Capacitación Kubernetes

Servicio de capacitación y entrenamiento ARIU

Clase 3
Objetos

Agenda

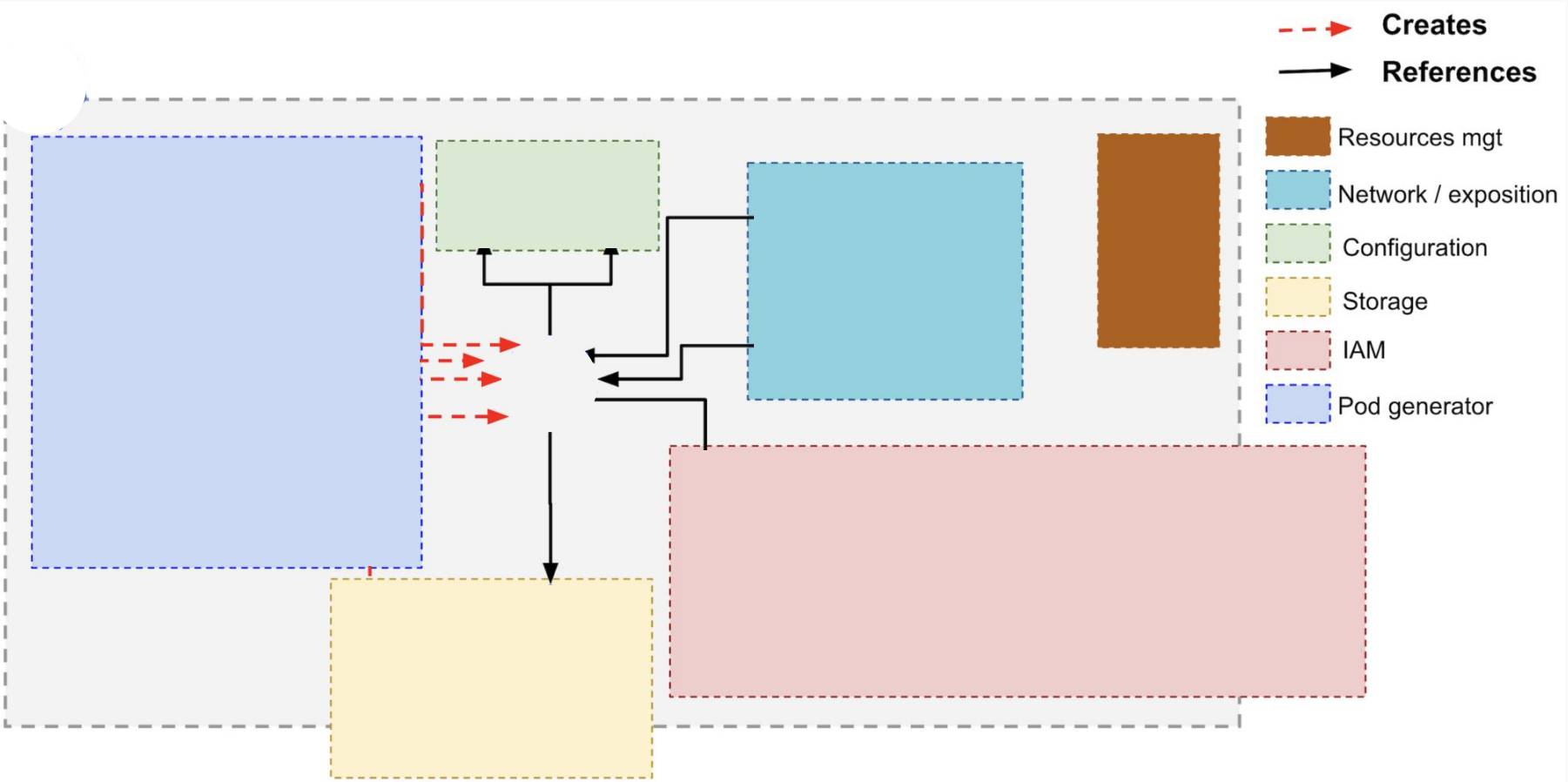
- **Que es un objeto en kubernetes**
- Sintaxis YAML
- Objetos Core
- Objetos de Red
- Objetos de Control

Qué es un Objeto

- Un objeto de Kubernetes es un "registro de intención".
- Una vez que has creado el objeto, el sistema de Kubernetes se pondrá en marcha para asegurar que el objeto existe.
- Los Objetos son representaciones de la configuración del clúster y sus aplicaciones.
- Se pueden describir en manifiestos declarativos en YAML o JSON

Para la explicación posterior los agrupamos según su funcionalidad:

Grupos de Objetos



Agenda

- Que es un objeto en kubernetes
- **Sintaxis YAML**
- Objetos Core
- Objetos de Red
- Objetos de Control

Sintaxis YAML

- YAML (YAML Ain't Markup Language) es un formato de serialización de datos legible por humanos.
- Se utiliza en Kubernetes para definir la configuración de los objetos del clúster.
- Su estructura está basada en **indentación** y uso de caracteres especiales como ":" y "-".

```yaml
# Comentarios precedidos por "#"
nombre: Juan Perez  # Clave-Valor
edad: 30
hobbies:
  - Leer libros   # Listas
  - Correr
```

Sintaxis YAML

```yaml
# Anidación de estructuras de datos
equipo:
  lider:
    nombre: Pedro
    edad: 35
  miembros:
    - nombre: Carla
      edad: 28
    - nombre: Luis
      edad: 32
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mi-pod
spec:
  containers:
  - name: nginx-container
    image: nginx:latest
```

Agenda

- Que es un objeto en kubernetes
- Sintaxis YAML
- **Objetos Core**
- Objetos de Red
- Objetos de Control

¿Qué es un Namespace?

- Un mecanismo para organizar y separar recursos en un clúster de Kubernetes.
- Crea un ámbito virtual dentro del clúster.
- No todos los objetos están en un espacio de nombres

¿Por qué usar Namespaces?

- Agrupar y organizar recursos.
- Limitar el alcance de los recursos a usuarios o equipos.
- Evitar colisiones de nombres.
- Restringir el uso de recursos de computación

Namespace existente en la instalación de Kubernetes
default:

- Es el predeterminado para crear recursos si no se especifica uno.
- Se utiliza para aplicaciones y servicios que no requieren segregación específica.

kube-system:

- Contiene los recursos críticos del sistema de Kubernetes.
- Se encuentran los componentes principales del clúster, como kube-dns, kube-proxy
- Se recomienda evitar modificar o eliminar recursos en este Namespace.

Escenario posible de uso:

- Namespace: "**app-desarrollo**"
  - Equipo de desarrollo trabaja en este Namespace.
  - Recursos de prueba y desarrollo se crean aquí.
- Namespace: "**app-produccion**"
  - Aplicaciones en producción se despliegan aquí.
  - Acceso limitado a equipos autorizados.

- Revisar 01-namespace.yml
- Mostrar namespaces
  - kubectl get namespace
- Crear namespace
  - kubectl create namespace prueba
- Borrar namespaces
  - kubectl delete ns prueba
- Crear namespaces con manifiesto
  - kubectl apply -f 01-namespace.yml
- Borrar namespaces con manifiesto
  - kubectl delete -f 01-namespace.yml
- Crear nuevamente namespace con manifiesto
  - k apply -f 01-namespace.yml

¿Qué es un Pod?

- Un Pod es la unidad principal en Kubernetes.
- Representa un único proceso que se ejecuta en el clúster.

Características del Pod

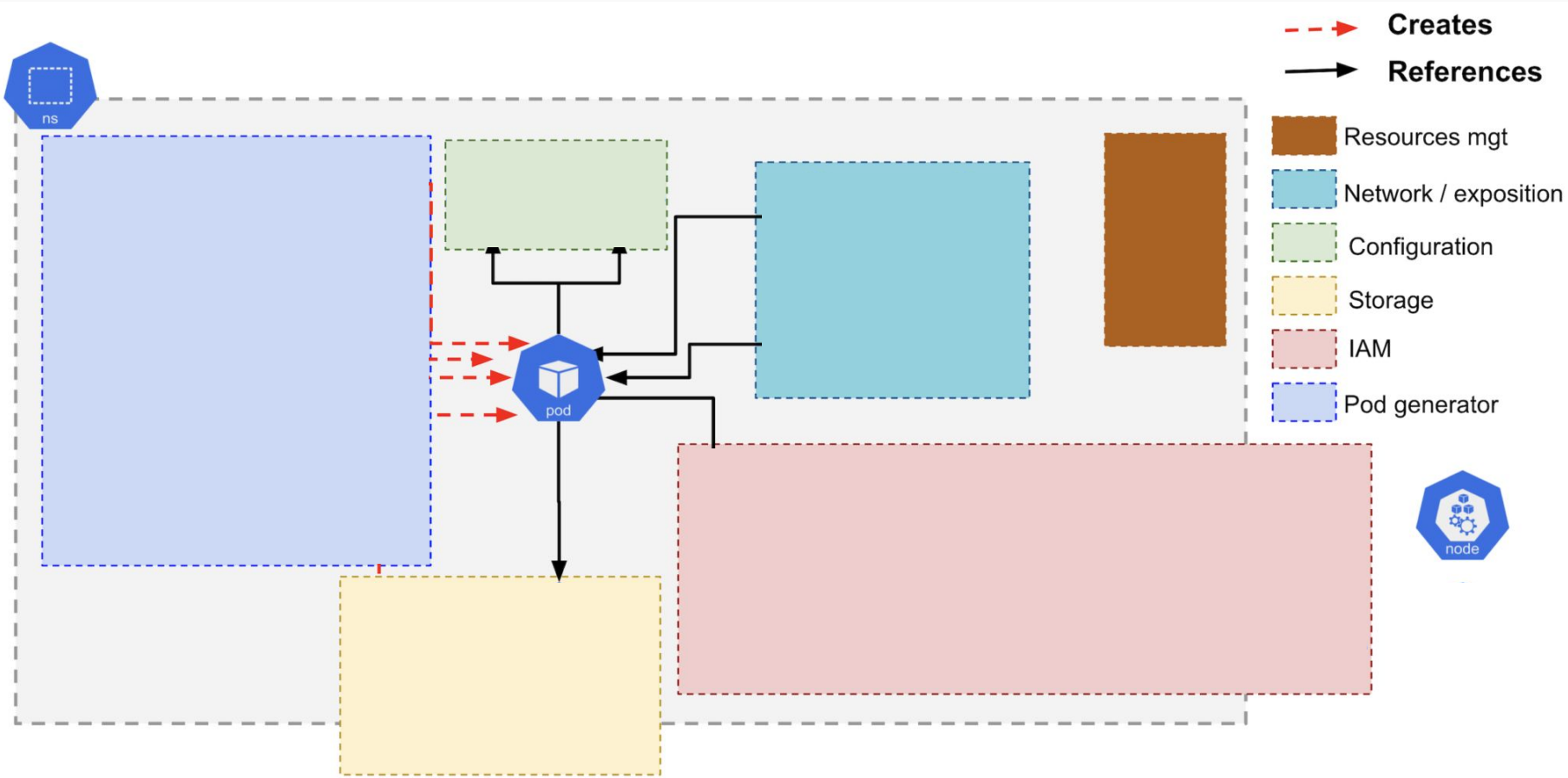
- Puede contener uno o más contenedores.
- Los contenedores comparten el mismo espacio de red y volúmenes.
- Posee una dirección IP única para facilitar la comunicación.

Buenas Prácticas

- Cada Pod debe tener una sola responsabilidad.
- Agrupa contenedores relacionados en un solo Pod.
- Usa un Pod por instancia de contenedor.
- No dependas de IPs fijas para los Pods, utiliza DNS en su lugar.
- Los Pods son efímeros y pueden ser recreados fácilmente por el sistema de Kubernetes en caso de fallas o actualizaciones. No se deben usar para almacenar datos persistentes.
- No se deberían crear Pods directamente, deberían usar siempre controladores incluso para Pods individuales

- Revisar 02-pod.yml
- Crear Pod con manifiesto
  - k apply -f 02-pod.yml -n prueba
- Mostrar Pod creado
  - k get pod -n prueba
- Mostrar descripción
  - k describe pod nginx -n prueba
- Borrar pod
  - k delete pod nginx -n prueba
- Usar plugin ns
  - k ns prueba
- Crear nuevamente de manifiesto
  - k apply -f 02-pod.yml
- Ver logs de pod
  - k logs nginx --tail 100 -f
- Revisar en qué nodo fué creado
  - k get pod -o wide

Grupos de Objetos - Core



Agenda

- Que es un objeto en kubernetes
- Sintaxis YAML
- Objetos Core
- **Objetos de Red**
- Objetos de Control

¿Qué es un Service en Kubernetes?

- Un Service es un recurso abstracto que define una política de acceso a los pods.
- Proporciona una dirección IP y un nombre DNS único para acceder a los pods que pertenecen al Service.
- Permite la exposición de aplicaciones dentro o fuera del clúster de Kubernetes.

¿Cómo se usa un Service?

- Un Service utiliza etiquetas (labels) para seleccionar los pods a los que debe dirigirse.
- Evitar exponer todos los pods directamente. Solo los accesos válidos a la aplicación.
- Se puede definir diferentes tipos de Service según la forma de exposición: ClusterIP, NodePort, LoadBalancer, o ExternalName.

Tipos de servicios

Cada tipo de servicio tiene su propio propósito y casos de uso.

ClusterIP

- Expone el servicio en una dirección IP interna del clúster.
- Solo es accesible desde dentro del clúster y es el valor predeterminado si no se especifica un tipo al crear el servicio.
- Para acceder desde internet público, se puede utilizar un Ingress o un port-forward.

Casos de uso

- Ejemplo: Un servicio que aloja una base de datos MySQL
- Utilizar ClusterIP cuando desees exponer un servicio internamente, solo accesible desde dentro del clúster.
- Útil para comunicación entre microservicios en el mismo clúster.

Tipos de servicios

NodePort

- Expone el servicio en cada IP de nodo a través de un puerto estático (NodePort).
- El tráfico se reenvía desde ese puerto a través del servicio.
- Permite el acceso externo directo al servicio mediante la dirección IP del nodo y el puerto asignado.

Casos de uso

- Usar NodePort cuando necesites acceder a una aplicación desde fuera del clúster mediante un puerto estático en cada nodo.
- Puede ser útil para pruebas locales y desarrollo, pero no es la opción más segura para entornos de producción.

Tipos de servicios

LoadBalancer

- Provee el servicio externamente utilizando un balanceador de carga externo (por ejemplo, un Load Balancer en la nube).
- El clúster debe estar integrado con un proveedor de servicios en la nube o tener un balanceador de carga configurado.

Casos de uso

- Utiliza LoadBalancer para exponer aplicaciones que requieran alta disponibilidad y balanceo de carga.

Tipos de servicios

ExternalName

- Asigna un nombre DNS a un servicio externo fuera del clúster de Kubernetes.
- No involucra proxying, redirección o reenvío; simplemente se configura el DNS del clúster para devolver un registro CNAME con el valor del nombre de host externo.

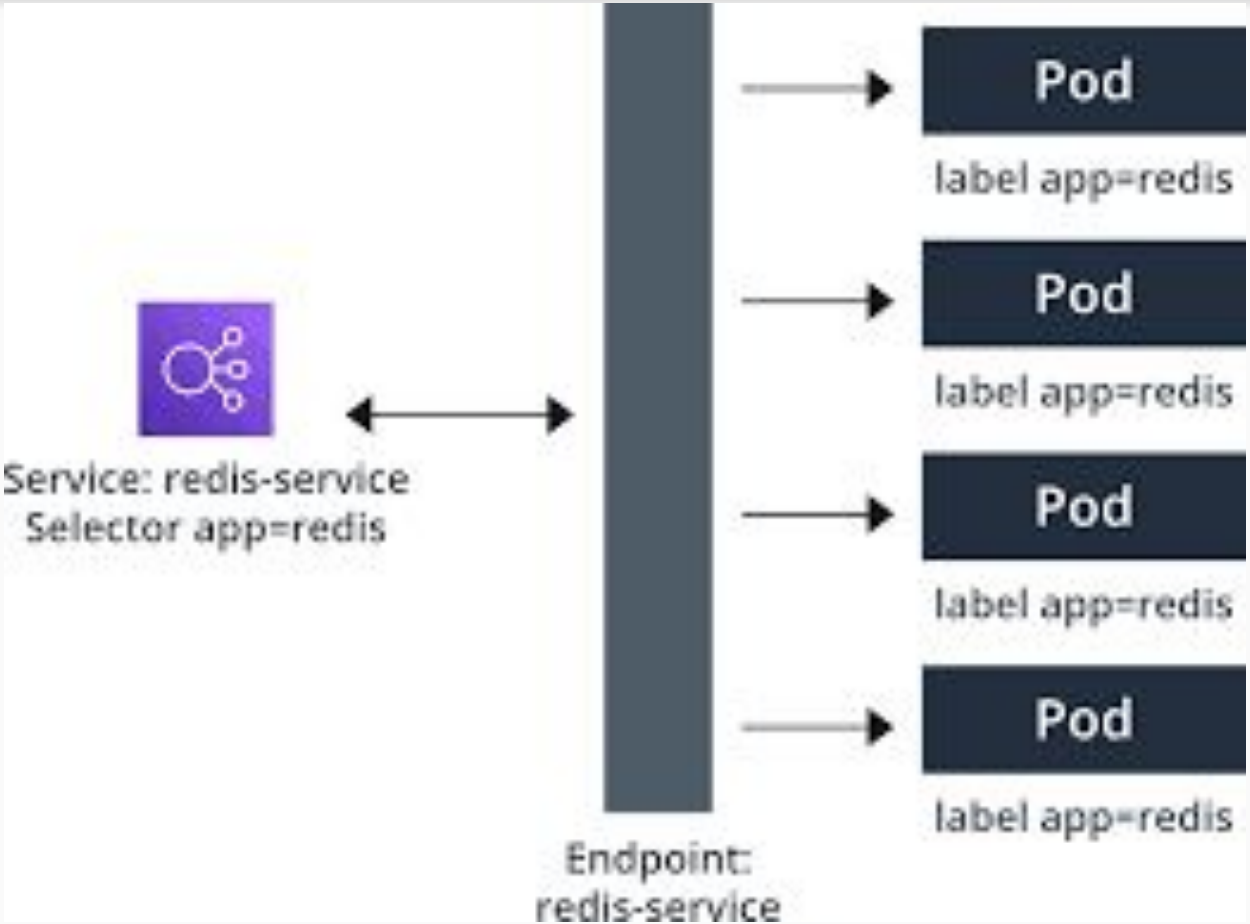
Casos de uso

- Ejemplo: Un servicio que apunta a una base de datos alojada en un servidor externo.
- Útil cuando se necesita acceder a servicios externos desde tus pods sin exponerlos internamente.

Qué es un Endpoint

- Los Endpoints son utilizados para acceder y gestionar múltiples instancias de un servicio sin preocuparse por las ubicaciones individuales de los Pods
- Los Endpoints se crean automáticamente cuando se crea un Servicio y existe un pod o más al cual enlazarse
- Los Endpoints direccionan el tráfico a los Pods correctos, independientemente de su ubicación en el clúster

Red - EndPoint



Dentro del servidor

- Probar curl a la ip de pod
  - k get pod -o wide
  - curl http://<ip-pod-nginx>
- Levantar servicio
  - k apply -f 03-services.yml
- Revisar EndPoint
  - k get endpoints
  - k describe endpoints nginx-html-ci
- Probar curl a la ip del servicio
  - k get services
  - k get svc
  - curl http://<ip-services>
  - curl http://localhost:30280
  - curl http://<ip-loadbalancer>

- Bajar pod
  - k delete pod nginx
  - k get svc
  - k get endpoints
- Probar curl a la ip del servicio
  - curl http://<ip-services>
- Levantar Pod
  - k apply -f 02-pod.yml

Fuera del servidor

- Probar acceso a la aplicación con un forward
  - k port-forward service/nginx-html-ci 5555:80
  - Probar en navegador http://localhost:5555
  - Probar en navegador http://ip-load-balancer

¿Qué es Ingress?

- Ingress es un recurso en Kubernetes que permite la exposición de servicios y aplicaciones hacia el mundo exterior.
- Actúa como una capa de entrada que administra el enrutamiento del tráfico entrante a los servicios backend dentro del clúster.
- Proporciona una forma centralizada y flexible de administrar las reglas de enrutamiento sin afectar directamente a los servicios individuales.

Propósito

- Evita exponer servicios directamente mediante NodePort o LoadBalancer puede ser limitante y complicado en entornos complejos.
- Centralizar logs de accesos al cluster en un único punto (controlador ingress)

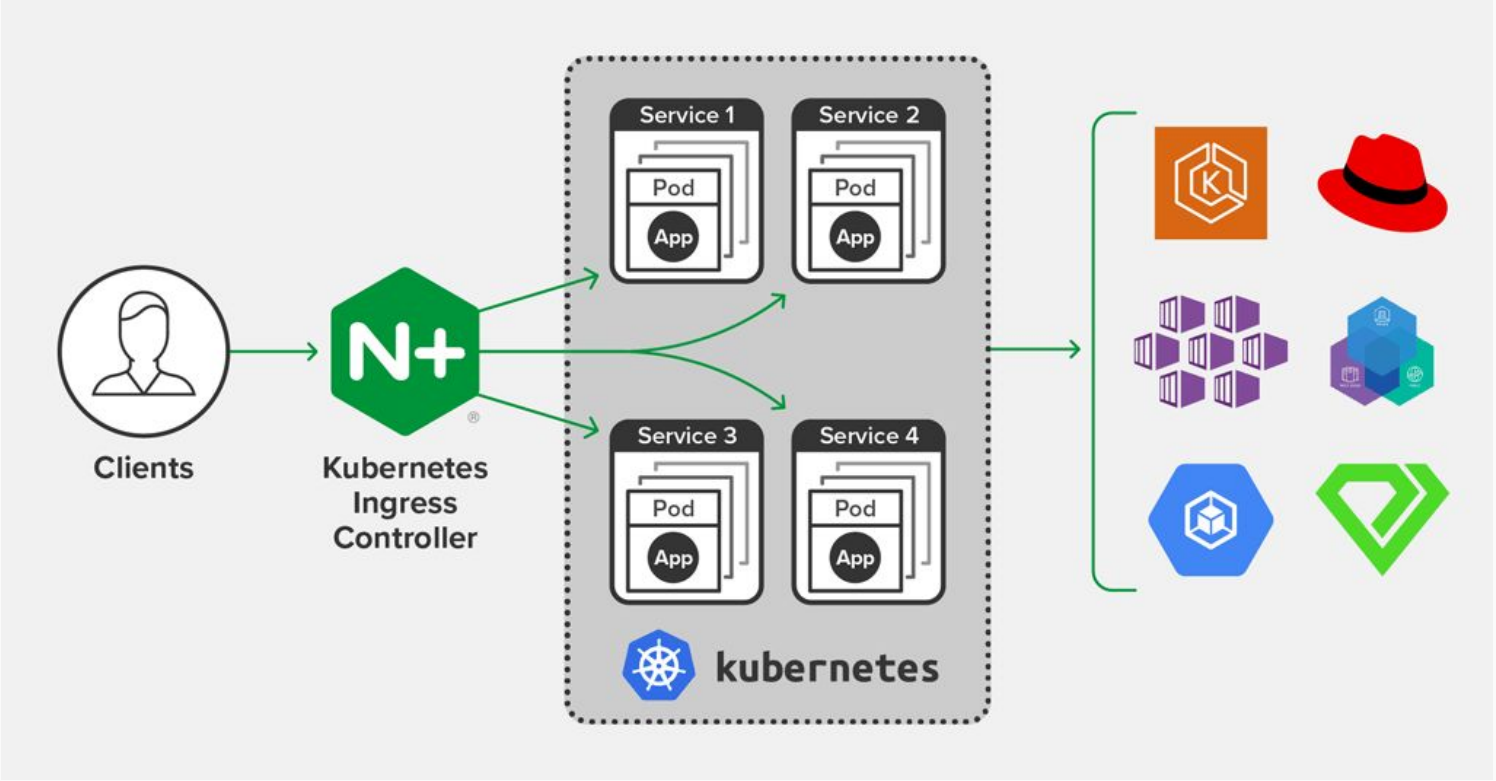
Cómo funciona

- Cuando una solicitud llega desde el exterior, Ingress procesa la solicitud.
- Mediante reglas de enrutamiento definidas, redirige la solicitud al servicio backend correspondiente basándose en el host o la ruta de la solicitud.

Ingress-nginx con Cert-manager

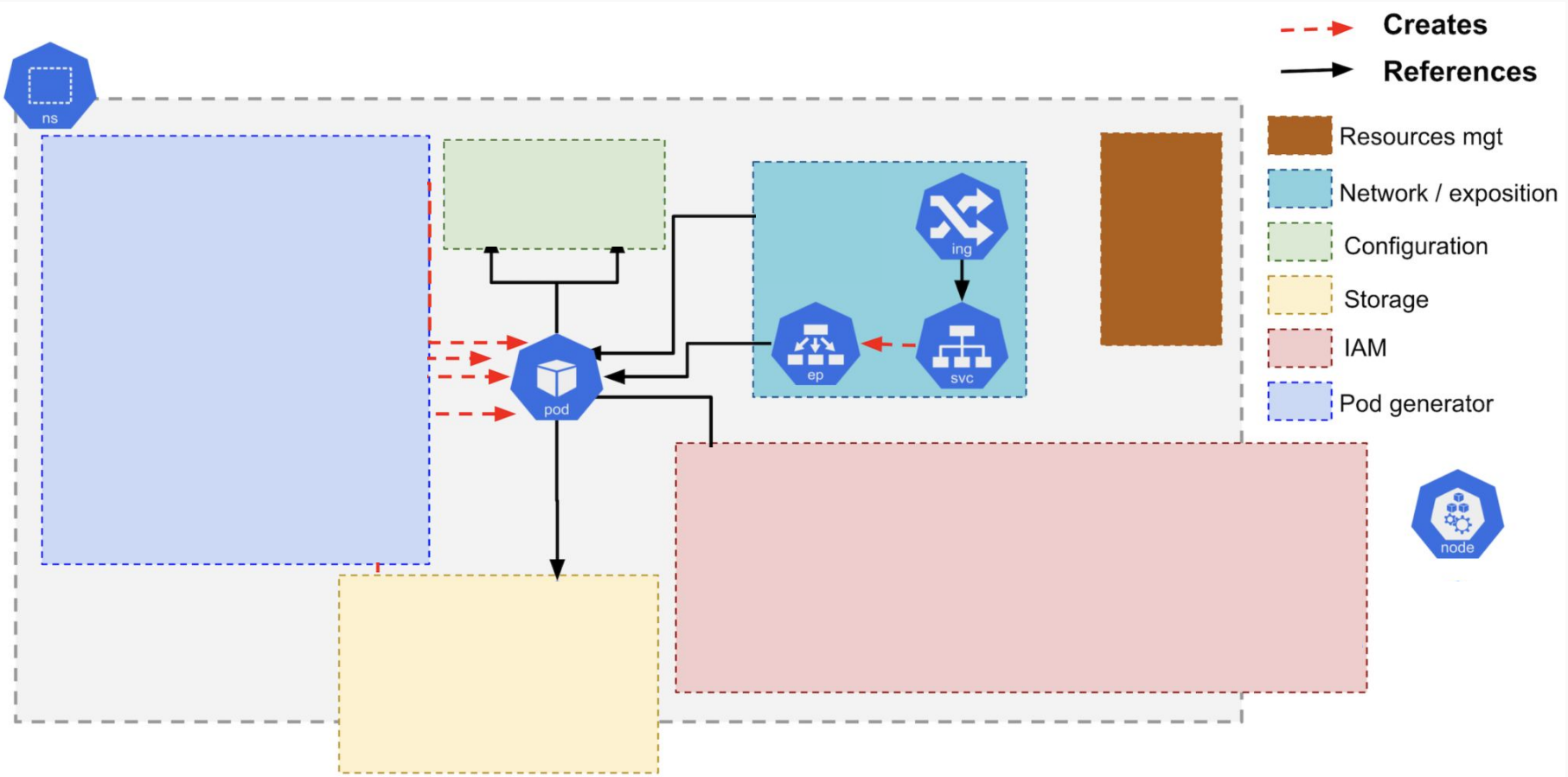
- Cert-Manager automatiza la gestión de certificados, lo que ahorra tiempo y esfuerzo en la emisión y renovación de certificados Let's Encrypt.
- Garantiza que las conexiones entre los clientes y el clúster sean seguras y cifradas.

Ingress


Clients
Kubernetes
Ingress
Controller
Service 1
Service 2
Service 3
Service 4
Pod
App
kubernetes

- Crear ingress
  - modificar 04-ingress.yml con url apuntada con *
  - k apply -f 04-ingress.yml
  - k get ingress
  - k get certificate
- Acceder a url mediante un navegador
- Bajar pod
  - k delete pod nginx
- Acceder a url mediante un navegador

Grupos de Objetos - Red



Agenda

- Que es un objeto en kubernetes
- Sintaxis YAML
- Objetos Core
- Objetos de Red
- **Objetos de Control**

Introducción

- Los objetos de control en Kubernetes son esenciales para la gestión y administración de aplicaciones en el clúster.
- Su enfoque declarativo y funciones de autocorrección aseguran la estabilidad, alta disponibilidad y escalabilidad de las aplicaciones.
- Facilitan la administración y operación de workloads en Kubernetes.

Tipos de controladores de recursos

- ReplicaSet (RS)
- Deployment
- StatefulSet (STS)
- DaemonSet (DS)

ReplicaSet

- Para mantener un número específico de réplicas de Pods.
- Reiniciar automáticamente Pods que fallan
- Utiliza etiquetas de selección para identificar los Pods controlados.

Funcionamiento

- Especificar el nombre, las réplicas deseadas y las plantillas de los Pods

- Aplicar manifiesto RS
  - k apply -f 05-replicaset.yml
- Revisar cantidad de Pod
  - k get pod
- Borrar Pod
  - k delete pod nginx-<id>
- Revisar cantidad de Pod
  - k get pod
- Actualizar y aplicar manifiestos (cantidad)
  - k edit rs nginx (cambiar a dos replicas)
  - k get pod

Deployment:

- Es un objeto de control de nivel superior que utiliza ReplicaSet en su implementación.
- Permite la gestión de actualizaciones y rollbacks sin tiempo de inactividad.
- Recomendado para la mayoría de los casos de despliegue y actualización de aplicaciones.
- Admite diferentes estrategias de actualización, como **RollingUpdate** y **Recreate**

RollingUpdate

- Esta estrategia es la predeterminada y es la más utilizada en la mayoría de los casos.
- Se crea una nueva réplica de la aplicación con la nueva versión y, a continuación, se elimina gradualmente una réplica antigua a la vez.
- La aplicación esté siempre disponible durante el proceso de actualización
- Minimiza el impacto en la disponibilidad.

Recreate

- Primero elimina todas las réplicas de la versión anterior de la aplicación antes de crear las nuevas réplicas con la versión actualizada.
- No hay un proceso gradual de actualización, la aplicación puede experimentar un tiempo de inactividad durante la actualización.

- Borrar ReplicaSet
  - k delete rs nginx
  - k get pod
- Aplicar deploy
  - k apply -f 06-deploy.yml
- Verificar ReplicaSet y Pod
  - k get rs
  - k get pod
- Cambiar con edit versión a 1.21
  - k edit deploy nginx
  - k get pod
- Ver el cambio en los Pod
  - k describe pod nginx-<id>

Statefulset

- Gestiona el despliegue y escalado de un conjunto de Pods.
- Garantiza el orden y unicidad de dichos Pods.
- Ideales para aplicaciones que requieren identidad única
- Usado cuando se requiere un despliegue y escalado ordenado.
- Los StatefulSets no proporcionan ninguna garantía de la terminación de los pods cuando se elimina un StatefulSet

- Crear Statefulset
  - k apply -f 07-statefulset.yml
- Verificar
  - k get pods
  - k get statefulset
  - k get svc
  - k describe svc
- Probar eliminar las dos réplicas
  - k delete pod postgresql-db-0 postgresql-db-1; k get pod -w

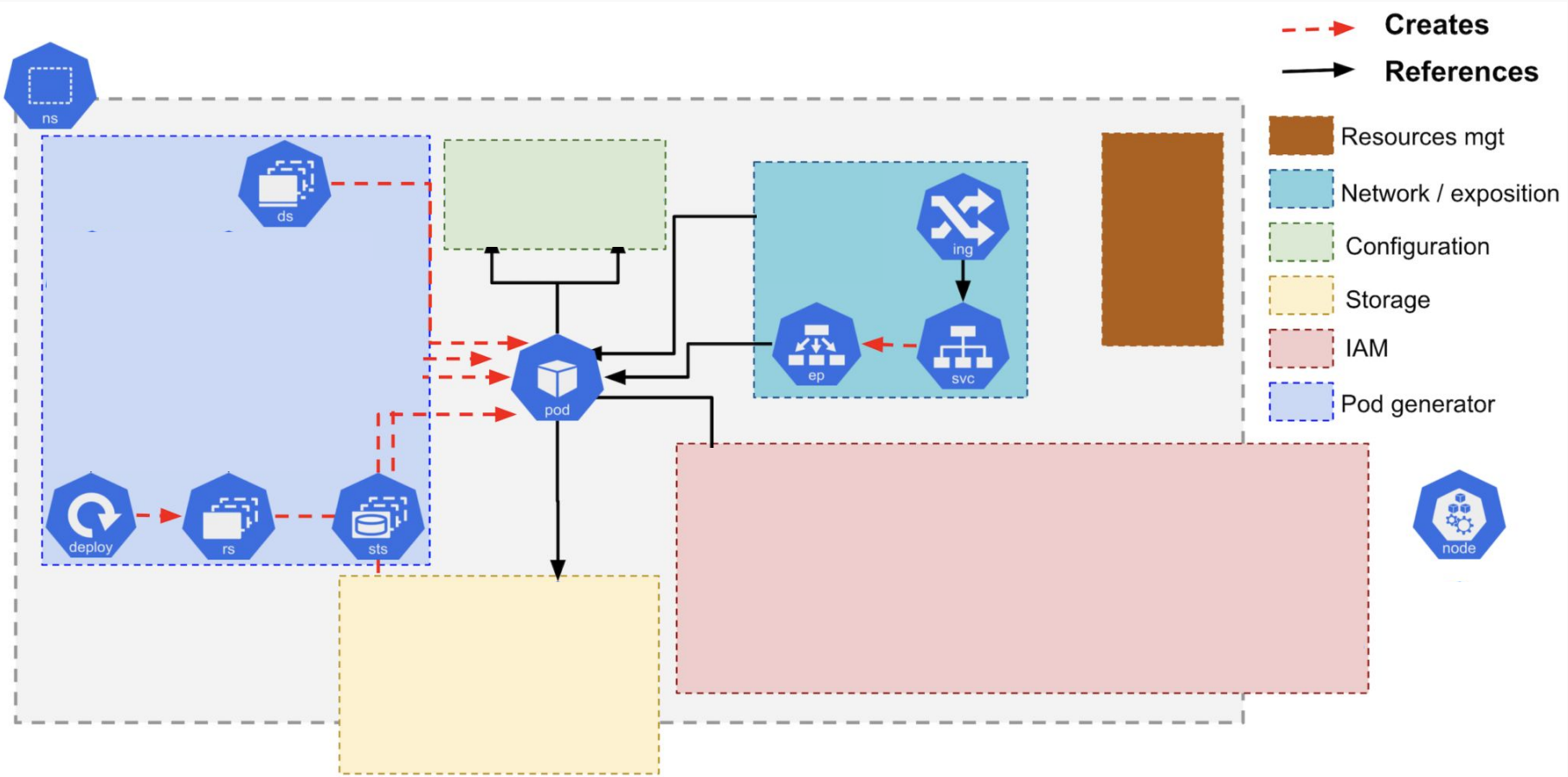
DaemonSet

- Garantiza que en todos los nodos ejecute una copia de un Pod.
- Si se añaden nodos nuevos, Pods son añadidos a los mismos.
- Si se elimina nodos del clúster, dichos Pods se destruyen.
- Al eliminar un DaemonSet se limpian todos los Pods que han sido creados.
- Casos de uso: monitores de recursos, agentes de logging o herramientas de supervisión, servicios de red o proxies de red

- Crear daemonset
  - k apply -f 08-daemonset.yml
- Verificar estado
  - k get ds
- Ver labels de los nodos
  - k get node --show-labels
- Agregar label a nodo
  - k label nodes ed-k8s-002 monitoreo=habilitado
  - k label nodes ed-k8s-m01 monitoreo=habilitado

- Verificar label agregado
  - k get node --show-labels
- Verificar que se levantó el ds
  - k get ds
- Verificar pod de node-exporter
  - k get pod -o wide
- Eliminar label de nodo
  - k label nodes ed-k8s-m01 monitoreo-
- Verificar
  - k get ds

Grupos de Objetos - Red



A R I U

Asociación Redes de Interconexión Universitaria