Introduzione a Kubernetes

Orchestrator for containers

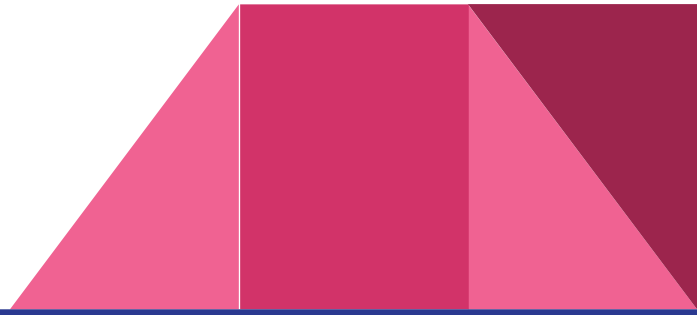
Querci Torello
torello@sighup.io

Container – questi sconosciuti

In origine avevamo le macchine reali

Non ci sono problemi di dipendenza dal momento che ciascuna macchina ha la sua versione del SO e del SW sul quale si appoggiano i vari applicativi

Ma:
- ho tante macchine da gestire, aggiornare, manutenere
- più macchine ho e più facile che qualcuna si rompa
- ciascuna macchina costa
- consumano corrente e di solito sono scariche

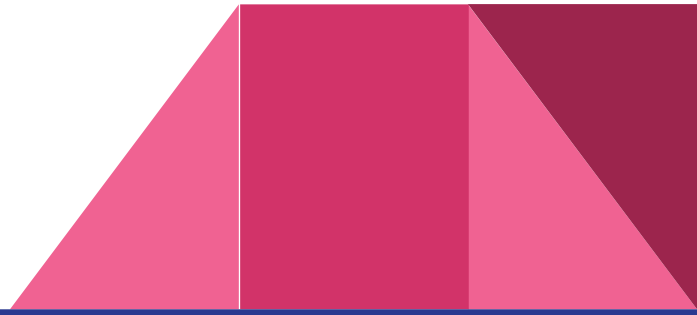


Container – questi sconosciuti – parte II

Poi arrivano le macchine virtuali

Risolvono il problema della moltiplicazione delle macchine in quanto uno stesso HW consente di comportarsi come se fosse più macchine allo stesso tempo.

Ma:
- ho tante macchine (virtuali) da gestire, aggiornare, manutenere
- ciascuna macchina reale fa girare più macchine virtuali ma per ciascuna di queste ho un sistema operativo in esecuzione

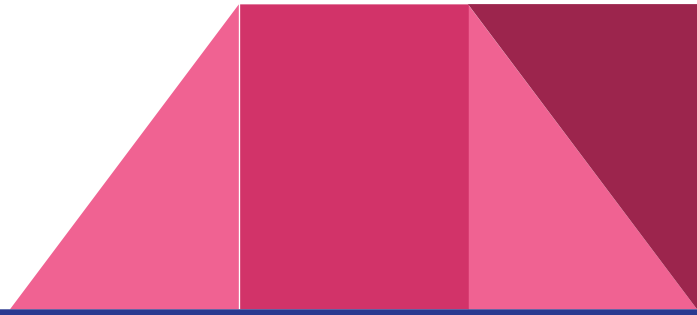


Container – questi sconosciuti – parte III

Arrivano i container

I containers sono delle porzioni di spazi di sistema nei quali, le applicazioni e le proprie dipendenze, vengono eseguite con risorse (CPU, RAM, block I/O e network) isolate pur mantenendo comune il kernel.

Si può dire che i container stanno alle macchine virtuali come i thread stanno ai processi



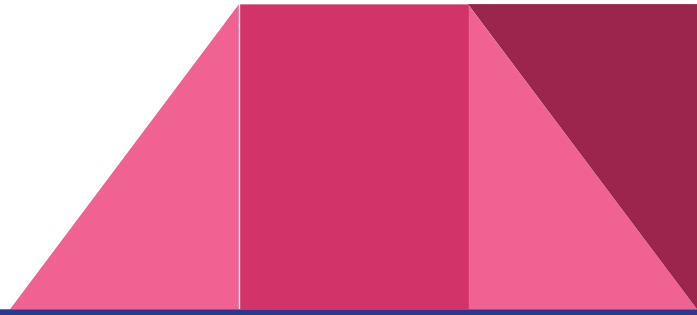
Cosa è Kubernetes

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery.

(https://kubernetes.io/)

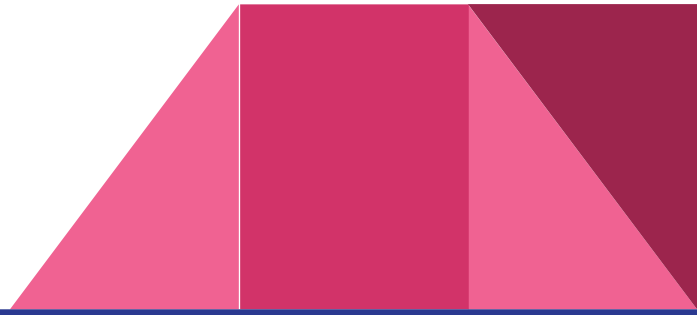
E’ la terza generazione di orchestrator sviluppata da Google basata sull’esperienza di Borg e Omega

E’ stato donato alla CNCF per assicurare la vendor neutrality



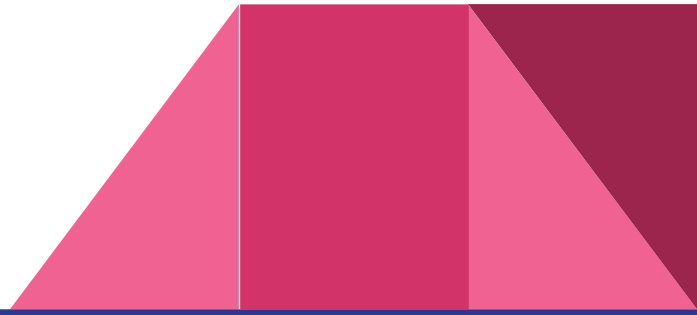
Cosa Kubernetes può fare per me?

- Mantenere la tua applicazione up e running ad ogni costo
- Mantenere la tua applicazione up e running ad ogni costo
- Mantenere la tua applicazione up e running ad ogni costo
- Mantenere la tua applicazione up e running ad ogni costo



Cosa Kubernetes può fare per me?

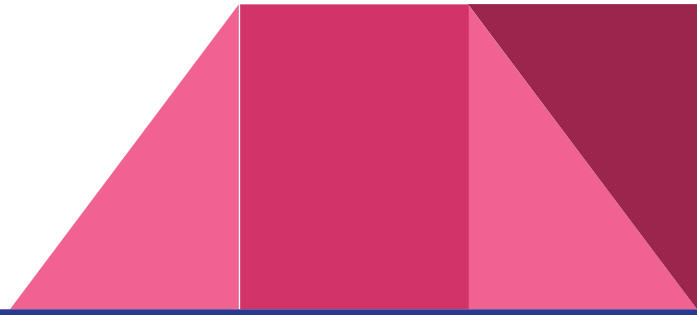
- Gestire la messa in esecuzione di un container specificando l’immagine Docker da usare
- Raggruppare più container in un’unica unità logica (POD) che condivide lo stesso stack di rete
- Monitorare l’esecuzione di dei miei container e in caso di crash provvedere a farli ripartire
- Monitorare che i miei container rispondano a delle richieste e nel caso provvedere a farli ripartire
- Gestire la scalabilità up and down dei container (ops POD)



Cosa Kubernetes può fare per me? - parte II

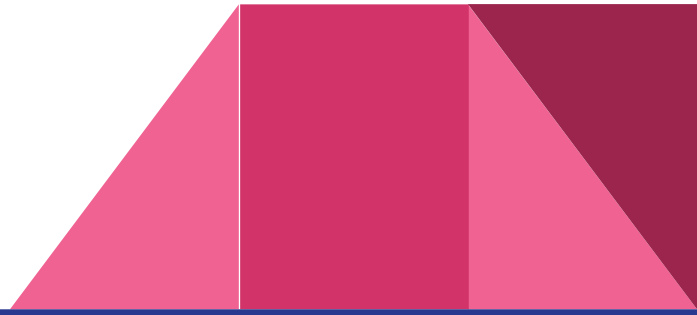
- Gestire le risorse di Storage tramite i Persistent Volume e Persistent Volume Claim
- Effettuare il discovery dei servizi in esecuzione sul cluster (DNS)
- Pubblicare all’esterno servizi in modo che siano accessibili all’esterno del cluster
- Avere ambienti multipli paralleli che vivono di vita propria (Namespace)

- Il tutto senza dovermi preoccupare di quale macchina esegue cosa



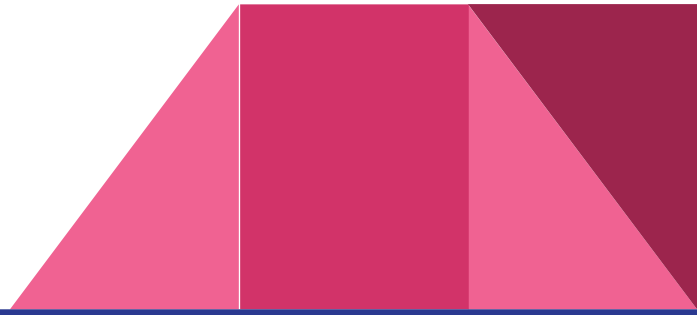
Come interagisco con Kubernetes

- **kubectl** è la parola chiave per accedere all’intero cluster
- posso specificare le azioni sia tramite riga di comando che tramite file JSON o YAML
- esiste una dashboard per interagire con kubernetes tramite web

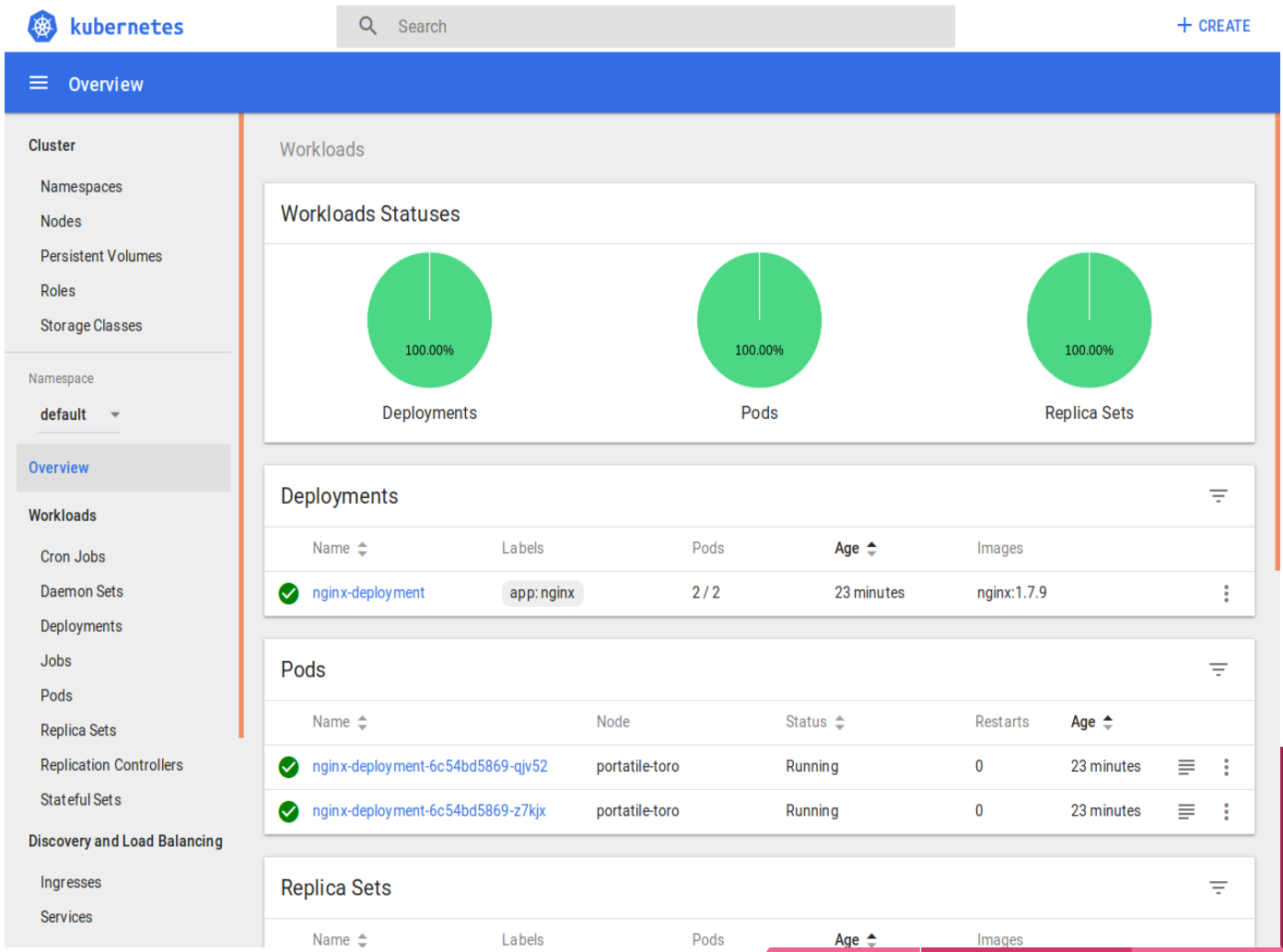


Come interagisco con Kubernetes - **kubectl**

```
# kubectl get po
NAME                                READY     STATUS    RESTARTS   AGE
nginx-deployment-6c54bd5869-qjv52   1/1       Running   0          17m
nginx-deployment-6c54bd5869-z7kjx   1/1       Running   0          17m
```



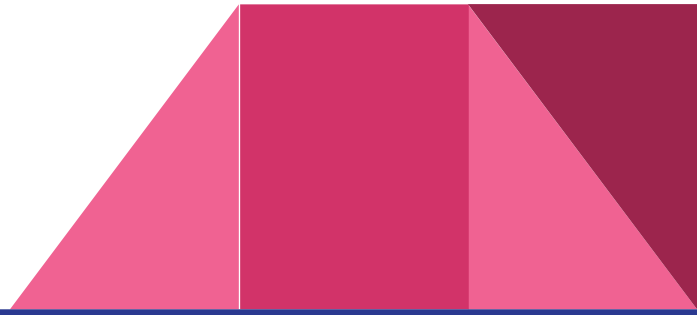
Come interagisco con Kubernetes - Dashboard



Dove gira Kubernetes

Kubernetes è agnosico sul tipo di HW reale o virtuale può girare ovunque:

- Bare metal
- Virtual Machine
- Private Cloud
- Public Coud (offerto dai maggiori player come Google, Microsoft e Amazon)



Concetti base - POD

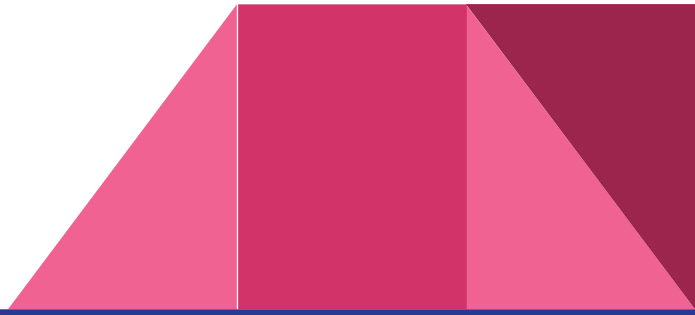
Un Pod è il mattone fondamentale di Kubernetes. Un Pod è l’unità più piccola e più semplice nel modello a oggetti di Kubernetes. Un Pod rappresenta un processo in esecuzione sul cluster.

Un Pod incapsula un container applicativo (o più container), risorse di storage, un indirizzo IP univoco, e quello che serve affinchè il container possa essere eseguito. Può consistere in un solo container o più container che condividono risorse.

I container di un Pod sono in esecuzione sulla stessa macchina e condividono la stessa rete

Concetti base – POD – esempio

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx

spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
```



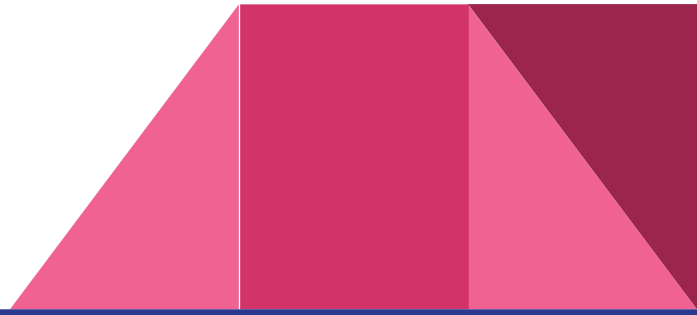
Concetti base – POD – esempio

```
# kubectl get po
NAME      READY     STATUS    RESTARTS   AGE
nginx     1/1       Running   0          17m
```

Concetti base - service

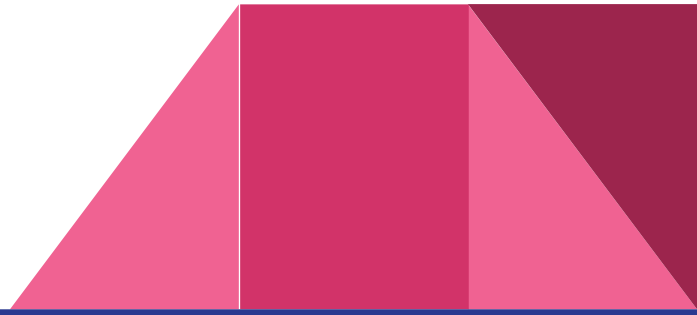
Un **service** Kubernetes è una astrazione che devinisce una insieme logico di Pod e una politica tramite la quale accedere ad essi.

L’insieme dei Pod individuati da un **service** è in genere determinata da un Label Selector.



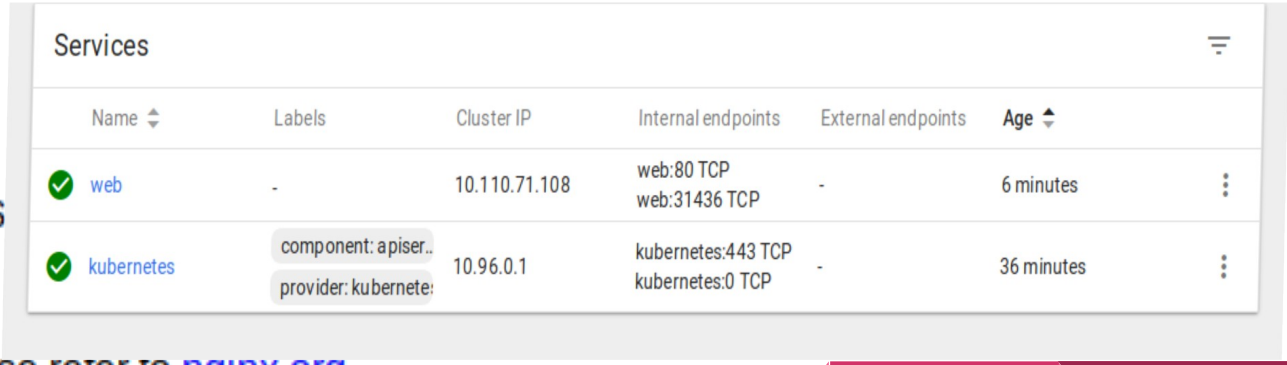
Concetti base – service – esempio

```
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  ports:
  - port: 80
  selector:
    app: nginx
  type: NodePort
```



Concetti base – service – output

```
# kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        33m
web          NodePort    10.110.71.108   <none>        80:31436/TCP   4m
```



Concetti base - ReplicaSet

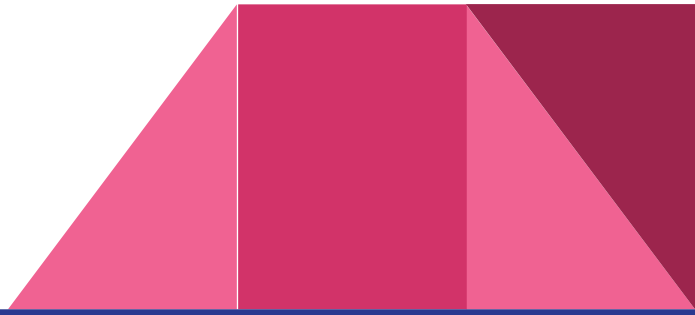
Un **ReplicaSet** assicura che un determinato numero di POD sono in esecuzione in un dato momento.

```
# kubectl get rs
NAME                DESIRED   CURRENT   READY     AGE
nginx-deployment    2         2         2         34s
```

Un Pod definito manualmente non ha questa caratteristica. Se muore nessuno lo fa ripartire. Un ReplicaSet con Replicas=1 controlla che il Pod sia su e nel caso in cui muoia lo fa ripartire.

Concetti base – ReplicaSet – esempio

```
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
  labels:
    app: nginx
  name: nginx-deployment
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:1.7.9
        imagePullPolicy: IfNotPresent
        name: nginx
        ports:
        - containerPort: 80
          protocol: TCP
```

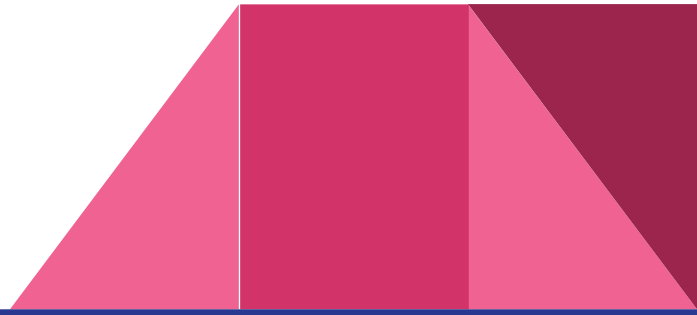


Concetti base - Deployment

Tramite un **Deployment** viene descritto lo stato desiderato ed il Deployment controller effettua i cambiamente necessari per passare dallo stato attuale allo stato desidestato.

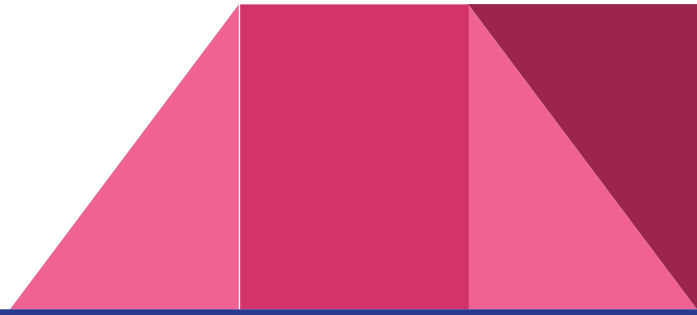
Tramite i Deployment vengono generati in automatico, o aggiornati, i ReplicaSet necessari. I ReplicaSet generati da un Deployment non devono essere manipolati manualmente.

Tramite i Deployment è possibile aggiornare un sistema senza dare interruzione di servizi



Concetti base – Deployment – esempio

```
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```



Concetti base – Deployment

```
# kubectl get deployment
NAME               DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2         2         2            2           7s
```

```
# kubectl get rs
NAME                          DESIRED   CURRENT   READY     AGE
nginx-deployment-5964dfd755   1         1         0         7s
nginx-deployment-6c54bd5869   3         3         3         2m
```

```
# kubectl get rs
NAME                          DESIRED   CURRENT   READY     AGE
nginx-deployment-5964dfd755   3         3         3         2m
nginx-deployment-6c54bd5869   0         0         0         4m
```

Concetti base – Deployment

L’apply di un Deployment genera tutti i passi necessari per passare da una configurazione running ad un’altra configurazione running.

```
# kubectl rollout status deployment/nginx-deployment
deployment "nginx-deployment" successfully rolled out
```

Posso fare l’undo di un aggiornamento del deployment

```
# kubectl rollout undo deployment/nginx-deployment
deployment "nginx-deployment"
```

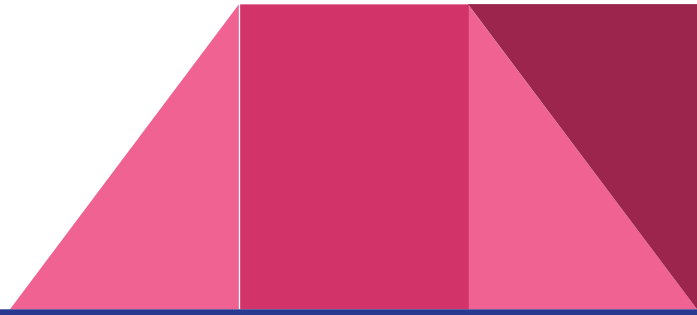
Altre tipologie di Pod

- StatefulSets – gestisce la distribuzione e lo scaling di un insieme di Pod. Fornisce la garanzia circa l'ordinamento e l'unicità dei Pod.

- DemonSet – assicura che su tutti (o alcuni) nodi ci sia in esecuzione una copia di un determinato Pod. Quando un nodo viene aggiunto al cluster I Pod sono aggiunti al Nodo automaticamente.

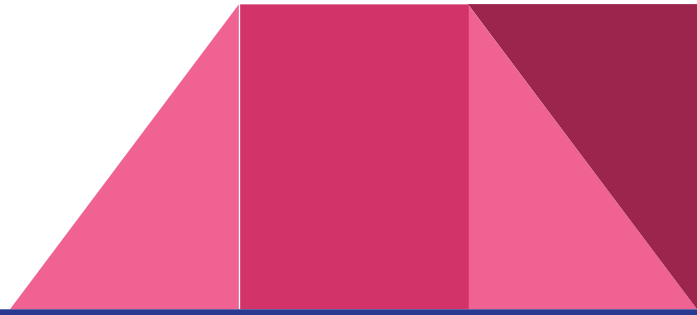
- Job – un Job crea uno o più Pod e si assicura che un determinato numero di questi termini con successo.Quanto viene raggiunto il numero richiesto di completamenti, il Job stesso è completo.

- CronJob – è l'equivalente di un crontab solo per Pod



Gestione della configurazione

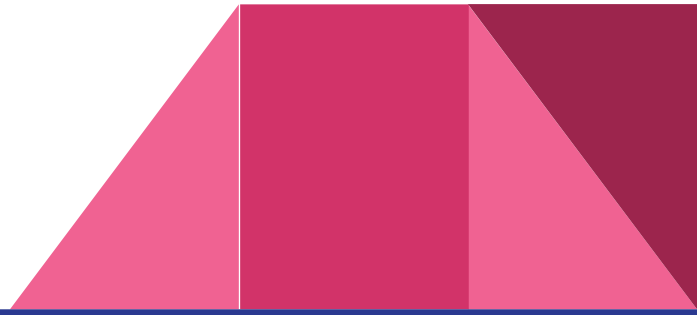
- ConfigMap – sono delle coppie chiavi-valore a cui i Pod possono accedere sia come se fossero variabili d’ambiente che se fossero file
- Secret – sono analoghi ai ConfigMap solo che i dati sono crittografati (o meglio sono in Base64)



Concetti base – Configmap

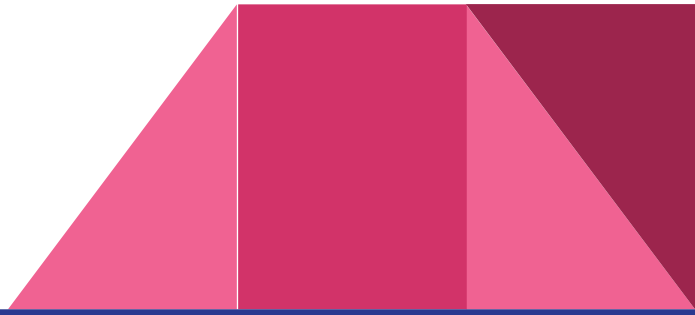
```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-config
  labels:
    app: nginx-config

spec:
  containers:
  - name: nginx-config
    image: nginx:1.7.9
    ports:
    - containerPort: 80
    env:
    - name: GAME_TITLE
      valueFrom:
        configMapKeyRef:
          name: game-config
          key: title
    volumeMounts:
    - name: g-config
      mountPath: /config
  volumes:
  - name: g-config
    configMap:
      name: game-config
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: game-config
  namespace: default
data:
  game.properties: |
    enemies=aliens
    lives=3
    enemies.cheat=true
    enemies.cheat.level=noGoodRotten
    secret.code.passphrase=UUDDLRLRBAS
    secret.code.allowed=true
    secret.code.lives=30
  ui.properties: |
    color.good=purple
    color.bad=yellow
    allow.textmode=true
    how.nice.to.look=fairlyNice
  title: Best Game
```



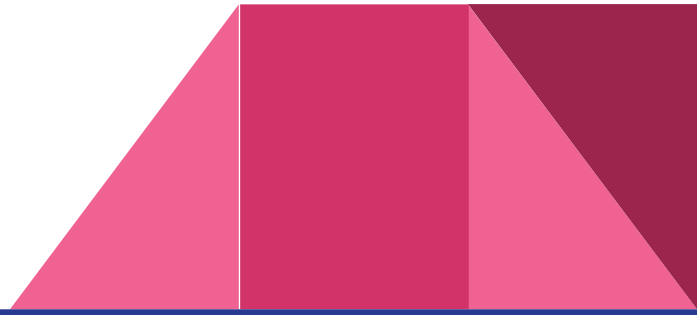
Concetti base – Configmap in azione

```
# kubectl exec -it nginx-config bash
root@nginx-config:/# ls /config
game.properties  title    ui.properties
root@nginx-config:/# echo "$GAME_TITLE"
Best Game
root@nginx-config:/#
```



Persistenza dei dati

- Volume – a ciascun pod viene associato un volume esterno alla macchina dove memorizzare permanentemente i dati
- PersistentVolume e PersistentVolumeClaim – consente una gestione più asttratta delle risorse di storage
- Dynamic Volume Provisioning – consente di creare al volo i volumi di storage Job



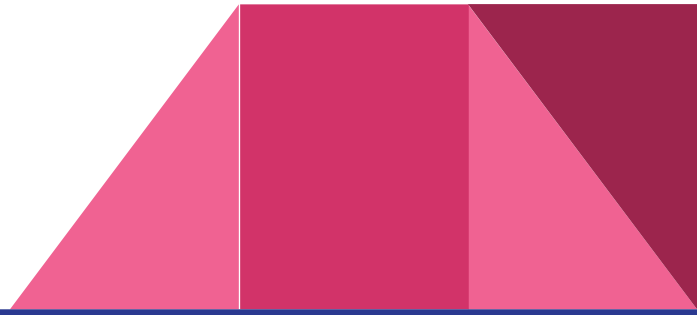
Namespace

Kubernetes supporta cluster virtuali sullo stesso cluster fisico tramite i namespace.

Di default sono presenti tre namespace:

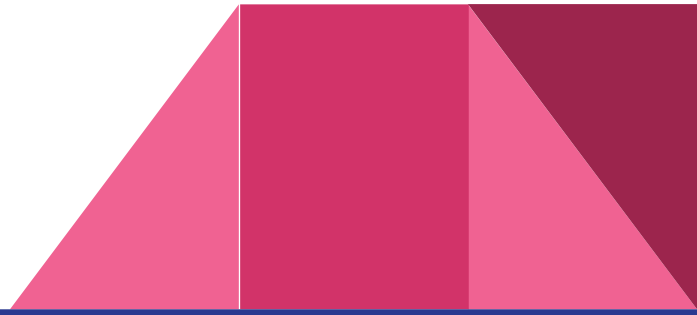
- default
- kube-system
- kube-public

Ciascun utente può aggiungerne a piacimento



Come provarlo

- Minikube
- Cluster com Macchine virtuali – immagini pronte con il progetto Atomic di RedHat
- Piccoli cluster di prova sulle varie piattaforme GCP – AWS – Azur
- HW fisico



Domande?

Q & A

Grazie a tutti !!!!

Querci Torello
torello@sighup.io