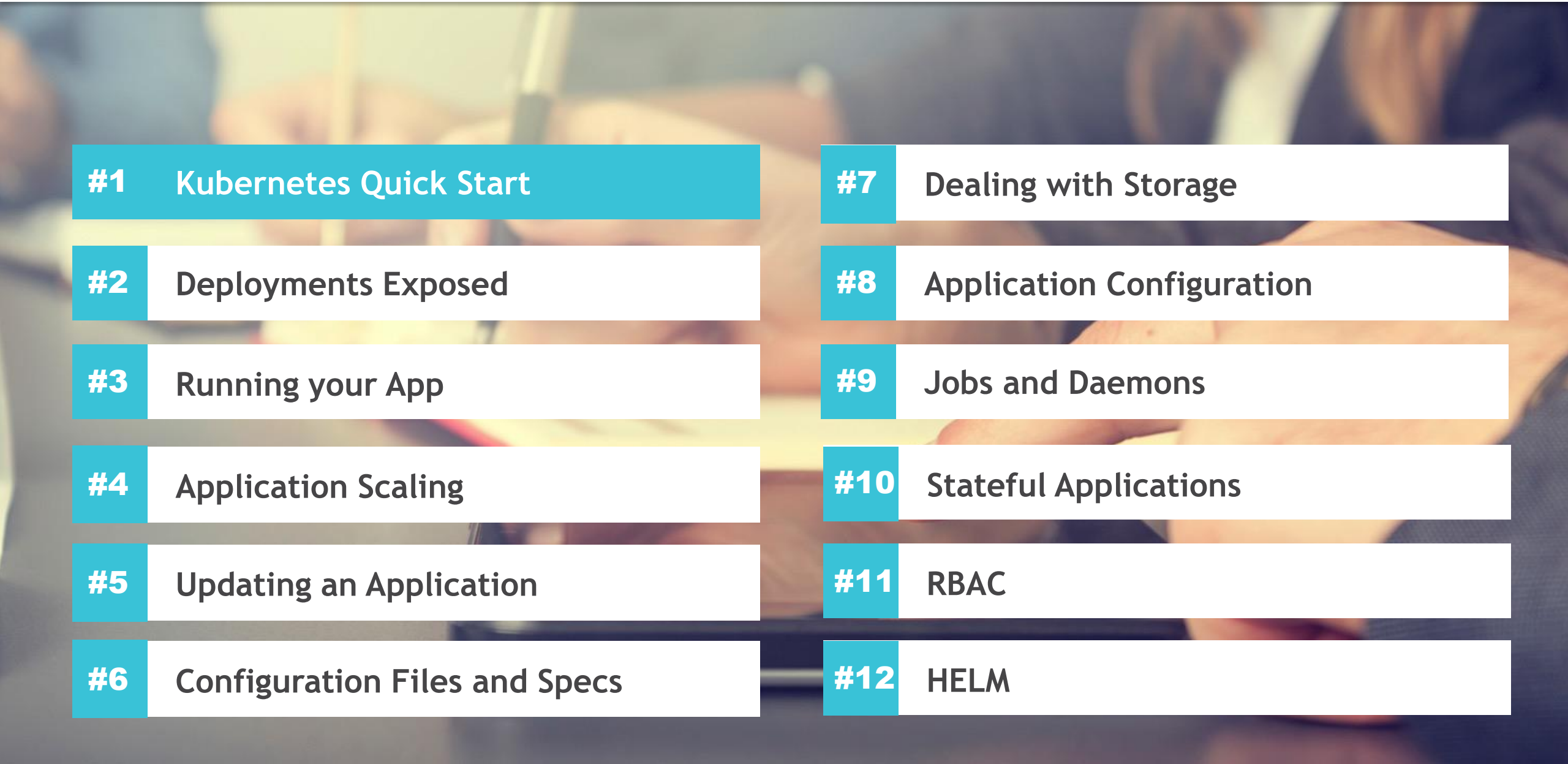

epam
KUBERNETES
2021

KUBERNETES QUICK START



#1	Kubernetes Quick Start
#2	Deployments Exposed
#3	Running your App
#4	Application Scaling
#5	Updating an Application
#6	Configuration Files and Specs
#7	Dealing with Storage
#8	Application Configuration
#9	Jobs and Daemons
#10	Stateful Applications
#11	RBAC
#12	HELM

KUBERNETES QUICK START - DOCKER

Q&A

- What is Docker?
- Why we need it?
- What is the difference between a Docker Container and a Virtual Machine?

KUBERNETES QUICK START – What is Kubernetes?

KUBERNETES QUICK START – Why Kubernetes exist?

- Immutable
- Declarative
- Self-healing
- Decoupling

KUBERNETES QUICK START - KUBERNETES OVERVIEW

KUBERNETES

- An open-source platform
- Founded by Google
- Schedules and manages containers
- Provides service discovery
- Provides load balancing
- Provides autoscaling
- Provides high availability
- Provides declarative updates

KUBERNETES CONCEPTIONS

- Nodes
- Namespace
- Pods
- Controllers
- Labels
- Volumes
- Job
- Kubectl

KUBERNETES QUICK START - Core components

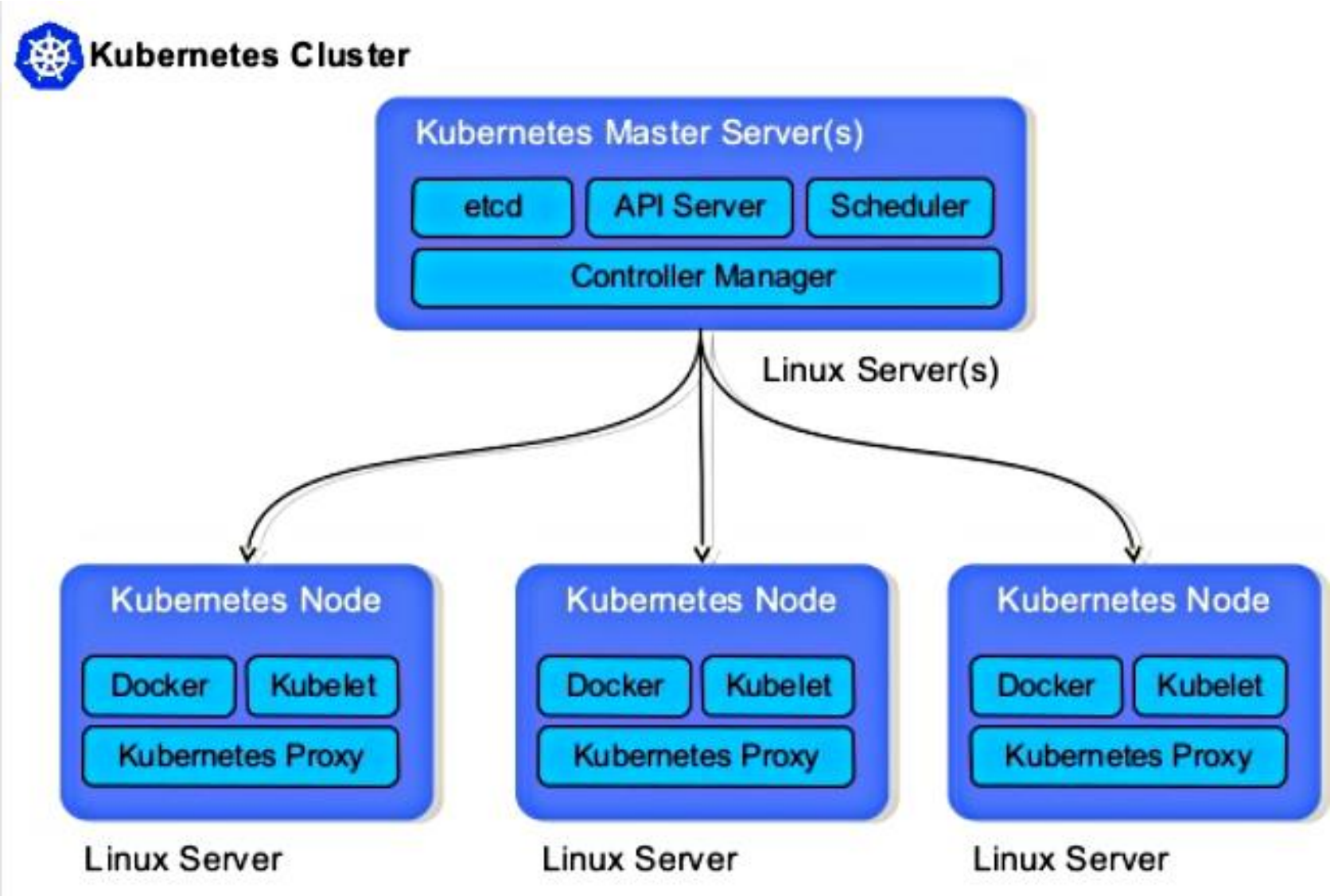
Core components

- etcd
- API server
- Controller-manager
- Scheduler
- Kubelet
- Kube-proxy

Additional required components:

- Docker (cri)
- Network
- DNS

KUBERNETES QUICK START - KUBERNETES OVERVIEW

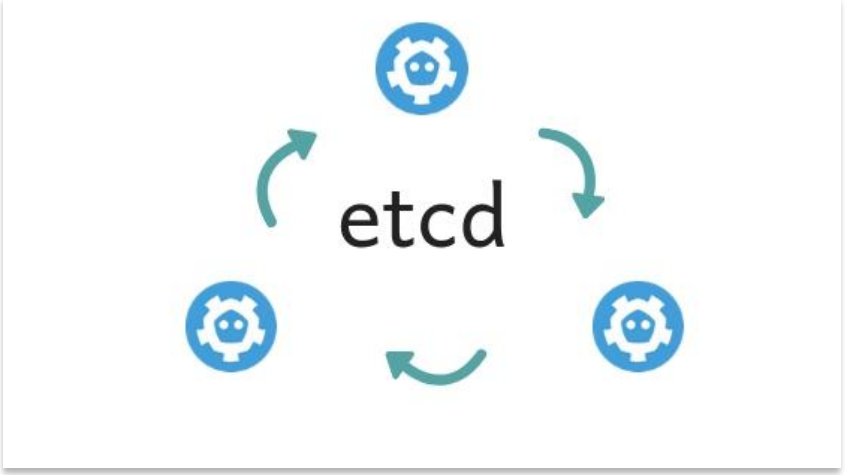


KUBERNETES QUICK START – API server

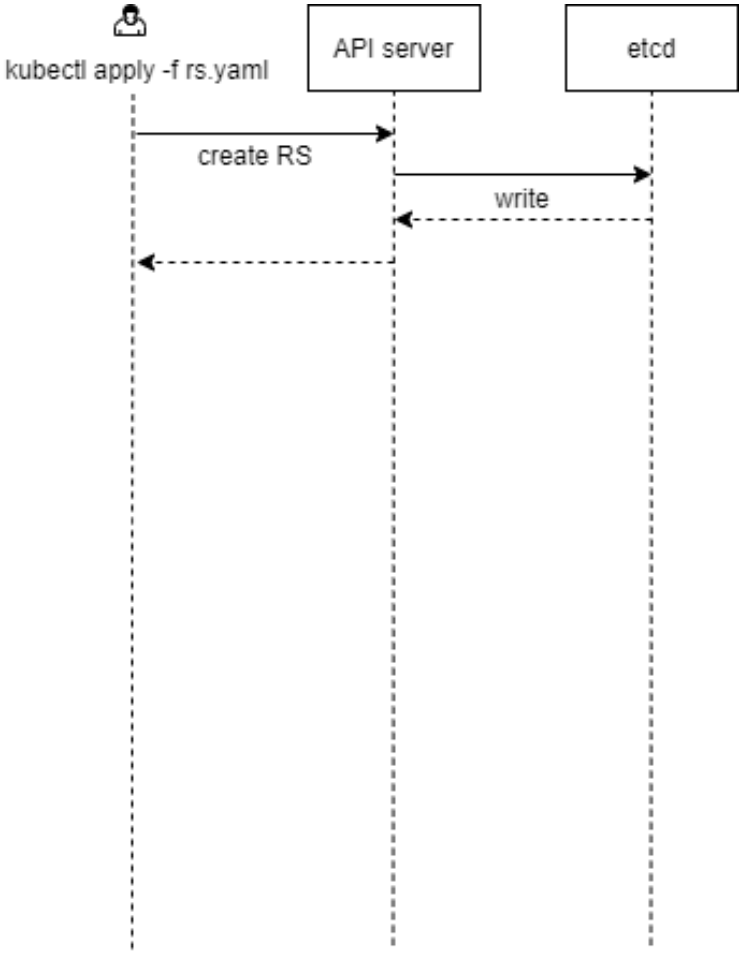
- Central component of k8s cluster
- Use REST API
- Provide Authentication and authorization
- Only API server connects to etcd

KUBERNETES QUICK START – Etcd

- Stores all information about cluster and workload
- etcdctl – is a utility to manage etcd cluster
- Has 2 versions of API v2/v3
- Requires fast disks and low latency network



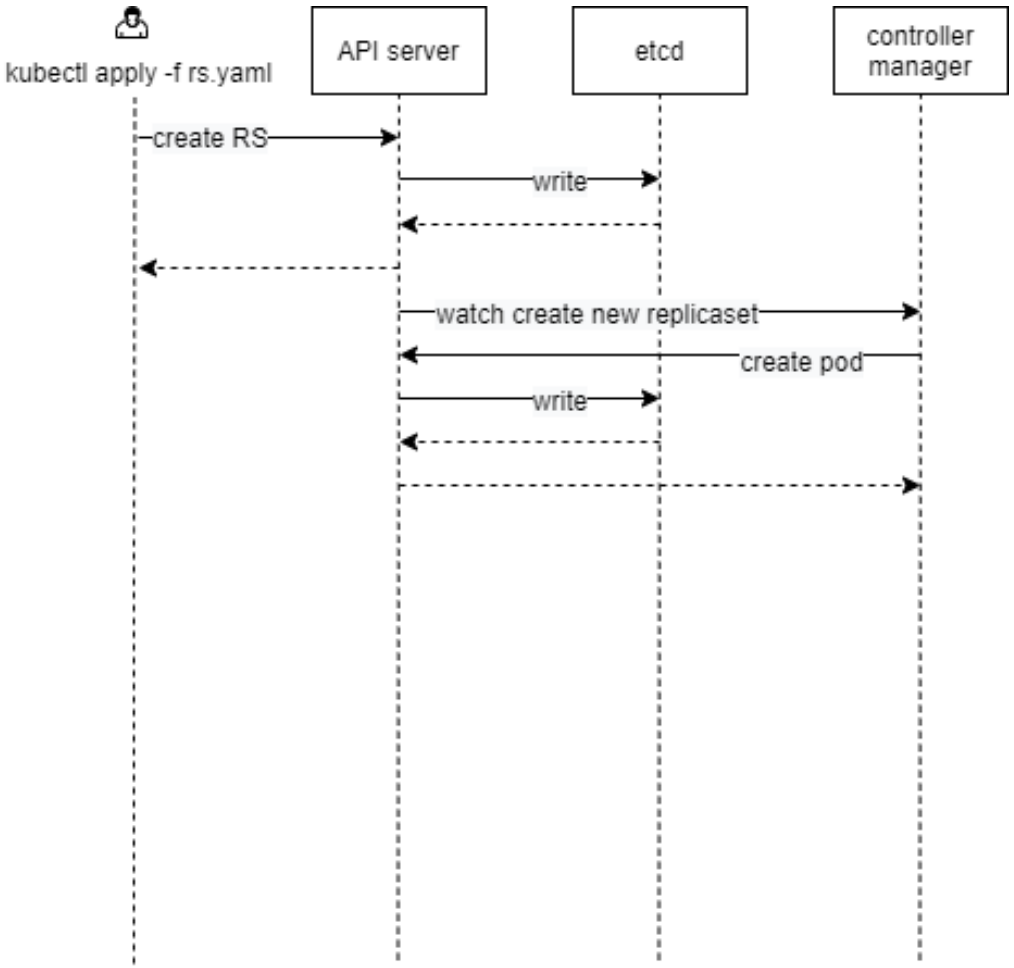
KUBERNETES QUICK START – CREATE RS



KUBERNETES QUICK START – Controller manager

- Multiple controllers
  - Node controller
  - Replication controller
  - Endpoints controller
  - … and many others
- Garbage collector

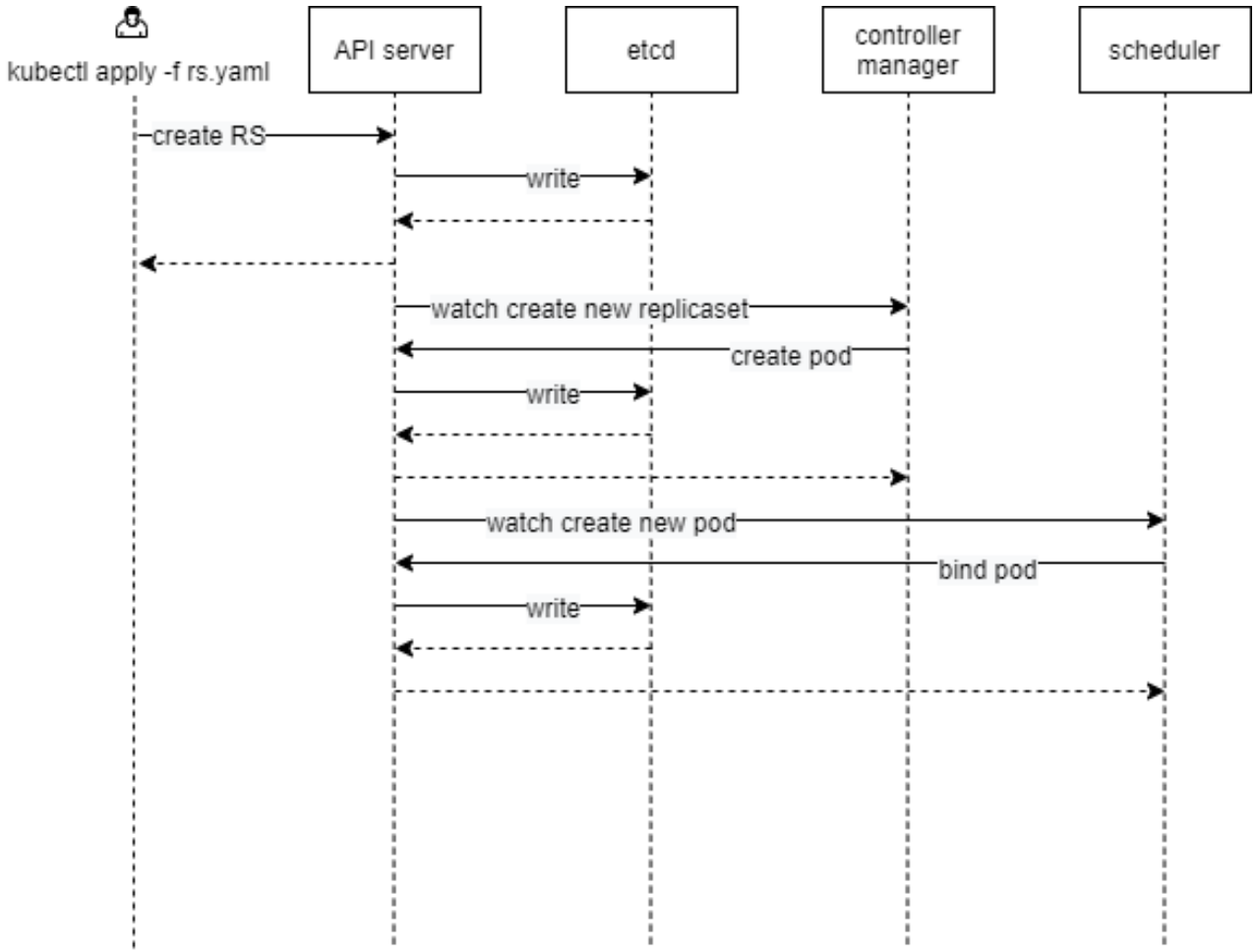
KUBERNETES QUICK START – Controller manager



KUBERNETES QUICK START – Scheduler

- Schedules PODs on Nodes

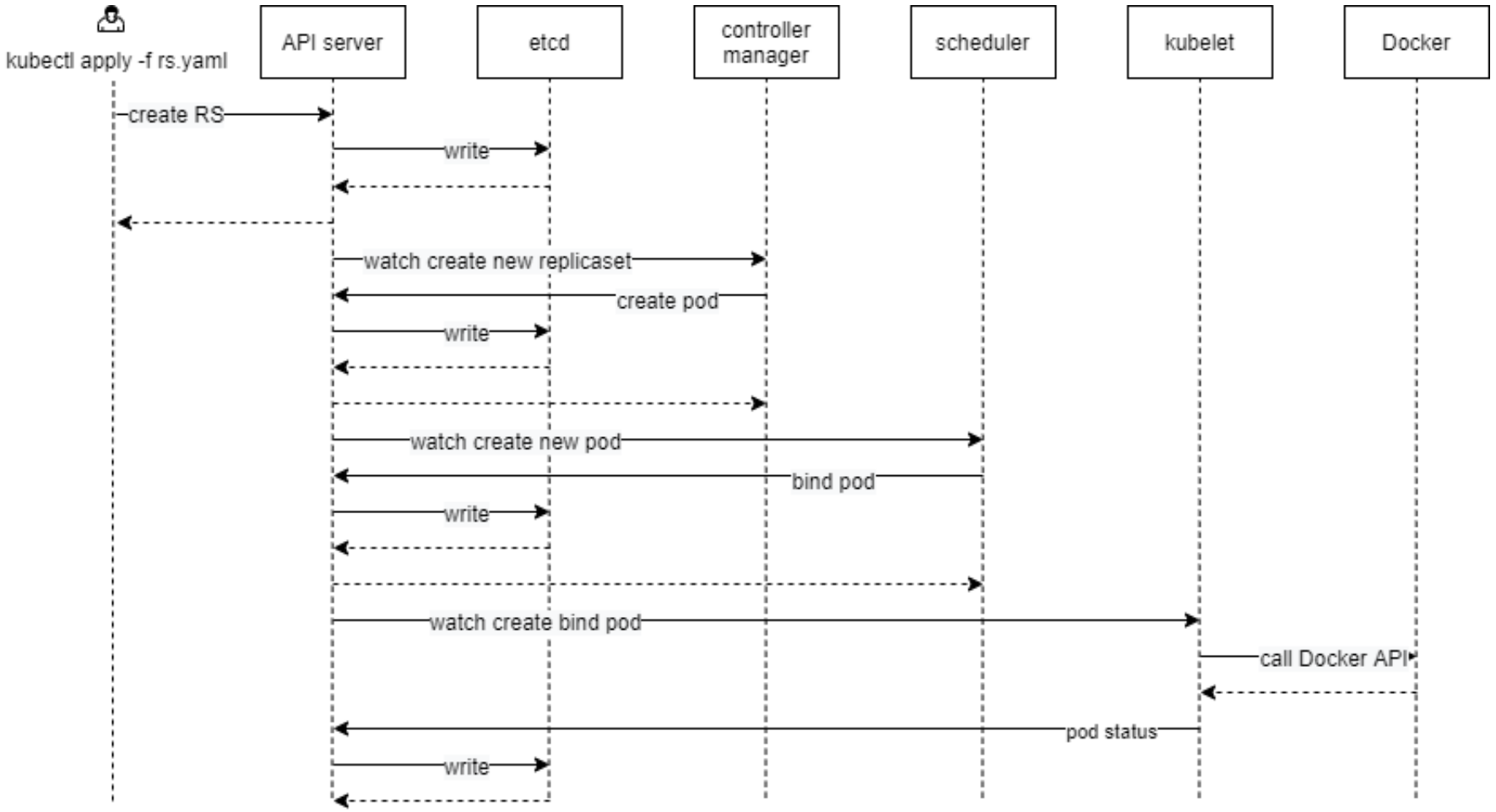
KUBERNETES QUICK START – Scheduler



KUBERNETES QUICK START – Kubelet

- Works on every Node
- Single component that couldn’t be placed inside Docker
- Manages Docker
- Creates PODs

KUBERNETES QUICK START – Kubelet



KUBERNETES QUICK START – Kube-proxy

- Works on every Node
- Manages network rule
- Creates Service (iptables or ipvs)

KUBERNETES QUICK START – Where we can deploy K8S?

On-premise

- On VMs
- On BareMetal
- VMs + BareMetal (for example, masters on VM, workers on BareMetal)

Cloud

- Master managed by cloud
  - EKS - Amazon
  - GKE - Google cloud
  - AKS - Microsoft Azure
- Yandex Cloud
- Mail Cloud Solutions

Cloud VMs

- Amazon EC2-based

KUBERNETES QUICK START – What enough clouds provide

- Ingress controller
- Load Balancer
- Disks
- Auto-scaling
- Pod network

KUBERNETES QUICK START - Tools for deploying K8S

Cloud

- Terraform
- Kops
- Clouds CLI

Self-hosted/Bare metal

- Ansible Kubespray
- Kubeadm

KUBERNETES QUICK START – Terraform

```
resource "google_container_cluster" "primary" {
  name     = "my-gke-cluster"
  location = "us-central1"

  # We can't create a cluster with no node pool defined, but we want to only use
  # separately managed node pools. So we create the smallest possible default
  # node pool and immediately delete it.
  remove_default_node_pool = true
  initial_node_count       = 1

  master_auth {
    username = ""
    password = ""

    client_certificate_config {
      issue_client_certificate = false
    }
  }
}

resource "google_container_node_pool" "primary_preemptible_nodes" {
  name       = "my-node-pool"
  location   = "us-central1"
  cluster    = google_container_cluster.primary.name
  node_count = 1

  node_config {
    preemptible  = true
    machine_type = "e2-medium"

    metadata = {
      disable-legacy-endpoints = "true"
    }

    oauth_scopes = [
      "https://www.googleapis.com/auth/logging.write",
      "https://www.googleapis.com/auth/monitoring",
    ]
  }
}
```

KUBERNETES QUICK START - Kops

Create cluster

```
export KOPS_STATE_STORE=gs://kubernetes-clusters/

PROJECT=`gcloud config get-value project`

export KOPS_FEATURE_FLAGS=AlphaAllowGCE

kops create cluster simple.k8s.local --zones us-central1-a --state ${KOPS_STATE_STORE}/ --project=${PROJECT}
```

Show cluster

```
kops get cluster --state ${KOPS_STATE_STORE}
```

Link

KUBERNETES QUICK START - Kubeadm

Init cluster

kubeadm init <args>

- Interactive mode or flags
- Requirements
- Install process

Adding members

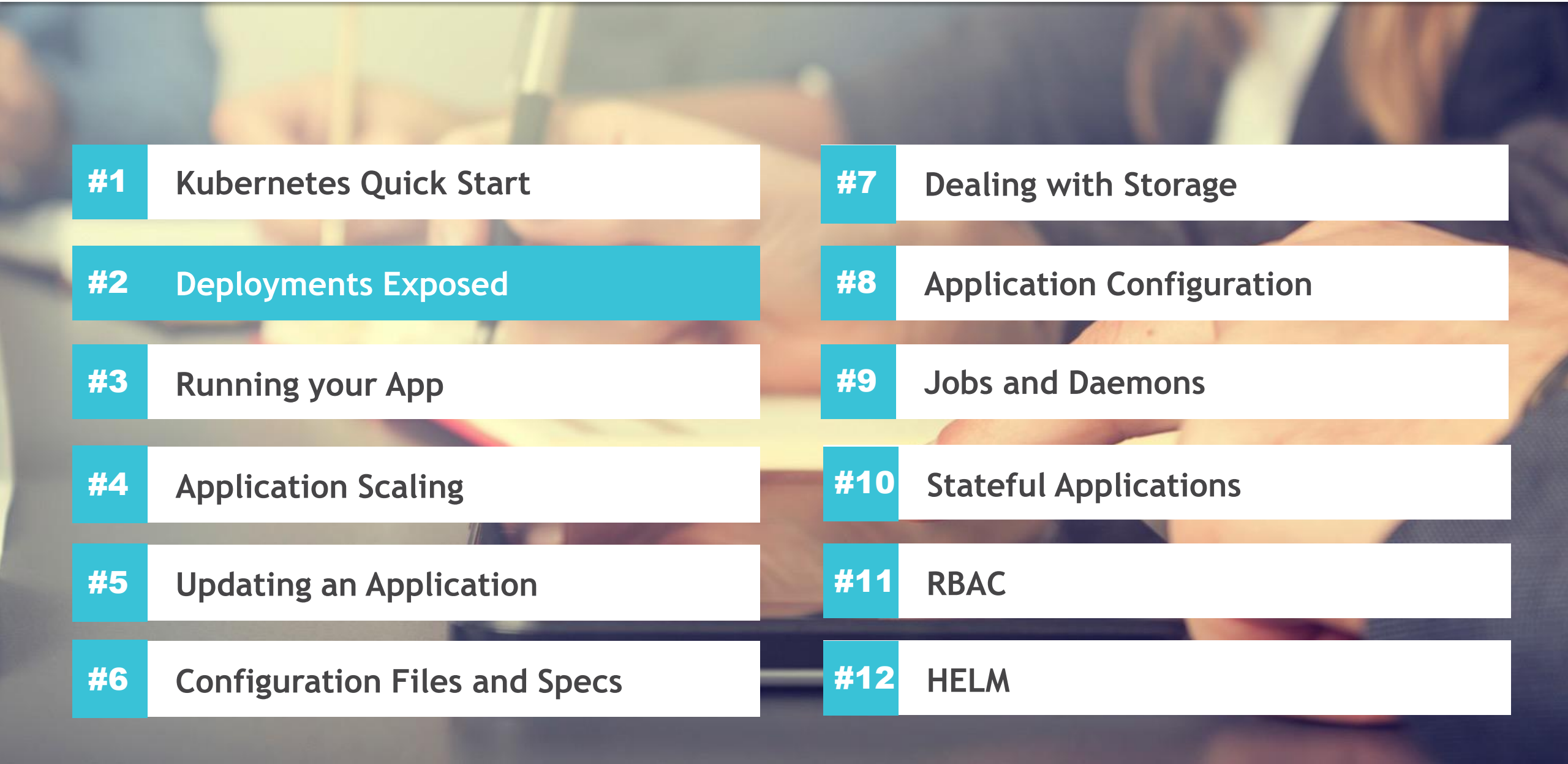
kubeadm join --token <token> <control-plane-host>:<control-plane-port> \ --discovery-token-ca-cert-hash sha256:<hash>

KUBERNETES QUICK START – Ansible Kubespray

```
# Copy ``inventory/sample`` as ``inventory/mycluster``
cp -rfp inventory/sample inventory/mycluster
# Update Ansible inventory file with inventory builder
declare -a IPS=(10.10.1.3 10.10.1.4 10.10.1.5)
CONFIG_FILE=inventory/mycluster/hosts.yaml python3 contrib/inventory_builder/inventory.py ${IPS[@]}
# Review and change parameters under ``inventory/mycluster/group_vars``
cat inventory/mycluster/group_vars/all/all.yml
cat inventory/mycluster/group_vars/k8s-cluster/k8s-cluster.yml
# Deploy Kubespray with Ansible Playbook - run the playbook as root
ansible-playbook -i inventory/mycluster/hosts.yaml --become --become-user=root cluster.yml
```

Link

KUBERNETES QUICK START



#1	Kubernetes Quick Start
#2	Deployments Exposed
#3	Running your App
#4	Application Scaling
#5	Updating an Application
#6	Configuration Files and Specs
#7	Dealing with Storage
#8	Application Configuration
#9	Jobs and Daemons
#10	Stateful Applications
#11	RBAC
#12	HELM

DEPLOYMENTS EXPOSED - NAMESPACES

1. **Namespaces**
2. Pods and Containers
3. Deployment
4. Services
5. Ingress
6. Selector, Labels and Ports
6. Network plugins

DEPLOYMENTS EXPOSED - NAMESPACES

NAMESPACES

- virtual clusters spaces provided by the same physical cluster
- intended for use in environments with many users
- uniqueness of resources
- namespaces resource quota.

DEPLOYMENTS EXPOSED - PODS AND CONTAINERS

1. Namespaces
2. **Pods and Containers**
3. Deployment
4. Services
5. Ingress
6. Selector, Labels and Ports
6. Network plugins

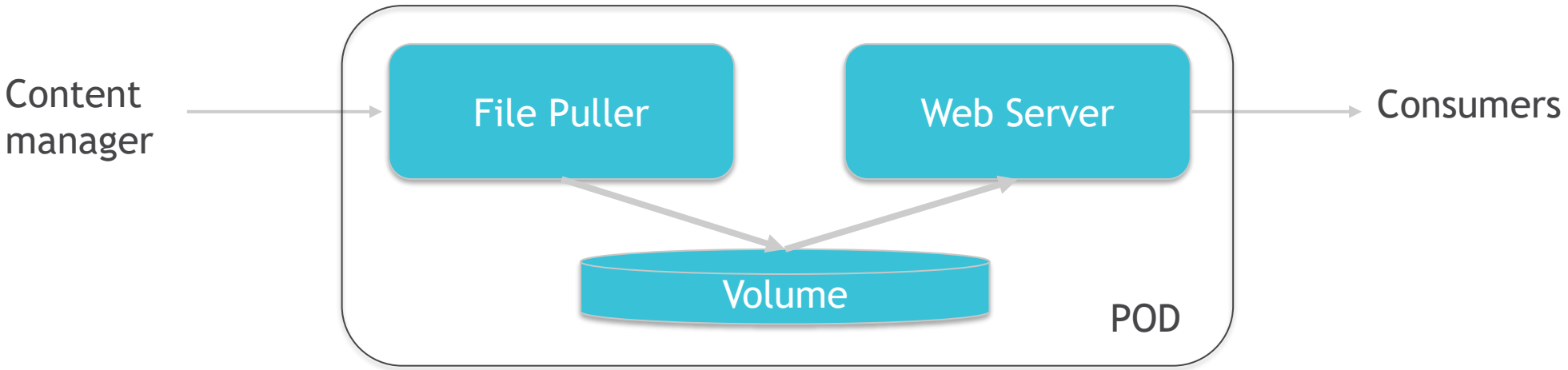
DEPLOYMENTS EXPOSED - PODS AND CONTAINERS

POD

- Is the basic building block of Kubernetes–the smallest
- Abstraction container or multiple containers
- Has a unique network IP address in the cluster

DEPLOYMENTS EXPOSED - PODS AND CONTAINERS

MULTIPLY CONTAINERS



DEPLOYMENTS EXPOSED - PODS AND CONTAINERS

INIT CONTAINERS

Are specialized Containers that run before app Containers

ADVANTAGES:

- They can contain and run utilities that are not desirable to include in the app Container image for security reasons.
- They run to completion before any app Containers start, whereas app Containers run in parallel.
- Init Containers provide an easy way to block or delay the startup of app Containers until some set of preconditions are met.

DEPLOYMENTS EXPOSED - DEPLOYMENT

1. Namespaces
3. Pods and Containers
2. **Deployment**
4. Services
5. Ingress
6. Selector, Labels and Ports
6. Network plugins

DEPLOYMENTS EXPOSED - DEPLOYMENT

DEPLOYMENT

- provides declarative updates for Pods and ReplicaSets.
- change the actual state to the desired state at a controlled rate for you.
- You should not manage ReplicaSets owned by a Deployment

DEPLOYMENTS EXPOSED - DEPLOYMENT

A typical use case is:

- Create a Deployment to rollout a ReplicaSet
- Declare the new state of the Pods
- Rollback to an earlier Deployment revision
- Scale up the Deployment to facilitate more load
- Pause the Deployment to apply multiple fixes
- Use the status of the Deployment

DEPLOYMENTS EXPOSED - DEPLOYMENT

Creating a Deployment

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.12
        ports:
        - containerPort: 80
```

Kubectl:

- kubectl apply -f deployment/nginx_deployment.yaml --record
- kubectl get deploy
- kubectl rollout status deployment nginx
- kubectl get rs
- kubectl get pods -o wide
- kubectl describe deploy/nginx

NOTE:

Setting the kubectl flag --record to true allows you to record current command in the annotations of the resources being created or updated. It will be useful for future introspection; for example, to see the commands executed in each Deployment revision.

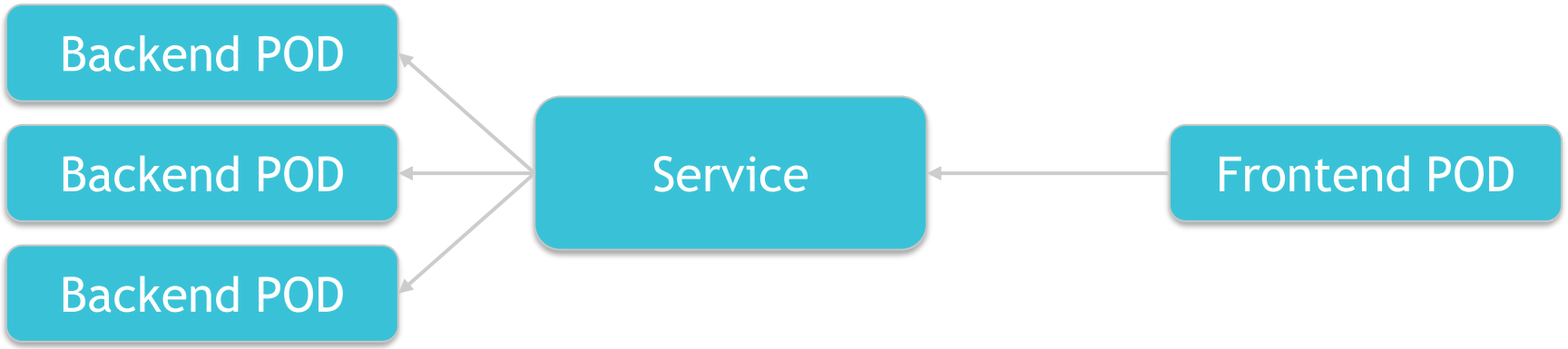
DEPLOYMENTS EXPOSED - SERVICES

1. Namespaces
2. Pods and Containers
3. Deployment
4. **Services**
5. Ingress
6. Selector, Labels and Ports
6. Network plugins

DEPLOYMENTS EXPOSED - SERVICES

SERVICES

- is an abstraction which defines a logical set of Pods and a policy by which to access them
- The set of Pods targeted by a Service is determined by a Label Selector.
- “Normal” (not headless) Services resolves to the cluster IP of the Service
- “Headless” (without a cluster IP) Services resolves to the set of IPs of the pods selected by the Service



DEPLOYMENTS EXPOSED - SERVICES

Creating a Service

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

NOTE:
Kubernetes Services support TCP and UDP for protocols. The default is TCP

QA