

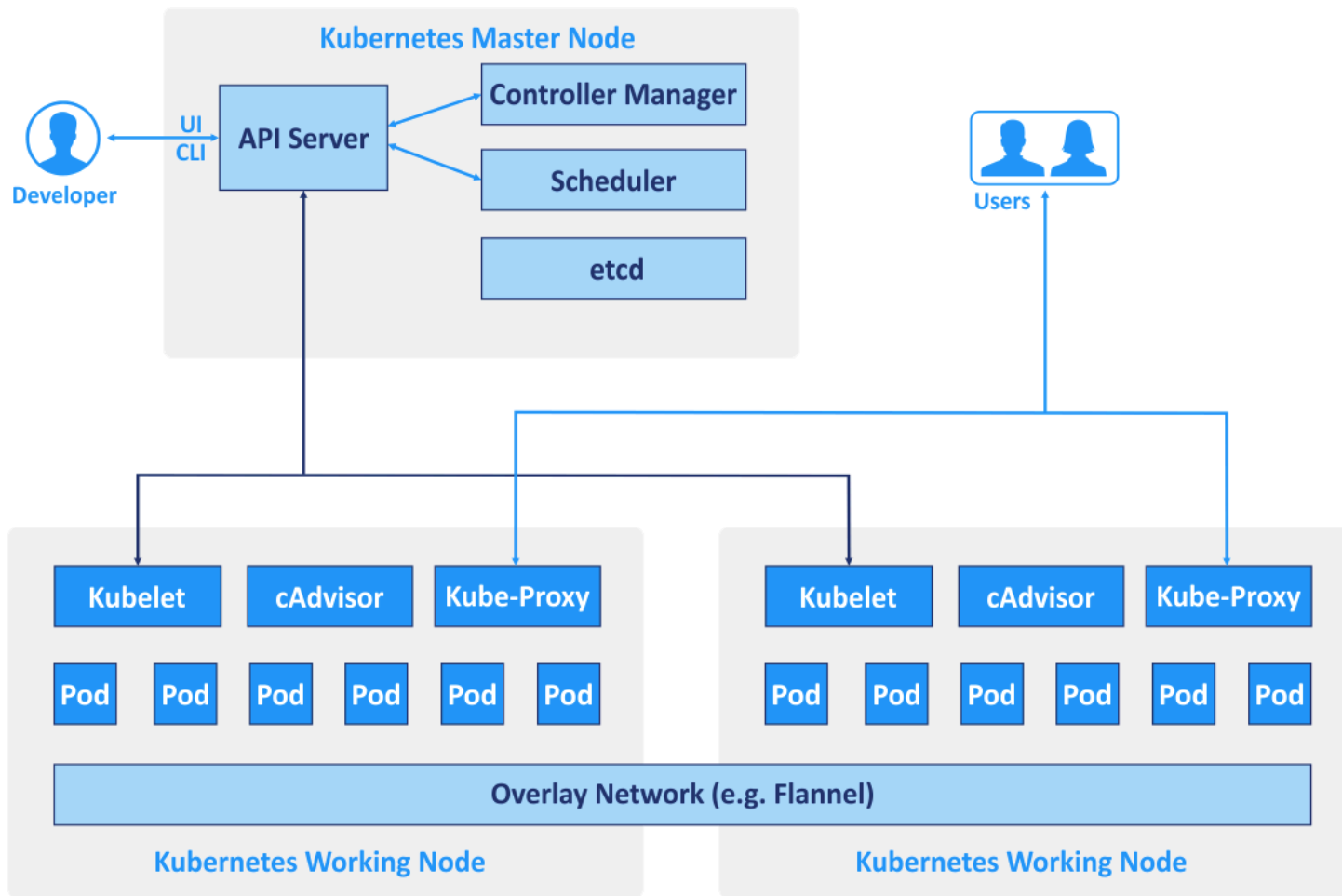
# Kubernetes

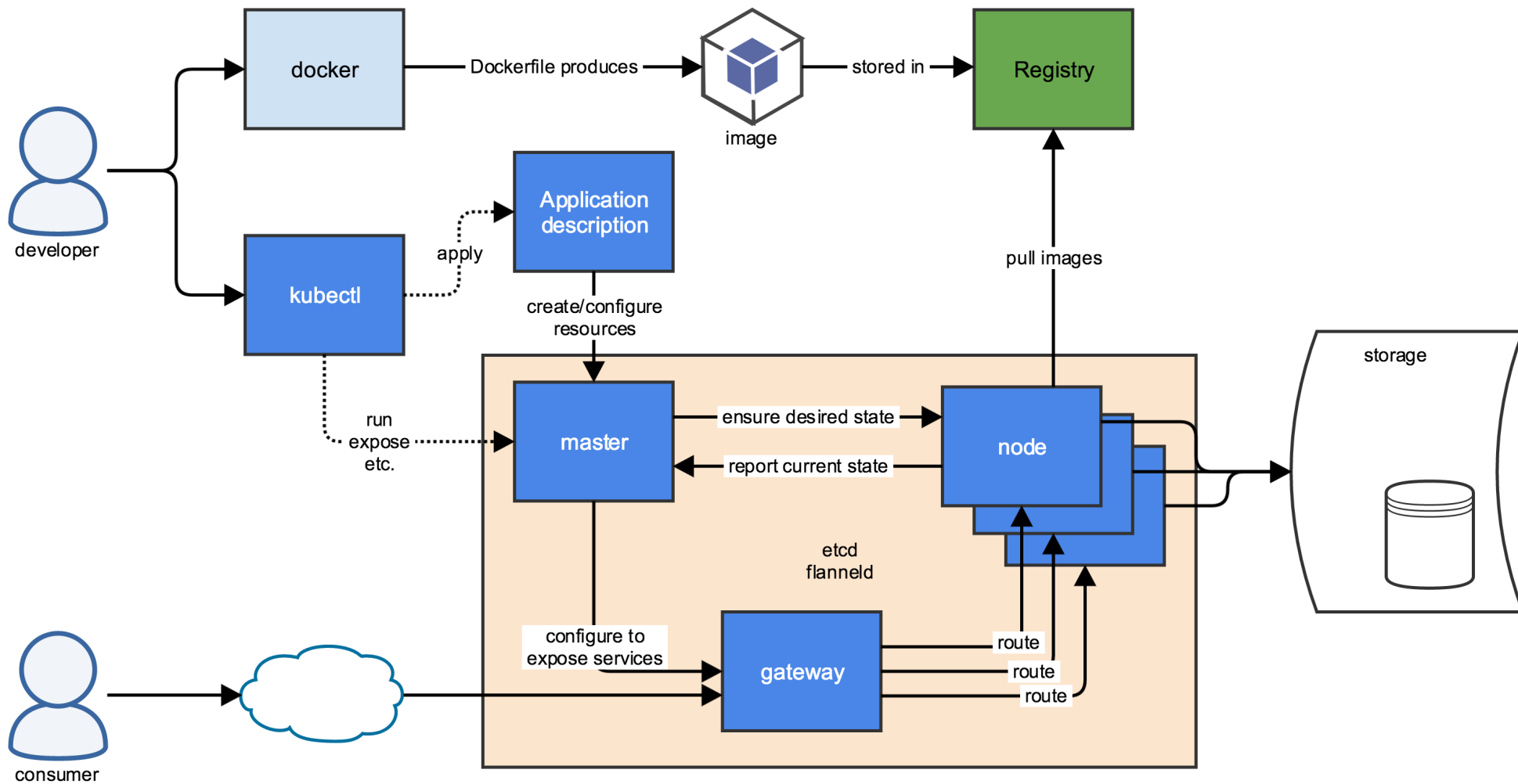


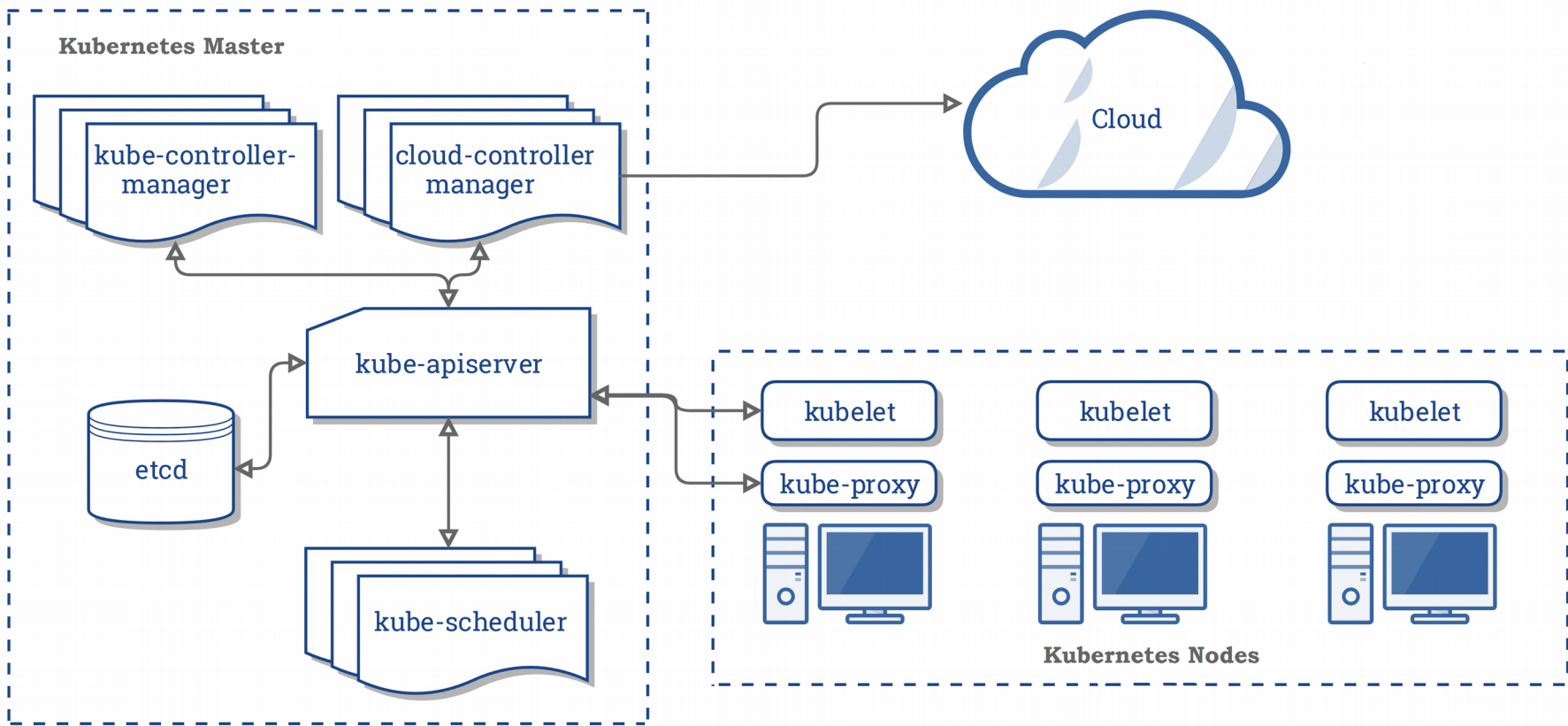
# Cluster

- Conjunto de equipos gestionado por Kubernetes.
- Nodo Master.
  - Plano de control.
  - Gestiona los nodos worker.
- Nodo Worker.
  - Componentes de la aplicación.
- Ejecuta contenedores (Docker, CRI-O, rktkit..).

- Los principales proveedores de cloud computing ofrecen este servicio mediante un “cloud virtual”
- Ya preparado para kubernetes.
-







# Nodos

- Equipo para ejecutar pods.
- Kubelet es el agente que está en cada nodo.
- Kube-proxy es un proxy de red en cada nodo.
- Container runtime: docker, containerd, cri-o, rktlet
  - kubectl get node
  - kubectl describe node <NODO>
- Node Status
  - Direcciones
  - Condiciones
  - Capacity y Allocatable
  - Info

```
Conditions:
  Type           Status  LastHeartbeatTime      LastTransitionTime      Reason                    Message
  ----           -
MemoryPressure   False   Sun, 01 Dec 2019 16:42:00 +0100   Sat, 30 Nov 2019 17:49:07 +0100   KubeletHasSufficientMemory   kubelet has sufficient memory available
DiskPressure     False   Sun, 01 Dec 2019 16:42:00 +0100   Sat, 30 Nov 2019 17:49:07 +0100   KubeletHasNoDiskPressure     kubelet has no disk pressure
PIDPressure      False   Sun, 01 Dec 2019 16:42:00 +0100   Sat, 30 Nov 2019 17:49:07 +0100   KubeletHasSufficientPID      kubelet has sufficient PID available
Ready           True    Sun, 01 Dec 2019 16:42:00 +0100   Sat, 30 Nov 2019 17:49:07 +0100   KubeletReady                 kubelet is posting ready status

Addresses:
  InternalIP: 10.0.2.15
  Hostname:   minikube

Capacity:
  cpu:                2
  ephemeral-storage: 17784772Ki
  hugepages-2Mi:      0
  memory:             2038624Ki
  pods:              110

Allocatable:
  cpu:                2
  ephemeral-storage: 16390445849
  hugepages-2Mi:      0
  memory:             1936224Ki
  pods:              110

System Info:
Machine ID:          114524b0cd904ef4ac7d4b3aa2d5fbfb
System UUID:         48D75059-2C87-4E0E-B87C-3867349CCD6A
Boot ID:             c3bf6a31-e954-4ecf-8e2e-b1a50327af39
Kernel Version:      4.15.0
OS Image:            Buildroot 2018.05
Operating System:    linux
Architecture:        amd64
Container Runtime Version: docker://18.6.2
Kubelet Version:     v1.14.0
Kube-Proxy Version:  v1.14.0
Non-terminated Pods: (24 in total)
```



# Controladores

- Node Controller
  - Responsable del nodo.
- Replication Controller
  - Mantiene el número concreto de pods con sus réplicas.
- Endpoint Controller
  - Rellena de contenido los objetos “endpoint” (servicios, pod)
- Service Account y Token Controller
  - Gestiona las cuentas y los tokens de acceso al API para los namespaces.

# Entidades de kubernetes

# Namespaces

- Nombres para establecer particiones del cluster virtual en el cluster real.
- Se emplea para agrupar y separar equipos independientes.

# Labels

- Par clave/valor asociado a objetos definidas por el usuario, sin semántica.
- Se usan para cualificar, organizar y seleccionar objetos.
- Se aplican al crear el objeto o dinámicamente.

# Selector de campo

- Para seleccionar objetos/recursos en funcion de valores de campos.
- metadata.name
- metadata.namespace
- status.phase
  - `kubectl get pods --field-selector status.phase=Running`

# Roles

- Role
  - Se aplica a nivel de namespace
- ClusterRole
  - Se aplica a nivel de cluster
    - cluster-admin
    - admin
    - edit
    - view
- RoleBinding
- ClusterRoleBinding

# Pods

- Unidad básica de ejecución.
- Conjunto de uno o más contenedores constituyendo una instancia única de una aplicación.
- Dirección IP única a nivel de pod.
  - Los contenedores comparten la dirección IP
  - Coordinación en el uso de puertos.
- Almacenamiento
  - Volúmenes compartidos entre los contenedores.

# Pod con varios contenedores

- Todos los contenedores se ubican automáticamente en el mismo nodo.
  - Pueden compartir recursos.
  - Un pod tiene una única IP.
    - Los contenedores puede comunicarse con “localhost”.
  - Los puertos de los contenedores son únicos en el pod.
    - Los contenedores tienen que coordinarse en el uso de puertos.
  - Pueden compartir Volúmenes.




# Pods y Controladores

- Un controlador puede crear y gestionar varios pods.
  - Replicación, Rollout,
- Pod Template
  - Especificación de un pod que se usa en otros objetos
- Controladores de replicación, Jobs y DaemonSets

# Init Containers

- Se ejecutan antes que los contenedores “regulares” del pod.
- Ejecutan hasta acabar.
- Cuando acaba uno, se ejecuta el siguiente.
  - Si falla la ejecución, se rearranca el pod hasta que termina bien.

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox:1.28
    command: ['sh', '-c', 'echo The app is running! && sleep 3600']
  initContainers:
  - name: init-myservice
    image: busybox:1.28
    command: ['sh', '-c', 'until nslookup myservice; do echo waiting for myservice; sleep 2; done;']
  - name: init-mydb
    image: busybox:1.28
    command: ['sh', '-c', 'until nslookup mydb; do echo waiting for mydb; sleep 2; done;']
```




# Topología de pods

- Se utilizan las etiquetas en los nodos del cluster.

NAME	STATUS	ROLES	AGE	VERSION	LABELS
node1	Ready	<none>	4m26s	v1.16.0	node=node1, zone=zoneA
node2	Ready	<none>	3m58s	v1.16.0	node=node2, zone=zoneA
node3	Ready	<none>	3m17s	v1.16.0	node=node3, zone=zoneB
node4	Ready	<none>	2m43s	v1.16.0	node=node4, zone=zoneB

```
kind: Pod
apiVersion: v1
metadata:
  name: mypod
  labels:
    foo: bar
spec:
  topologySpreadConstraints:
    - maxSkew: 1
      topologyKey: zone
      whenUnsatisfiable: DoNotSchedule
      labelSelector:
        matchLabels:
          foo: bar
  containers:
    - name: pause
      image: k8s.gcr.io/pause:3.1
```



# Controllers - ReplicaSet

- Mantener un conjunto estable de réplicas de un pod.

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  # modify replicas according to your case
  replicas: 3
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
        - name: php-redis
          image: gcr.io/google_samples/gb-frontend:v3
```

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: frontend-scaler
spec:
  scaleTargetRef:
    kind: ReplicaSet
    name: frontend
  minReplicas: 3
  maxReplicas: 10
  targetCPUUtilizationPercentage: 50
```

# Controllers - Deployments

- Conjunto de pods.
- Arrancar.
- Actualizar.
- Revertir actualización.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
            - containerPort: 80
```



## Controllers – StatefulSets

- Gestión y escalado de un conjunto de pods.
- Garantiza el orden y la “exclusividad” de estos pods.

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  ports:
    - port: 80
      name: web
  clusterIP: None
  selector:
    app: nginx
```

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: nginx # has to match .spec.template.metadata.labels
  serviceName: "nginx"
  replicas: 3 # by default is 1
  template:
    metadata:
      labels:
        app: nginx # has to match .spec.selector.matchLabels
    spec:
      terminationGracePeriodSeconds: 10
      containers:
        - name: nginx
          image: k8s.gcr.io/nginx-slim:0.8
          ports:
            - containerPort: 80
              name: web
          volumeMounts:
            - name: www
              mountPath: /usr/share/nginx/html
      volumeClaimTemplates:
        - metadata:
            name: www
          spec:
            accessModes: [ "ReadWriteOnce" ]
            storageClassName: "my-storage-class"
            resources:
              requests:
                storage: 1Gi
```

# Controllers – ReplicationController

- Obsoleto.
- Deployment + ReplicaSet

# Controllers - DaemonSet

- Asegurar que se ejecuta una copia de un pod en los nodos (todos o parte).
- Cluster storage daemon
  - Glusterd, ceph
- Logs collection daemon
  - Fluentd, logstash
- Monitoring daemon
  - Prometheus, flowmill, collectd, Dynatrace OneAgent, appDynamics agent, Datadog agent, Instana


```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd-elasticsearch
  namespace: kube-system
  labels:
    k8s-app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
    spec:
      tolerations:
        - key: node-role.kubernetes.io/master
          effect: NoSchedule
      containers:
        - name: fluentd-elasticsearch
          image: quay.io/fluentd_elasticsearch/fluentd:v2.5.2
          resources:
            limits:
              memory: 200Mi
            requests:
              cpu: 100m
              memory: 200Mi
          volumeMounts:
            - name: varlog
              mountPath: /var/log
            - name: varlibdockercontainers
              mountPath: /var/lib/docker/containers
              readOnly: true
      terminationGracePeriodSeconds: 30
      volumes:
        - name: varlog
          hostPath:
            path: /var/log
        - name: varlibdockercontainers
          hostPath:
            path: /var/lib/docker/containers
```



# Controllers - Jobs

- Para ejecutar un conjunto de pods asegurando que un número definido de pods terminan bien.
  - Fallo de un nodo.
- Paralelismo.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi-with-timeout
spec:
  backoffLimit: 5
  activeDeadlineSeconds: 100
  template:
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]
        restartPolicy: Never
```



# Controllers - CronJob

- Para crear jobs planificados en el tiempo.

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: hello
              image: busybox
              args:
                - /bin/sh
                - -c
                - date; echo Hello from the Kubernetes cluster
          restartPolicy: OnFailure
```

# Endpoint Slices (alpha)

- Referencias a un conjunto de endpoints de red.
- Agrupar servicios cuando hay un número muy elevado.

```
apiVersion: discovery.k8s.io/v1alpha1
kind: EndpointSlice
metadata:
  name: example-abc
  labels:
    kubernetes.io/service-name: example
addressType: IP
ports:
  - name: http
    protocol: TCP
    port: 80
endpoints:
  - addresses:
    - "10.1.2.3"
    - "2001:db8::1234:5678"
    conditions:
      ready: true
    hostname: pod-1
    topology:
      kubernetes.io/hostname: node-1
      topology.kubernetes.io/zone: us-west2-a
```



# Servicios

- Exponer una aplicación ejecutando en unos pods como un servicio de red.
  - Ocultan los pods.
- Tiene una IP que sirve para acceder a los pods.

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```


- Sin selector de pod hay que añadir el endpoint manualmente

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

```
apiVersion: v1
kind: Endpoints
metadata:
  name: my-service
subsets:
  - addresses:
    - ip: 192.0.2.42
    ports:
    - port: 9376
```

- Multipuerto

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 9376
    - name: https
      protocol: TCP
      port: 443
      targetPort: 9377
```




- Variables de entorno automáticas

```
REDIS_MASTER_SERVICE_HOST=10.0.0.11  
REDIS_MASTER_SERVICE_PORT=6379  
REDIS_MASTER_PORT=tcp://10.0.0.11:6379  
REDIS_MASTER_PORT_6379_TCP=tcp://10.0.0.11:6379  
REDIS_MASTER_PORT_6379_TCP_PROTO=tcp  
REDIS_MASTER_PORT_6379_TCP_PORT=6379  
REDIS_MASTER_PORT_6379_TCP_ADDR=10.0.0.11
```

- ClusterIP
  - Accesible únicamente desde dentro del cluster.
- NodePort
  - El servicio se expone en cada nodo con la IP del nodo y el puerto.
- LoadBalancer
  - El servicio se expone utilizando un loadbalancer, creando rutas entre la direccion expuesta y las direcciones internas.
- ExternalName
  - Asocia el servicio al contenido del campo externalName por DNS.

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
  clusterIP: 10.0.171.239
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
      - ip: 192.0.2.127
```



```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 9376
  externalIPs:
    - 80.11.12.10
```

# Ingress

- Ingress expone rutas http y https desde fuera del cluster a servicios dentro del cluster.
- Las rutas se controlan mediante reglas.
- Si no hay reglas, el tráfico se enruta al backend por defecto.
  - Definido en el controlador Ingress.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        backend:
          serviceName: test
          servicePort: 80
```



- Servicio único

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: test-ingress
spec:
  backend:
    serviceName: testsvc
    servicePort: 80
```



- Más de un servicio

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: simple-fanout-example
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: service1
          servicePort: 4200
      - path: /bar
        backend:
          serviceName: service2
          servicePort: 8080
```





- Enrutar por “Host header”


```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: name-virtual-host-ingress
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - backend:
          serviceName: service1
          servicePort: 80
  - host: bar.foo.com
    http:
      paths:
      - backend:
          serviceName: service2
          servicePort: 80
```

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: name-virtual-host-ingress
spec:
  rules:
  - host: first.bar.com
    http:
      paths:
      - backend:
          serviceName: service1
          servicePort: 80
  - host: second.foo.com
    http:
      paths:
      - backend:
          serviceName: service2
          servicePort: 80
  - http:
      paths:
      - backend:
          serviceName: service3
          servicePort: 80
```

# Network Policies


- Especificación de como grupos de pods tienen permiso para comunicarse entre si con otros endpoints de la red
  - Reglas mediante “labels”.
- Ingress para tráfico entrante.
- Egress para tráfico saliente.

```
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
```




# Permitir, Denegar Ingress

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all
spec:
  podSelector: {}
  ingress:
    - {}
  policyTypes:
    - Ingress
```




```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
  podSelector: {}
  policyTypes:
    - Ingress
```



# Permitir, Denegar Egress

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all
spec:
  podSelector: {}
  egress:
  - {}
  policyTypes:
  - Egress
```

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  - Egress
```



# VOLÚMENES

<https://kubernetes.io/docs/concepts/storage/volumes/>

- awsElasticBlockStore
- azureDisk
- azureFile
- cephfs
- cinder
- configMap
- csi
- downwardAPI
- emptyDir
- fc (fibre channel)
- flexVolume
- flocker
- gcePersistentDisk
- gitRepo (deprecated)
- glusterfs
- hostPath
- iscsi
- local
- nfs
- persistentVolumeClaim
- projected
- portworxVolume
- quobyte
- rbd
- scaleIO
- secret
- storageos
- vsphereVolume

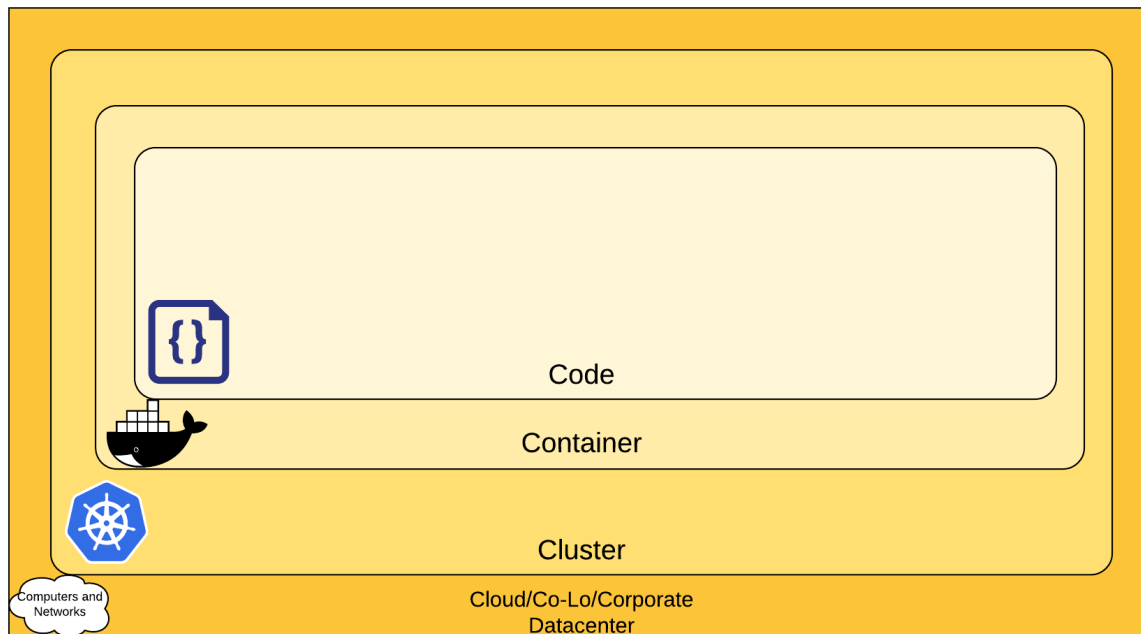
# PersistentVolume

- Separación entre provisión y uso.
- PersistentVolume
  - Almacenamiento provisionado por un administrador, o dinámicamente mediante la API.
    - Recurso del cluster.
- PersistentVolumeClaim.
  - Petición de almacenamiento de un usuario.
  - Un Pod consume recursos de un nodo.
  - Un PVC consume recursos PV.

# Ciclo de vida de un volumen

- Provisión
  - Estática o dinámica. Se genera un PV.
- Ajuste (binding).
  - Entre una petición PVC y un PV.
- Uso
  - Como un volumen en un pod.
- Liberación.
  - Cuando se elimina la PVC, el PV sigue existiendo, “desasociado”, pero con datos.
  - Hay que liberarlo manualmente.
- Borrado.
- Reciclado.
  - Eliminación de los datos y libre para nuevo uso.

# Seguridad





## Cloud Provider Security Table

IaaS Provider	Link
Alibaba Cloud	<a href="https://www.alibabacloud.com/trust-center">https://www.alibabacloud.com/trust-center</a>
Amazon Web Services	<a href="https://aws.amazon.com/security/">https://aws.amazon.com/security/</a>
Google Cloud Platform	<a href="https://cloud.google.com/security/">https://cloud.google.com/security/</a>
IBM Cloud	<a href="https://www.ibm.com/cloud/security">https://www.ibm.com/cloud/security</a>
Microsoft Azure	<a href="https://docs.microsoft.com/en-us/azure/security/azure-security">https://docs.microsoft.com/en-us/azure/security/azure-security</a>
VMWare VSphere	<a href="https://www.vmware.com/security/hardening-guides.html">https://www.vmware.com/security/hardening-guides.html</a>