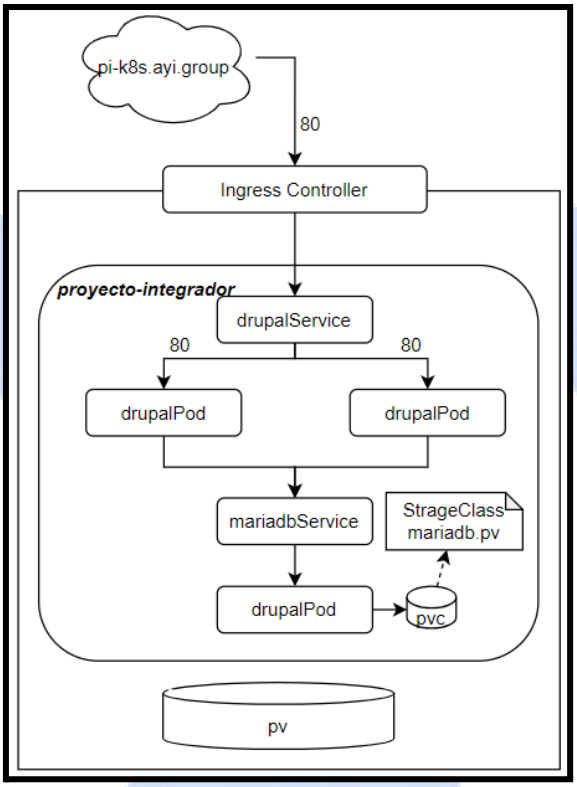
Índice

Requerimientos

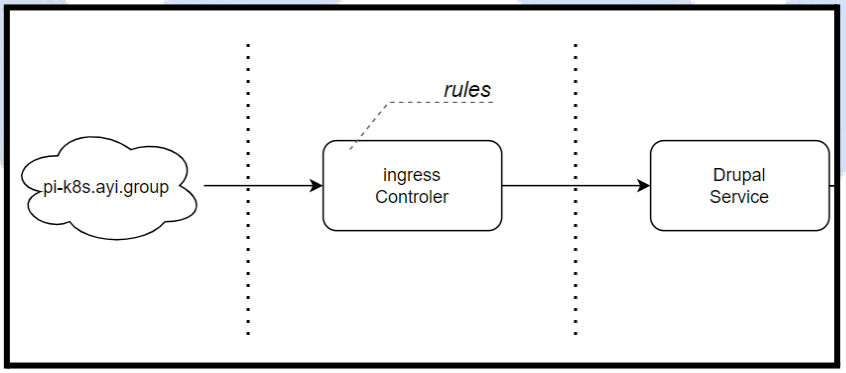
Primarios

a. Debe estar basado en *drupal* como interfaz web y contar con *mariadb* como base de datos
b. Debe contar con el dominio *pi-k8s.ayi.group* como dominio del ingress
c. Los servicios definidos para drupal y mariadb no deben ser consumidos exteriormente por su puerto. Debe ser el ingress controller quien acceda a ellos.
d. Debe contar con una división lógica de nombre *drupal-mariadb*
e. El clúster debe contar con nodo master y nodo esclavo siendo el máximo de nodos, dos.
f. Cada nodo debe contar con al menos una réplica de drupal y mariadb.
g. El proyecto debe ser capaz de ser reactivo cuando se eliminen los pods levantando correctamente las nuevas replicas.
h. Al momento de navegar por la web de drupal, los datos almacenados deben aparecer sin importar cuál réplica de drupal haya recibido las peticiones.

Arquitectura



Ingress Controller



Se define el ingress controller para el dominio *pi-k8s.ayi.group* redireccionando al servicios de drupal.

Ingress expone las rutas HTTP y HTTPS desde fuera del clúster a los servicios dentro del clúster. El enrutamiento del tráfico está controlado por reglas definidas en el recurso Ingress.

Se puede configurar un Ingress para brindar servicios de direcciones URL accesibles externamente, equilibrar la carga del tráfico, finalizar SSL/TLS y ofrecer alojamiento virtual basado en el nombre. Un controlador de Ingress es responsable de cumplir con el Ingress, generalmente con un balanceador de carga, aunque también puede configurar su enrutador perimetral o interfaces adicionales para ayudar a manejar el tráfico.
Un Ingress no expone puertos o protocolos arbitrarios. La exposición de servicios distintos de HTTP y HTTPS a Internet suele utilizar un servicio de tipo Service.Type=NodePort o Service.Type=LoadBalancer .

Requisitos previos

Debe tener un controlador de Ingress para satisfacer un Ingress. Solo crear un recurso de Ingress no tiene efecto.Crear un clúster de minikube

Si aún no ha configurado un clúster localmente, ejecute minikube start para crear un clúster.

Habilitar el controlador de ingreso

1. Para habilitar el controlador de entrada NGINX, ejecute el siguiente comando:
   minikube addons enable ingress
2. Verifique que el controlador de entrada NGINX se esté ejecutando
   kubectl get pods -n ingress-nginx

Nota: Puede pasar hasta un minuto antes de que vea que estos pods funcionan correctamente.
La salida es similar a:

```
NAME                                        READY   STATUS      RESTARTS    AGE
ingress-nginx-admission-create-g9g49        0/1     Completed   0           11m
ingress-nginx-admission-patch-rqp78         0/1     Completed   1           11m
ingress-nginx-controller-59b45fb494-26npt   1/1     Running     0           11m
```

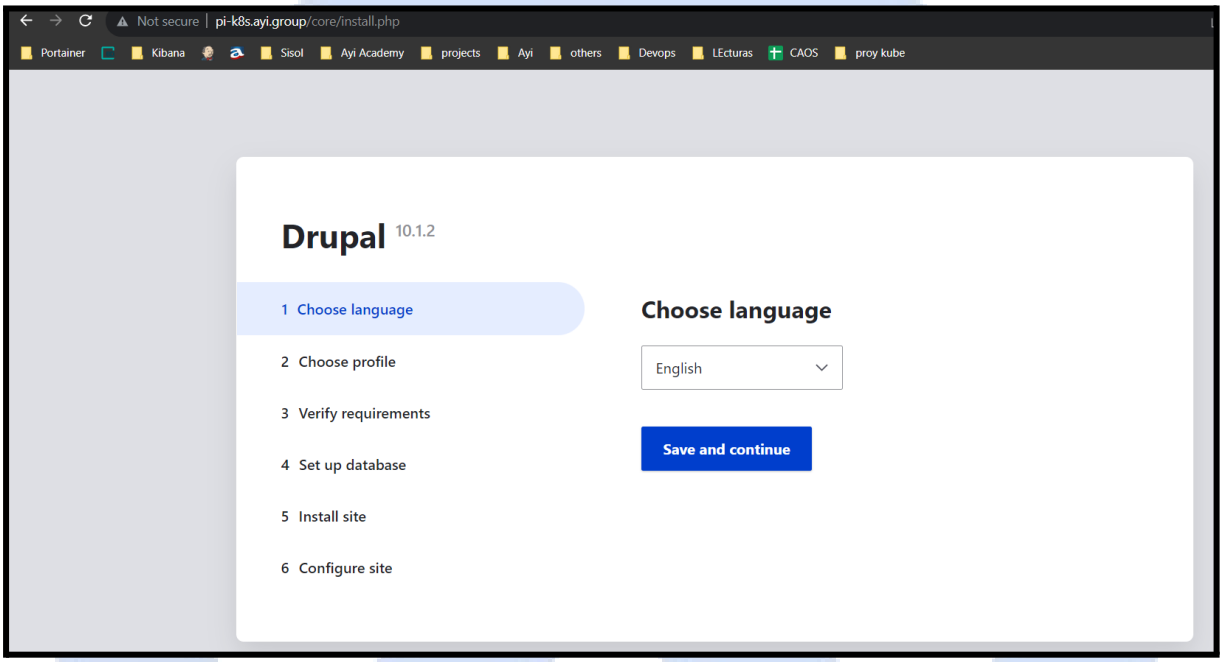
3. Para habilitar el controlador dns de entrada NGINX, ejecute el siguiente comando:
   minikube addons enable ingress-dns

Recurso ingress

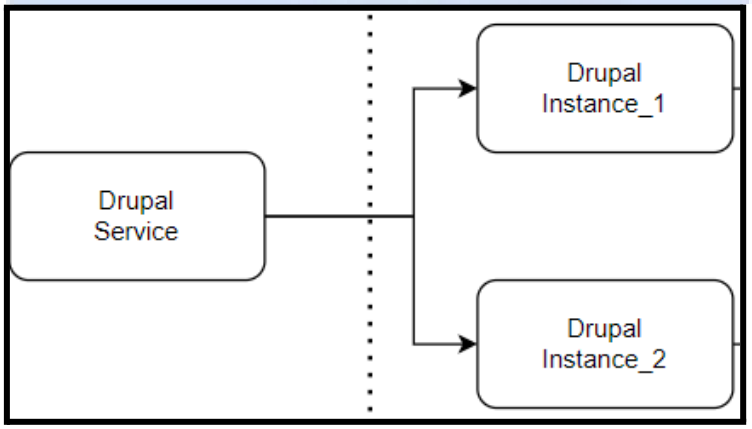
```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
 name: ingress-nginx
 annotations:
   nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
    - host: pi-k8s.ayi.group
      http:
        paths:
        - path: /
```

```
      pathType: Prefix
      backend:
        service:
          name: drupal-service
          port:
            number: 80
```

Resultado



Drupal



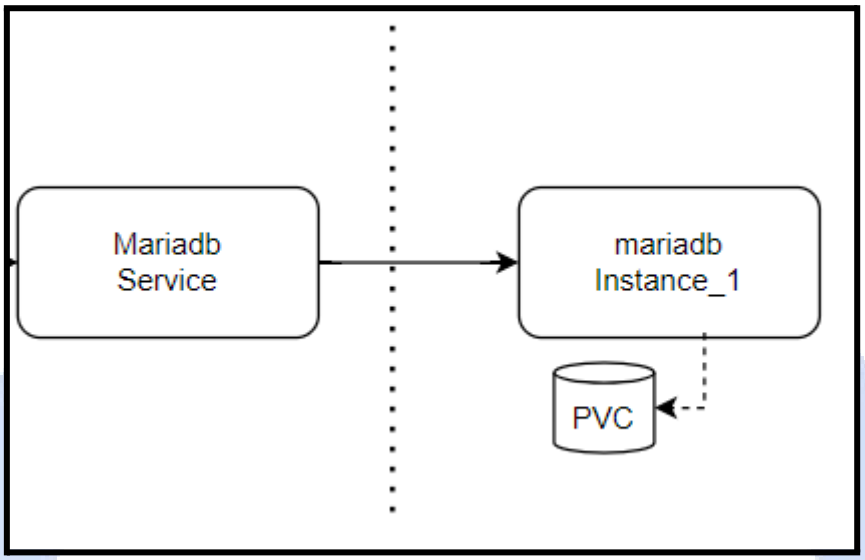
drupal deployment

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: drupal
  labels:
    app: drupal
spec:
  selector:
    matchLabels:
      app: drupal
      tier: frontend
      vol: mariadb
  replicas: 2
  template:
    metadata:
      labels:
        app: drupal
        tier: frontend
        vol: mariadb
    spec:
      containers:
      - name: drupal
        image: drupal:latest
        ports:
          - containerPort: 30080
            name: drupal
        volumeMounts:
          - name: drupal
            mountPath:
/var/www/html/modules
            subPath: modules
          - name: drupal
            mountPath:
/var/www/html/profiles
            subPath: profiles
          - name: drupal
            mountPath:
/var/www/html/themes
            subPath: themes
      volumes:
        - name: drupal
          persistentVolumeClaim:
            claimName: drupal-pv
```

drupal service

```
---
apiVersion: v1
kind: Service
metadata:
  name: drupal-service
  labels:
    app: drupal
    tier: frontend
    vol: mariadb
spec:
  type: NodePort
  ports:
  - port: 80
    nodePort: 30080
  selector:
    app: drupal
    tier: frontend
    vol: mariadb
```

Mariadb



mariadb pod

```
---
apiVersion: v1
kind: Pod
metadata:
  name: mariadb-vol
  labels:
    app: mariadb-vol
  namespace: proyecto-integrador
spec:
  containers:
    - name: mariadb
      image: mariadb:latest
      ports:
        - containerPort: 3306
          protocol: TCP
      volumeMounts:
        - mountPath: /var/lib/mysql
          name: pv-storage
      env:
        - name: MYSQL_ROOT_PASSWORD
          value: drupal
        - name: MYSQL_DATABASE
          value: drupal
        - name: MYSQL_USER
          value: drupal
        - name: MYSQL_PASSWORD
```

```
          value: drupal
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
        claimName: storage-local-1
```

mariadb service

```
---
apiVersion: v1
kind: Service
metadata:
  name: mariadb-vol
  labels:
    app: mariadb
    tier: database
    vol: "true"
    typevol: pv
  namespace: proyecto-integrador
spec:
  ports:
    - port: 3306
      protocol: TCP
  selector:
    app: mariadb-vol
  type: ClusterIP
```

mariadb pv

```
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: volume-local
  labels:
    type: local
spec:
  storageClassName: volume.local
#  claimRef:
#    namespace: proyecto-integrador
#    name: storage-local-1
#    kind: PersistentVolumeClaims
  capacity:
    storage: 3Gi
  accessModes:
```

```
    - ReadWriteOnce
  hostPath:
    path: "/home/devuser/proyectoKube/volumes"
#  persistentVolumeReclaimPolicy: Retain
#  volumeMode: Filesystem
```

mariadb pvc

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: storage-local-1
  labels:
    app: mariadb
    type: pvc
  namespace: proyecto-integrador
spec:
  storageClassName: volume.local
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
       storage: 3Gi
#  volumeMode: Filesystem
```

Referencias

- cursos kubernetes:
https://naranjax.udemy.com/course/kubernetes-al-completo/learn/lecture/34015230#overview
- ingress: https://kubernetes.io/docs/concepts/services-networking/ingress/
- ingress minikube:
https://kubernetes.io/docs/tasks/access-application-cluster/ingress-minikube/
- deployment vs statefulset vs daemonset:
https://www.linkedin.com/pulse/implementando-en-kubernetes-deployment-vs-statefulset-de-le%C3%B3n/?originalSubdomain=es