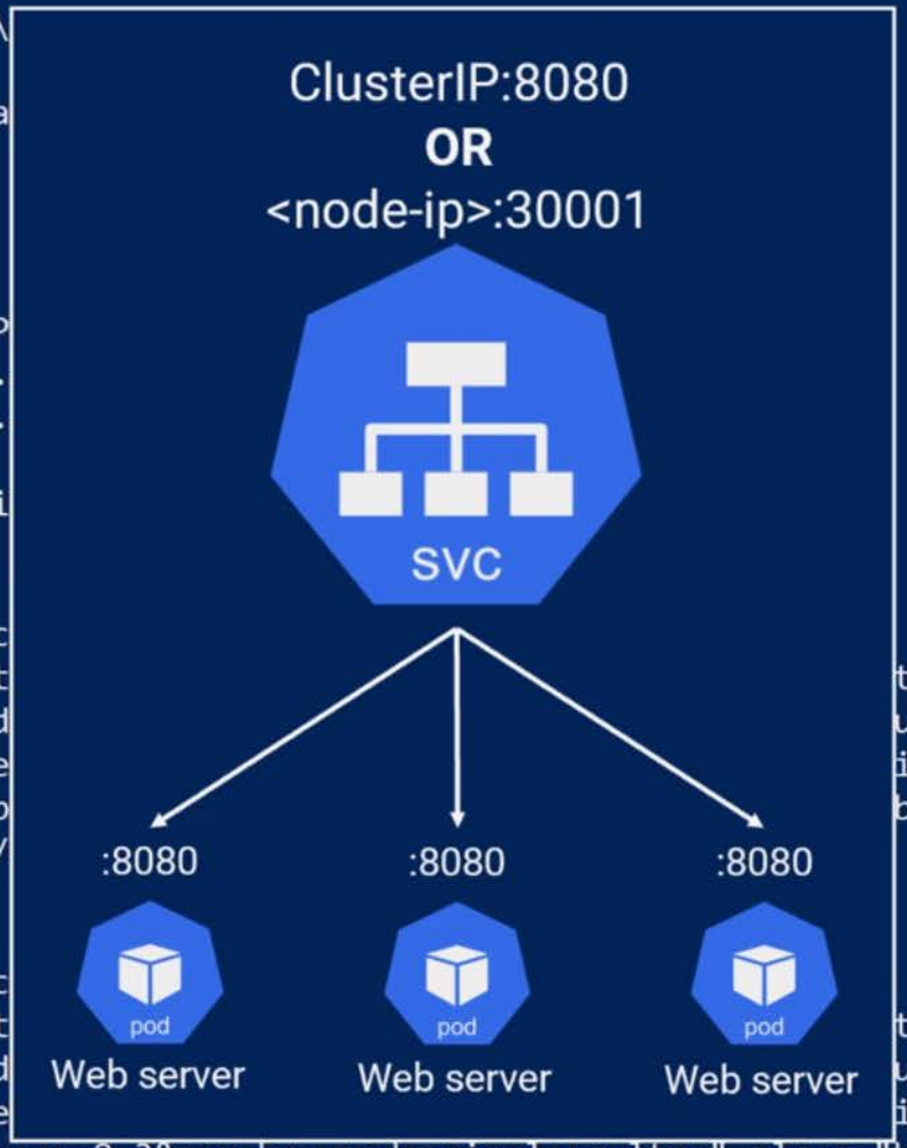
* Kube-proxy in **IPVS** mode does create dummy interfaces on the *Service Network* (usually called `kube-ipvs0`). Kube-proxy in IPTABLES mode does not.

root@pingtest-6769c69dd5-h2x76: /

```
PS C:\temp\k8s>
PS C:\temp\k8s> kubectl apply -f .\
service "hello-svc" created
deployment.apps "hello-deploy" crea
PS C:\temp\k8s>
PS C:\temp\k8s>
PS C:\temp\k8s>
PS C:\temp\k8s> kubectl get svc
NAME         TYPE        CLUSTER-IP
hello-svc    NodePort    10.15.244.
kubernetes   ClusterIP   10.15.240.
PS C:\temp\k8s>
PS C:\temp\k8s> kubectl exec -it pi
root@pingtest-6769c69dd5-h2x76:/#
root@pingtest-6769c69dd5-h2x76:/#
root@pingtest-6769c69dd5-h2x76:/# c
<html><head><title>ACG loves K8S</t
.1.1/css/bootstrap.min.css"/></head
Kubernetes!!!</h1><p></p><p> <a hre
_1_3?ie=UTF8&amp;qid=1531240306&amp
/a></p><p></p></div></div></body></
root@pingtest-6769c69dd5-h2x76:/#
root@pingtest-6769c69dd5-h2x76:/#
root@pingtest-6769c69dd5-h2x76:/# c
<html><head><title>ACG loves K8S</t
.1.1/css/bootstrap.min.css"/></head
Kubernetes!!!</h1><p></p><p> <a hre
_1_3?ie=UTF8&amp;qid=1531240306&amp
/a></p><p></p></div></div></body></
```

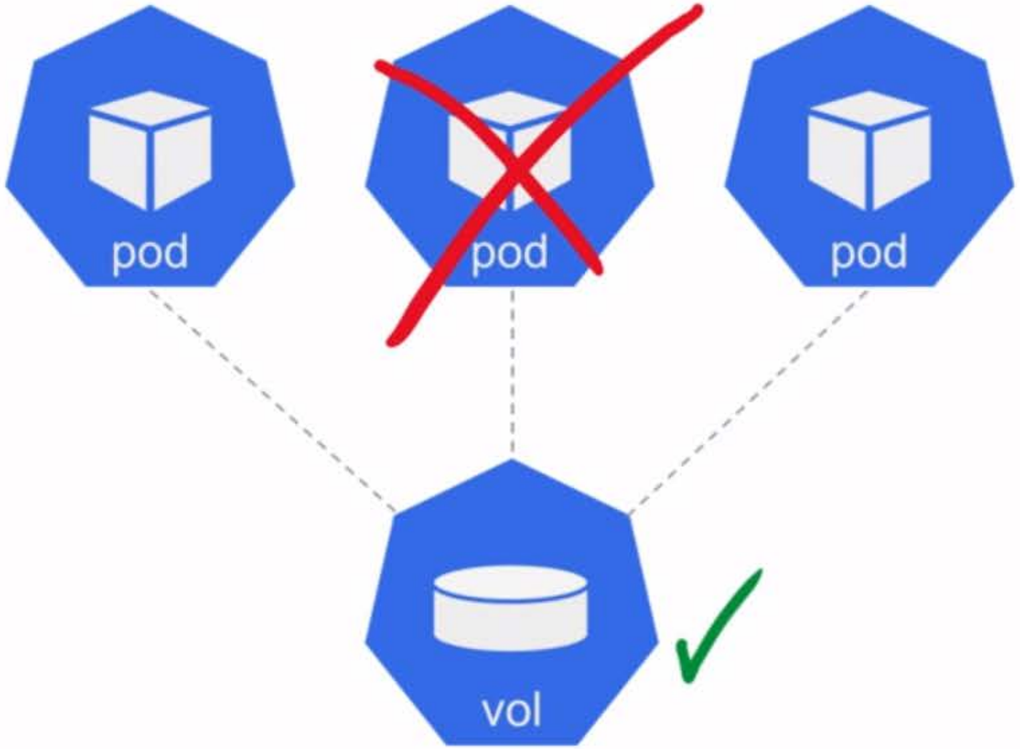
tdna.bootstrapcdn.com/bootstrap/3
umbotron"><h1>A Cloud Guru loves
igel-Poulton/dp/1521823634/ref=sr
btn btn-primary">Kubernetes Book<

tdna.bootstrapcdn.com/bootstrap/3
umbotron"><h1>A Cloud Guru loves
igel-Poulton/dp/1521823634/ref=sr
btn btn-primary">Kubernetes Book<



in the cluster plus the NodePort value.

High-level Storage Requirements



High-level Storage Requirements

A CLOUD GURU

Storage is Vital!

File & Block
First-class Citizens in
Kubernetes

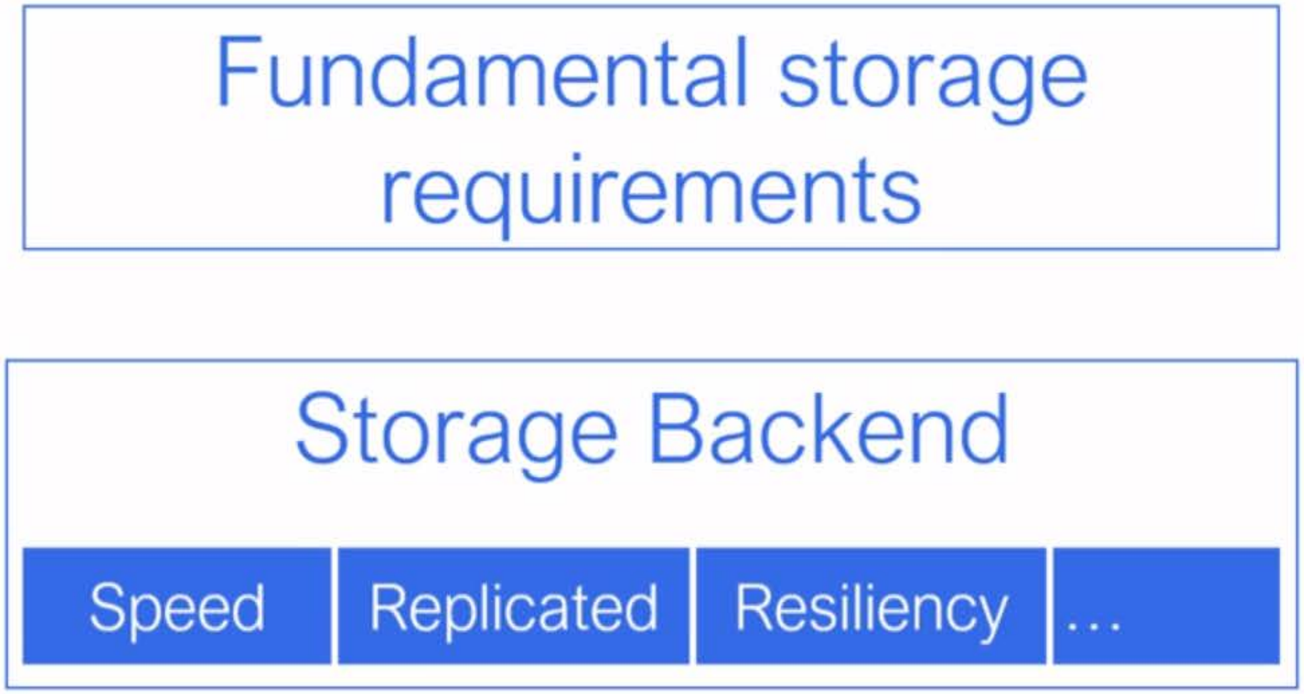
High-level Storage Requirements

A CLOUD GURU

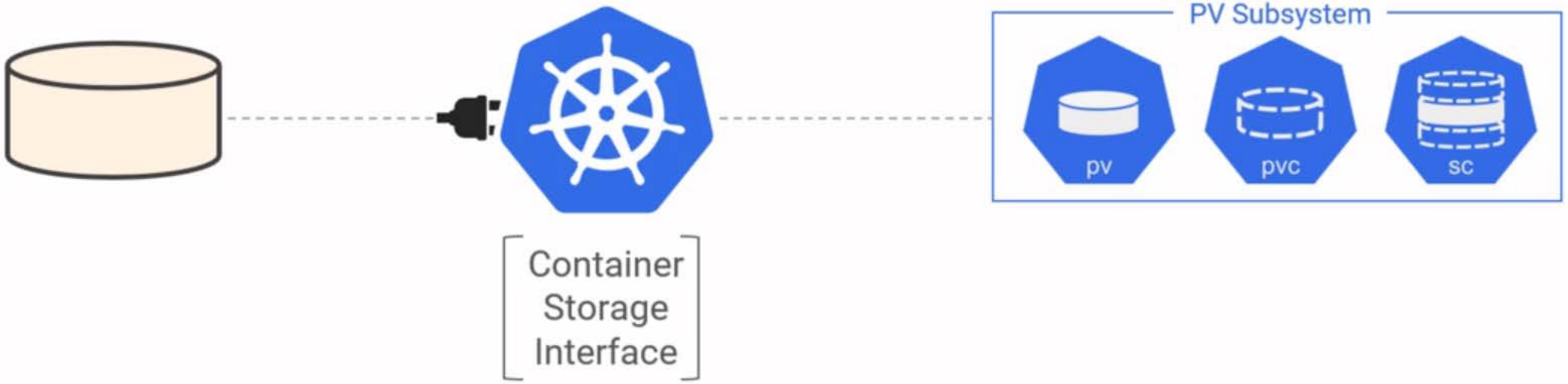
File & Block
First-class Citizens in
Kubernetes

- Standards-based
- Pluggable backend
- Rich API

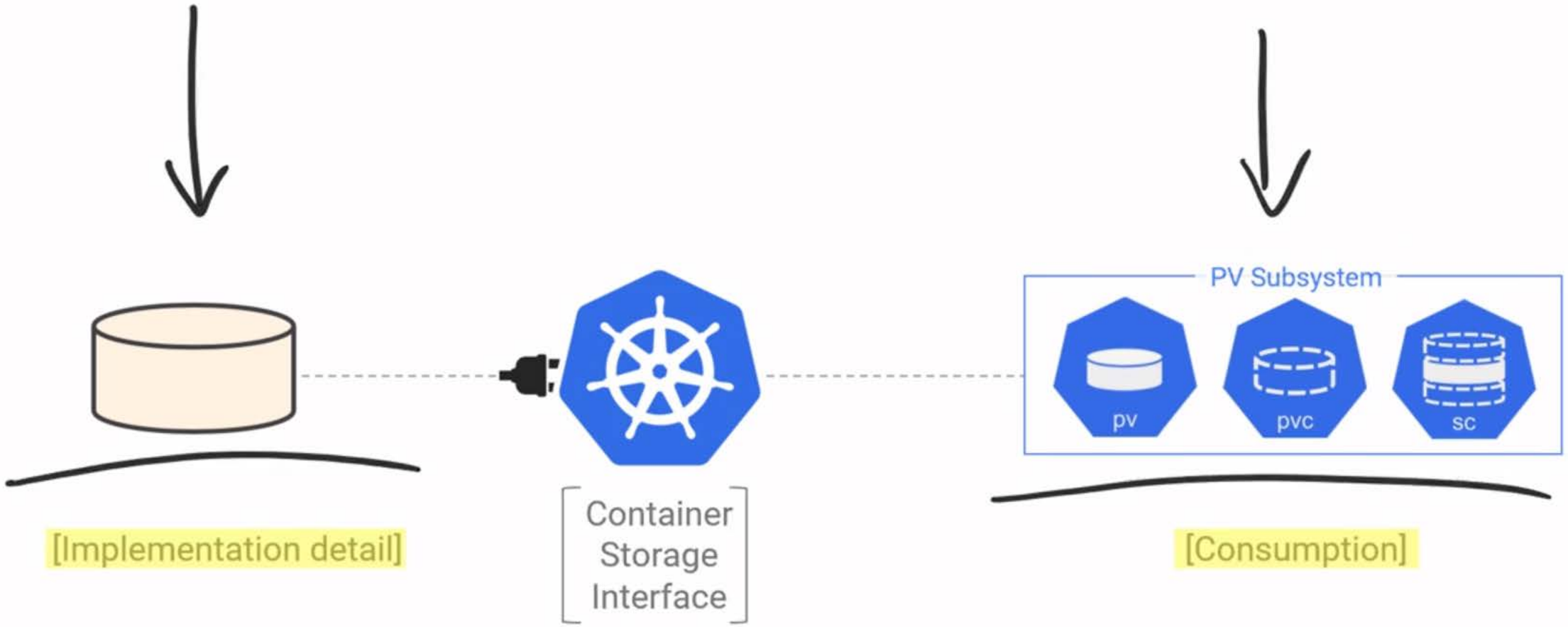
High-level Storage Requirements



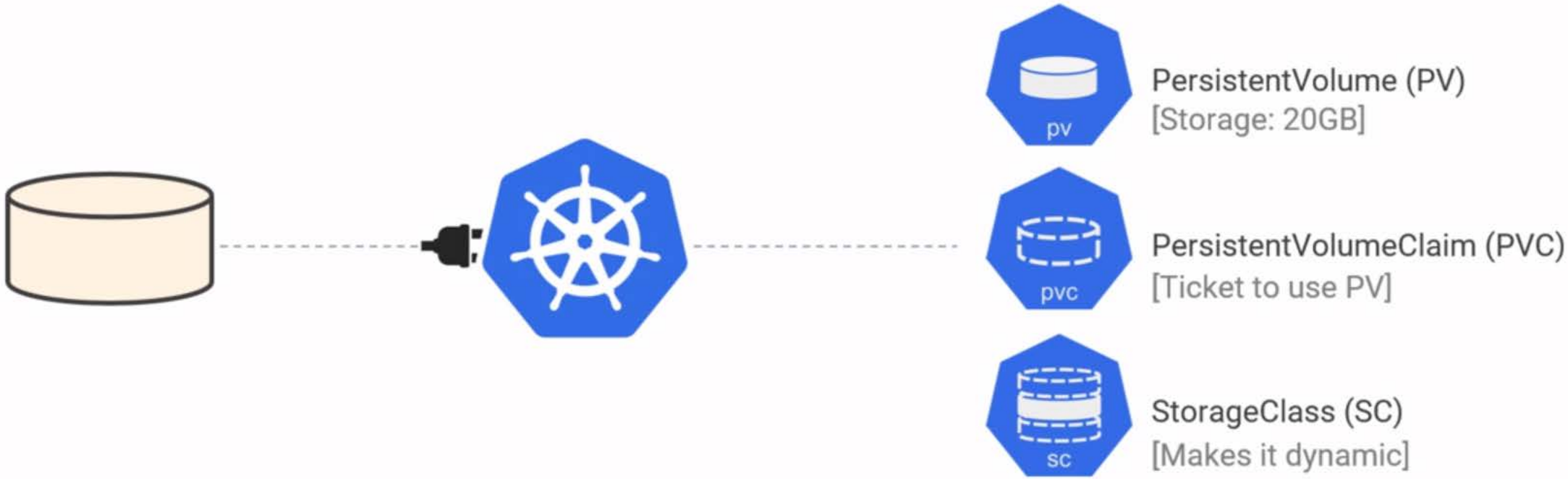
High-level Storage Requirements



High-level Storage Requirements



High-level Storage Requirements



Container Storage Interface (CSI)

@nigelpoulton

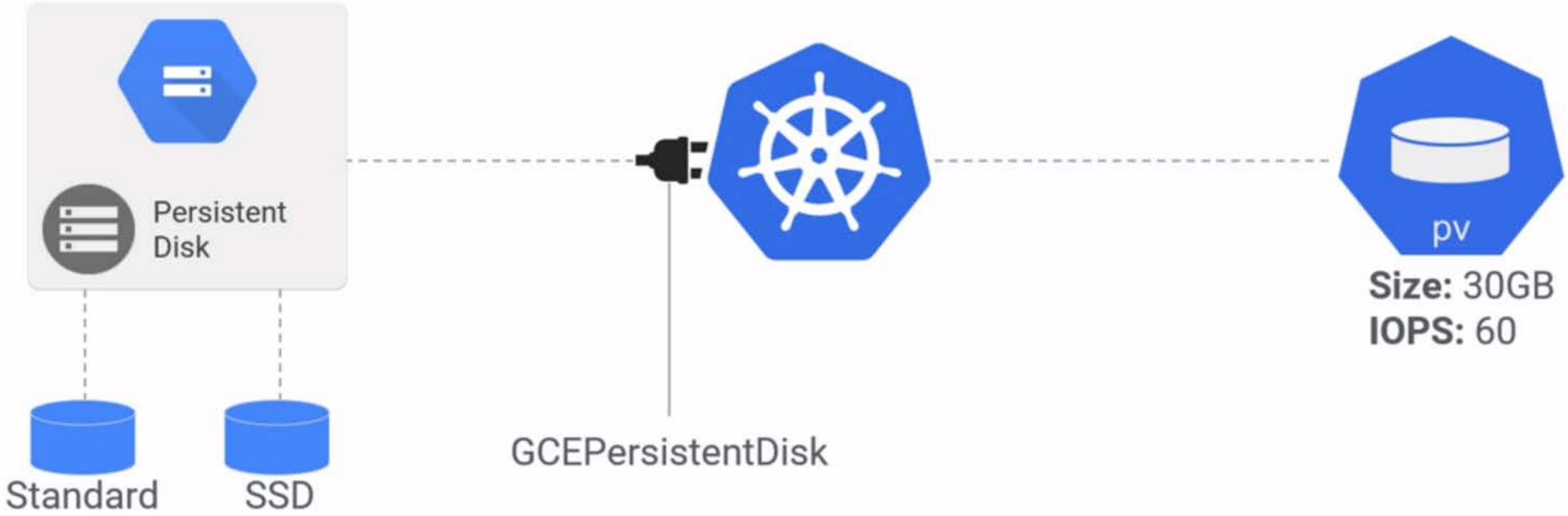
Out-of-tree

Open-standard

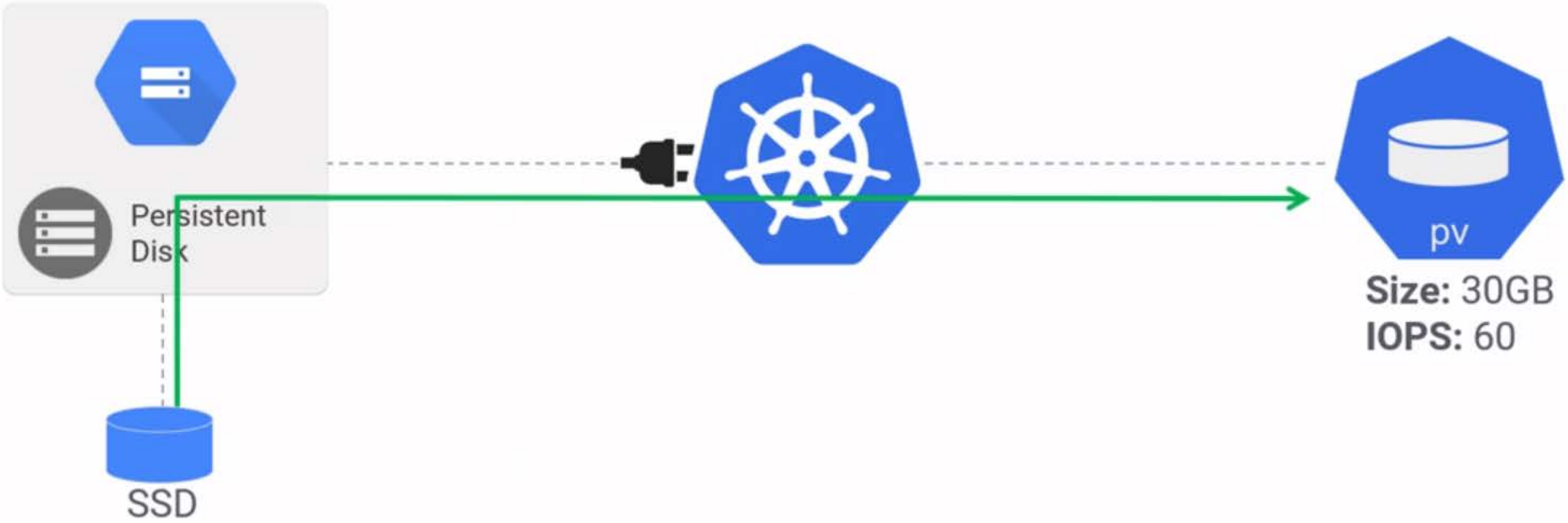
https://github.com/container-storage-interface/spec

Still Beta in K8s 1.11

The K8s PersistentVolume Subsystem



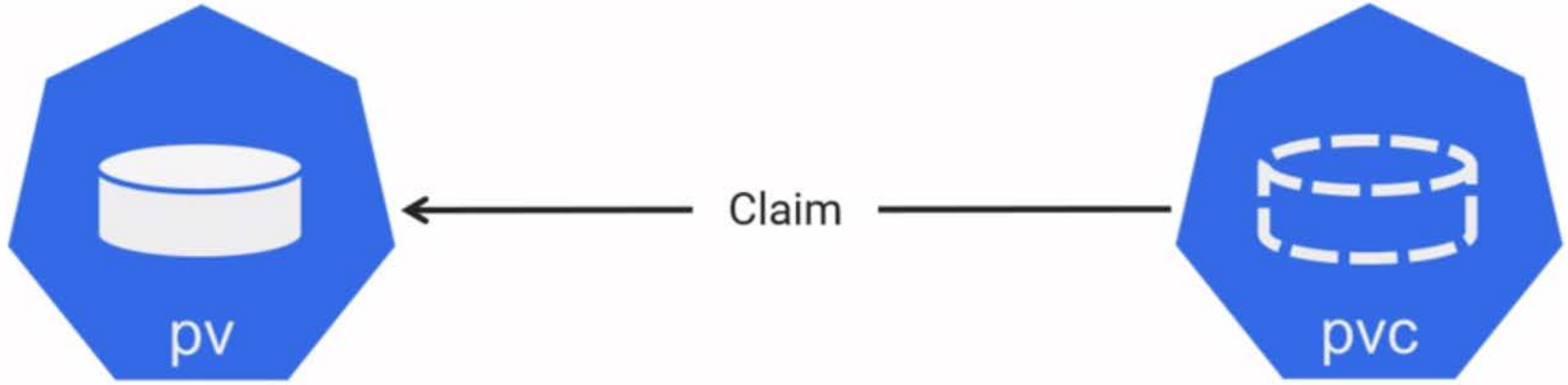
The K8s PersistentVolume Subsystem



The K8s PersistentVolume Subsystem


```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv1
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: ssd
  capacity:
    storage: 10Gi
...
```

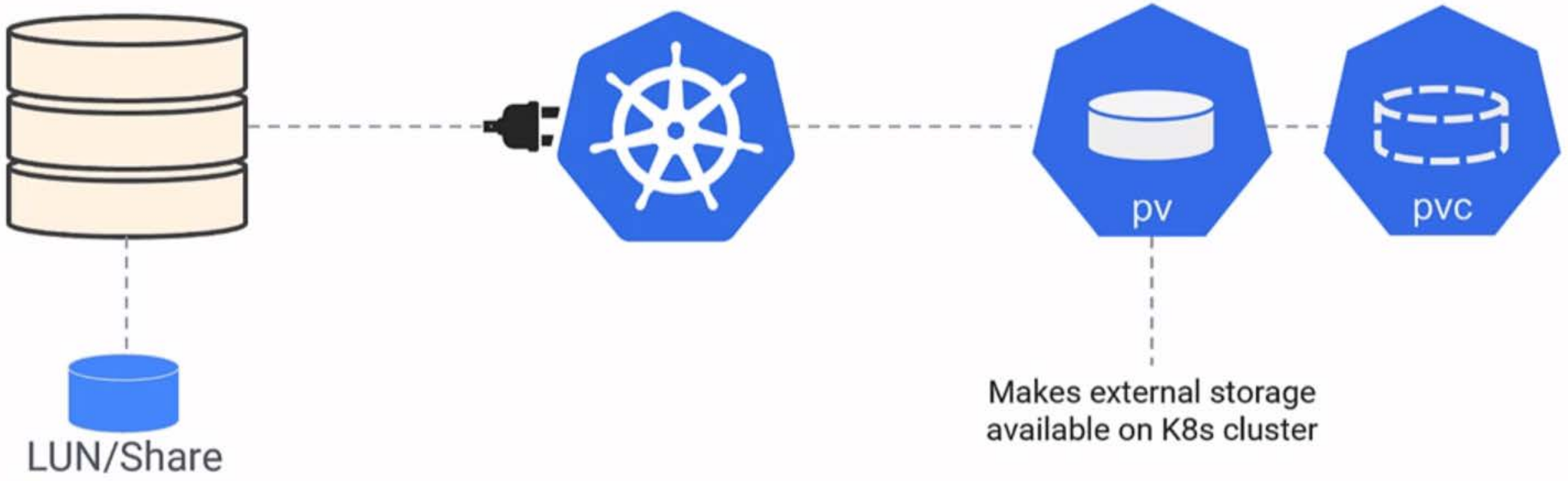
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc1
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: ssd
  resources:
    requests:
      storage: 10Gi
```



Size: 30GB
IOPS: 60

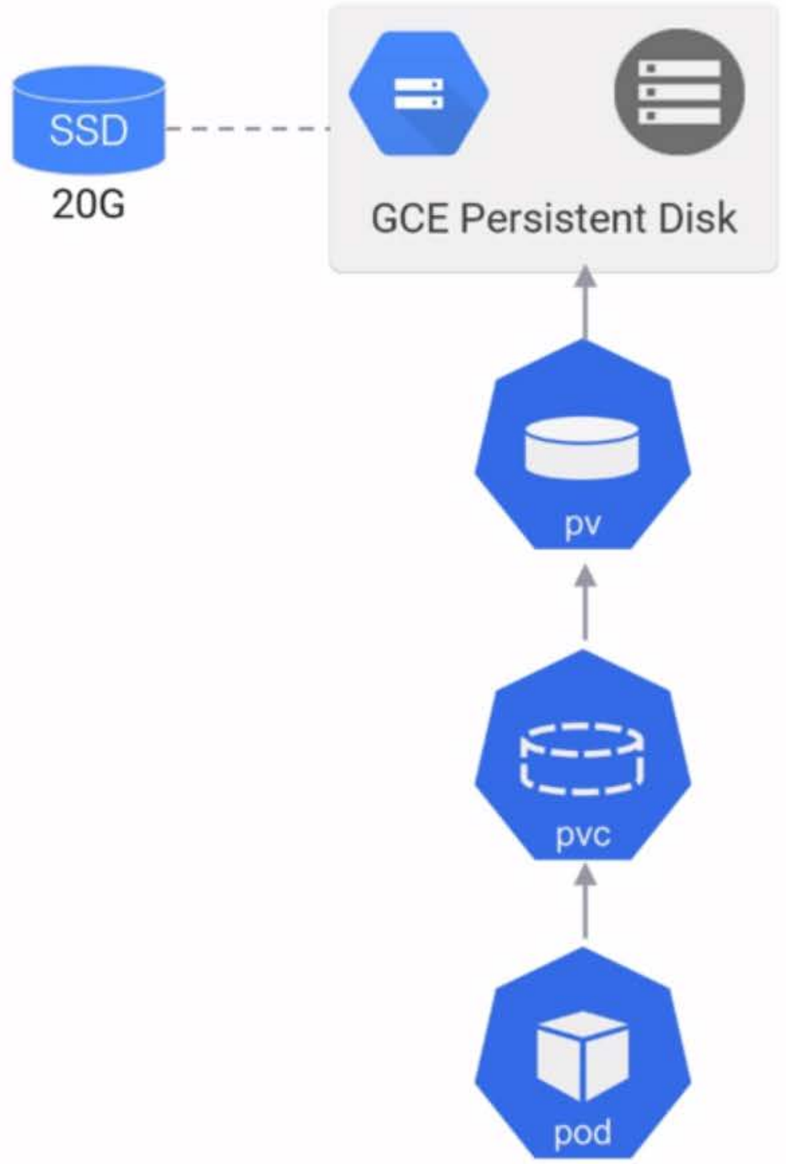
gke-pv.yml ● | gke-pvc.yml ✕ | gke-volpod.yml

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc1
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: ssd
  resources:
    requests:
      storage: 20Gi
```



gke-pv.yml ● | gke-pvc.yml | gke-volpod.yml ×

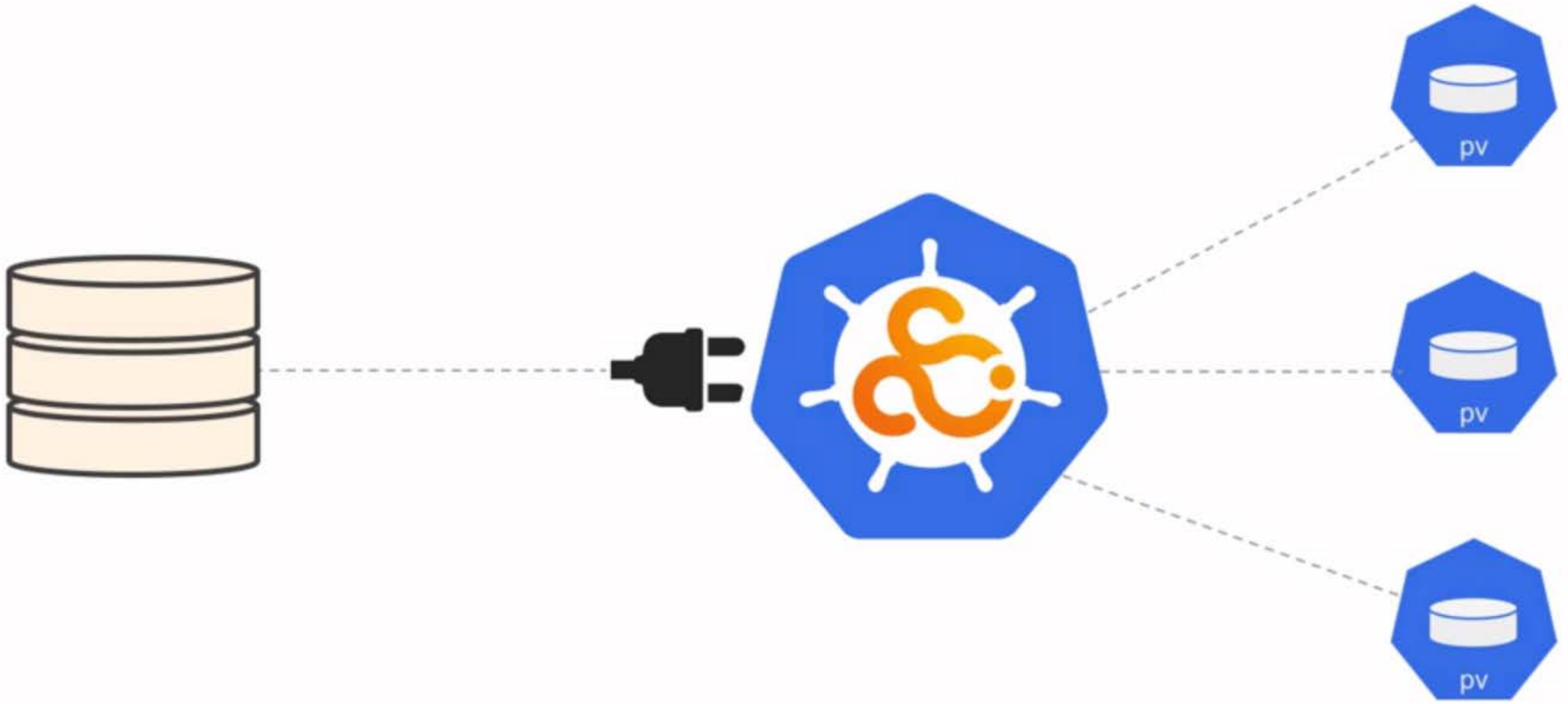
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: volpod
spec:
  volumes:
    - name: data
      persistentVolumeClaim:
        claimName: pvc1
  containers:
  - image: ubuntu:latest
    name: ubuntu-ctr
    command:
    - /bin/bash
    - "-c"
    - "sleep 60m"
    imagePullPolicy: IfNotPresent
    volumeMounts:
    - mountPath: /data
      name: data
```



ubuntu@mgr1: ~

```
ubuntu@mgr1:~$
ubuntu@mgr1:~$ cat aws-pvc.yml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: aws-fast
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: fast
  resources:
    requests:
      storage: 10Giubuntu@mgr1:~$
ubuntu@mgr1:~$
ubuntu@mgr1:~$
ubuntu@mgr1:~$
ubuntu@mgr1:~$ kubectl apply -f aws-pvc.yml
persistentvolumeclaim/aws-fast created
ubuntu@mgr1:~$ kubectl get pvc
NAME       STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
aws-fast   Bound     pvc-3ec88edf-8ab6-11e8-8a01-060a650e58be   10Gi       RWO            fast           26s
ubuntu@mgr1:~$
ubuntu@mgr1:~$
ubuntu@mgr1:~$
ubuntu@mgr1:~$
ubuntu@mgr1:~$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS    CLAIM              STORAGECLASS
  REASON    AGE
pvc-3ec88edf-8ab6-11e8-8a01-060a650e58be   10Gi       RWO            Delete           Bound     default/aws-fast   fast
          1m
ubuntu@mgr1:~$
```

Recap



Recap

Making things dynamic!

The Big Picture

From code to Kubernetes

```
package main
import (
    "fmt"
    "reflect"
)
func main() {…
```

Coding

Source code
& dependencies

[Container registry]

Docker

Dockerfile
$ docker build…

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: acg-deploy
  labels:
    app: acg
spec:
  replicas: 3
  ...
  strategy:
  type: Recreate
  ...
```

Kubernetes

$ kubectl

Recap


```
package main
import (
    "fmt"
    "reflect"
)
func main() {…
```

```
LastWriteTime      Length Name
-------------      ------ ----
26/07/2018 12:41         views
26/07/2018 12:39    388  app.js
26/07/2018 20:07    508  Dockerfile
26/07/2018 12:36    345  package.json
26/07/2018 12:30    360  readme.md
26/07/2018 21:40    407  web-deploy.yml
26/07/2018 21:40    184  web-lb.yml
26/07/2018 12:22    197  web-nodeport.yml
```

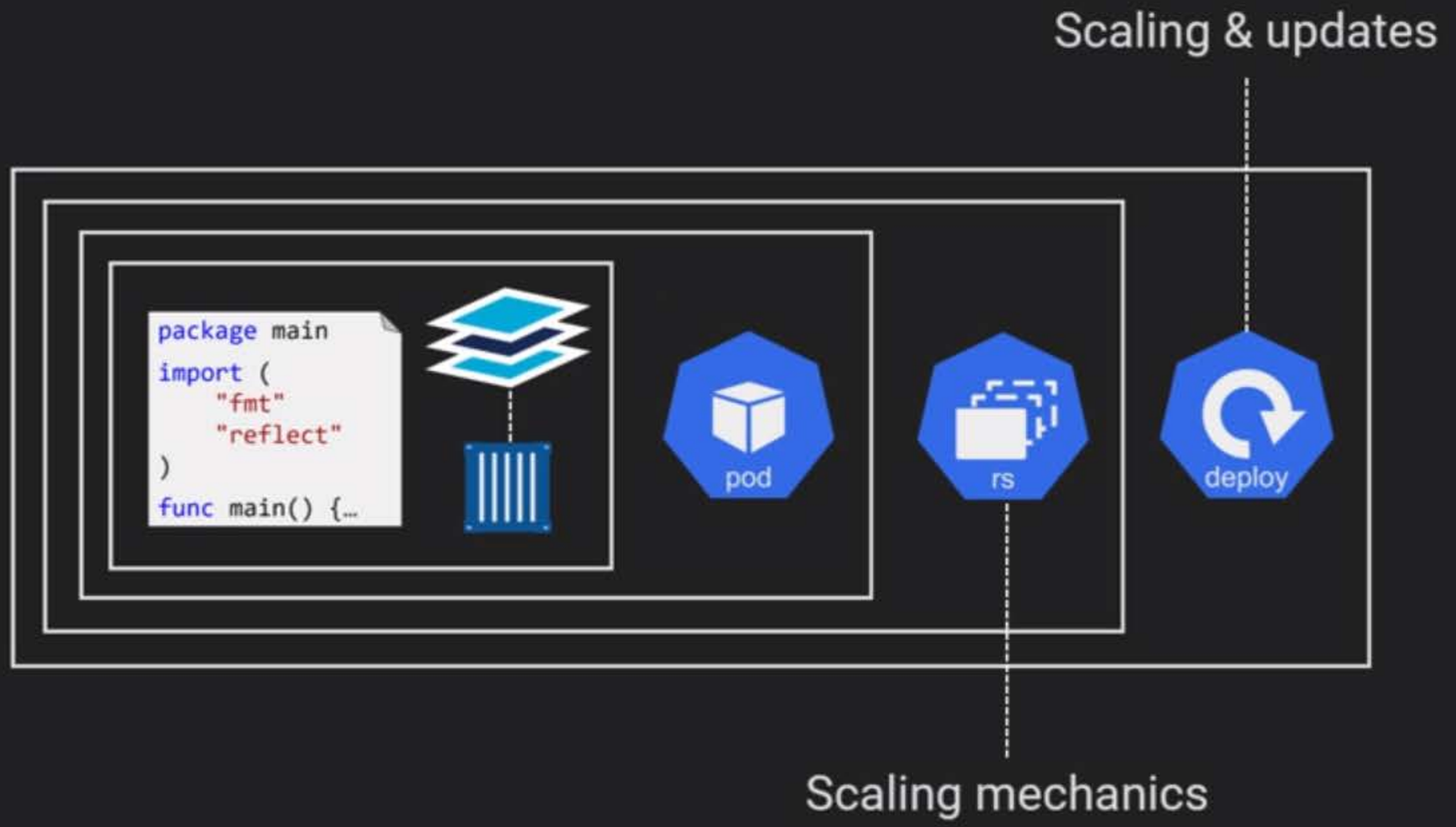
```
FROM centos:centos7
RUN yum -y install nodejs np
COPY . /src
RUN cd /src; npm install
EXPOSE 8080
CMD cd /src && node ./app.js
```

$ docker build…

[Container registry]

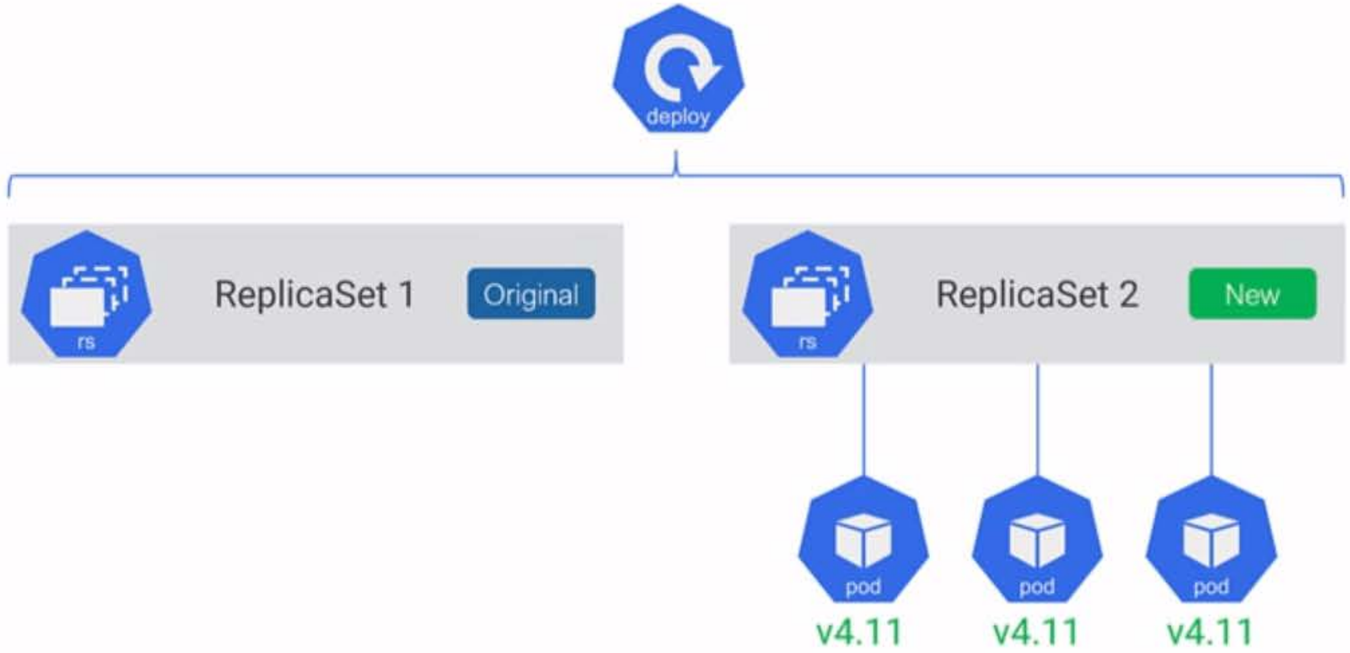
$ kubectl apply -f ...

Deployment Theory



deployment.yml

```yaml
  selector:
    matchLabels:
      run: prod-redis
  replicas: 3
  minReadySeconds: 300
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
  template:
    metadata:
      labels:
        run: prod-redis
    spec:
      containers:
      - image: redis:4.11
        name: redis
```



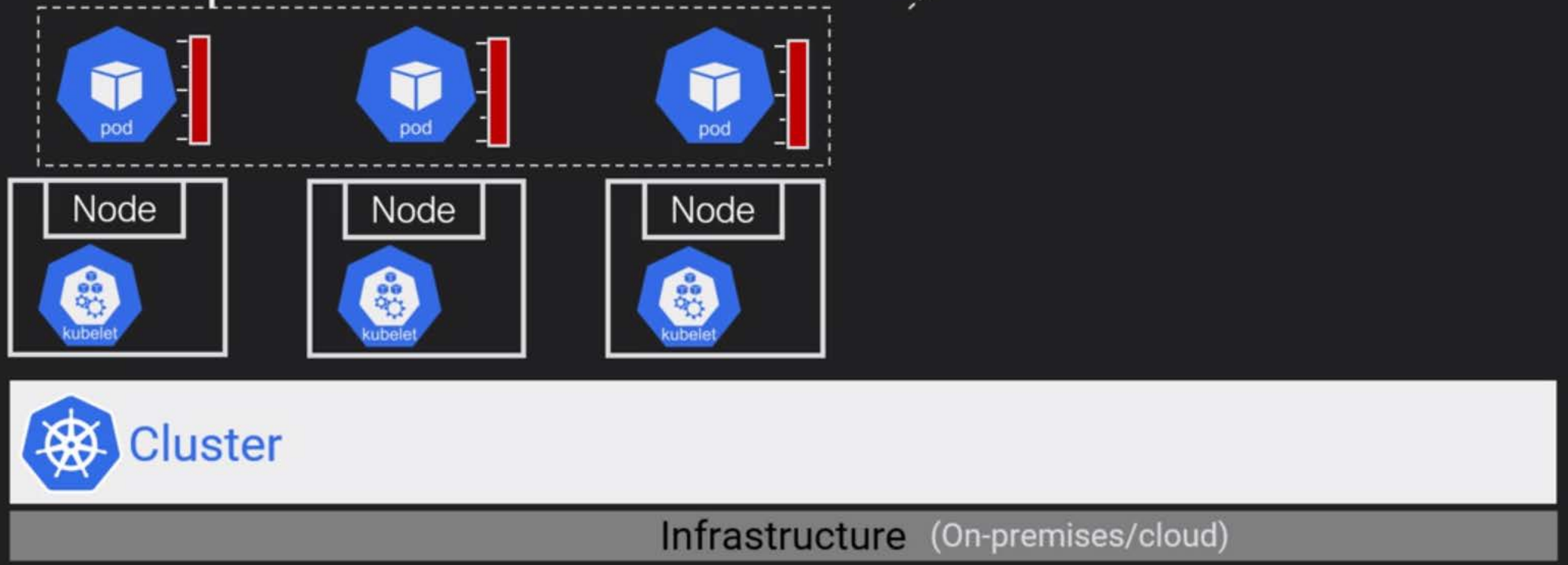
Big Picture

A CLOUD GURU

```
apiVersion: apps/v1
kind: Deployment
...
spec:
  replicas: 4
  ...
```

Scale to 4

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
...
spec:
  scaleTargetRef:
    name: my-deployment
    ...
```

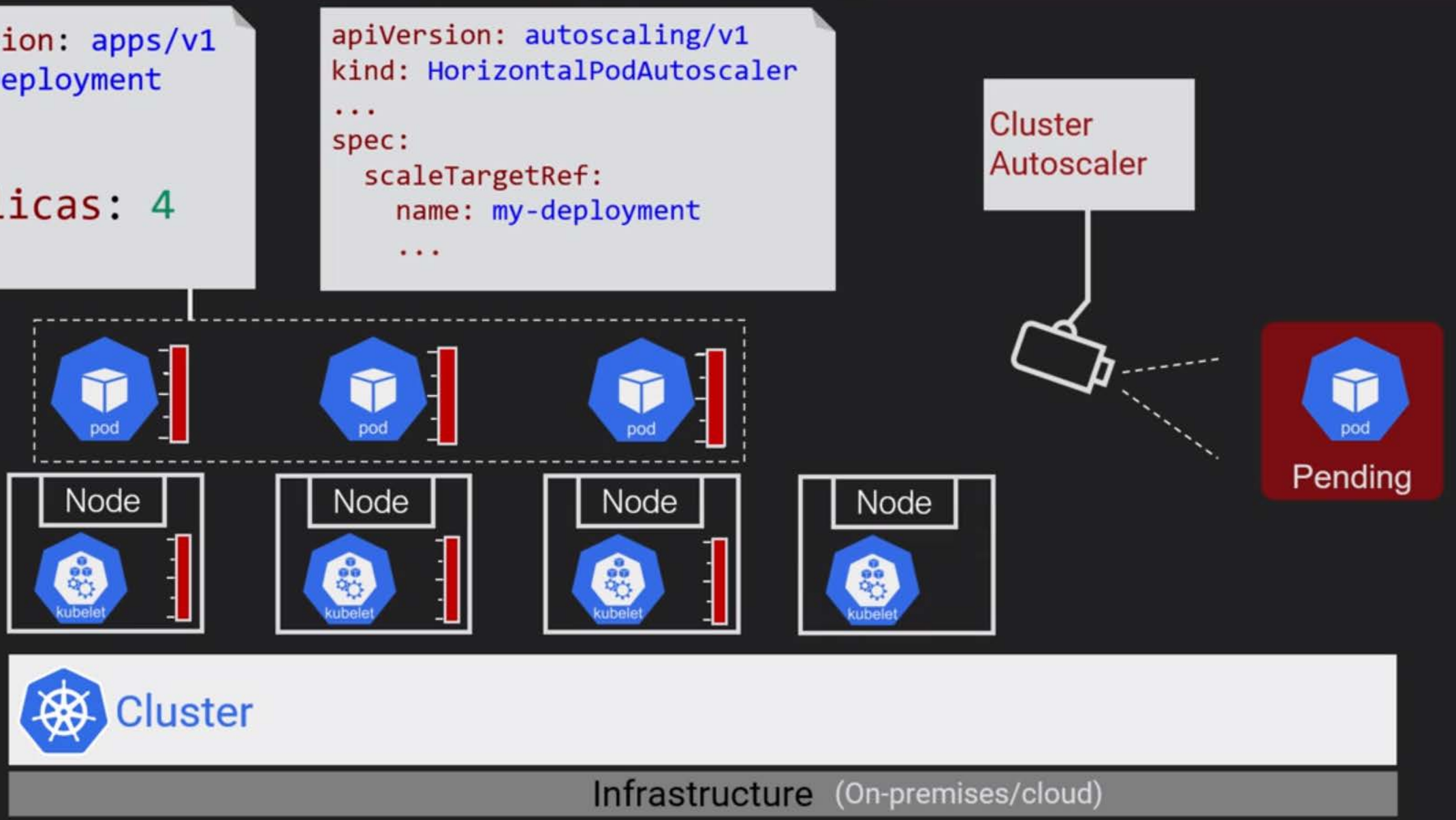


Big Picture

A CLOUD GURU

```
apiVersion: apps/v1
kind: Deployment
...
spec:
  replicas: 4
  ...
```

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
...
spec:
  scaleTargetRef:
    name: my-deployment
    ...
```

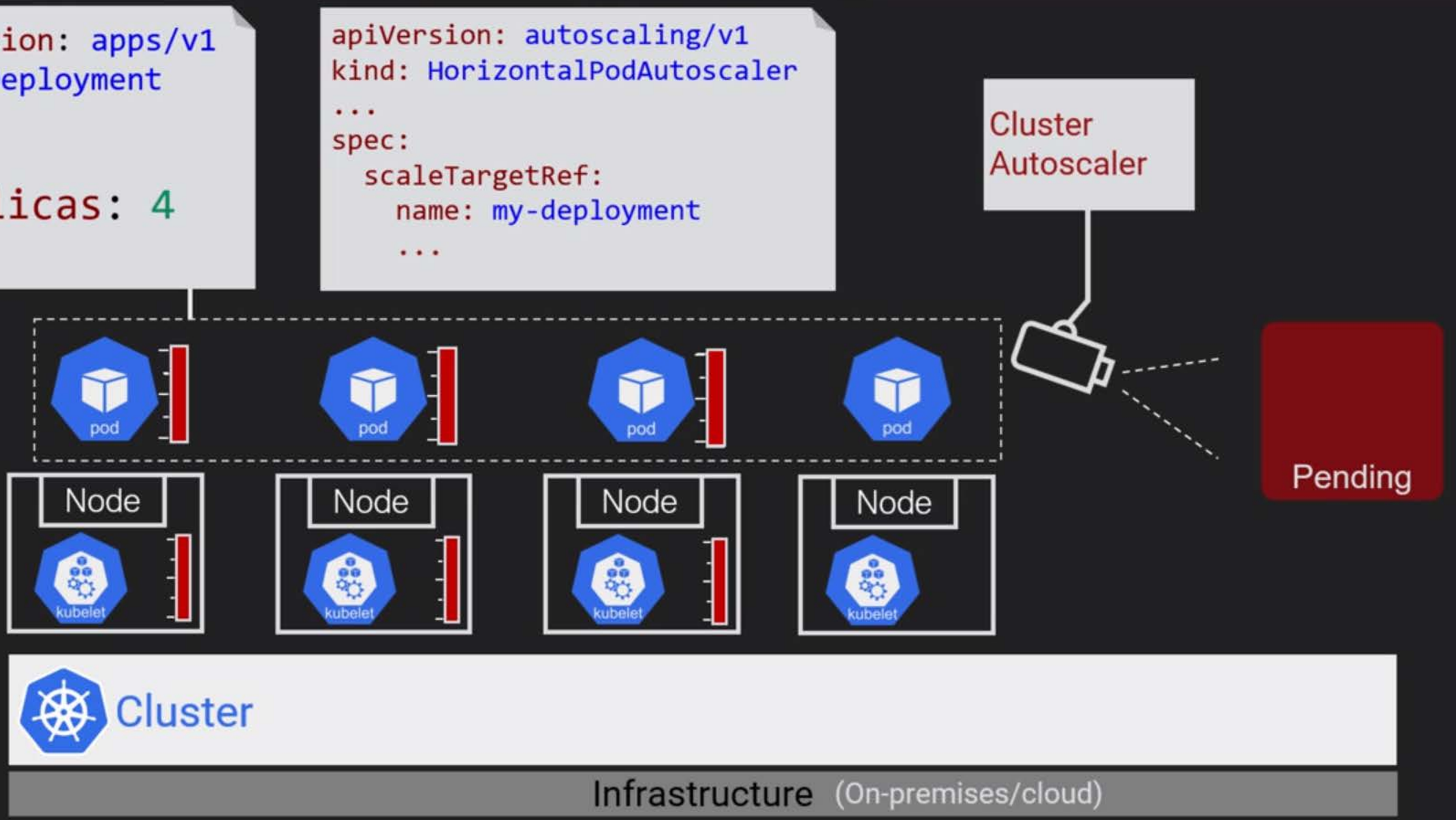


Big Picture

A CLOUD GURU

```
apiVersion: apps/v1
kind: Deployment
...
spec:
  replicas: 4
  ...
```

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
...
spec:
  scaleTargetRef:
    name: my-deployment
    ...
```



Horizontal Pod Autoscaler - Theory

A CLOUD GURU

Horizontal Pod Autoscaler

Horizontal Pod Autoscaler - Theory

A CLOUD GURU


```yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: acg-test
  namespace: acg
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: acg-deploy
  minReplicas: 1
  maxReplicas: 10
targetCPUUtilizationPercentage: 50
```

```
1000m = 1CPU
1 = 1CPU

500m = ½CPU
0.5 = ½CPU

250m = ¼CPU
0.25 = ¼CPU
```


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: acg-deploy
spec:
  replicas: 2
  ...
    spec:
      containers:
      - image: nginx:1.12
        name: nginx
        resources:
          limits:
            cpu: 1
          requests:
            cpu: 0.2 “200m”
```

So for us here, 0.2 is the same as saying 200 millicores.

Horizontal Pod Autoscaler - Theory


```yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: acg-test
  namespace: acg
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: acg-deploy
  minReplicas: 1
  maxReplicas: 10
targetCPUUtilizationPercentage: 50
```


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: acg-deploy
spec:
  replicas: 2
  ...
    spec:
      containers:
      - image: nginx:1.12
        name: nginx
        resources:
          limits:
            cpu: 1
          requests:
            cpu: 0.2
```

Okay, if we attach it to this deployment,

Horizontal Pod Autoscaler - Theory


```yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: acg-test
  namespace: acg
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: acg-deploy
  minReplicas: 1
  maxReplicas: 10
targetCPUUtilizationPercentage: 50
```


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: acg-deploy
spec:
  replicas: 2
  ...
    spec:
      containers:
      - image: nginx:1.12
        name: nginx
        resources:
          limits:
            cpu: 1
          requests:
            cpu: 0.2
```

The autoscaling/v2 API will bring custom metrics etc.

Horizontal Pod Autoscaler - Theory

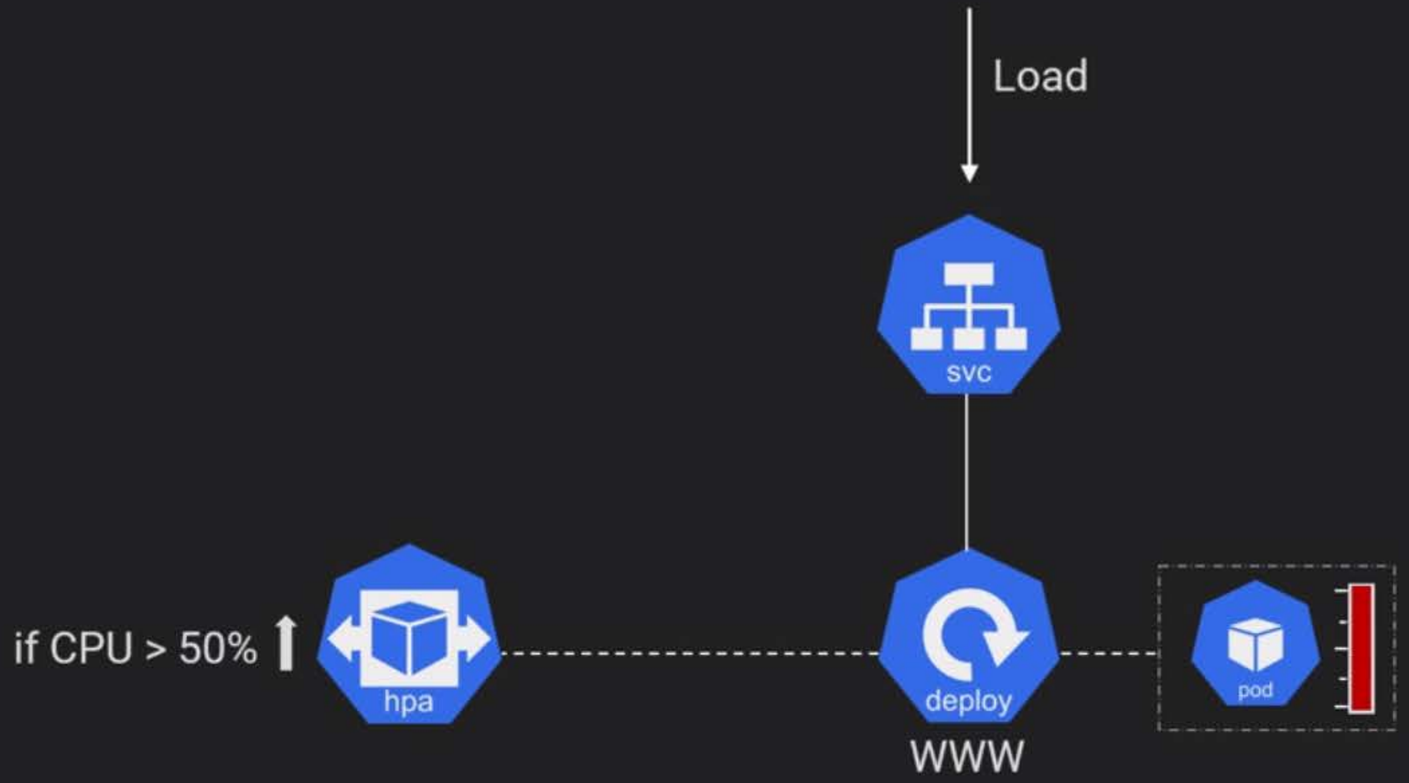
A CLOUD GURU


```yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: acg-test
  namespace: acg
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: acg-deploy
  minReplicas: 1
  maxReplicas: 10
targetCPUUtilizationPercentage: 50
```


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: acg-deploy
spec:
  replicas: 3
  ...
    spec:
      containers:
      - image: nginx:1.12
        name: nginx
        resources:
          limits:
            cpu: 1
          requests:
            cpu: 0.2
```

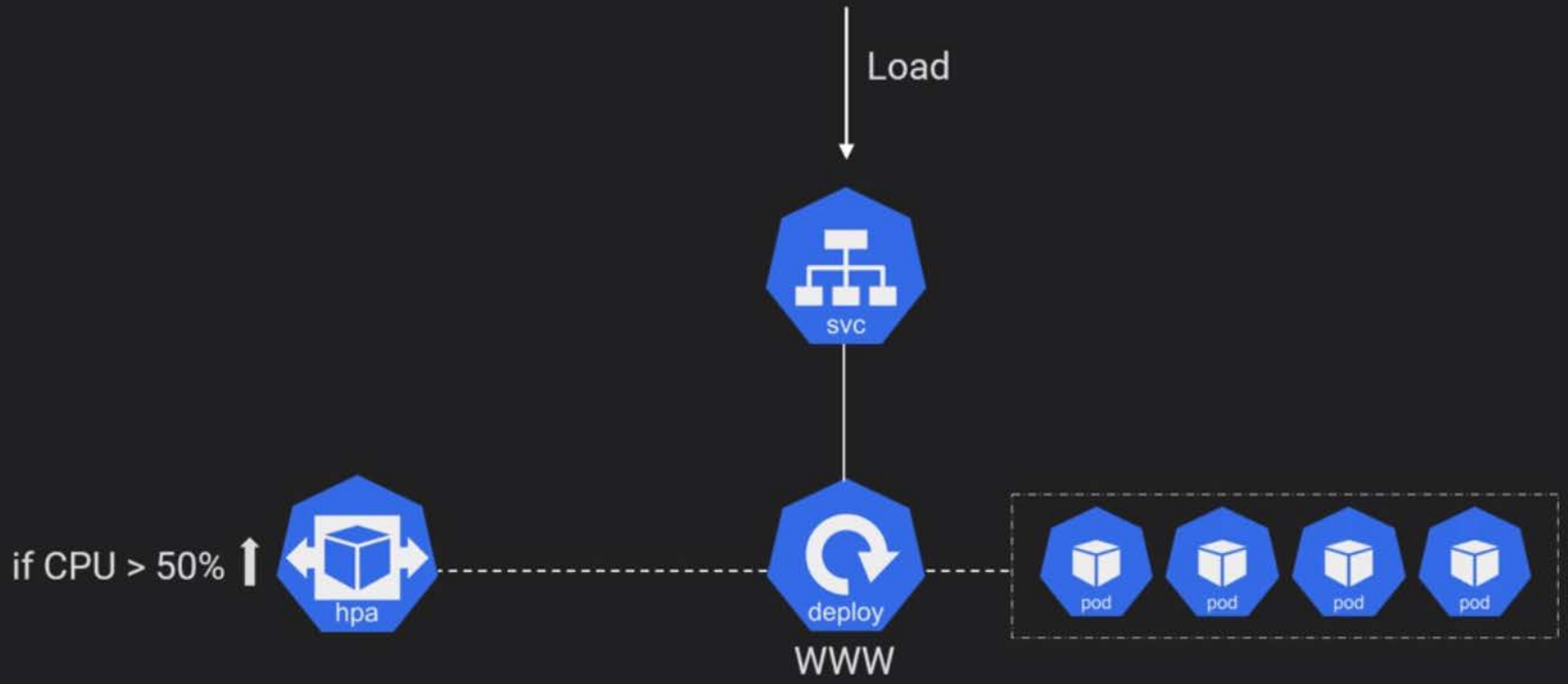
The autoscaling/v2 API will bring custom metrics etc.

Horizontal Pod Autoscaler - Demo

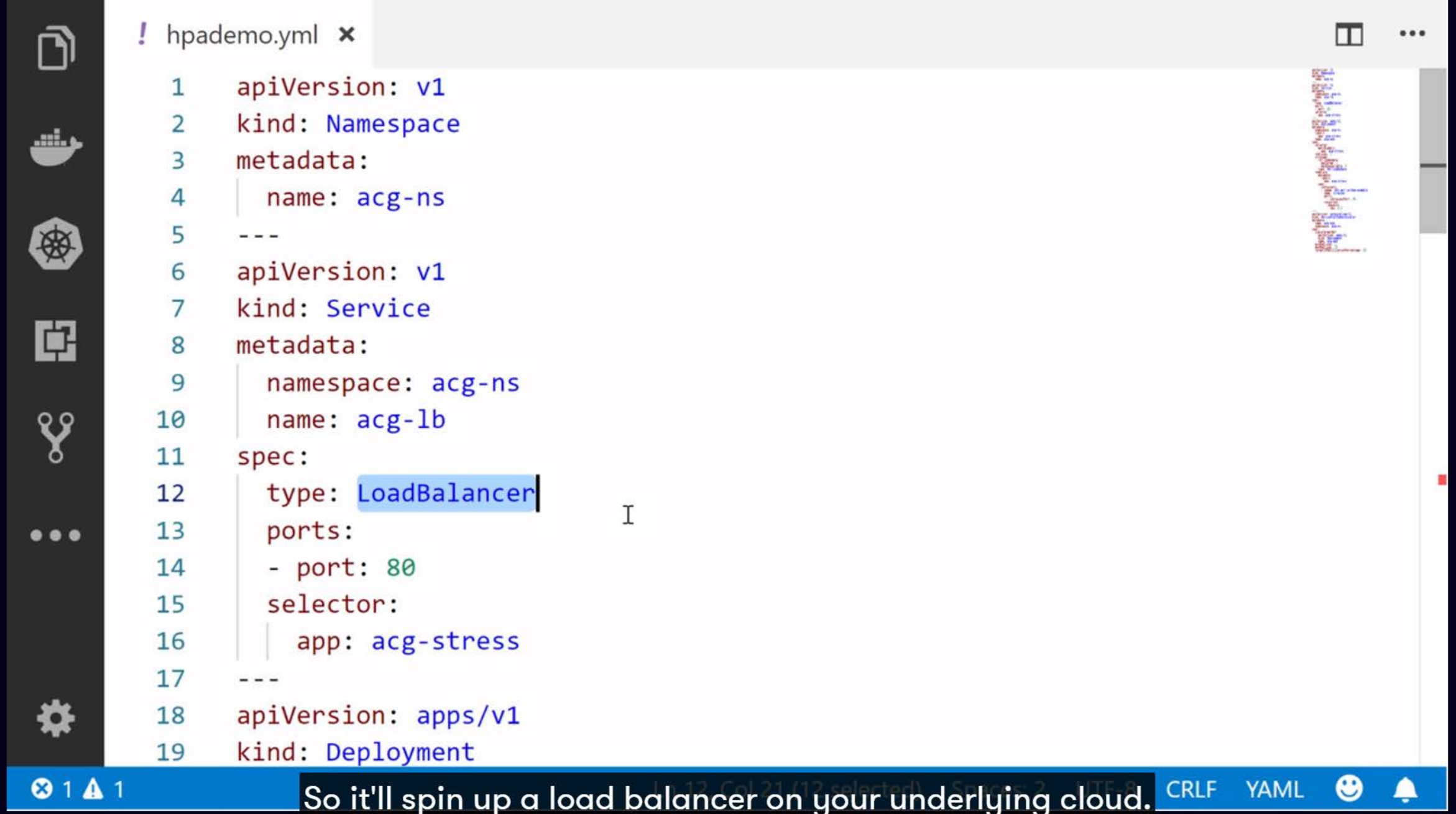


And as we're doing that, it's gonna ramp up on CPU,

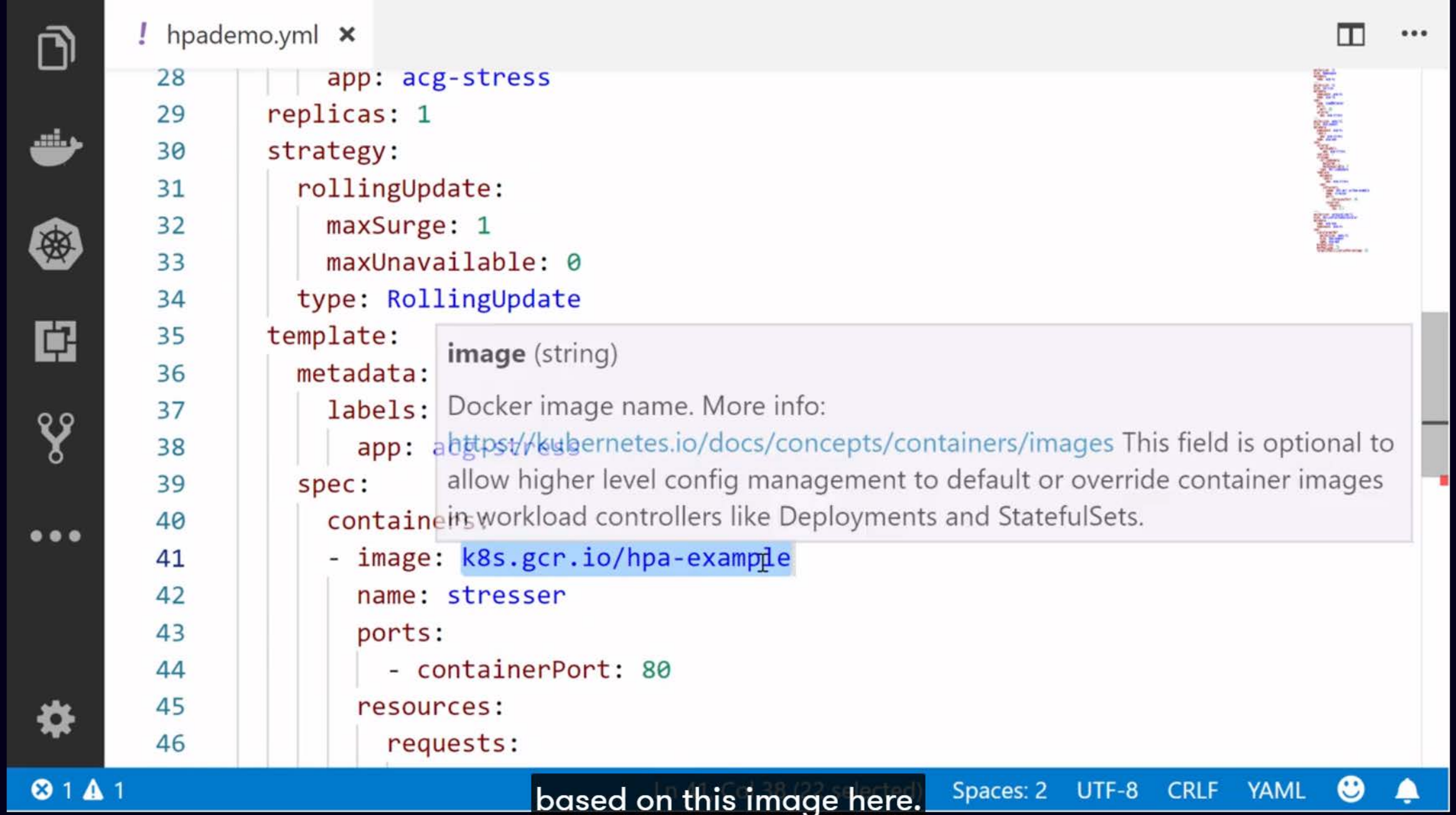
Horizontal Pod Autoscaler - Demo



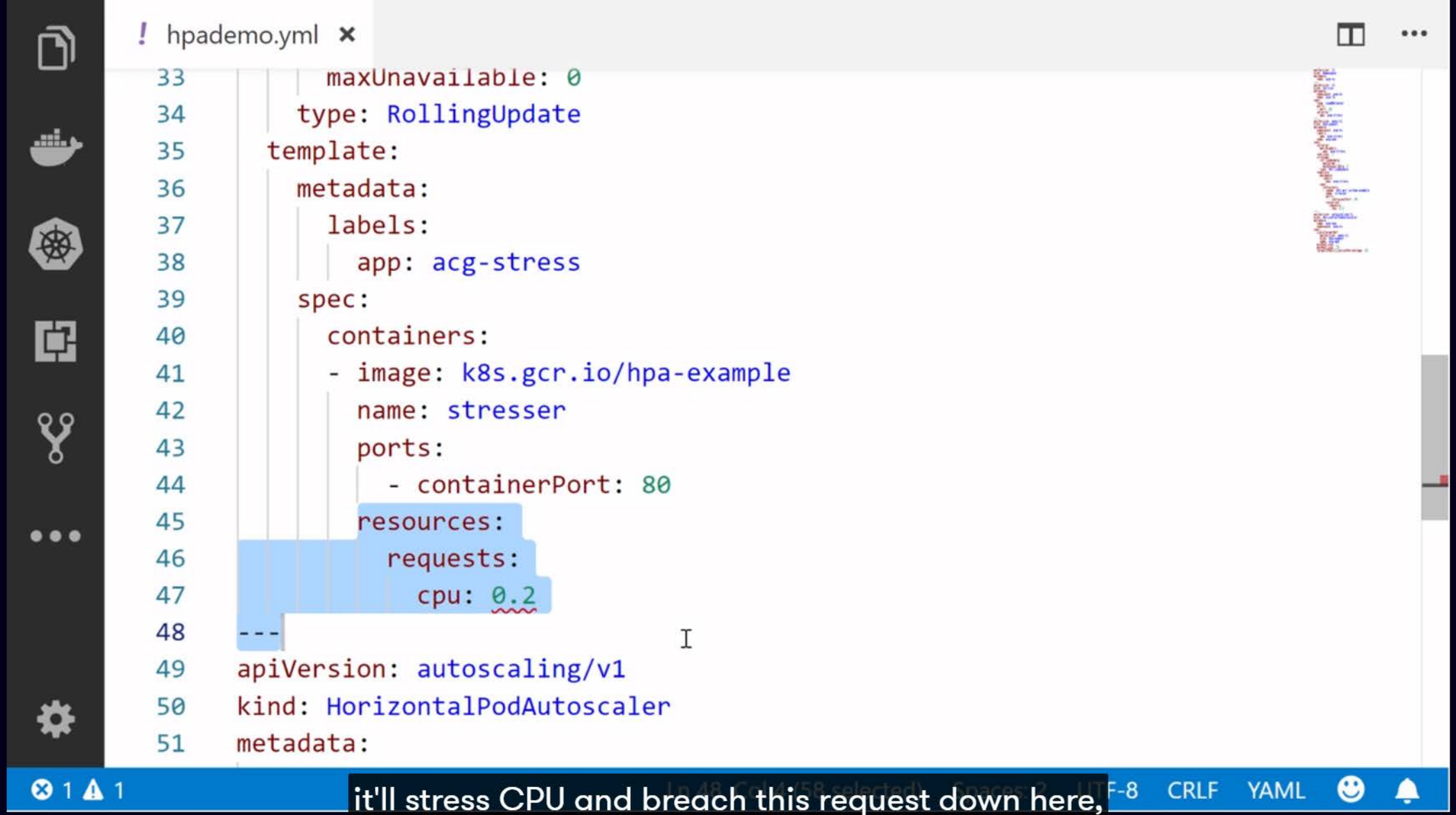
And that's the plan.



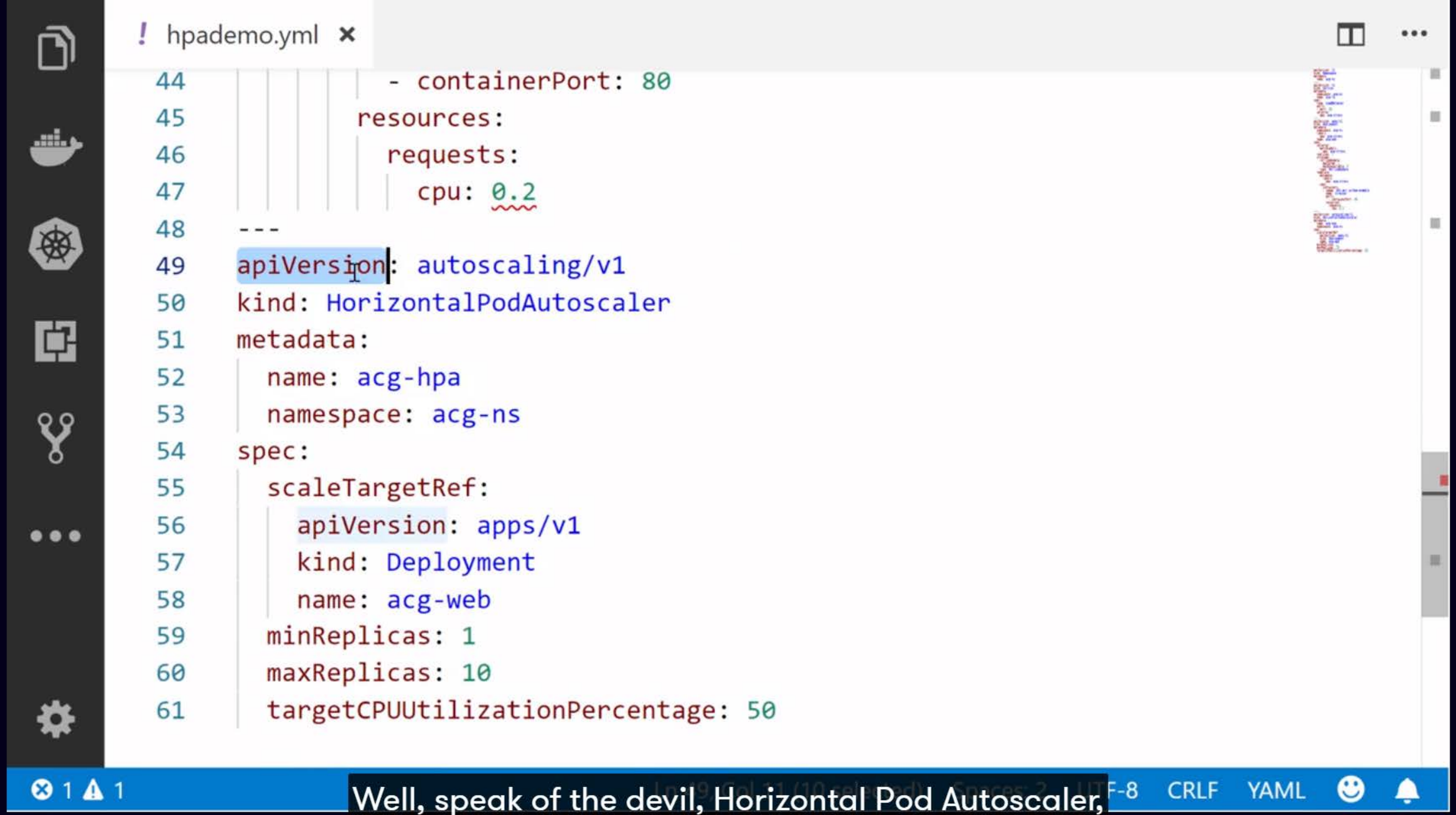
So it'll spin up a load balancer on your underlying cloud.



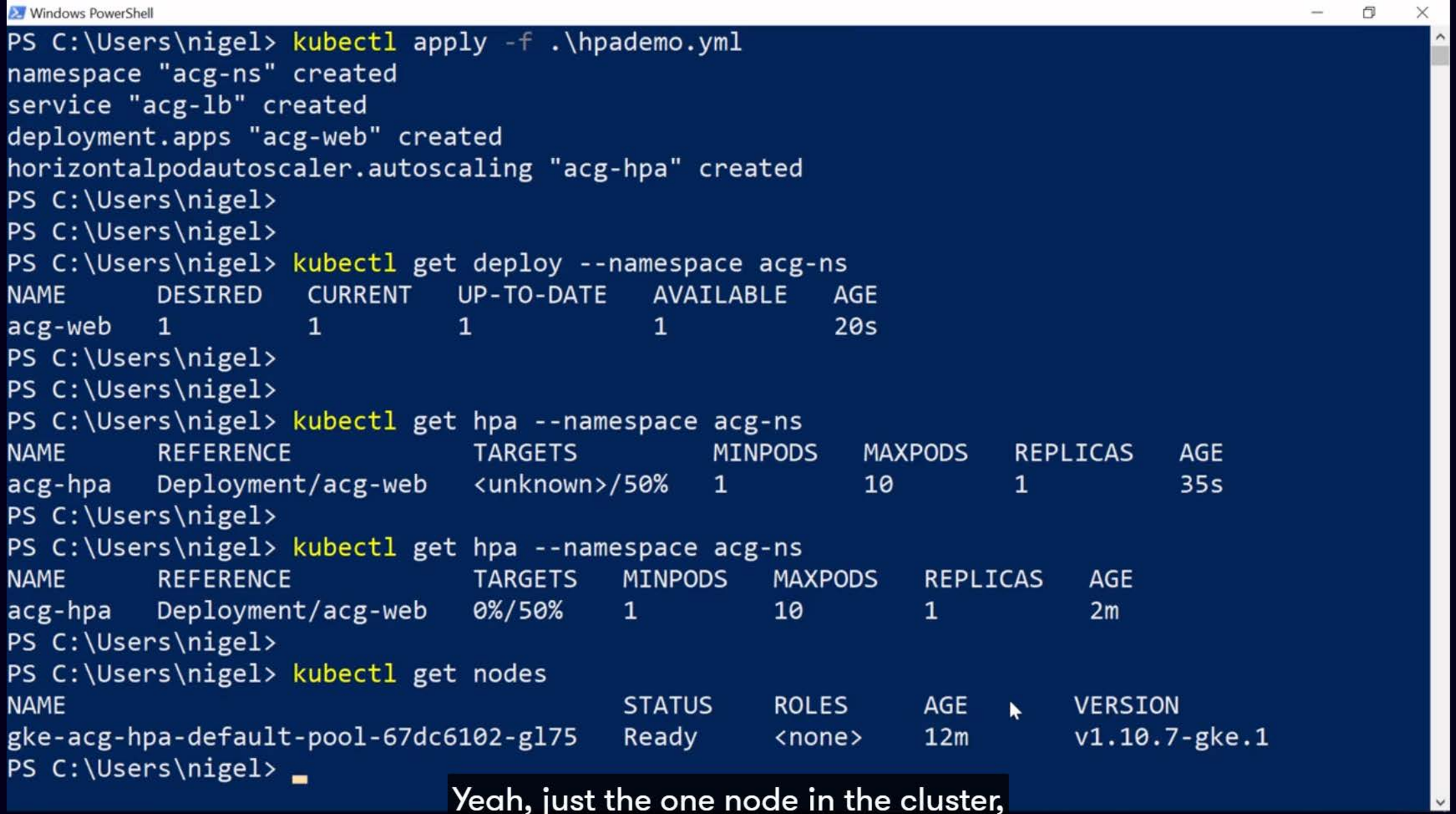
based on this image here.



it'll stress CPU and breach this request down here,



Well, speak of the devil, Horizontal Pod Autoscaler,


Windows PowerShell
PS C:\Users\nigel> kubectl apply -f .\hpademo.yml
namespace "acg-ns" created
service "acg-lb" created
deployment.apps "acg-web" created
horizontalpodautoscaler.autoscaling "acg-hpa" created
PS C:\Users\nigel>
PS C:\Users\nigel>
PS C:\Users\nigel> kubectl get deploy --namespace acg-ns
NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE
acg-web 1 1 1 1 20s
PS C:\Users\nigel>
PS C:\Users\nigel>
PS C:\Users\nigel> kubectl get hpa --namespace acg-ns
NAME REFERENCE TARGETS MINPODS MAXPODS REPLICAS AGE
acg-hpa Deployment/acg-web <unknown>/50% 1 10 1 35s
PS C:\Users\nigel>
PS C:\Users\nigel> kubectl get hpa --namespace acg-ns
NAME REFERENCE TARGETS MINPODS MAXPODS REPLICAS AGE
acg-hpa Deployment/acg-web 0%/50% 1 10 1 2m
PS C:\Users\nigel>
PS C:\Users\nigel> kubectl get nodes
NAME STATUS ROLES AGE VERSION
gke-acg-hpa-default-pool-67dc6102-gl75 Ready <none> 12m v1.10.7-gke.1
PS C:\Users\nigel>
Yeah, just the one node in the cluster,

Cluster Autoscaler - Theory


```
apiVersion: apps/v1
kind: Deployment
spec:
...
    spec:
      resources:
        requests:
            cpu: 1
```

Always use
Pod Resource Requests

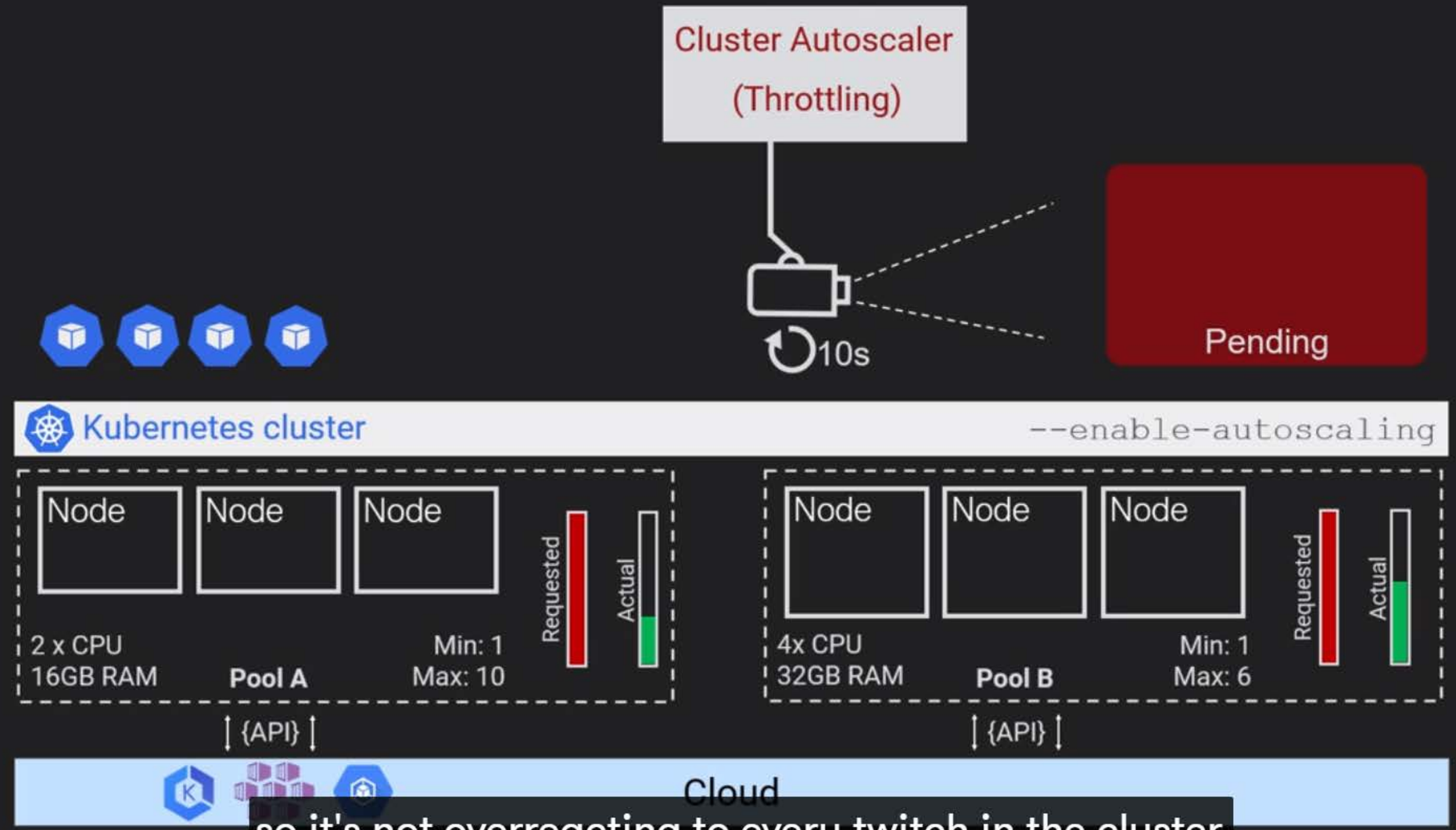
Cluster Autoscaler - Theory

A CLOUD GURU

Cluster Autoscaler	Works on **requested** values
Horizontal Pod Autoscaler	Works on **actual** values

Wells it's a pro tip to schedule all of your pods

Cluster Autoscaler - Theory



so it's not overreacting to every twitch in the cluster.

Cluster Autoscaler - Theory

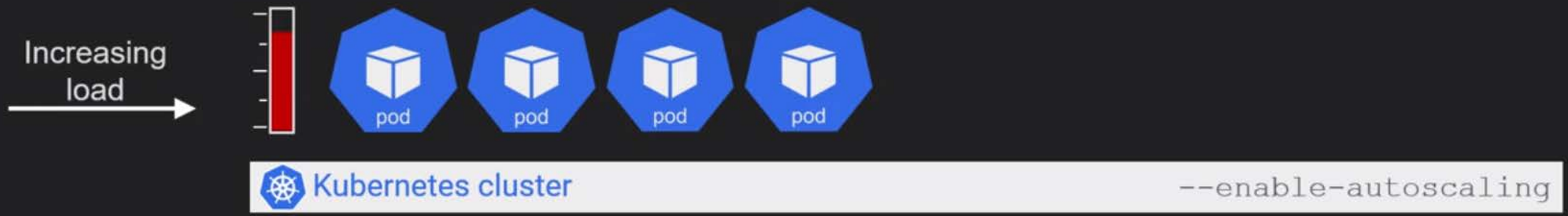
Don’t mess with the node pools

Check your cloud for support

Test performance on big clusters

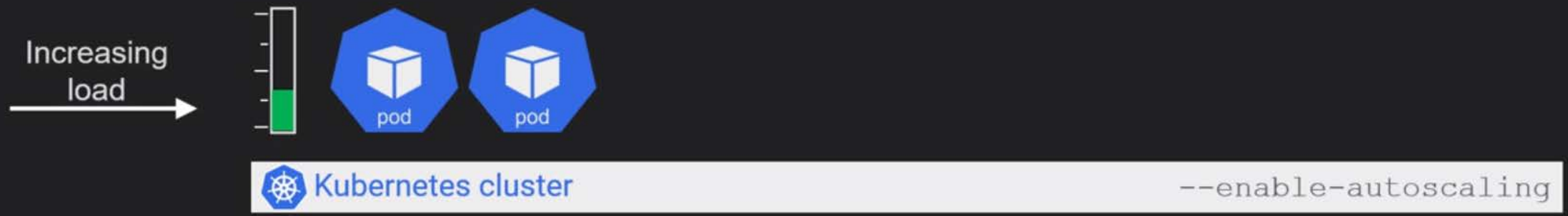
Because you might experience performance issues.

Recap



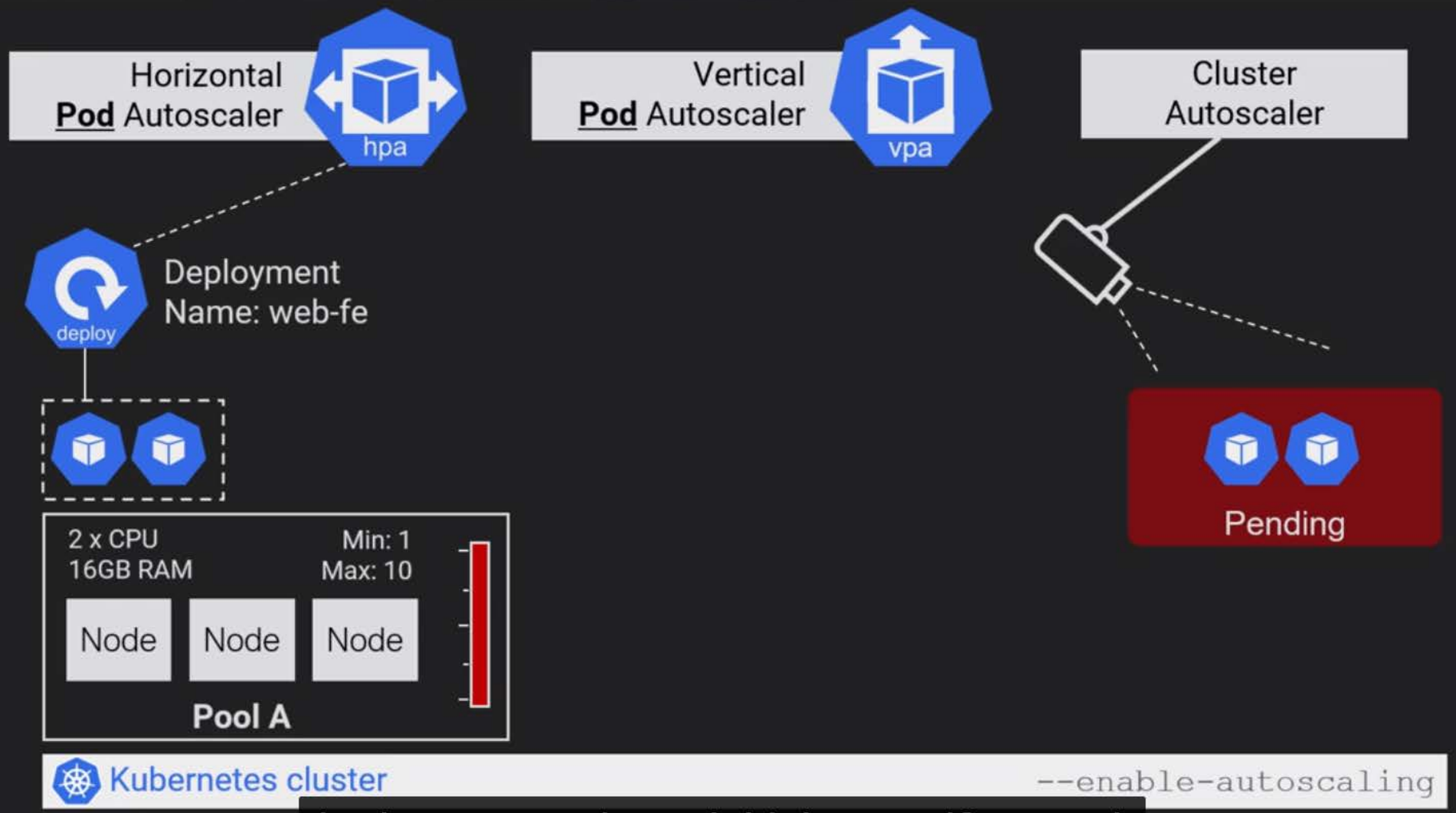
As pressure builds, we scale up,

Recap



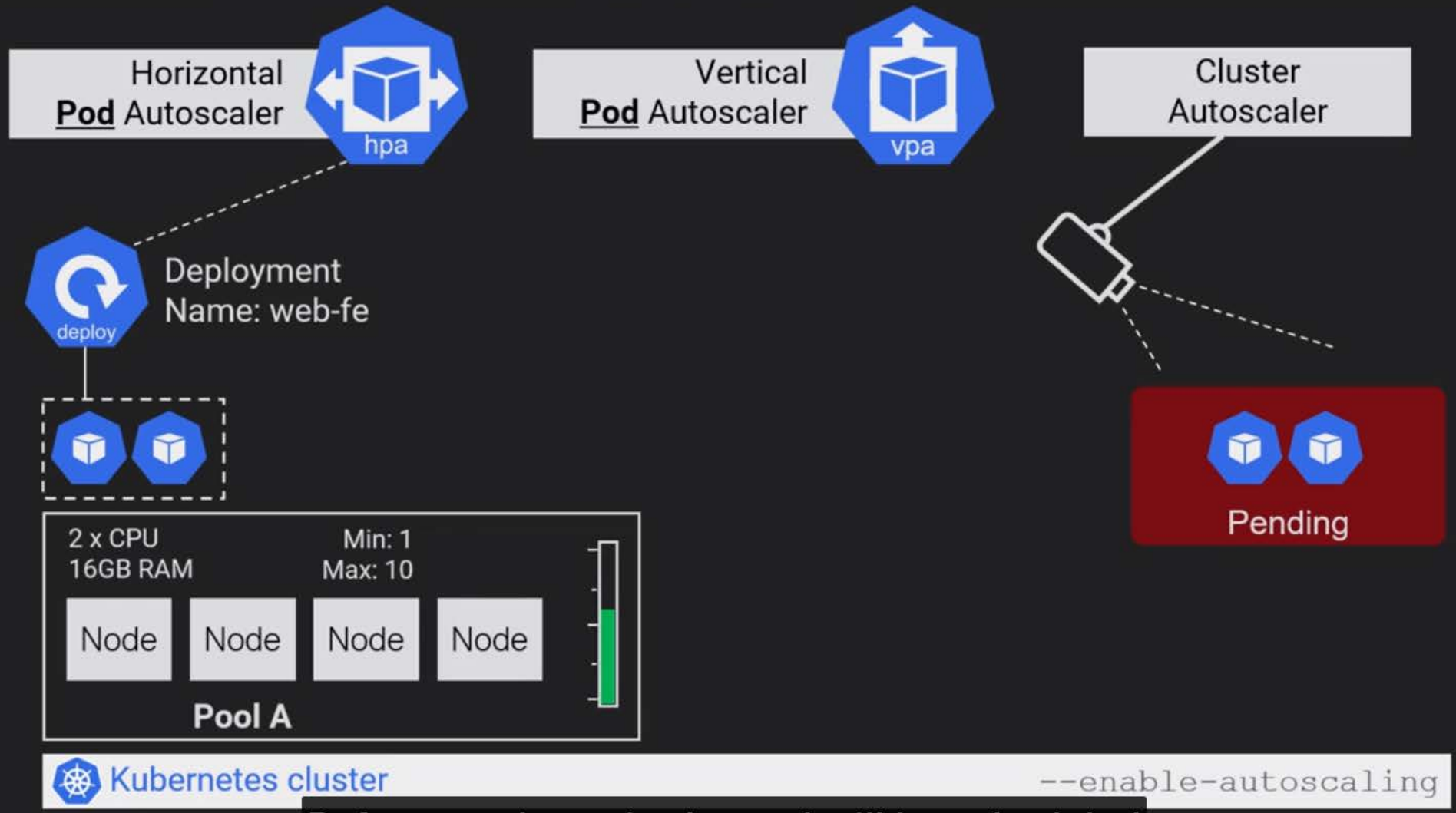
But for it to be properly useful,

Recap



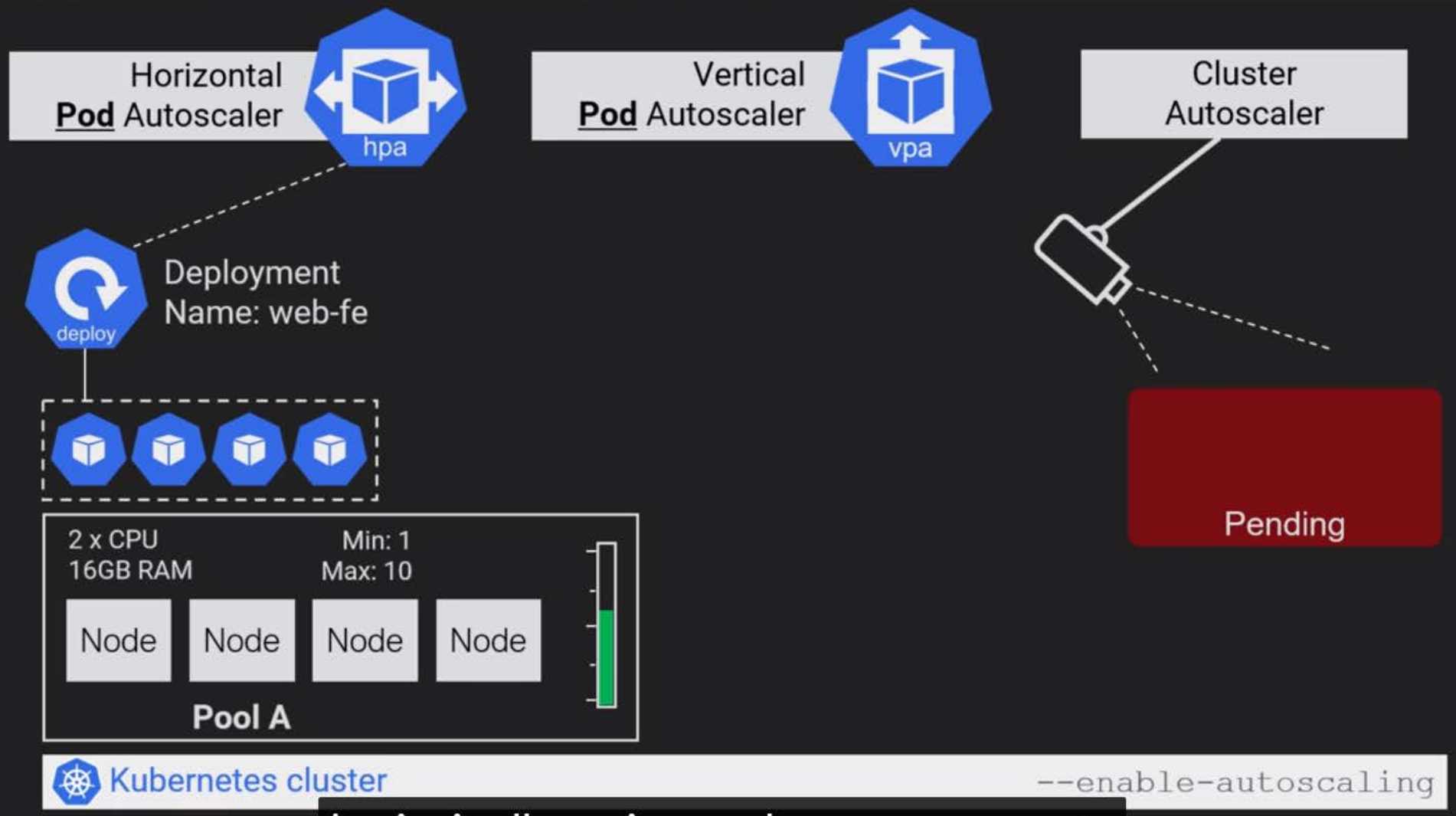
that's gonna wake up I think every 10 seconds

Recap



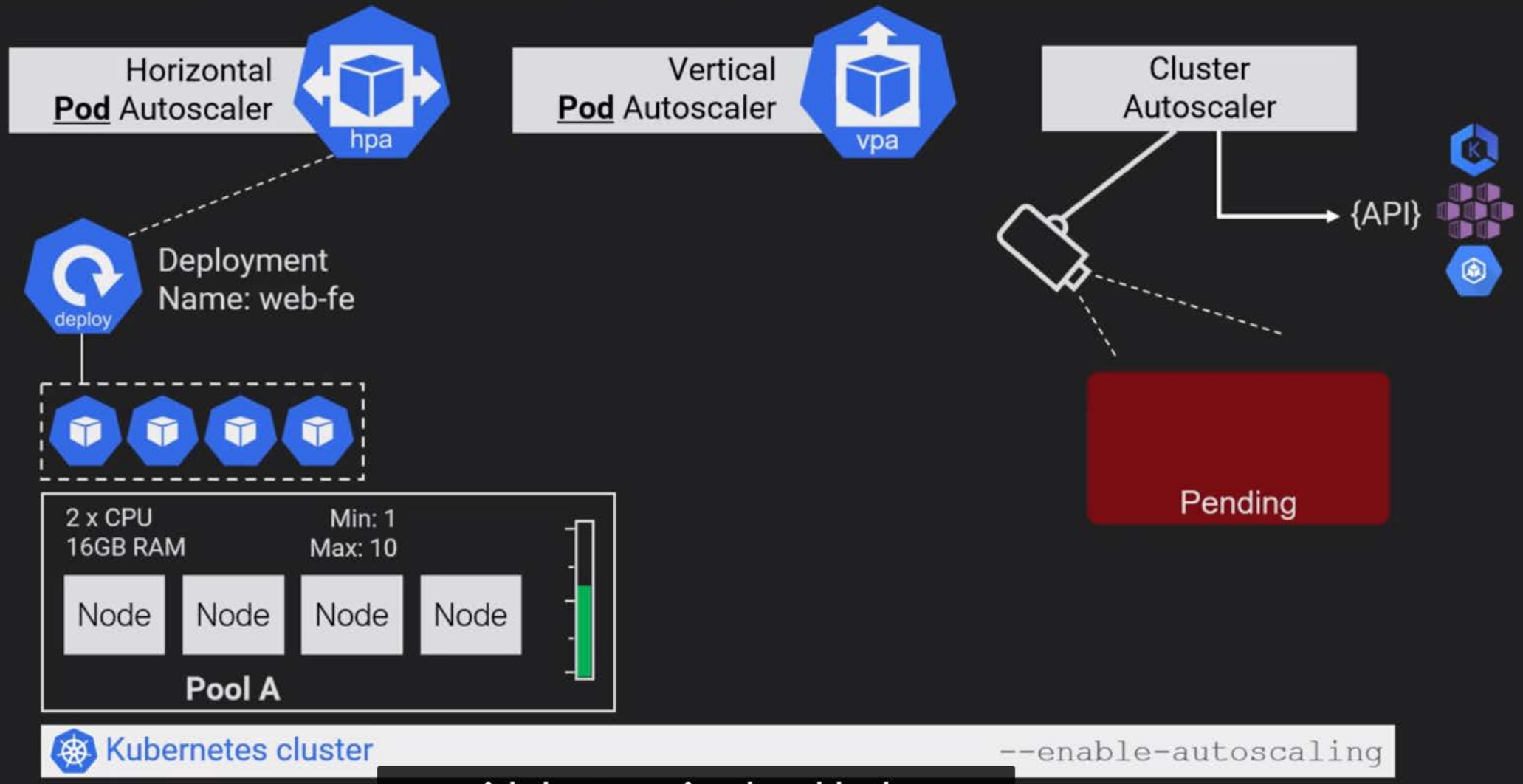
Before you know it, the pod will be scheduled

Recap



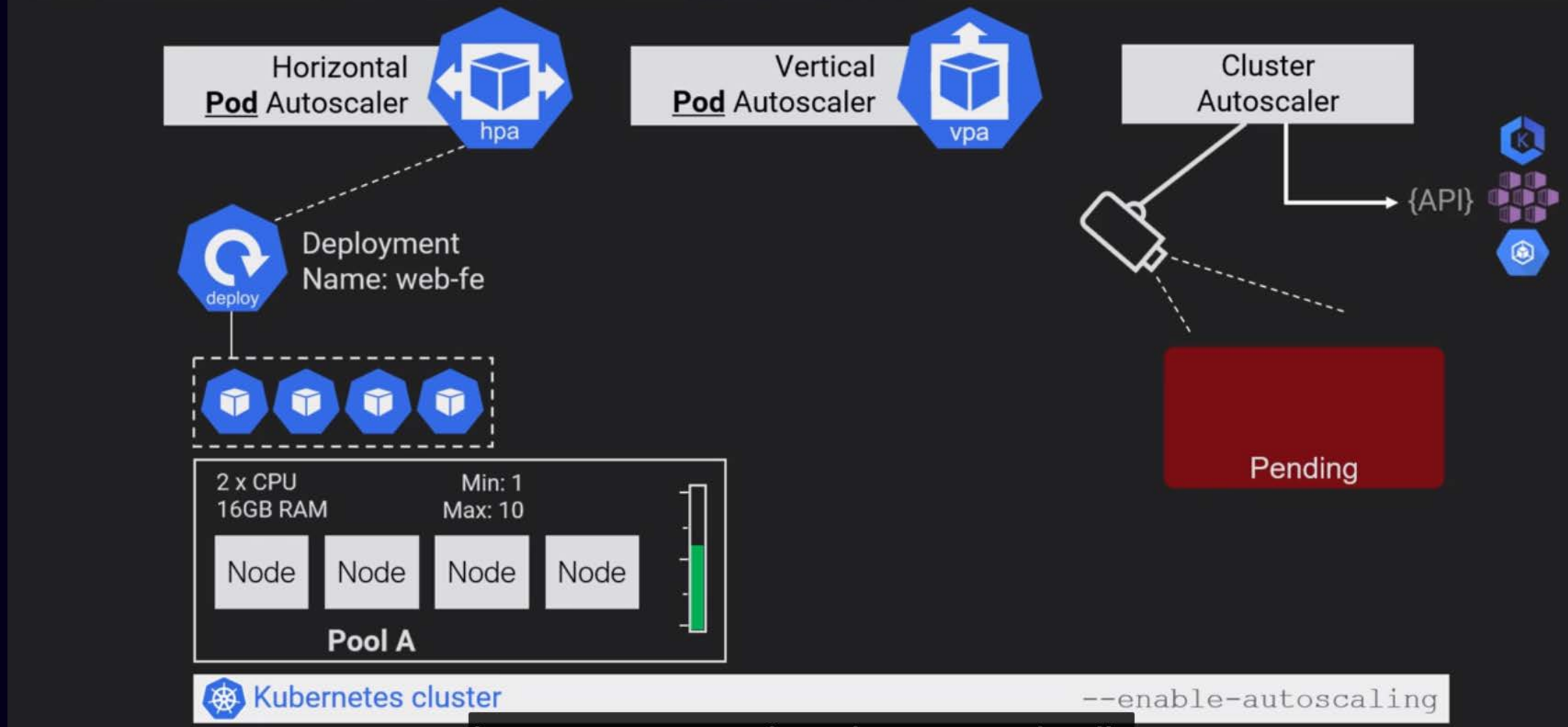
Again, it all requires pod resource requests.

Recap



we said that service load balancers

Recap



because you see, it makes external calls

Scaling Applications

Recap

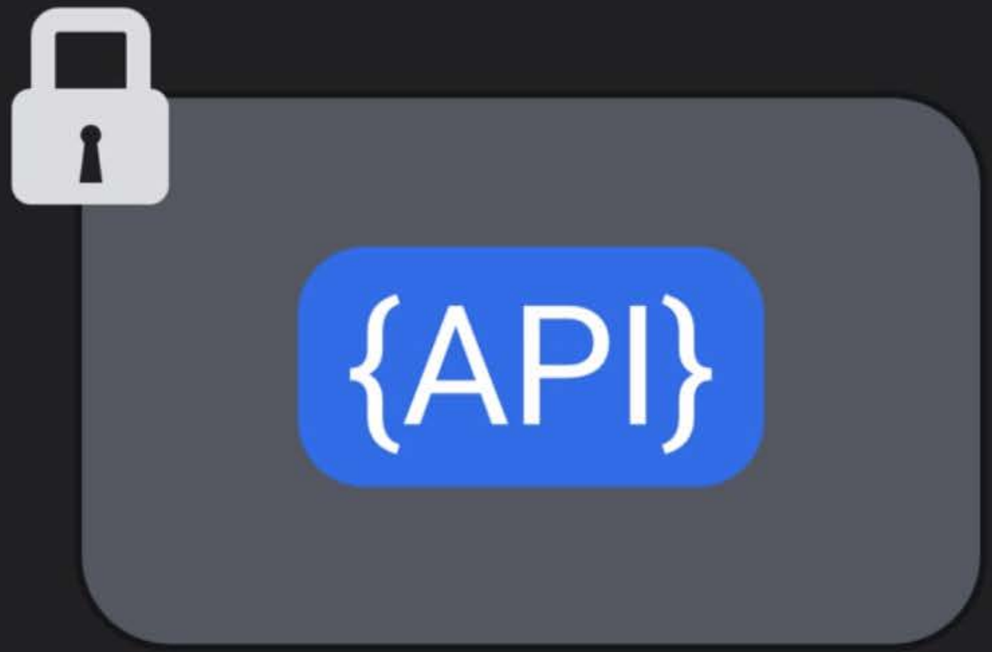
A CLOUD GURU

autoscaling/v1
- CPU

autoscaling/v2
- CPU, memory, **custom metrics**

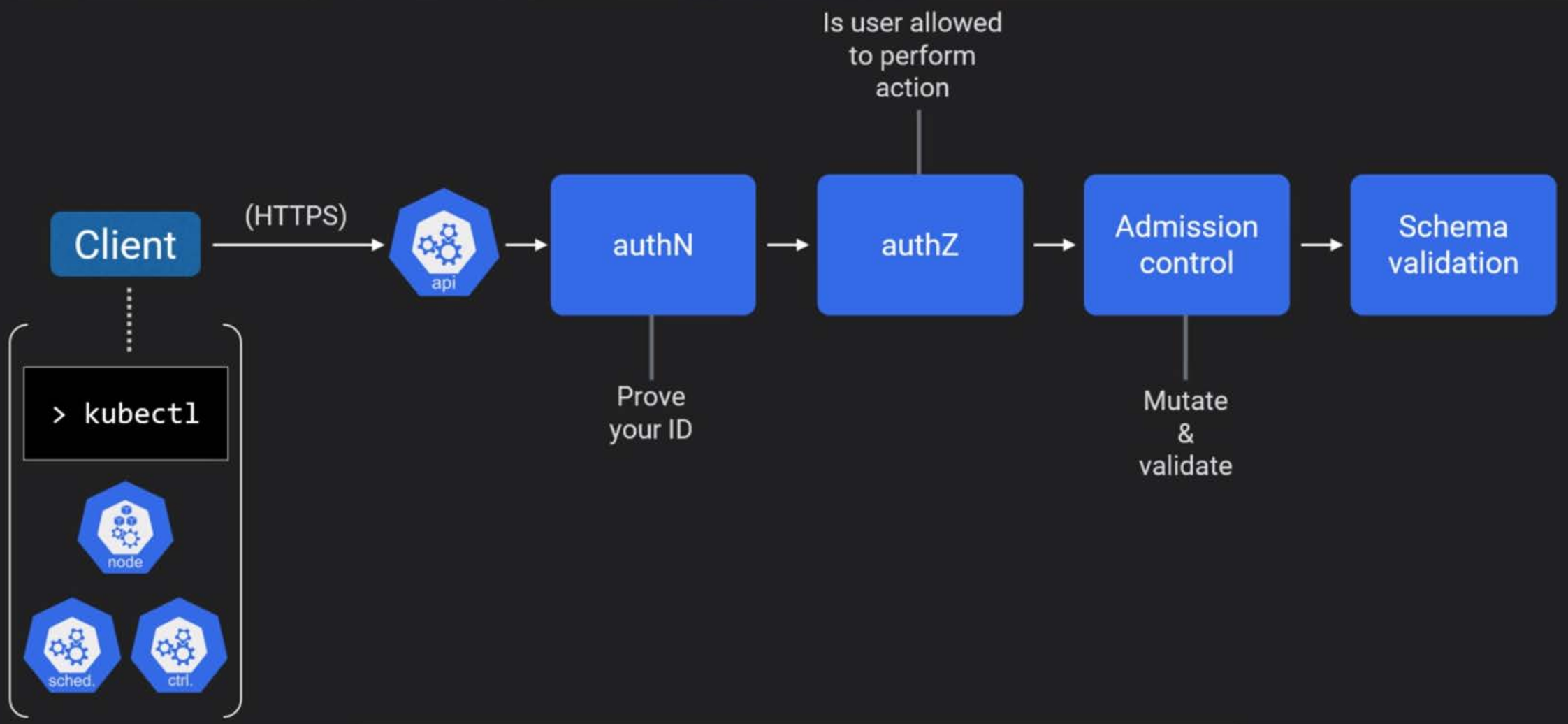
custom metrics is where the magic is gonna be.

Big Picture



So, normal interaction with the API server looks like this.

Big Picture



But if the request passes through all of these gates?

Big Picture

Some clusters open an insecure local port!

Bypasses authN and authZ!!

Disable for prod!

So, whether it's even open or enabled in the first place,

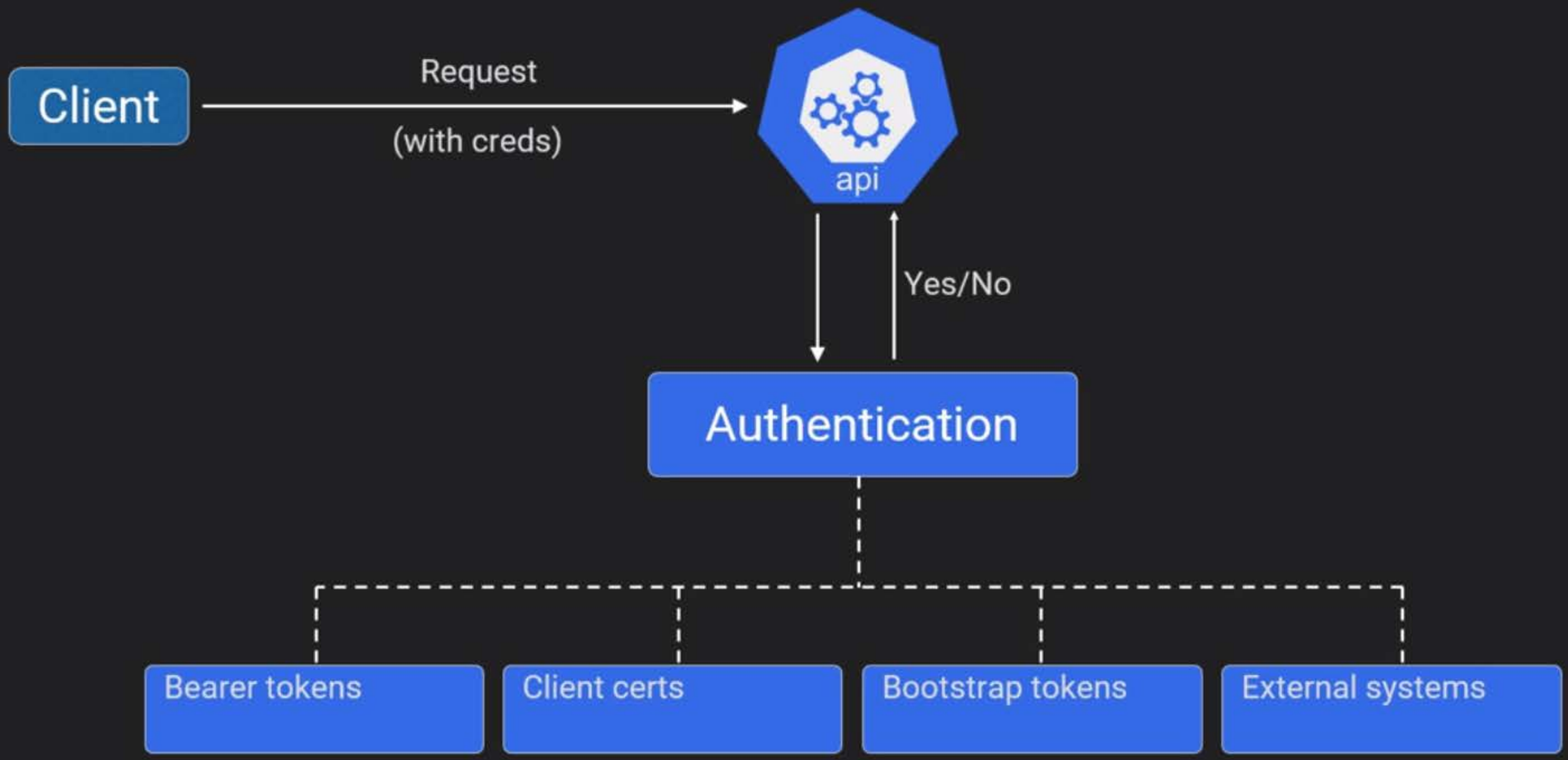
Big Picture

RBAC

- Enabled since 1.6
- GA since 1.8
- Deny-by-default

Then you have to specifically add allow rules

Authentication



And you can have them all enabled at the same time.

Authentication

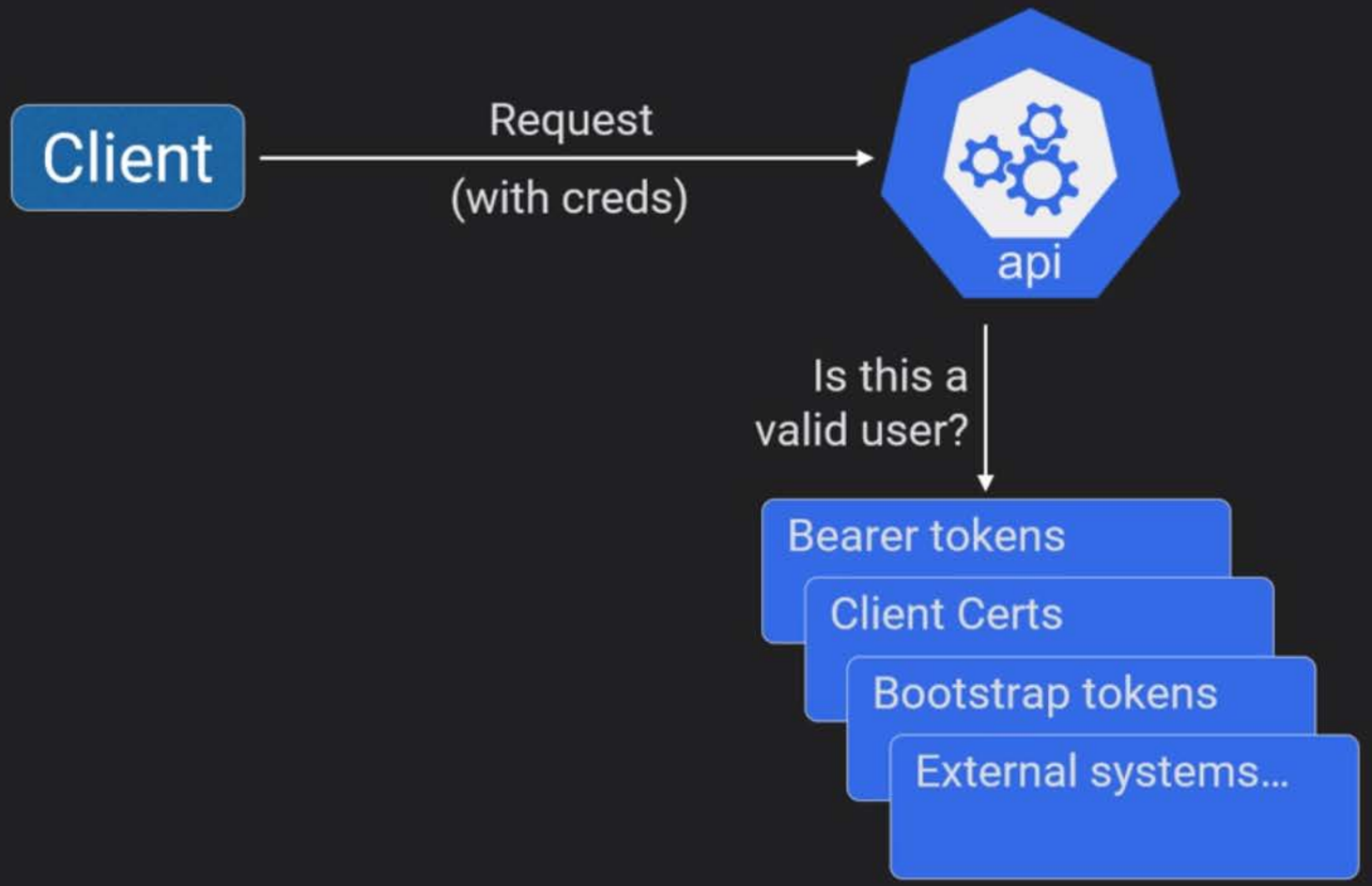
Kubernetes does **NOT** do Users!!

Manage Users externally

- Active Directory
- IAM
- Other...

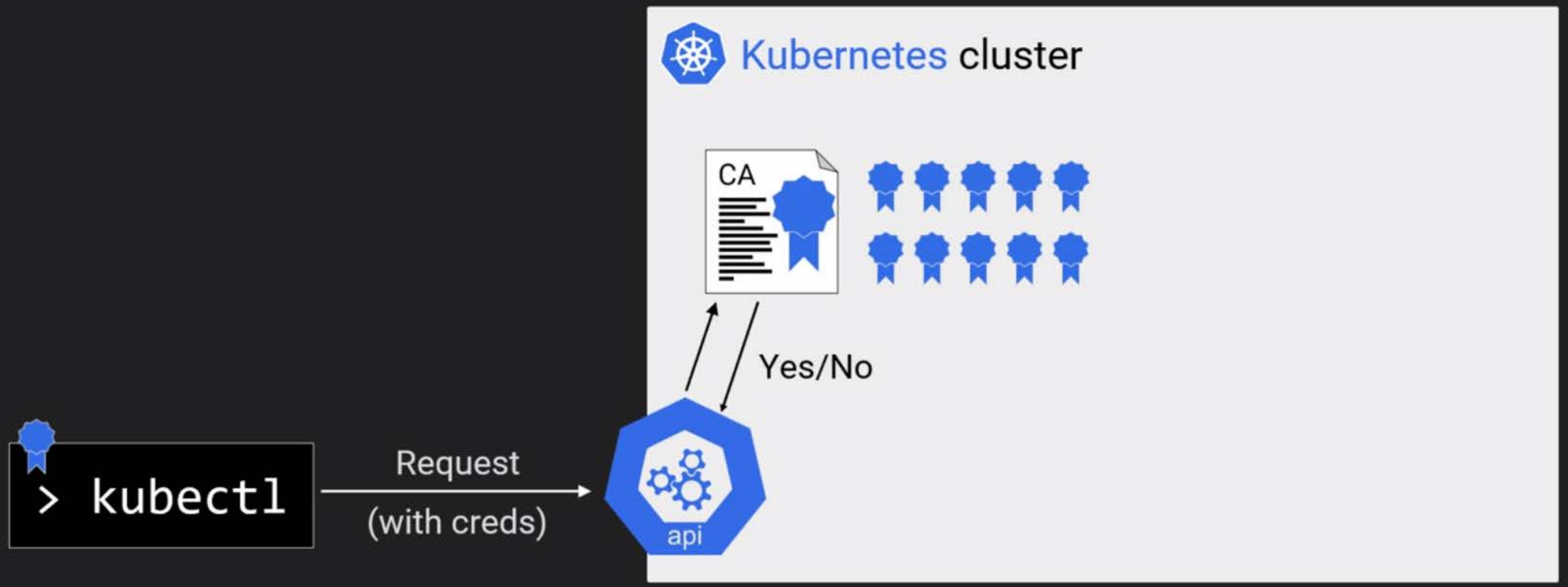
But it can totally be active directory or your clouds,

Authentication



right big hand wave going on here,

Authentication



and says, "Yep, that's one of ours or no, not created here."

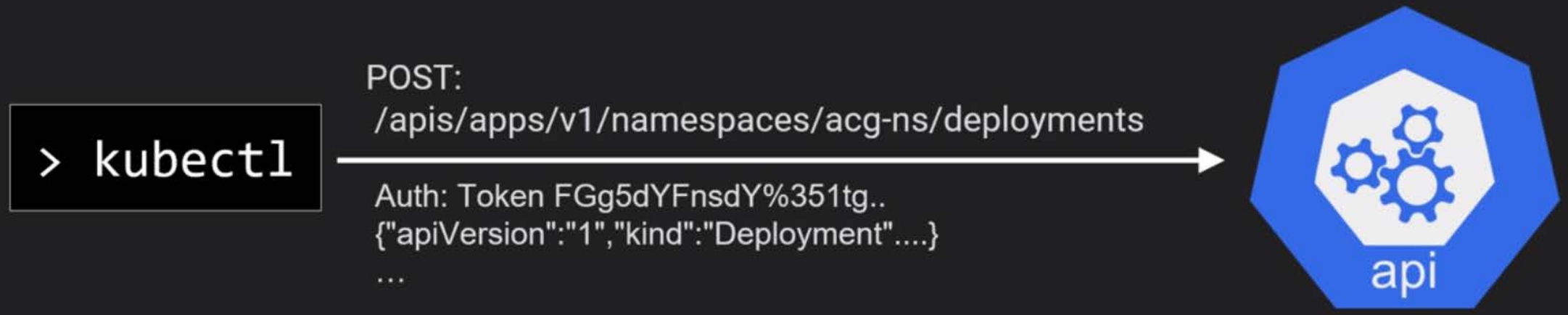
Authentication

Service Accounts

- For `System` components
- Managed by Kubernetes
- You can (should) manage them

as to how you wanna manage these.

Authentication



API Group:	apps/v1
Subject:	nigel
Verb:	create (HTTP POST method)
Resource:	Deployments
Namespace:	acg

Can nigel create new deployments in the ACG namespace?