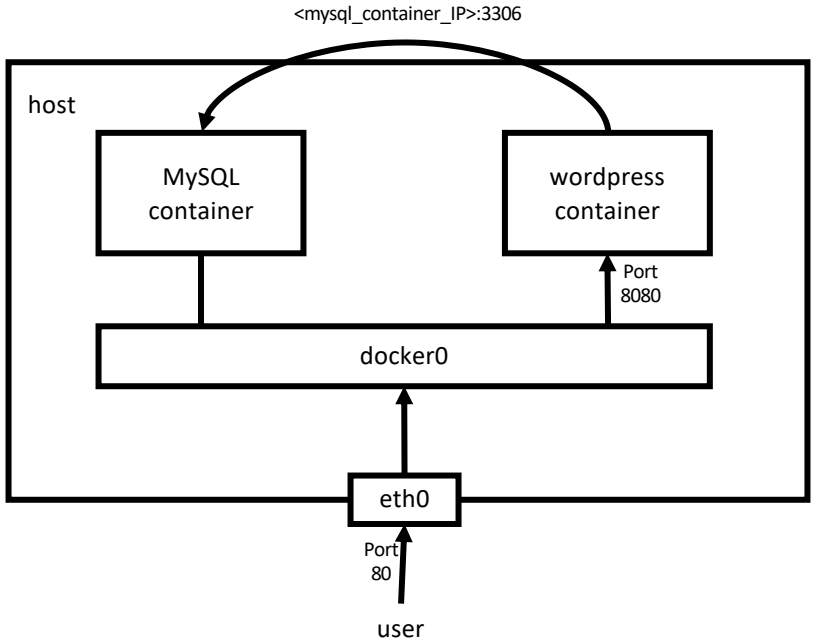
Introduction to K8S workload

莊家儁

Wordpress + MySQL的例子



https://github.com/ogre0403/kubernetes-hands-on

Kubernetes Basic Concept

- Pod
- Label & Annotation
- Built-in workload
  - Deployment
- Service
- Secret & configMap
- Ingress
- Volume

Kubernetes - Pod

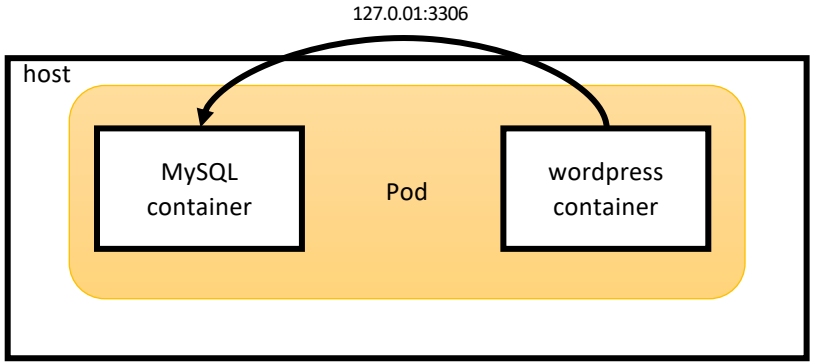
- Pod為 Kubernetes 中最小的部署單位 Pod
  - 由一至多個容器所組成
  - 通常將耦合性較高之container放置在同一個pod
  - 同一個pod的container會在同一個node
- 共享同樣的 network space
  - Pod內的container可以用localhost溝通
- 共享儲存空間 (volumes)
  - 在Pod定義volume，container可以掛載這些定義好的volume

Wordpress + MySQL:
一個pod有多個Conatiner

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql-wordpress-pod
spec:
  containers:
  - name: mysql
    image: mysql:5.6
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: Password1234
  - name: wordpress
    image: wordpress:4.8-apache
    env:
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
    - name: WORDPRESS_DB_PASSWORD
      value: Password1234
```

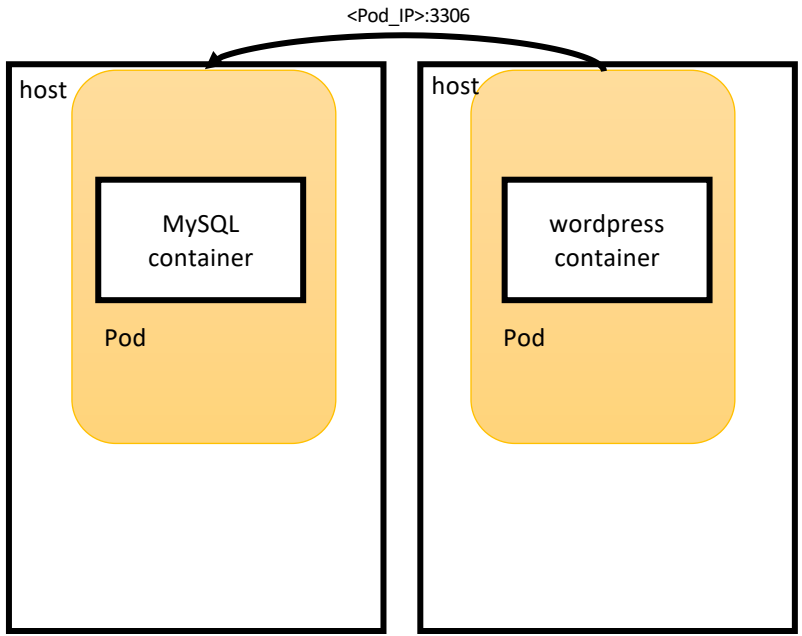
```
# 透過port-forward“暫時”可以存取

$ kubectl port-forward mysql-wordpress-pod  8080:80
```



Wordpress + MySQL:
一個pod只有一個container

需手動找出Pod IP



Kubernetes Basic Concept

- Pod
- Label & Annotation
- Built-in workload
  - Deployment
- Service
- Secret & configMap
- Ingress
- Volume

Label & Annotations

```
"environment" : "dev", "environment" : "qa", "environment" : "production"
"tier" : "frontend", "tier" : "backend", "tier" : "cache"
"partition" : "customerA", "partition" : "customerB"
```

- 相同點
  - key/value pair
  - 每個物件可以同時擁有許多個labels (or annotation)
  - 皆位於 object 的 metadata 欄位內
- 不同點
  - Label 可以透過selector進行篩選
    - kubectl get node –l <key>=<value>
  - annotation做為沒有識別用途的標籤
    - 程式讀取annotation的值進行特定工作
    - Eg: prometheus.io/path: “/metrics”

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql-pod
  labels:
    app: blog
    tier: mysql
  annotations:
    compnay: 'NCHC'
spec:
  containers:
  - name: mysql
    image: mysql:5.6
```

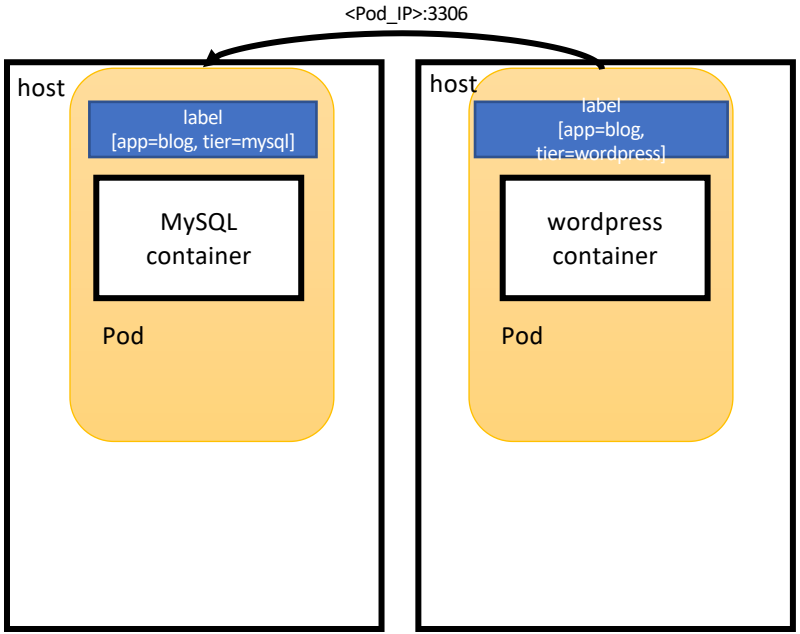
Label的應用：讓Pod運行在特定節點上

- 在Pod Definition裡使用NodeSelector
- 找出目前node的label
  - kubectl get node – show-labels
- 在node上打上label
  - kubectl label node <name> <key>=<value>
- 在node上移除label
  - kubectl label node <name> <key>-

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
  nodeSelector:
    hardware: high-memory
```

Wordpress + MySQL:
每個Pod都不同的Label

需手動找出Pod IP




```
$ kubectl  get pod -o wide --show-labels
$ kubectl get pod -o wide -l app=blog
```

Kubernetes Basic Concept

- Pod
- Label & Annotation
- Built-in workload
  - Deployment
- Service
- Secret & configMap
- Ingress
- Volume

Kubernetes built-in workload

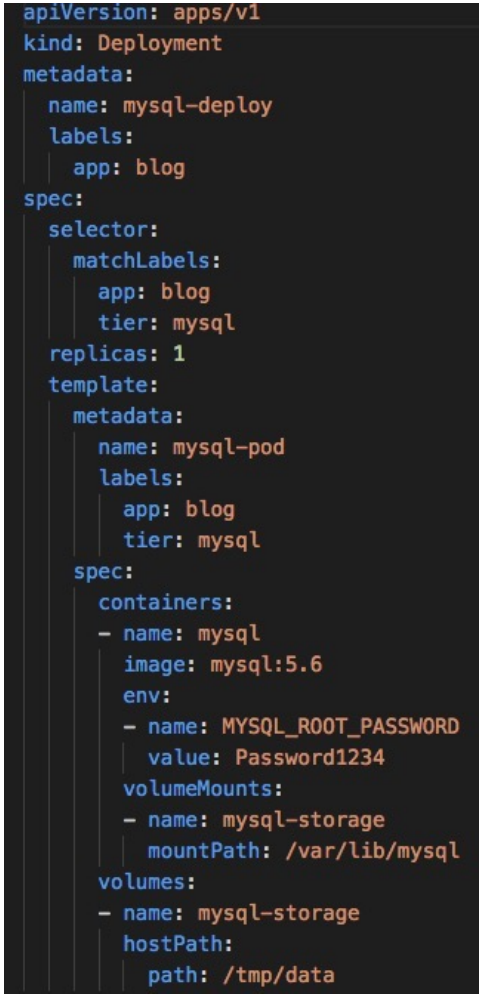
- Pod 是K8S裡最小的部built-in workload署單位，但是不建議直接使用Pod，而是用built-in workload
  - Deployment
  - Job/CronJob
  - StatefulSet
  - DemonSet
- Built-in workload有相對應的controller協助管理

Deployments

- Deployment 的功用
  - 確保 Pod 數量(replicas)滿足所設定的值
  - 變動Pod的數量(Scale)
  - 滾動升級(Rolling update)
  - 回滾(Roll back)的機制

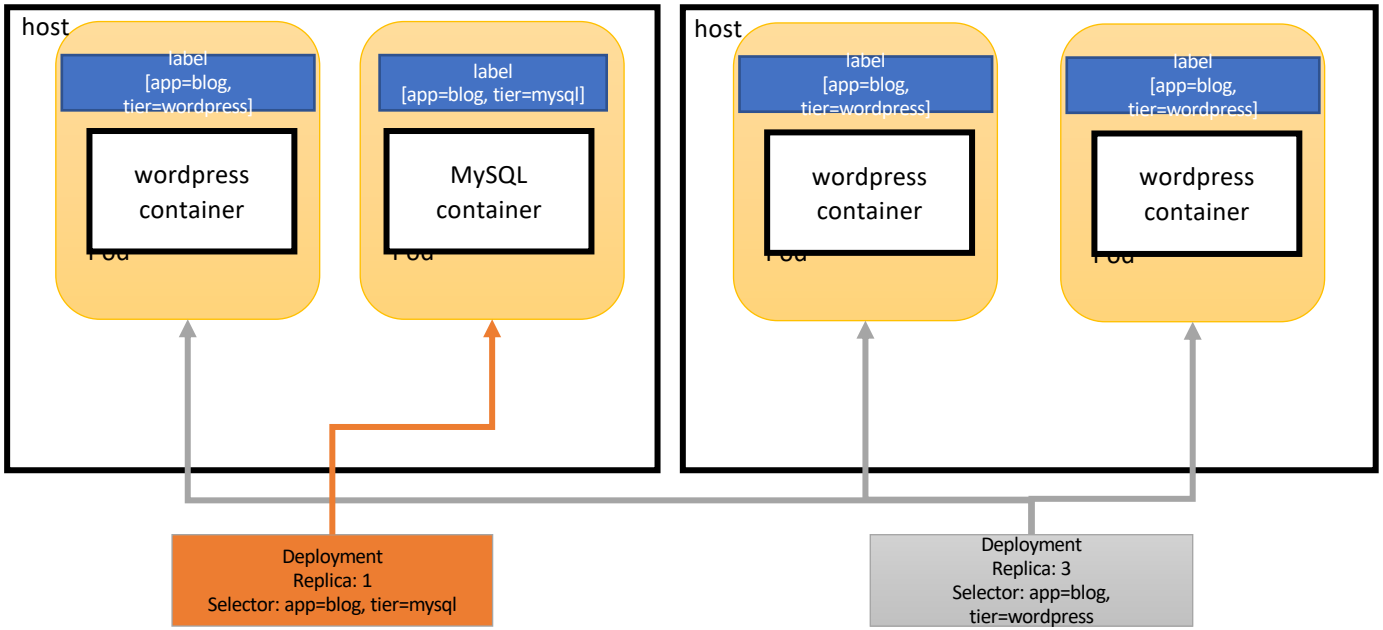
Selector

Pod Definition



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deploy
  labels:
    app: blog
spec:
  selector:
    matchLabels:
      app: blog
      tier: mysql
  replicas: 1
  template:
    metadata:
      name: mysql-pod
      labels:
        app: blog
        tier: mysql
    spec:
      containers:
      - name: mysql
        image: mysql:5.6
        env:
        - name: MYSQL_ROOT_PASSWORD
          value: Password1234
        volumeMounts:
        - name: mysql-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-storage
        hostPath:
          path: /tmp/data
```

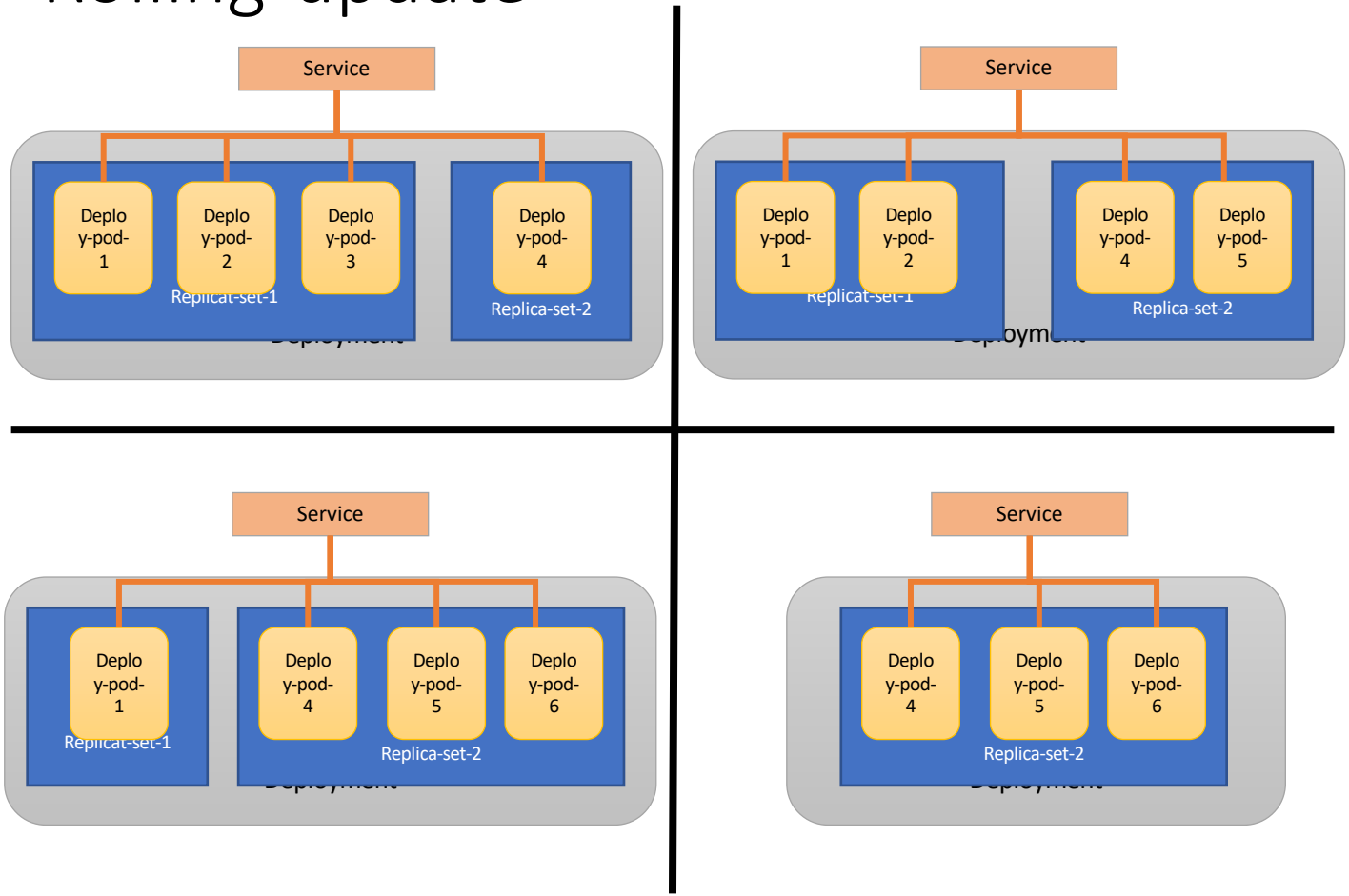
Wordpress + MySQL的例子



Deployment 的主要功能

- Scaling a Deployment
  - 擴展pod的數量
- Rolling update (zero down time update)
  - 可以達到無停機服務遷移
- Roll back
  - 當update 完後發現此次的升級造成服務發生不穩定的狀況，可以利用 rollback 來回復到先前的狀態。

Rolling-update



常用 Deployment 指令

Deployment相關指令	指令功能
kubectl scale deploy <deployment> --replicas=n	Scale deployment 物件的pod數
kubectl edit deploy <deployment>	編輯特定deployment物件
kubectl set image deployment <deployment> <container>=<image>	將deployment管理的pod升級到特定image版本
kubectl rollout status deploy <deployment>	查詢目前某deployment升級狀況
kubectl rollout history deploy <deployment>	查詢目前某deployment升級的歷史紀錄
kubectl rollout undo deploy <deployment> --to-revision=n	回滾Pod到某個特定版本
kubectl rollout undo deploy <deployment>	回滾Pod到先前一個版本

scale

Rolling update

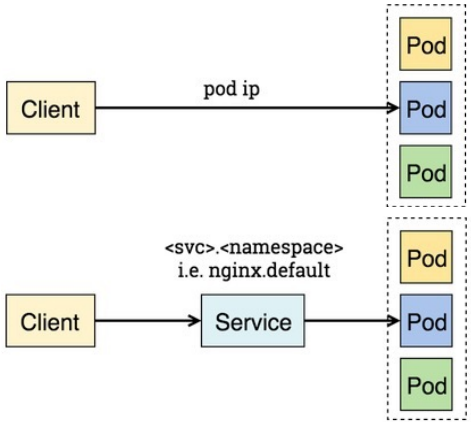
Roll back

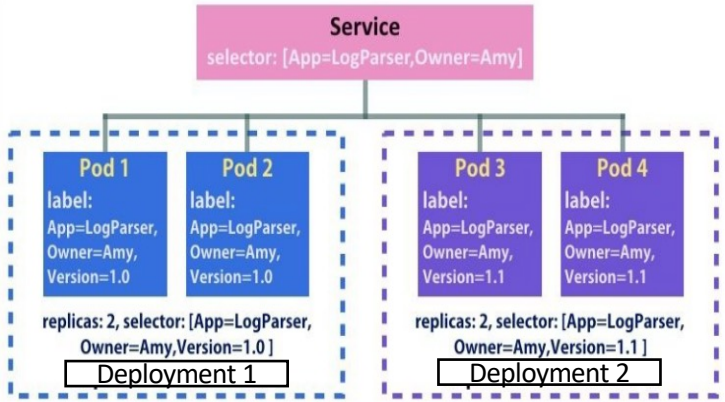
Kubernetes Basic Concept

- Pod
- Label & Annotation
- Built-in workload
  - Deployment
- Service
- Secret & configMap
- Ingress
- Volume

為什麼需要Service

- Pod的 IP 會變動，不應該透過Pod IP進行存取
- Deployment 有多個replica, 都有各自的Pod IP
- 透過Service的cluster IP，service會將至cluster IP的流量導入不同的pod.
  - 透過Label與selector決定service與pod的對應
  - 可以透過<svc>.<namespace> 的DNS來存取，而不透過IP





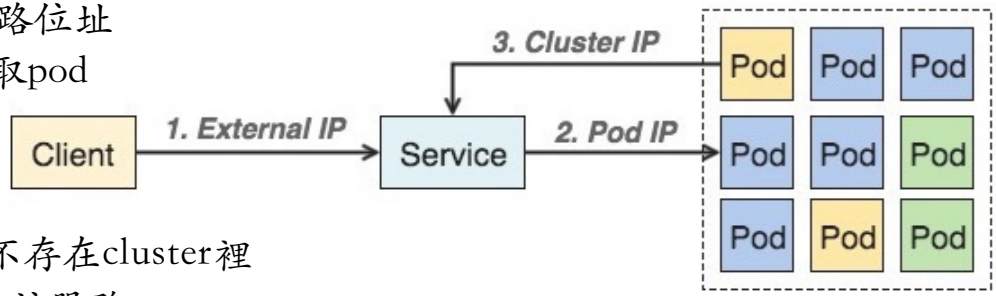
Service

- Service有三種類型
  - ClusterIP、NodePort、LoadBalancer

- Kubernetes 內有三種不同的IP
  - Pod 有 Pod IP
  - Service 有
    - ClusterIP: 建立svc 就會產生
    - External IP： 需由cloud provider動態分配

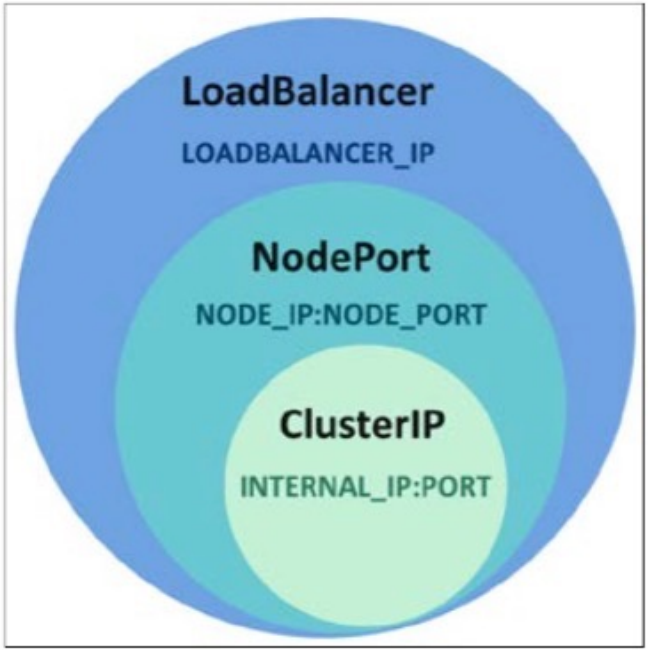
Kubernetes 的三種IP

- Pod IP
  - 每個 Pod 在overlay network上的網路位址
  - Cluster 內的 Pod/node 可用此ip存取pod
- Cluster IP
  - service在叢集內部的網路位址
  - 是一個virtual ip，不是real ip，且不存在cluster裡
  - 只有Cluster 內的 Pod 可用此ip存取該服務
- External IP
  - 透過cloud provider提供
  - Service暴露在外的位址，供外部使用者連線




```
root@ubuntu:~# kubectl get pod -o wide
NAME                              READY     STATUS    RESTARTS   AGE       IP
nfs-provisioner-5bfd94c4b-86g7w   1/1       Running   0          11d       172.20.0.170
root@ubuntu:~# kubectl get svc -n=ingress-nginx
NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
default-http-backend   ClusterIP   10.68.90.254    <none>        80/TCP
```

Service type



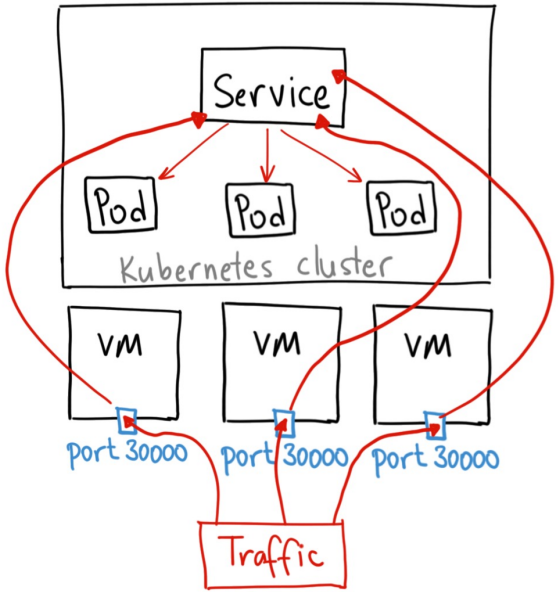
- ClusterIP: svc.Port→svc.targetPort
  1. Expose 一個cluster內部可存取的IP

- NodePort: nodePort→svc.Port→svc.targetPort
  1. Expose 一個cluster內部可存取的IP
  2. 在每個node的ip都expose相同的port

- Loadbalancer: LB→nodePort →svc.port→svc.targetPort
  1. Expose 一個cluster內部可存取的IP
  2. 在每個node的ip都expose相同的port
  3. Cloud provider提供一個LB的IP，此IP綁定到expose 的nodePort

Node Port

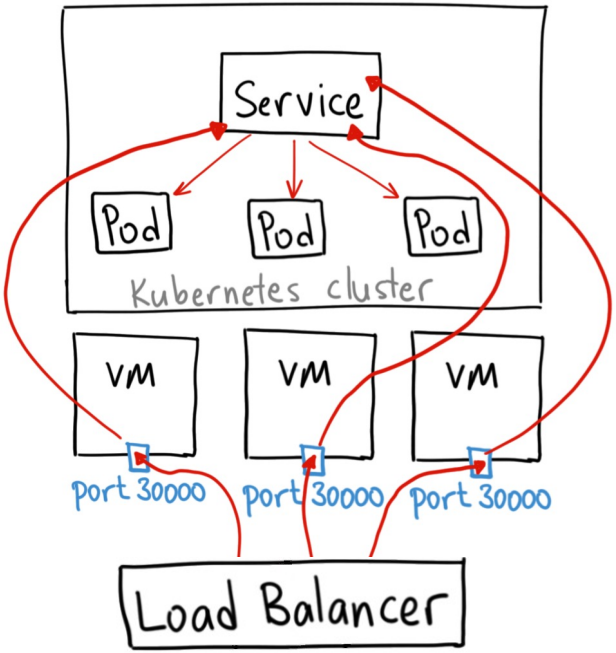
- Expose service to outside cluster
- One service uses one nodePort




```
root@ubuntu:~# kubectl get svc -n=ingress-nginx
NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE
default-http-backend   ClusterIP   10.68.90.254    <none>        80/TCP                       11d
ingress-nginx          NodePort    10.68.163.140   <none>        80:32447/TCP,443:37619/TCP   10d
```

LoadBalancer

- Provided by cloud provider
- 若沒有Cloud provider, 則會一直pending




```
NAME          TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE
hub           ClusterIP      10.68.169.105   <none>        8081/TCP                     12m
proxy-api     ClusterIP      10.68.242.208   <none>        8001/TCP                     12m
proxy-http    ClusterIP      10.68.144.45    <none>        8000/TCP                     12m
proxy-public  LoadBalancer   10.68.89.253    <pending>     80:24837/TCP,443:22383/TCP   12m
```

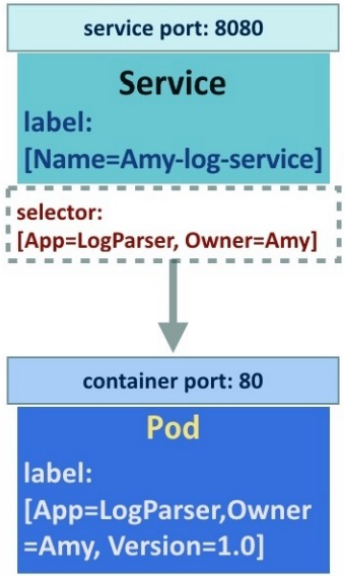
- 用expose建立Service
  - 可以用在pod、deploy

```
$ kubectl expose pod <POD_NAME> --labels="Name=Amy-log-service"
--selector="App=LogParser,Owner=Amy" --port=8080 --target-port=80
--name="my-service"
```

```
$ kubectl expose deploy nginx-deploy --name="service-deploy"  --port 80
```

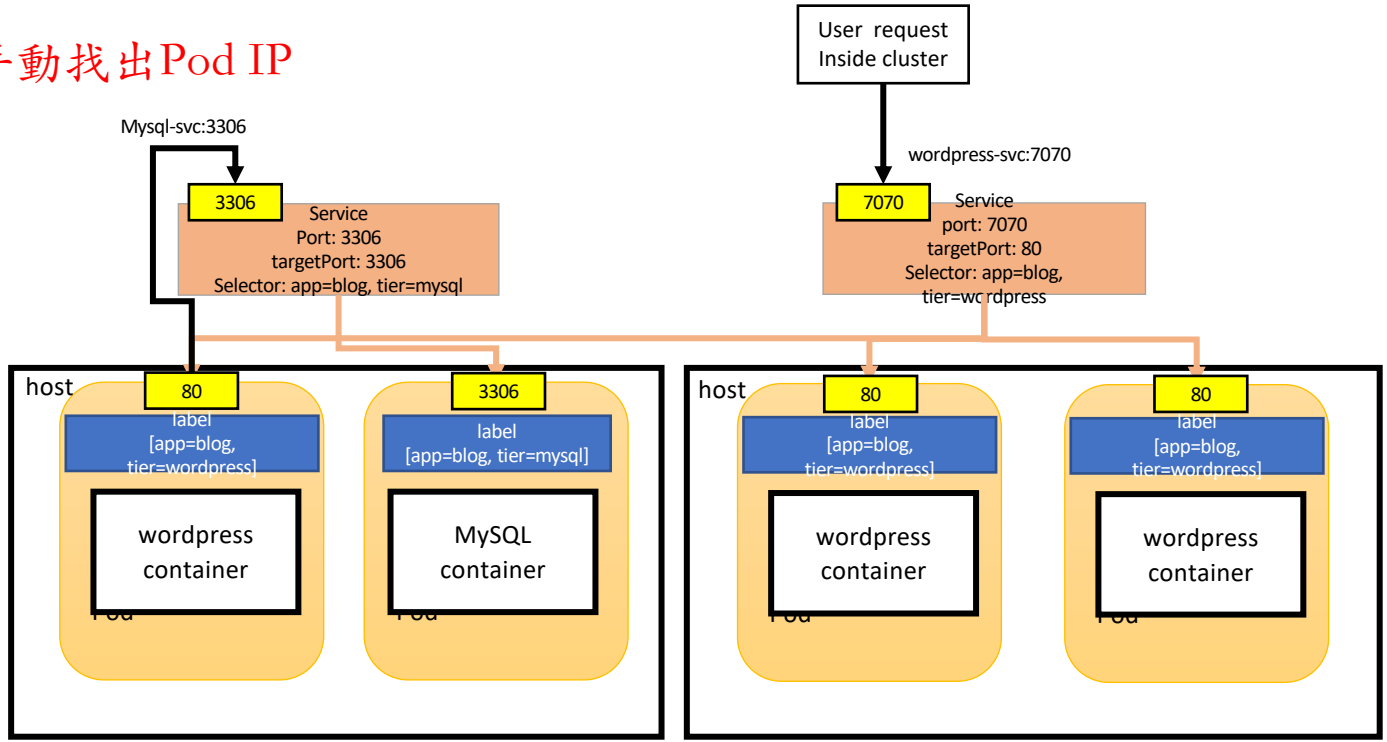
- 用YAML檔描述Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
  label:
    Name: Amy-log-service
selector:
  App: LogParser
  Owner: Amy
spec:
  type: ClusterIP
  ports:
  - port: 8080
    targetPort: 80
```

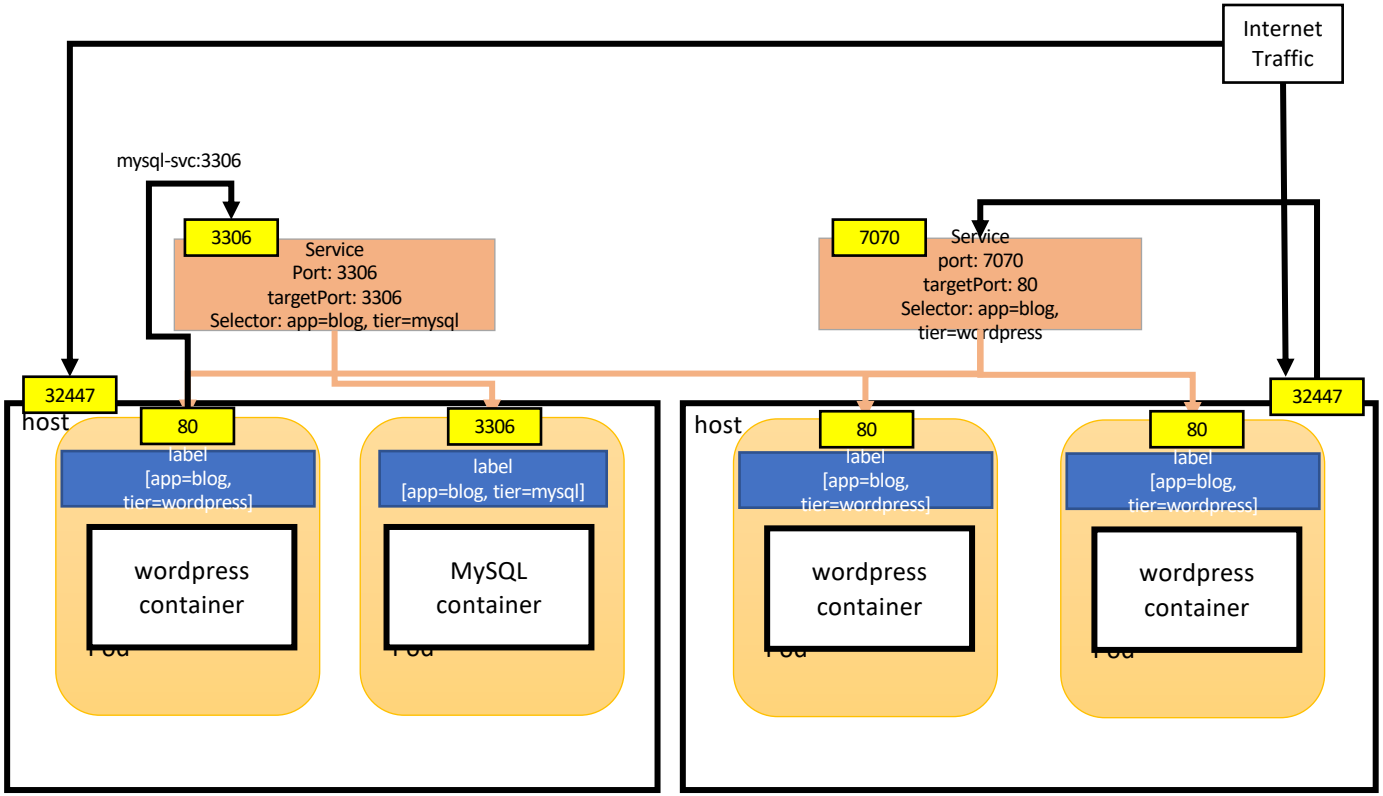


Wordpress + MySQL:
建立ClusterIP的Service，讓deployment互連

不需手動找出Pod IP



Wordpress + MySQL的：
建立NodePort的Service讓外部存取



如何讓外部存取Pod

- 透過 Service提供的機制
  - NodePort
  - Load Balancer

- 透過某台主機的網路
  - HostPort
  - 將容器的port和所在節點的port做對應，透過節點ip：port來訪問容器
    - kubectl apply -f example/04-service/hostnetwork/hostport.yaml
    - 透過hostIP:8088 存取

- Ingress (後續補充)
  - 只能用在網頁類型服務

Summary

- Service有ClusterIP, NodPort, LoadBalancer三種
- 透過Service，可以用<SVC>.<NAMEPSACE>存取Pod
- 要讓外部存取pod，可以使用NodPort, LoadBalancer，Ingress, HostPort

Kubernetes Basic Concept

- Pod
- Label & Annotation
- Built-in workload
  - Deployment
- Service
- Secret & configMap
- Ingress
- Volume

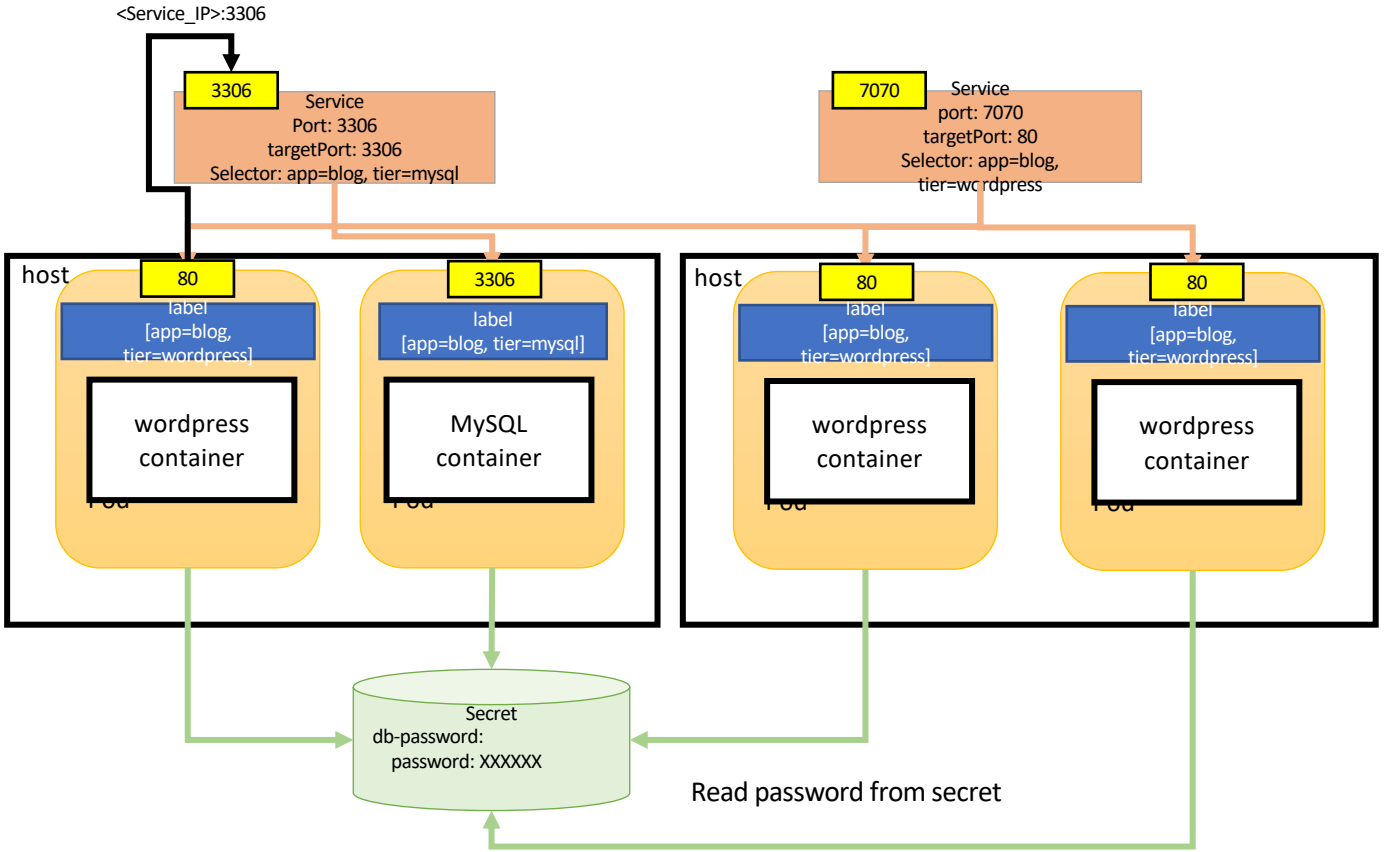
Secret

- 避免將機密資訊直接存放在Pod裡
- 透過mount / 環境變數 在 Pod裡使用
- 其實不安全
- 機密資料需先透過base64編碼
- 大小只能1MB

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: db-password
type: Opaque
data:
  password: UGFzc3dvcmQxMjM0
```

```yaml
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
    volumeMounts:
    - name: db-secret
      mountPath: "/etc/db-secret"
      readOnly: true
  volumes:
  - name: db-secret
    secret:
      secretName: db-password
```

Wordpress + MySQL的例子



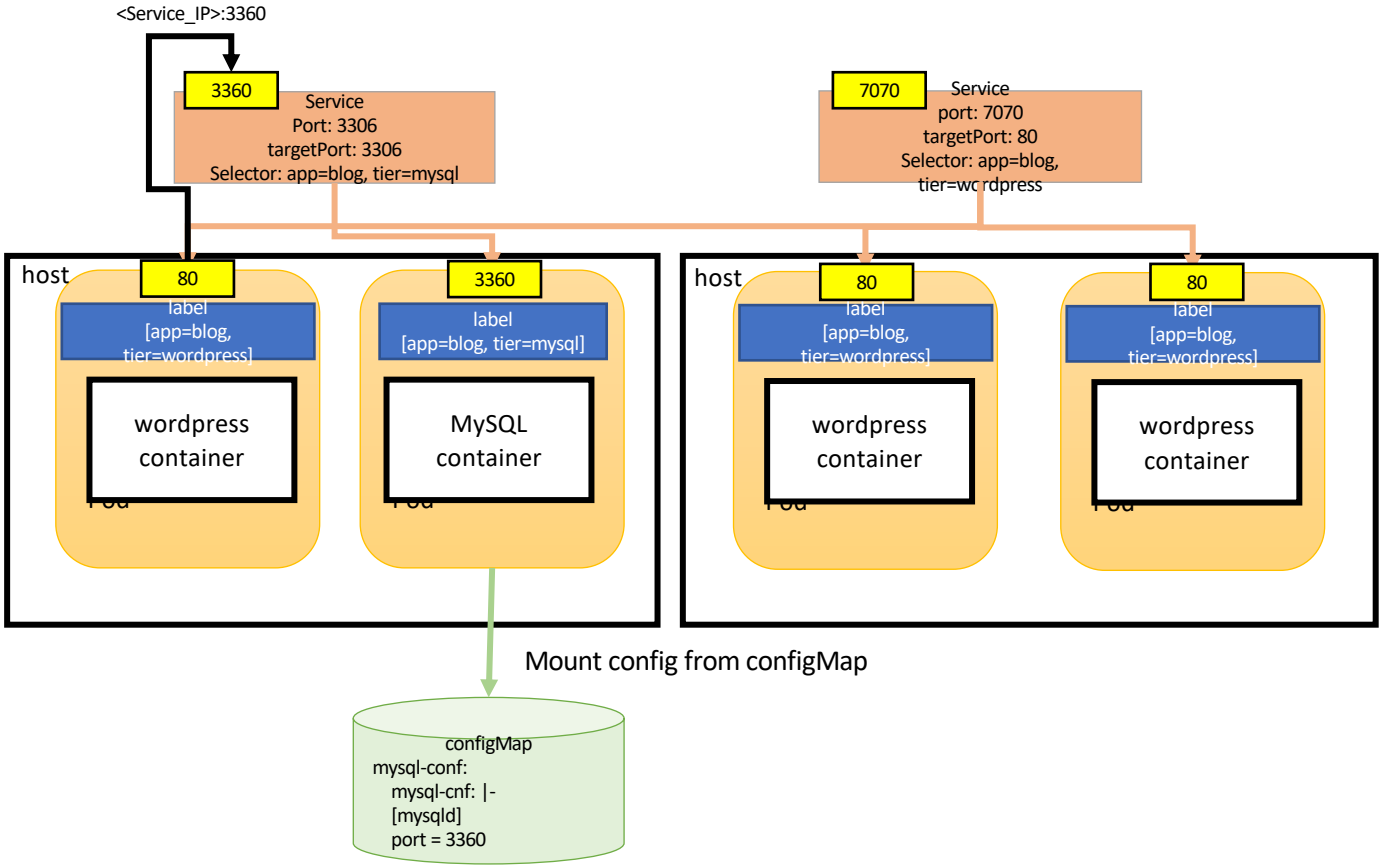
ConfigMap

- 避免將非機密資訊直接存放在Pod裡
- 透過mount / 環境變數 在 Pod裡使用
- 其實不安全
- ~~機密資料需先透過base64編碼~~
- 大小只能1MB

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-conf
data:
  mysql-cnf: |-
    [mysqld]
    port = 3360
  mysql-port: "3360"
```

```yaml
spec:
  containers:
  - name: mysql
    image: mysql:5.6
    env:
    - name: MYSQL_ROOT_PASSWORD
      valueFrom:
        secretKeyRef:
          name: db-password
          key: password
    volumeMounts:
    - name: mysql-storage
      mountPath: /var/lib/mysql
    - name: mysql-conf
      mountPath: /etc/mysql/
  volumes:
  - name: mysql-storage
    hostPath:
      path: /tmp/data
  - name: mysql-conf
    configMap:
      name: mysql-conf
      items:
      - key: mysql-cnf
        path: my.cnf
```

在Wordpress + MySQL的例子中



Kubernetes Basic Concept

- Pod
- Label & Annotation
- Built-in workload
  - Deployment
- Service
- Secret & configMap

核心元件

- Ingress (請先安裝ingress-controller再進行範例)
- Volume

開放第三方提供擴充元件

```
$ helm repo add bitnami https://charts.bitnami.com/bitnami
$ helm repo update
$ helm search repo  bitnami/nginx
$ helm install ingress-controller --set kind=DaemonSet --set service.type=ClusterIP --set daemonset.useHostPort=true bitnami/nginx-ingress-controller

$ helm install ingress-controller --namespace ingress-controller     --set nodeSelector."kubernetes\\.io/hostname"=master   --set kind=Deployment --set service.type=ClusterIP --set hostNetwork=true --version 8.0.11 bitn
```

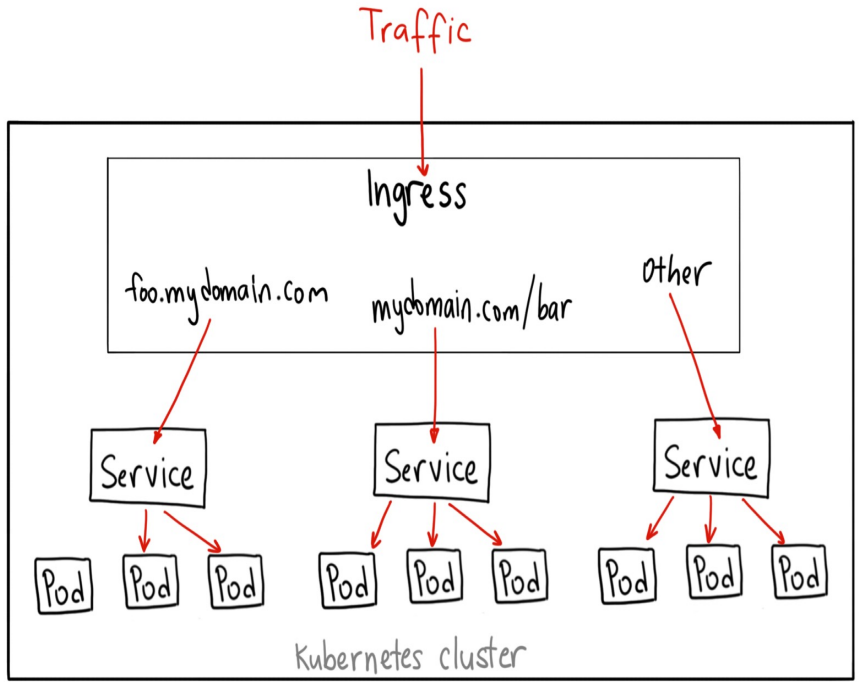
Ingress

- 如何 expose service給外部？
  - NodePort: 每個服務要對應一個nodePort
  - Load Balancer: 必需有cloud provider支援
  - HostPort: 使用實體主機上的網路對外溝通
  - Ingress: 支援L7

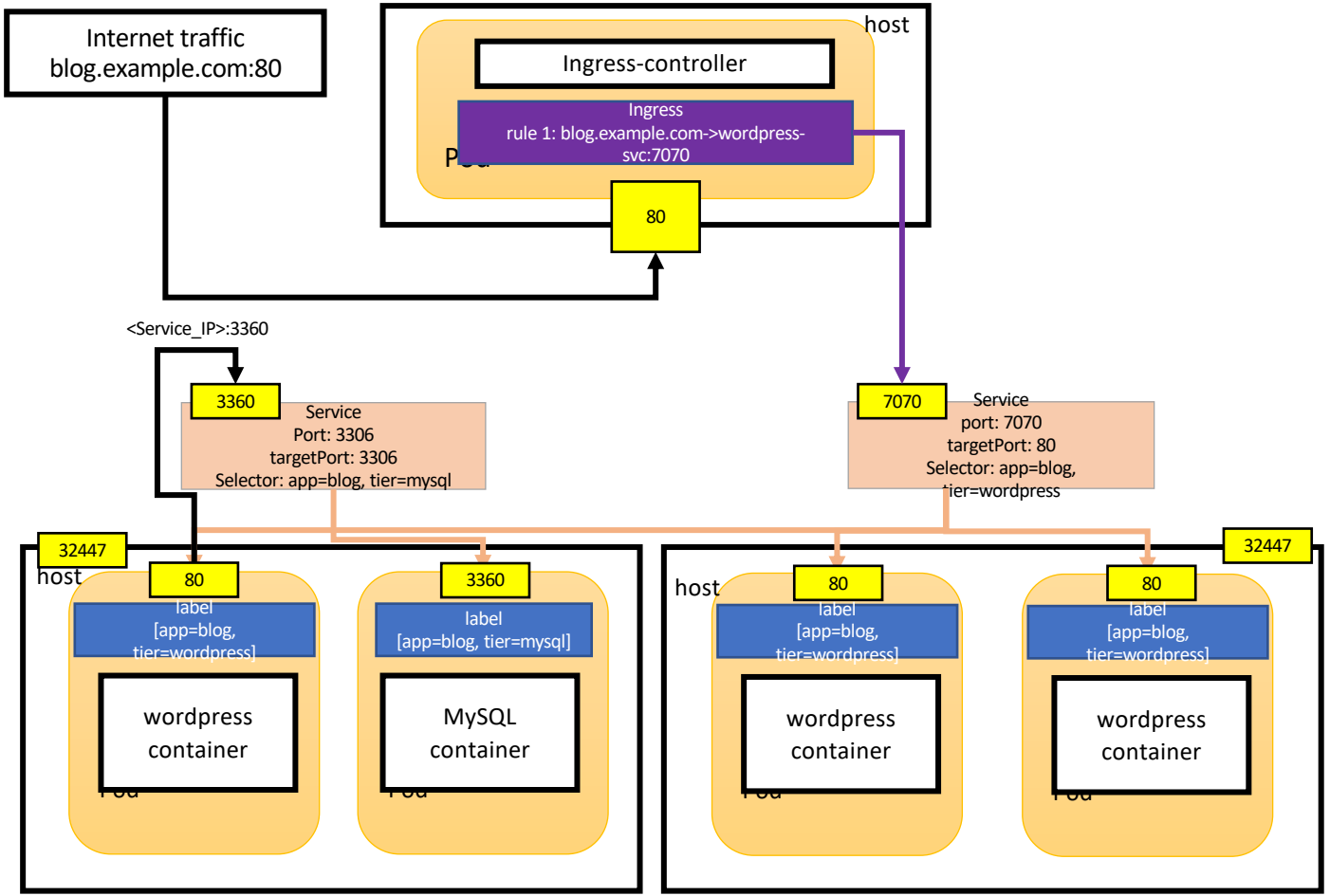
- Ingress 包括
  - Ingress
    - 設定轉發的規則
  - Ingress-controller
    - 有不同的實作方法，nginx-controller是用nginx當proxy進行轉發
  - Service for ingress-controller
    - 讓外面存取不同服務的request都進入ingress-controller

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx
  name: wordpress-ingress
spec:
  rules:
  - host: blog.140-110-136-74.nip.io
    http:
      paths:
      - backend:
          service:
            name: wordpress-svc
            port:
              number: 7070
        pathType: ImplementationSpecific
```

Rule-1



Ingress controller 採用hostPort



Kubernetes Basic Concept

- Pod
- Label & Annotation
- Built-in workload
  - Deployment
- Service
- Secret & configMap
- Ingress
- Volume (請先安裝NFS server & nfs-provider再進行範例)

Volume

- Volume & Persistence Volume
- Persistence Volume & Persistence Volume Claim
- Persistence Volume & StroageClass
- CSI Persistence Volume

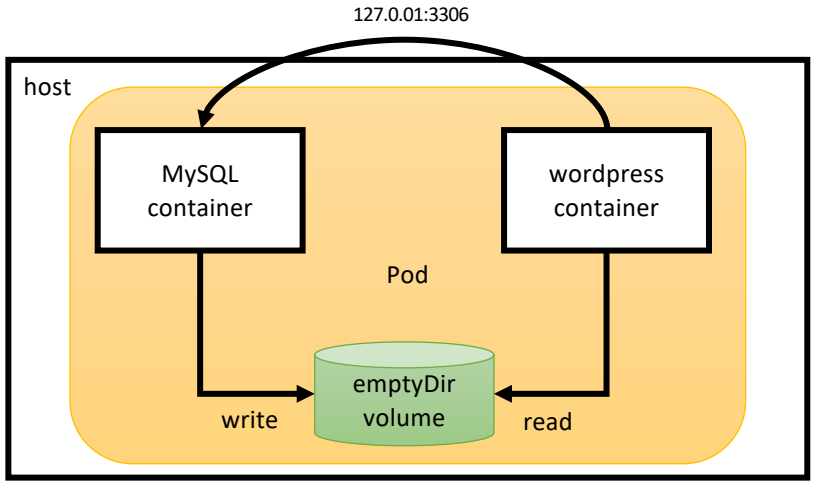
為什麼需要Volume

- 容器裡的檔案是ephemeral，容器重啟後，資料就消失了
  - 要透過Volume進行持久化儲存
- Pod裡的container，可以共用volume
- Volume週期和pod一樣
- Kubernetes has different volume plugin

Temp	Local	Network
• emptyDir	• hostPath	• GlusterFS • CephRBD • gitRepo • secret • flocker • gcePersistentDisk • AWS ElasticBlockStore (EBS) • NFS • iSCSI • Fibre Channel • Cinder

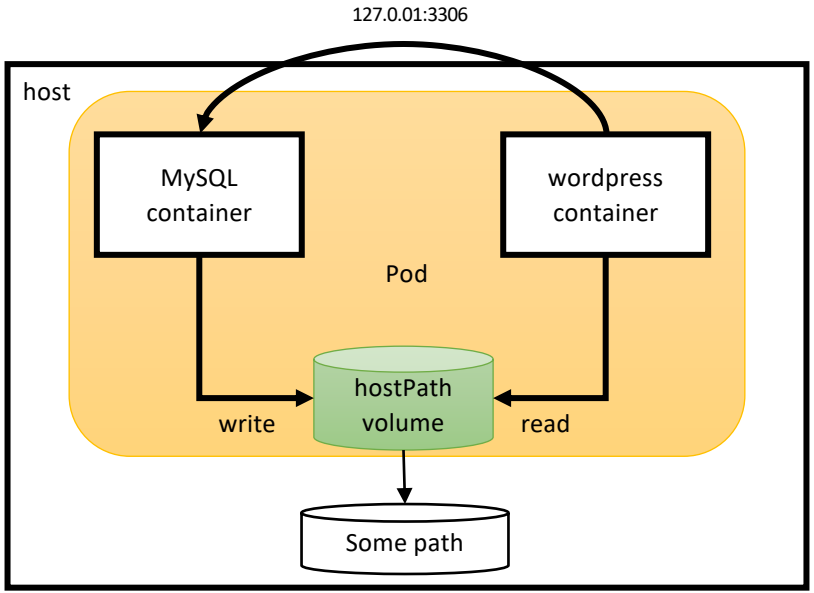
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql-wordpress-pod
spec:
  containers:
  - name: mysql
    image: mysql:5.6
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: Password1234
    volumeMounts:
    - name: mysql-storage
      mountPath: /var/lib/mysql
  - name: wordpress
    image: wordpress:4.8-apache
    env:
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
    - name: WORDPRESS_DB_PASSWORD
      value: Password1234
    volumeMounts:
    - name: mysql-storage
      mountPath: /var/lib/wordpress
  volumes:
  - name: mysql-storage
    emptyDir: {}
```

- 使用emptyDir
  - 生命週期和pod相同
    - 當pod被移除時，emptyDir也會被清空
    - 但Container 重啟，資料仍會保留
  - 同一個pod的container間，可以共享資料



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql-wordpress-pod
spec:
  containers:
  - name: mysql
    image: mysql:5.6
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: Password1234
    volumeMounts:
    - name: mysql-storage
      mountPath: /var/lib/mysql
  - name: wordpress
    image: wordpress:4.8-apache
    env:
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
    - name: WORDPRESS_DB_PASSWORD
      value: Password1234
    volumeMounts:
    - name: mysql-storage
      mountPath: /var/lib/wordpress
  volumes:
  - name: mysql-storage
    hostPath:
      path: /tmp/data
```

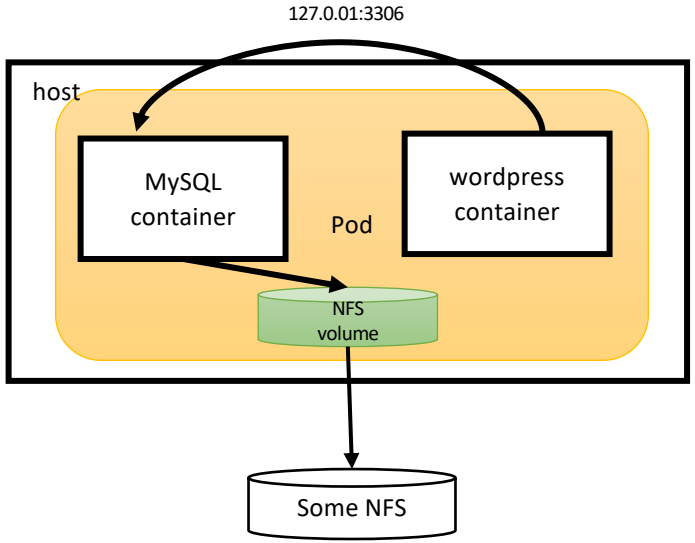
- 使用hostPath
  - 將host上的某個目錄掛載至Pod
  - 資料會保存在host上，重啟pod不會清空資料
  - 但container 移動到別的host上，資料遺失



Wordpress + MySQL的例子

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql-wordpress-pod
spec:
  containers:
  - name: mysql
    image: mysql:5.6
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: Password1234
    volumeMounts:
    - name: mysql-storage
      mountPath: /var/lib/mysql
  - name: wordpress
    image: wordpress:4.8-apache
    env:
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
    - name: WORDPRESS_DB_PASSWORD
      value: Password1234
  volumes:
  - name: mysql-volumes
    nfs:
      server: 192.168.2.31
      path: /nfs-data
```

- 使用nfs



使用者

3. 在YAML裡使用

2. 設定好告知使用者

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql-wordpress-pod
spec:
  containers:
  - name: mysql
    image: mysql:5.6
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: Password1234
    volumeMounts:
    - name: mysql-storage
      mountPath: /var/lib/mysql
  - name: wordpress
    image: wordpress:4.8-apache
    env:
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
    - name: WORDPRESS_DB_PASSWORD
      value: Password1234
  volumes:
  - name: mysql-volumes
    nfs:
      server: 192.168.2.31
      path: /nfs-data
```

管理者

1. 讓K8S支援

儲存設備業者

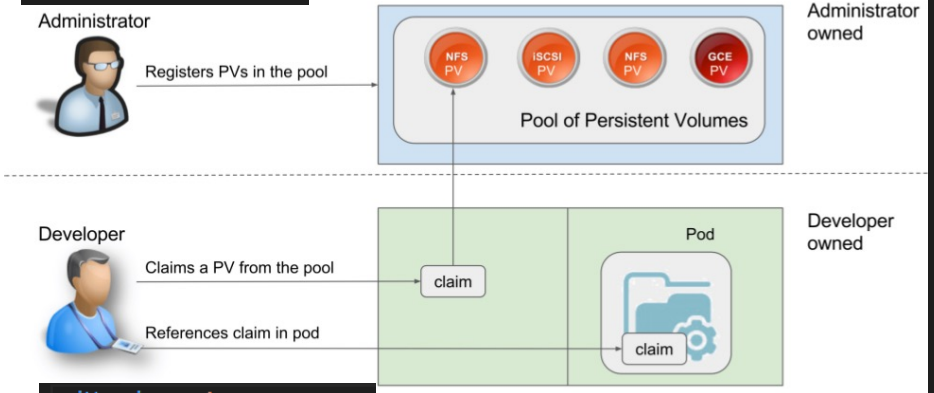
Persistence Volume (PV) & Persistence Volume Claim(PVC)

- Abstraction of Physical storage
  - Separate **Dev**lopment & **Op**erator

- PersistentVolume (PV)
  - A networked storage provisioned **by an administrator**
  - Operator view

- PersistentVolumeClaim (PVC)
  - Request for storage **from a user**
  - Development view

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: 192.168.2.31
    path: /nfs-data
```




```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 1Gi
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql-wordpress-pod
spec:
  containers:
  - name: mysql
    image: mysql:5.6
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: Password1234
    volumeMounts:
    - name: mysql-volumes
      mountPath: /var/lib/mysql
  - name: wordpress
    image: wordpress:4.8-apache
    env:
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
    - name: WORDPRESS_DB_PASSWORD
      value: Password1234
  volumes:
  - name: mysql-volumes
    persistentVolumeClaim:
      claimName: nfs-pvc
```

使用者

3-0. 宣告想用的儲存
3-1. 綁定成功後使用

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 1Gi
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql-wordpress-pod
spec:
  containers:
  - name: mysql
    image: mysql:5.6
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: Password1234
    volumeMounts:
    - name: mysql-volumes
      mountPath: /var/lib/mysql
  - name: wordpress
    image: wordpress:4.8-apache
    env:
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
    - name: WORDPRESS_DB_PASSWORD
      value: Password1234
  volumes:
  - name: mysql-volumes
    persistentVolumeClaim:
      claimName: nfs-pvc
```

管理者

2. 建立可用的 Volume

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: 192.168.2.31
    path: /nfs-data
```

1. 讓K8S支援

儲存設備業者

Storage Class(SC)

- Before User request a PVC, administrator need to create PV in advanced.
  - Static Provisioning

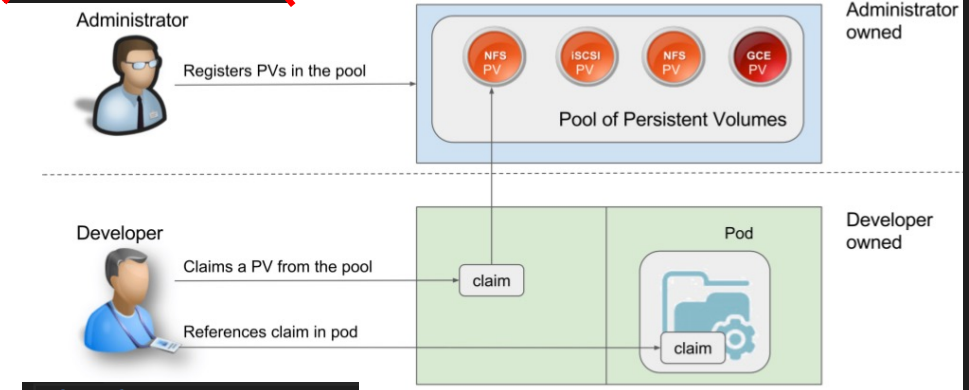
- If storage volumes can be created on-demand
  - Dynamic Provisioning
  - use storage class to enable Dynamic Provisioning
  - Provisioner is required for provisioning PVs
  - Specify SC in PVC
    - If SC is not specified in PVC, default SC is used
    - If there is no match SC, PVC is pending

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: 192.168.2.31
    path: /nfs-data
```

Create PVs on-demand

Storage Class

provisioner




```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 1Gi
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql-wordpress-pod
spec:
  containers:
  - name: mysql
    image: mysql:5.6
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: Password1234
    volumeMounts:
    - name: mysql-volumes
      mountPath: /var/lib/mysql
  - name: wordpress
    image: wordpress:4.8-apache
    env:
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
    - name: WORDPRESS_DB_PASSWORD
      value: Password1234
  volumes:
  - name: mysql-volumes
    persistentVolumeClaim:
      claimName: nfs-pvc
```

使用者

3-0. 宣告想用的儲存
3-1. 透過Provisioner建立volume
3-2. 綁定成功後使用

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 1Gi
```

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: 192.168.2.31
    path: /nfs-data
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql-wordpress-pod
spec:
  containers:
  - name: mysql
    image: mysql:5.6
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: Password1234
    volumeMounts:
    - name: mysql-volumes
      mountPath: /var/lib/mysql
  - name: wordpress
    image: wordpress:4.8-apache
    env:
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
    - name: WORDPRESS_DB_PASSWORD
      value: Password1234
  volumes:
  - name: mysql-volumes
    persistentVolumeClaim:
      claimName: nfs-pvc
```

管理者

2. 安裝provisioner

Storage Class

provisioner

1-0. 讓K8S支援
1-1. 開發Provisioner

儲存設備業者

CSI Persistence Volume

- 透過CSI, 各家廠商可以實作自己的Storage Driver

NFS 的PV 與CSI版本的NFS PV用法

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: 192.168.2.31
    path: /nfs-data
```

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-nfs
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteMany
  mountOptions:
    - hard
    - nfsvers=4.1
  csi:
    driver: nfs.csi.k8s.io
    volumeHandle: unique-volumeid
    volumeAttributes:
      server: 192.168.2.201
      share: /
```

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: new-pv
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 5Gi
  csi:
    driver: ch.ctrox.csi.s3-driver
    volumeAttributes:
      secretNamespace: default
      secretName: user-s3-secret
    volumeHandle: ogre0403
```

CSI版本的NFS 與 S3 PV用法

使用者

3-0. 宣告想用的儲存
3-1. 透過Provisioner建立volume
3-2. 綁定成功後使用

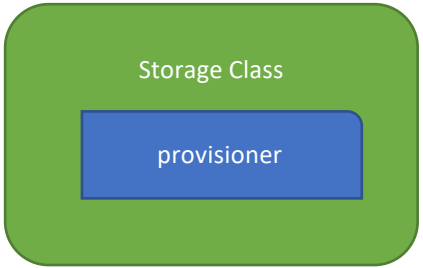
```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 1Gi
```

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: 192.168.2.31
    path: /nfs-data
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql-wordpress-pod
spec:
  containers:
  - name: mysql
    image: mysql:5.6
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: Password1234
    volumeMounts:
    - name: mysql-volumes
      mountPath: /var/lib/mysql
  - name: wordpress
    image: wordpress:4.8-apache
    env:
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
    - name: WORDPRESS_DB_PASSWORD
      value: Password1234
  volumes:
  - name: mysql-volumes
    persistentVolumeClaim:
      claimName: nfs-pvc
```

管理者

2. 安裝provisioner



儲存設備業者

1-0. 開發csi driver, 不需整合至K8S
1-1. 開發Provisioner

Summary

- Pod透過volume持久化資料
- 透過PVC，提供volume的抽象概念， PVC需綁定至底層的一個PV
  - 使用者不需要了解Volume是從那裡來的
  - PV需事先由管理者建立
- 透過storage class與Provisioner, 可以自動產生PV
- 透過CSI, 設備廠商可以自行擴充讓K8S支援新的產品

Exercise

- 撰寫yaml檔，部署nginx
  - 建立一個deployment.yaml，使用nginx image
  - svc.yaml, 採用nodeport
  - 可以由本機連到NGINX的deployment