Kubernetes Continued

Cloud-native PhD Course at LTH

Fall 2019

Lars Larsson

Kubernetes Under the Hood

etcd

OpenID Connect

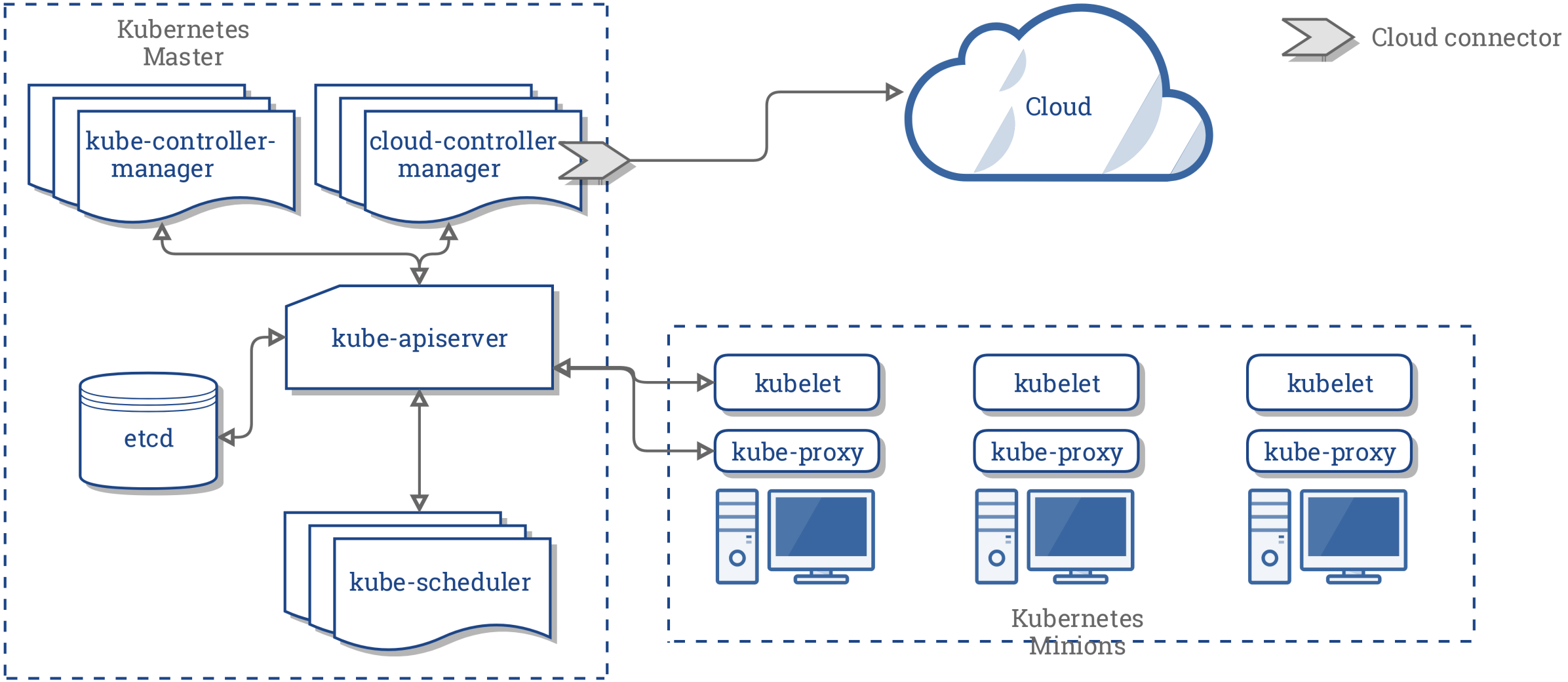
Design Patterns

Helm Package Manager

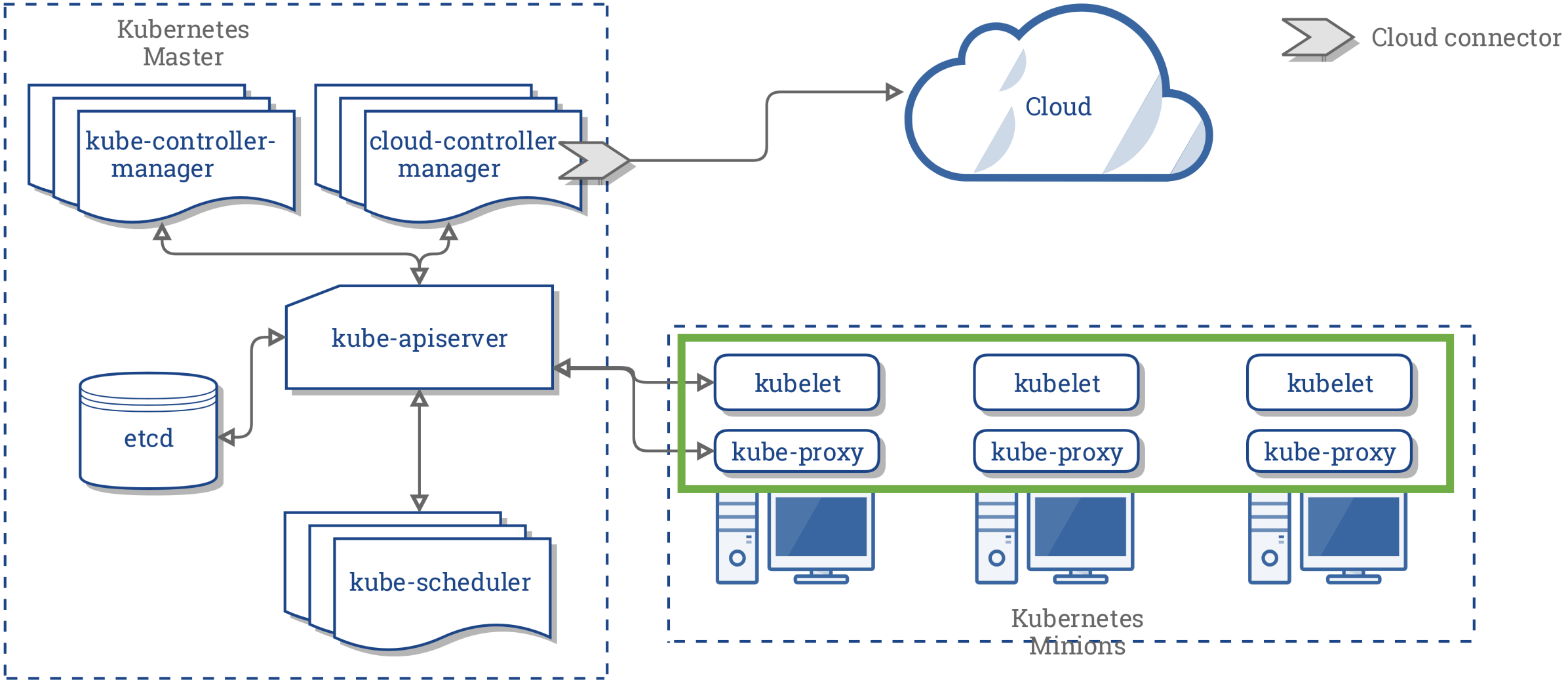
Kubernetes Under the Hood

- Architecture
- Networking
- Security
  - Network Policies
  - Role-Based Access Control to Kubernetes API
- Storage
- Extensions

Architecture



Networking


Kubernetes
Master
kube-controller-
manager
cloud-controller
manager
Cloud
Cloud connector
kube-apiserver
etcd
kube-scheduler
kubelet
kubelet
kubelet
kube-proxy
kube-proxy
kube-proxy
Kubernetes
Minions

Networking :: Pod to Pod

- Container Network Interface (CNI)
  - Many different providers
  - Different functionality (L2 or L3 in OSI terms)
  - Flannel, Calico, and Weave most common

- Essentially tunnel inter-Pod traffic in e.g. UDP packets between nodes
  - Use protocol like ARP, BGP, or even just data stored etcd for destination

- Read more
  - Cluster Networking

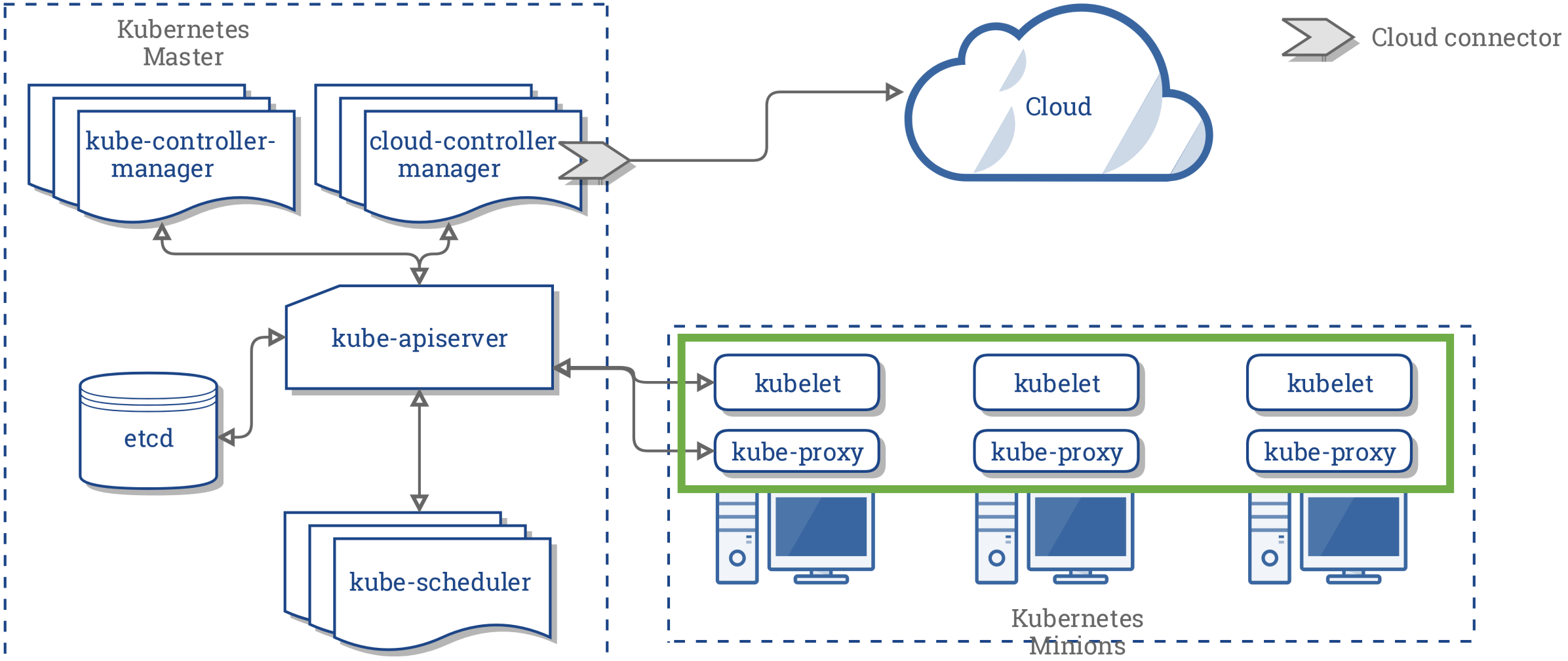
Networking :: Services

- Services get virtual IPs reachable on each host
  - Forwards traffic to registered Endpoints (Pods w/ successful liveness)
  - Managed by kube-proxy

- Service implementations
  - (user-space – no longer used)
  - Iptables (common default)
  - IPVS (high-performance and high scalability)

- Further reading
  - Services

Security :: Network Policies



Security :: Network Policies

- Limit incoming and outgoing traffic to/from Pods
  - Think "firewall"
- Requires support from CNI provider!
- Example YAML definition on next slide
- Further reading
  - Network Policies

Security :: Network Policy :: Example

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

Pods this policy pertains to

Allow incoming from IP range...

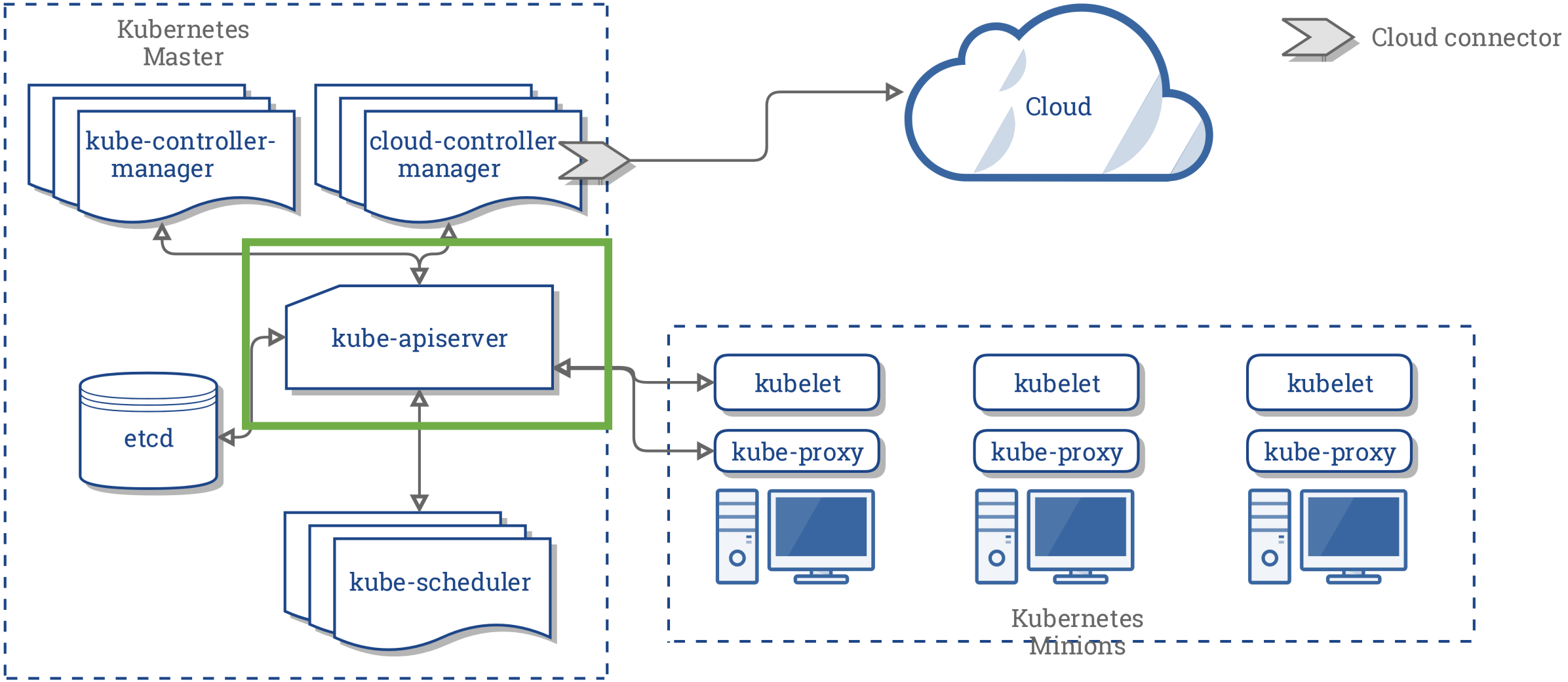
...or from any in this namespace...

...or any "frontend" labeled Pod...

...for TCP traffic on this port.

Allow outgoing only to this IP range and only TCP traffic on this port

Security :: Role-Based Access Control



Security :: Role-Based Access Control

- All API calls happen within context of some user
  - Can assign roles to users via a binding

- Role (namespaced) vs. ClusterRole (not confined to namespace)
- RoleBinding vs. ClusterRoleBinding

- API Groups, Resources, Verbs
- Subjects {User, Group, ServiceAccount}

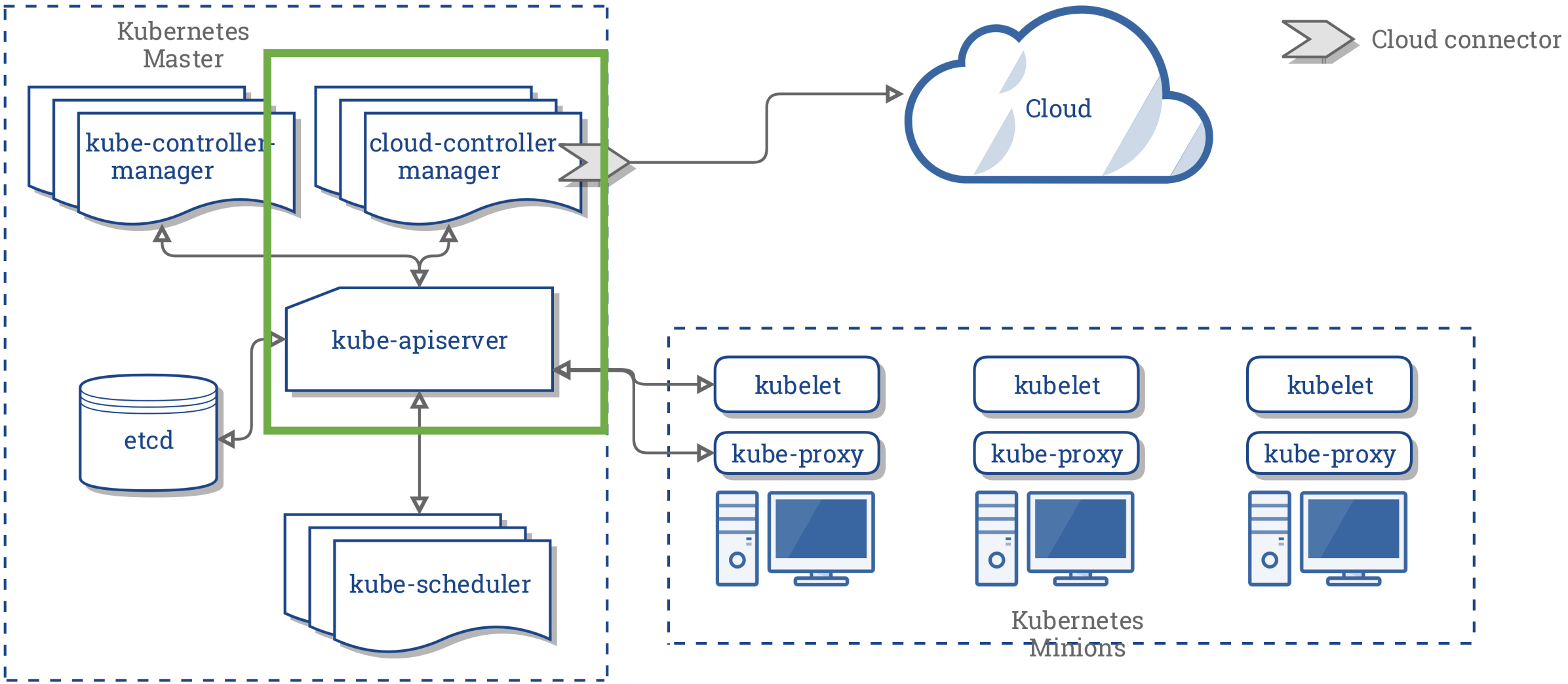
- Further reading
  - Using RBAC Authorization
  - Authenticating

Security :: Role-Based Access Control :: Example

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

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: jane # Name is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role #this must be Role or ClusterRole
  name: pod-reader # must match name of Role
  apiGroup: rbac.authorization.k8s.io
```

Storage


Kubernetes
Master
kube-controller-
manager
cloud-controller
manager
kube-apiserver
etcd
kube-scheduler
Cloud
Cloud connector
kubelet
kube-proxy
kubelet
kube-proxy
kubelet
kube-proxy
Kubernetes
Minions

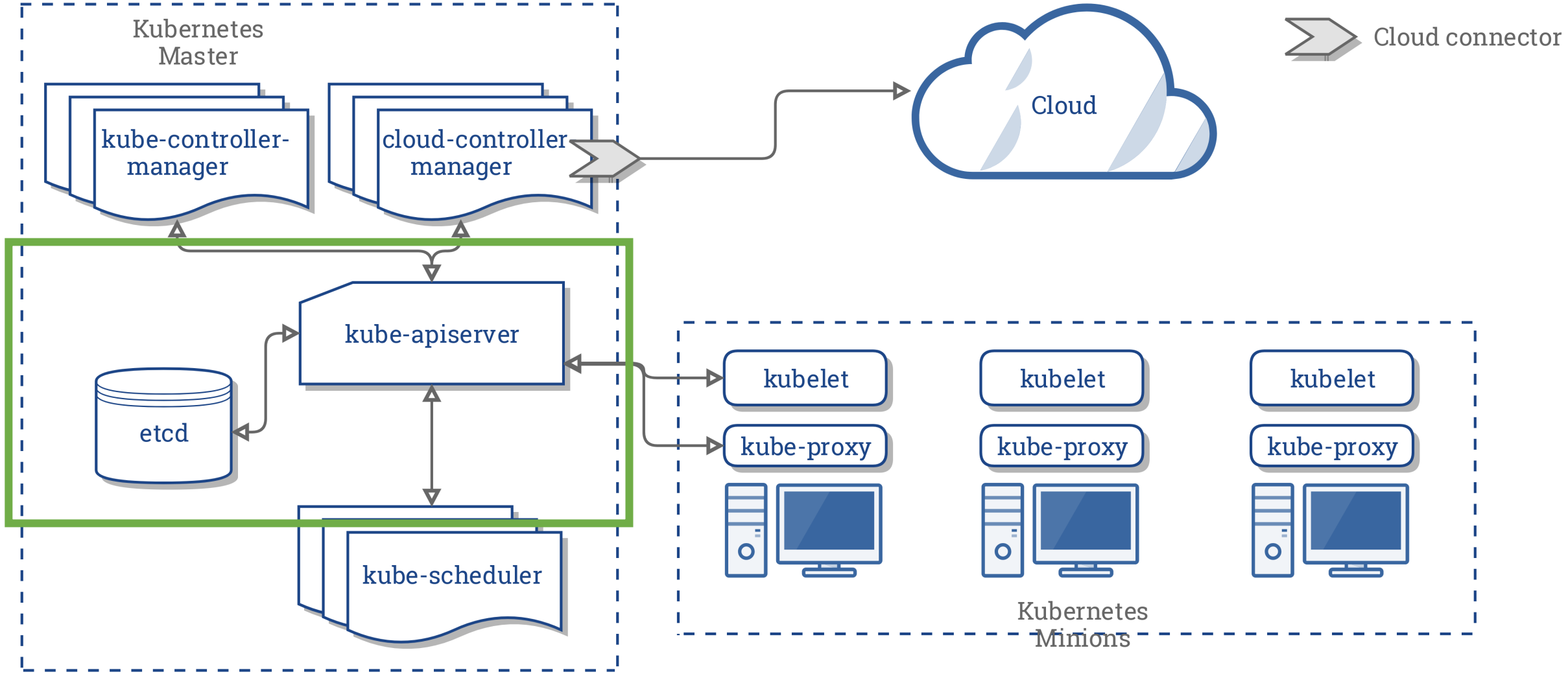
Storage

- Persistent Volume support via Container Storage Interface (CSI)
  - Cloud-specific ones (e.g. Cinder)
  - Networked File Systems (e.g. NFS, GlusterFS, …)
  - Proprietary ones (e.g. Portworx)

- StorageClass chosen for Persistent Volume

- Further reading
  - Persistent Volumes
  - Storage Classes
  - Dynamic Volume Provisioning

Extensions


Kubernetes
Master
kube-controller-
manager
cloud-controller
manager
Cloud
Cloud connector
kube-apiserver
etcd
kube-scheduler
kubelet
kube-proxy
kubelet
kube-proxy
kubelet
kube-proxy
Kubernetes
Minions

Extensions

- Custom Resource Definition (CRD)
  - Add custom API object
  - Immediate first-class support in e.g. kubectl
- Typically CRD tied to some Controller / Operator
  - Acts upon data managed in CRD object
- Further reading
  - Custom Resources
  - Extend the Kubernetes API with CustomResourceDefinitions

Kubernetes Under the Hood

etcd

OpenID Connect

Design Patterns

Helm Package Manager

etcd

- Distributed key-value store

- ***The*** core component in Kubernetes
  - Single source of truth
  - Stores state of all API objects and all events that occur
  - API service frontend to etcd

- Raft protocol
  - Sensitive to slow networks and slow disk performance

- Further reading
  - Learning etcd
  - Raft Consensus Algorithm (has cool animation!)

Kubernetes Under the Hood

etcd

OpenID Connect

Design Patterns

Helm Package Manager

OpenID Connect

- Enables third-party authentication
  - Google, Facebook, Twitter, …
- Kubernetes can use OpenID Connect together with RBAC
  - Dex easy to set up
- Resource Server: service you want to use
- Client: software you use to authenticate (web browser!)
- Authentication Server / Identity Provider (IdP): service that authenticates you

OpenID Connect Flow

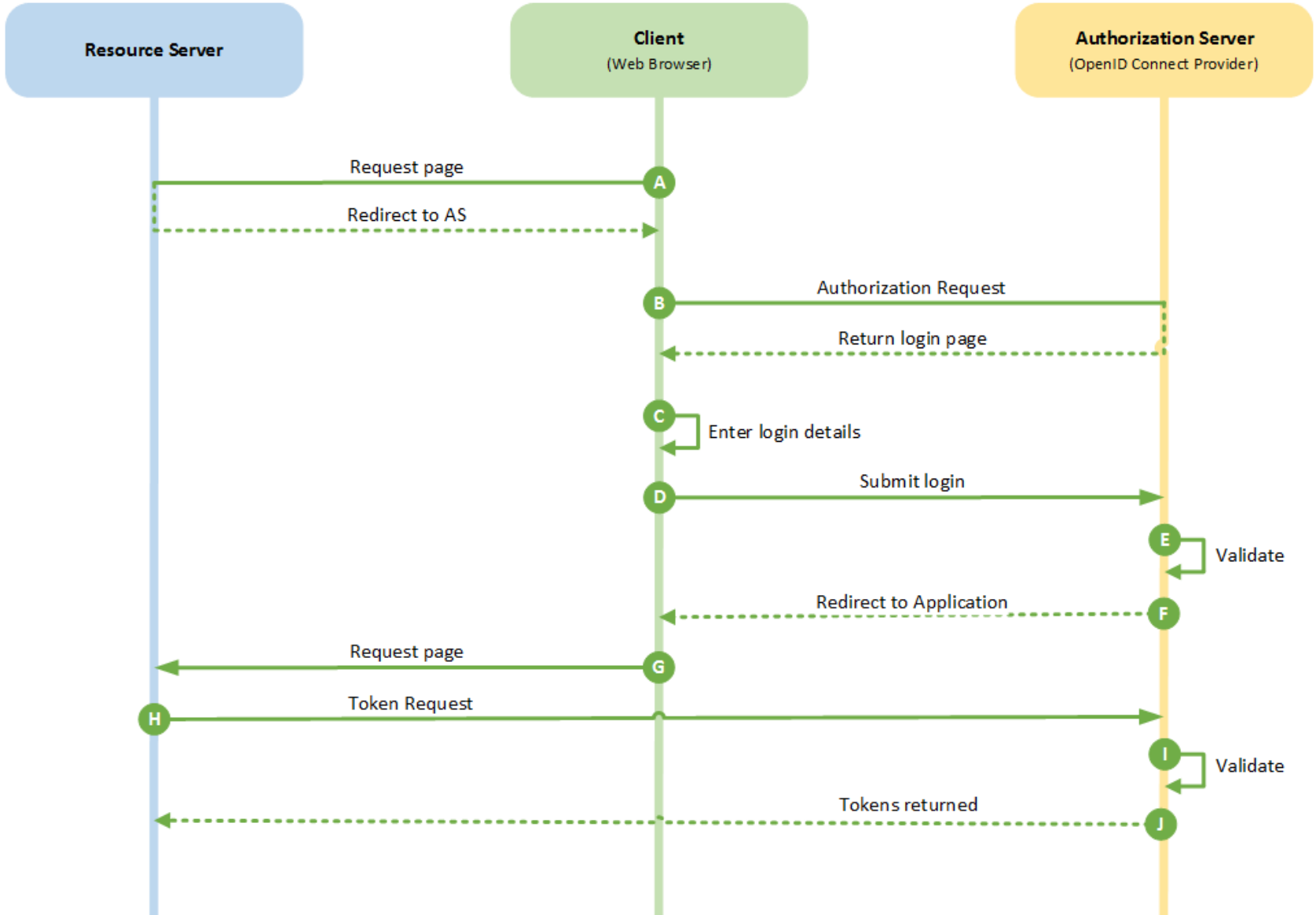


Image credits: [M&S Consulting]

Kubernetes Under the Hood

etcd

OpenID Connect

Design Patterns

Helm Package Manager

Design Patterns in Kubernetes

Sidecar

Ambassador

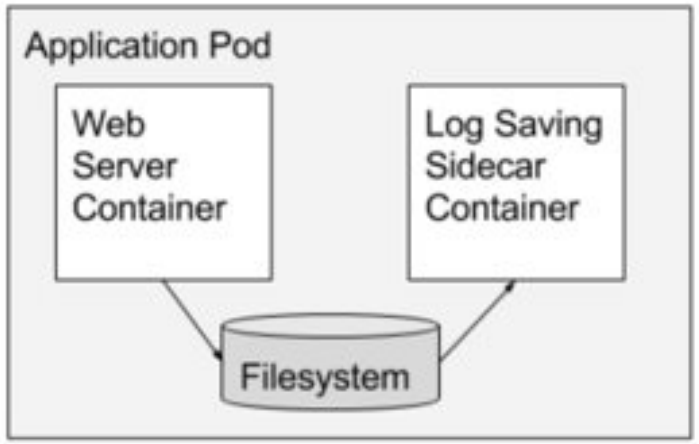
Adapter

Leader election

Work queue

Scatter/gather

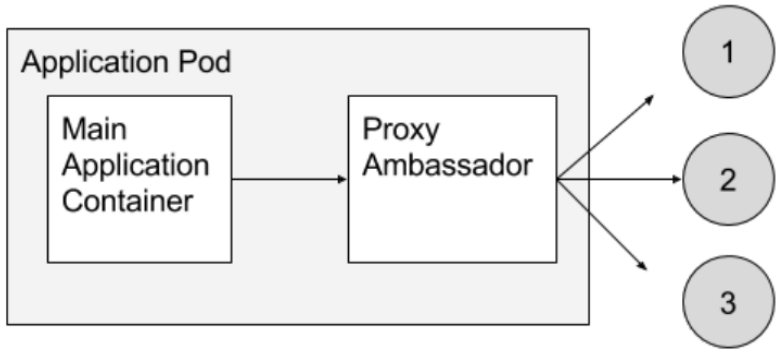
Design Pattern :: Sidecar



- One process per container
  - So do you bake in all functionality into one?
- Sidecar is a helper to main process
  - E.g. Synchronizing file contents, sampling requests, managing log files...
- Composability!

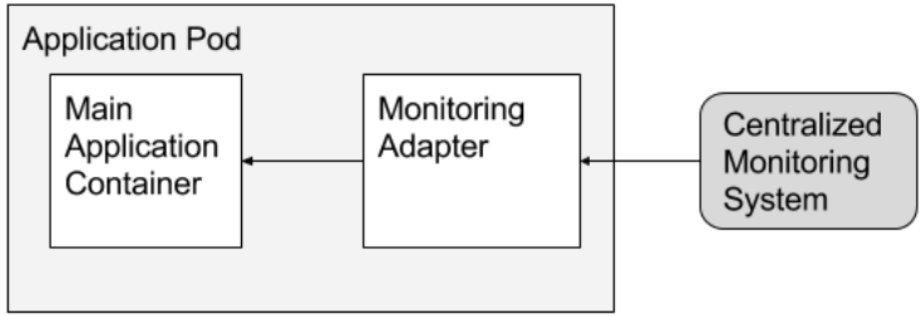
Design Pattern :: Ambassador

- Proxies and (typically) simplifies calls from main process to a service
- Hides complexity
- Reduces coupling



Design Pattern :: Adapter

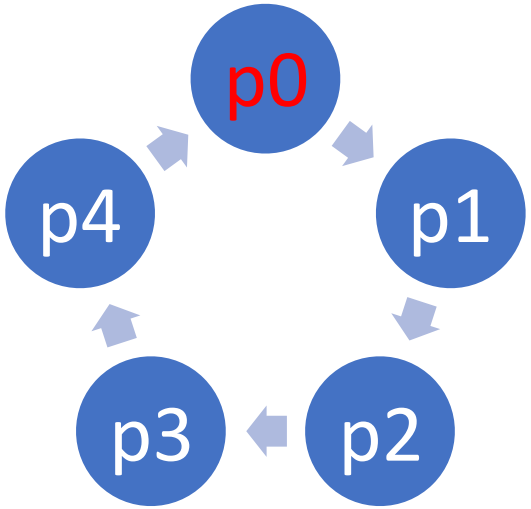
- Proxies and (typically) simplifies calls **to** main process ***from*** a service
- Hides complexity
- Reduces coupling



- E.g. a metrics collection service can assume that all our Pods are compatible with some protocol X – either natively or via an adapter

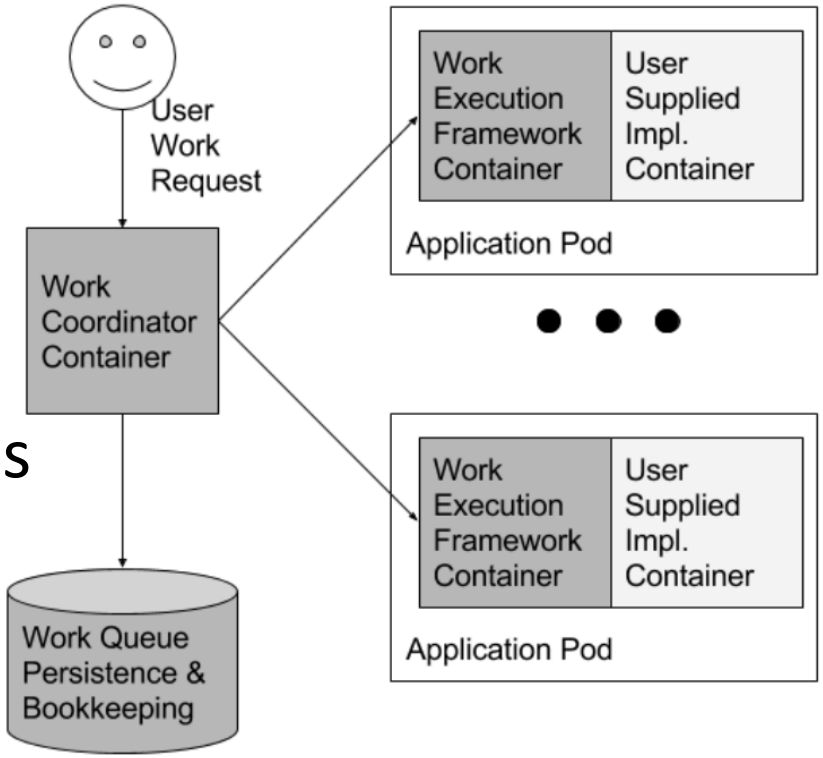
Design Pattern :: Leader election

- Distributed algorithms (upcoming lecture!) often need a leader or coodinator
  - This is very complex (and *theoretically impossible*)!
  - Why re-invent the wheel? Just offer a "leader election" container and interface with it!



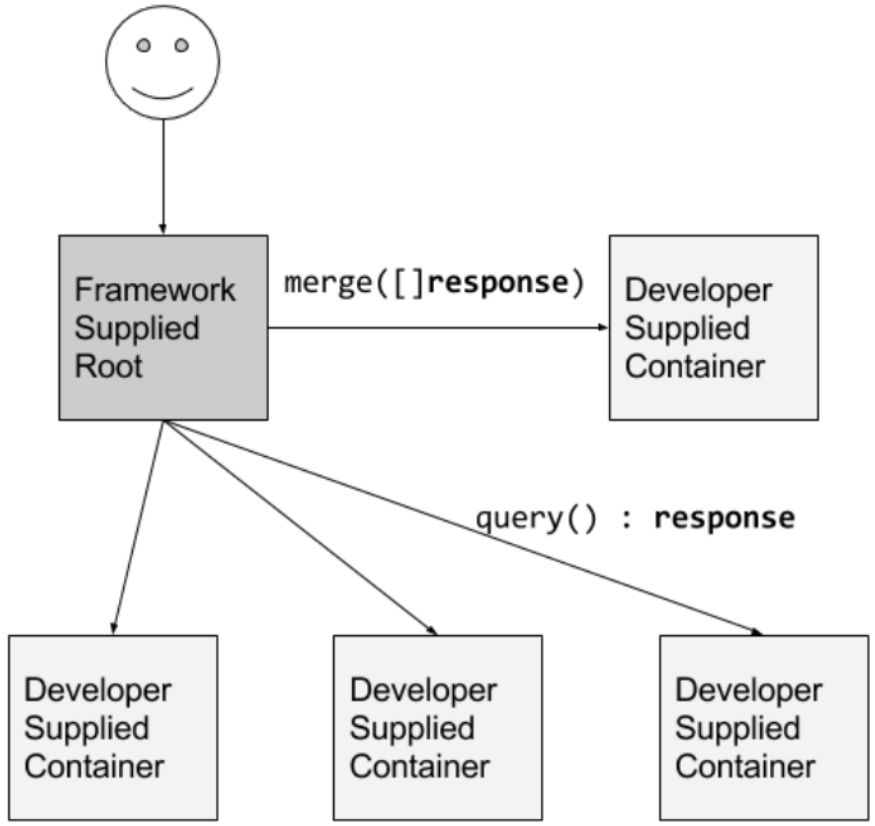
Design Pattern :: Work queue

- Data processing can often be done in parallel
- More workers = less waiting on results
- Create a work queue and spawn as many workers as you can (afford)
  - Each worker is assigned tasks and marks as completed
  - Worker failure? Work item not marked as finished, so other worker claims it



Design Pattern :: Scatter/gather

- Data processing can often be done in parallel
- More workers = less waiting on results
- Have workers work on sub-problems!
  - Coordinating process collects sub-results and creates overall result



Kubernetes Under the Hood

etcd

OpenID Connect

Design Patterns

Helm Package Manager

Helm Package Manager

- Fully featured applications can get quite complex!

- Helm lets you deploy applications as-a-whole ("Charts")
  - Template language for differences in deployments and variables
  - Manages life-cycle of entire application, not its parts

- Tiller: (optional but useful) server-side component
- Helm: command-line interface tool

- Further reading
  - Helm documentation

Summary

- Kubernetes architecture is highly modular
  - Plug-and-play, use what you want, make it yours
  - Extensible API as well
- etcd is the single source of all truth in Kubernetes
- OpenID Connect lets applications authenticate users via third party
- Design patterns help you quickly stand up a service with standard components and abstractions
- Helm package manager can help deploy even complex applications and manage their life-cycle

Next week - Distributed systems

The prisoner problem

There are *N* prisoners. At random times, the guards will randomly select one prisoner to visit a room in which a 2-way switch is located. A prisoner that is visiting the room can operate the switch at their will. The warden asks if the prisoners can tell him when all prisoners have been in the room at least one time each. If they are correct he will let them free, if they are wrong they will all be executed. Prisoners have no way of communicating with each other apart from during a brief session before the process starts during which they will have to agree upon a scheme for how to solve the task. Moreover, it is not possible to detect the current status or change of the switch's state from outside the room (i.e., no lamp light is visible through a window or through the door, you cannot hear the sound of the switch changing, *et cetera*).