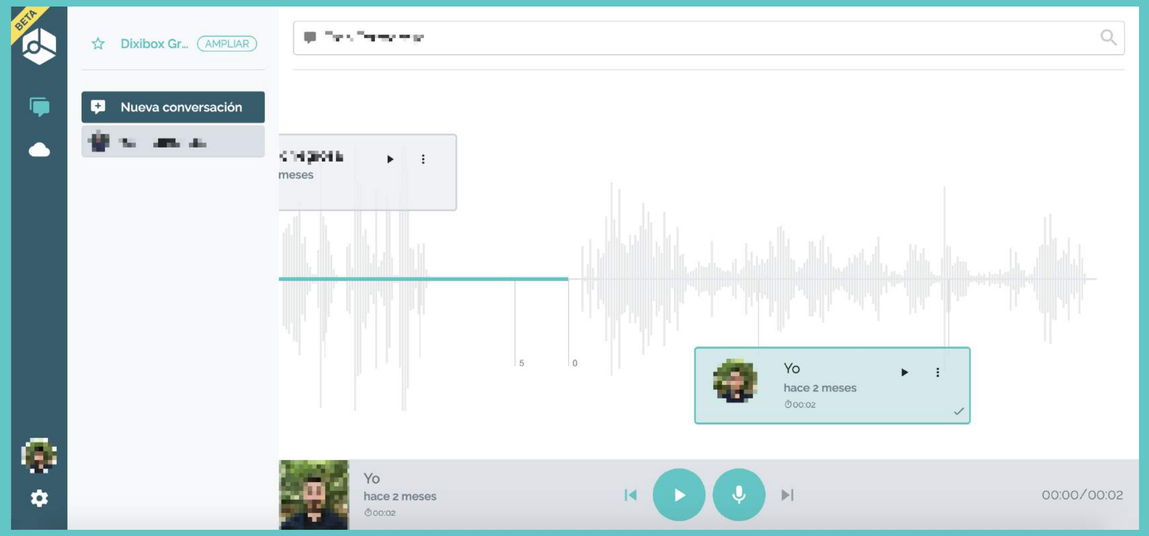
Introducción a Kubernetes

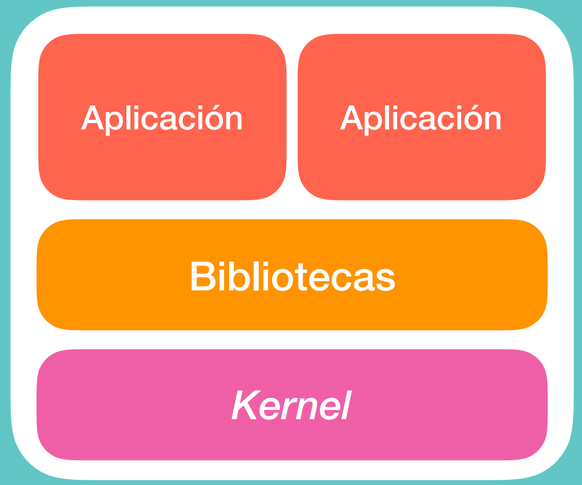
Sobre nosotros

- **Buzón de voz** en la nube.
- Permite **crear**, **almacenar**, **enviar** y **recibir** notas de voz y archivos de **audio**.
- Es posible **enviar** notas de voz **aunque la persona receptora no tenga cuenta**.
- Ofrece **integraciones** con otros servicios, como **Twitter** o **Zendesk**.

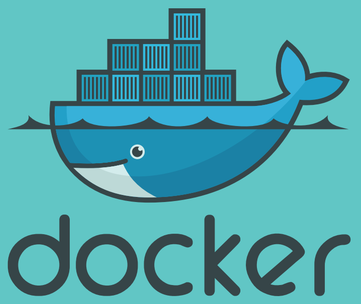
dixibox

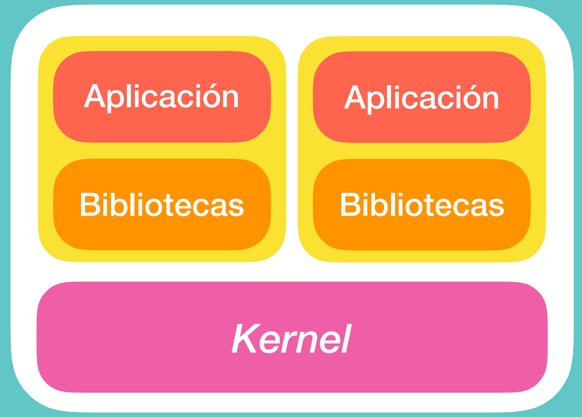


Docker



- Software de **virtualización**.
- Permite **empaquetar**, **distribuir** y **ejecutar** software con todas sus dependencias en **contenedores**.
- Utiliza las características de aislamiento de recursos del *kernel* de **Linux**.
- La contenedorización es **más ligera** que la virtualización.



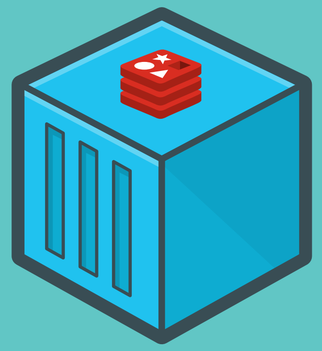
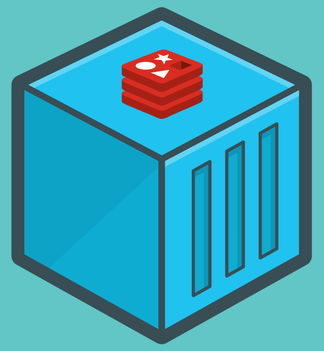


Docker



Imágenes



Contenedores

Kubernetes

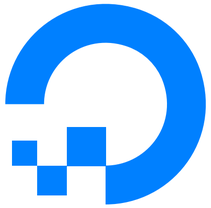
- Software de **orquestación** de **contenedores**.
- Del griego **κυβερνήτης** (/ky.ber.nɛ̌ː.tɛːs/): capitán o timonel.
- Desarrollado por **Google** y donado a la Cloud Native Computing Foundation.
- Permite **desplegar**, **escalar**, **coordinar** y **gestionar** contenedores de software.
- Ofrece **resiliencia**, **escalabilidad**, **tenencia múltiple** y **reconciliación**.
- Tiene un **bajo acoplamiento**.

kubernetes

Kubernetes como servicio

Google Cloud

Microsoft Azure



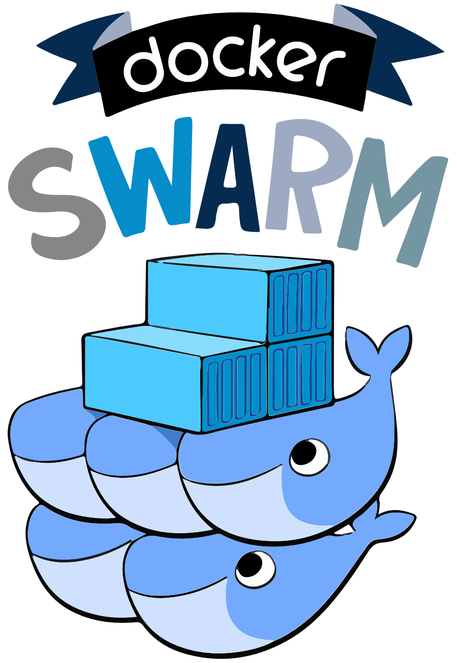
DigitalOcean

OVH.com

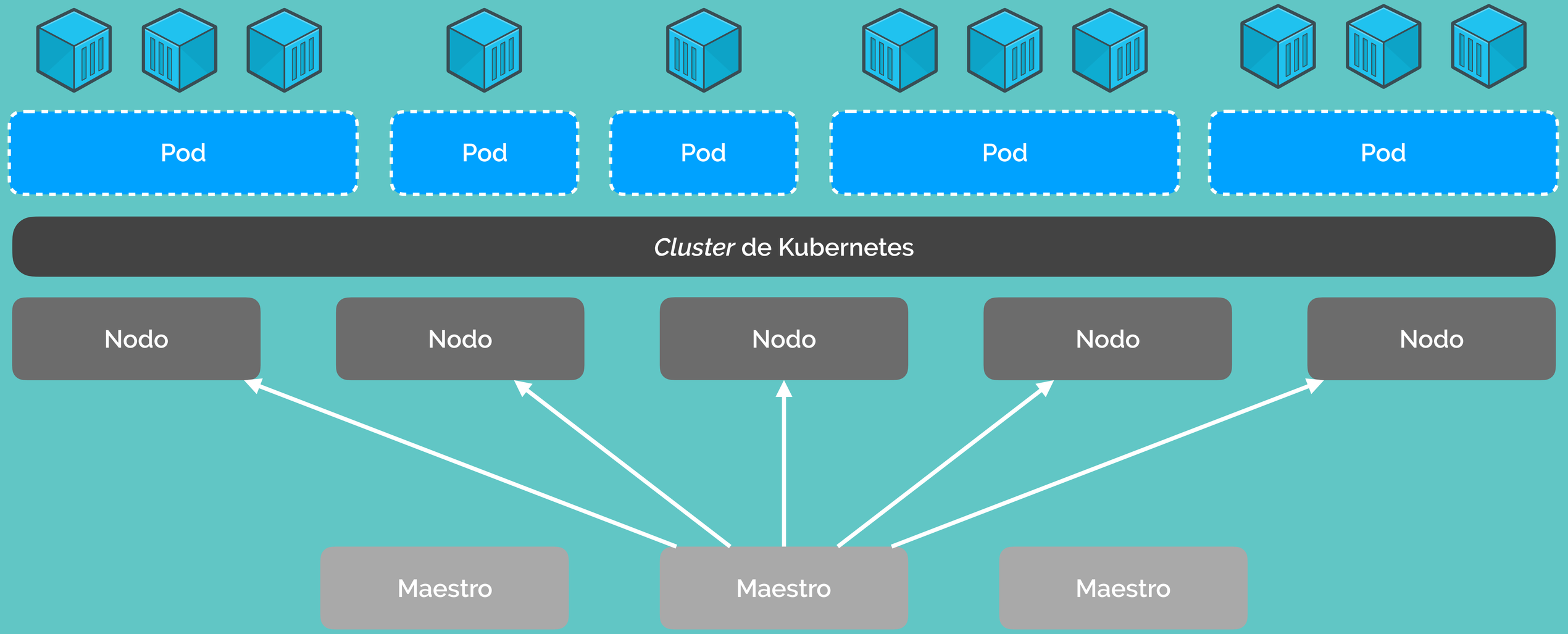
Otros orquestadores

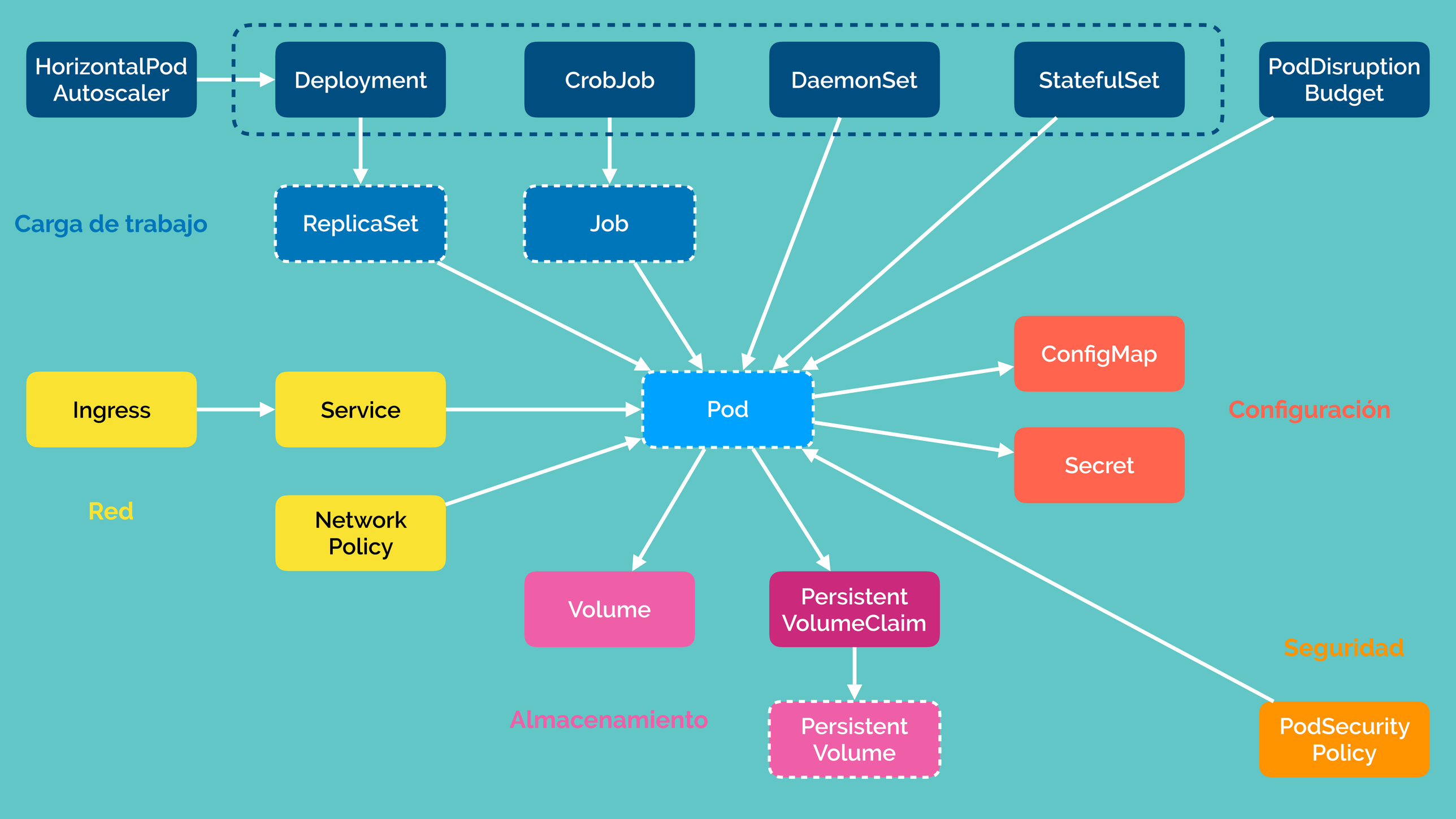
Apache
MESOS™

HashiCorp
Nomad



Kubernetes: arquitectura




HorizontalPod Autoscaler
Deployment
CrobJob
DaemonSet
StatefulSet
PodDisruption Budget
Carga de trabajo
ReplicaSet
Job
ConfigMap
Ingress
Service
Pod
Configuración
Secret
Red
Network Policy
Volume
Persistent VolumeClaim
Seguridad
Almacenamiento
Persistent Volume
PodSecurity Policy

Objetos

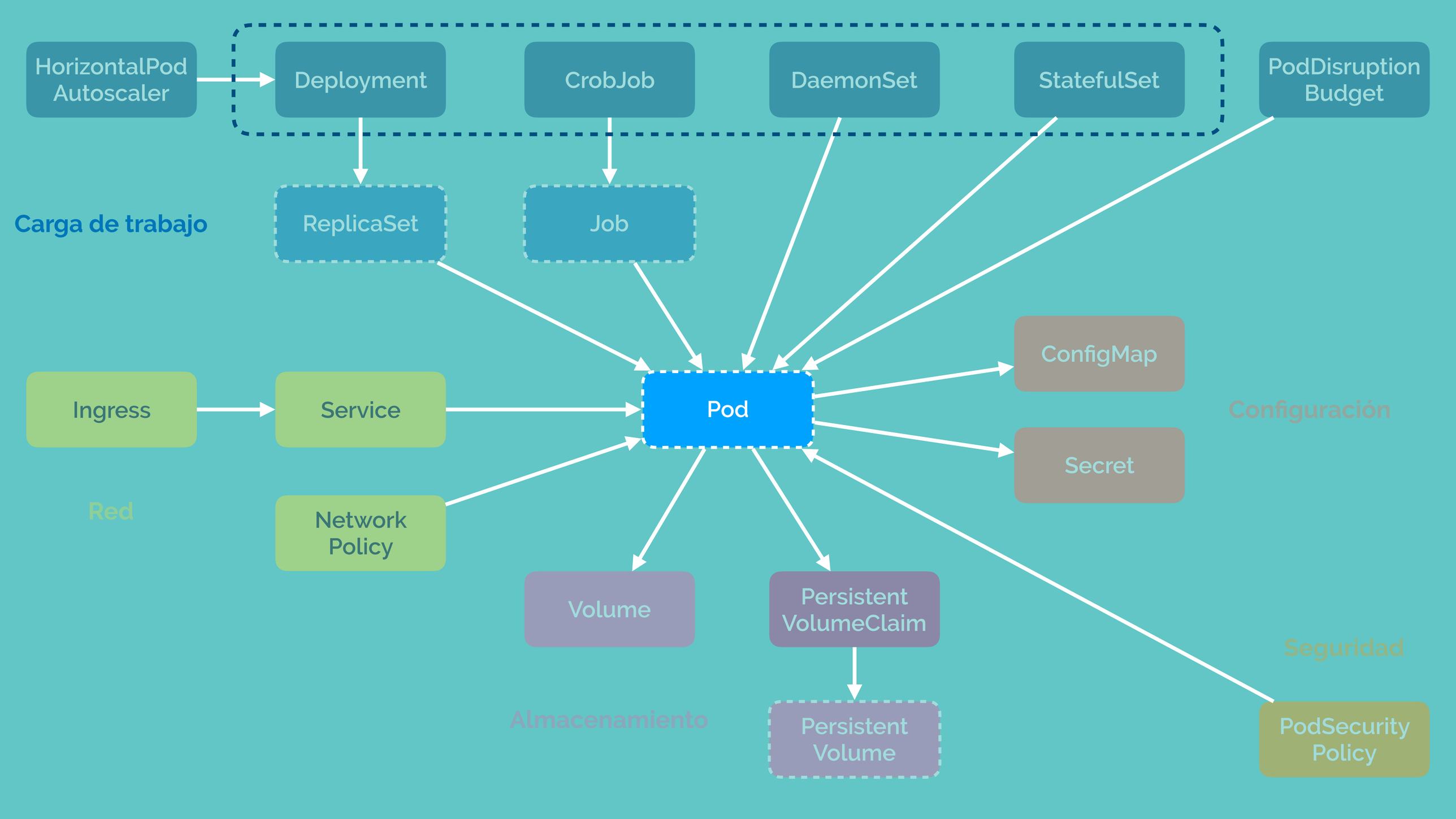
- Kubernetes define bloques básicos (**objetos**) que representan **recursos**.
- Los objetos se definen mediante **especificaciones** en formato **YAML**.
- Una especificación define el **estado deseado** junto con algunos **metadatos**.

```yaml
kind: string
apiVersion: string

metadata:
  name: string
  namespace: string
  labels: {}
  annotations: {}

spec: {}
```

Objetos: metadatos

- Además del **nombre** y del **espacio de nombres**, existen las etiquetas y anotaciones.
- Las **etiquetas** son **identificativas** y relacionan objetos entre sí.
- Las **anotaciones no** son **identificativas**.

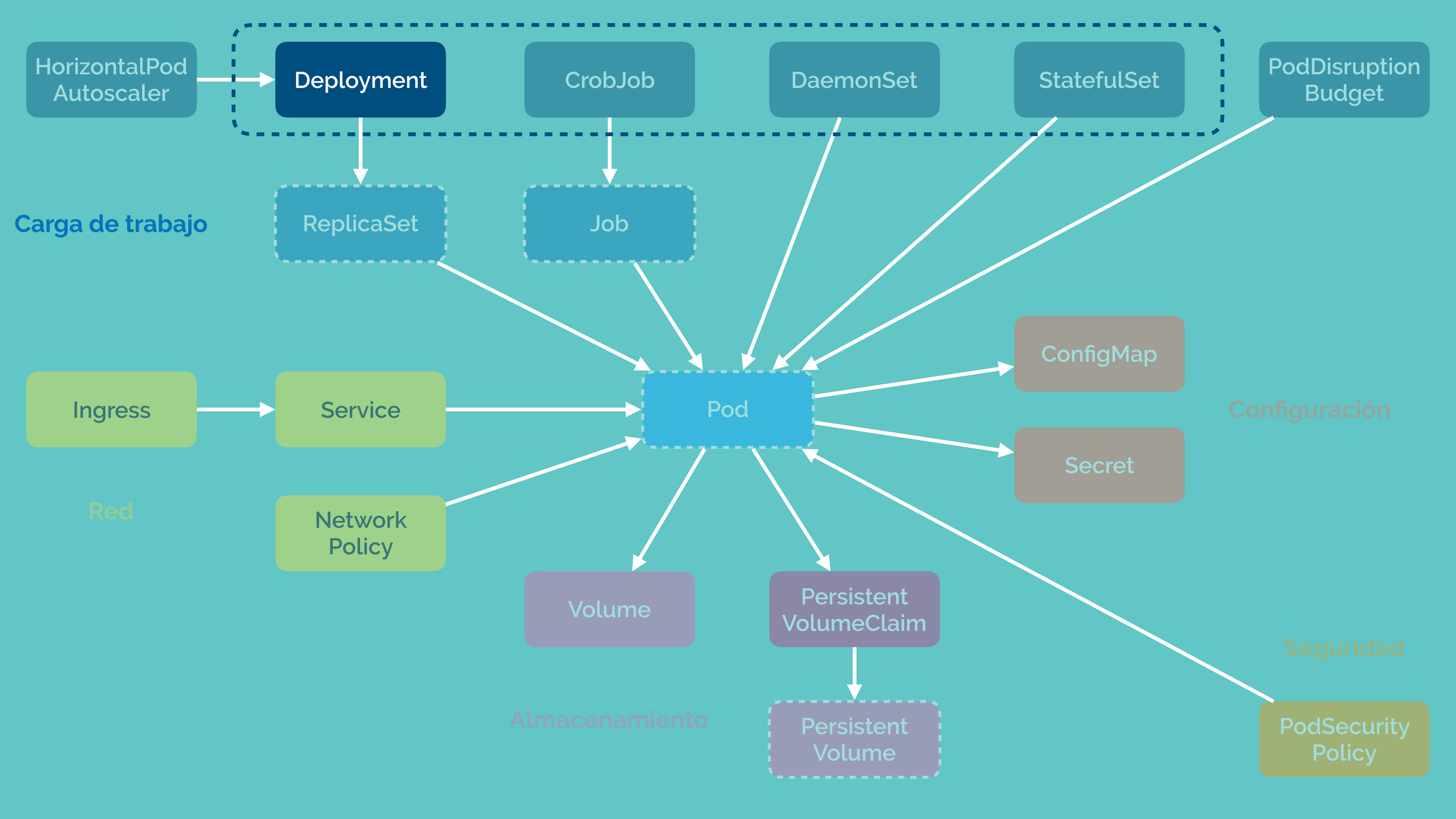
```yaml
metadata:
  name: hello-k8s
  namespace: dixibox
  labels:
    app: hello-k8s
    role: replica
  annotations:
    kubernetes.io/change-cause: 'Release 1.1.0'
    prometheus.io/scrape: 'true'
```


HorizontalPod Autoscaler
Deployment
CrobJob
DaemonSet
StatefulSet
PodDisruption Budget
Carga de trabajo
ReplicaSet
Job
ConfigMap
Ingress
Service
Pod
Configuración
Secret
Red
Network Policy
Volume
Persistent VolumeClaim
Seguridad
Almacenamiento
Persistent Volume
PodSecurity Policy

Objetos:

- Es el objeto **elemental** de Kubernetes y la **unidad mínima** de despliegue.
- **Encapsula** uno o varios **contenedores**.
- Tiene una **IP propia** privada dentro del *cluster*.
- No se trabaja directamente con ellos, sino con los **controladores**.

```yaml
kind: Pod
apiVersion: v1

metadata:
  name: hello-k8s
  namespace: dixibox
  labels:
    app: hello-k8s

spec:
  containers:
    - name: app
      image: busybox
      command:
        - sh
        - '-c'
        - 'echo "Hello Kubernetes!" && sleep 3600’
```


HorizontalPod Autoscaler
Deployment
CrobJob
DaemonSet
StatefulSet
PodDisruption Budget
Carga de trabajo
ReplicaSet
Job
ConfigMap
Ingress
Service
Pod
Configuración
Secret
Red
Network Policy
Volume
Persistent VolumeClaim
Seguridad
Almacenamiento
Persistent Volume
PodSecurity Policy

Objetos: Deployment

- Se encarga de mantener en ejecución un **conjunto** de *pods* idénticos (réplicas).
- Se asume que los *pods* deben estar en ejecución de forma permanente.
- Al hacer **cambios**, el controlador los aplica siguiendo una **estrategia**.
- Permite **deshacer cambios**.
- Incluye la especificación del *pod* dentro de él.

Objetos:

Deployment

```yaml
kind: Deployment
apiVersion: apps/v1

metadata:
  name: website
  namespace: dixibox
  labels:
    app: website

spec:
  replicas: 2
  revisionHistoryLimit: 4

  strategy:
    rollingUpdate:
      maxUnavailable: 1

  selector:
    matchLabels:
      app: website
```

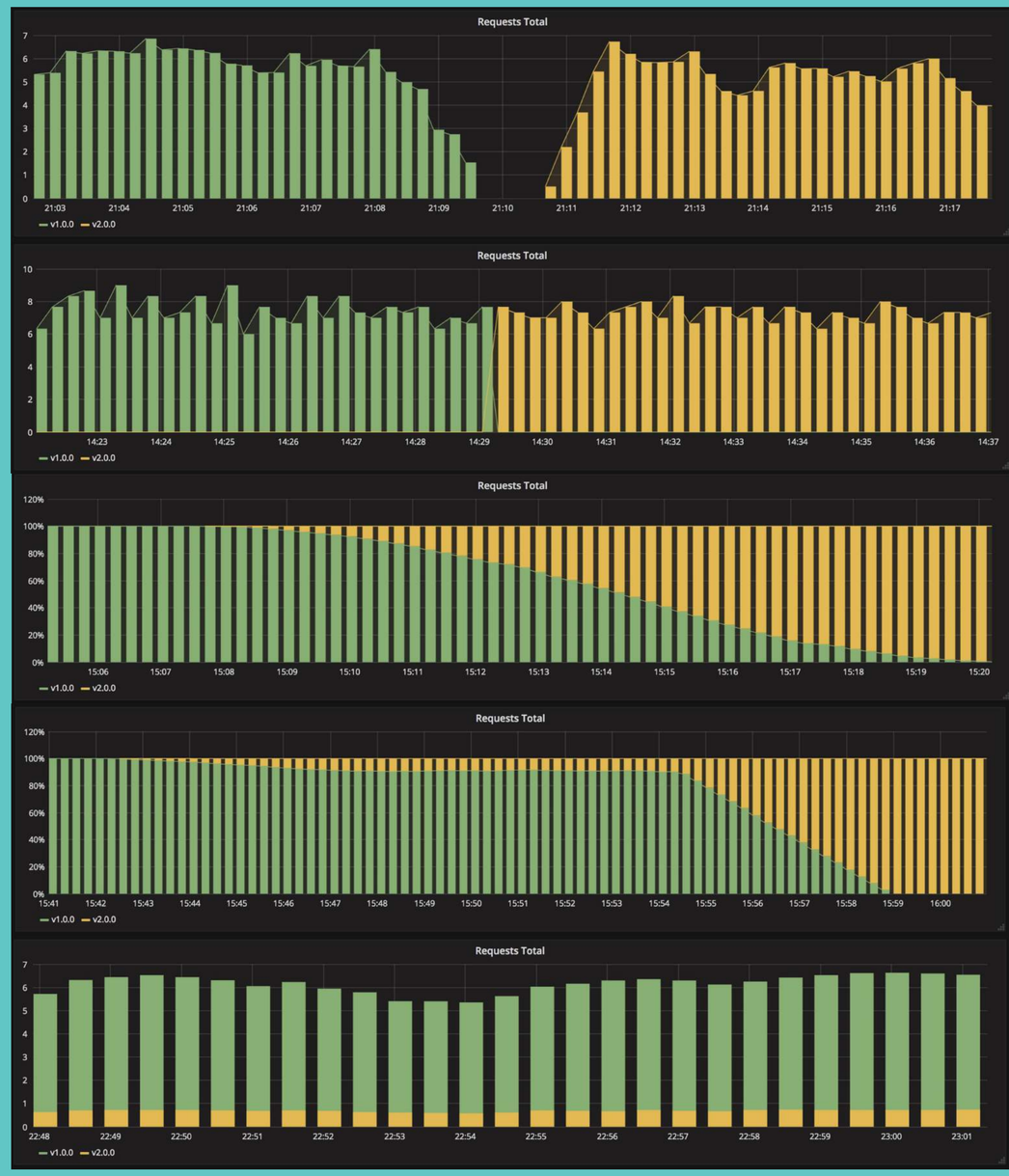
Pod

```yaml
  template:
    metadata:
      labels:
        app: website

    spec:
      containers:
        - name: app
          image: nginx:1.15.10-alpine
          imagePullPolicy: Always
          ports:
            - name: http
              containerPort: 3000
          lifecycle:
            preStop:
              exec:
                command:
                  - nginx
                  - '-s'
                  - quit
```

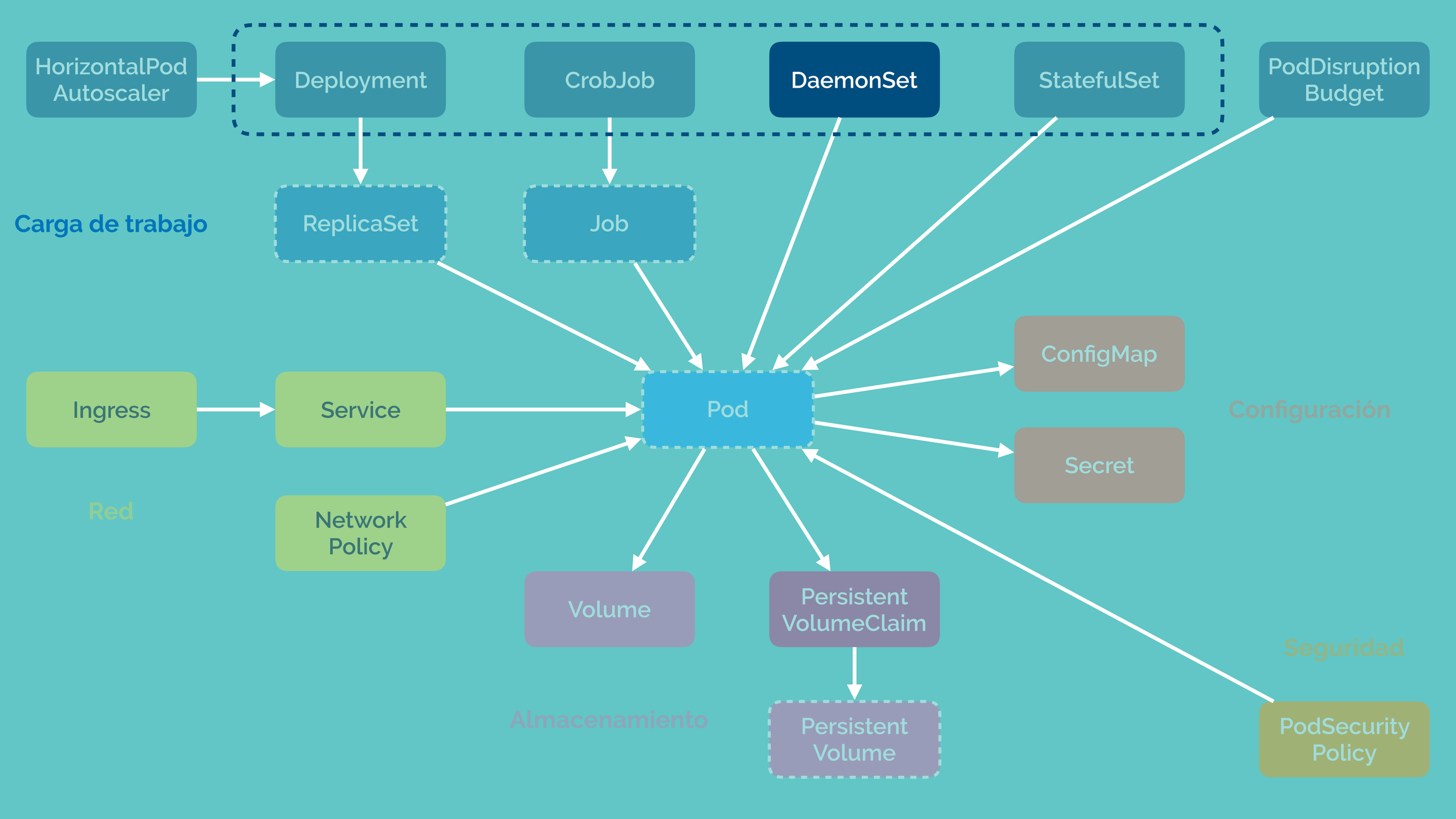
```yaml
          resources:
            requests:
              cpu: 10m
              memory: 20Mi
            limits:
              cpu: 20m
              memory: 30Mi
          livenessProbe:
            tcpSocket:
              port: http
            initialDelaySeconds: 5
            timeoutSeconds: 2
```

Estrategias de actualización

- ***Recreate*.** Se eliminan los *pods* viejos y a continuación se crean los nuevos.
- ***Blue/green*.** La operación inversa que con *recreate*.
- ***Rolling*.** Se van creando los *pods* nuevos mientras se van eliminando los viejos, todo de forma progresiva.
- ***Canary*.** Se crean *pods* nuevos para un subconjunto específico de usuarios y se va ampliando el conjunto hasta el total, de forma cada vez más acentuada.
- ***A/B testing*.** Se crean *pods* nuevos únicamente para un subconjunto de usuarios en base a algún criterio.



Fuente: Deployment Strategies on Kubernetes


HorizontalPod Autoscaler
Deployment
CrobJob
DaemonSet
StatefulSet
PodDisruption Budget
Carga de trabajo
ReplicaSet
Job
ConfigMap
Ingress
Service
Pod
Configuración
Secret
Red
Network Policy
Volume
Persistent VolumeClaim
Seguridad
Almacenamiento
Persistent Volume
PodSecurity Policy

Objetos: DaemonSet

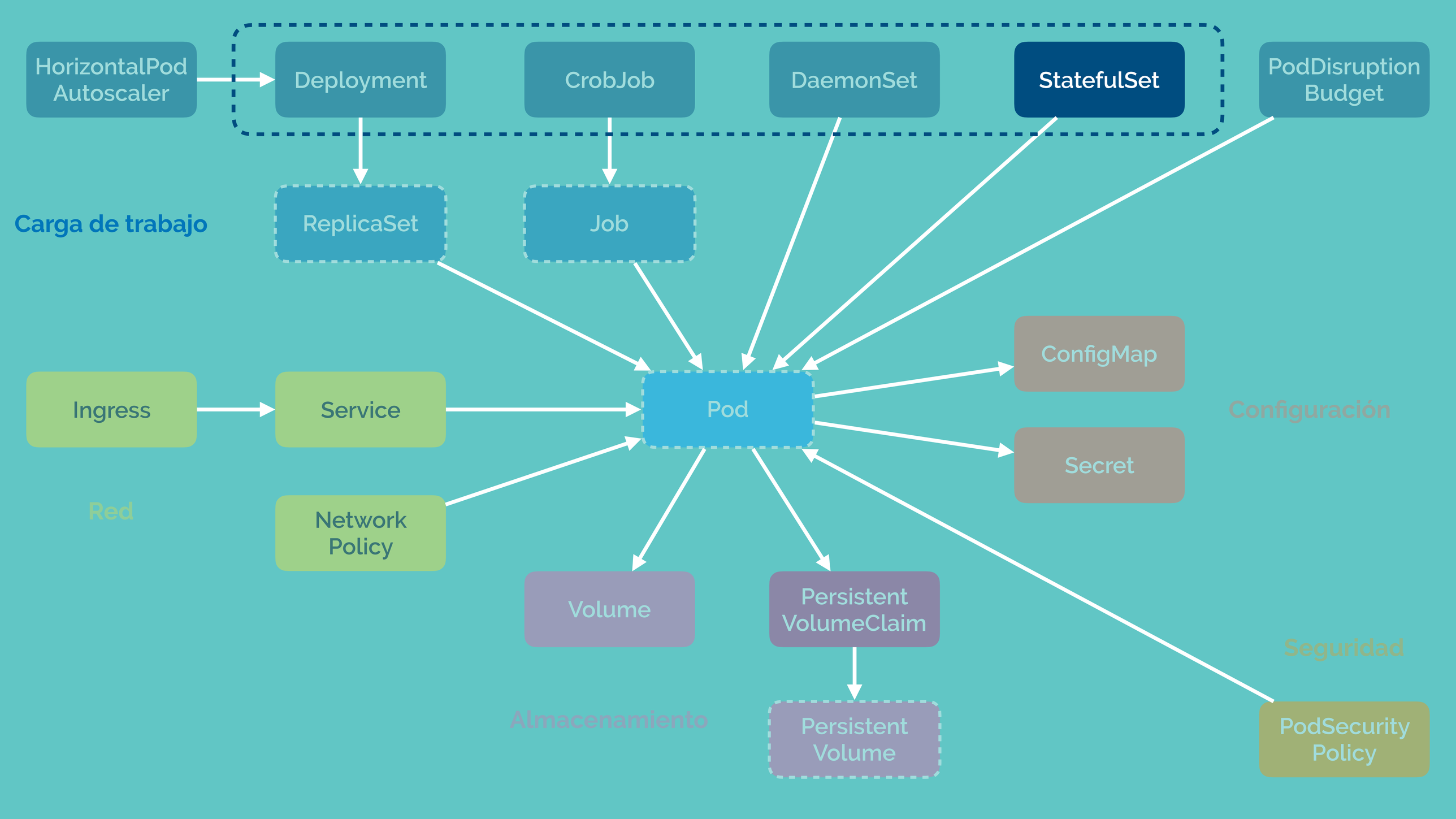
- **Similar** a un objeto de tipo *deployment*.
- Tiene **tantas réplicas como nodos**.
- **Cada réplica** se ejecuta **en un nodo diferente**.
- Útil para **casos de uso específicos**, como:
  - Recolección de métricas de todos los nodos.
  - Ofrecer servicios dependientes del nodo.

Objetos: DaemonSet

```yaml
kind: DaemonSet
apiVersion: apps/v1

metadata:
  name: node-exporter
  namespace: monitoring
  labels:
    app: node-exporter

spec:
  selector:
    matchLabels:
      app: node-exporter

  template:
    metadata:
      labels:
        app: node-exporter
```

Pod

```yaml
spec:
  priorityClassName: system-node-critical

  containers:
    - name: app
      image: prom/node-exporter:v0.17.0
      args:
        - --path.procfs=/host/proc
        - --path.sysfs=/host/sys
      ports:
        - name: metrics
          containerPort: 9100
      resources:
        limits:
          cpu: 10m
          memory: 50Mi
        requests:
          cpu: 10m
          memory: 50Mi
```


HorizontalPod Autoscaler
Deployment
CrobJob
DaemonSet
StatefulSet
PodDisruption Budget
Carga de trabajo
ReplicaSet
Job
ConfigMap
Ingress
Service
Pod
Configuración
Secret
Red
Network Policy
Volume
Persistent VolumeClaim
Seguridad
Almacenamiento
Persistent Volume
PodSecurity Policy

Objetos: StatefulSet

- **Similar** a un objeto de tipo *deployment* pero **con estado**.
- **Cada réplica** tiene una **identidad** propia y es **única**.
- Las **réplicas** se **crean**, **destruyen** y **escalan** de forma **ordenada**.
- Al tener estado, cada réplica tiene su propia **persistencia**.

Objetos:

StatefulSet

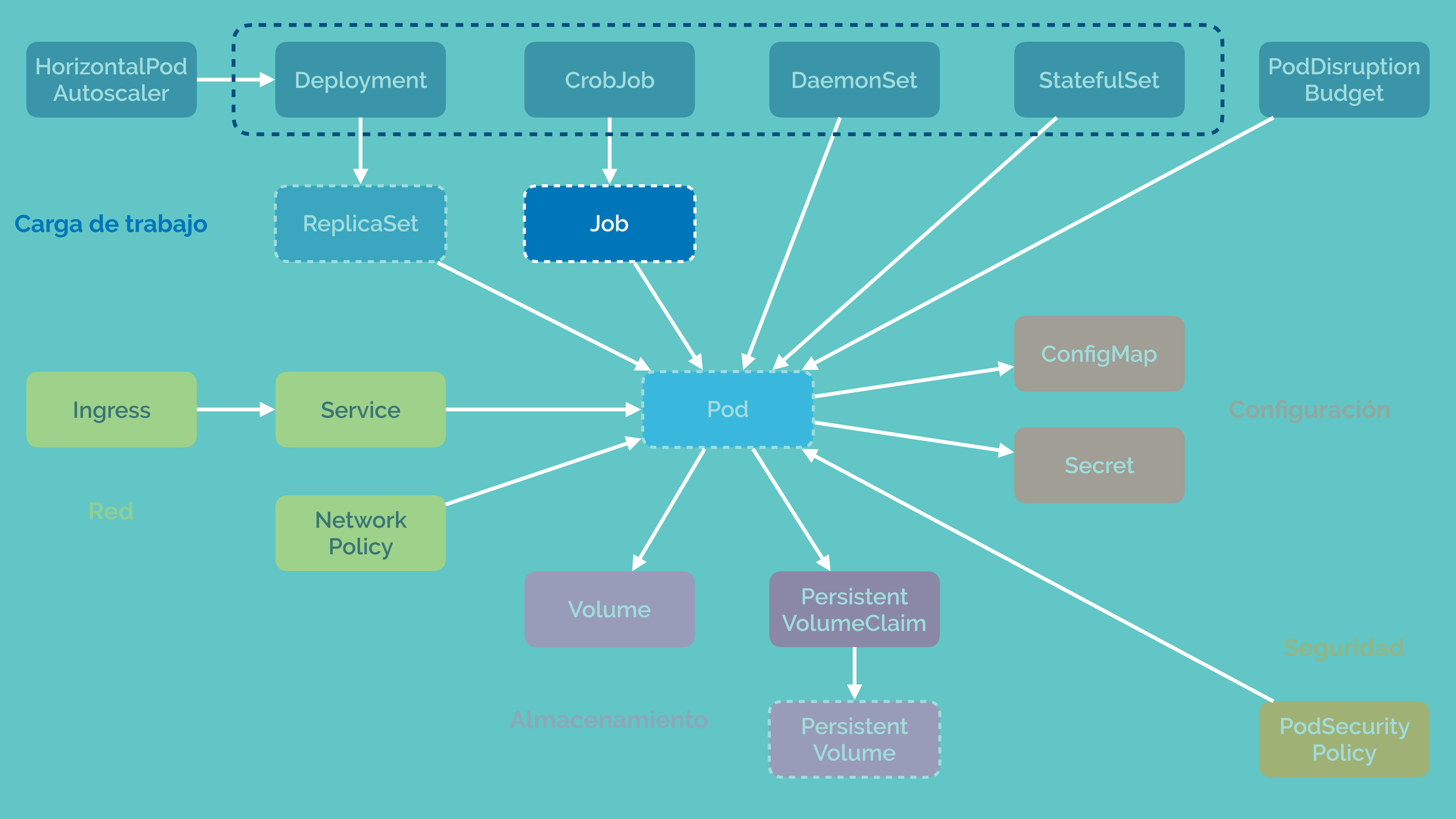
```yaml
kind: StatefulSet
apiVersion: apps/v1

metadata:
  name: mariadb
  namespace: dixibox
  labels:
    app: mariadb

spec:
  replicas: 1
  serviceName: mariadb

  updateStrategy:
    type: RollingUpdate

  selector:
    matchLabels:
      app: mariadb

  template:
    metadata:
      labels:
        app: mariadb
```

Pod

```yaml
    spec:
      terminationGracePeriodSeconds: 300
      containers:
        - name: app
          image: mariadb:10.3.13
          volumeMounts:
            - name: data
              mountPath: /var/lib/mysql
            - name: config
              mountPath: /etc/mysql/conf.d/
my.cnf
              subPath: my.cnf
          ports:
            - name: mariadb
              containerPort: 3306
          resources:
            requests:
              cpu: 100m
              memory: 256Mi
            limits:
              cpu: 500m
              memory: 512Mi
          livenessProbe:
            exec:
              command: ['sh', '-c', 'exec
mysqladmin status -uroot -
p$MYSQL_ROOT_PASSWORD']
            initialDelaySeconds: 120
          readinessProbe:
            exec:
              command: ['sh', '-c', 'exec
mysqladmin status -uroot -
p$MYSQL_ROOT_PASSWORD']
            initialDelaySeconds: 15
          securityContext:
            runAsUser: 999

      volumes:
        - name: config
          configMap:
            name: mariadb

  volumeClaimTemplates:
    - metadata:
        name: data
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: 4Gi
```

Volume

Persistent VolumeClaim


HorizontalPod Autoscaler
Deployment
CrobJob
DaemonSet
StatefulSet
PodDisruption Budget
Carga de trabajo
ReplicaSet
Job
ConfigMap
Ingress
Service
Pod
Configuración
Secret
Red
Network Policy
Volume
Persistent VolumeClaim
Seguridad
Almacenamiento
Persistent Volume
PodSecurity Policy

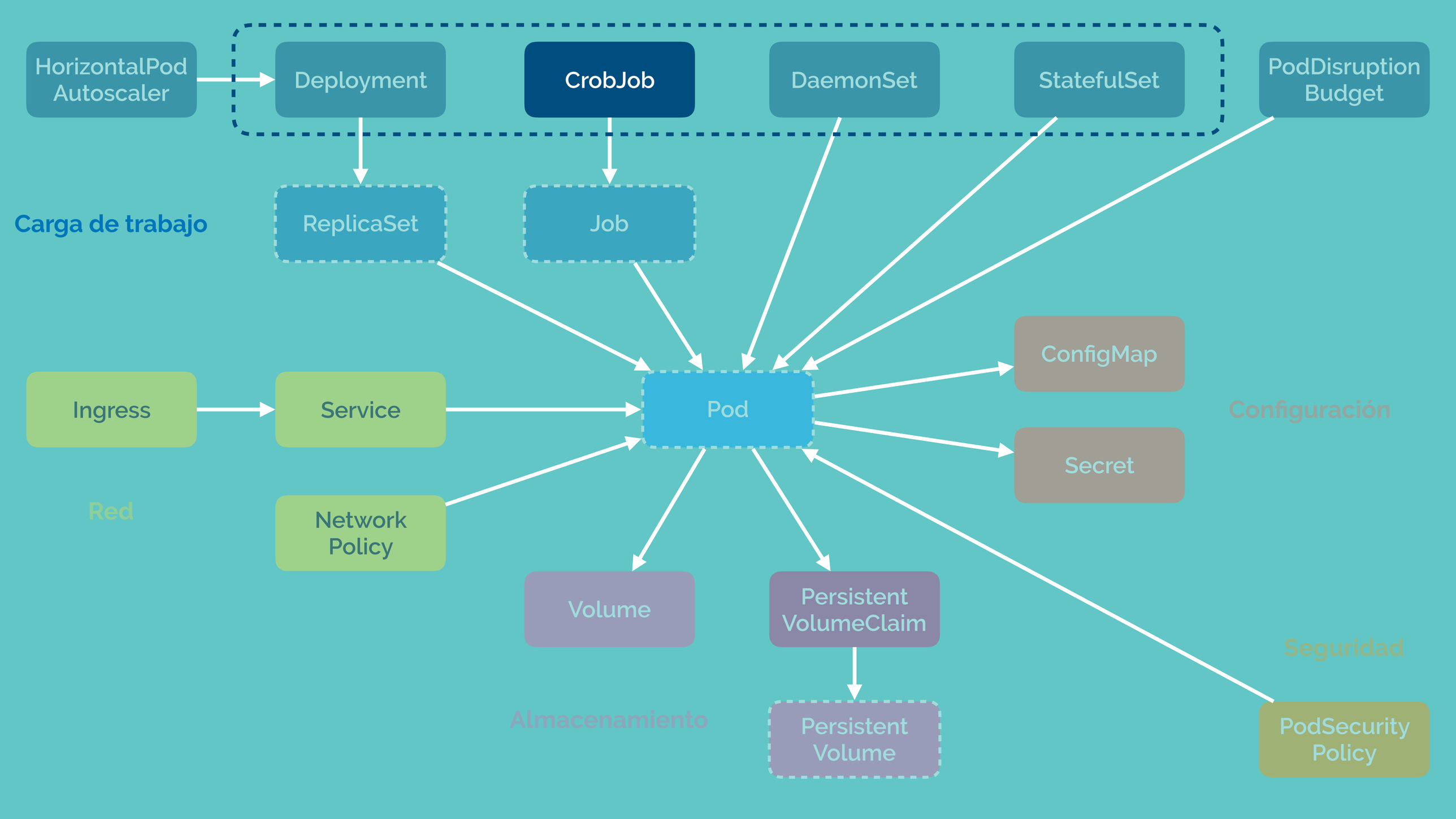
Objetos:

Job

- Un objeto de tipo *job* **crea** uno o diversos *pods*.
- Se asume que **la ejecución debe terminar**.
- Se asegura que **terminan** su ejecución **correctamente**.

Pod

```yaml
kind: Job
apiVersion: batch/v1

metadata:
  name: pi

spec:
  backoffLimit: 4

  template:
    spec:
      restartPolicy: Never

      containers:
        - name: app
          image: perl
          command:
            - perl
            - -Mbignum=bpi
            - -wle
            - print bpi(2000)
```


HorizontalPod Autoscaler
Deployment
CrobJob
DaemonSet
StatefulSet
PodDisruption Budget
Carga de trabajo
ReplicaSet
Job
ConfigMap
Ingress
Service
Pod
Configuración
Secret
Red
Network Policy
Volume
Persistent VolumeClaim
Seguridad
Almacenamiento
Persistent Volume
PodSecurity Policy

Objetos:

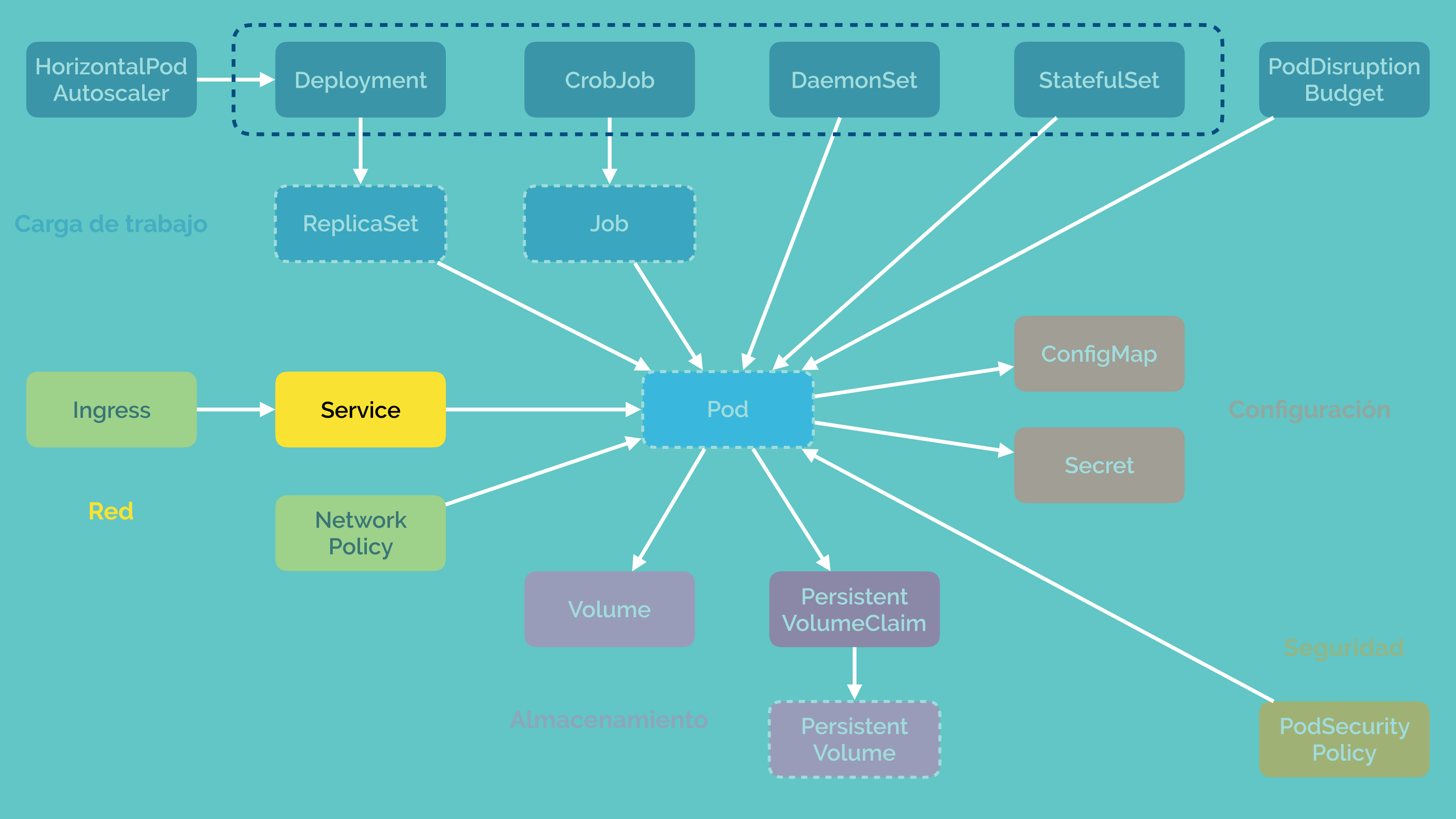
CronJob

- Un objeto de tipo *cron job* **crea** un ***job*** de forma **repetida** y **planificada** en el tiempo.
- Las repeticiones tienen la **misma sintaxis** que las tareas **Cron** de **UNIX**.
- Tienen bastantes **limitaciones**.
- **Ineficaces** para **repeticiones** muy **frecuentes**.

Job

Pod

```yaml
kind: CronJob
apiVersion: batch/v1beta1

metadata:
  name: hello

spec:
  schedule: '*/1 * * * *'

  jobTemplate:
    spec:
      template:
        spec:
          restartPolicy: OnFailure

          containers:
            - name: app
              image: busybox
              args:
                - /bin/sh
                - -c
                - echo Hello from the Kubernetes cluster
```


HorizontalPod Autoscaler
Deployment
CrobJob
DaemonSet
StatefulSet
PodDisruption Budget
Carga de trabajo
ReplicaSet
Job
ConfigMap
Ingress
Service
Pod
Configuración
Secret
Red
Network Policy
Volume
Persistent VolumeClaim
Seguridad
Almacenamiento
Persistent Volume
PodSecurity Policy

Objetos:

Service

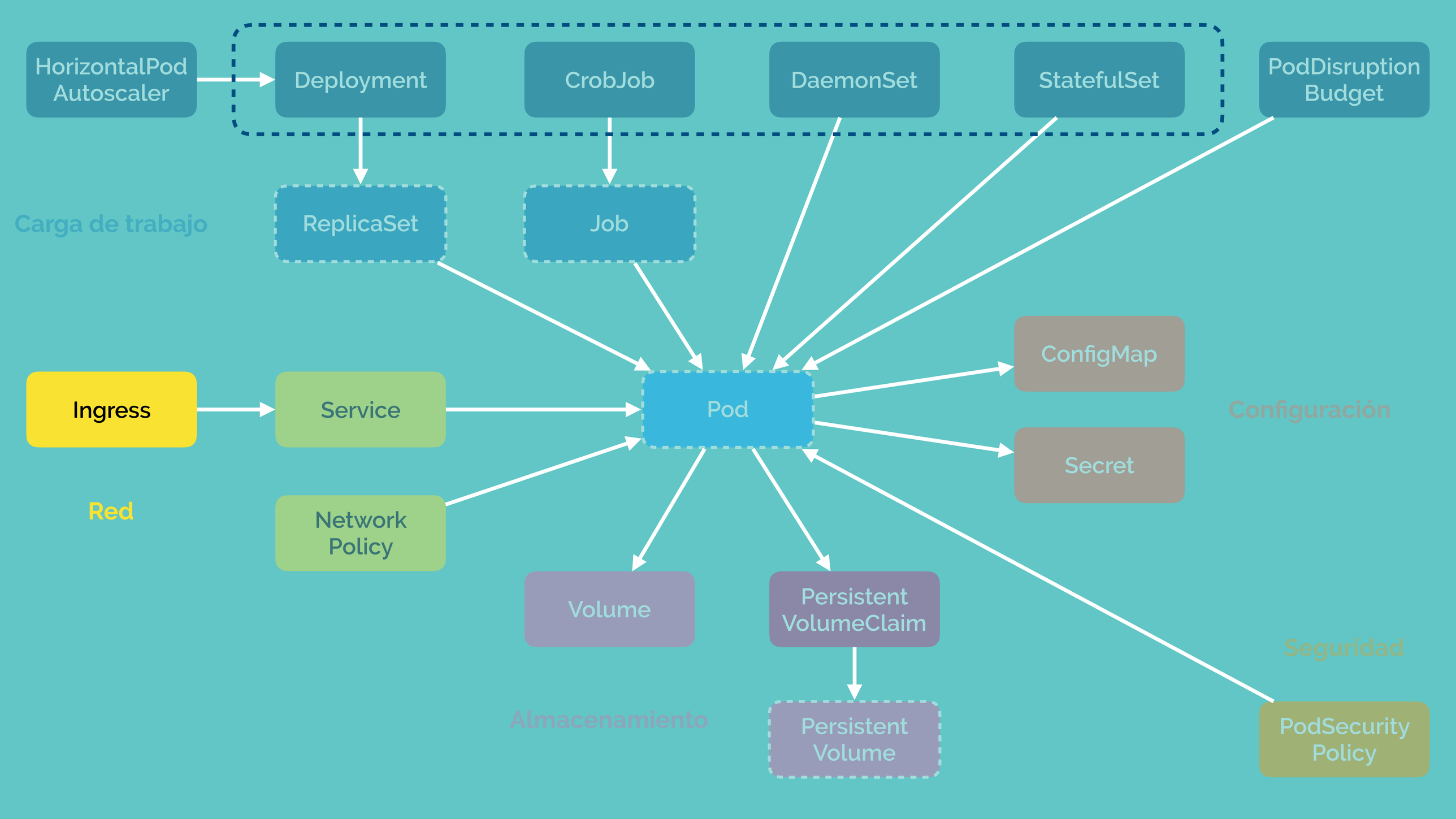
- Los ***pods*** son **efímeros**, por lo que sus **direcciones de red cambian**.
- Los **servicios exponen** conjuntos de ***pods*** bajo un **único nombre lógico**.
- Pueden actuar como **balanceadores de carga** muy primitivos.
- Son **imprescindibles** para **exponer *pods*** dentro y fuera del *cluster* a través de la red.
- Los hay de diferentes **tipos**: **ClusterIP**, **NodePort**, **LoadBalancer** y **ExternalName**.

Objetos:

Service

```yaml
kind: Service
apiVersion: v1

metadata:
  name: website
  namespace: dixibox
  labels:
    app: website

spec:
  selector:
    app: website

  ports:
    - name: http
      port: 80
      targetPort: http
```

```yaml
kind: Service
apiVersion: v1

metadata:
  name: mariadb
  namespace: dixibox
  labels:
    app: mariadb
  annotations:
    prometheus.io/scrape: 'true'

spec:
  selector:
    app: mariadb

  clusterIP: None

  ports:
    - name: mariadb
      port: 3306
      targetPort: mariadb
```


HorizontalPod Autoscaler
Deployment
CrobJob
DaemonSet
StatefulSet
PodDisruption Budget
Carga de trabajo
ReplicaSet
Job
ConfigMap
Ingress
Service
Pod
Configuración
Secret
Red
Network Policy
Volume
Persistent VolumeClaim
Seguridad
Almacenamiento
Persistent Volume
PodSecurity Policy

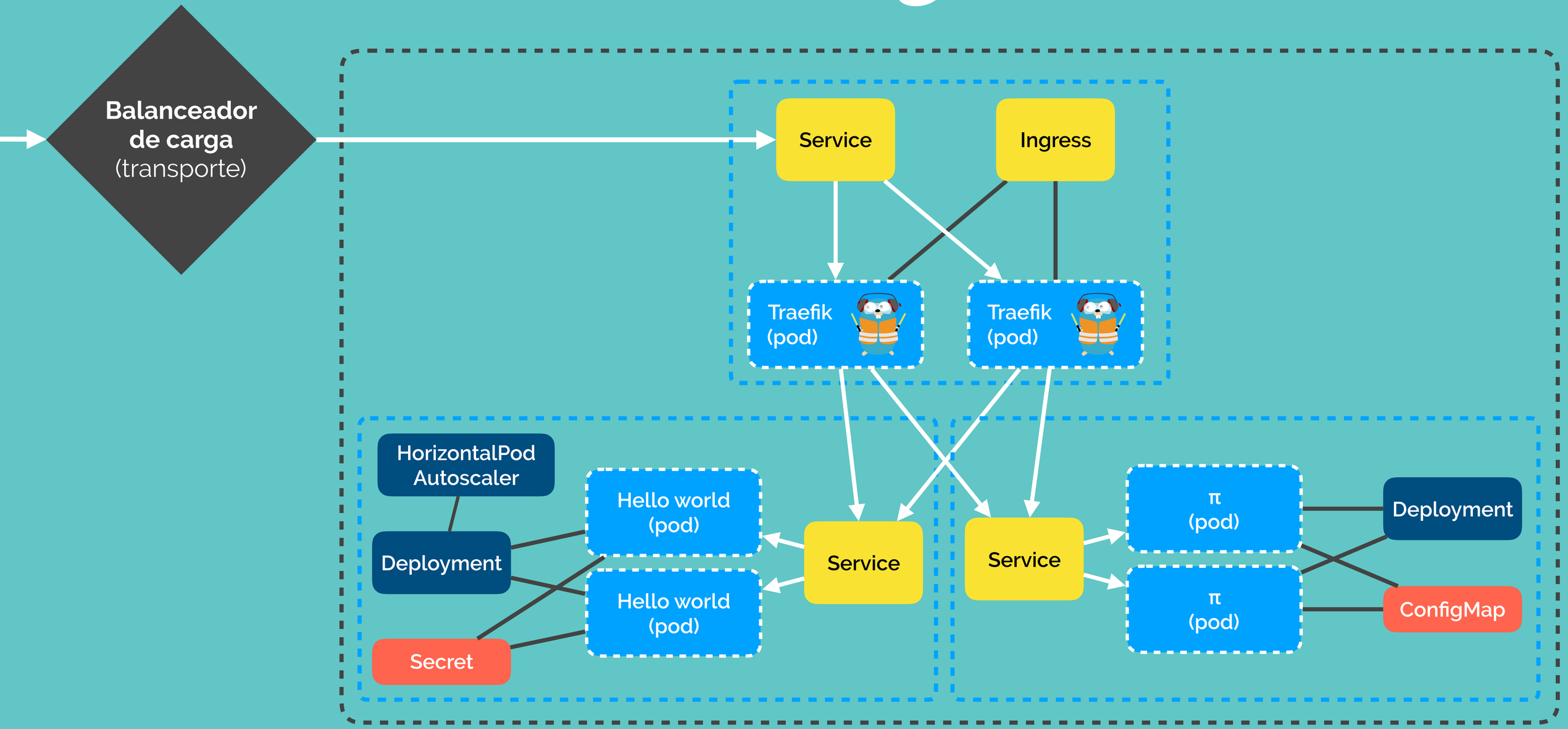
Objetos:

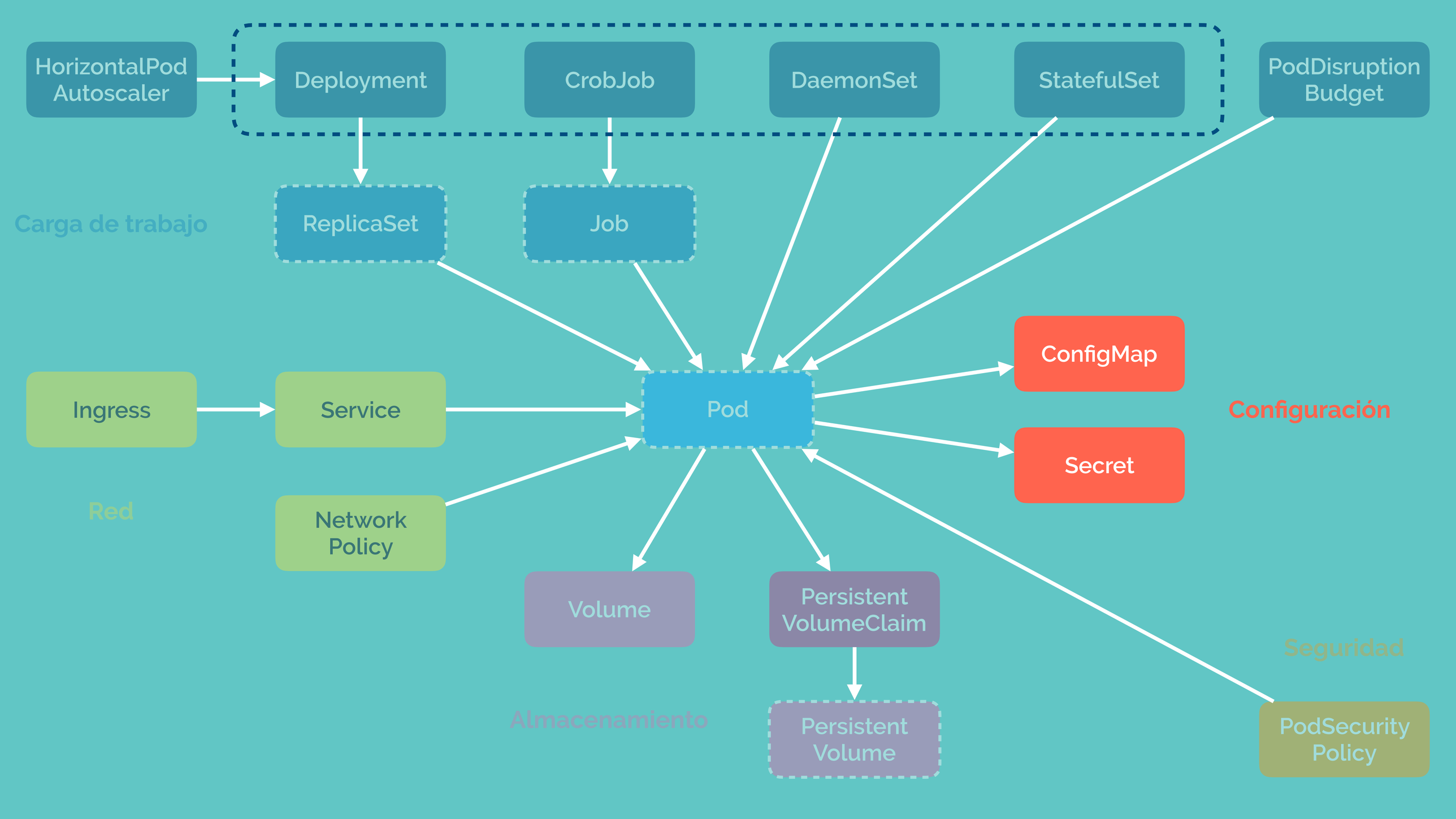
Ingress

- Es una extensión que **nace** por la necesidad de **ahorrar** costes **en balanceadores** de carga.
- Es un **enrutador** de **alto nivel** (capa de aplicación).
- **Necesita** un **controlador** (un proxy inverso) que no viene por defecto (Traefik o Nginx, por ejemplo).
- **Requiere** un único **balanceador de carga externo** (nivel de transporte) por *cluster*, en lugar de uno por servicio.

```yaml
kind: Ingress
apiVersion: extensions/v1beta1

metadata:
  name: face
  namespace: dixibox
  labels:
    app: face
  annotations:
    ingress.kubernetes.io/hsts-max-age: '31536000'
    ingress.kubernetes.io/hsts-include-subdomains: 'true'
    ingress.kubernetes.io/hsts-preload: 'true'
    ingress.kubernetes.io/force-hsts: 'true'

spec:
  rules:
    - host: dixibox.com
      http:
        paths:
          - backend:
              serviceName: face
              servicePort: http
```

Services e ingresses




HorizontalPod Autoscaler
Deployment
CrobJob
DaemonSet
StatefulSet
PodDisruption Budget
Carga de trabajo
ReplicaSet
Job
ConfigMap
Ingress
Service
Pod
Configuración
Secret
Red
Network Policy
Volume
Persistent VolumeClaim
Seguridad
Almacenamiento
Persistent Volume
PodSecurity Policy

Objetos:

ConfigMap

Secret

- Ambos permiten **almacenar** variables de **configuración**.
- Los ***config maps*** almacenan **configuración no sensible** que puede (y debe) estar en los repositorios de software.
- Los ***secrets*** almacenan **configuración sensible** como contraseñas, *tokens* o claves.
- Los **contenedores no deben contener** ningún tipo de **configuración**. Esta debe ser **inyectada** en los ***pods*** en tiempo de ejecución.

Objetos:

ConfigMap

Secret

```yaml
kind: ConfigMap
apiVersion: v1

metadata:
  name: brain
  namespace: dixibox
  labels:
    app: brain

data:
  APP_ENV: production
  APP_URL: https://brain.dixibox.com
  DB_HOST: mariadb
  DB_DATABASE: brain
  DB_USERNAME: brain
  REDIS_HOST: redis
```

```yaml
kind: Secret
apiVersion: v1

metadata:
  name: brain
  namespace: dixibox
  labels:
    app: brain

type: Opaque

stringData:
  APP_KEY: v3ryS3cr3tK3y
  DB_PASSWORD: dbP4ssw0rd
  AWS_SECRET_ACCESS_KEY: 4wsS3cr3t
  RECAPTCHA_SECRET_KEY: r3c4ptch4S3cr3t
  STRIPE_SECRET: str1p3S3cr3t
  STRIPE_ENDPOINT_SECRET: an0th3rOn3
```

Kubectl

- **Interfaz de línea de comandos** (CLI) para interactuar con la API de Kubernetes.
- Permite crear objetos y aplicar cambios en los mismos dada su especificación (**configuración declarativa**).
- También es posible interactuar con el *cluster* a través de comandos específicos (**configuración imperativa**).

Helm

- **Gestionar** los objetos **manualmente** es **tedioso** y **repetitivo**.
- **Helm** es el **gestor de paquetes** de Kubernetes.
- Los **paquetes** se llaman **charts** y tienen una estructura concreta.
- Permite el uso de **plantillas** para las especificaciones de los objetos.
- Dispone de un **repositorio** de paquetes **oficial**.
- Es recomendable utilizarlo únicamente como software de plantillas en el lado del cliente, aunque ofrece otras opciones.

Kubernetes en Dixibox

- **Prometheus** recolecta métricas, **Loki** recolecta registros (logs) y **Grafana** los muestra.
- **Jaeger** recopila y muestra datos sobre la ejecución del software (*tracing*).
- **Traefik** es el *ingress controller* y actúa como proxy inverso HTTP.
- **Restic** realiza copias de seguridad incrementales.

Kubernetes en Dixibox

- El **registro de Docker** almacena las imágenes de Docker privadas.
- **Jenkins** se encarga de ejecutar las tuberías (*pipelines*) de integración continua:
  - Construye contenedores Docker y los guarda en el registro.
  - Realiza análisis estático.
  - Ejecuta las pruebas (*tests*) unitarias, de característica y de integración.
  - Despliega el software con **Helm** y **Kubectl**.


dixibox