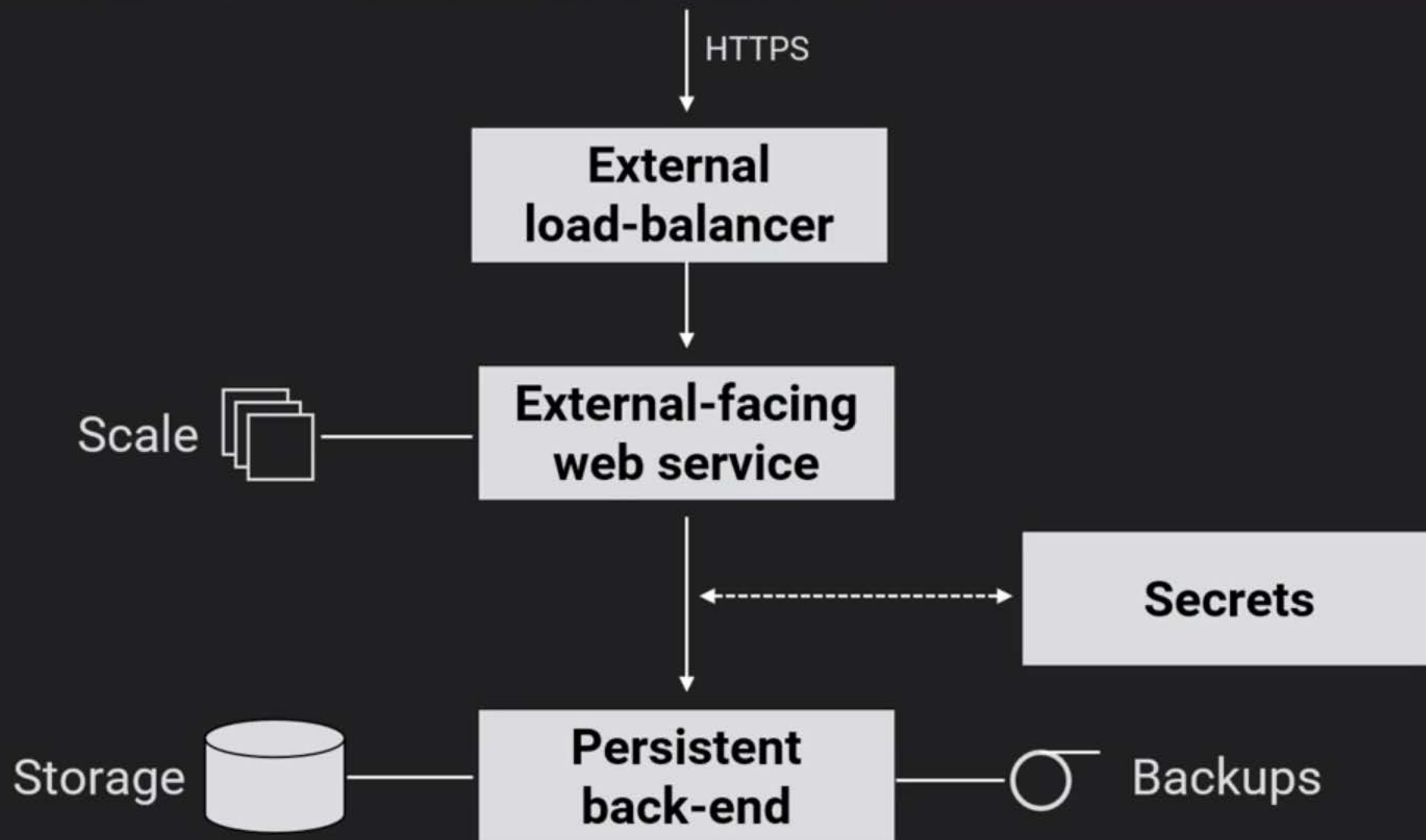
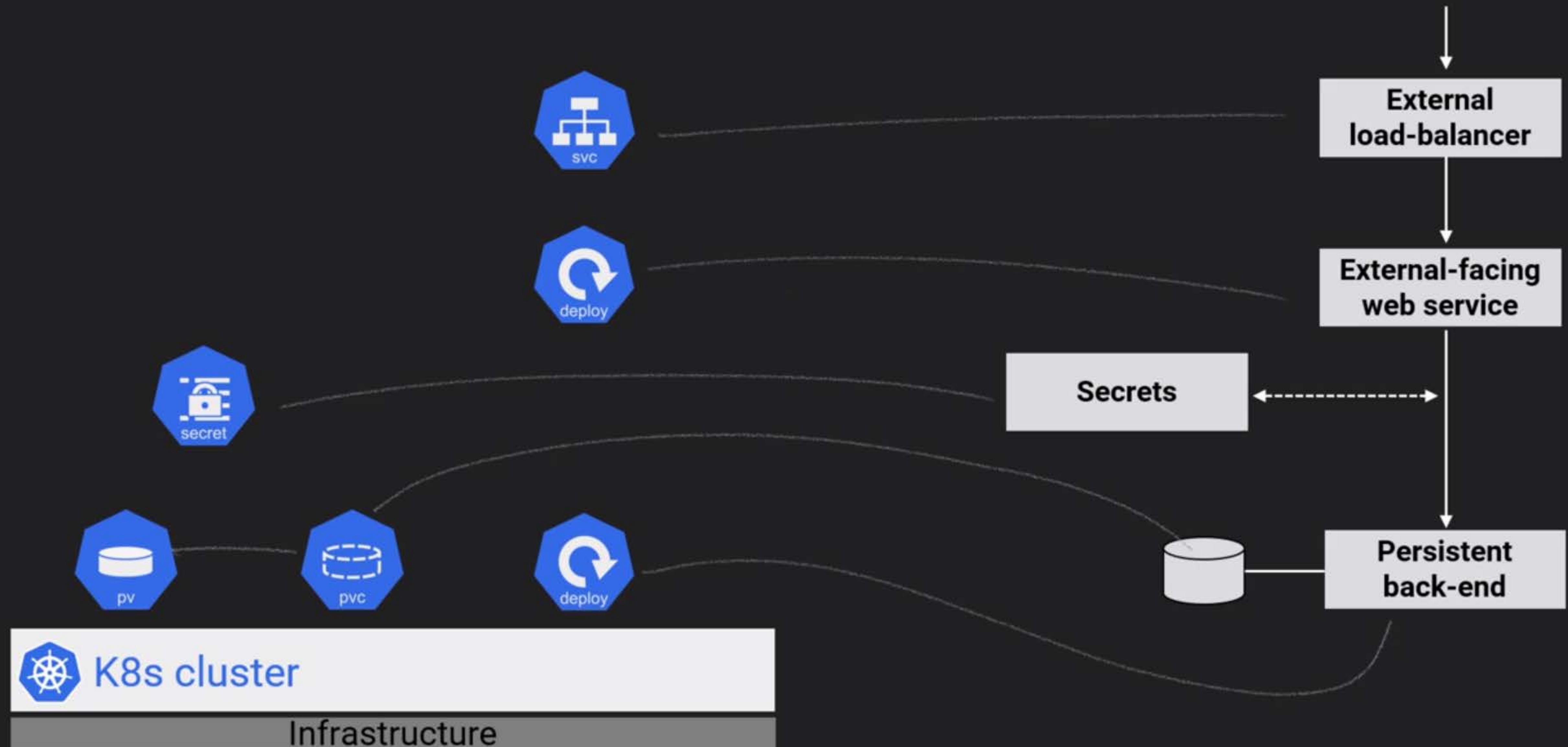


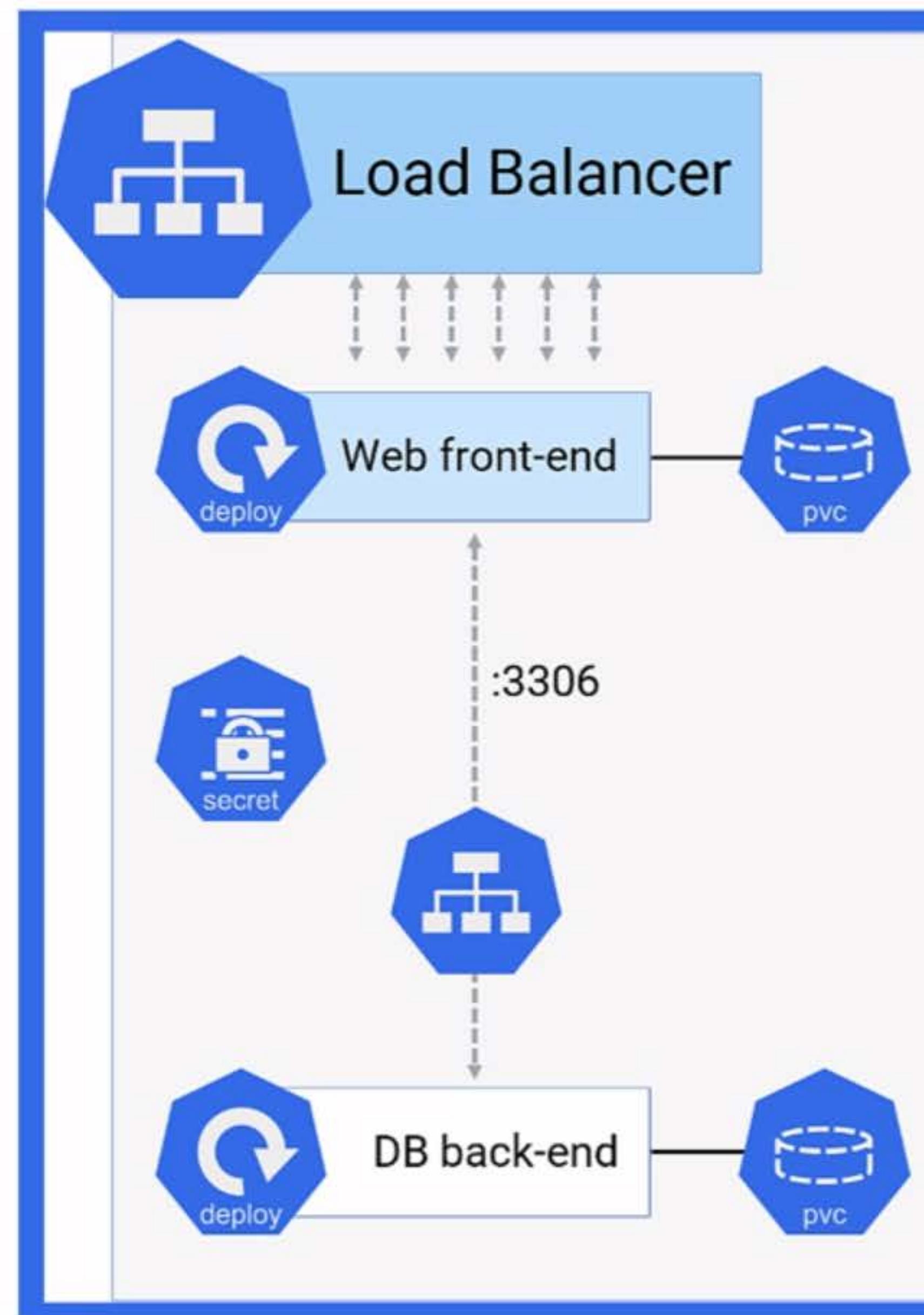
Kubernetes App Theory



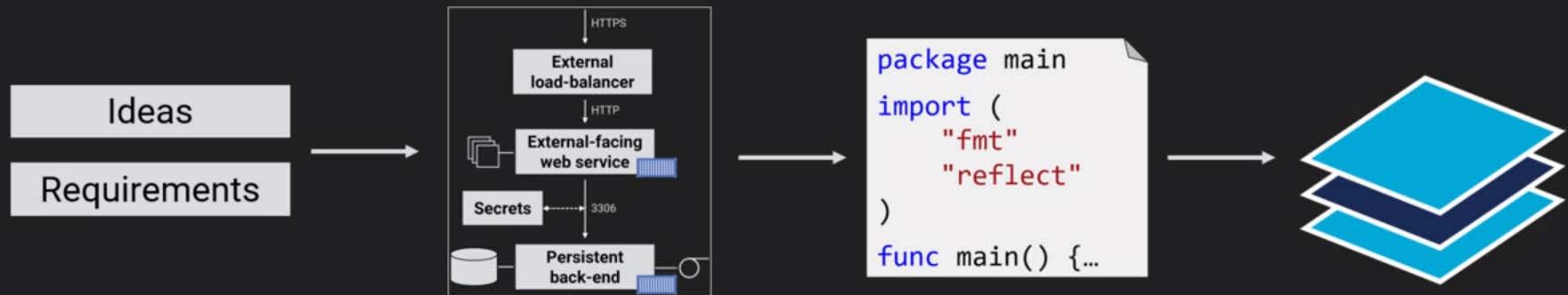
Kubernetes App Theory



```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: wordpress
5    labels:
6      app: wordpress
7  spec:
8    ports:
9      - port: 80
10   selector:
11     app: wordpress
12     tier: frontend
13   type: LoadBalancer
14 ---
15   apiVersion: v1
16   kind: PersistentVolumeClaim
17   metadata:
```



Recap





House Rules

All Nodes can
talk

All Pods can
talk
(No NAT)

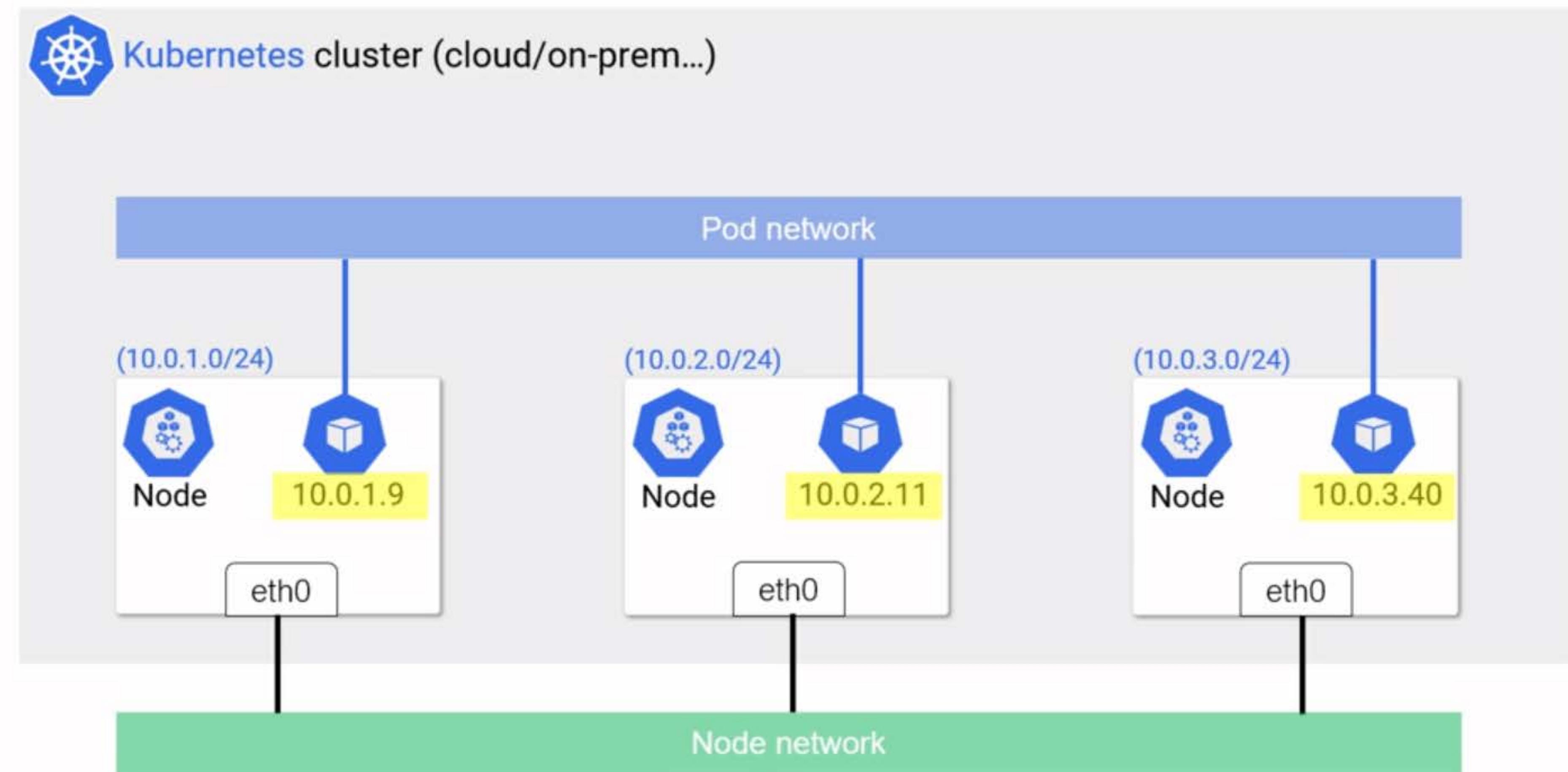
Every Pod gets
its own IP

Kubernetes Networking Basics

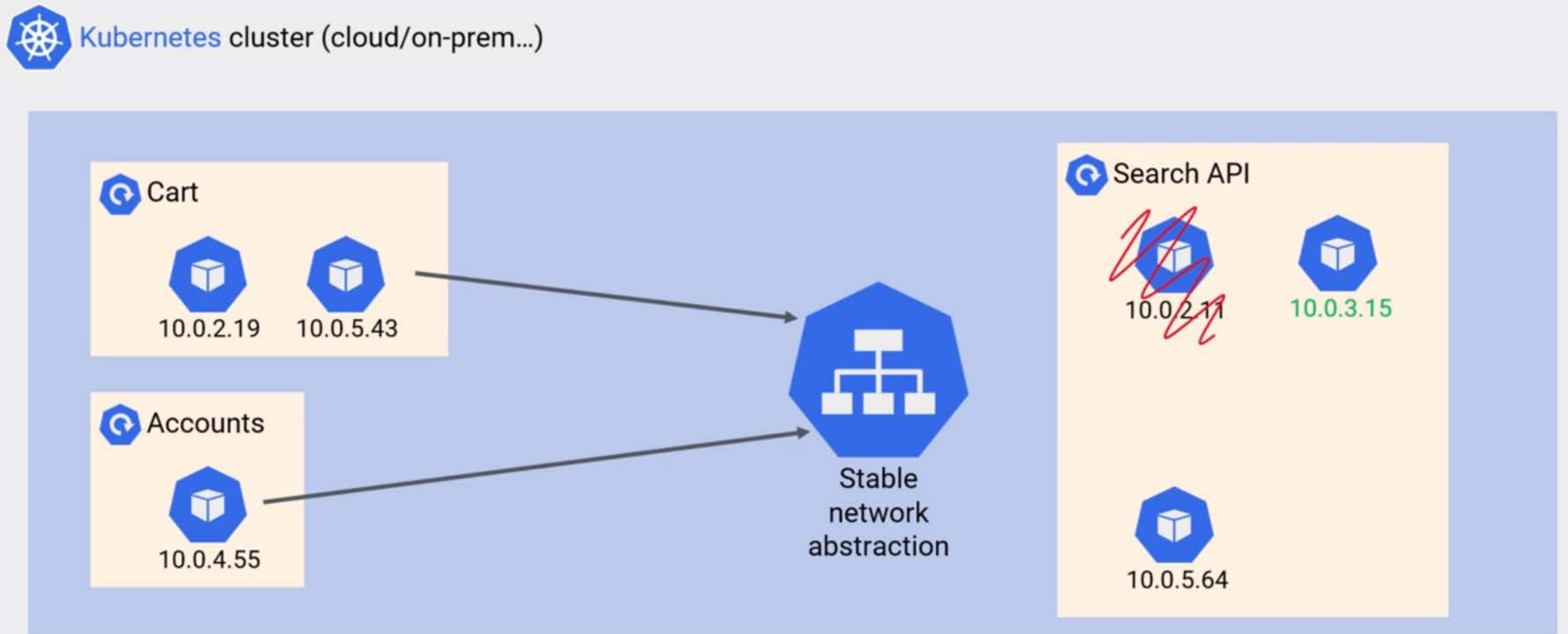


Pod network
CNI plugin
- Big flat network

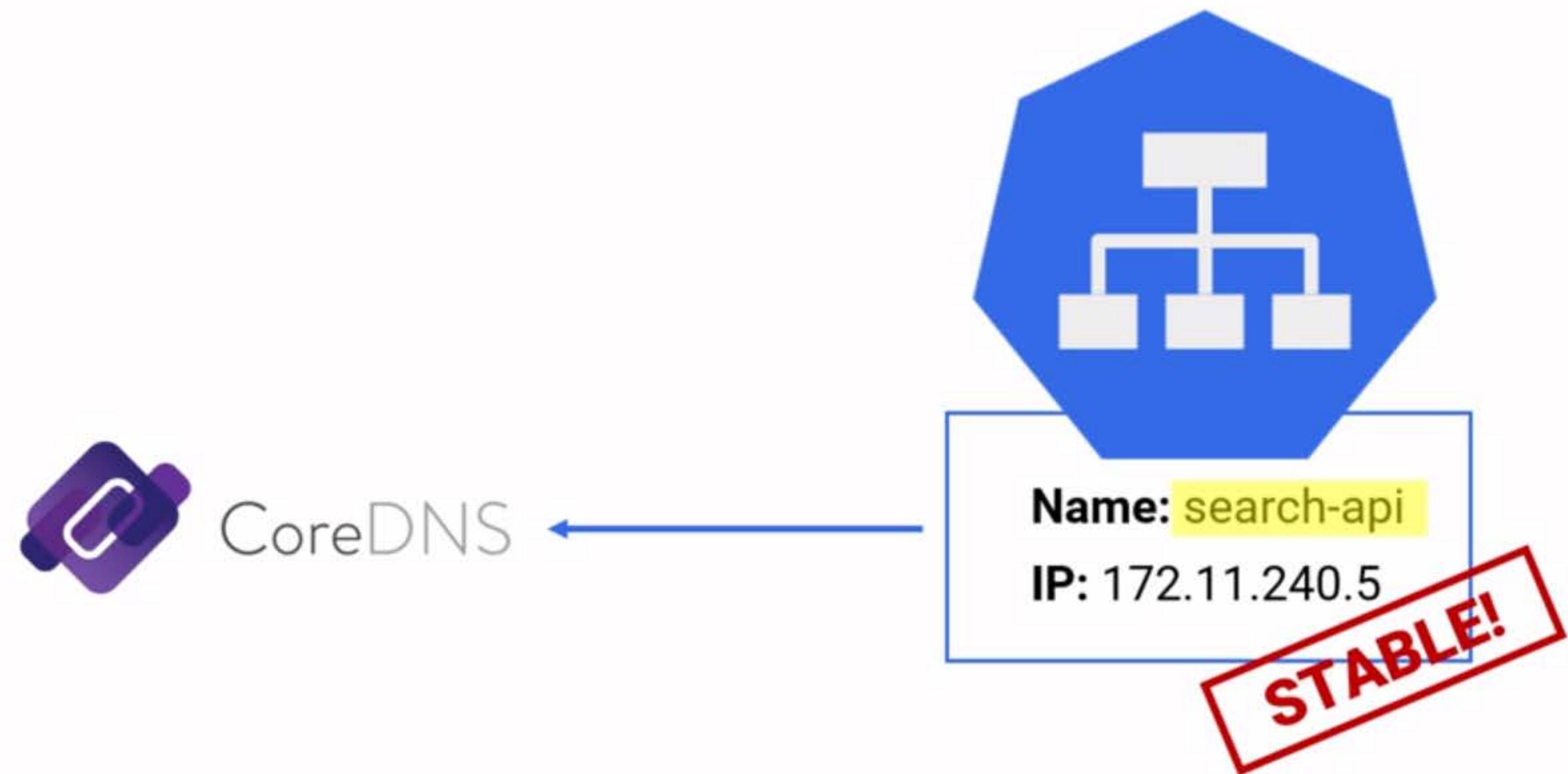
Node network
- 443 (HTTPS)
- Pod networking



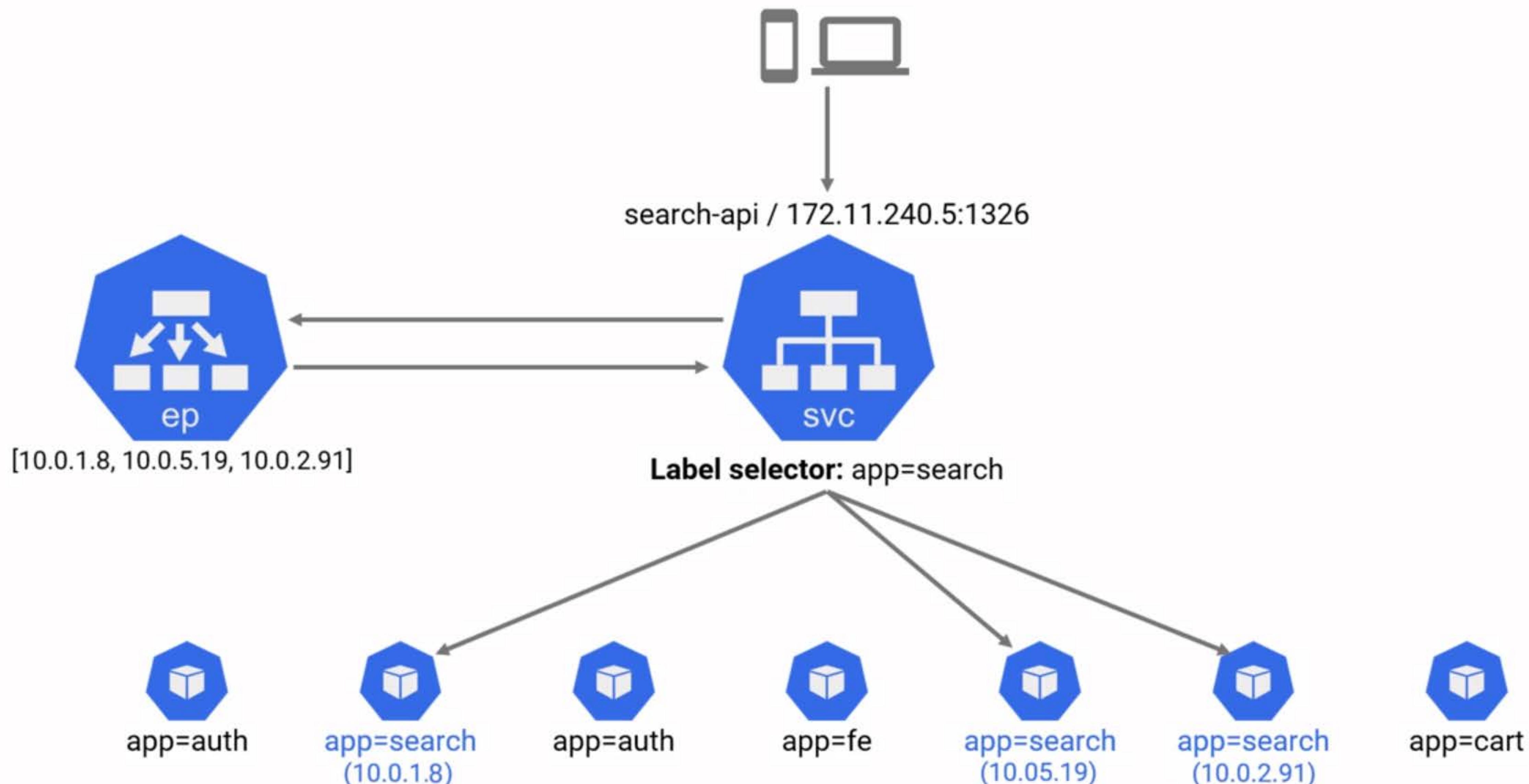
Kubernetes Services



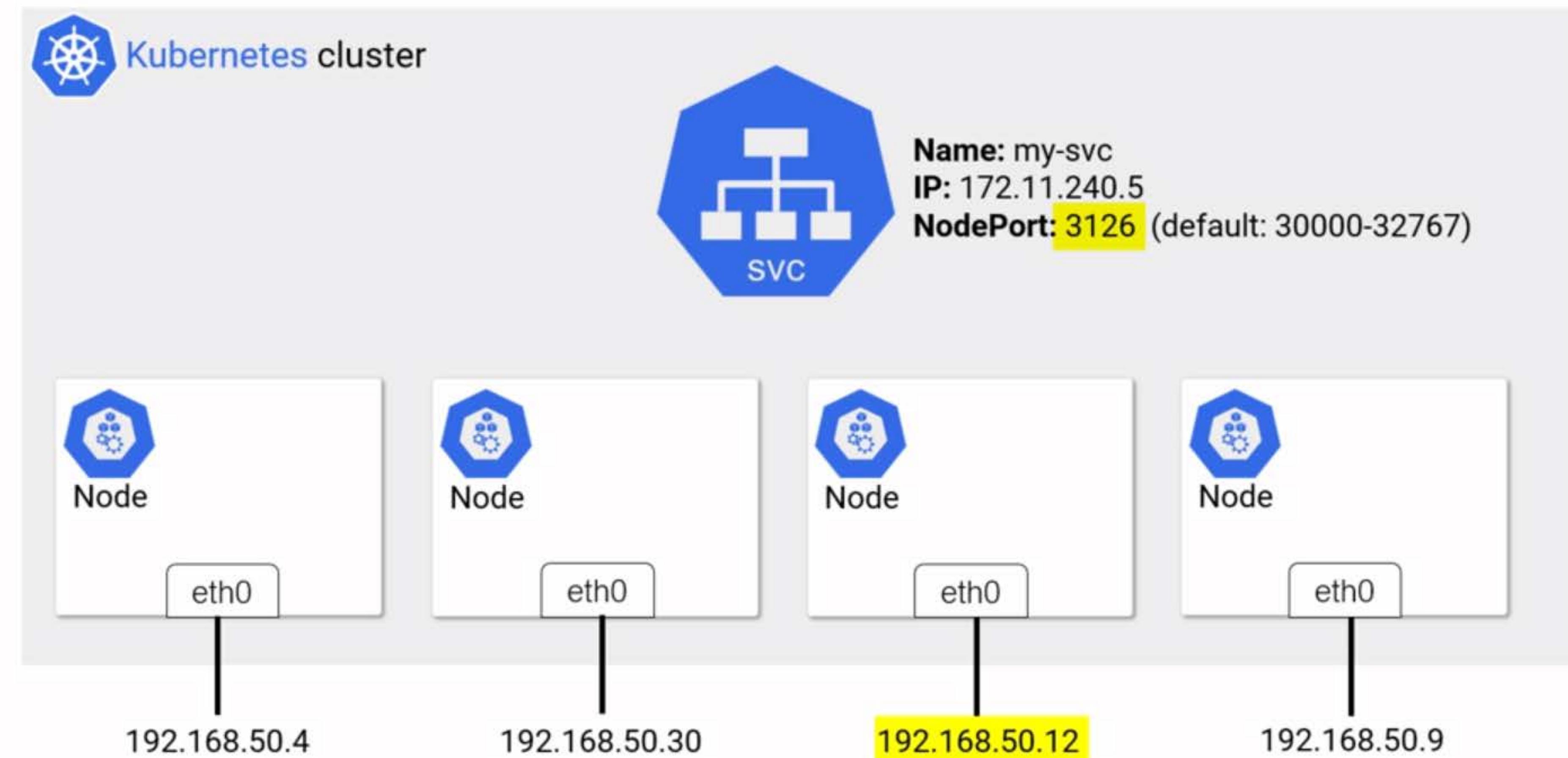
Kubernetes Services



Kubernetes Services



Kubernetes Networking Basics



Kubernetes Services



LoadBalancer:

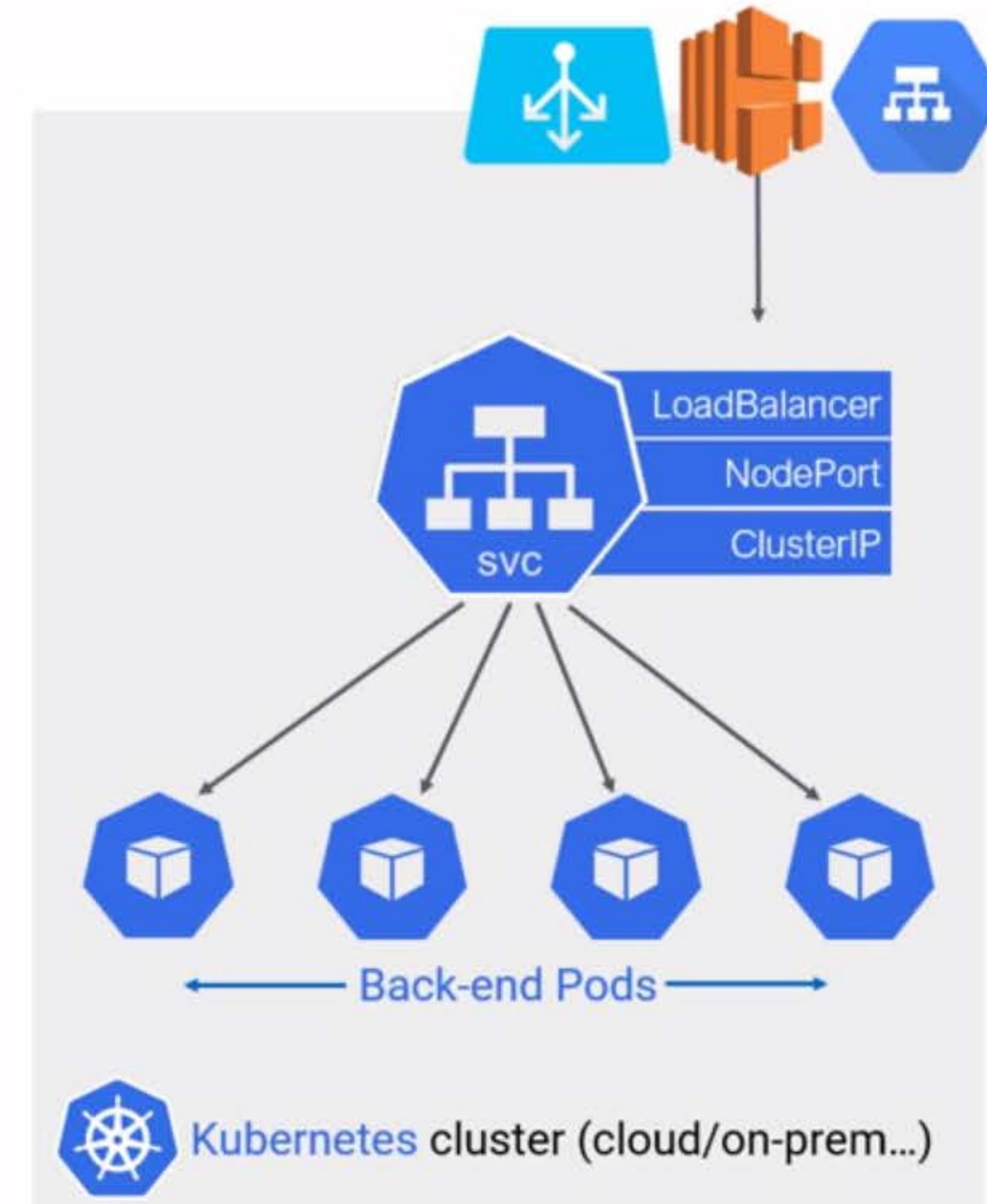
- Integrates with public cloud platform

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

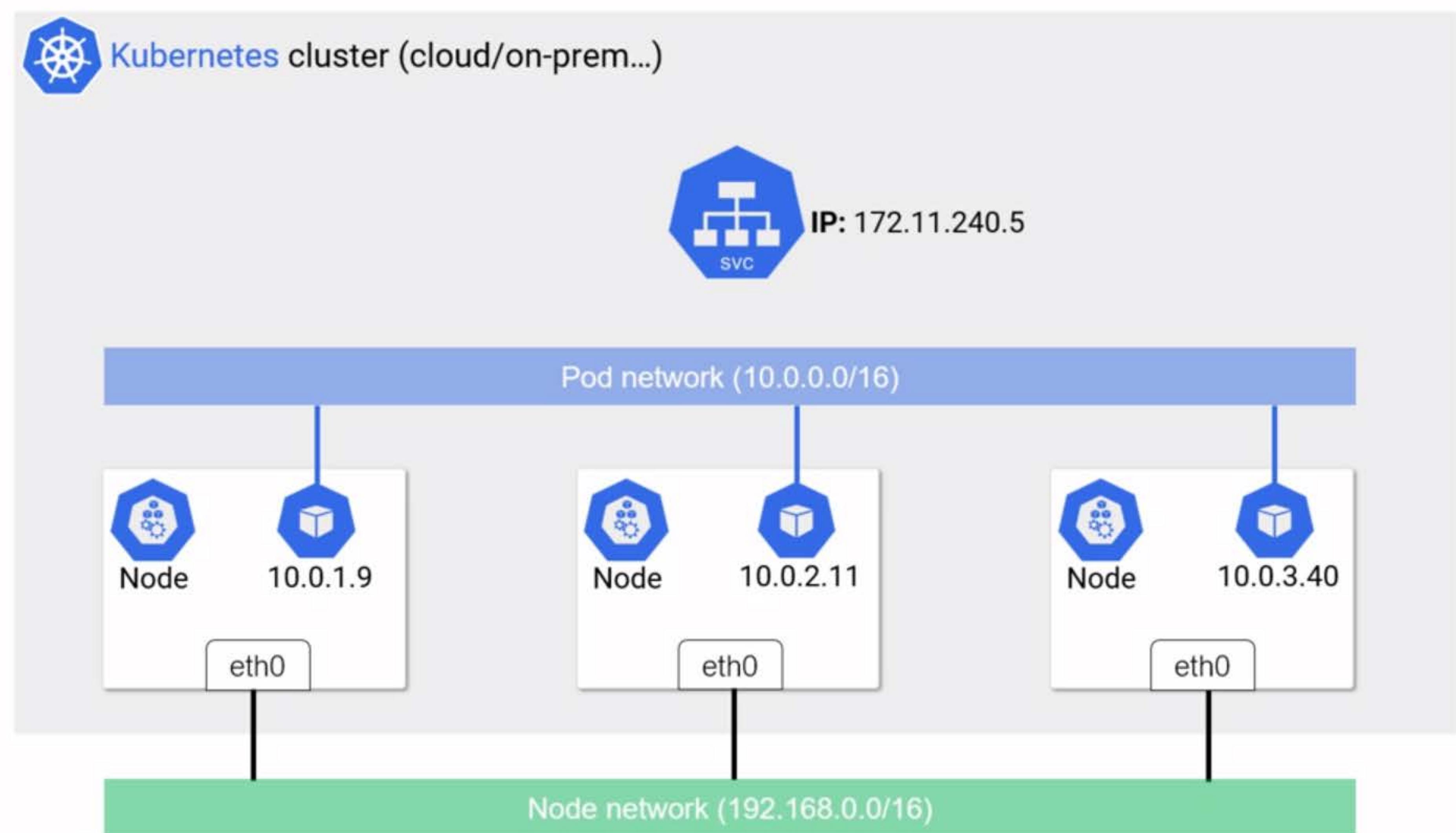
Node network

- 192.168.0.0/16

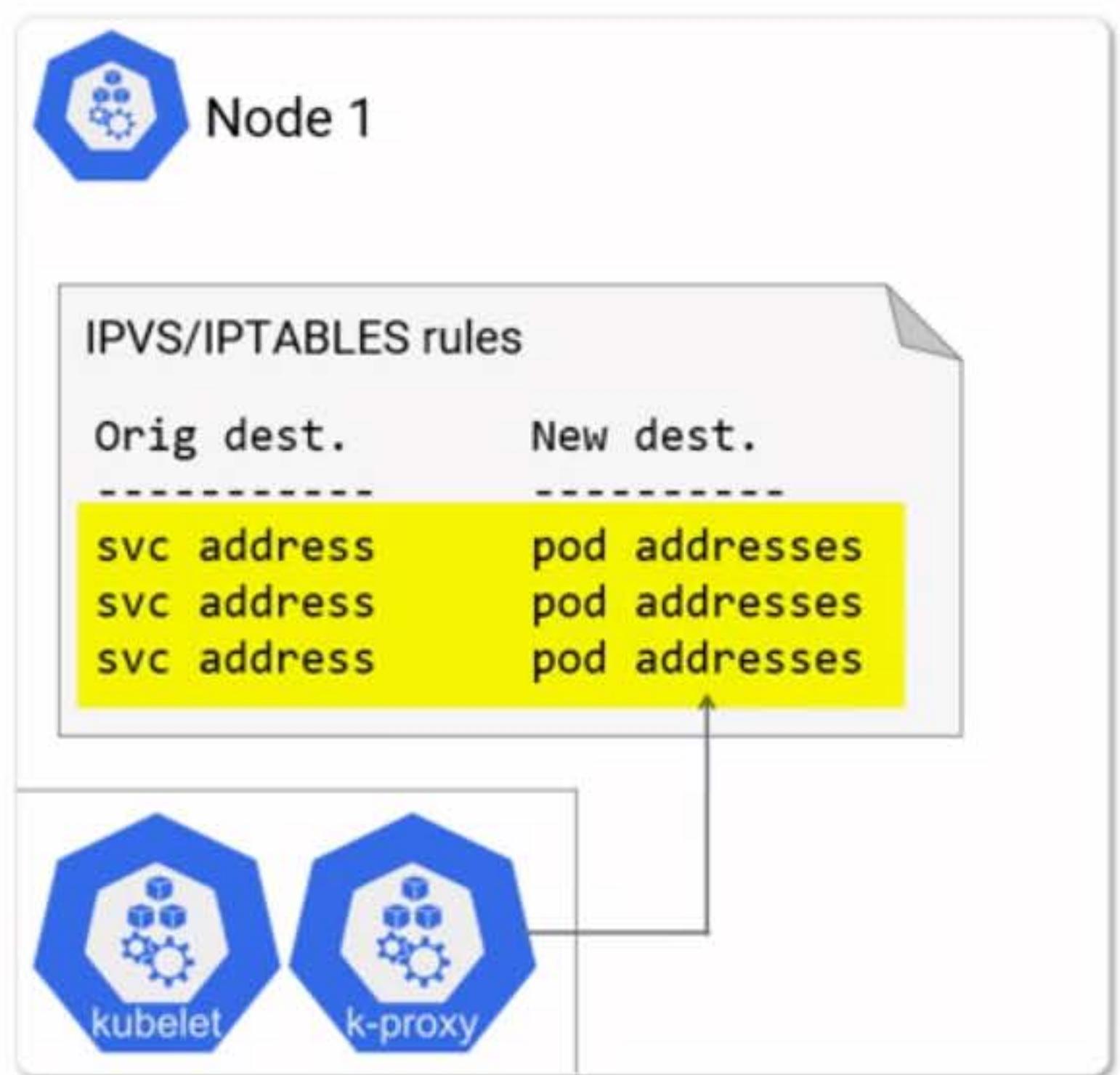
Service network

- 172.11.11.0/24

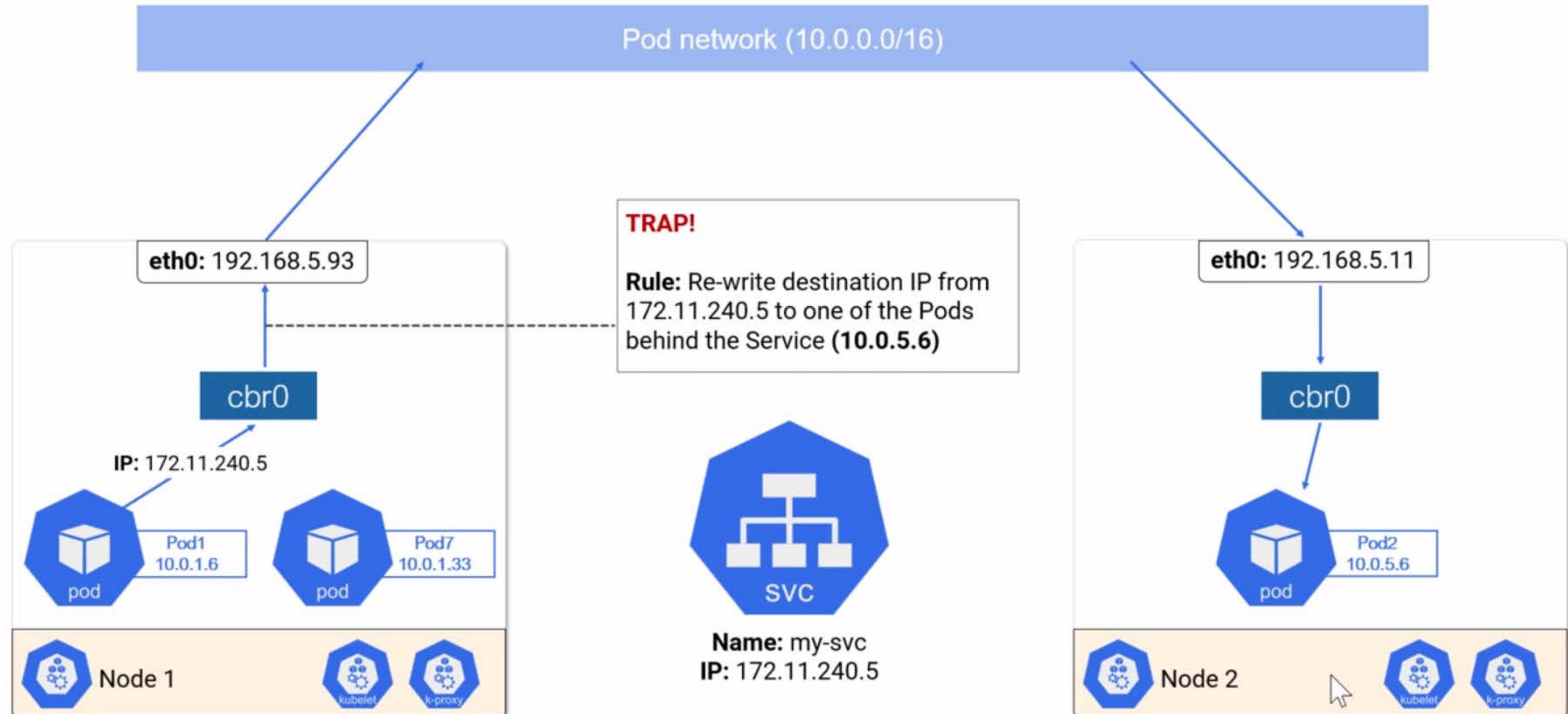
* Example network addresses for maximum clarity. Not the defaults that Kubernetes uses.



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 \longleftrightarrow API Server
- Pod network ports

Not implemented by Kubernetes

Pod Network

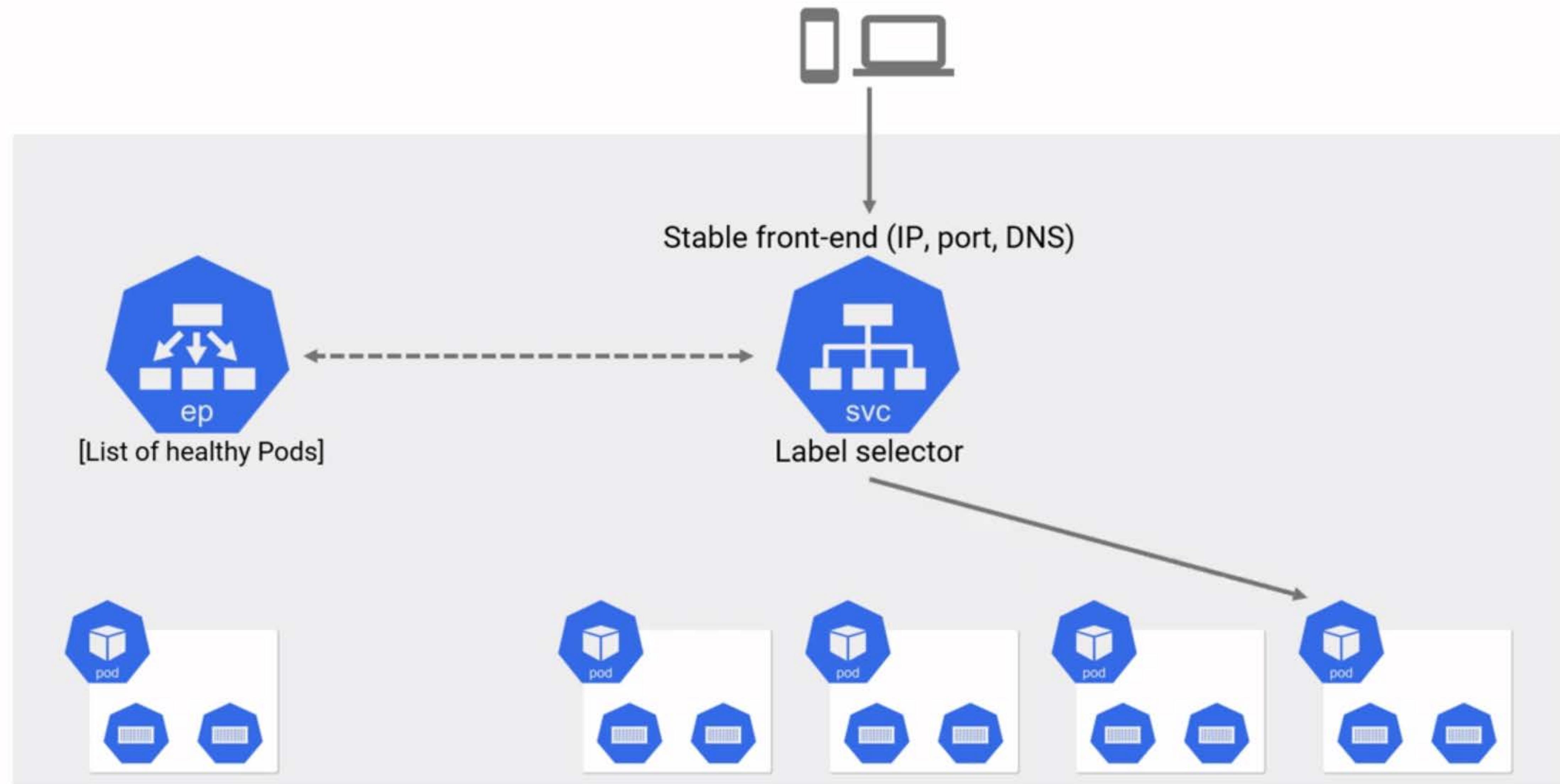
Implemented via CNI plugins

Big & flat

IP-per-Pod

All Pods can communicate

Recap



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)

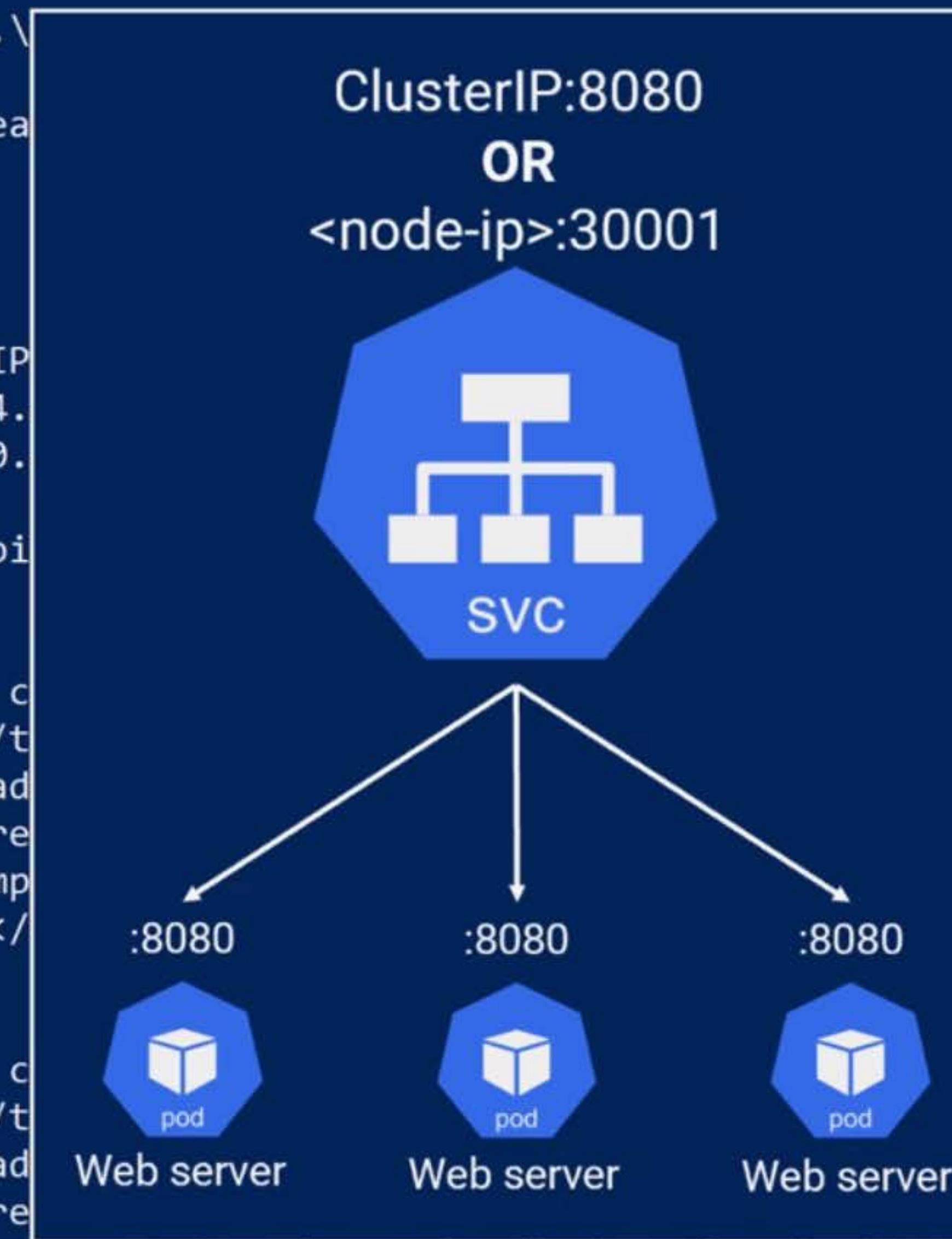
Prot LocalAddress:Port Scheduler Flags

	-> 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.

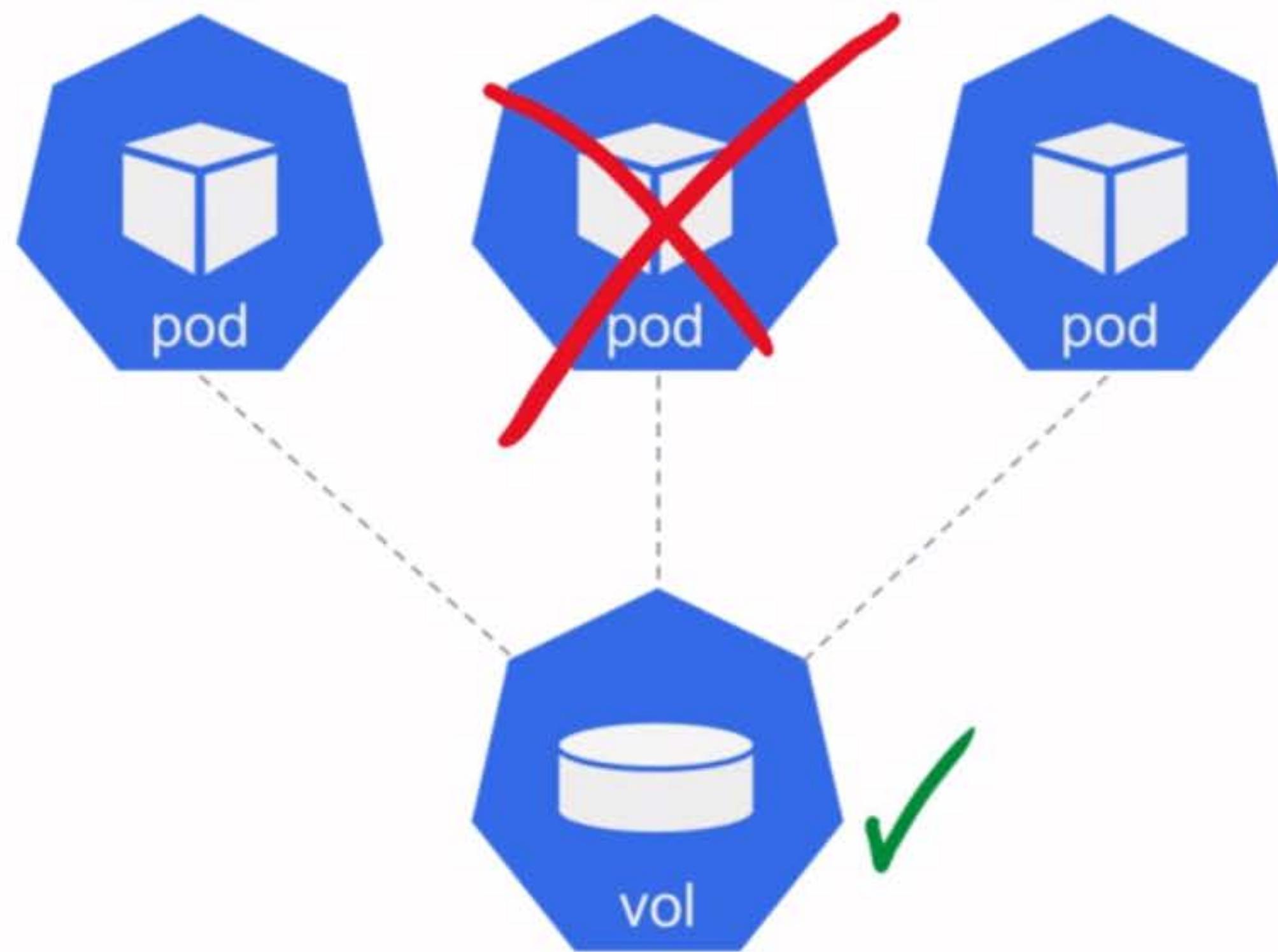
```
root@pingtest-6769c69dd5-h2x76: /
```

```
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:#
<html><head><title>ACG loves K8S</t
.1.1/css/bootstrap.min.css"/></head>
Kubernetes!!!</h1><p></p><p> <a href
_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:#
<html><head><title>ACG loves K8S</t
.1.1/css/bootstrap.min.css"/></head>
Kubernetes!!!</h1><p></p><p> <a href
_1_3?ie=UTF8&#amp;qid=1531240306&#amp;sr=8-3&#amp;keywords=nigel+poulton" class="btn btn-primary">Kubernetes Book</a></p><p></p></div></div><
```



in the cluster plus the NodePort value.

High-level Storage Requirements



High-level Storage Requirements



Storage is Vital!

File & Block
First-class Citizens in
Kubernetes

High-level Storage Requirements



File & Block
First-class Citizens in
Kubernetes

- Standards-based
- Pluggable backend
- Rich API



High-level Storage Requirements

Fundamental storage requirements

Storage Backend

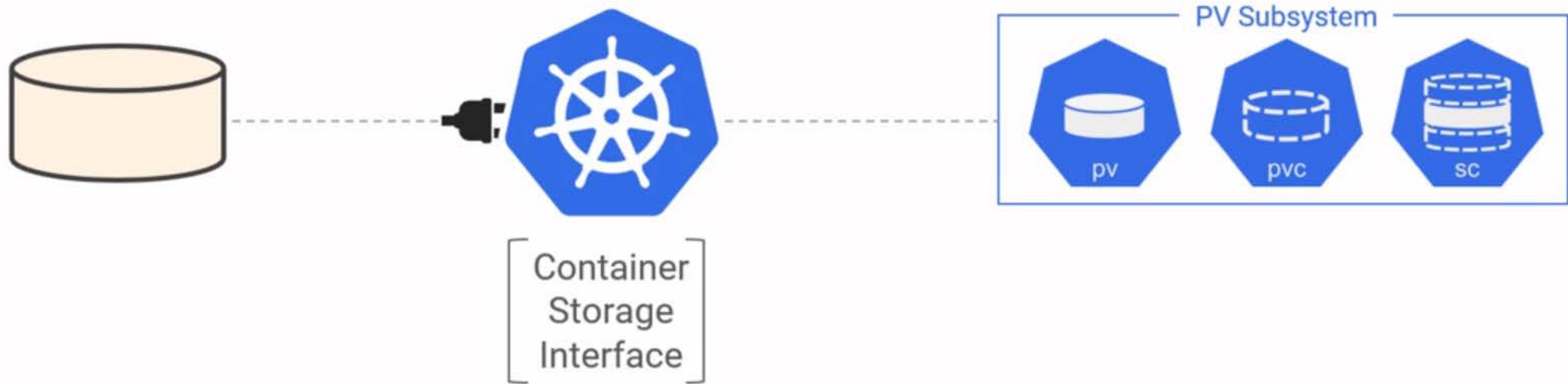
Speed

Replicated

Resiliency

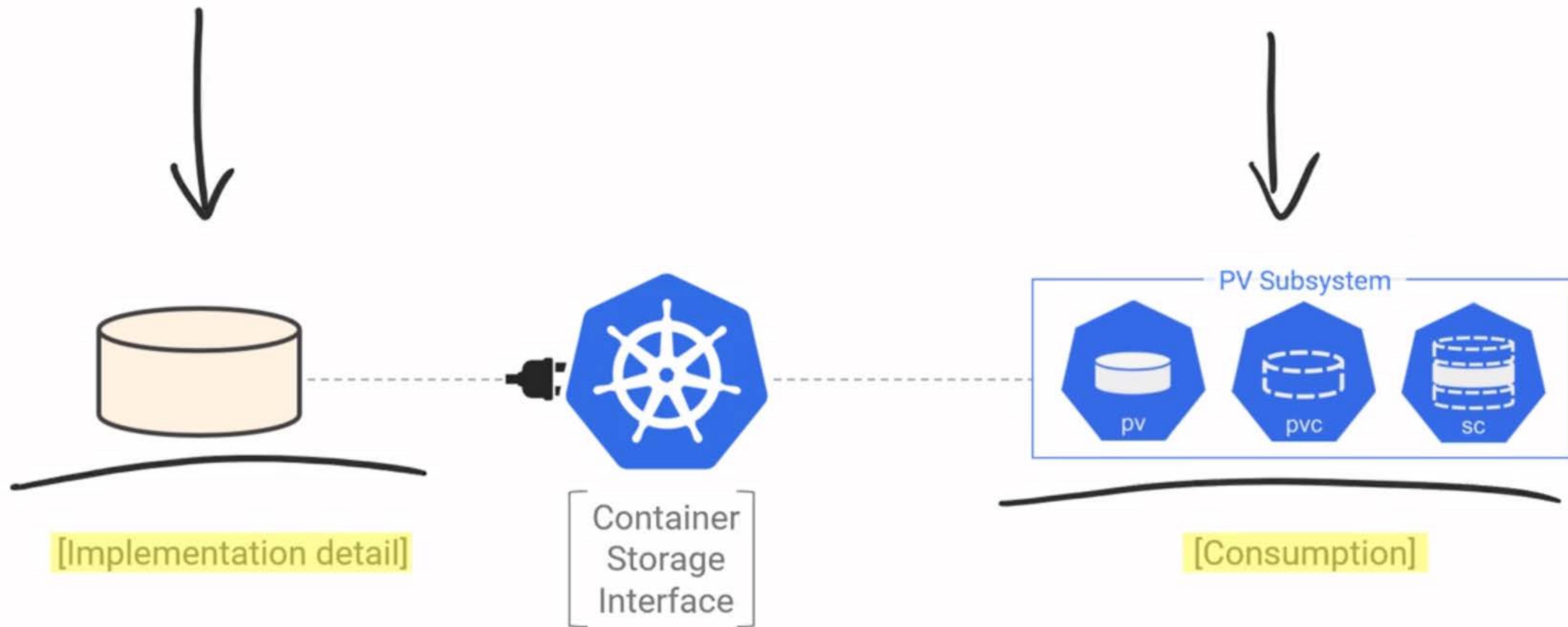
...

High-level Storage Requirements

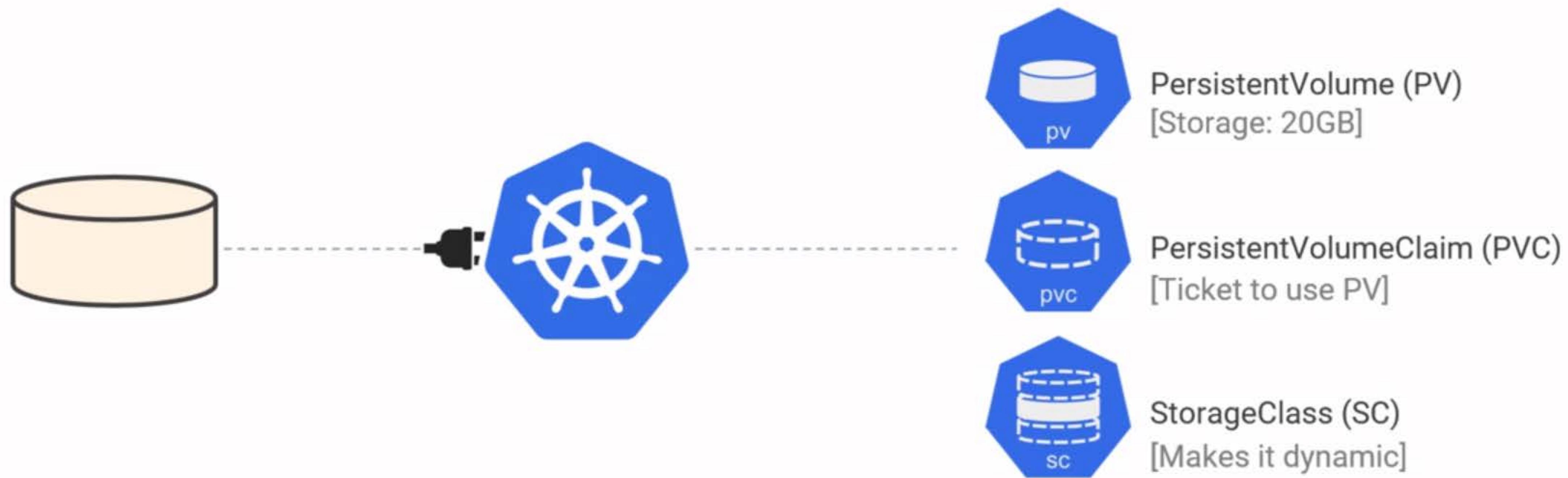




High-level Storage Requirements



High-level Storage Requirements



Container Storage Interface (CSI)



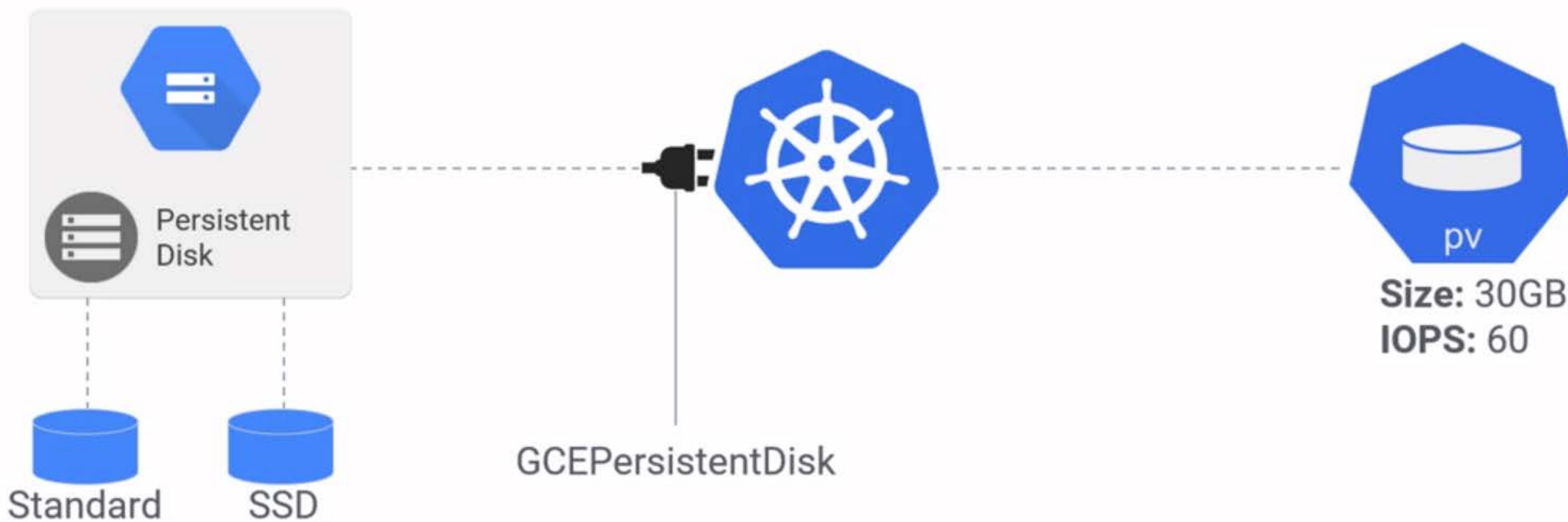
@nigelpoulton



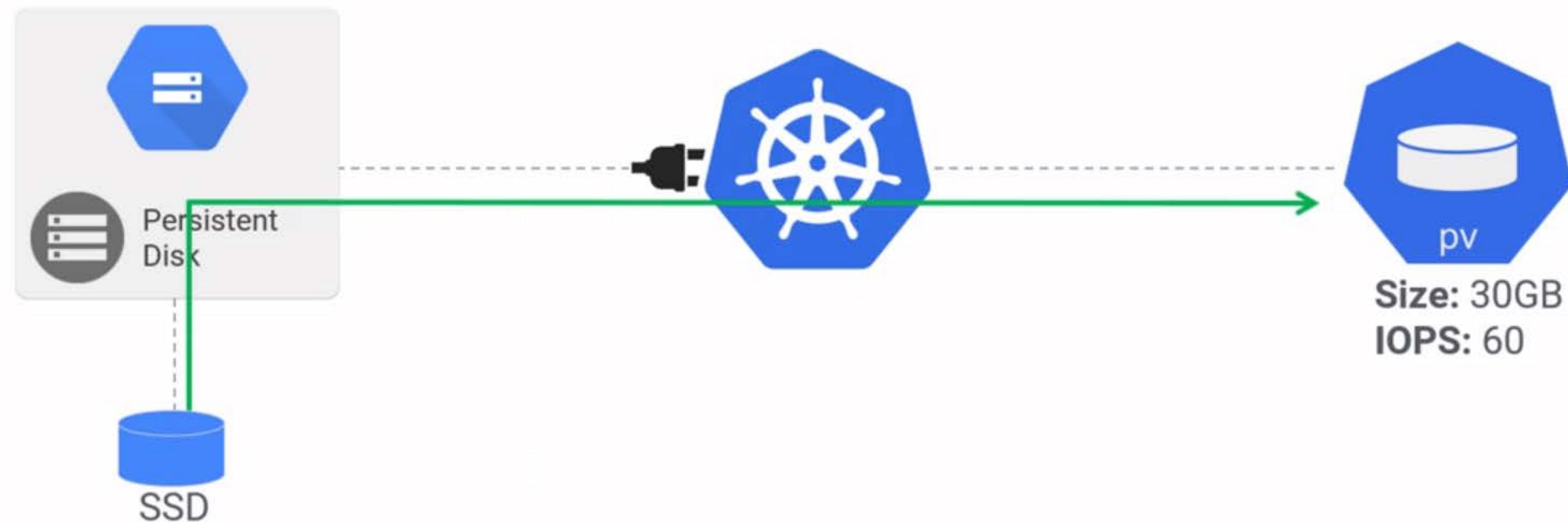
<https://github.com/container-storage-interface/spec>

Still Beta in K8s 1.11

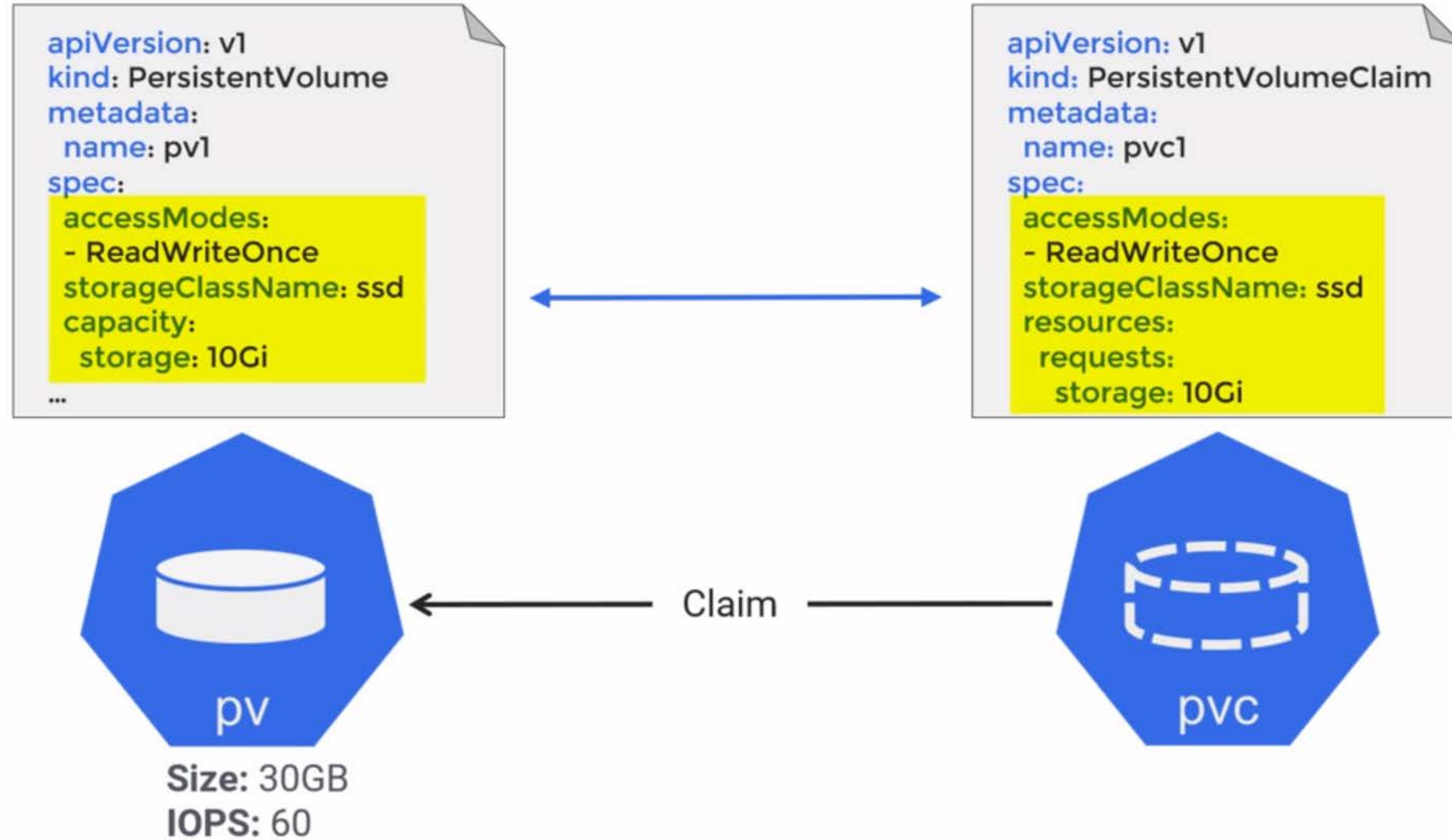
The K8s PersistentVolume Subsystem



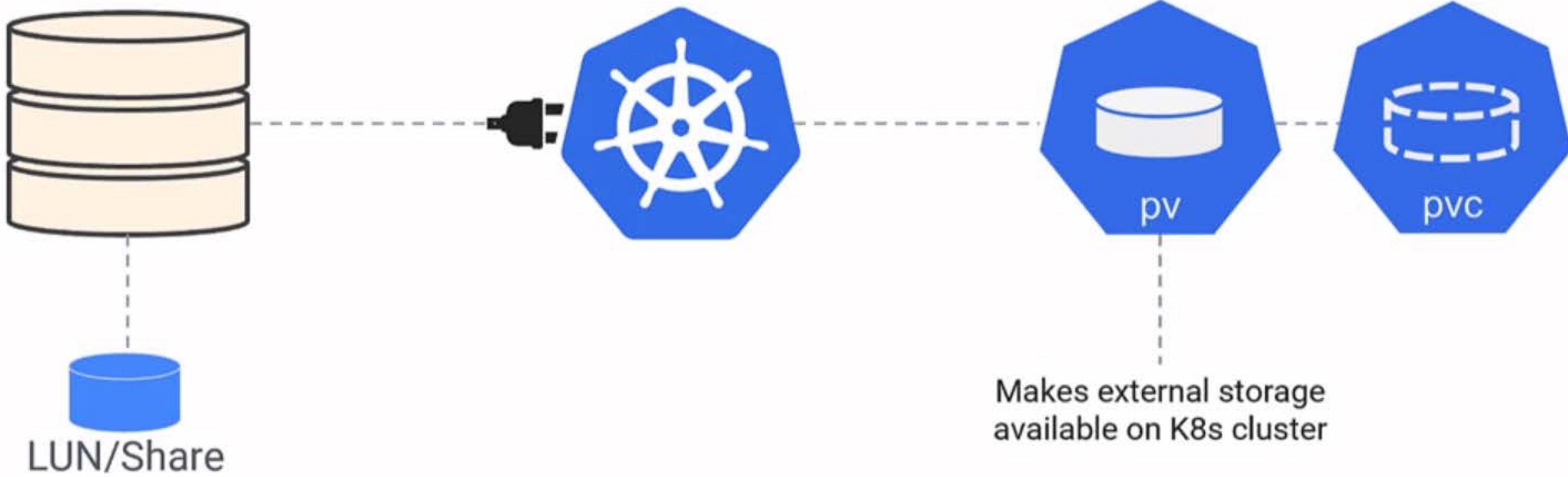
The K8s PersistentVolume Subsystem



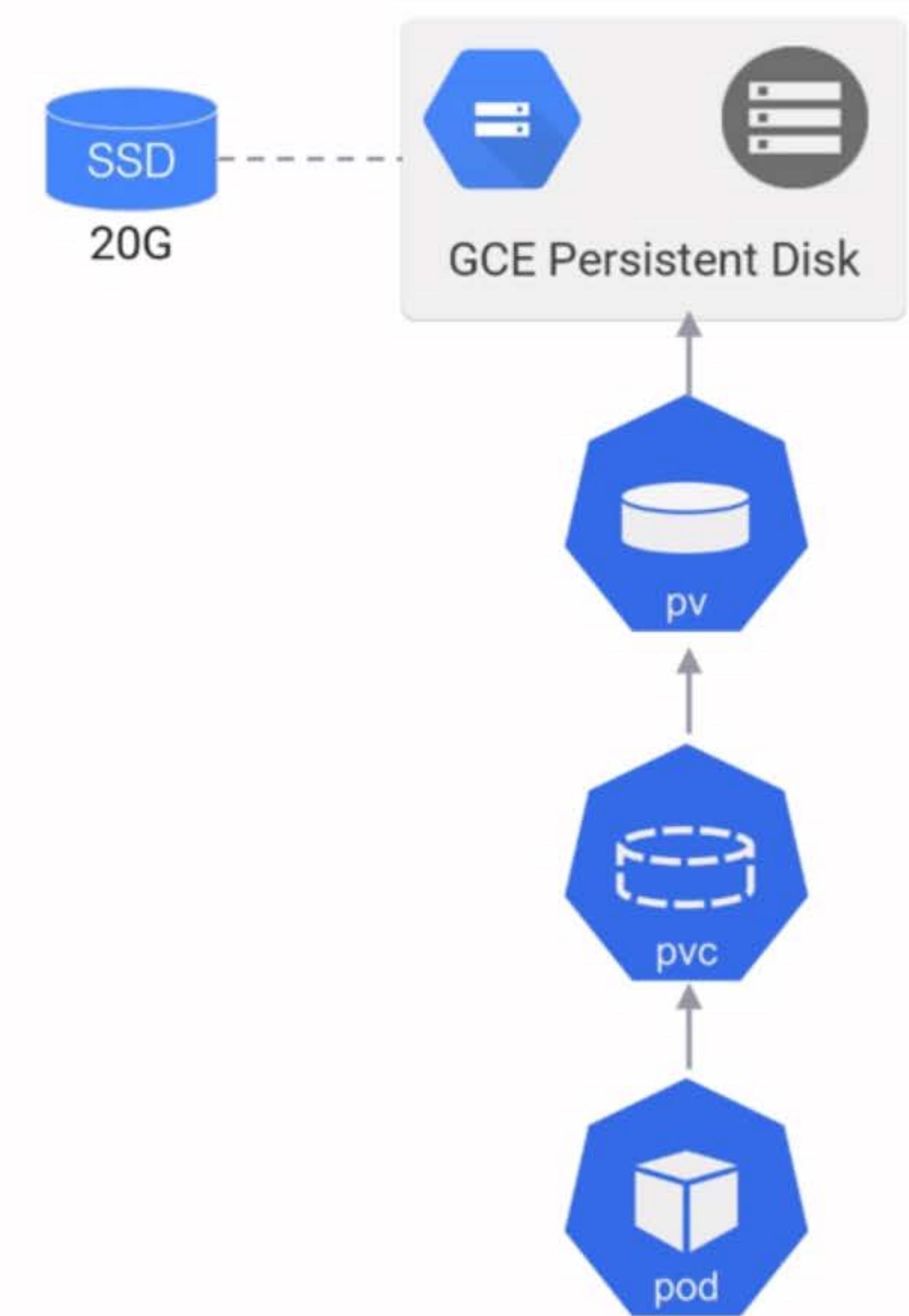
The K8s PersistentVolume Subsystem



```
1 apiVersion: v1
2 kind: PersistentVolumeClaim
3 metadata:
4   name: pvc1
5 spec:
6   accessModes:
7     - ReadWriteOnce
8   storageClassName: ssd
9 resources:
10  requests:
11    storage: 20Gi
```



```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: volpod
5  spec:
6    volumes:
7      - name: data
8        persistentVolumeClaim:
9          claimName: pvc1
10   containers:
11     - image: ubuntu:latest
12       name: ubuntu-ctr
13       command:
14         - /bin/bash
15         - "-c"
16         - "sleep 60m"
17       imagePullPolicy: IfNotPresent
18       volumeMounts:
19         - mountPath: /data
20           name: data
```



```
ubuntu@mgr1:~$  
ubuntu@mgr1:~$ cat aws-pvc.yml  
apiVersion: v1  
kind: PersistentVolumeClaim  
metadata:  
  name: aws-fast  
spec:  
  accessModes:  
    - ReadWriteOnce  
  storageClassName: fast  
resources:  
  requests:  
    storage: 10Gi
```

```
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  
aws-fast   Bound       pvc-3ec88edf-8ab6-11e8-8a01-060a650e58be
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
aws-fast	Bound	pvc-3ec88edf-8ab6-11e8-8a01-060a650e58be	10Gi	RWO	fast	26s
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					



Dynamically
create new PV

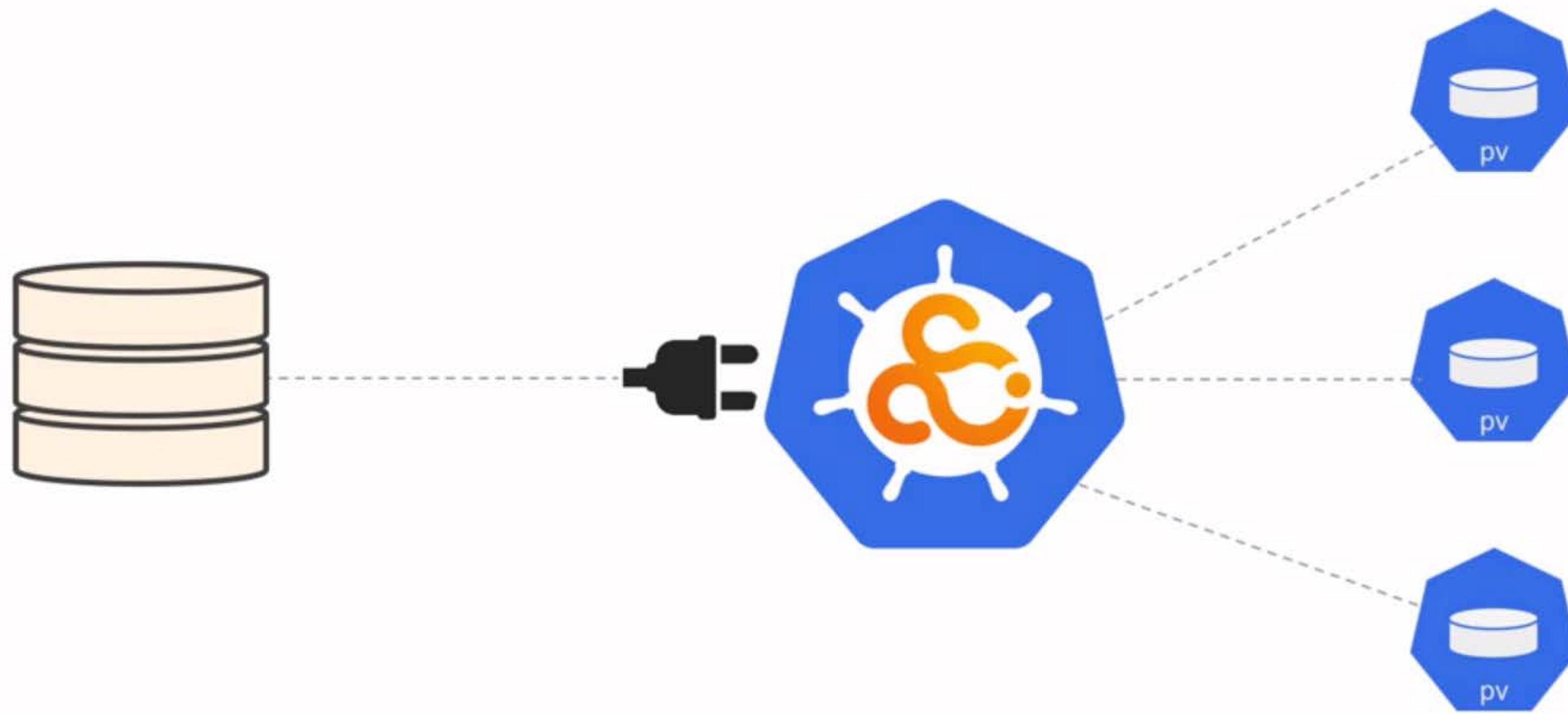


PV Subsystem
Control Loop

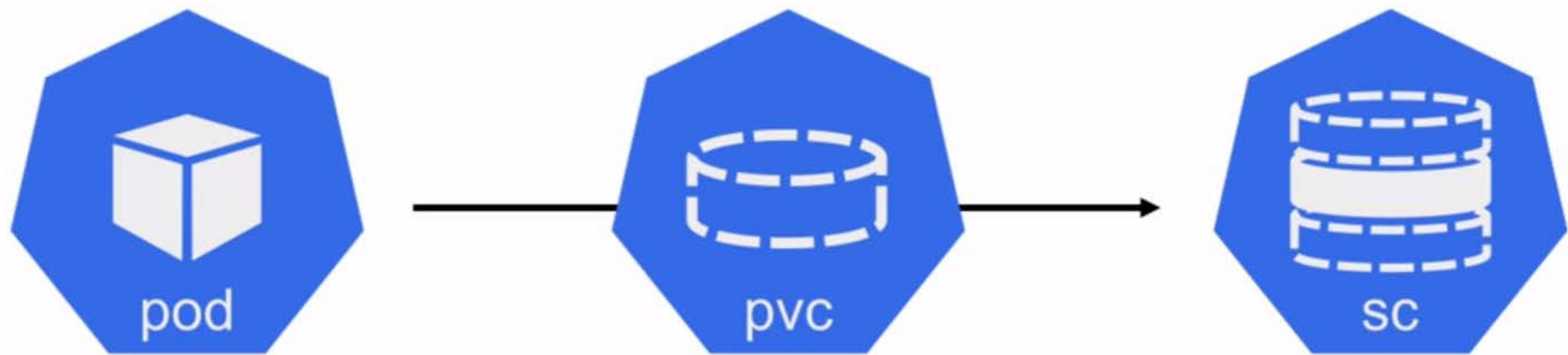
K8s API
Server

!!NEW PVC!!

Recap



Recap

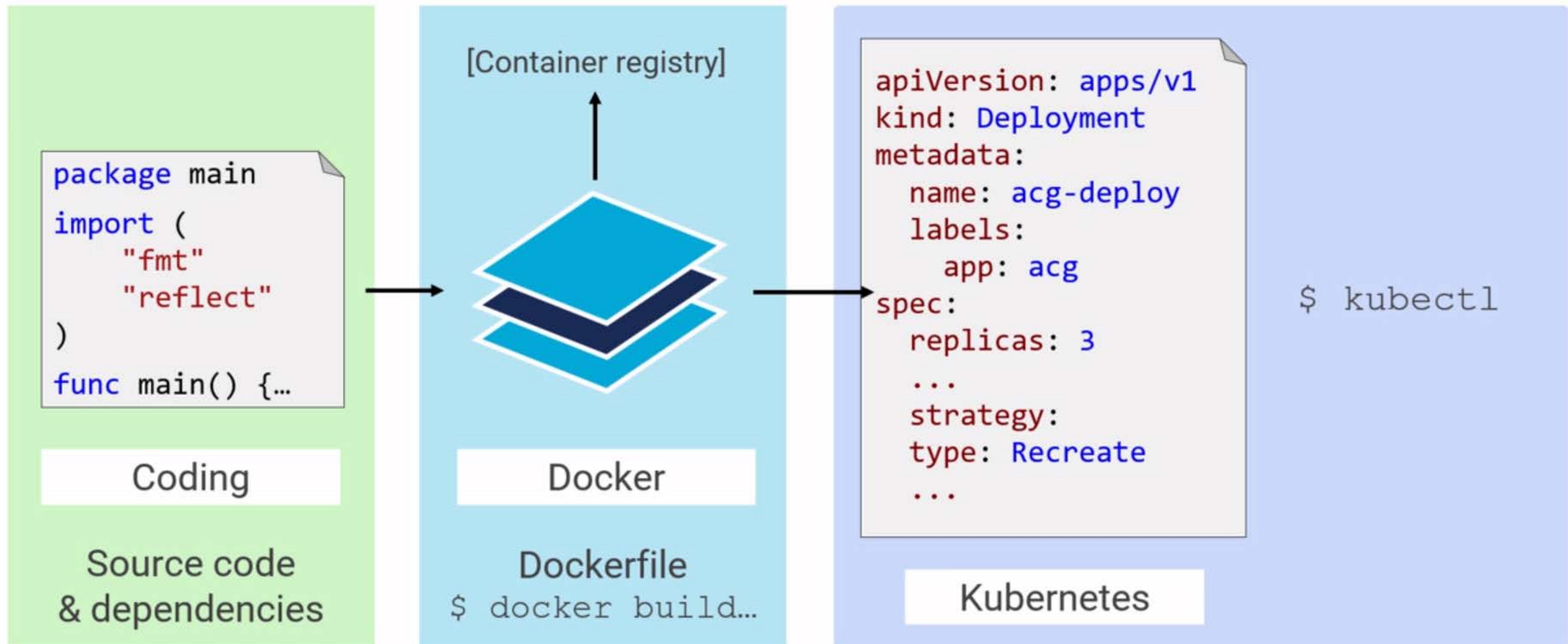


Making things
dynamic!

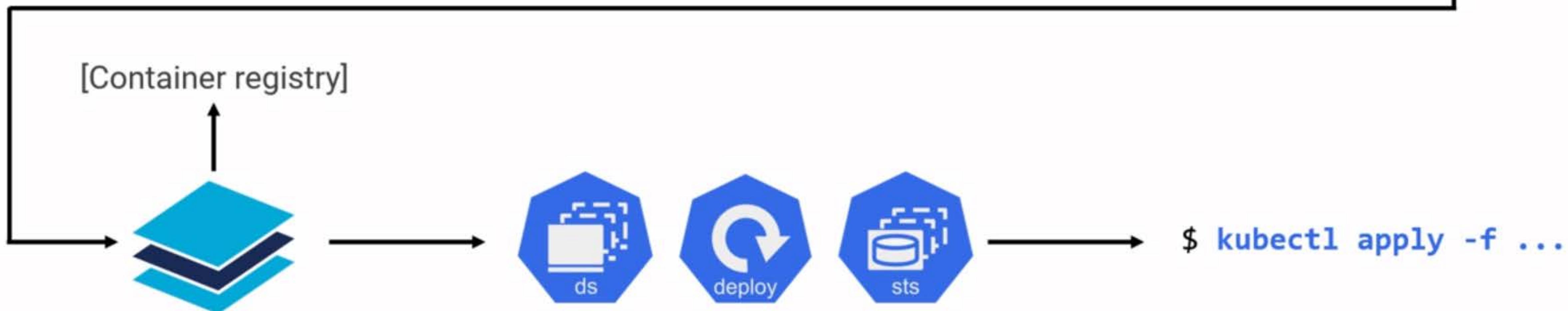
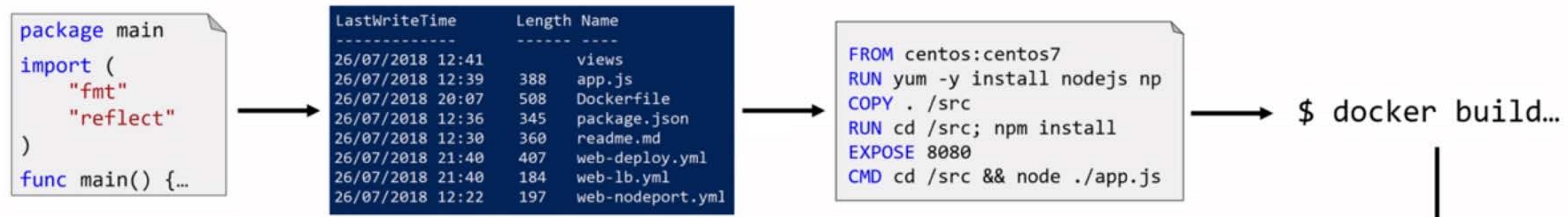
The Big Picture



From code to Kubernetes



Recap

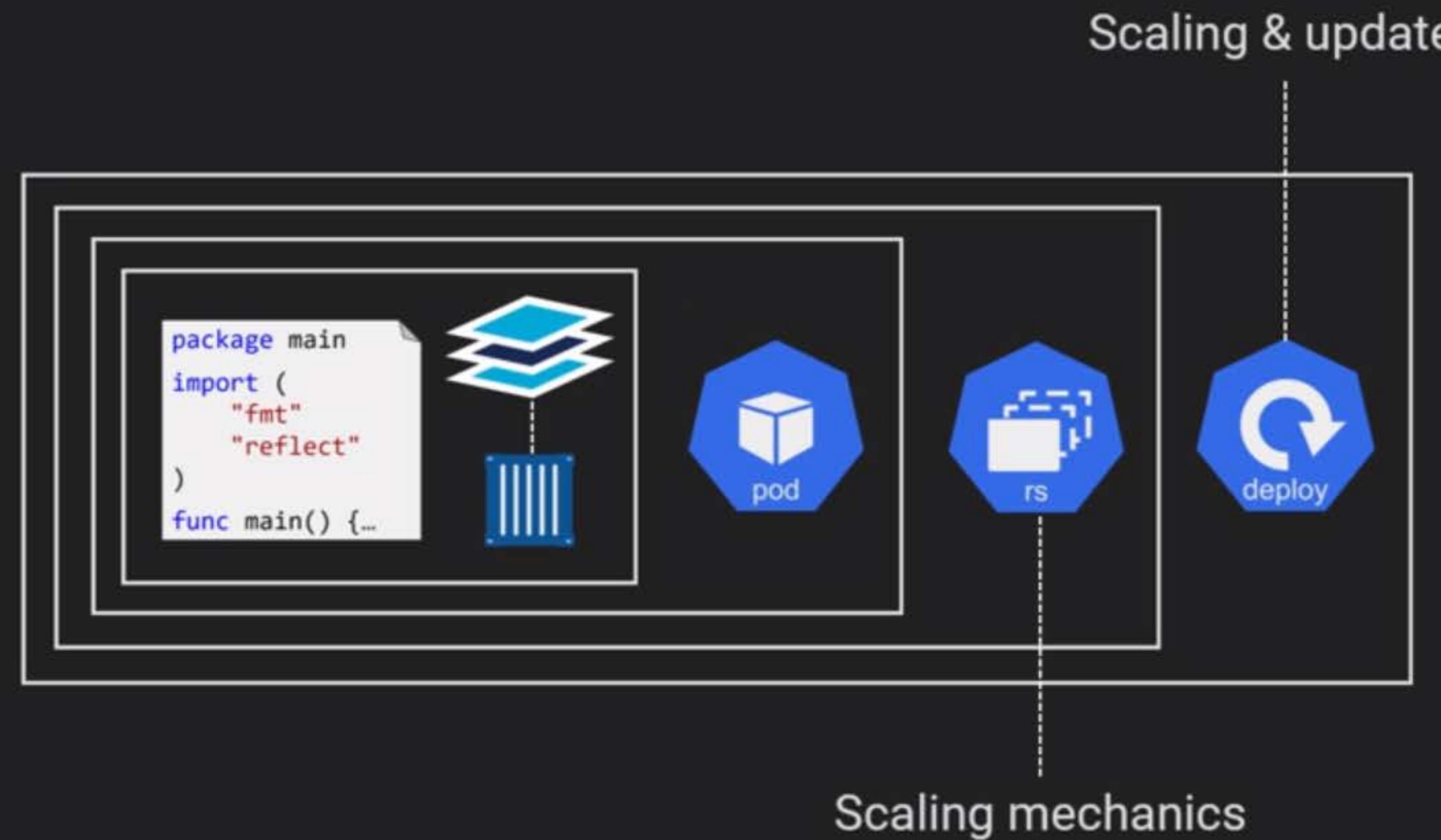


Deployment Theory

Press Esc to exit full screen

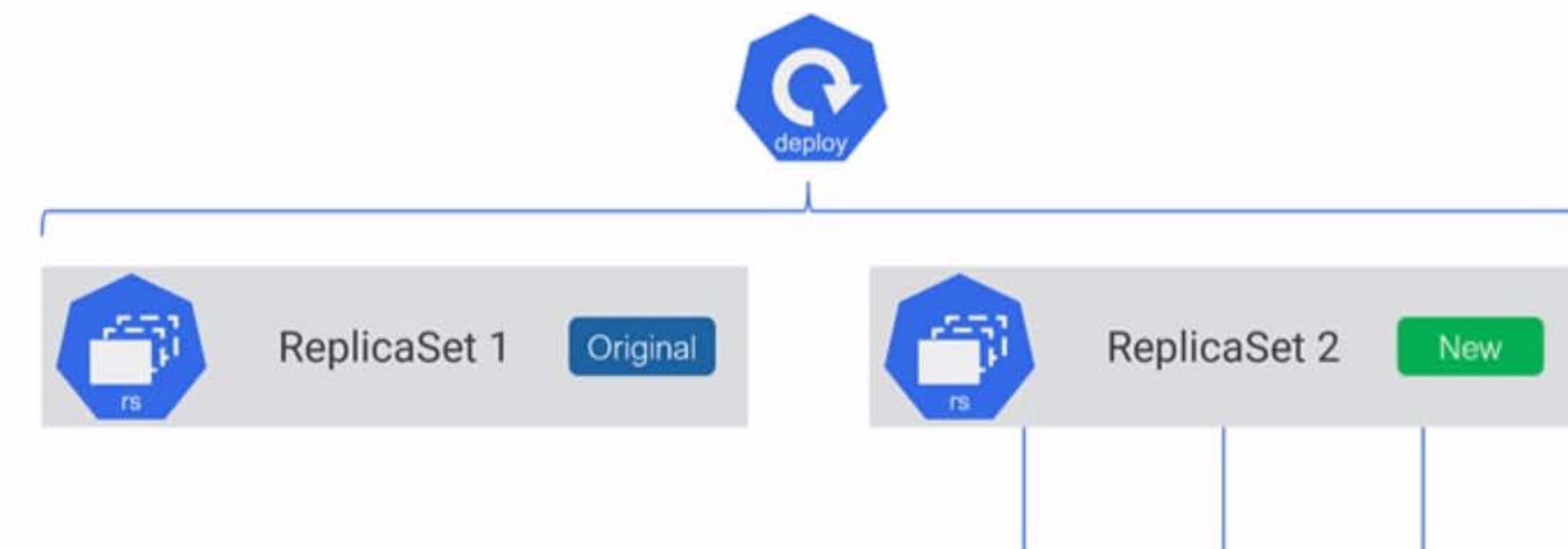


A CLOUD GURU



! deployment.yml

```
8   selector:  
9     matchLabels:  
10    run: prod-redis  
11  replicas: 3  
12  minReadySeconds: 300  
13  strategy:  
14    rollingUpdate:  
15      maxSurge: 1  
16      maxUnavailable: 0  
17      type: RollingUpdate  
18  template:  
19    metadata:  
20      labels:  
21        run: prod-redis  
22  spec:  
23    containers:  
24      - image: redis:4.11  
25        name: redis
```



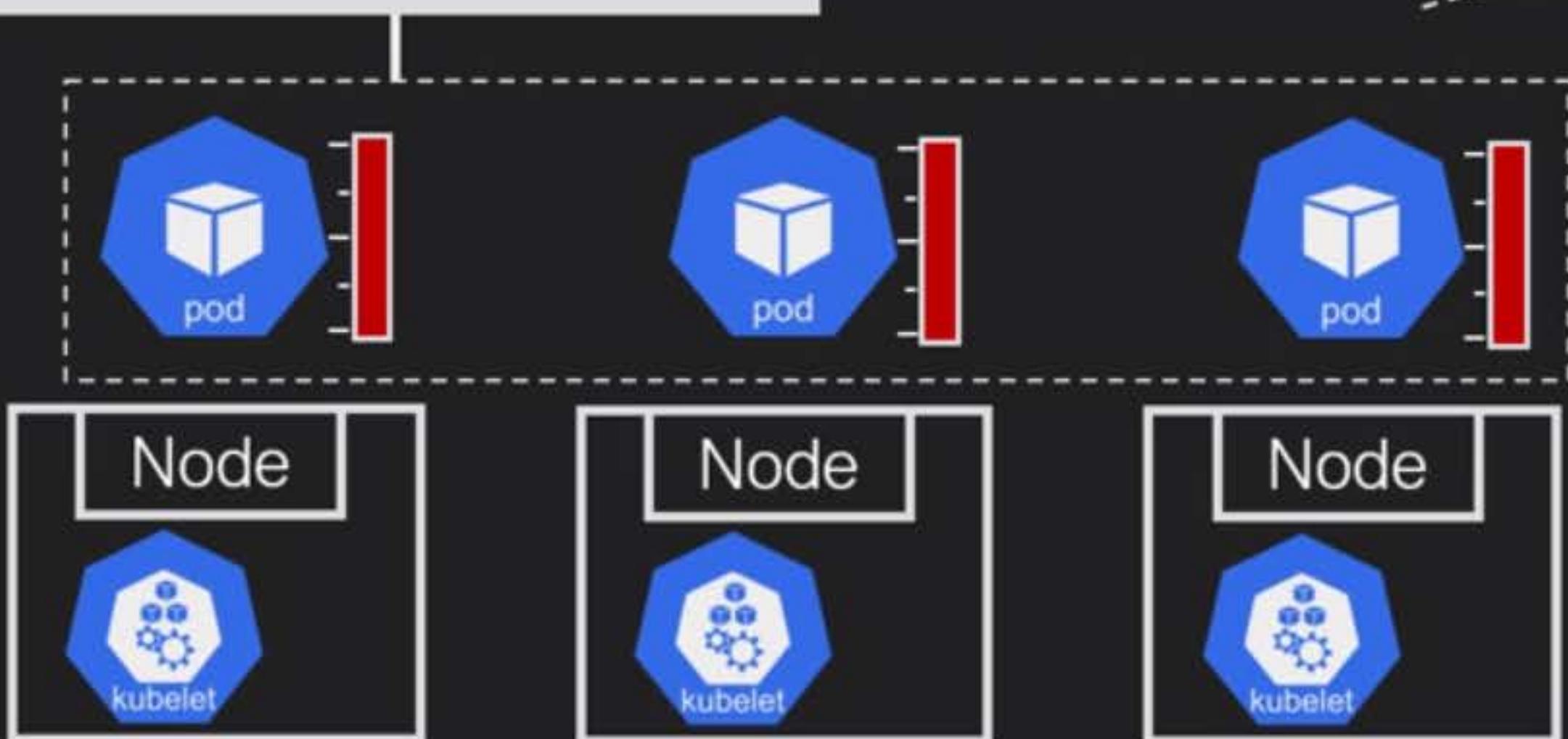
Big Picture



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

Scale to 4

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



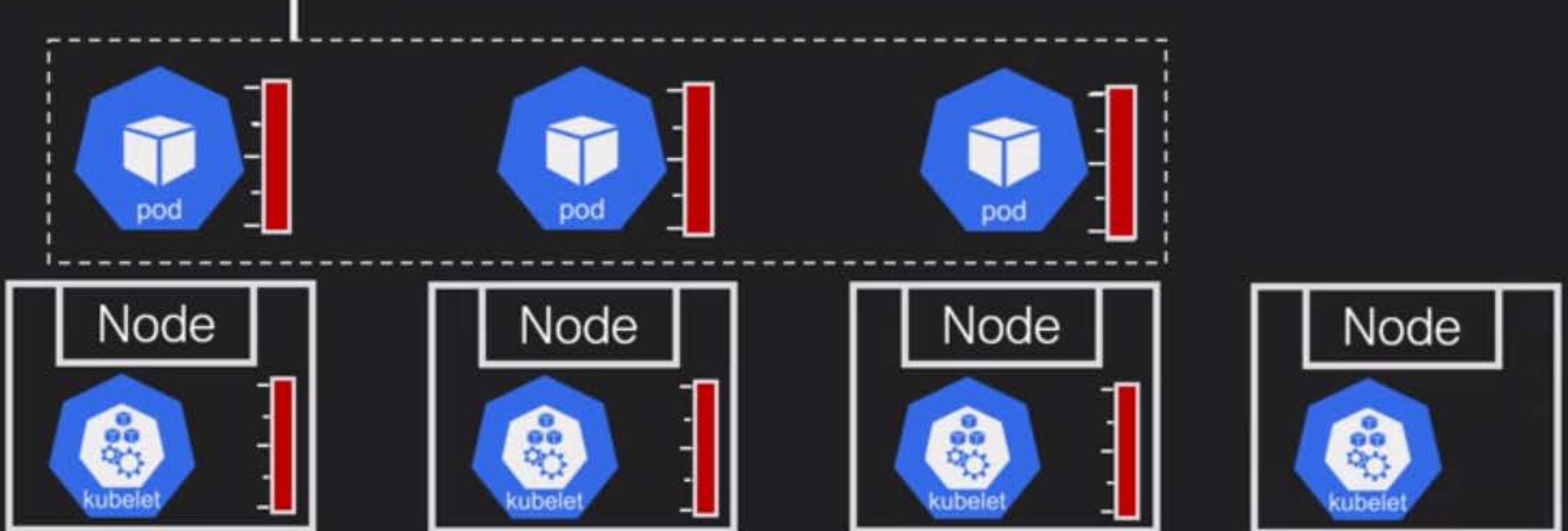
Infrastructure (On-premises/cloud)

Big Picture



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

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



Cluster
Autoscaler



Big Picture

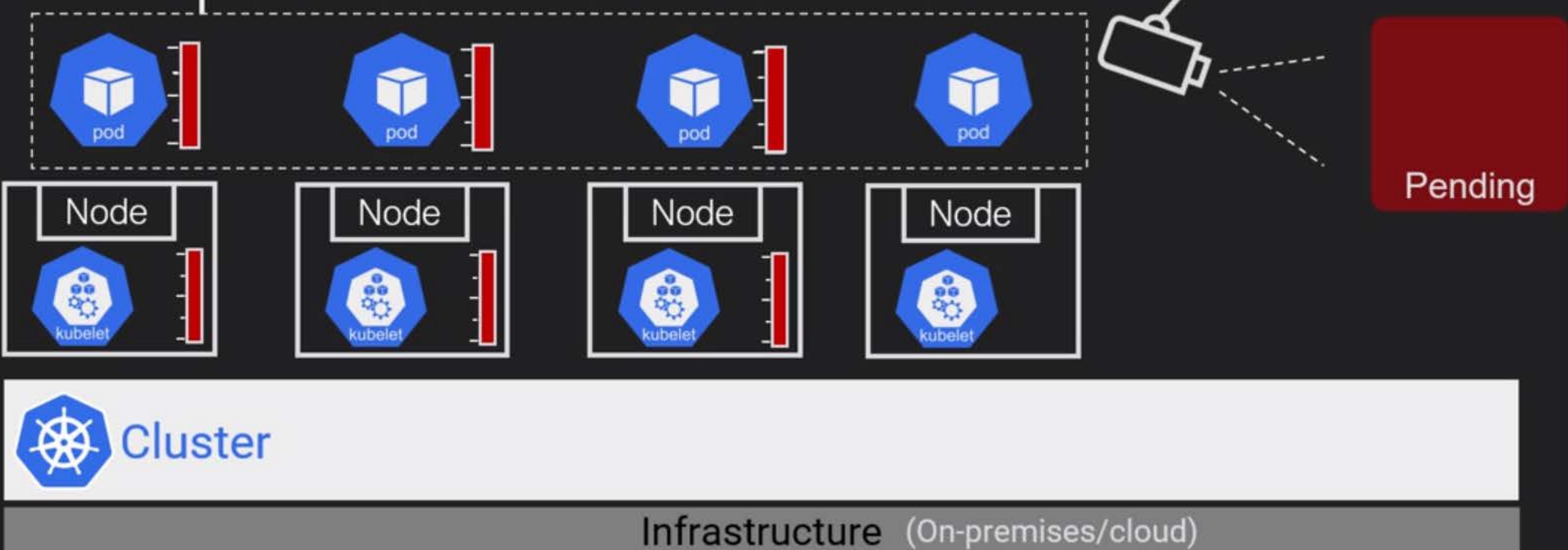


```
apiVersion: apps/v1
kind: Deployment
...
spec:
  replicas: 4
...
```

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

Cluster
Autoscaler

Pending



Horizontal Pod Autoscaler - Theory



Horizontal Pod Autoscaler



Horizontal Pod Autoscaler - Theory



```
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

```
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



```
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
```



```
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

```
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
```



```
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

```
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
```

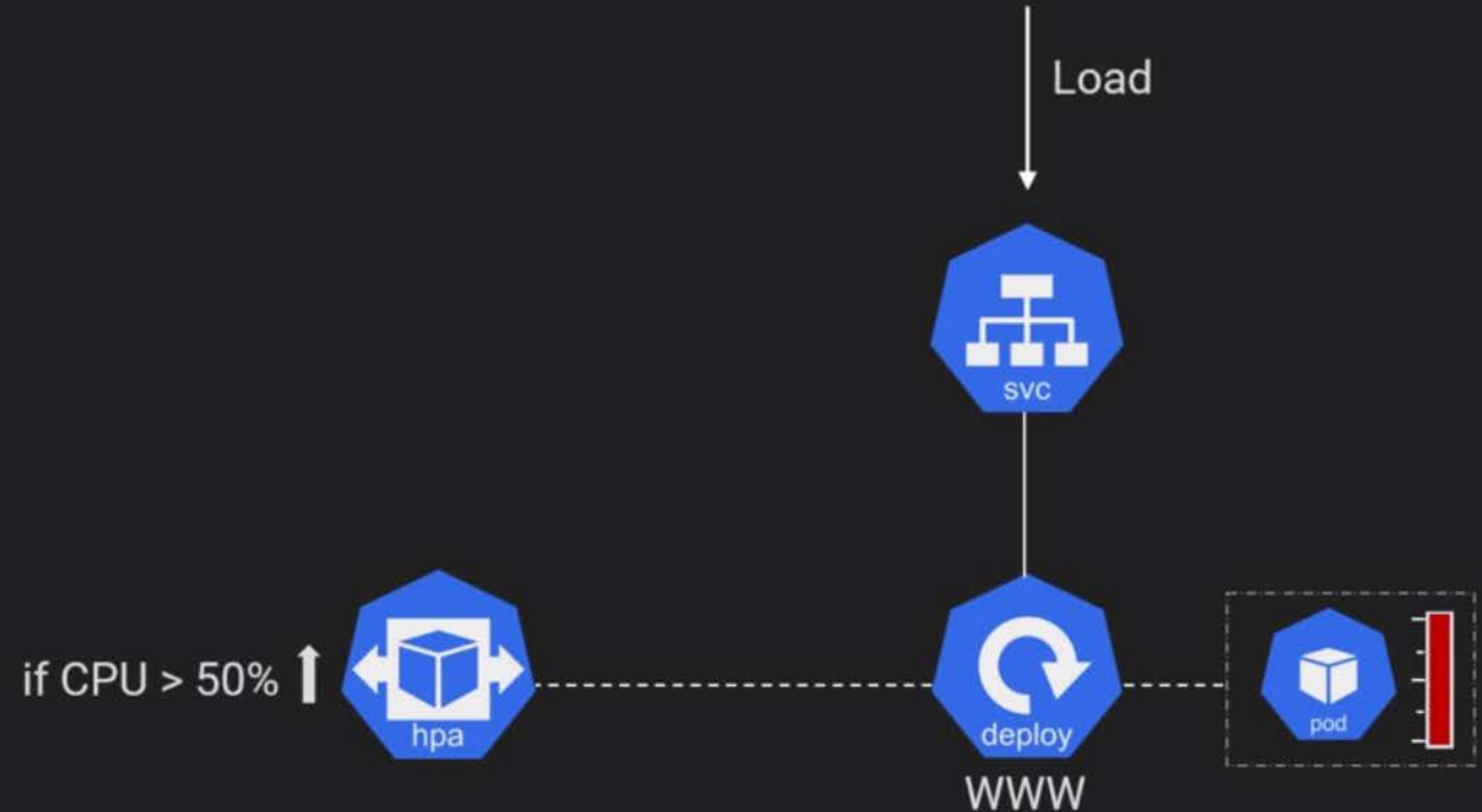


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: acg-deploy
spec:
  replicas: 3
  ...
spec:
  containers:
    - image: nginx:1.12
      name: nginx
      resources:
        limits:
          cpu: 1
        requests:
          cpu: 0.2
```



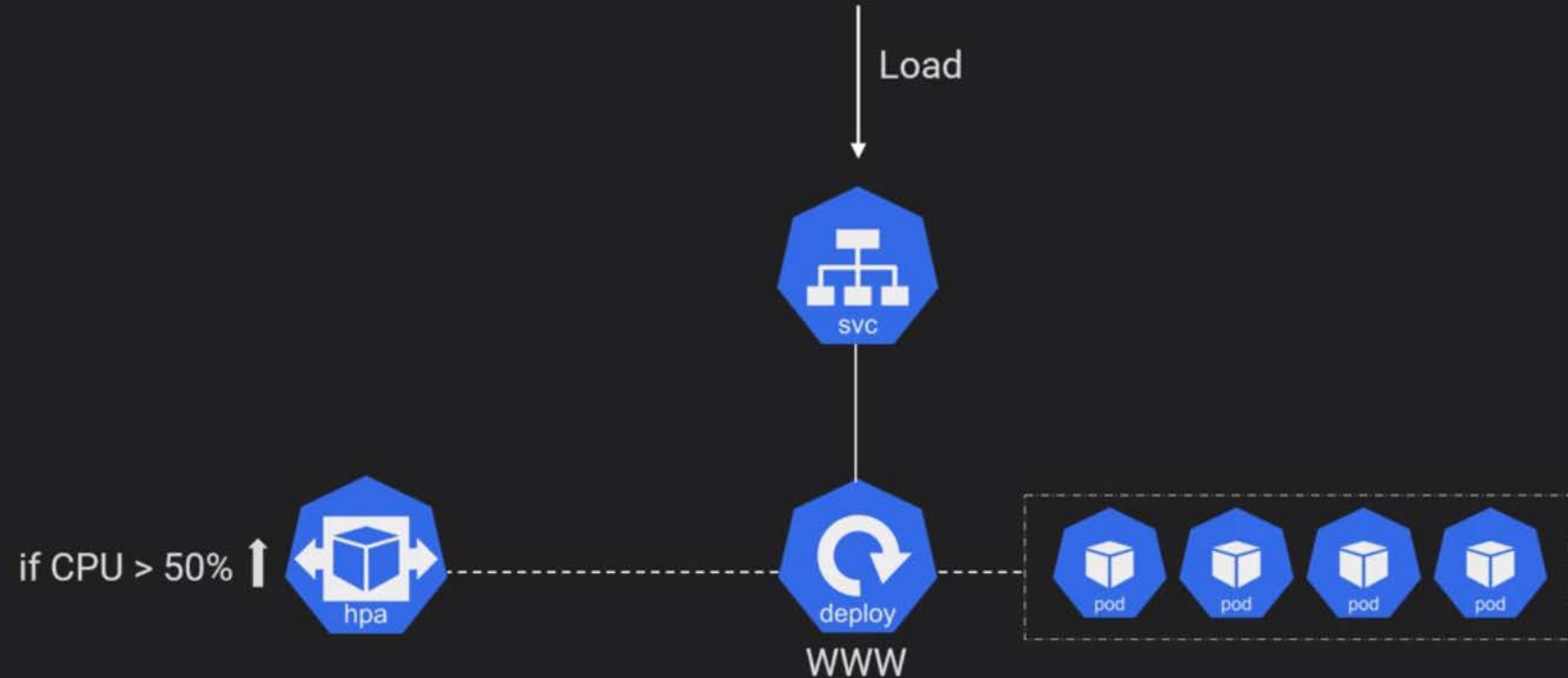
The [autoscaling/v2](#) API will bring custom metrics etc.

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.

! hpademo.yml ✘

```
1  apiVersion: v1
2  kind: Namespace
3  metadata:
4    name: acg-ns
5  ---
6  apiVersion: v1
7  kind: Service
8  metadata:
9    namespace: acg-ns
10   name: acg-lb
11   spec:
12     type: LoadBalancer
13     ports:
14       - port: 80
15     selector:
16       app: acg-stress
17   ---
18   apiVersion: apps/v1
19   kind: Deployment
```

✖ 1 ⚠ 1

Line 12 Col 21 (12 selected)

CRLF YAML



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

! hpademo.yml ✘

```
28     app: acg-stress
29     replicas: 1
30   strategy:
31     rollingUpdate:
32       maxSurge: 1
33       maxUnavailable: 0
34     type: RollingUpdate
35   template:
36     image (string)
37     metadata:
38       labels: Docker image name. More info:
39       app: https://kubernetes.io/docs/concepts/containers/images This field is optional to
40     spec: allow higher level config management to default or override container images
41     containers:
42       - image: k8s.gcr.io/hpa-example
43         name: stresser
44         ports:
45           - containerPort: 80
46         resources:
47           requests:
```

✖ 1 ⚠ 1

based on this image here.

[n 41 Col 38 /22 selected] Spaces: 2 UTF-8 CRLF YAML



! hpademo.yml ✘

```
33      maxUnavailable: 0
34      type: RollingUpdate
35      template:
36          metadata:
37              labels:
38                  app: acg-stress
39          spec:
40              containers:
41                  - image: k8s.gcr.io/hpa-example
42                      name: stresser
43                      ports:
44                          - containerPort: 80
45                      resources:
46                          requests:
47                              cpu: 0.2
48  -----
49      apiVersion: autoscaling/v1
50      kind: HorizontalPodAutoscaler
51      metadata:
```

✖ 1 ⚠ 1

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

In 48 Col 4/58 selected

Spaces: 2

UTF-8

CRLF

YAML



! hpademo.yml ✘

```
44     - containerPort: 80
45   resources:
46     requests:
47       cpu: 0.2
48   ---
49   apiVersion: autoscaling/v1
50   kind: HorizontalPodAutoscaler
51   metadata:
52     name: acg-hpa
53     namespace: acg-ns
54   spec:
55     scaleTargetRef:
56       apiVersion: apps/v1
57       kind: Deployment
58       name: acg-web
59     minReplicas: 1
60     maxReplicas: 10
61     targetCPUUtilizationPercentage: 50
```

✖ 1 ⚠ 1

Well, speak of the devil, Horizontal Pod Autoscaler,

F-8

CRLF

YAML



```
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-g175  Ready      <none>    12m      v1.10.7-gke.1
PS C:\Users\nigel>
```

Yeah, just the one node in the cluster,

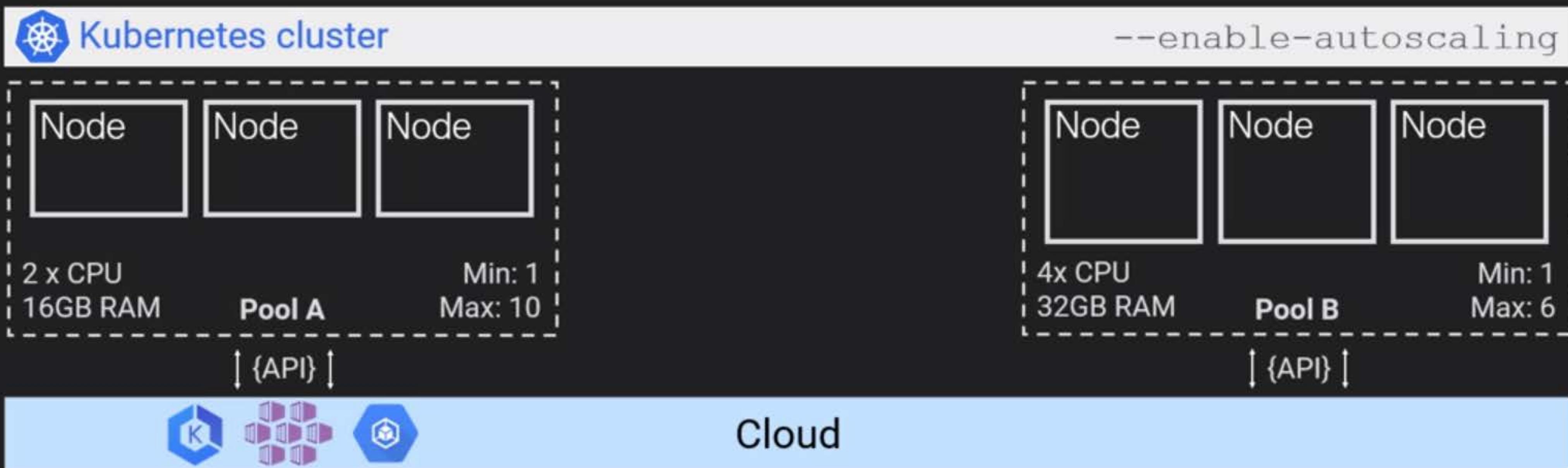
Cluster Autoscaler - Theory



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



Always use
Pod Resource Requests



Cluster Autoscaler - Theory



Cluster Autoscaler

Works on requested values

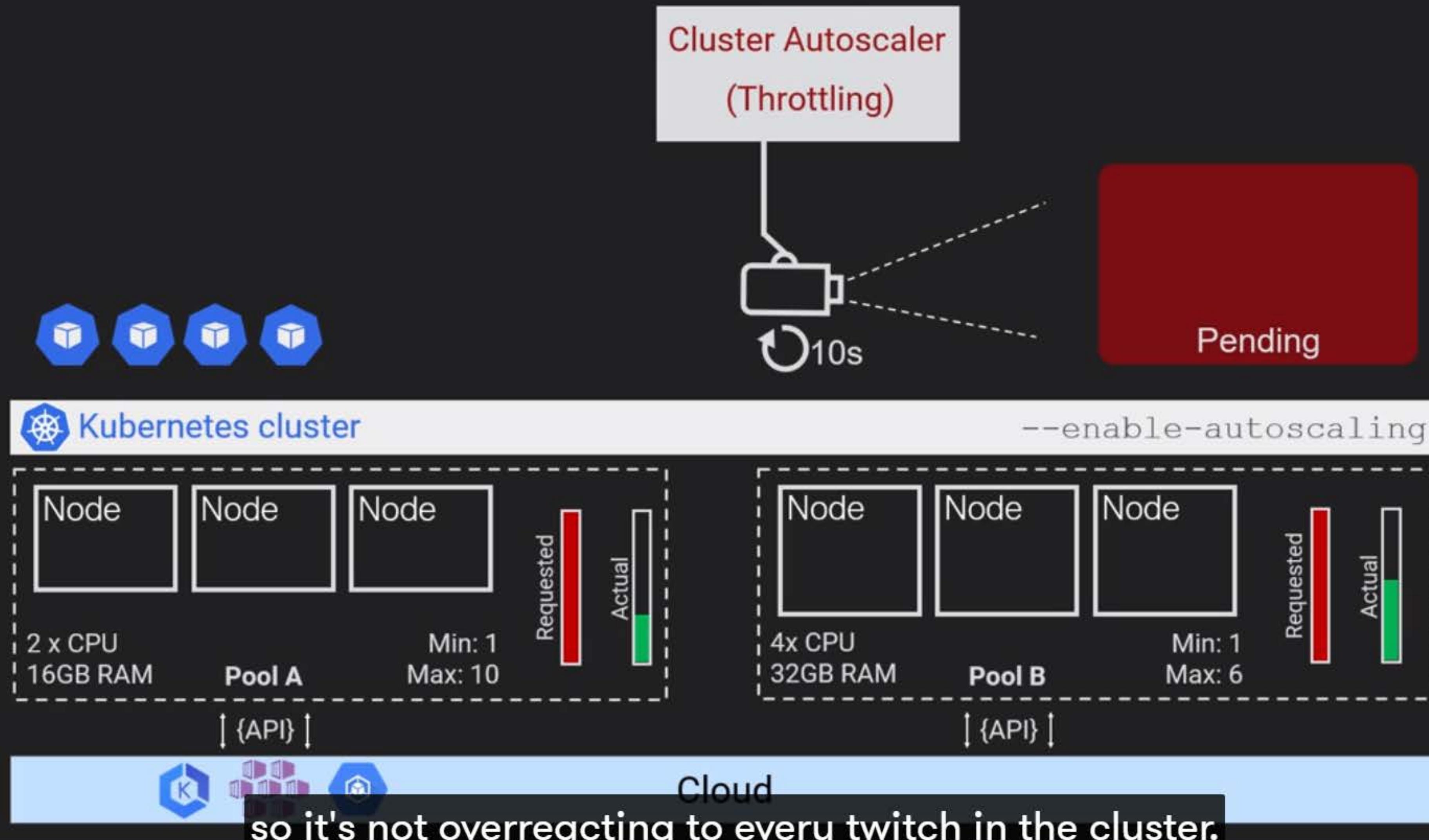
Horizontal Pod Autoscaler

Works on actual values

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



Cluster Autoscaler - Theory



Cluster Autoscaler - Theory



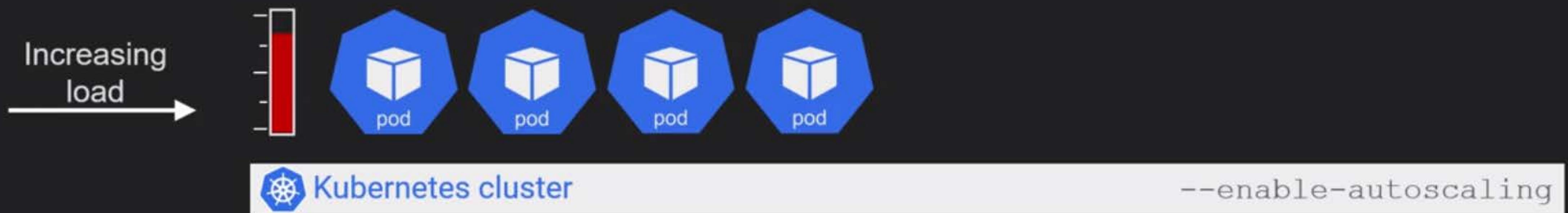
Don't mess with the node pools

Check your cloud for support

Test performance on big clusters

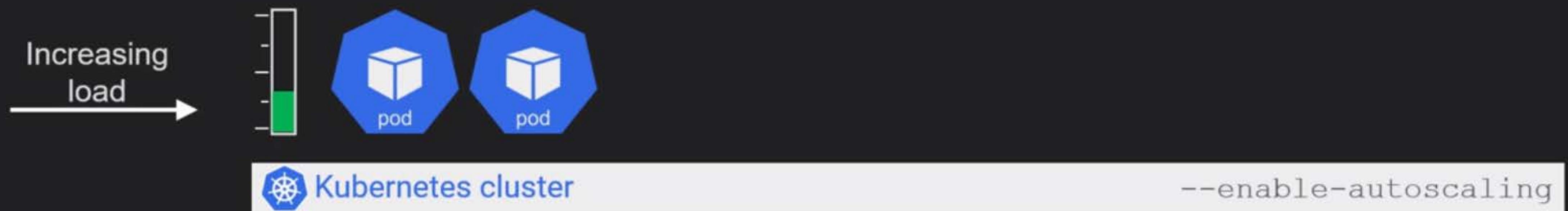
Because you might experience performance issues.

Recap



As pressure builds, we scale up,

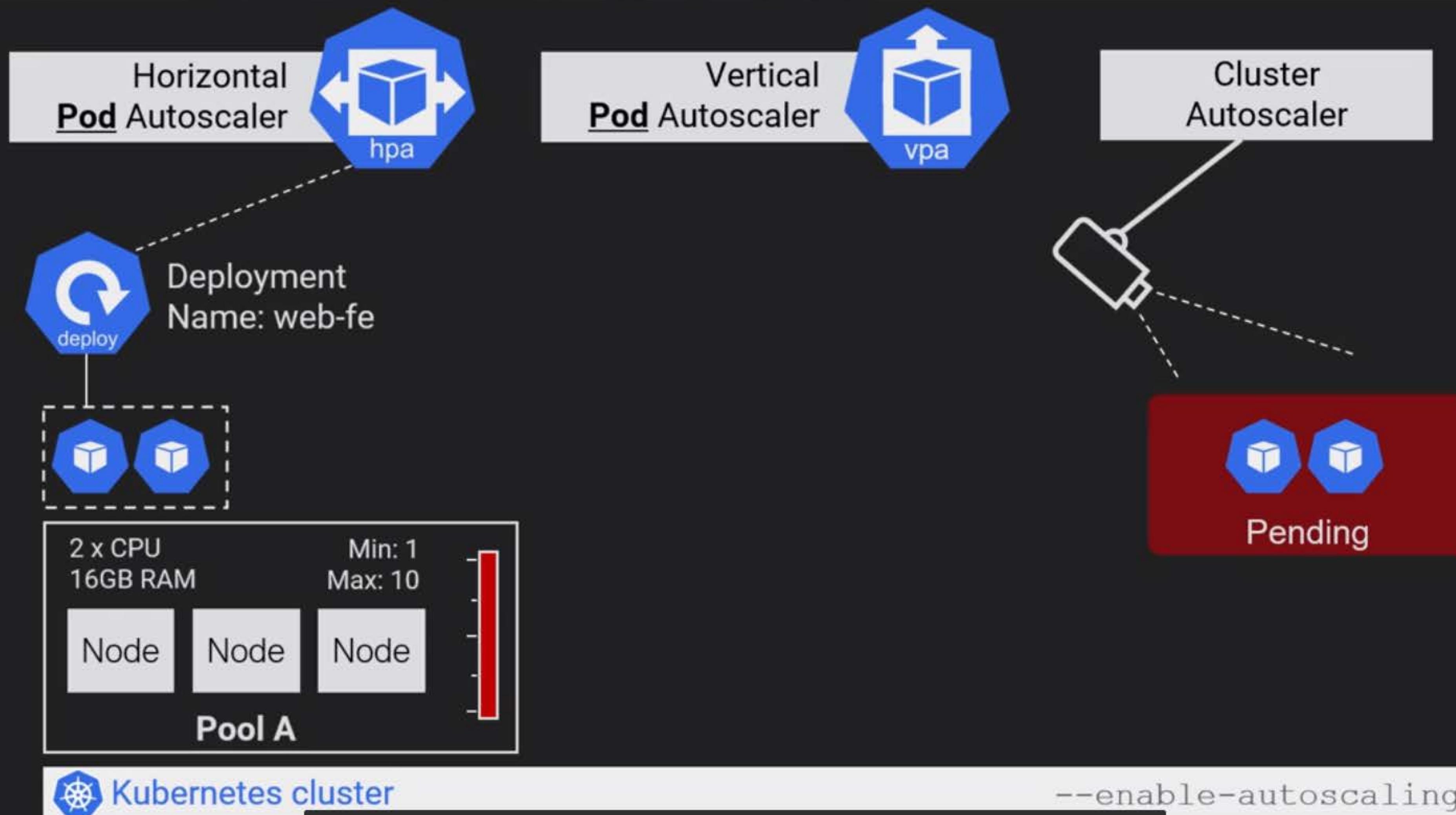
Recap



But for it to be properly useful,



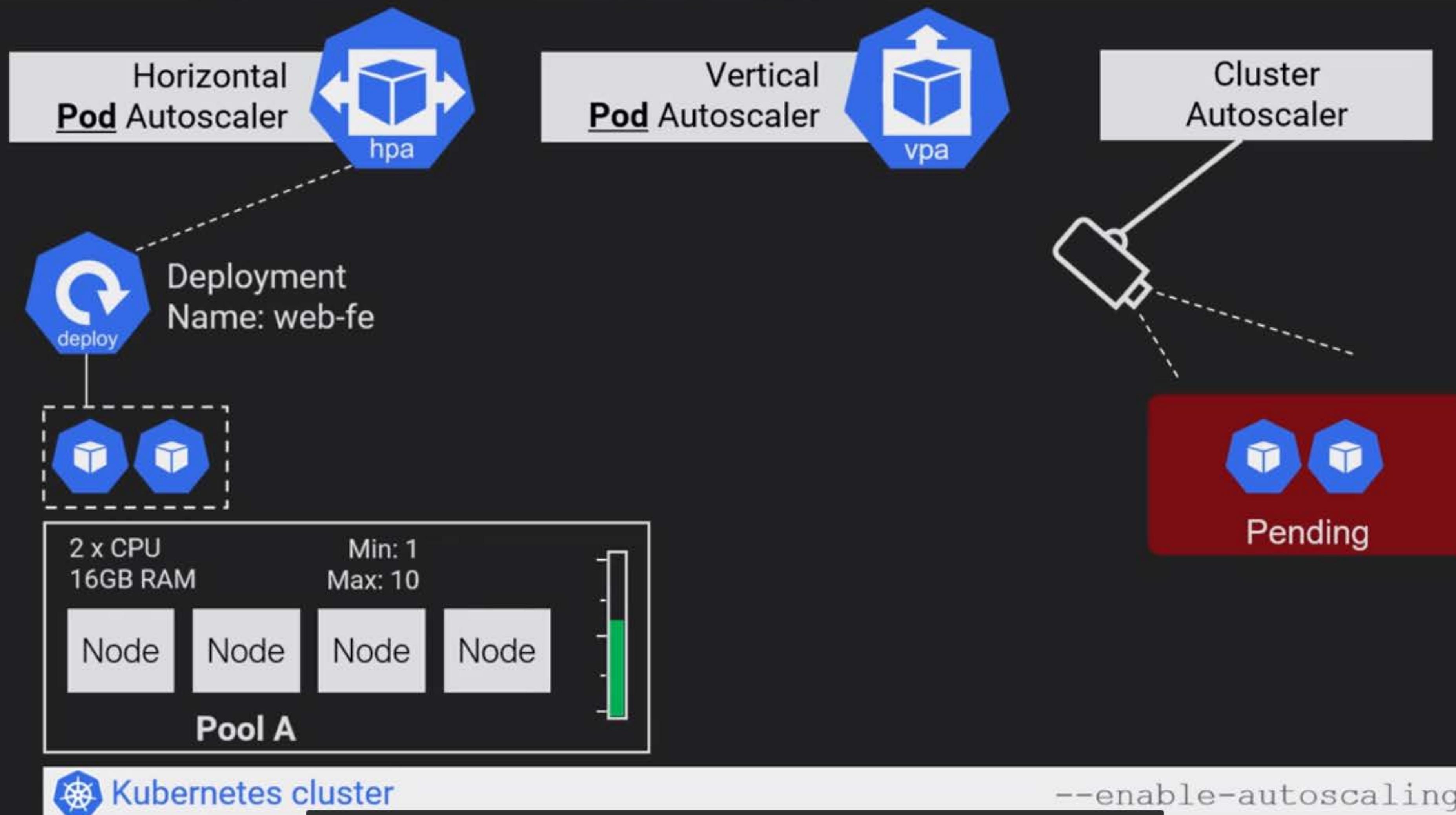
Recap



that's gonna wake up I think every 10 seconds

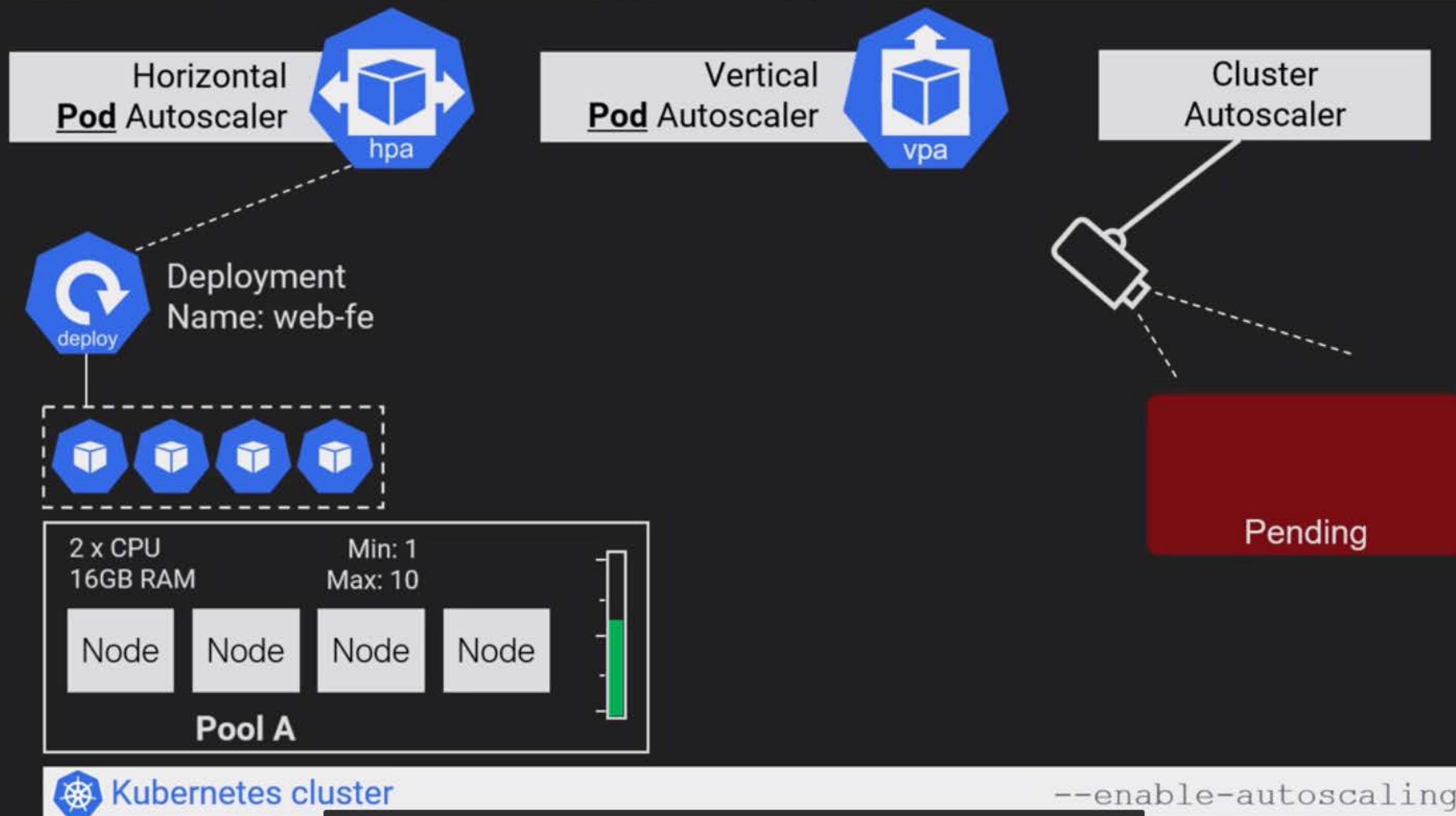


Recap



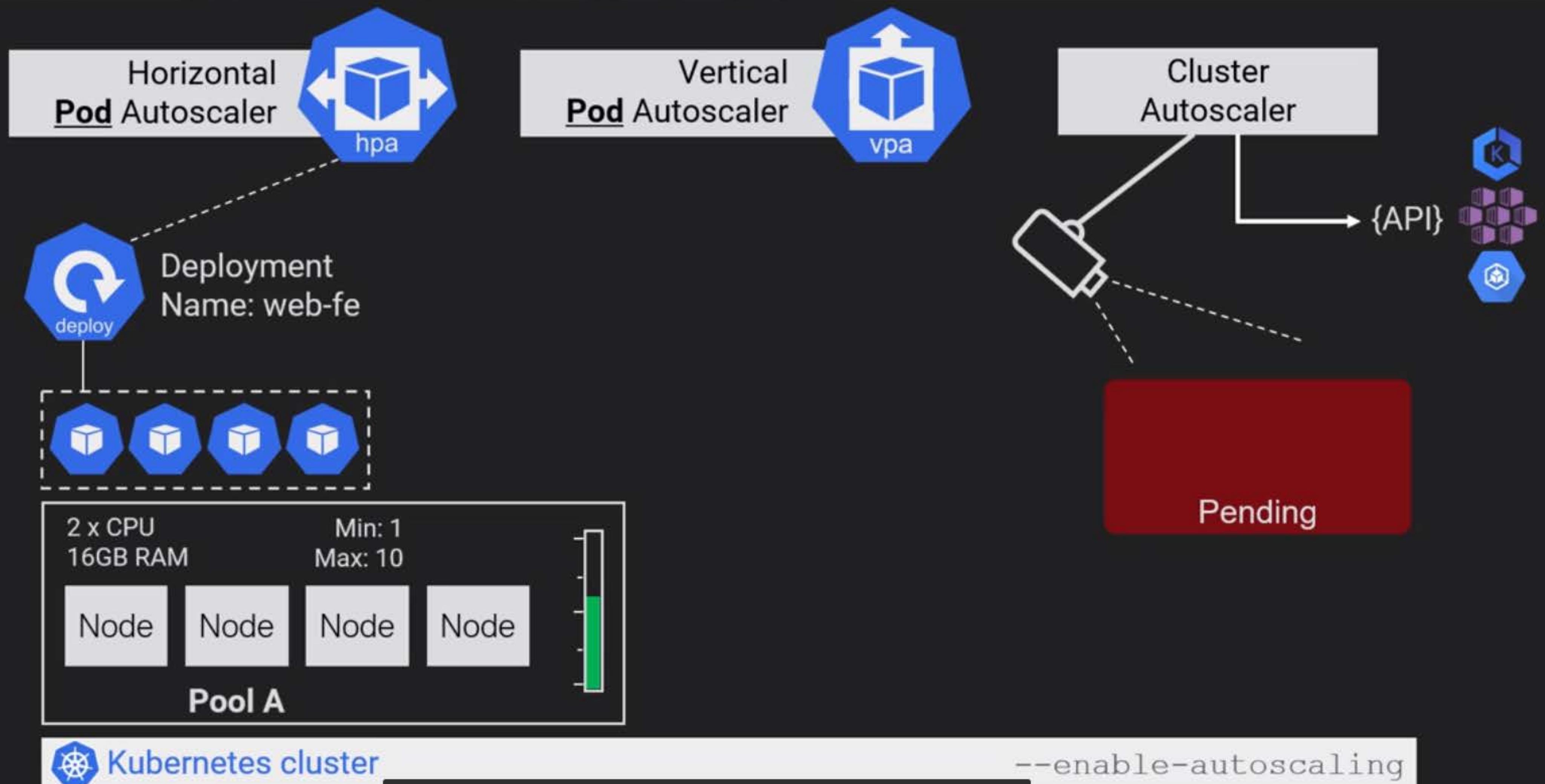
Before you know it, the pod will be scheduled

Recap



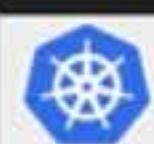
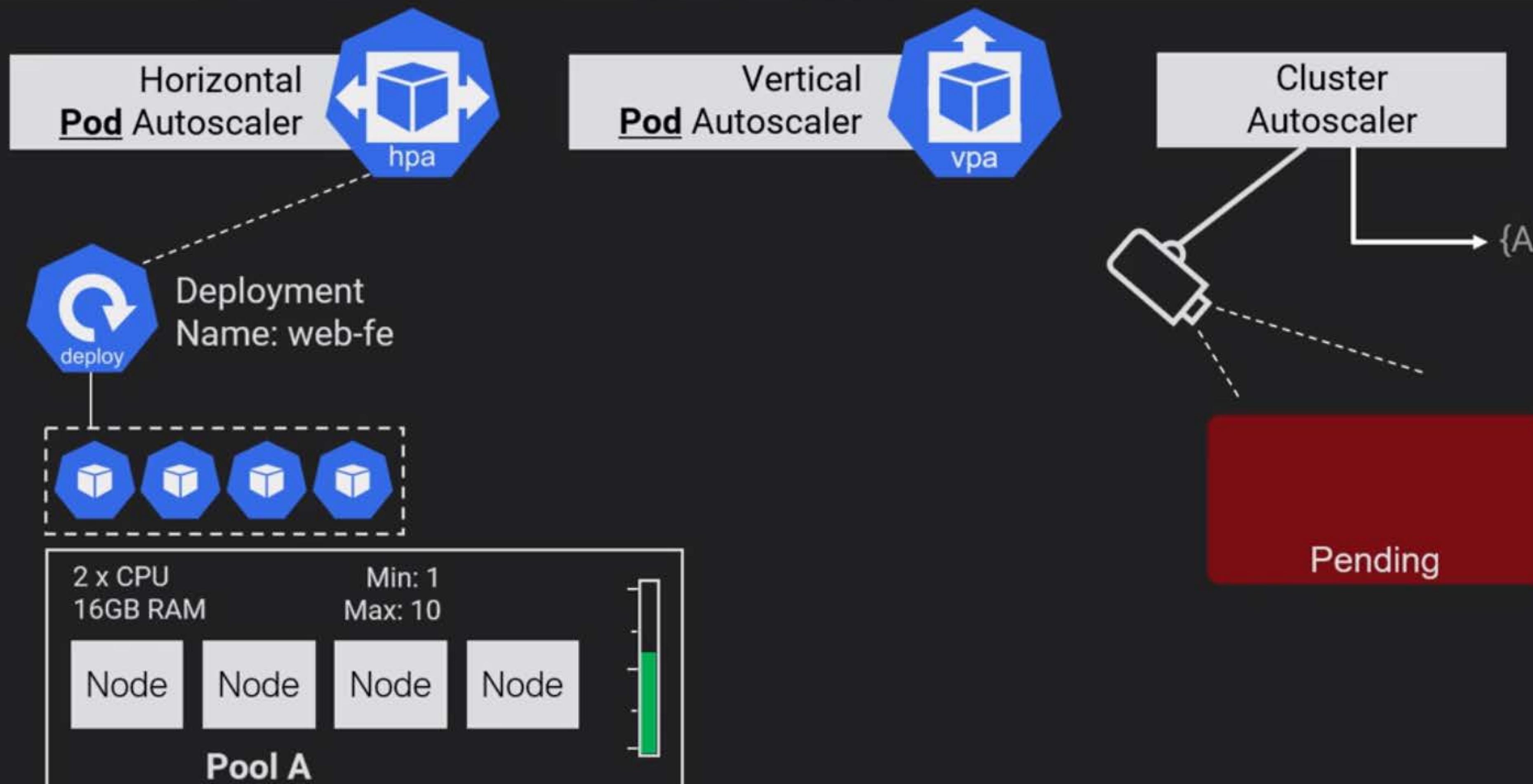
Again, it all requires pod resource requests.

Recap



we said that service load balancers

Recap



Kubernetes cluster

--enable-autoscaling

because you see, it makes external calls

Recap

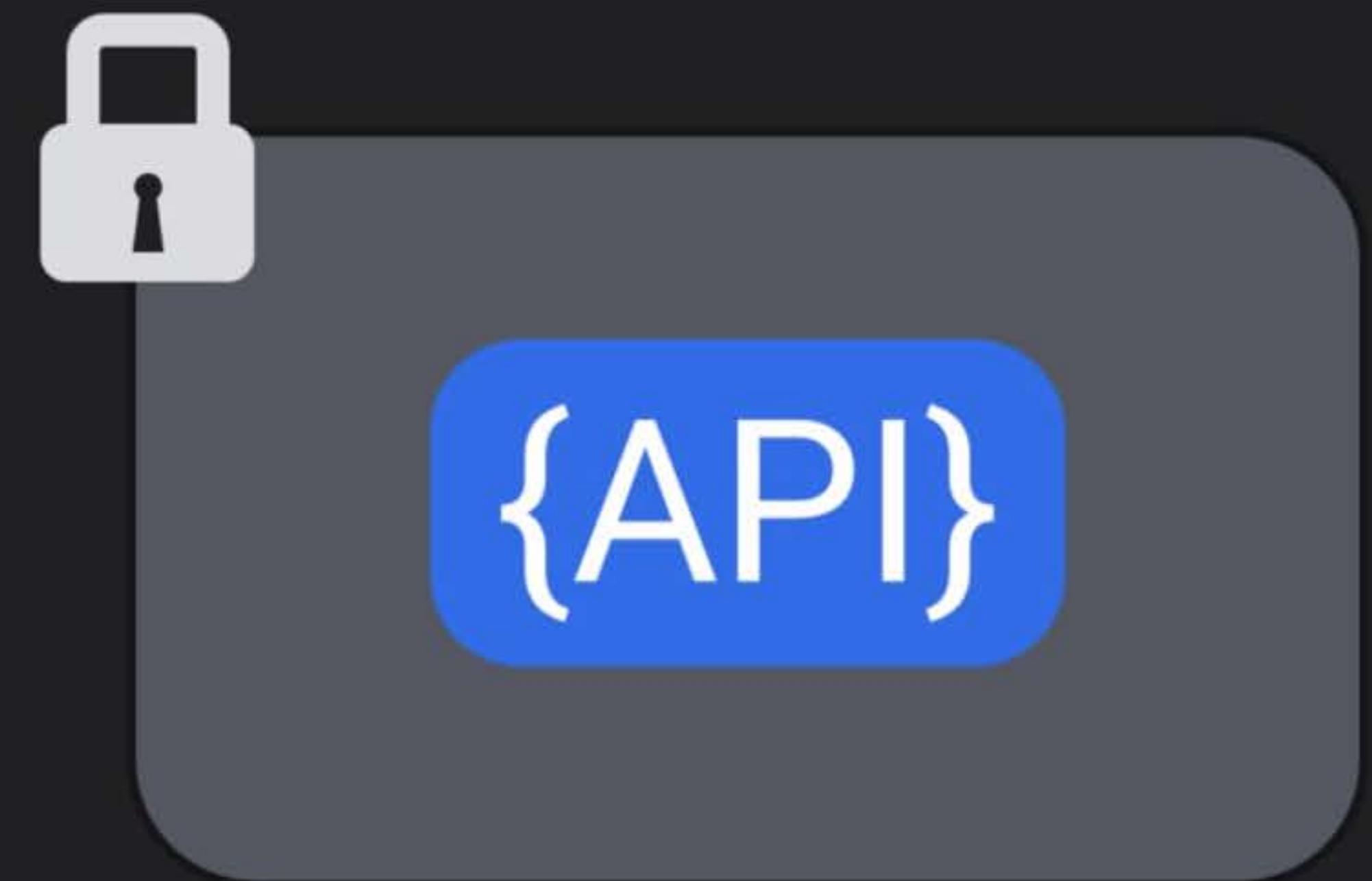


autoscaling/v1
- CPU

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

custom metrics is where the magic is gonna be.

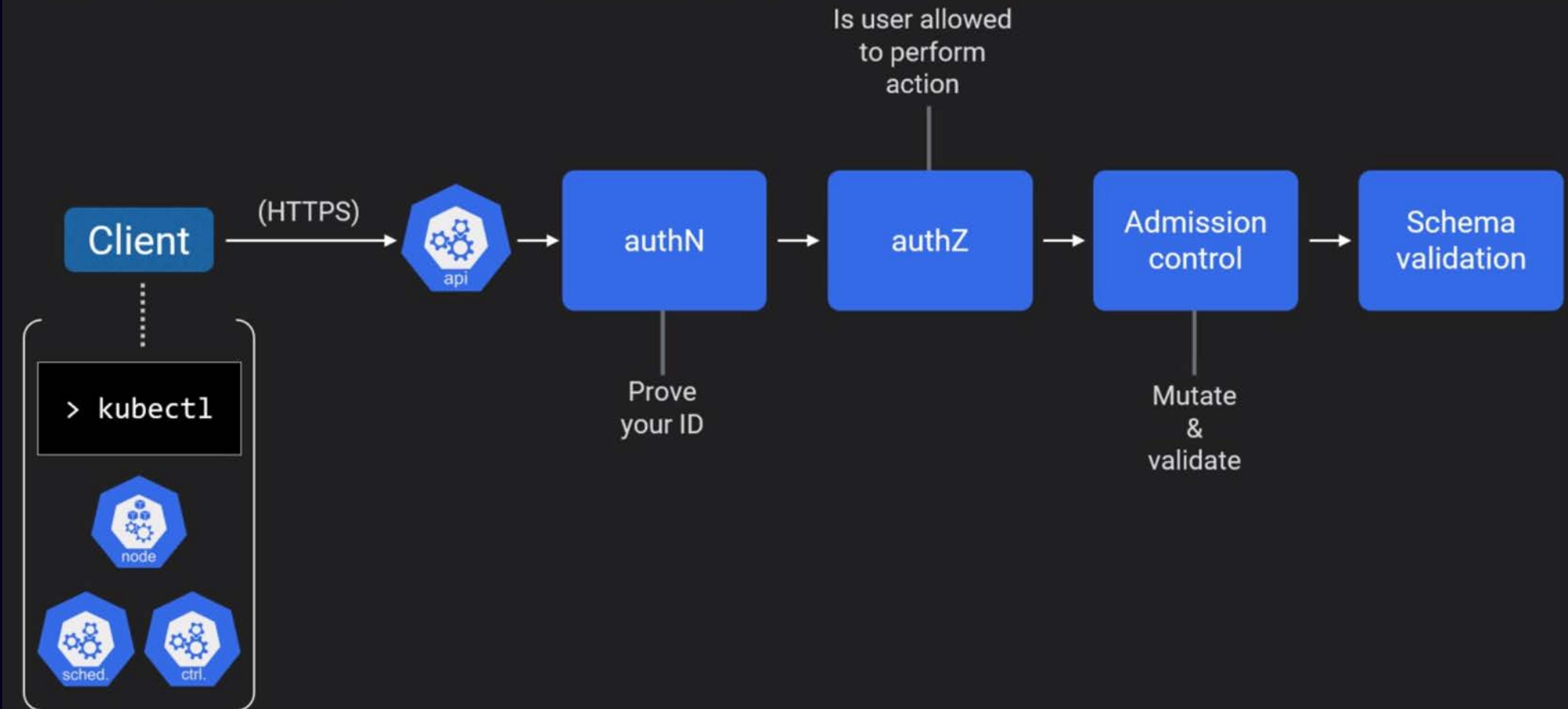
Big Picture



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

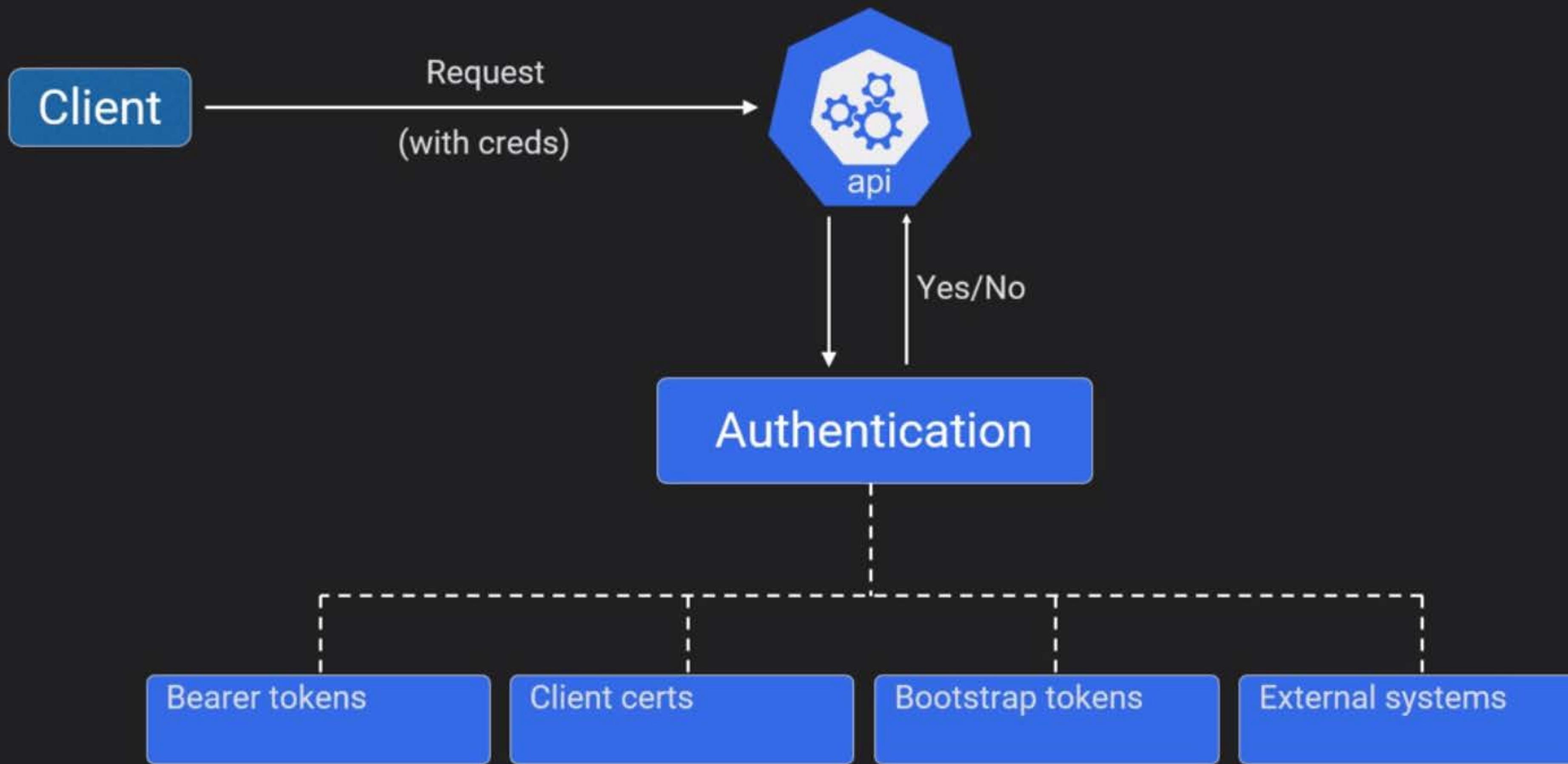
GA since 1.8

Deny-by-default

Then you have to specifically add allow rules



Authentication



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



Kubernetes does **NOT** do Users!!



Manage Users externally

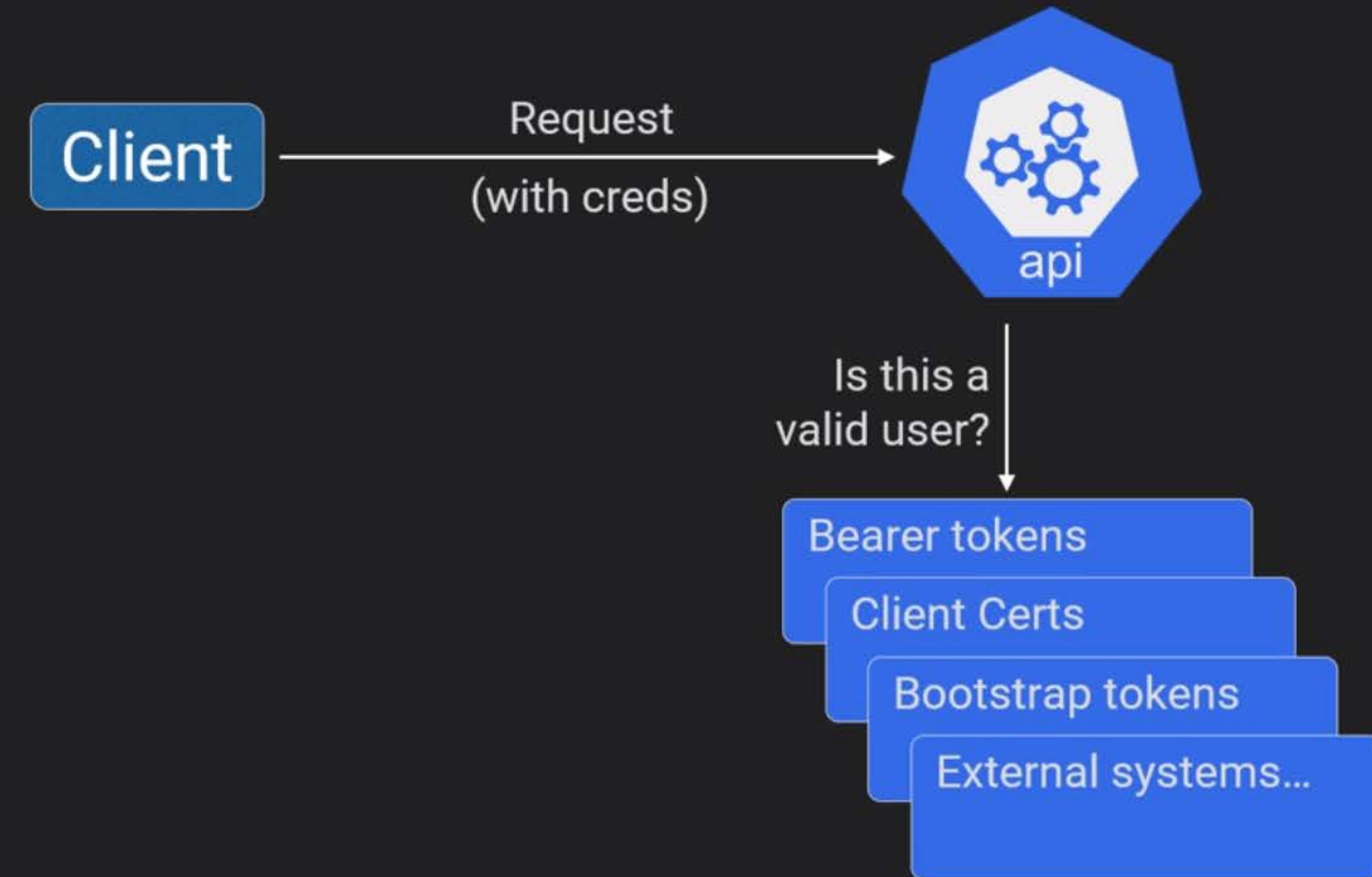
Active Directory

IAM

Other...

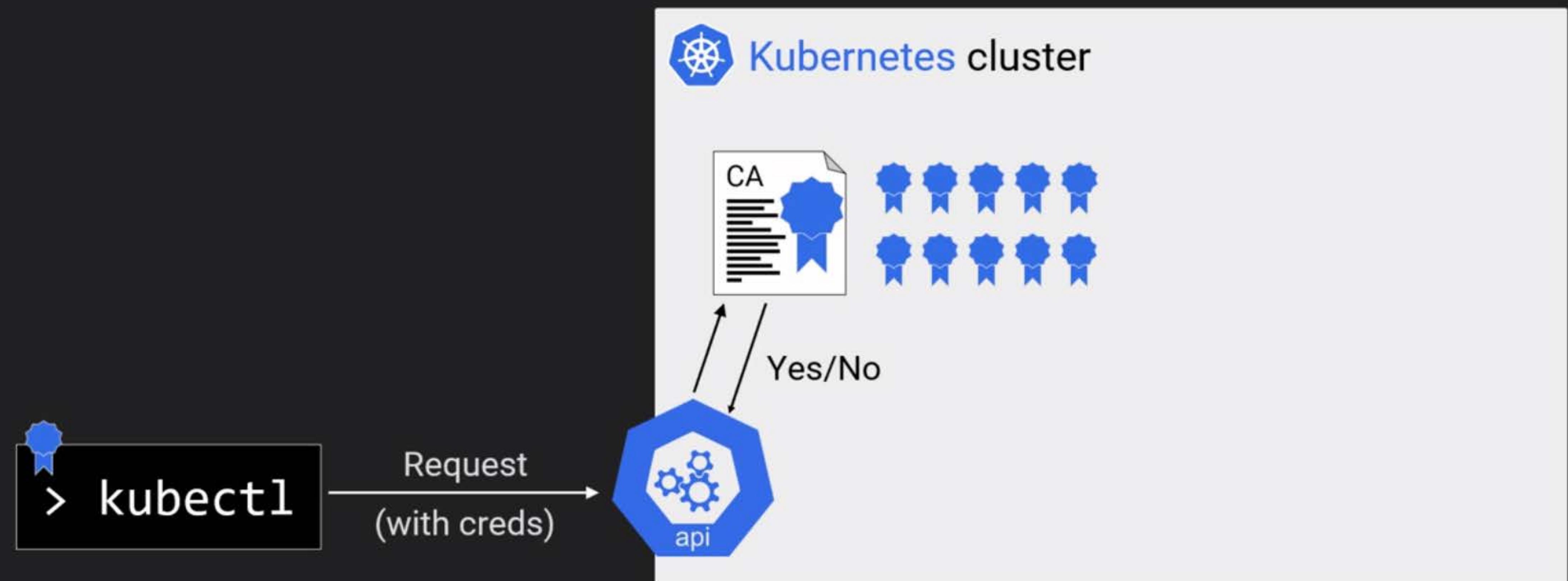
But it can totally be active directory or your clouds,

Authentication



right big hand wave going on here,

Authentication



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



Service Accounts

For System components

Managed by Kubernetes

**You can (should) manage
them**

as to how you wanna manage these.

Authentication



```
> kubectl
```

POST:

`/apis/apps/v1/namespaces/acg-ns/deployments`

Auth: Token FGg5dYFnsdY%351tg..

`{"apiVersion":"1","kind":"Deployment"....}``...`

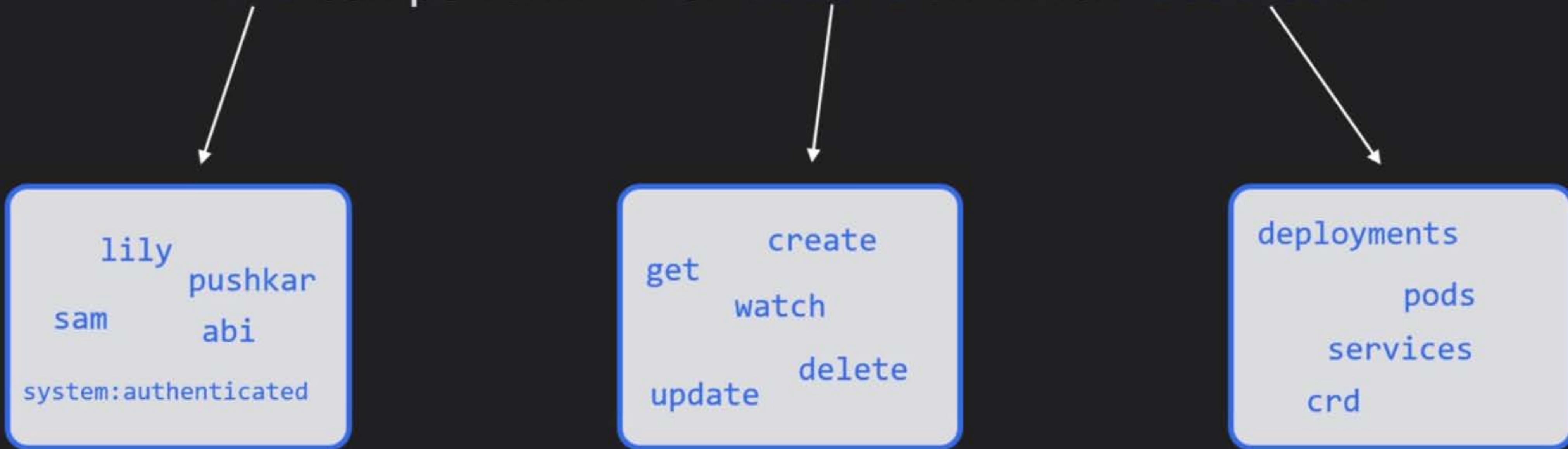
API Group:	apps/v1
Subject:	nigel
Verb:	create (HTTP POST method)
Resource:	Deployments
Namespace:	acg

Can nigel create new deployments in the ACG namespace?



Authorization

Who can perform which actions on which resources?

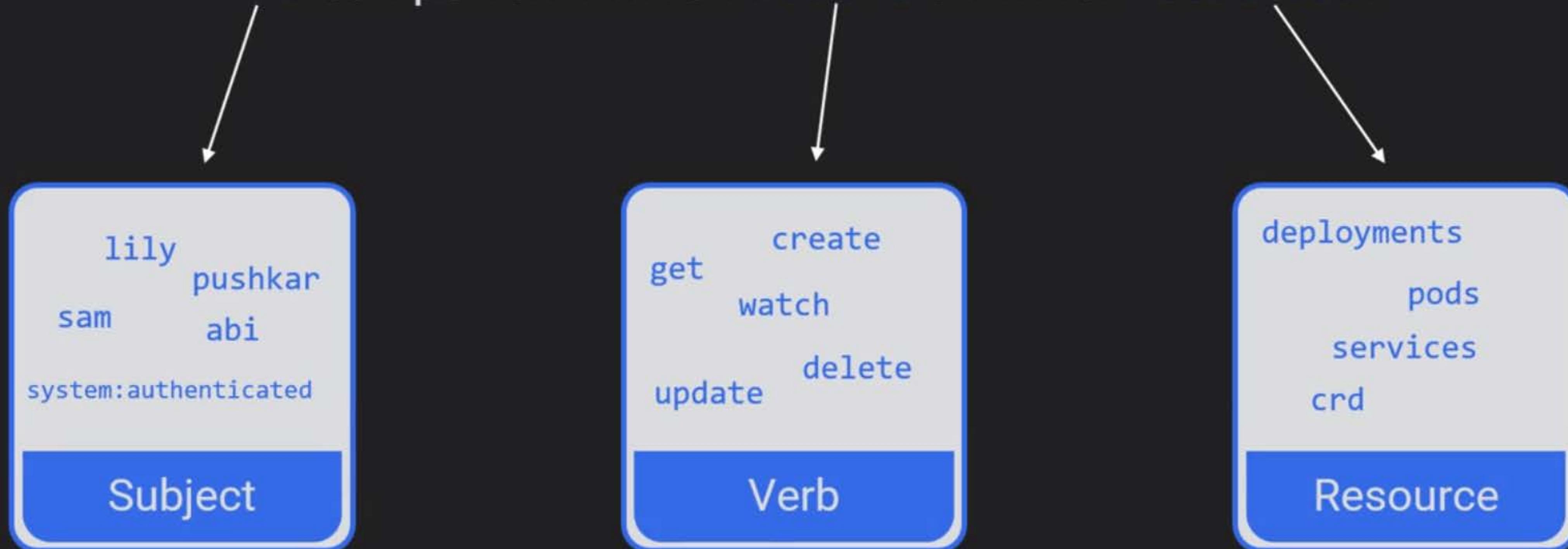


That's RBAC.



Authorization

Who can perform which actions on which resources?



So, put, get, and all that jazz.

Authorization



Roles & RoleBindings
(For *least privilege*)

if you get your heads around them, right,

Authorization



Powerful default users
(Too powerful for production)

Roles & RoleBindings
(For *least privilege*)

Roles, and RoleBindings.

Authorization



RBAC Role

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: acgrbac
  namespace: acg
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "list", "watch"]
```

Oh, and it lives under the `rbac.authorization` API group.



Authorization

RBAC RoleBinding

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: acgrbac
  namespace: acg
subjects:
- kind: User
  name: nigel@acg.com
  apiGroup: ""
roleRef:
  kind: Role
  name: acgrbac
  apiGroup: ""
```

So, again, it's a resource under the same API group.



Authorization

RBAC Role

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: acgrbac
  namespace: acg
rules:
- apiGroups: []
  resources: ["pods"]
  verbs: ["get", "list", "watch"]
```

RBAC RoleBinding

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: acgrbac
  namespace: acg
subjects:
- kind: User
  name: nigel@acg.com
  apiGroup: ""
roleRef:
  kind: Role
  name: acgrbac
  apiGroup: ""
```

That is our subject.



Authorization

RBAC Role

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: acgrbac
  namespace: acg
rules:
- apiGroups: []
  resources: ["pods"]
  verbs: ["get", "list", "watch"]
```

RBAC RoleBinding

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: acgrbac
  namespace: acg
subjects:
- kind: User
  name: nigel@acg.com
  apiGroup: ""
roleRef:
  kind: Role
  name: acgrbac
  apiGroup: ""
```



Phew. Boom.



Authorization

RBAC Role

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: acgrbac
  namespace: acg
rules:
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["get", "list", "watch"]
  - apiGroups: ["*"]
    resources: ["*"]
    verbs: ["*"]
```

RBAC RoleBinding

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: acgrbac
  namespace: acg
subjects:
  - kind: User
    name: nigel@acg.com
    apiGroup: ""
  - kind: Group
    name: ops
    apiGroup: rbac.authorization.k8s.io
  - kind: Group
    name: system:masters
    apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: acgrbac
  apiGroup: ""
```

Now, okay. This combination doesn't really make sense.



Authorization

Role | ClusterRole

```
kind: Role | ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: acgrbac
  namespace: acg
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "list", "watch"]
- apiGroups: [*"]
  resources: [*"]
  verbs: [*"]
```

RoleBinding | ClusterRoleBinding

```
kind: RoleBinding | ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: acgrbac
  namespace: acg
subjects:
- kind: User
  name: nigel@acg.com
  apiGroup: ""
- kind: Group
  name: ops
  apiGroup: rbac.authorization.k8s.io
- kind: Group
  name: system:masters
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: acgrbac
  apiGroup: ""
```

They are exactly the same as what we've already seen,



Authorization

Role | ClusterRole

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: acgrbac
  namespace: acg
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "list", "watch"]
- apiGroups: [*"]
  resources: [*"]
  verbs: [*"]
```

RoleBinding | ClusterRoleBinding

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: acgrbac
  namespace: acg
subjects:
- kind: User
  name: nigel@acg.com
  apiGroup: ""
- kind: Group
  name: ops
  apiGroup: rbac.authorization.k8s.io
- kind: Group
  name: system:masters
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: acgrbac
  apiGroup: ""
```

'cause they apply cluster-wide,



Authorization

ClusterRole

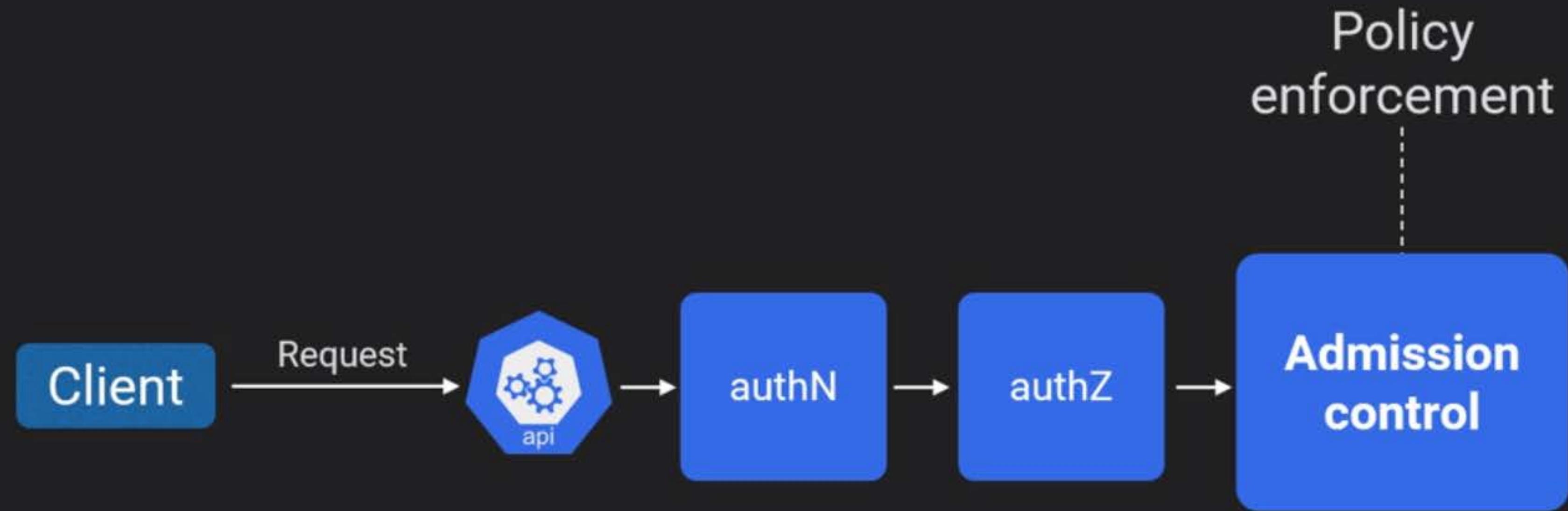
```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: acgrbac
  namespace: acg
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "list", "watch"]
- apiGroups: [*"]
  resources: [*"]
  verbs: [*"]
```

RoleBinding

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: acgrbac
  namespace: acg
subjects:
- kind: User
  name: nigel@acg.com
  apiGroup: ""
- kind: Group
  name: ops
  apiGroup: rbac.authorization.k8s.io
- kind: Group
  name: system:masters
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: acgrbac
  apiGroup: ""
```

but then, attach them to users, via namespaced RoleBindings.

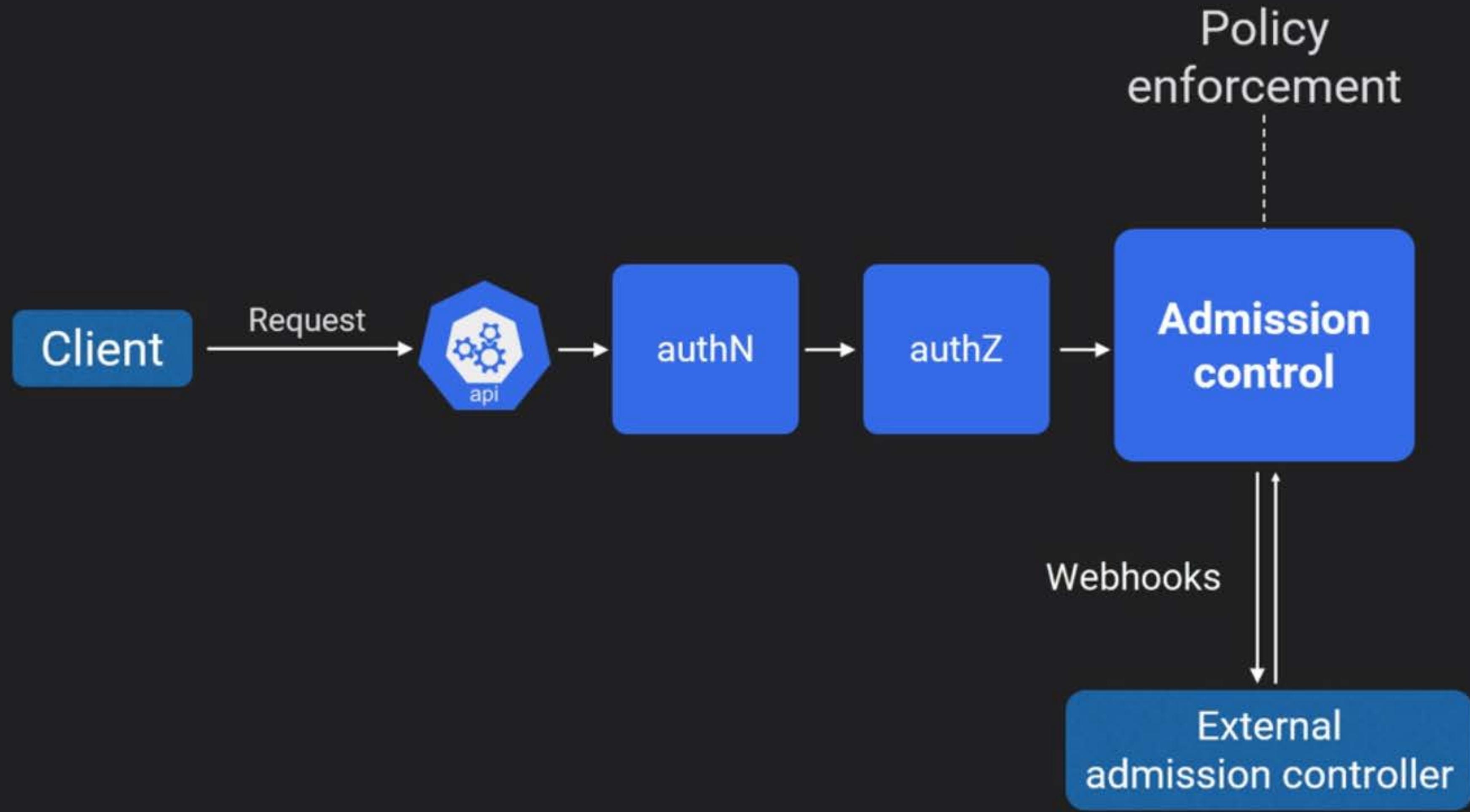
Admission Control



or a name space to get certain labels, yeah.



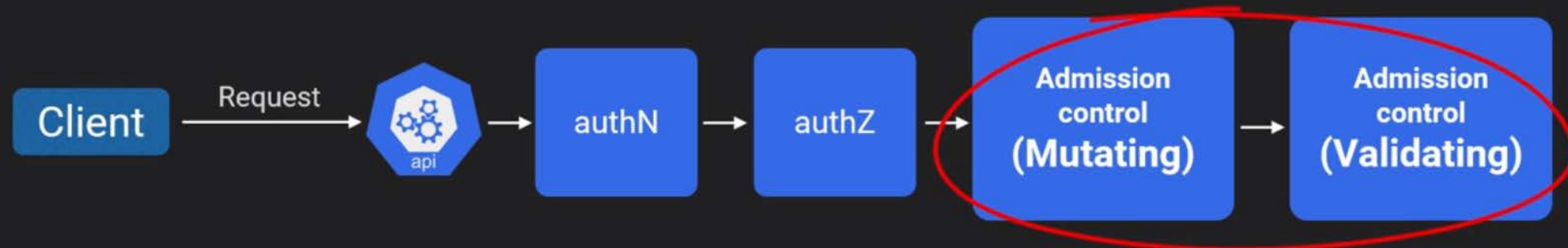
Admission Control



and compiled directly into API server.



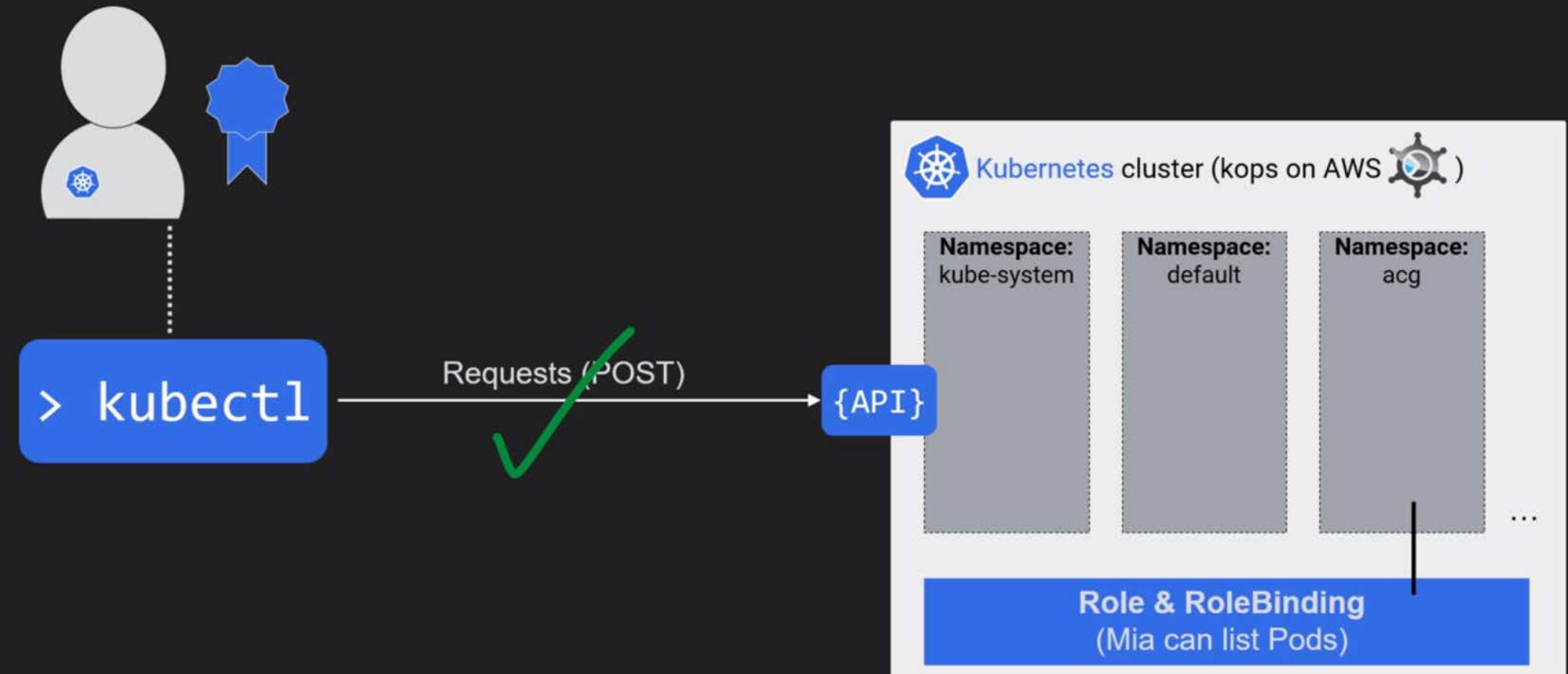
Admission Control



They kick in here so its slightly different stages in the



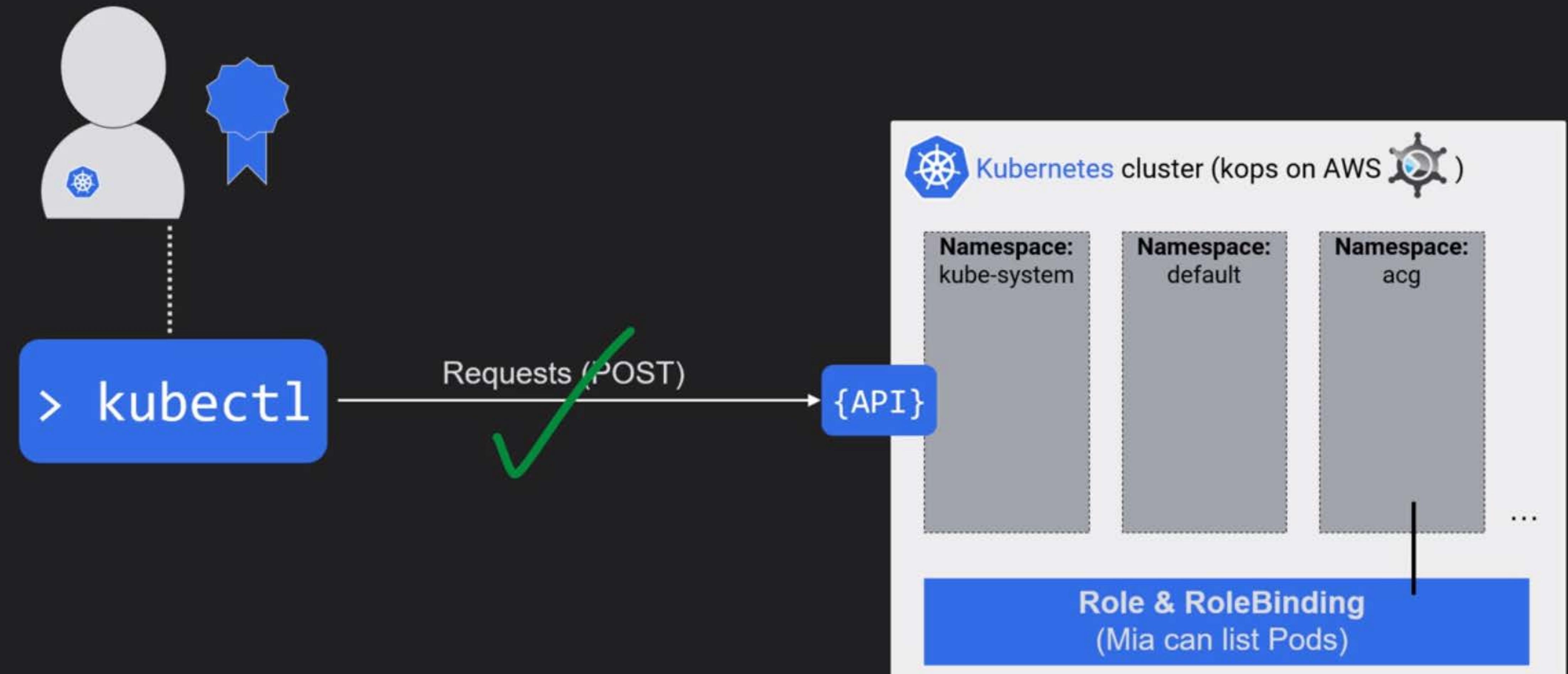
Demo



Well, hopefully.



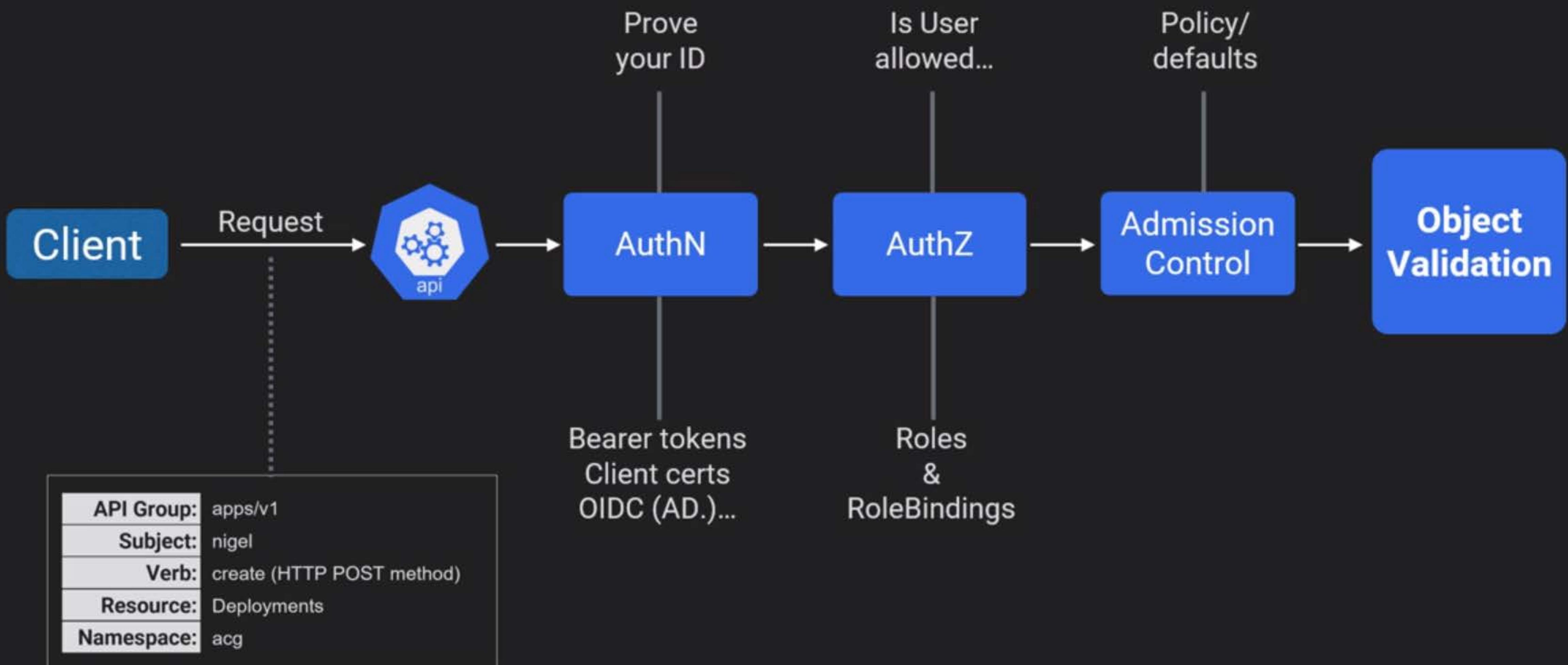
Demo



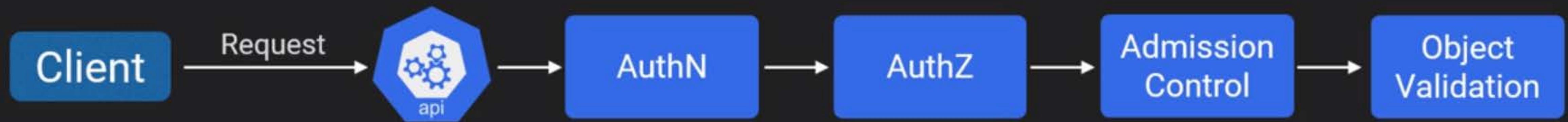
Well, hopefully.



Recap



Recap



RBAC is stable

PLAN
your
implementation

Additive model



Other Kubernetes Stuff...



DaemonSet



StatefulSet



job



CronJob



PodSecurityPolicy

Pod resource
requests and limits

ResourceQuota



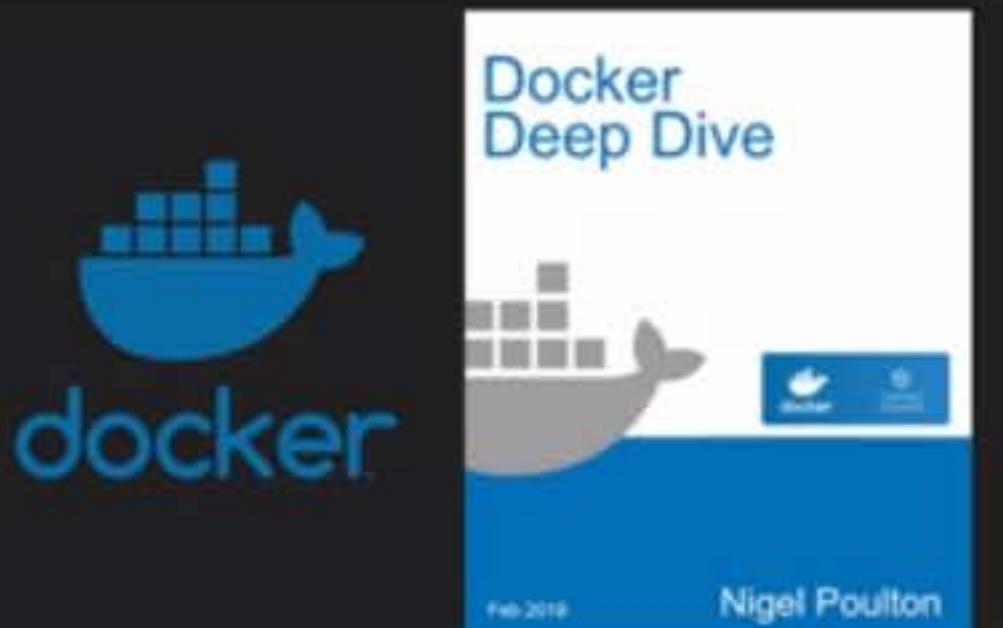
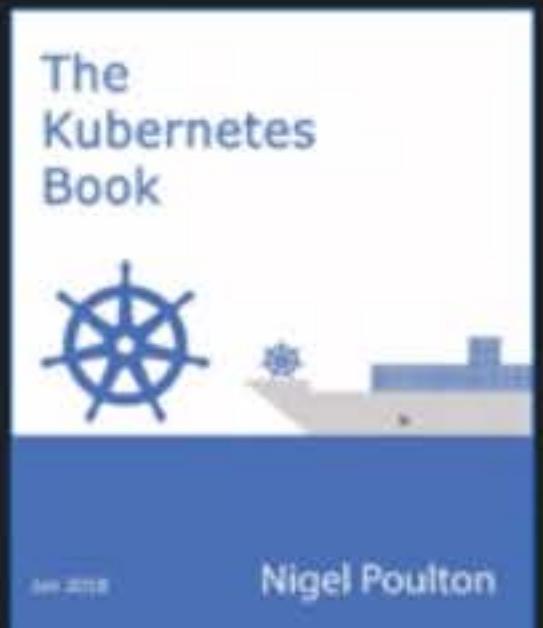
CustomResourceDefinition

I'm gonna say right, one of the best things

What Next...



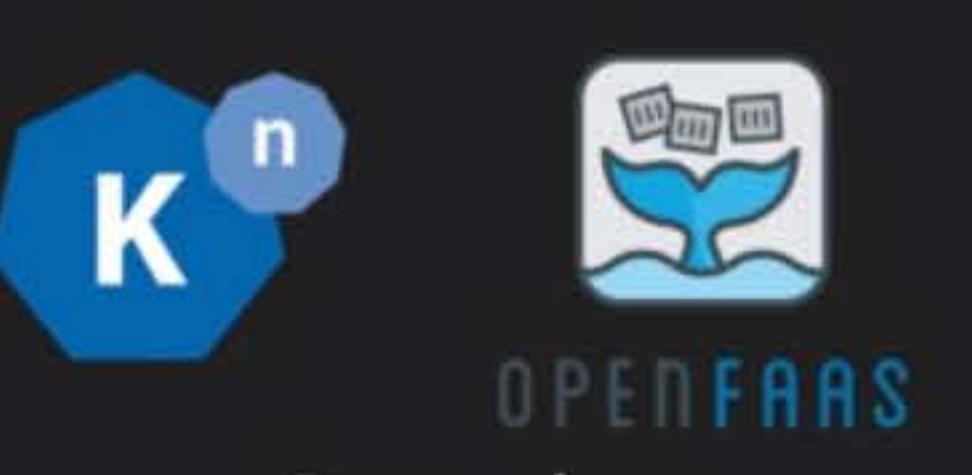
A CLOUD GURU



Community



Kubernetes
Podcast
from Google



OPENFAAS

Serverless



Service Meshes



Prometheus

{ API }



> kubectl apply...



Course Quiz - X Kubernetes D X Google Kube X Deploying Go X Deploying Wk X Create a CI/C X Final Prep - A X Best practices X Best practices X Understanding X +

acloud.guru/course/kubernetes-deep-dive/learn/next/quiz/watch?backUrl=%2Fcourses

RFR DSA AlgoExp WebOps CPP reference udemy RFR PPT GClass Courses Resume Builder Site Rating Image/Pdf => Word Grades IITM SAE CS course Stanford DSA »

QUESTION 10

Which of the following is a major benefit of the CSI?

- Vendor storage code no longer needs to exist in the main Kubernetes code tree
- Resources from 3rd-party storage systems can finally be accessed by Kubernetes resources
- More durable storage
- Faster storage

CHAPTER 11.3

Kubernetes Deep Dive - What's Next? Quiz

This course pairs well with our sandbox environment. Get your hands cloudy.

[Learn More](#)

[Open Cloud Playground](#)

[ALL LESSONS](#) [CURRENT LESSON](#)

RESOURCES

This lesson has no resources.

DISCUSSIONS

There are no discussions for this lesson yet.

[Next question](#) [Rate this question](#)

GOT A QUESTION?

89% 10:39 ENG 04-12-2020

Course Quiz - X Kubernetes D X Google Kuber X Deploying Go X Deploying W Create a CI/C Final Prep - A X Best practices X Best practices X Understandin X +

acloud.guru/course/kubernetes-deep-dive/learn/next/quiz/watch?backUrl=%2Fcourses

RFR DSA AlgoExp WebOps CPP reference udemy RFR PPT GClass Courses Resume Builder Site Rating Image/Pdf => Word Grades IITM SAE CS course Stanford DSA

QUESTION 12

You are troubleshooting a security issue where a user appears to have permissions to objects in a namespace that they shouldn't. You have looked at the ClusterRole and ClusterRoleBinding objects, and the rules and subjects look to be correct. You have checked namespace objects on the cluster, and they look fine. Which of the following is most likely the cause?

Your cluster is running Kubernetes 1.10 which does not enforce RBAC rules

You need to configure a RoleBinding instead of a ClusterRoleBinding

You are looking on the wrong cluster

The namespace controller is down

Good work!

ClusterRoles and ClusterRoleBindings apply across the entire cluster and are not subject to restrictions imposed by namespaces. A good practice is to configure cluster-wide rules via ClusterRoles, but enforce them within namespaces using RoleBindings.

Next question

GOT A QUESTION? Rate this question

88% 10:40 ENG 04-12-2020

Course Quiz - X Kubernetes D X Google Kube X Deploying Go X Deploying W X Create a CI/C X Final Prep - A X Best practices X Best practices X Understanding X +

acloud.guru/course/kubernetes-deep-dive/learn/next/quiz/watch?backUrl=%2Fcourses

RFR DSA AlgoExp WebOps CPP reference udemy RFR PPT GClass Courses Resume Builder Site Rating Image/Pdf => Word Grades IITM SAE CS course Stanford DSA

QUESTION 16

How is the Pod network usually implemented on a Kubernetes cluster?

As an isolated Linux bridge on each worker node.

Natively by the core Kubernetes code as a VXLAN overlay.

As an overlay network via a 3rd-party driver/plug-in.

Using a combination of MACVLAN and IPVLAN in the Linux kernel.

CHAPTER 11.3

Kubernetes Deep Dive - What's Next? Quiz

This course pairs well with our sandbox environment. Get your hands cloudy.

Learn More

Open Cloud Playground

ALL LESSONS CURRENT LESSON

RESOURCES

This lesson has no resources.

DISCUSSIONS

There are no discussions for this lesson yet.

GOT A QUESTION?

Next question Rate this question

87% 10:42 04-12-2020



Kubernetes Deep Dive - What's Next? Quiz

This course pairs well with our sandbox environment. Get your hands cloudy.

[Learn More](#)

Open Cloud Playground

[ALL LESSONS](#) [CURRENT LESSON](#)

RESOURCES

This lesson has no resources.

DISCUSSIONS

There are no discussions for
this lesson yet.

GOT A QUESTION?

QUESTION 18

You are deploying an application that exposes an API. You need other applications on your internal corporate LAN to be able to query the API. These applications will not be running on your Kubernetes cluster. Which type of Kubernetes Service would you deploy to make this work?

- ClusterIP
 - NodePort
 - LoadBalancer
 - Ingress

Good work!

NodePort Services are reachable from outside of the Kubernetes cluster by combining the IP address of any node in the cluster with a defined port. This allows clients that are not part of your Kubernetes cluster to be able to reach them so long as they know the name of any node in the cluster and the port that the Service is published on.

Next question

 Rate this question

Course Quiz - X Kubernetes D X Google Kuber X Deploying Go X Deploying W Create a CI/C Final Prep - A X Best practices X Best practices X Understandin X + - □ ×

acloud.guru/course/kubernetes-deep-dive/learn/next/quiz/watch?backUrl=%2Fcourses

RFR DSA AlgoExp WebOps CPP reference udemy RFR PPT GClass Courses Resume Builder Site Rating Image/Pdf => Word Grades IITM SAE CS course Stanford DSA »

QUESTION 25

Which of the following best describes the Kubernetes API?

- A monolithic versioned API with discrete endpoints for each Kubernetes object.
- A GRPC API
- A modular RESTful API that accepts CRUD-style operations.
- A monolithic RESTful API that accepts CRUD-style operations.

CHAPTER 11.3

Kubernetes Deep Dive - What's Next? Quiz

This course pairs well with our sandbox environment. Get your hands cloudy.

Learn More

Open Cloud Playground

ALL LESSONS CURRENT LESSON

RESOURCES

This lesson has no resources.

DISCUSSIONS

There are no discussions for this lesson yet.

GOT A QUESTION?

Next question

Rate this question

84% 10:45 04-12-2020

Course Quiz - X Kubernetes D X Google Kuber X Deploying Go X Deploying W Create a CI/C Final Prep - A X Best practices X Best practices X Understandin X +

acloud.guru/course/kubernetes-deep-dive/learn/next/quiz/watch?backUrl=%2Fcourses

RFR DSA AlgoExp WebOps CPP reference udemy RFR PPT GClass Courses Resume Builder Site Rating Image/Pdf => Word Grades IITM SAE CS course Stanford DSA

QUESTION 34

Which of the following describes a cloud-native app?

An application that leverages services that are native to public cloud platforms such as AWS, Azure and GCP.

An application that publishes an HTTPS web front-end.

An application made of loosely-coupled microservices that work together.

An application where all logic is coded into a single large binary.

Sorry!

Correct Answer

Cloud native applications tend to be microservices based, where individual components of the app are coded as individual services that work together to form a useful application. Quite often the individual services communicate over well-defined APIs.

Open Cloud Playground

ALL LESSONS CURRENT LESSON

RESOURCES

This lesson has no resources.

DISCUSSIONS

There are no discussions for this lesson yet.

GOT A QUESTION?

Next question Rate this question

82% 10:48 04-12-2020