



Introducción a Kubernetes

Autor: Víctor Díaz Marco
Publicación: 11 de abril de 2019
Actualización: 12 de abril de 2023



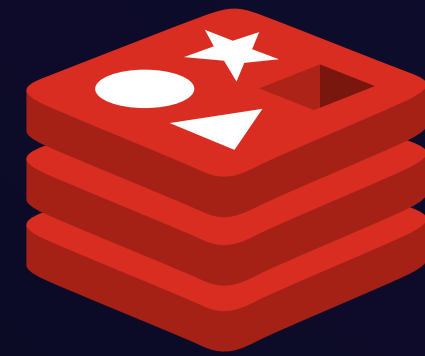
brutal.systems

ContenedORIZACIÓN

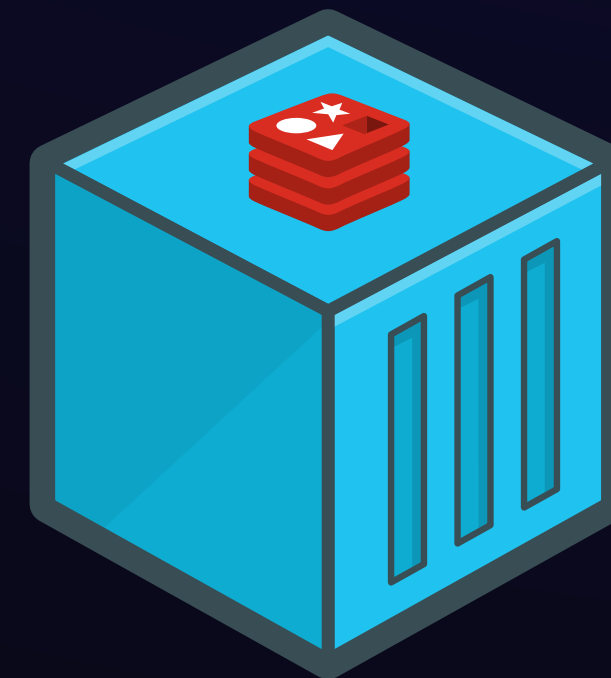
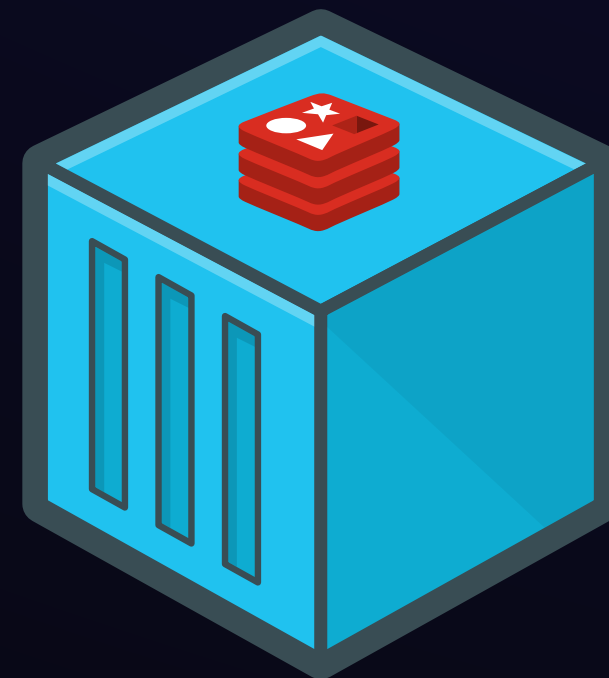
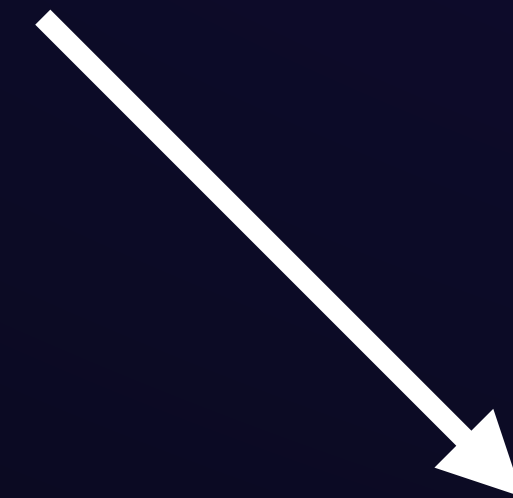
- Técnica de distribución de software.
- Permite empaquetar, distribuir y ejecutar software en paquetes que incluyen todas sus dependencias.
- Estandarizada por la Open Container Initiative, proyecto de la Fundación Linux.
- Utiliza las características de aislamiento de procesos del kernel de Linux.



Contenedorización



Imágenes



Contenedores

Kubernetes

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



kubernetes

Kubernetes como servicio



Google Cloud



DigitalOcean



OVH.com

Kubernetes con instalación propia



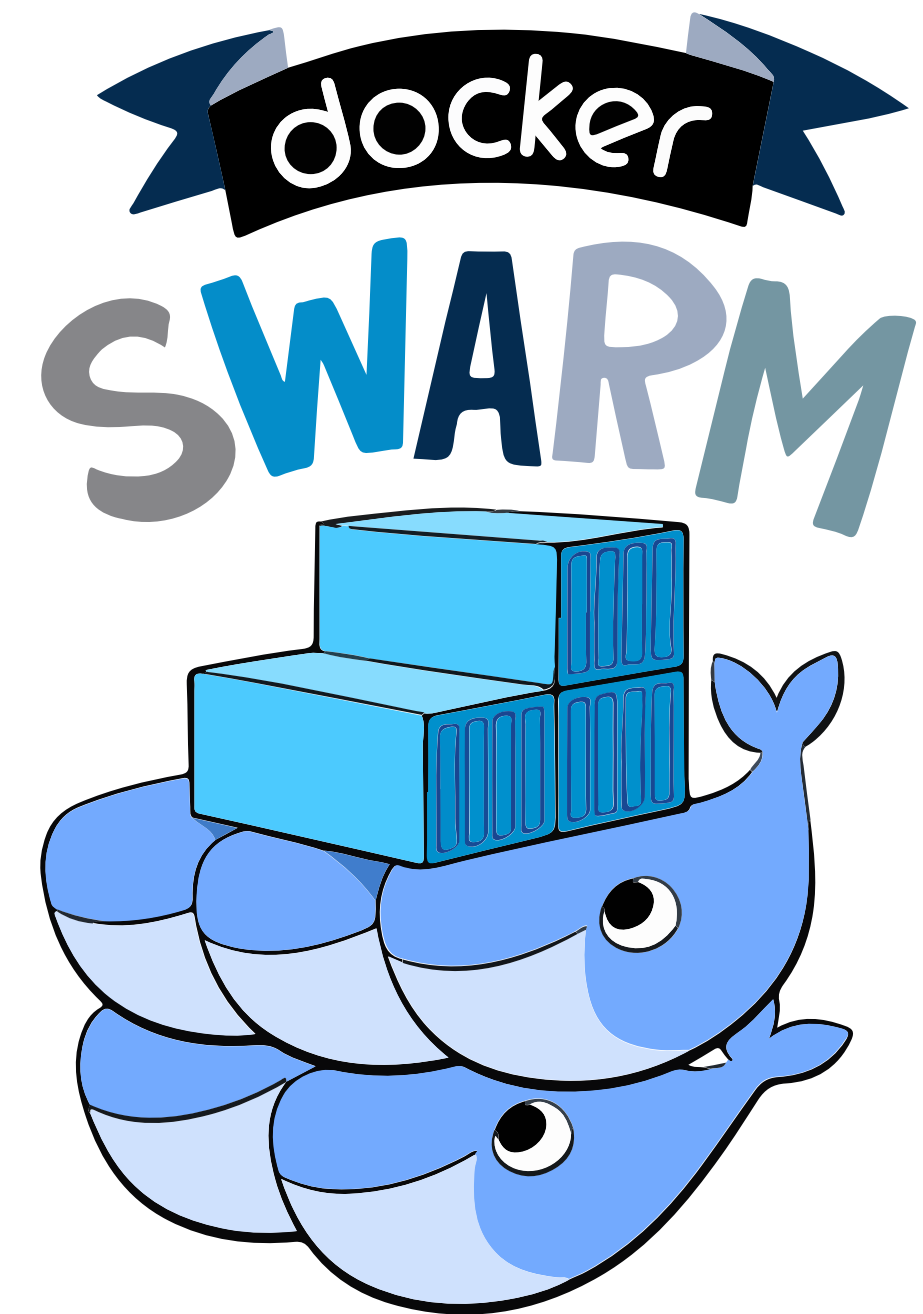
Otros orquestadores



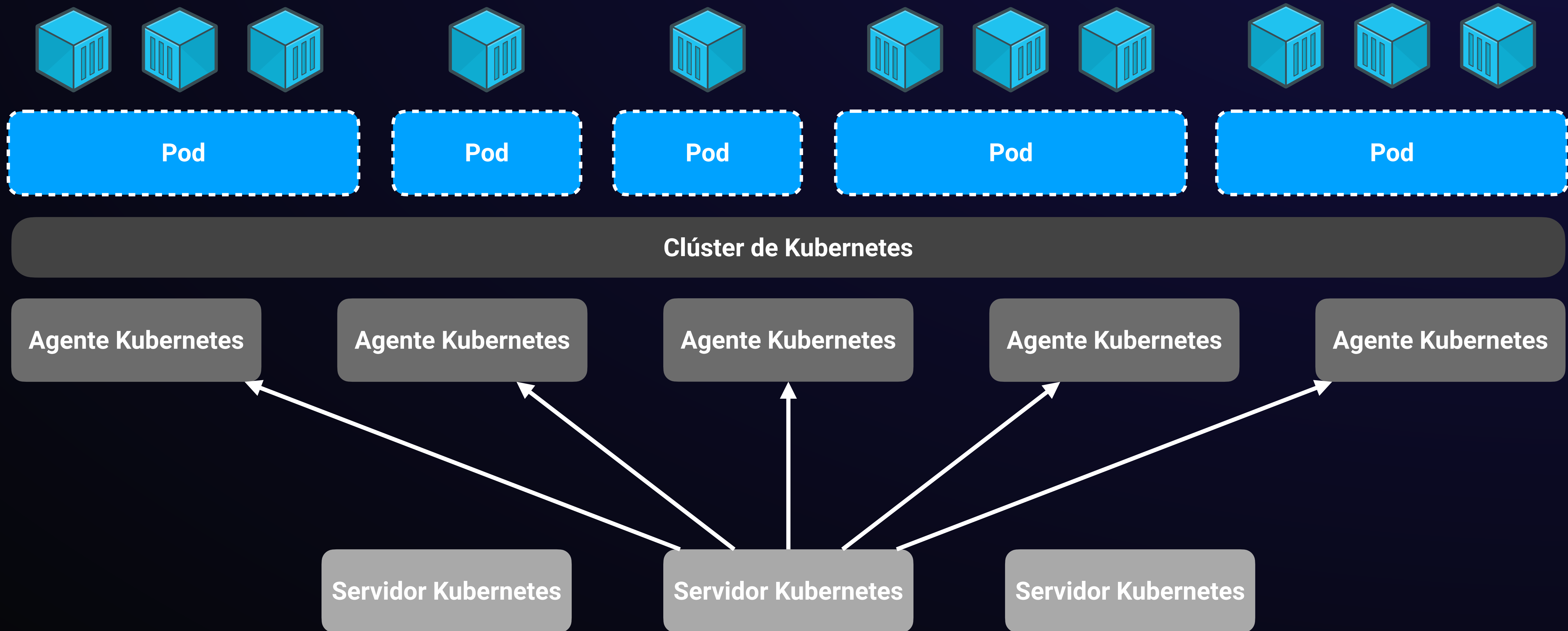
Apache
MESOSTM

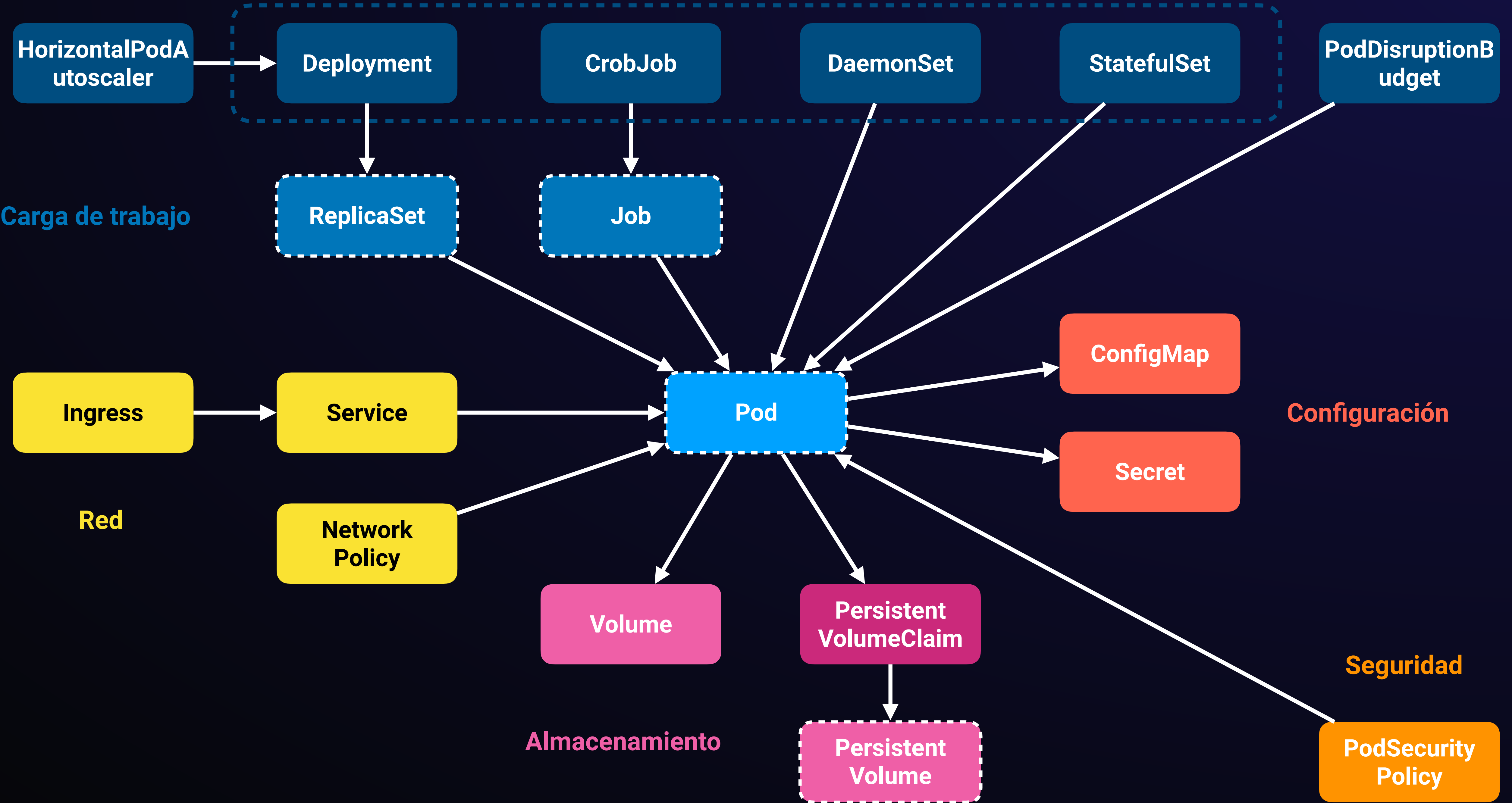


HashiCorp
Nomad



Kubernetes: arquitectura





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**.

```
kind: string
apiVersion: string

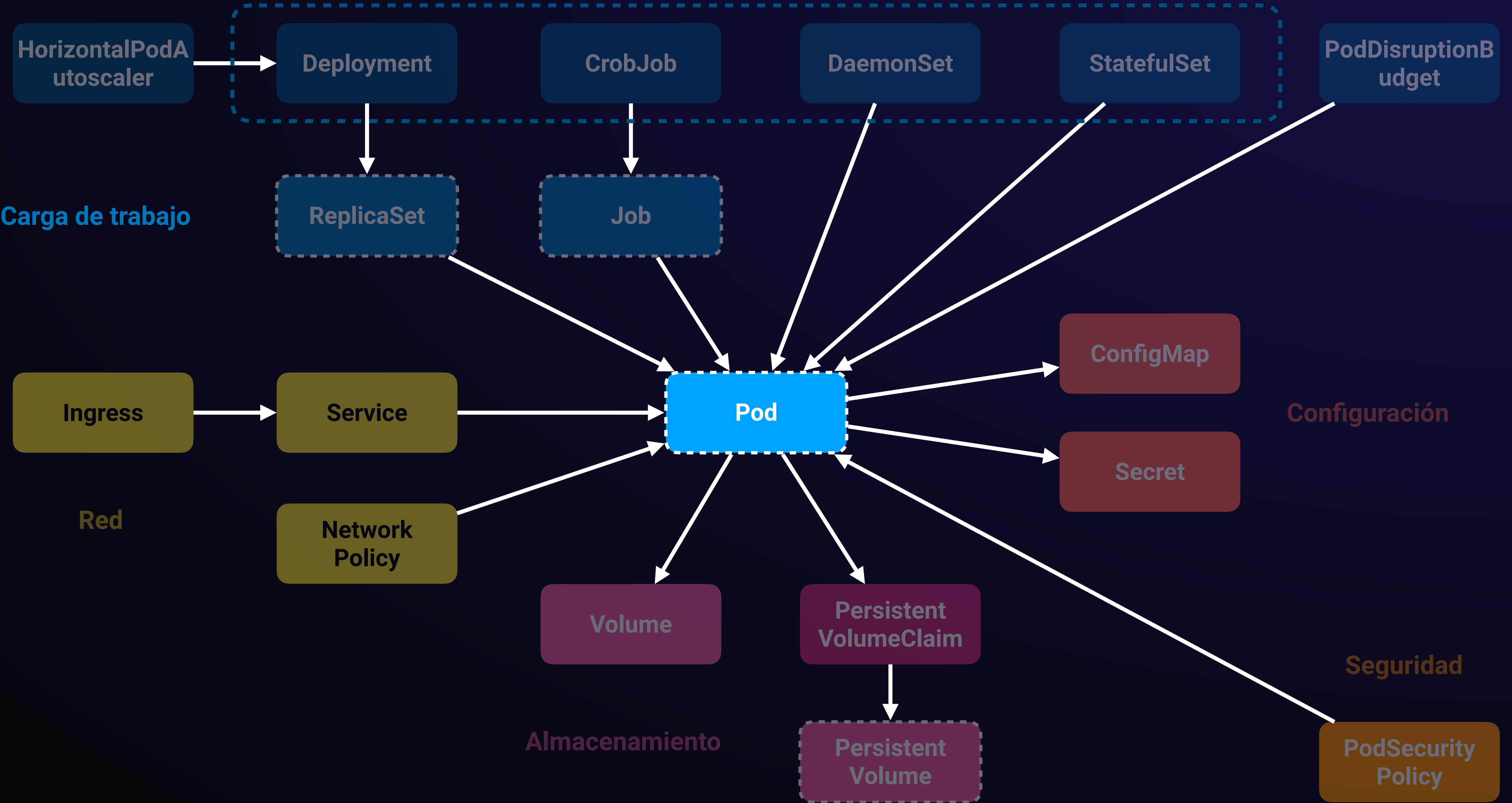
metadata:
  name: string
  namespace: string
  labels: {}
  annotations: {}

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**.

```
metadata:  
  name: superapi-proxy  
  namespace: backend  
  labels:  
    app.kubernetes.io/instance: superapi  
    app.kubernetes.io/component: proxy  
  annotations:  
    kubernetes.io/change-cause: 'Release 1.1.0'  
    prometheus.io/scrape: 'true'
```



Objetos:

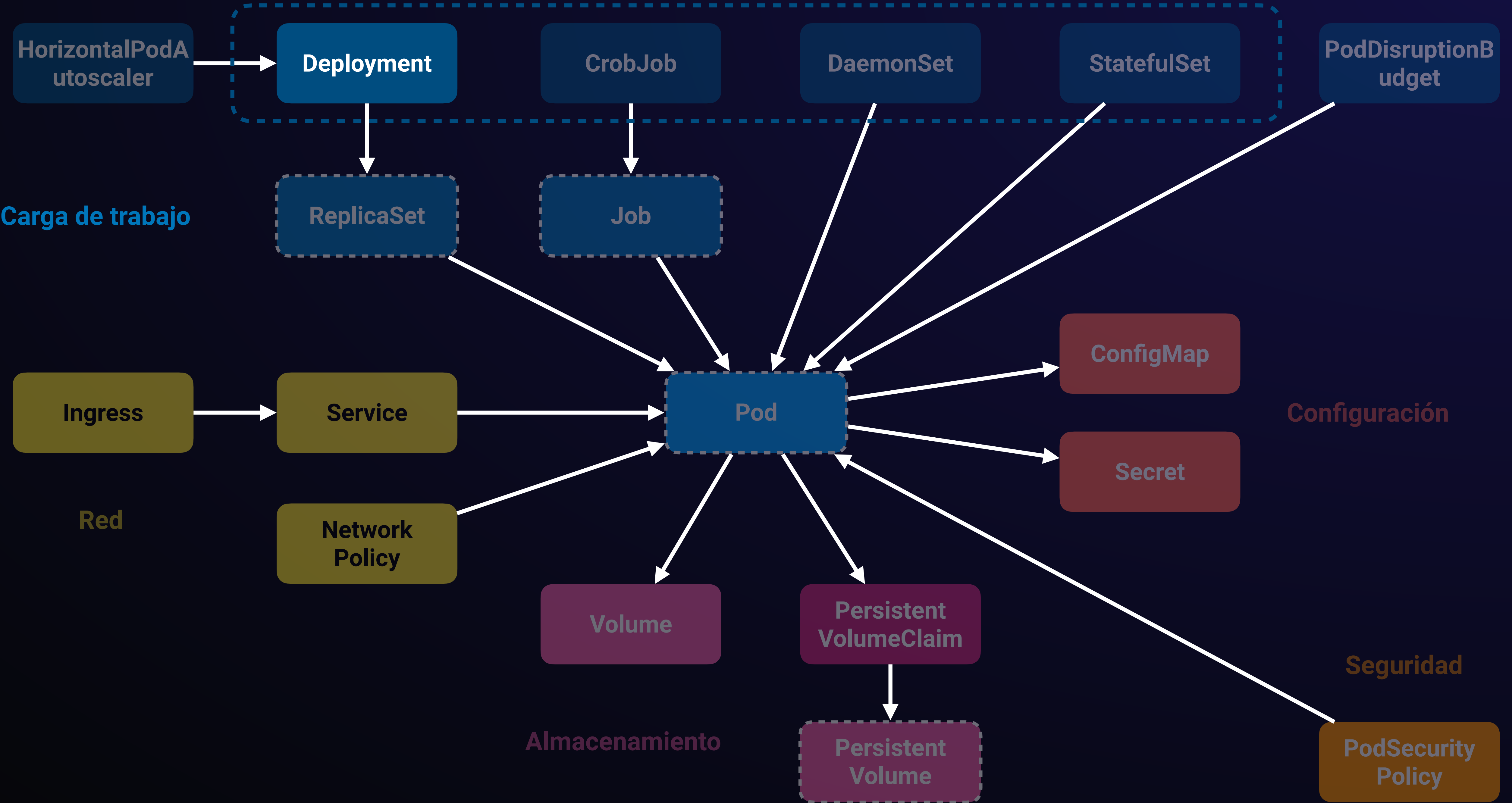
Pod

- 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**.

```
kind: Pod
apiVersion: v1

metadata:
  name: hello
  namespace: backend
  labels:
    app.kubernetes.io/instance: hello

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

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

```
kind: Deployment
apiVersion: apps/v1

metadata:
  name: website
  namespace: frontend
  labels:
    app.kubernetes.io/instance: website

spec:
  replicas: 2
  revisionHistoryLimit: 4

  strategy:
    rollingUpdate:
      maxUnavailable: 1

  selector:
    matchLabels:
      app.kubernetes.io/instance: website
```

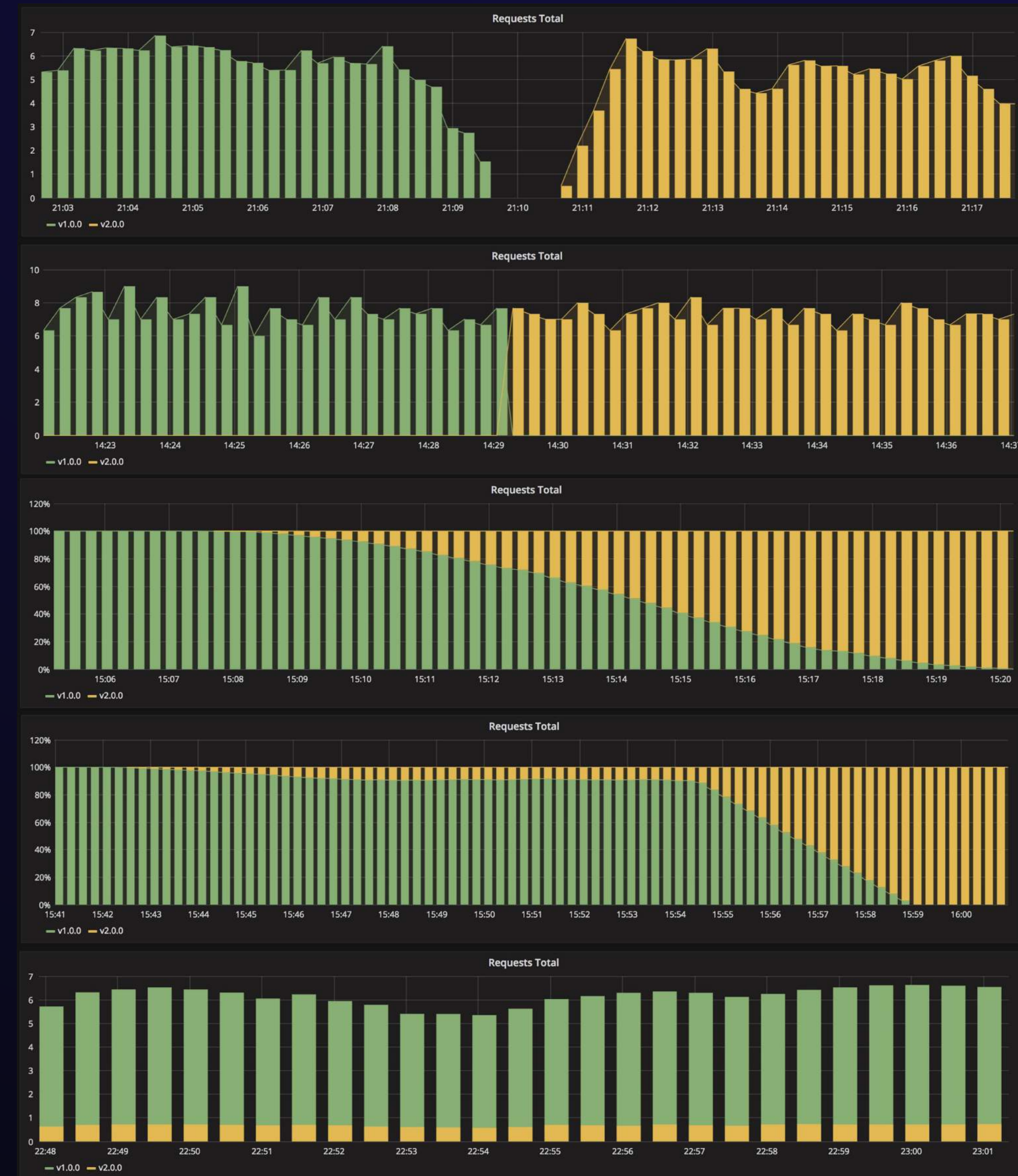
Pod

```
template:
  metadata:
    labels:
      app.kubernetes.io/instance: website
  spec:
    containers:
      - name: nginx
        image: nginx:1.18.0-alpine
        imagePullPolicy: Always
        ports:
          - name: http
            containerPort: 3000
        lifecycle:
          preStop:
            exec:
              command:
                - nginx
                - '-s'
                - quit
```

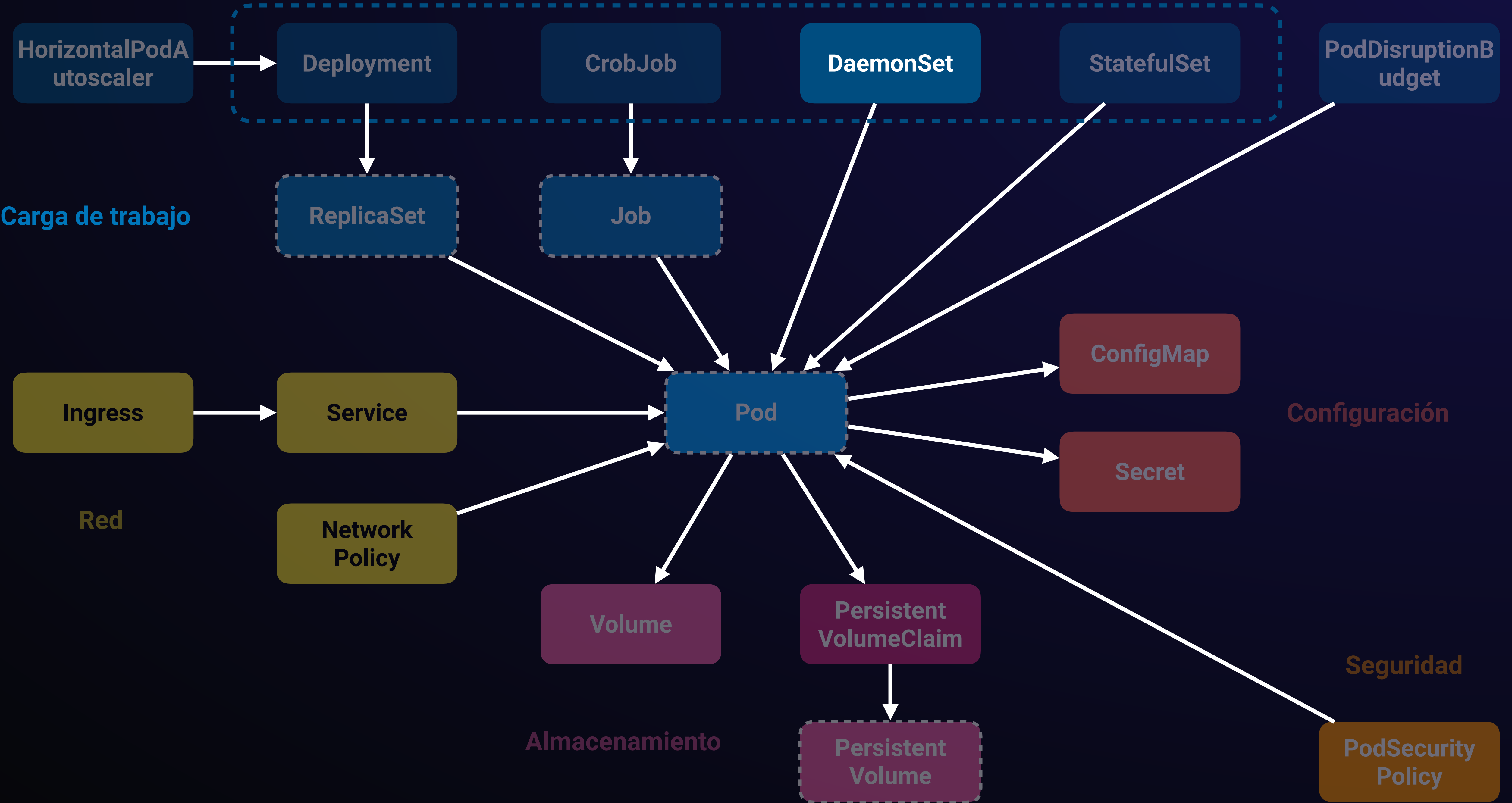
```
resources:
  requests:
    cpu: 10m
    memory: 20Mi
  limits:
    cpu: 20m
    memory: 30Mi
livenessProbe:
  tcpSocket:
    port: http
  initialDelaySeconds: 5
  timeoutSeconds: 2
```

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](#)



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

```
kind: DaemonSet
apiVersion: apps/v1

metadata:
  name: node-exporter
  namespace: monitoring
  labels:
    app.kubernetes.io/instance: node-exporter

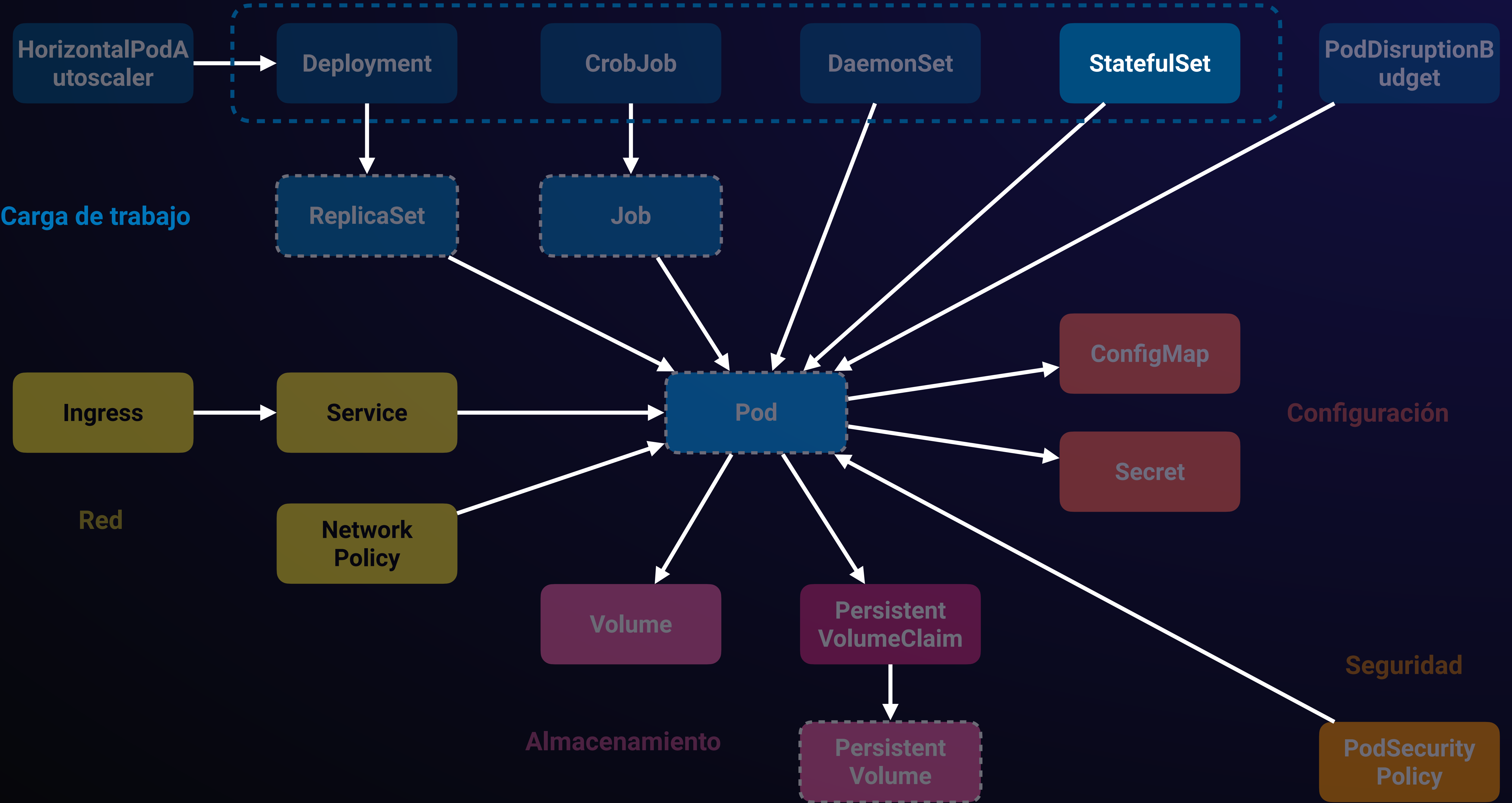
spec:
  selector:
    matchLabels:
      app.kubernetes.io/instance: node-exporter

  template:
    metadata:
      labels:
        app.kubernetes.io/instance: node-exporter
```

Pod

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

  containers:
    - name: default
      image: prom/node-exporter:v1.1.2
      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
```

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

```
kind: StatefulSet
apiVersion: apps/v1

metadata:
  name: mariadb
  namespace: persistence
  labels:
    app.kubernetes.io/instance: mariadb

spec:
  replicas: 1
  serviceName: mariadb

  updateStrategy:
    type: RollingUpdate

  selector:
    matchLabels:
      app.kubernetes.io/instance: mariadb

  template:
    metadata:
      labels:
        app.kubernetes.io/instance: mariadb

    spec:
      terminationGracePeriodSeconds: 300
```

Pod

```
containers:
  - name: default
    image: mariadb:10.5.9
    volumeMounts:
      - name: data
        mountPath: /var/lib/mysql
      - name: config
        mountPath: /etc/mysql/conf.d/
        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
```

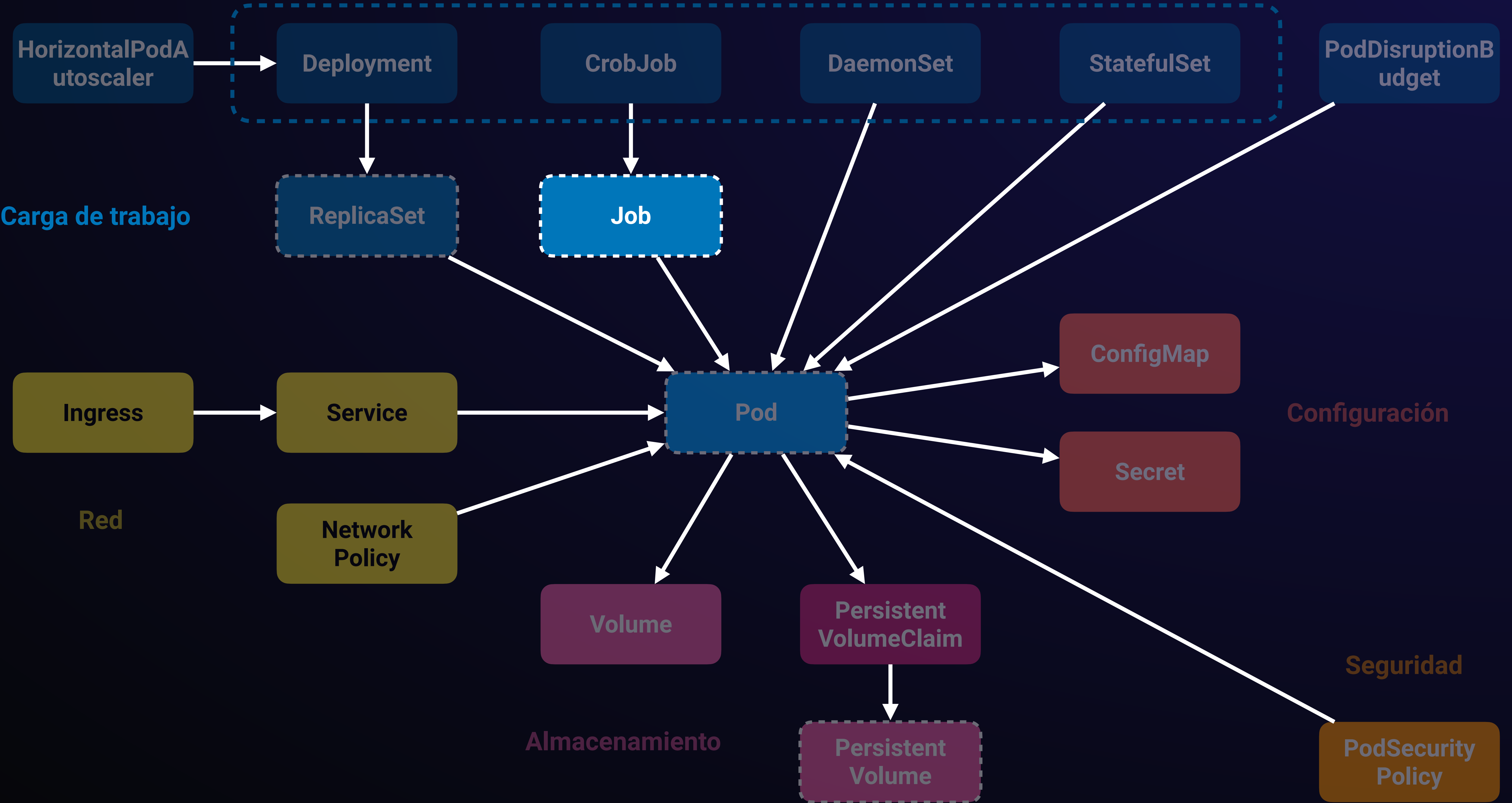
```
readinessProbe:
  exec:
    command: ['sh', '-c', 'exec
mysqladmin status -uroot
-p$MYSQL_ROOT_PASSWORD']
  initialDelaySeconds: 15
  securityContext:
    runAsUser: 999

volumes:
  - name: config
    configMap:
      name: mariadb

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

Volume

Persistent
VolumeClaim



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**.

```
kind: Job
apiVersion: batch/v1

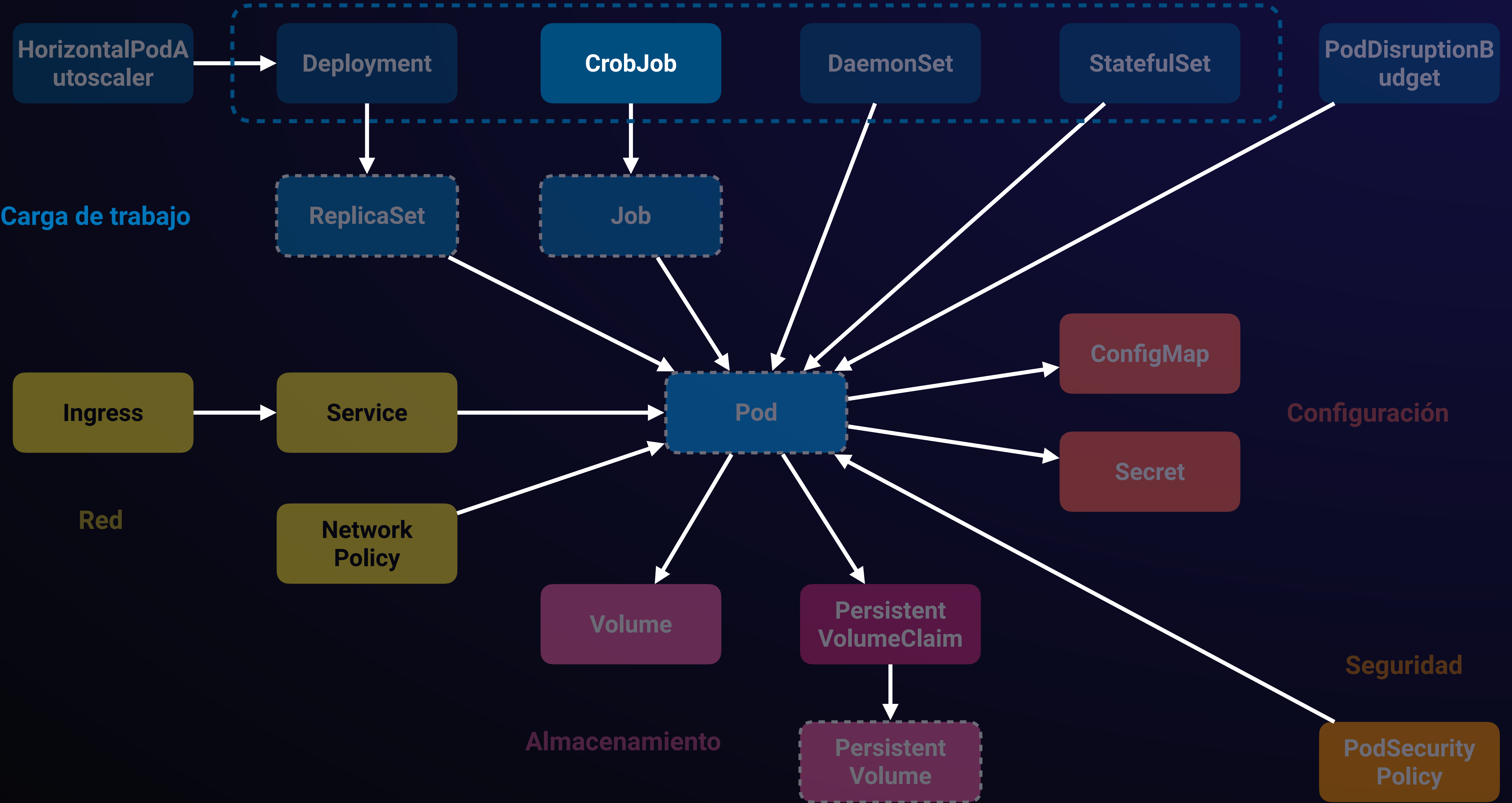
metadata:
  name: pi

spec:
  backoffLimit: 4

  template:
    spec:
      restartPolicy: Never

      containers:
        - name: default
          image: perl
          command:
            - perl
            - -Mbignum=bpi
            - -wle
```

Pod



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** por la sobrecarga que supone el arranque.

```
kind: CronJob
apiVersion: batch/v1beta1

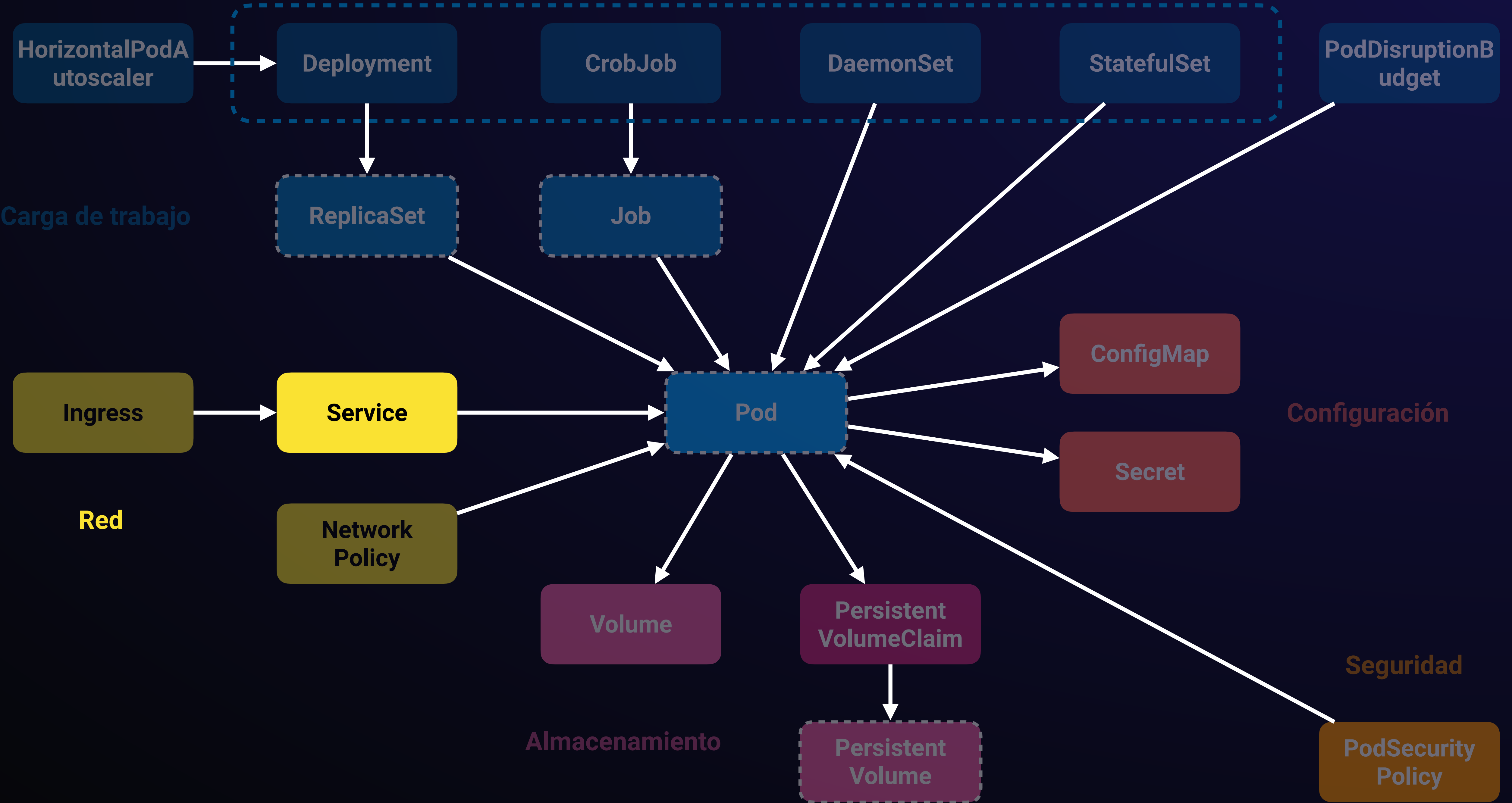
metadata:
  name: hello

spec:
  schedule: '*/1 * * * *'
  jobTemplate:
    spec:
      template:
        spec:
          restartPolicy: OnFailure

          containers:
            - name: default
              image: busybox
              args:
                - /bin/sh
                - -c
                - echo 'Hello from Kubernetes'
```

Job

Pod



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

```
kind: Service
apiVersion: v1

metadata:
  name: website
  namespace: frontend
  labels:
    app.kubernetes.io/instance: website

spec:
  selector:
    app.kubernetes.io/instance: website

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

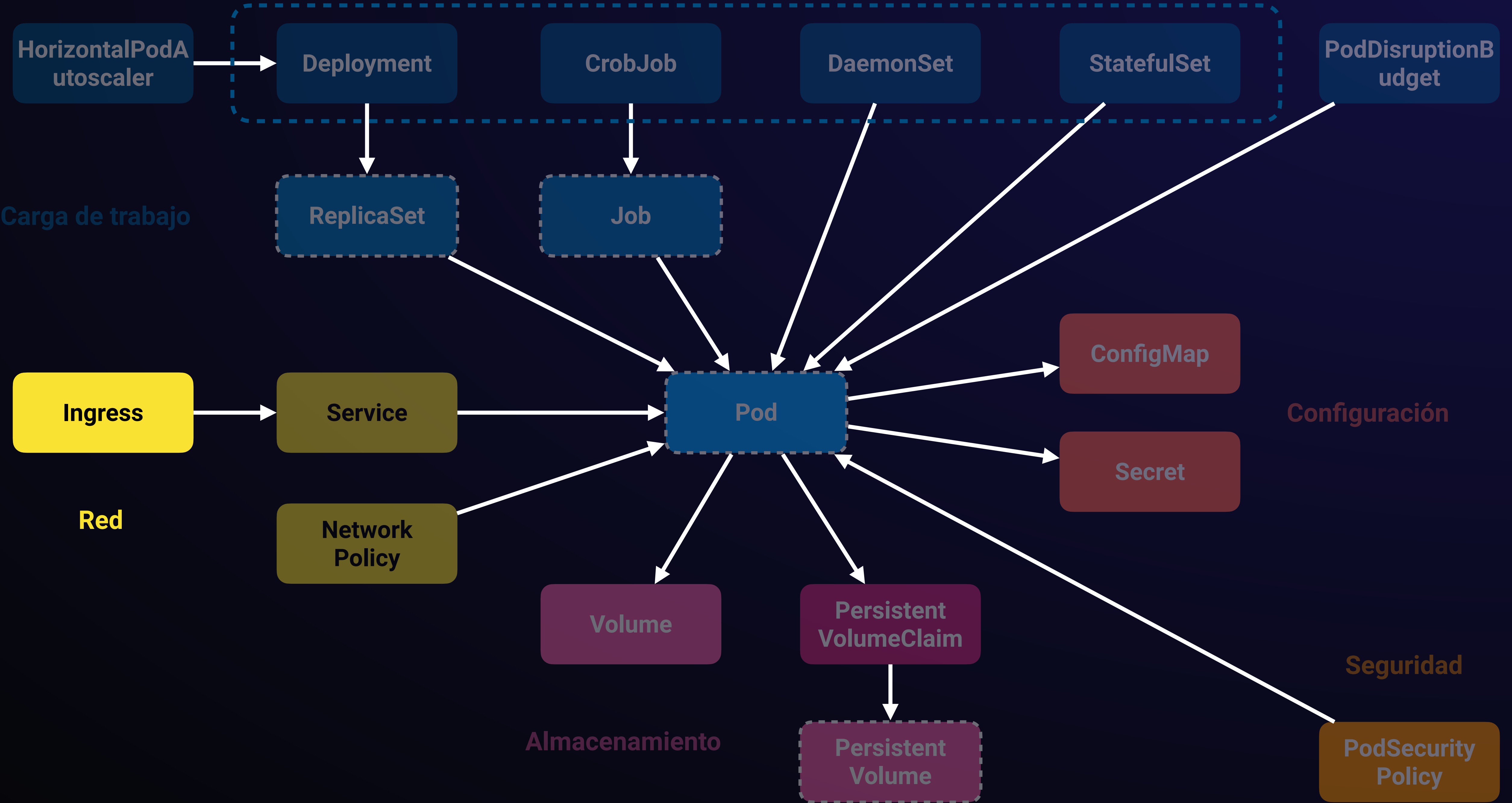
```
kind: Service
apiVersion: v1

metadata:
  name: mariadb
  namespace: persistence
  labels:
    app.kubernetes.io/instance: mariadb
  annotations:
    prometheus.io/scrape: 'true'

spec:
  selector:
    app.kubernetes.io/instance: mariadb

  clusterIP: None

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



Objetos:

Ingress

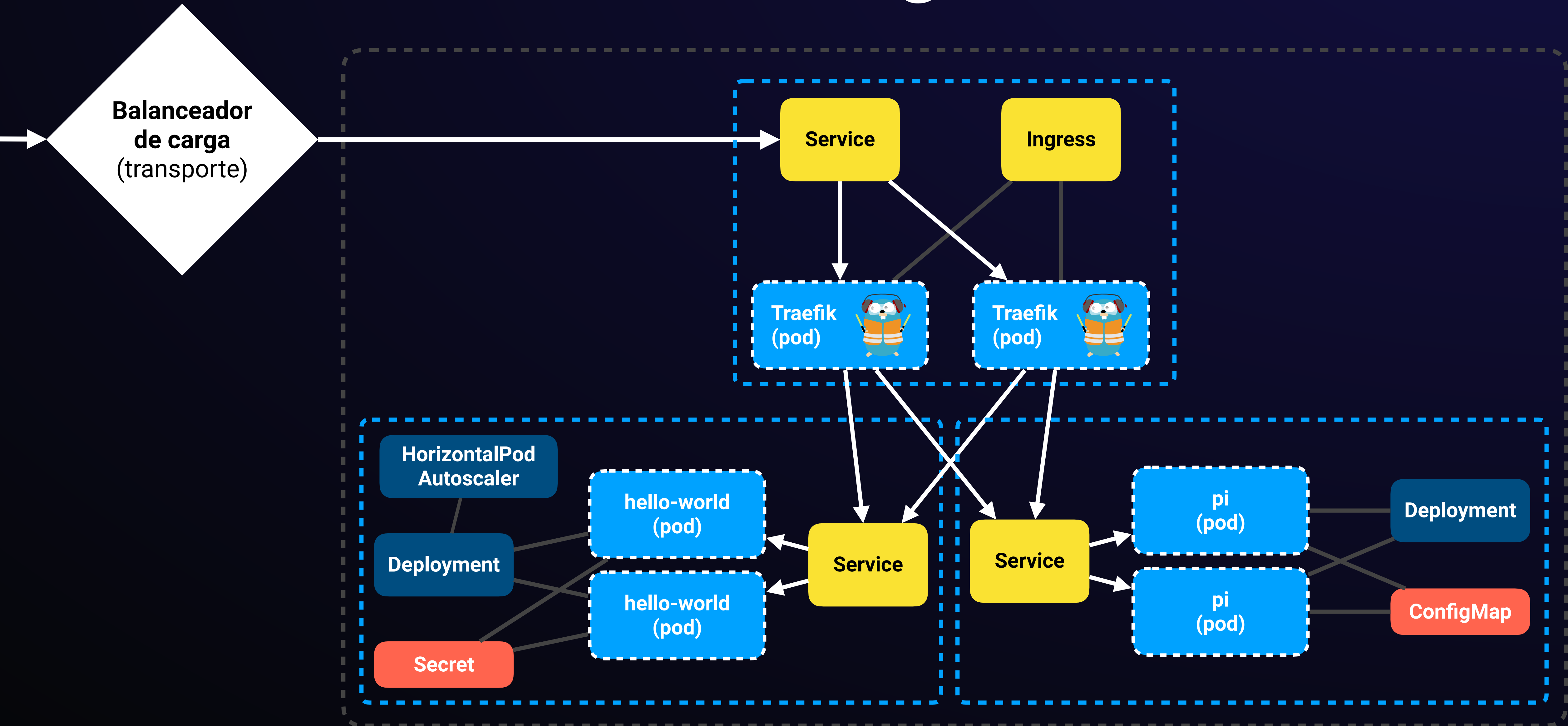
- Es una extensión que **nace** por la necesidad de **ahorrar costes en balanceadores** de carga.
- Es un **enrutador de alto nivel** (capa de aplicación).
- **Necesita un controlador** (un proxy inverso) que no viene por defecto (Traefik o Nginx, por ejemplo).
- **Requiere un único balanceador de carga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

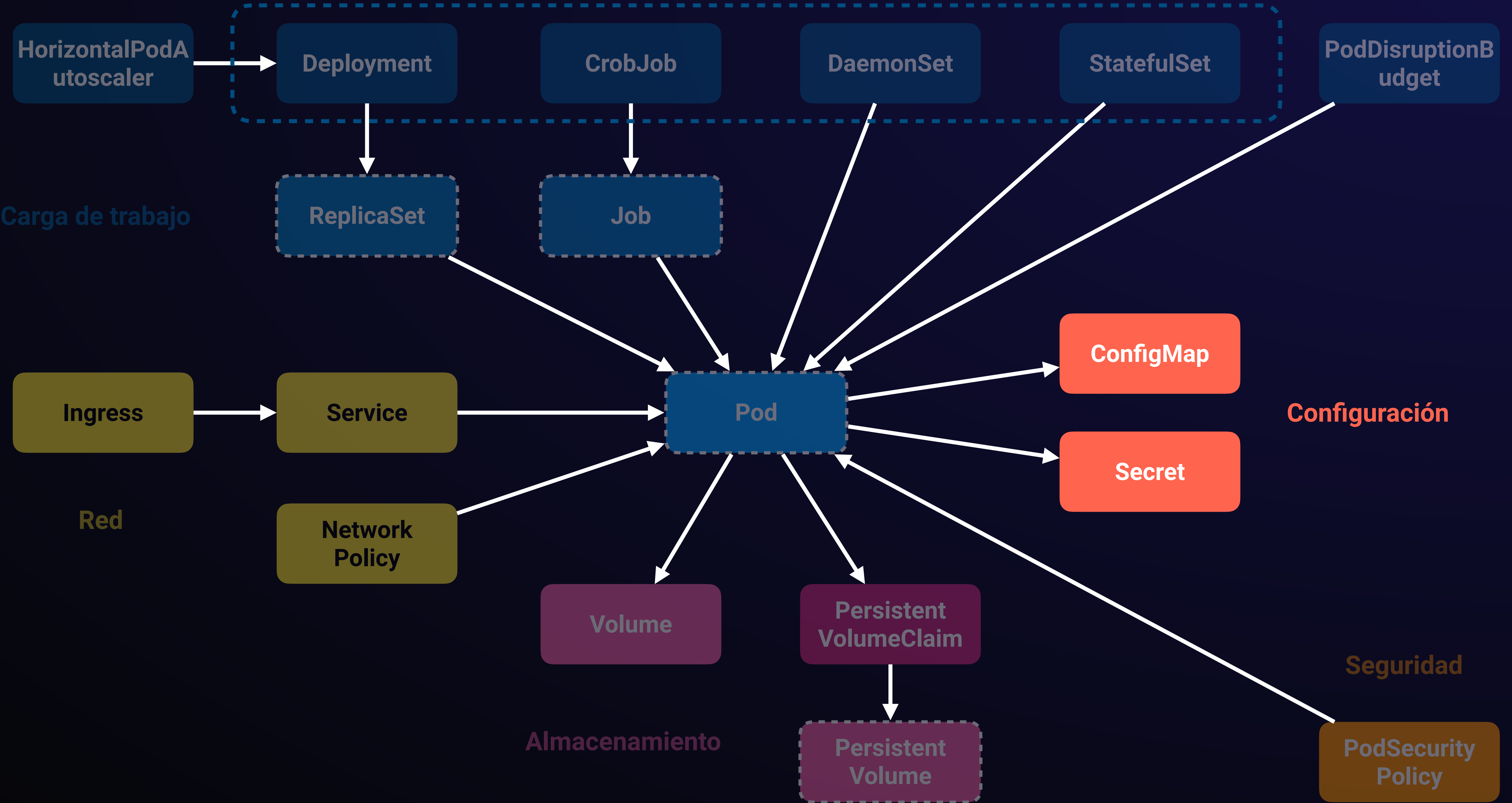
```
kind: Ingress
apiVersion: networking.k8s.io/v1

metadata:
  name: landing
  namespace: frontend
  labels:
    app.kubernetes.io/instance: landing
  annotations:
    traefik.ingress.kubernetes.io/router.entrypoints: h2

spec:
  rules:
    - host: brutal.systems
      http:
        paths:
          - backend:
              serviceName: landing
              servicePort: http
```


Services e ingresses





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

```
kind: ConfigMap
apiVersion: v1

metadata:
  name: superapi
  namespace: backend
  labels:
    app.kubernetes.io/instance: superapi

data:
  APP_ENV: production
  APP_URL: https://superapi.example.com
  DB_HOST: mariadb-0.vlc1.example.net
  DB_DATABASE: superapi
  DB_USERNAME: superapi
  REDIS_HOST: redis-0.vlc1.example.net
```

```
kind: Secret
apiVersion: v1

metadata:
  name: superapi
  namespace: backend
  labels:
    app.kubernetes.io/instance: superapi

type: Opaque

stringData:
  APP_KEY: v3ryS3cr3tK3y
  DB_PASSWORD: dbP4ssw0rd
  AWS_SECRET_ACCESS_KEY: 4wsS3cr3t
  RECAPTCHA_SECRET_KEY: r3c4ptch4S3cr3t
  STRIPE_SECRET: str1p3S3cr3t
  STRIPE_ENDPOINT_SECRET: an0th3r0n3
```


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.



cert-manager

- Es un controlador nativo para Kubernetes que se encarga de la **gestión de certificados X.509**.
- Utiliza el **protocolo ACME** para emitir y renovar certificados automáticamente.
- Hace uso de las definiciones personalizadas de recursos de Kubernetes (**CRD**).
- Guarda las **claves privadas en secretos** que otro software, normalmente el *ingress controller*, puede consumir.
- Se integra con **proveedores de DNS** para las **verificaciones**.



Traefik Proxy

- Es un ***ingress controller***, es decir, el ***proxy inverso*** que recibe y enruta las peticiones HTTP que llegan a un *cluster*.
- **Integración nativa** con Kubernetes: lee los objetos ***ingress*** y enruta hacia los servicios según su especificación.
- Trabaja en las **capas de aplicación y de transporte**.
- Es el **punto de terminación TLS**. Usa los certificados que *cert-manager* deja en los secretos.
- Hay otras opciones, como Nginx, HAProxy, Istio o Envoy.





Drone

- Es un **software de integración continua** que se puede desplegar en Kubernetes. También puede ejecutar las *pipelines* en el *cluster*.
- **Basado en contenedores.** Cada paso de la *pipeline* son una serie de comandos ejecutados en un contenedor.
- Las *pipelines* se definen en **YAML** y se guardan en el repositorio de software.
- Únicamente requiere un servidor externo con Docker Engine para la construcción de imágenes.






Drone



 Search repositories or jump to ...  0











Repositories → myorg/example-project → #18

example-project

[VIEW SOURCE](#)  [RESTART](#)  [DEPLOY](#) 

← ACTIVITY FEED

 #18. Fix K8s network policy.
 v0ctor pushed `823edeaf` to `develop` 01:48 · 5 months ago

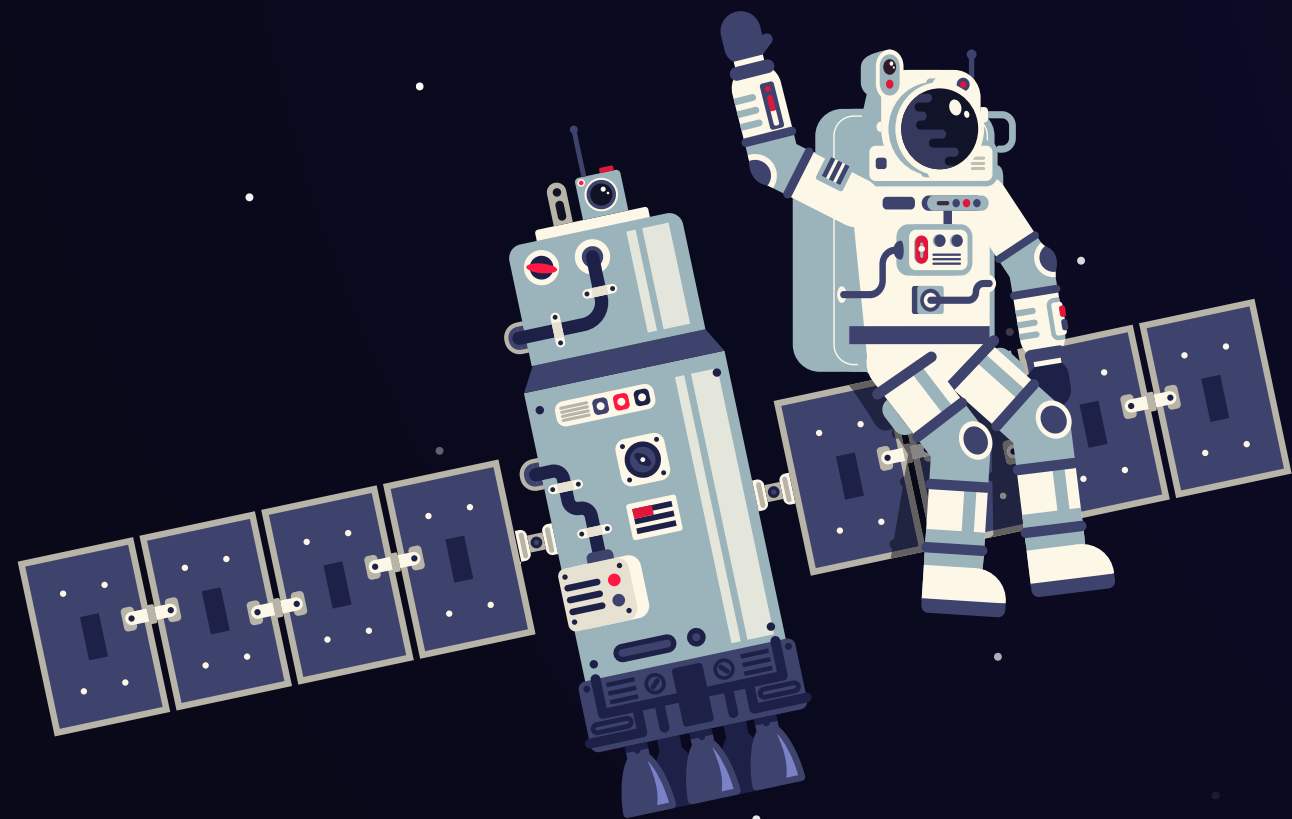
 default	01:48
 clone	00:04
 rdb	01:42
 prepare	00:06
 pre-build	00:49
 analyze	00:05
 test	00:35
 build	00:05
 pre-deploy	00:03
 deploy	00:04

default – deploy 00:04

```
1 + mkdir -p ~/.kube && echo "$KUBERNETES_CONFIG" > ~/.kube/config
2 + kubectl config use-context $(cat .drone/environment) && kubectl apply -f spec.yaml
3 Switched to context "staging".
4
5 networkpolicy.networking.k8s.io/passport-go created
6 poddisruptionbudget.policy/passport-go unchanged
7 secret/registry configured
8 configmap/passport-go configured
9 service/passport-go unchanged
10
11 deployment.apps/passport-go configured
12 horizontalpodautoscaler.autoscaling/passport-go unchanged
13 Warning: networking.k8s.io/v1beta1 Ingress is deprecated in v1.19+, unavailable in v1.22+; use networking.k8s.io/v1 Ingress
14 ingress.networking.k8s.io/passport-go configured
15 certificate.cert-manager.io/default unchanged
```

Conclusiones

- La **infraestructura como código** (IaC) permite tratar la infraestructura como una parte más del software.
- **Kubernetes y su ecosistema** proporcionan herramientas de alta calidad, con licencia libre y de código abierto para implementar IaC.
- Las herramientas presentadas permiten ofrecer servicios **altamente disponibles, resilientes y escalables**.
- Gran parte de los servicios de las organizaciones puede ser desplegado en Kubernetes, lo que supone un **ahorro de costes** muy significativo al reservar solo los recursos que se necesitan.
- La **automatización** que aportan estas prácticas proporciona un ahorro de tiempo que se puede **reinvertir** en más **innovación**.



brutal.systems

