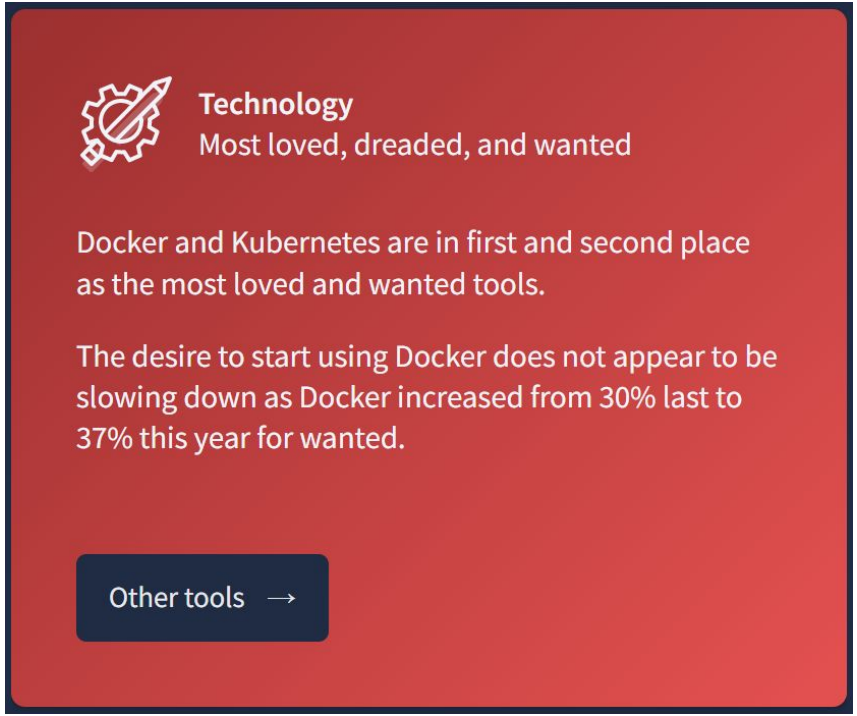
Let´s build Kubernetes

Level: Beginner

Kubernetes Container

Oliver Seitz
https://www.linkedin.com/in/oliver-seitz/



https://survey.stackoverflow.co/2022/#technology

(My) Goals

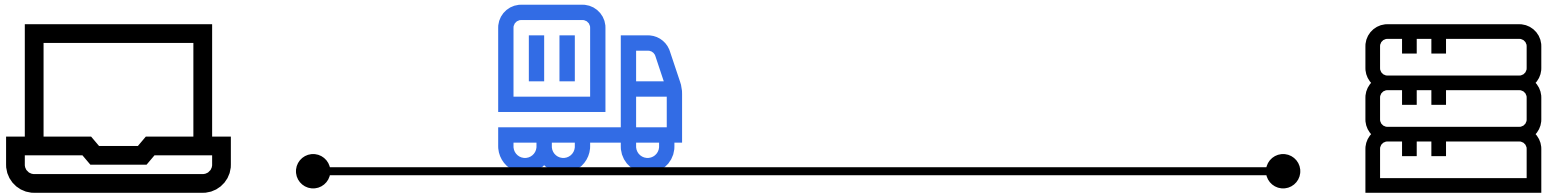
Run software

Unify configuration

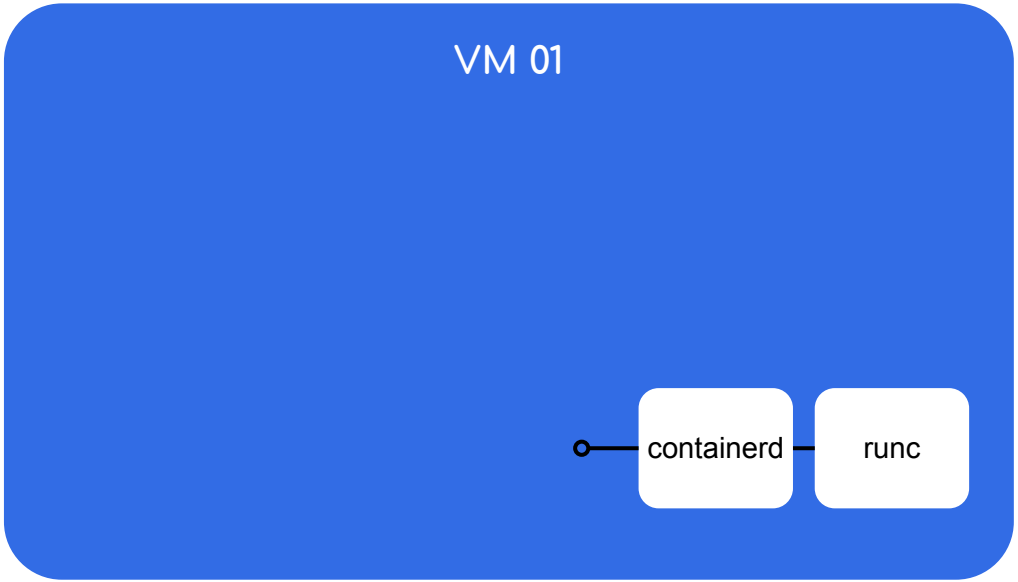
Decouple infrastructure

Packaging

Packaging → Container

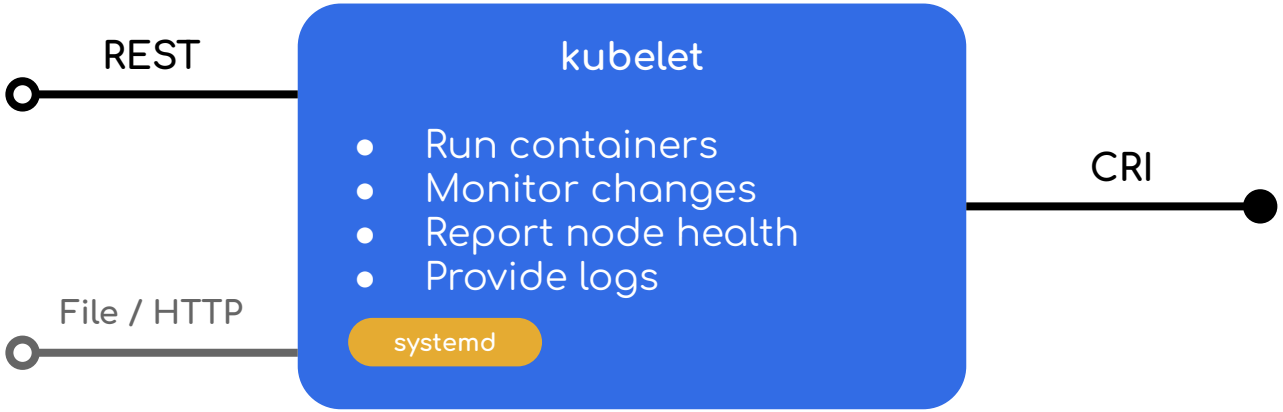


Let’s get started



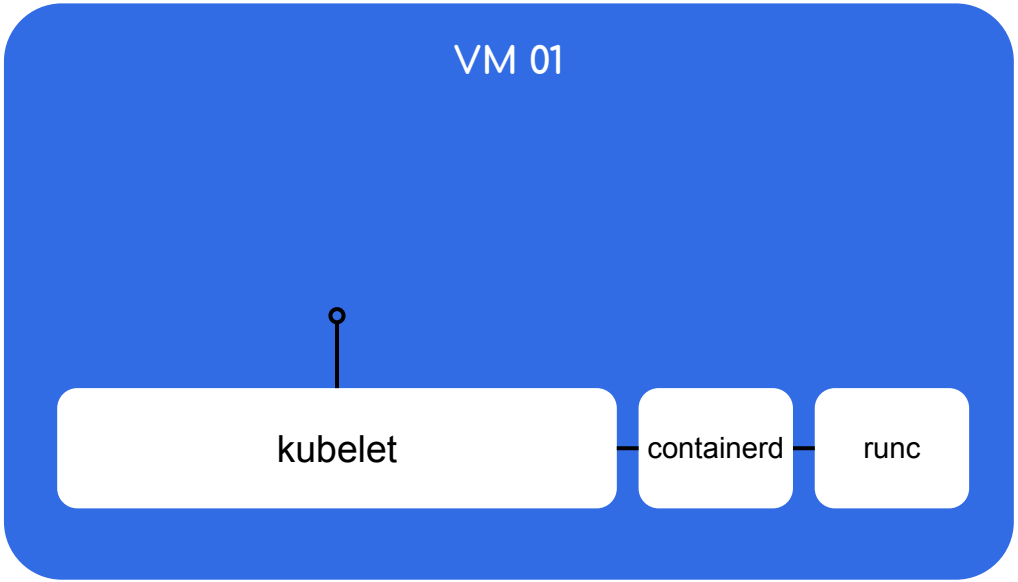
Who talks to containerd

Adding kubelet



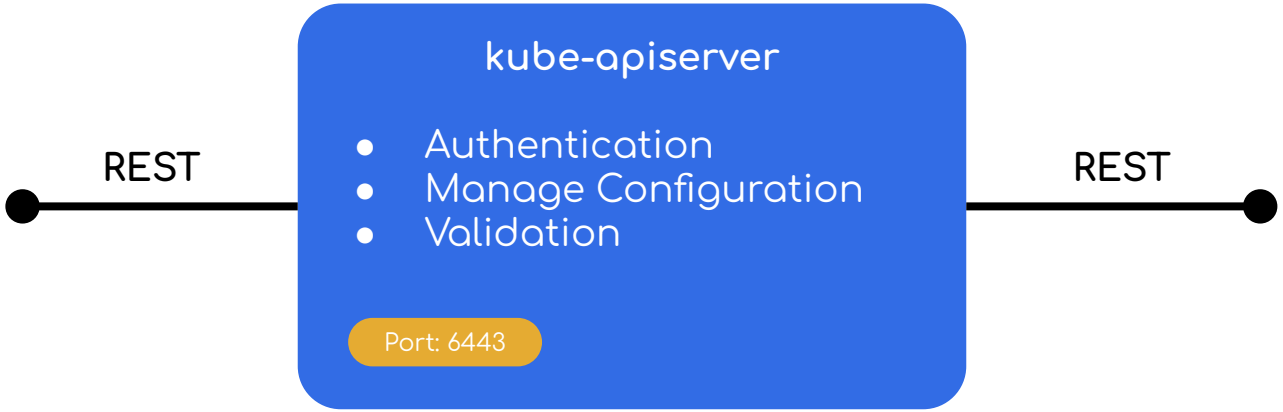
CRI = Container Runtime Interface

Adding kubelet

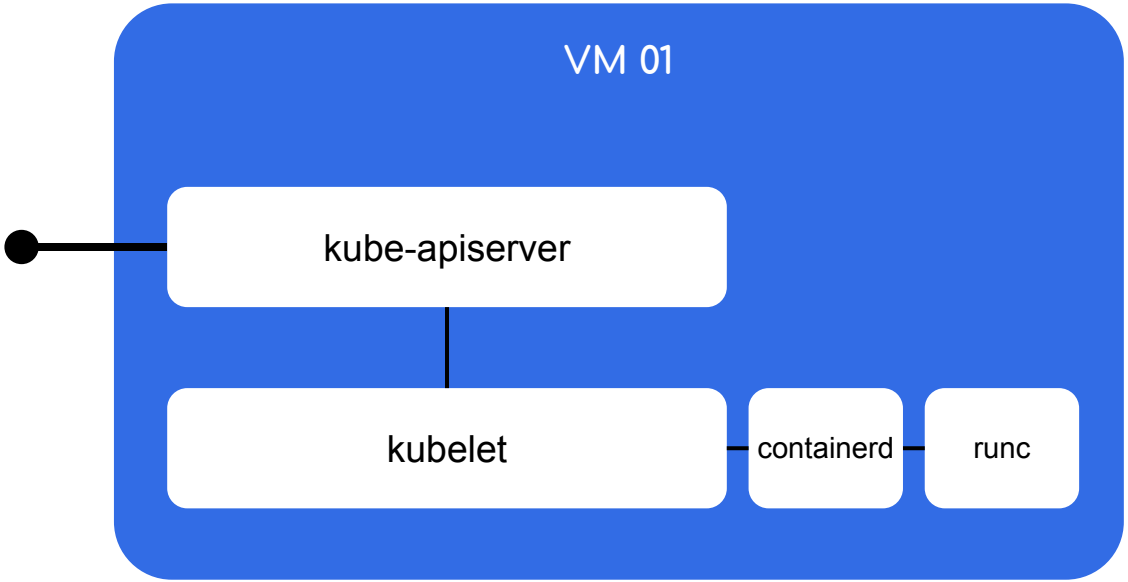


How to configure kubelet

Adding kube-apiserver



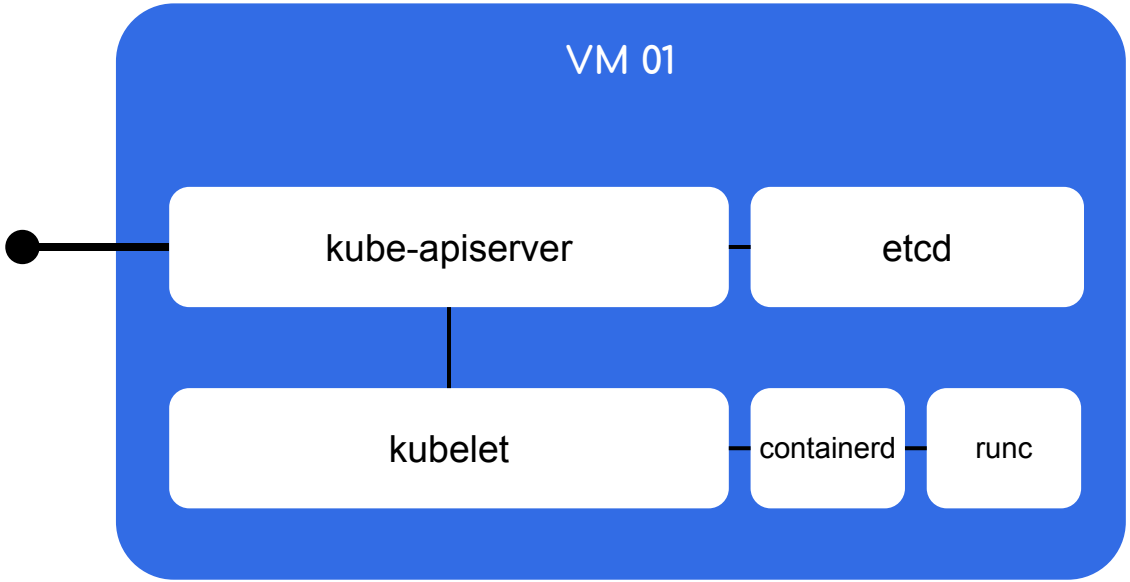
Adding kube-apiserver



- Remote management
- Push Information
- Pull Information
- Watch for changes
- Start/Stop containers
- Monitoring

What about Persistence

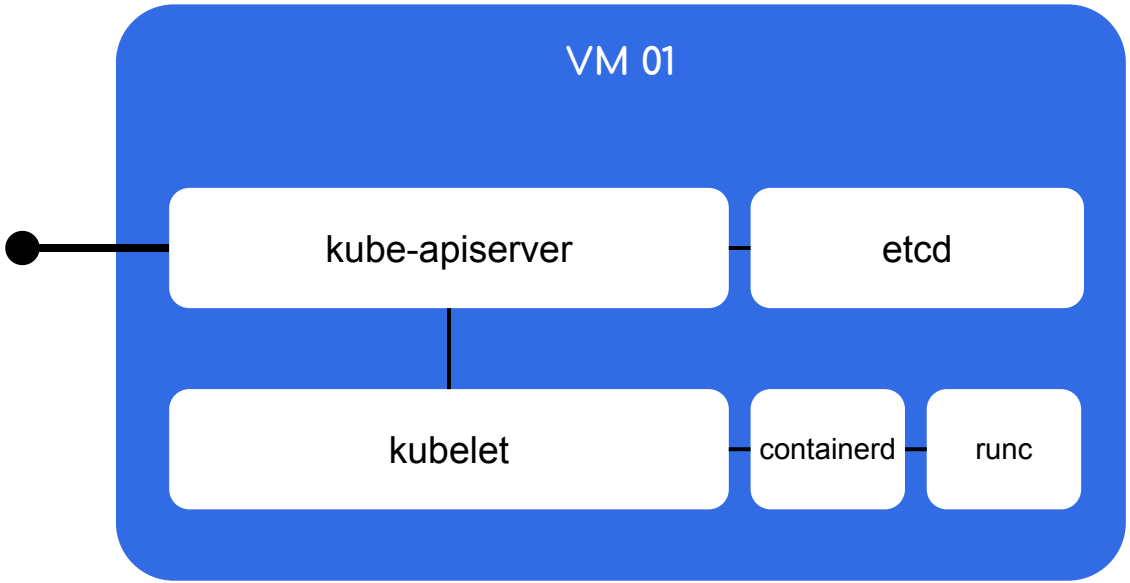
Adding etcd



> *A distributed,*
> *reliable key-value store [...]*
> *etcd*

- Stores configuration

Adding etcd



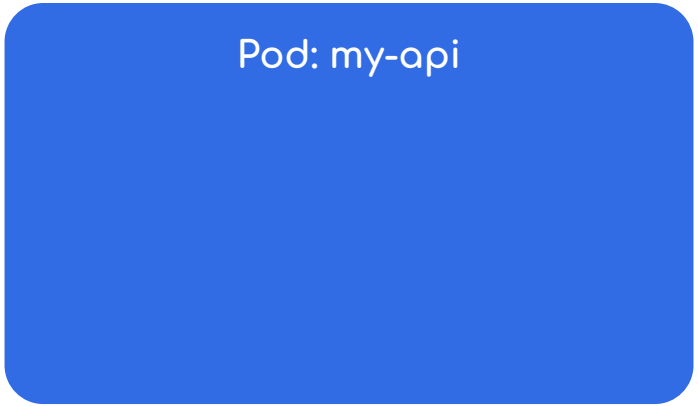
> *A distributed,*
> *reliable key-value store [...]*
> *etcd*

- Stores configuration
- Change Notification

Let's start a Container

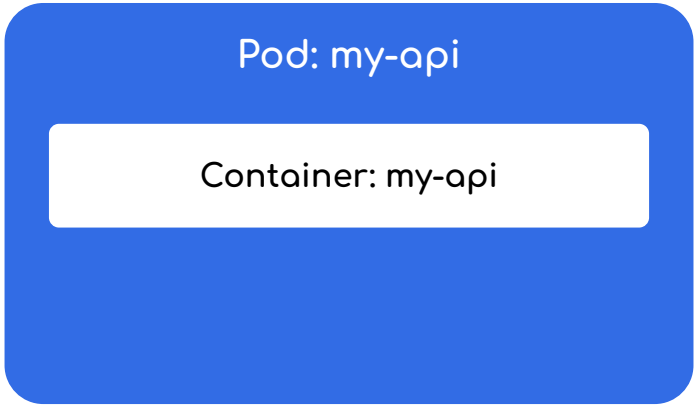
Let’s start a container!

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-api
spec:
  containers:
  - name: my-api
    image: my-api:1.14.2
```



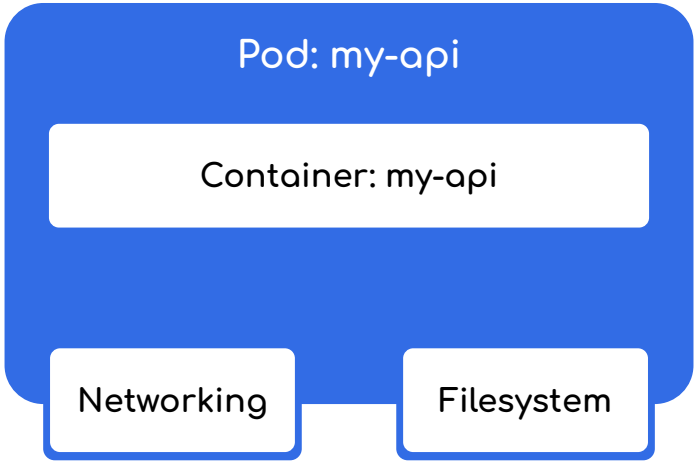
Let’s start a container!

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-api
spec:
  containers:
  - name: my-api
    image: my-api:1.14.2
```



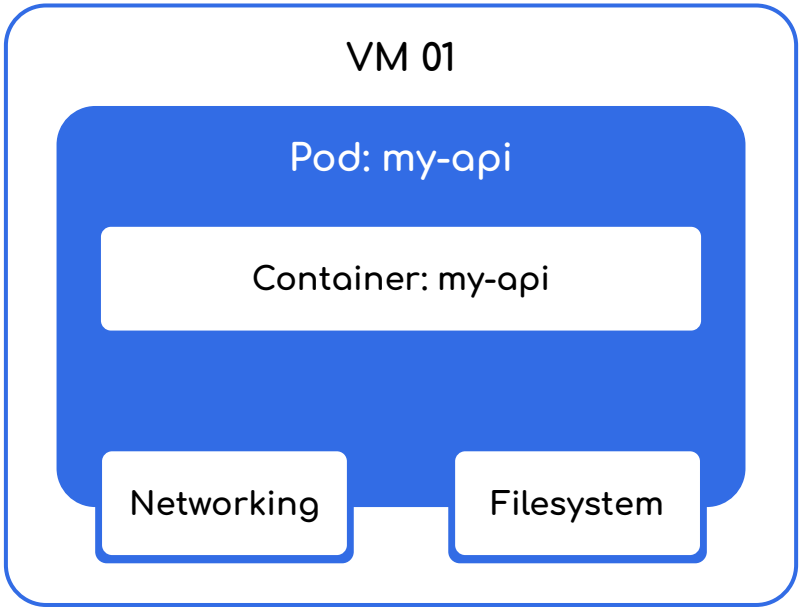
Let's start a container!

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-api
spec:
  containers:
  - name: my-api
    image: my-api:1.14.2
```



Let's start a container!

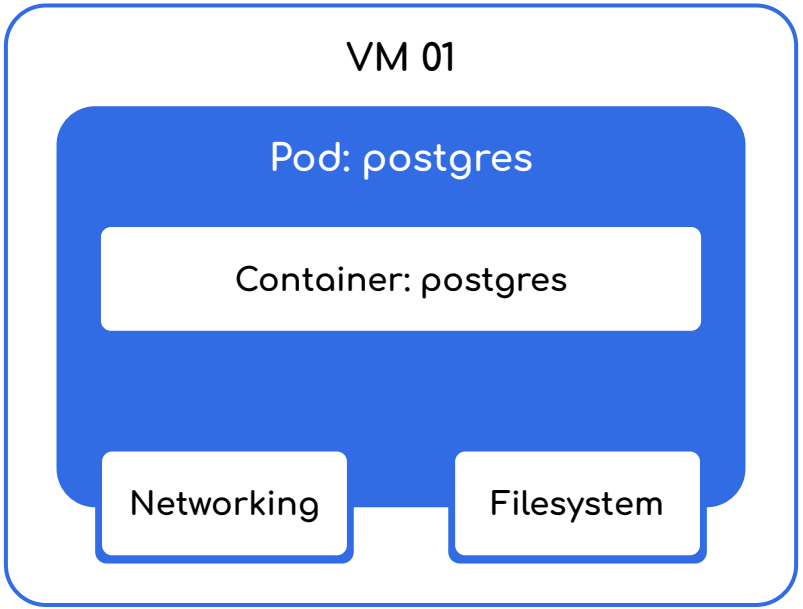
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-api
spec:
  containers:
  - name: my-api
    image: my-api:1.14.2
```



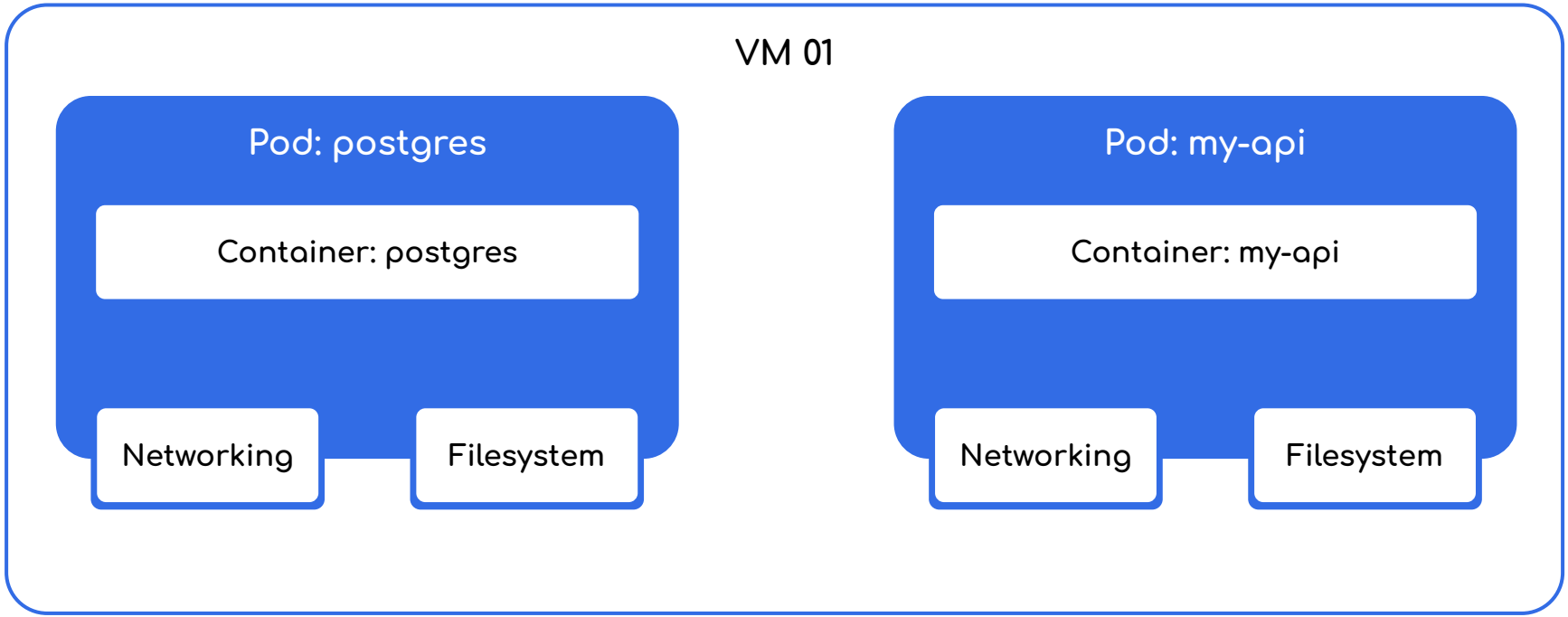
Let's start a 2nd Container

Let's start a second container!

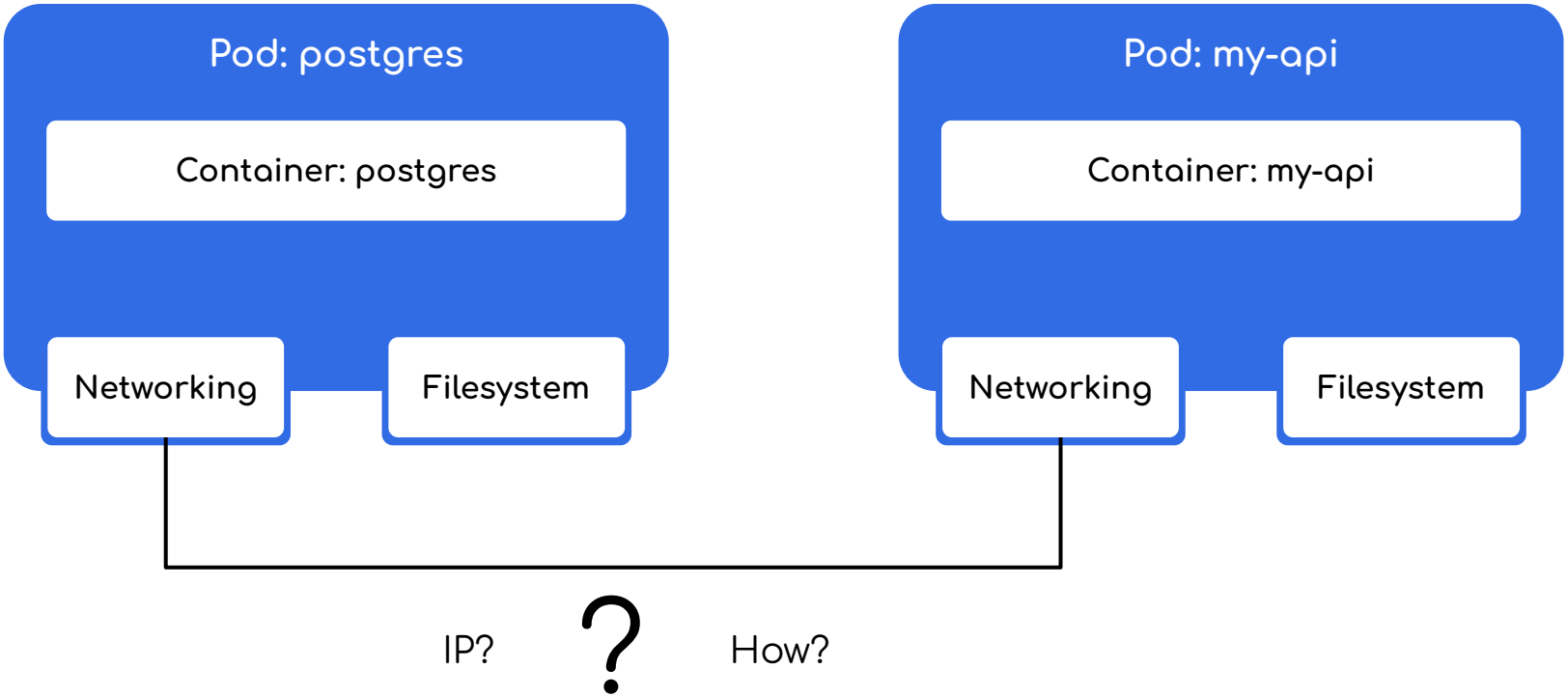
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: postgres
spec:
  containers:
  - name: postgres
    image: postgres:11.0.0
```



Let's start a second container!

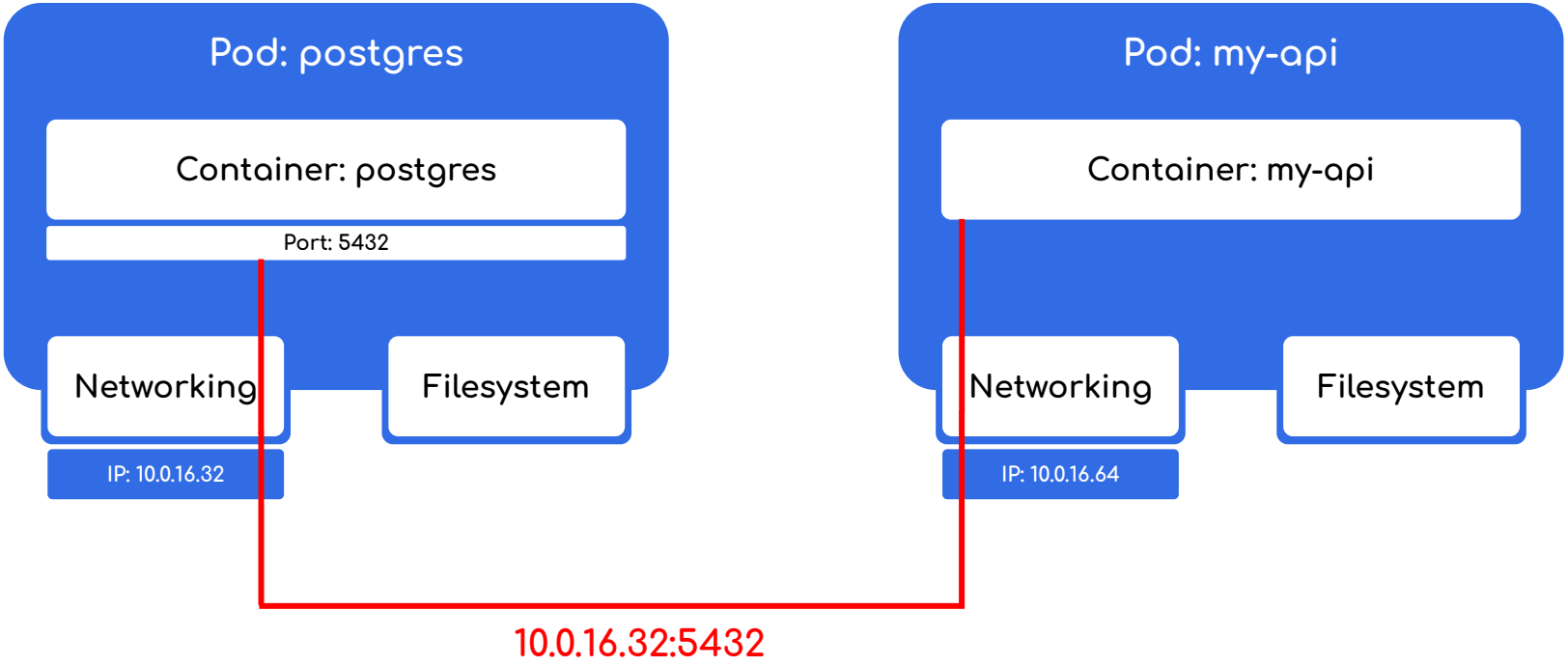


Let's start a second container!

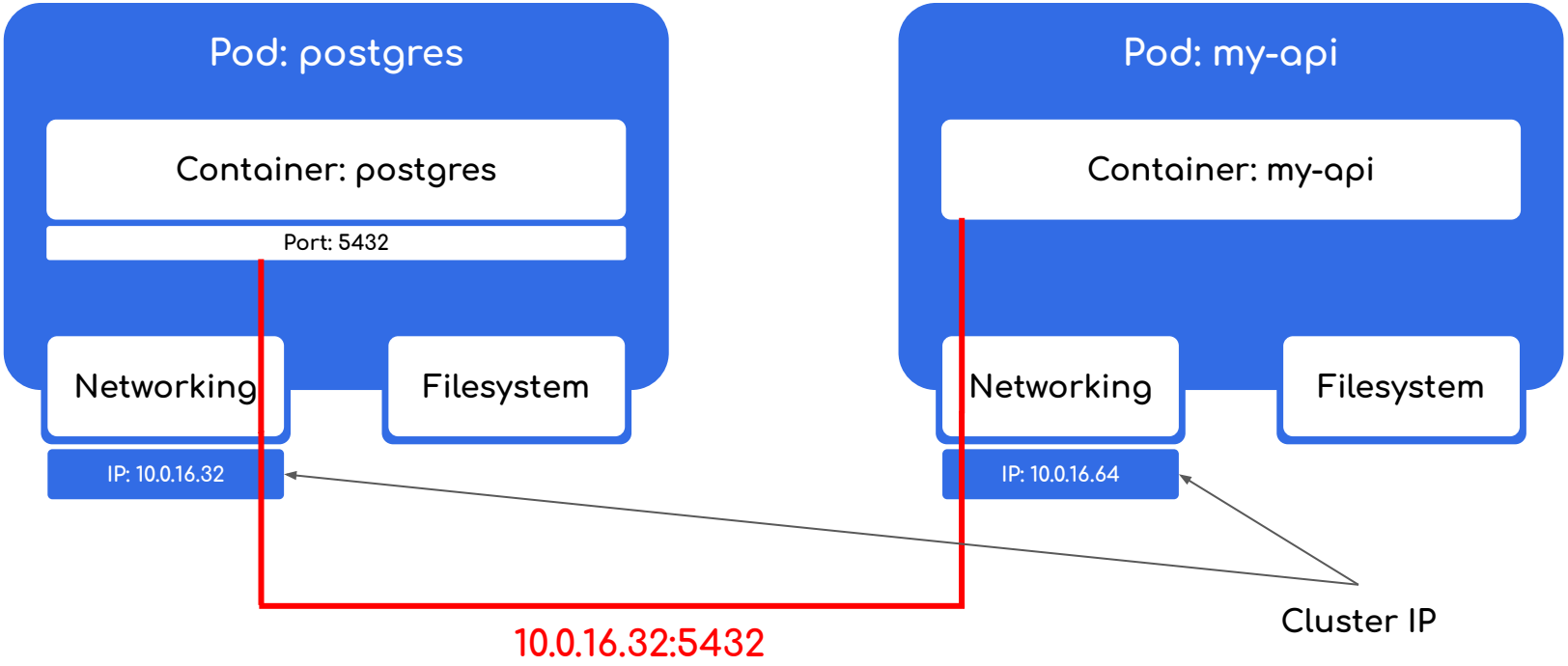


What about Networking

About Networking

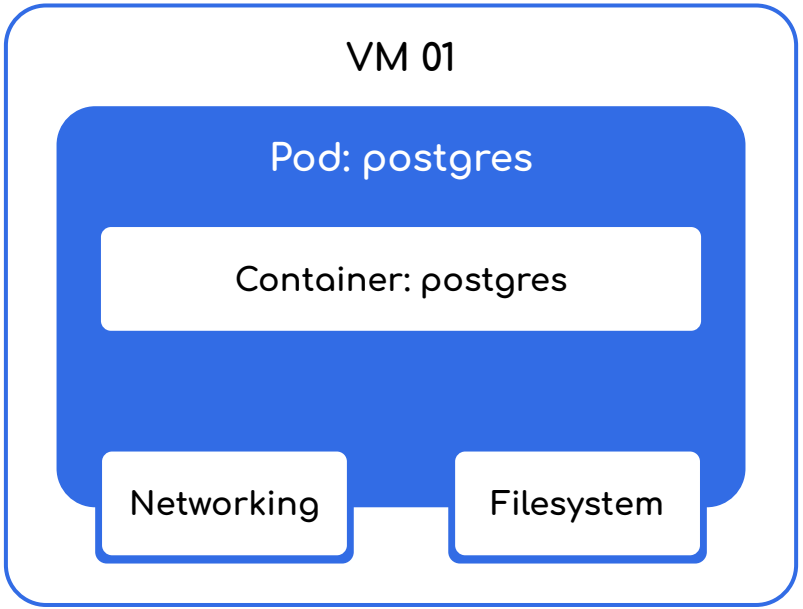


About Networking

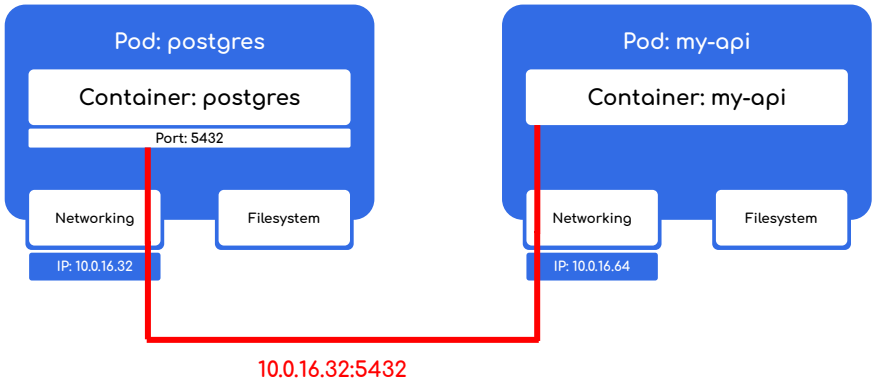


About Networking

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: postgres
spec:
  containers:
  - name: postgres
    image: postgres:11.0.0
    ports:
    - containerPort: 5432
      name: postgres_server
```



About Networking



- Unique IP per Pod (= ClusterIP)
- Each port listening on 0.0.0.0 is exposed!

What about Configuration

About configuration

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-api
spec:
  containers:
  - name: my-api
    image: my-api:1.14.2
    ports:
    - containerPort: 80
      name: http
    env:
    - name: CONNECTION_STRING
      value: postgresql://usr:pwd@10.0.16.32:5432/db
```

- Use environment variables
- Refresh configuration by restart

Introducing ConfigMaps

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: db-config
data:
  connection_string: postgres://....
```

- Decouples configuration

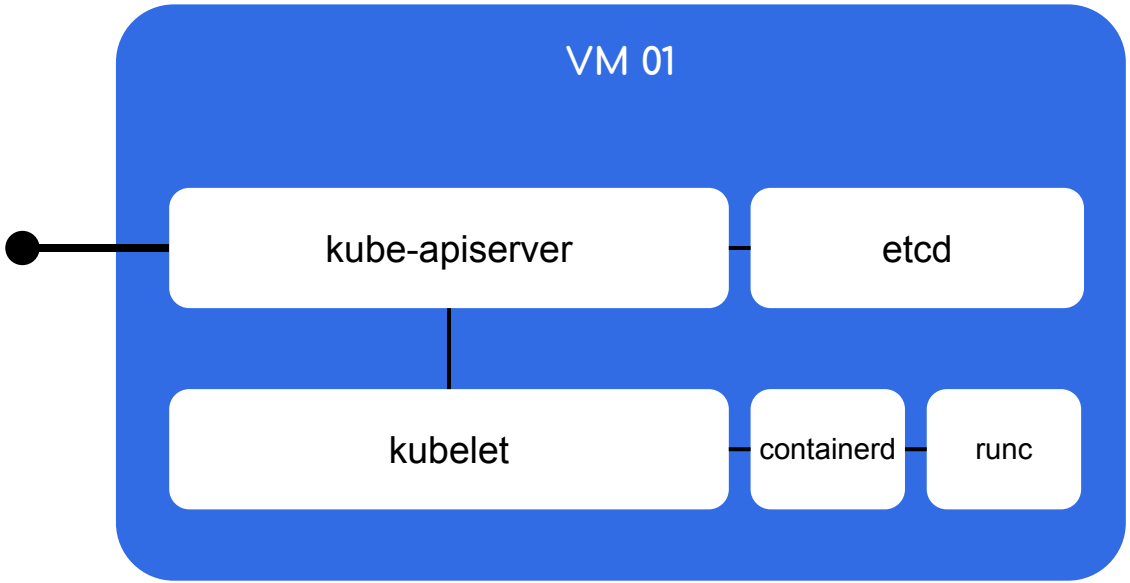
Using ConfigMaps

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-api
spec:
  containers:
  - name: my-api
    image: my-api:1.14.2
    ports: # ....
    env:
    - name: CONNECTION_STRING
      valueFrom:
        configMapKeyRef:
          name: db-config
          key: connection_string
```

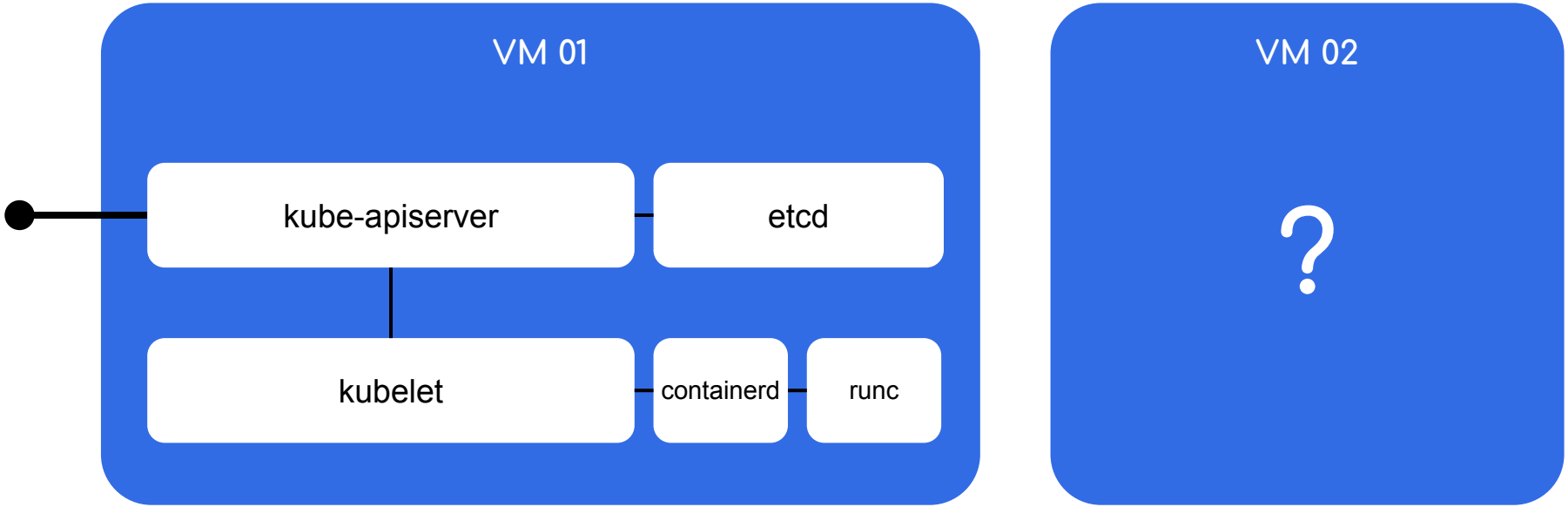
- Decouples configuration
- Allows to share configuration

Let's
Scale

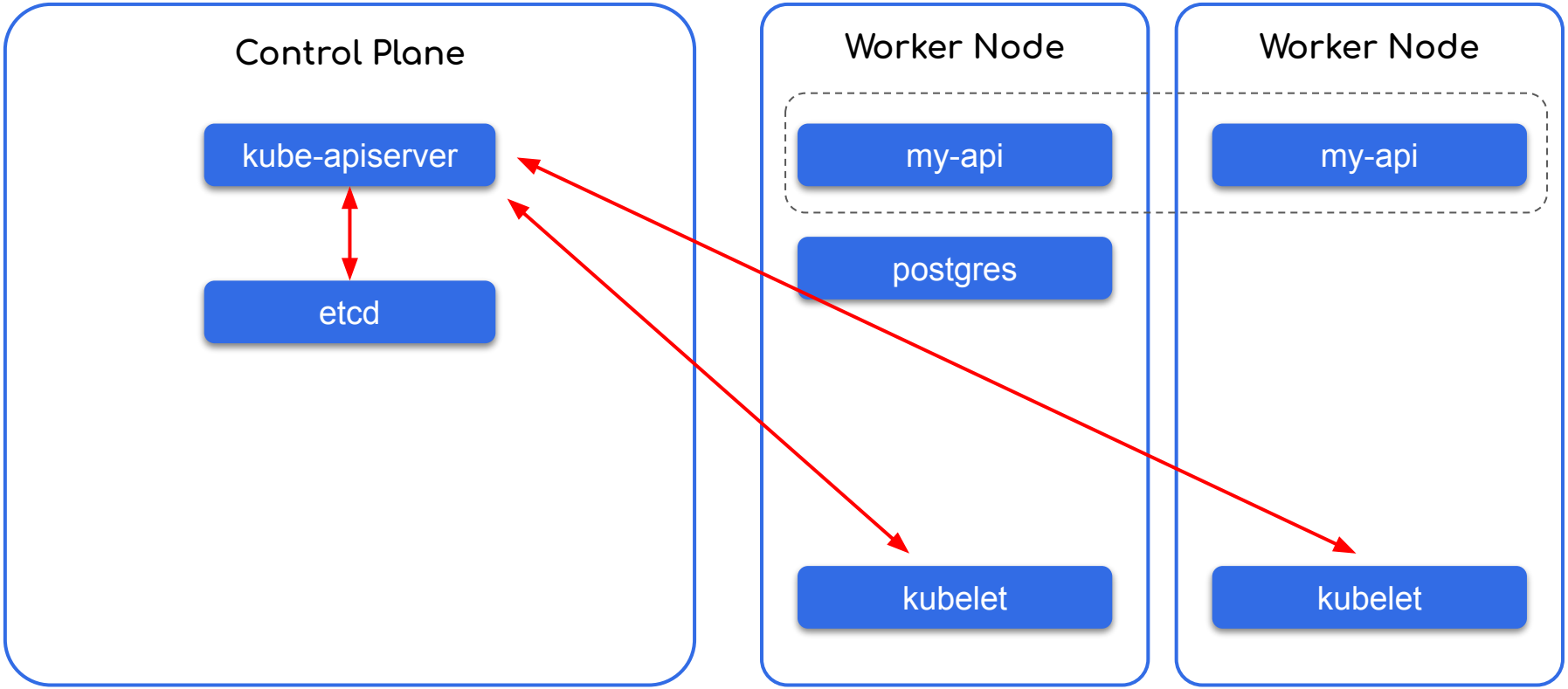
Let's scale



Let's add another VM



Separation of concerns



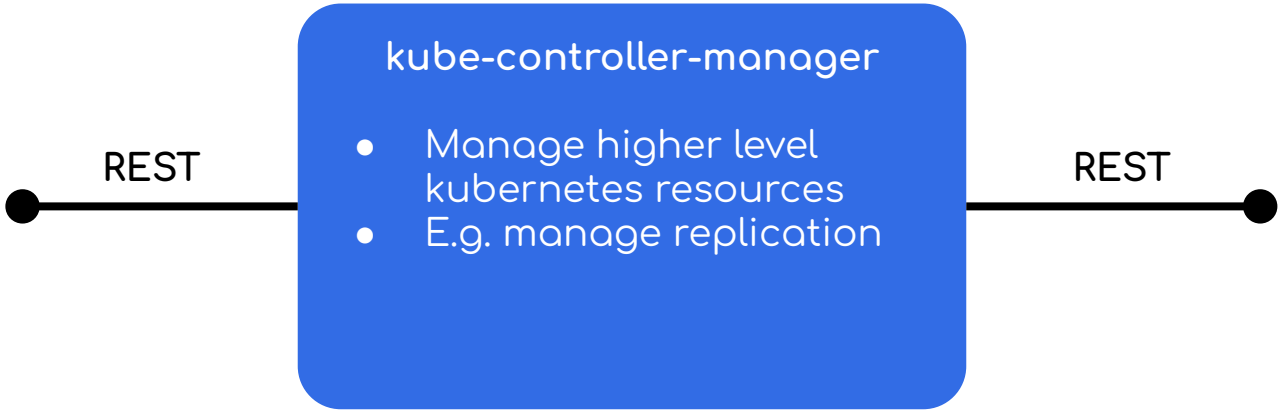
Introducing Deployments

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-api-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: my-api-pod
  template:
    metadata:
      labels:
        app: my-api-pod
    spec:
      containers:
      - name: my-api
        image: my-api:1.14.2
        # ...
```

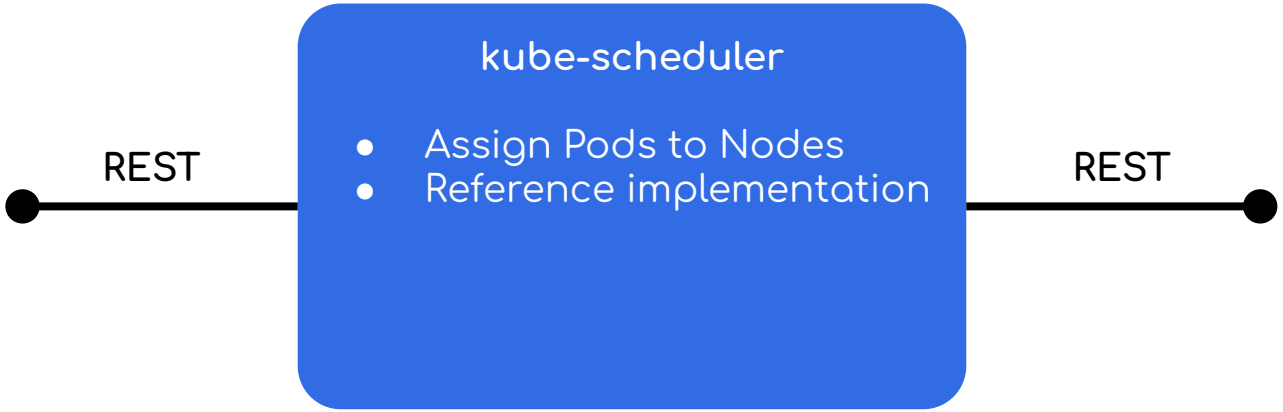
- Allows to run multiple replicas
- Template defines a single pod
- Each pod runs the same image

How do the containers get there

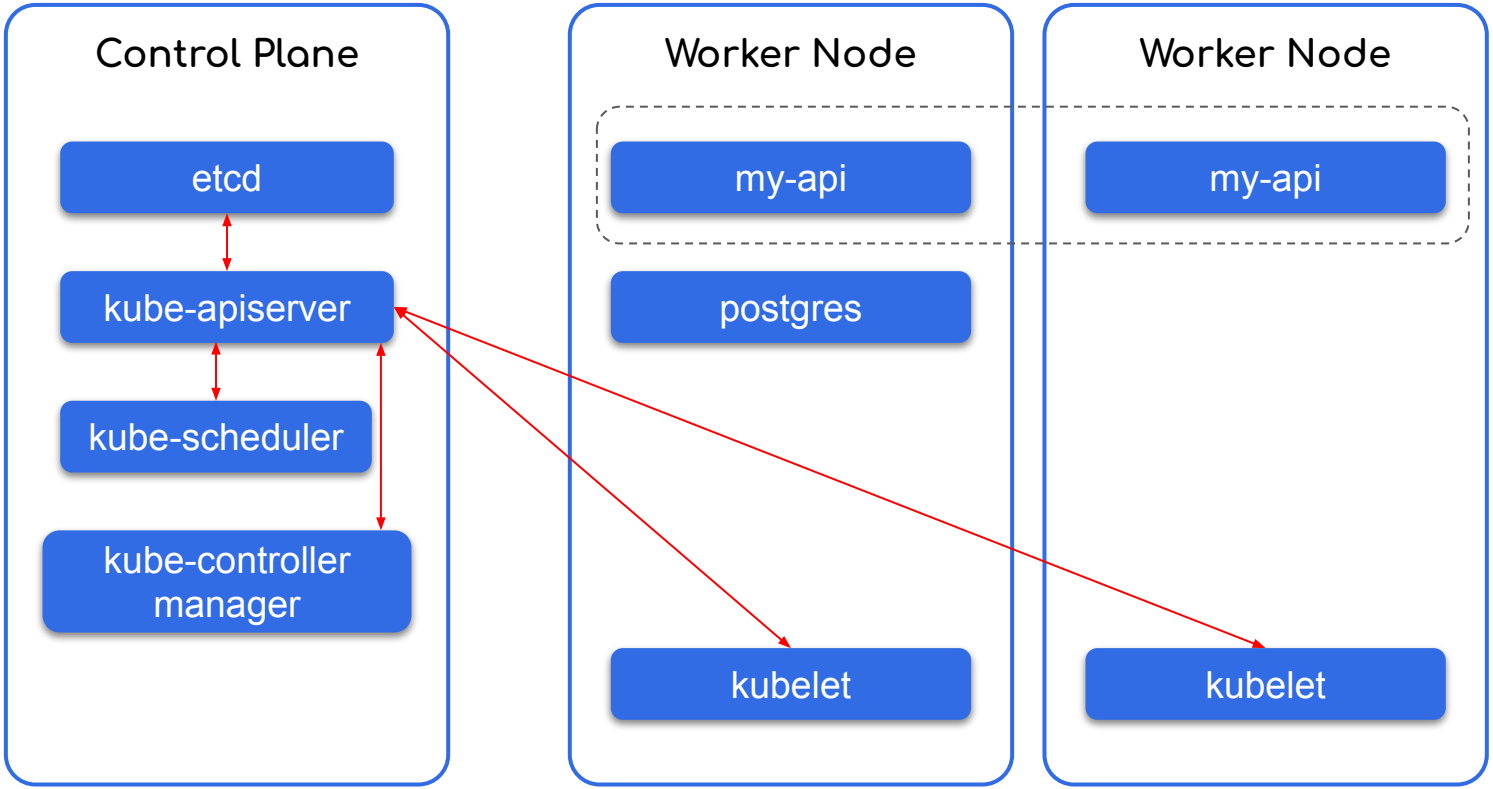
Adding kube-controller-manager



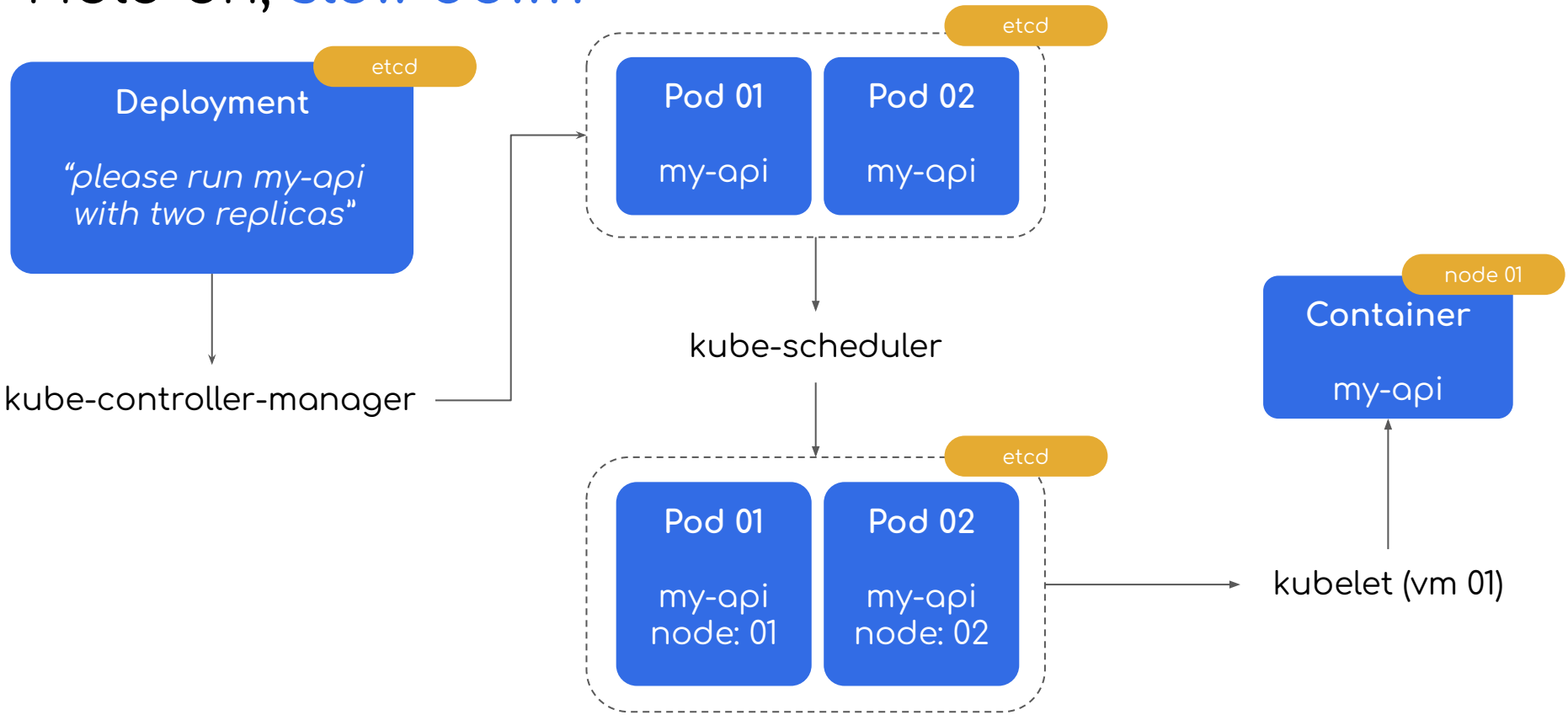
Adding kube-scheduler



Let's scale

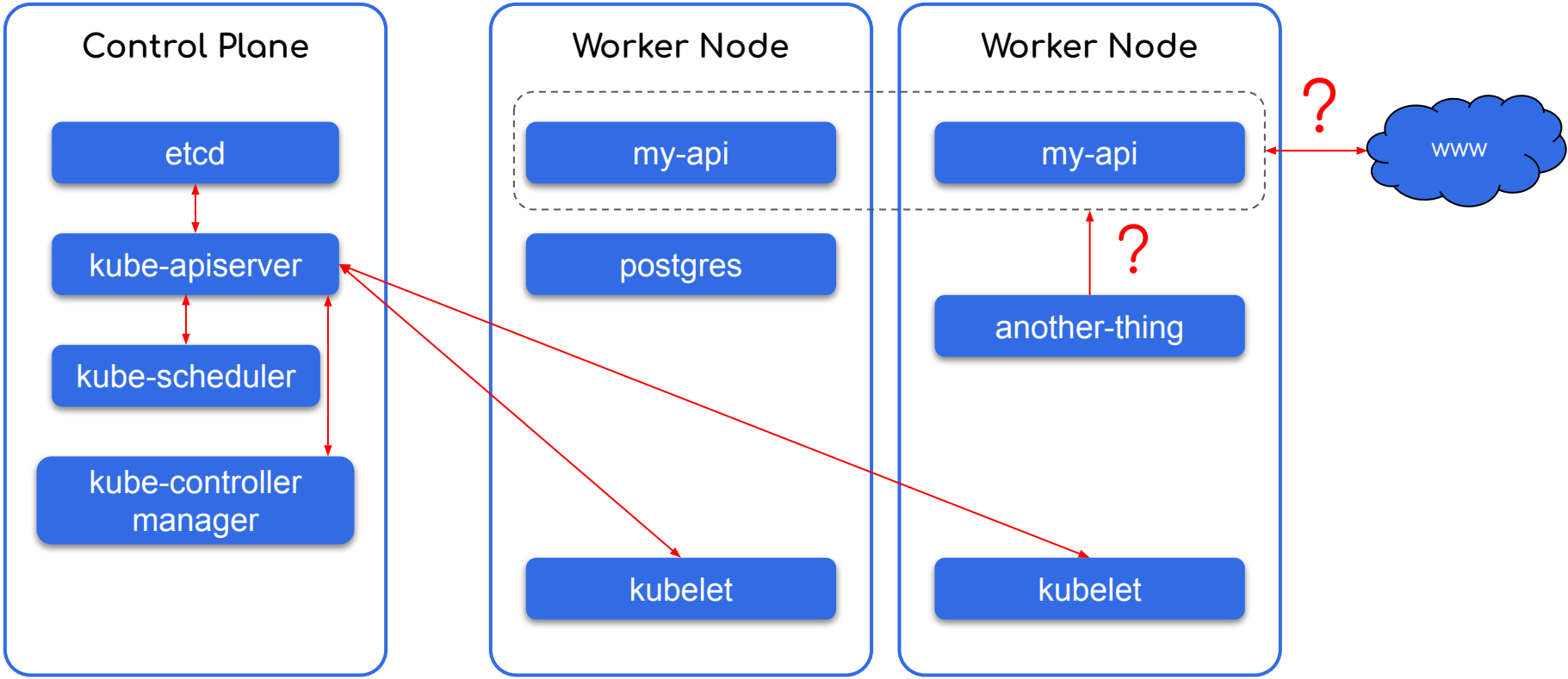

Control Plane
etcd
kube-apiserver
kube-scheduler
kube-controller manager
Worker Node
my-api
postgres
kubelet
Worker Node
my-api
kubelet

Hold on, slow down



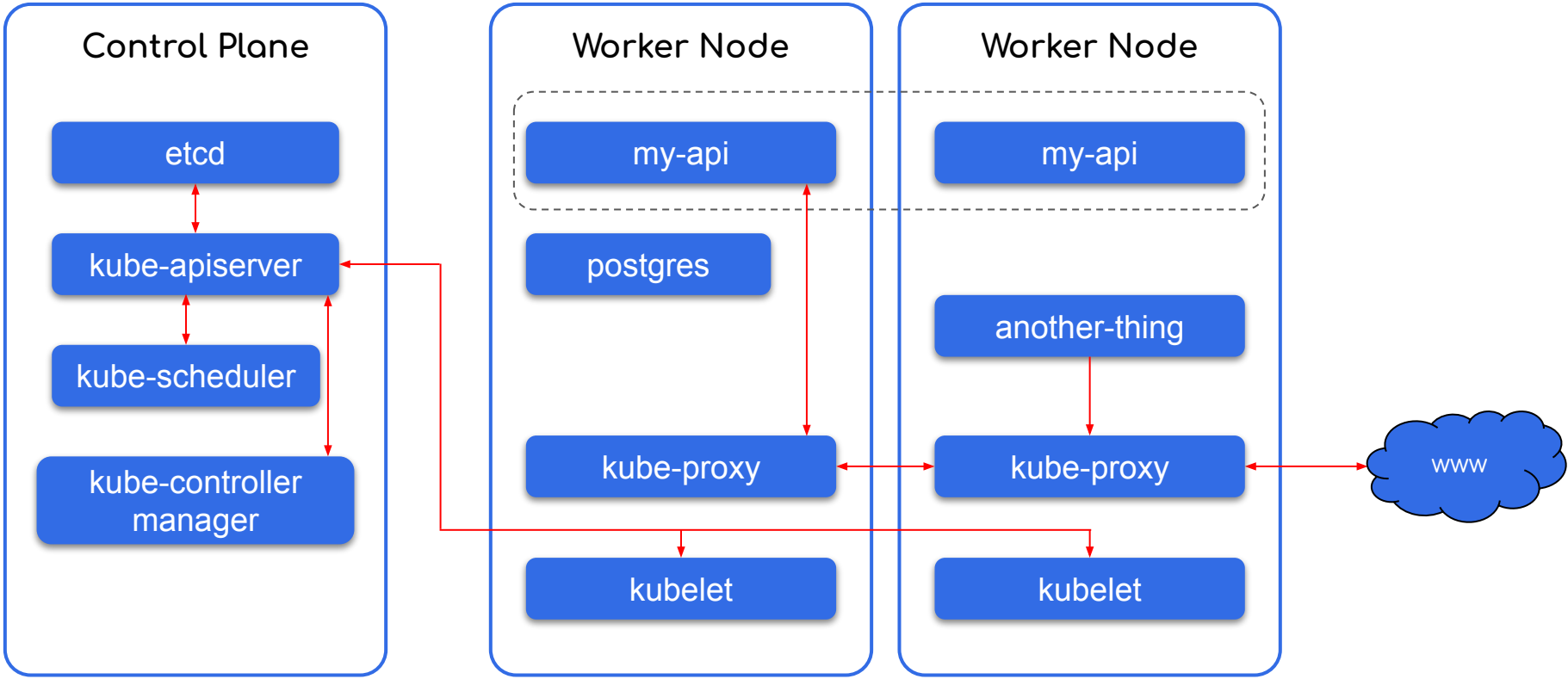
Networking
with replication

Networking with replication

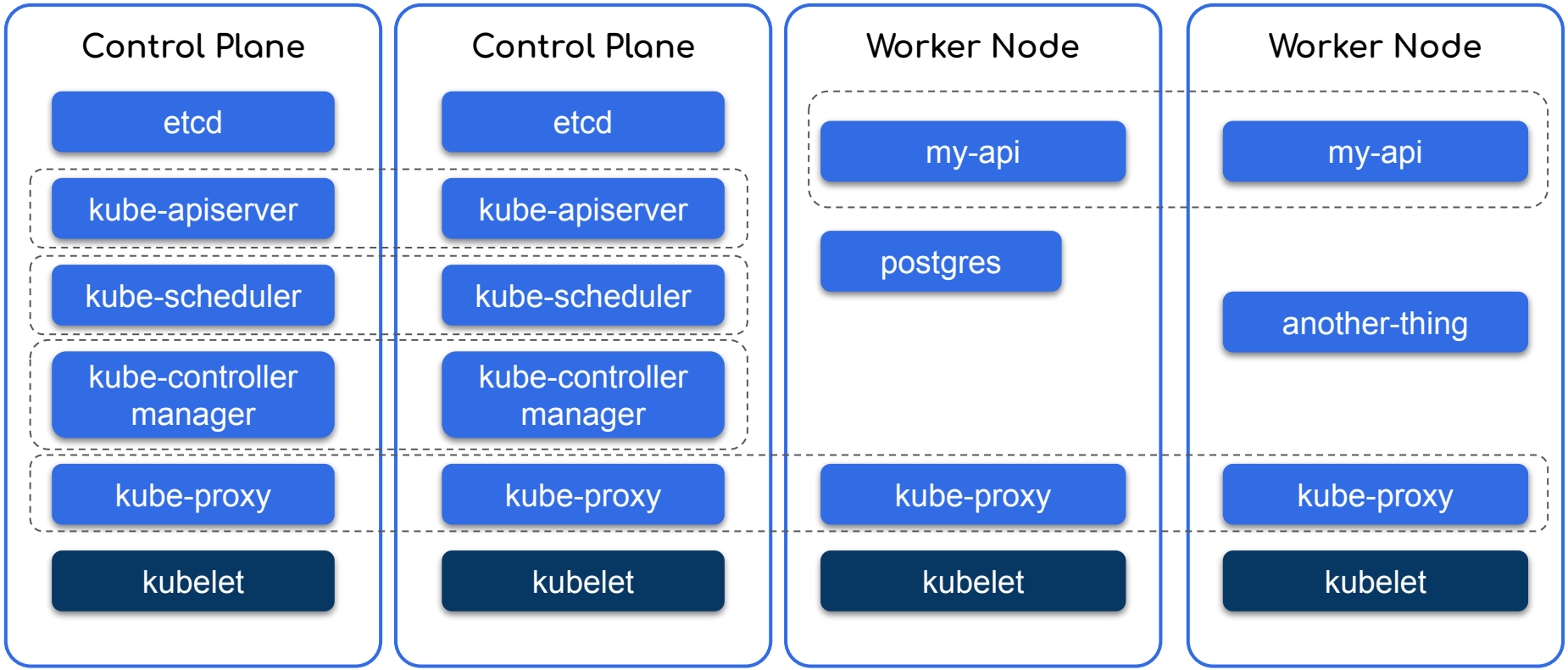


Adding kube-proxy

Adding kube-proxy



Adding control plane redundancy

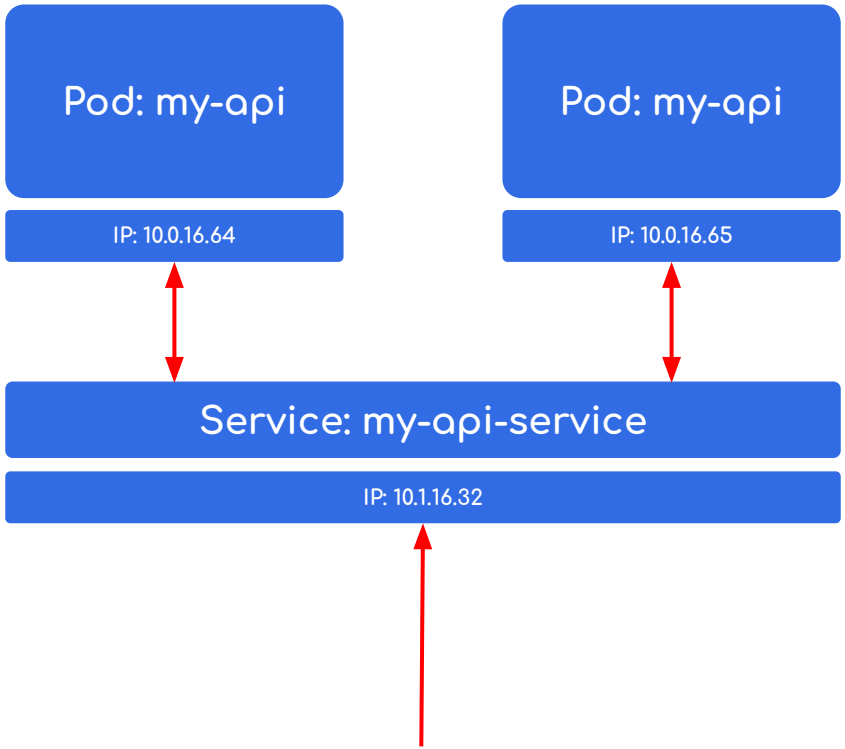


Using services

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

- Pods are ephemeral
- Pod IPs are ephemeral
- Services provide load-balancing
- Services are handled by kube-proxy
- Services can provide external connectivity

Using services



There is a lot more!

Ingress

Cert-Manager

Service Meshes

RBAC

Storage

Tooling

Autoscaling