Cloud Computing (LAB)

HANDS ON  kubernetes

Giuseppe Bisicchia
giuseppe.bisicchia@phd.unipi.it

Dipartimento di Informatica, Università di Pisa

Cosa faremo?

- Sperimenteremo con *Kubernetes* scoprendo come **creare**, **lanciare** e **gestire** un **Pod**.
- Impareremo a gestire **multiple repliche** di un'applicazione.
- Capiremo come rendere un'applicazione accessibile sia **all'interno** che **all'esterno** di un **cluster**.

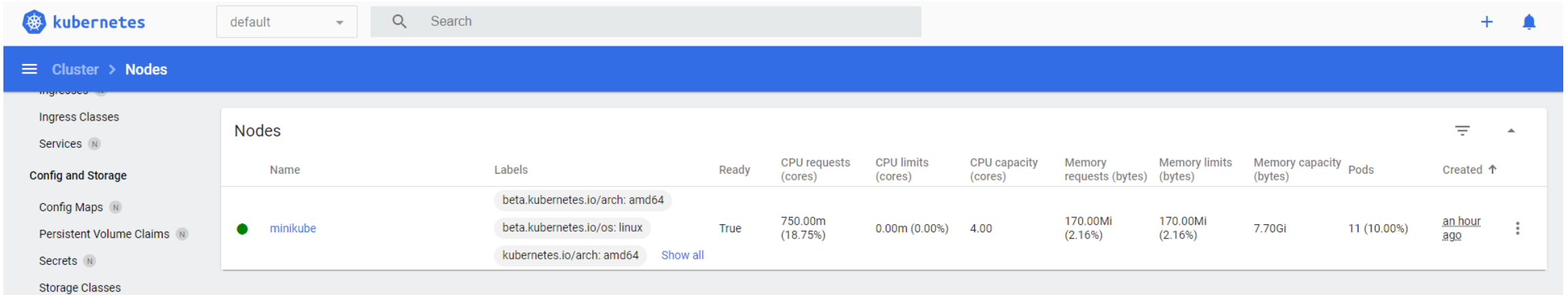
- **Suggerimento**: durante il laboratorio usate «*minikube dashboard*» per capire visualmente cosa sta accadendo.

Cluster

- Per iniziare abbiamo bisogno di un **cluster** su cui lavorare.
- Per i nostri scopi andrà benissimo un **cluster locale** formato da un **solo nodo** che creeremo con «*minikube start*».
- Per installare minikube seguire questa guida fino al **punto 3**: https://minikube.sigs.k8s.io/docs/start/

Cluster

- Possiamo verificare la presenza del nodo tramite la «minikube dashboard»



Esercizio: Primo Pod

- Create un Manifesto Kubernetes per lanciare un Pod contenente un solo container con immagine «*mhausenblas/simpleservice:0.5.0*»

Esercizio: Primo Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: simple-pod
spec:
  containers:
  - name: simple-pod
    image: mhausenblas/simpleservice:0.5.0
```

Interagiamo con il Pod

- Per lanciare il Pod eseguite: «*kubectl apply -f NOME_FILE*»
- Eseguite «*kubectl get pod*» per avere la lista dei Pod in esecuzione.
- Con «*kubectl logs NOME_POD*» possiamo ottenere informazioni sui log del Pod.
- «*kubectl describe pod NOME_POD*» è un comando molto potente che ci consente di avere numerose informazioni sul Pod (e.g., il suo IP interno).

Interagiamo con il Pod

- Eseguite «*minikube ssh*» per entrare in una shell con cui interagire con i container. Provate il comando «*curl IP_POD:9876/info*», che accade?
- Con «*kubectl get all*» possiamo visualizzare tutte le risorse create da Kubernetes, è presente altro oltre al nostro Pod?
- Eliminiamo il Pod con il comando «*kubectl delete pod NOME_POD*»

Lanciamo più repliche del nostro Pod

- Per quanto sia possibile gestire direttamente i Pod, questa **non è mai l'opzione consigliata**.
- Infatti i Pod possono essere **uccisi** in **qualsiasi momento** da Kubernetes ed esistono varie **astrazioni** per rendere più semplice ed efficace la gestione delle nostre applicazioni.

Esercizio: Più repliche

- Lanciate **due repliche** del precedente Pod.
- **Suggerimento**: per farlo create un nuovo Manifesto Kubernetes con «kind: Deployment».

Esercizio: Più repliche

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: simple-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: simple-deployment
  template:
    metadata:
      labels:
        app: simple-deployment
    spec:
      containers:
      - name: simple-pod
        image: mhausenblas/simpleservice:0.4.0
        ports:
        - containerPort: 9876
```

Interagiamo con il Deployment

- Per lanciare il deployment basta «*kubectl apply -f NOME_FILE*»
- Con «*kubectl get deployment*» possiamo vedere il contenuto del deployment.
- Possiamo osservare i Pod presenti con «*kubectl get pod*»

Interagiamo con il Deployment

- Usate «*kubectl get all*», Cosa vi aspettate di vedere? E' stato creato altro? Perché?
- Per modificare il numero di repliche basta modificare il manifesto e ri-eseguire apply. Provate prima ad aumentare e poi diminuire il numero di repliche e osservata sulla dashboard cosa accade. Alternativamente potete usare «*kubectl get pod --watch*» in un nuovo terminale.

Pod e Deployment

- Lanciate di nuovo il Pod singolo dell'esercizio precedente.
- Aprite la dashboard o eseguite il comando watch.
- Una volta che tutti i container sono in esecuzione lanciate il comando «*kubectl delete pod --all*» per eliminare tutti i Pod, che accade? Perché?
- Per eliminare tutto basta lanciare «*kubectl delete deployment,rs,pod --all*»

Perché usare i servizi?

- Al momento per accedere alla nostra applicazione dobbiamo **conoscere l'IP** di uno **specifico Pod** ed interagire direttamente con esso, anche se sono presenti più repliche.
- Inoltre, nel caso che un Pod non sia più disponibile dobbiamo usare l'IP di un altro Pod.
- Infine, quell'IP **non è accessibile al di fuori** del cluster.

Creiamo un servizio

- Con un **servizio** possiamo accedere i nostri Pod sia **all'interno** che **all'esterno** di un servizio, inoltre ci permetterà di accedere ad un deployment senza dover specificare un particolare Pod.
- In questo esperimento utilizzeremo il deployment precedente ed un ulteriore Pod. Il Pod aggiuntivo ci servirà per poter accedere ai Pod del deployment.

Creiamo un servizio

- Per iniziare lanciamo il deployment precedente con apply.
- Ora lanciate il Pod presente su moodle e successivamente eseguite il comando «*kubectl exec -it bash -- /bin/bash*» che permetterà di aprire una shell bash all'interno del container.
- All'interno della bash eseguite «*apt update && apt install dnsutils curl*» per installare dei tool che ci serviranno a breve.
- Utilizzate ora curl come visto precedentemente per accedere al deployment all'indirizzo «*IP_POD:9876/health*», come si deve fare?

Esercizio: Creiamo un Servizio

- Con i Servizi possiamo **disaccoppiare** l'acceso ad un deployment evitando di dover specificare un Pod specifico ed il relativo IP.
- Create quindi il manifesto di un servizio (i.e., «kind: Service») collegato al nostro deployment.

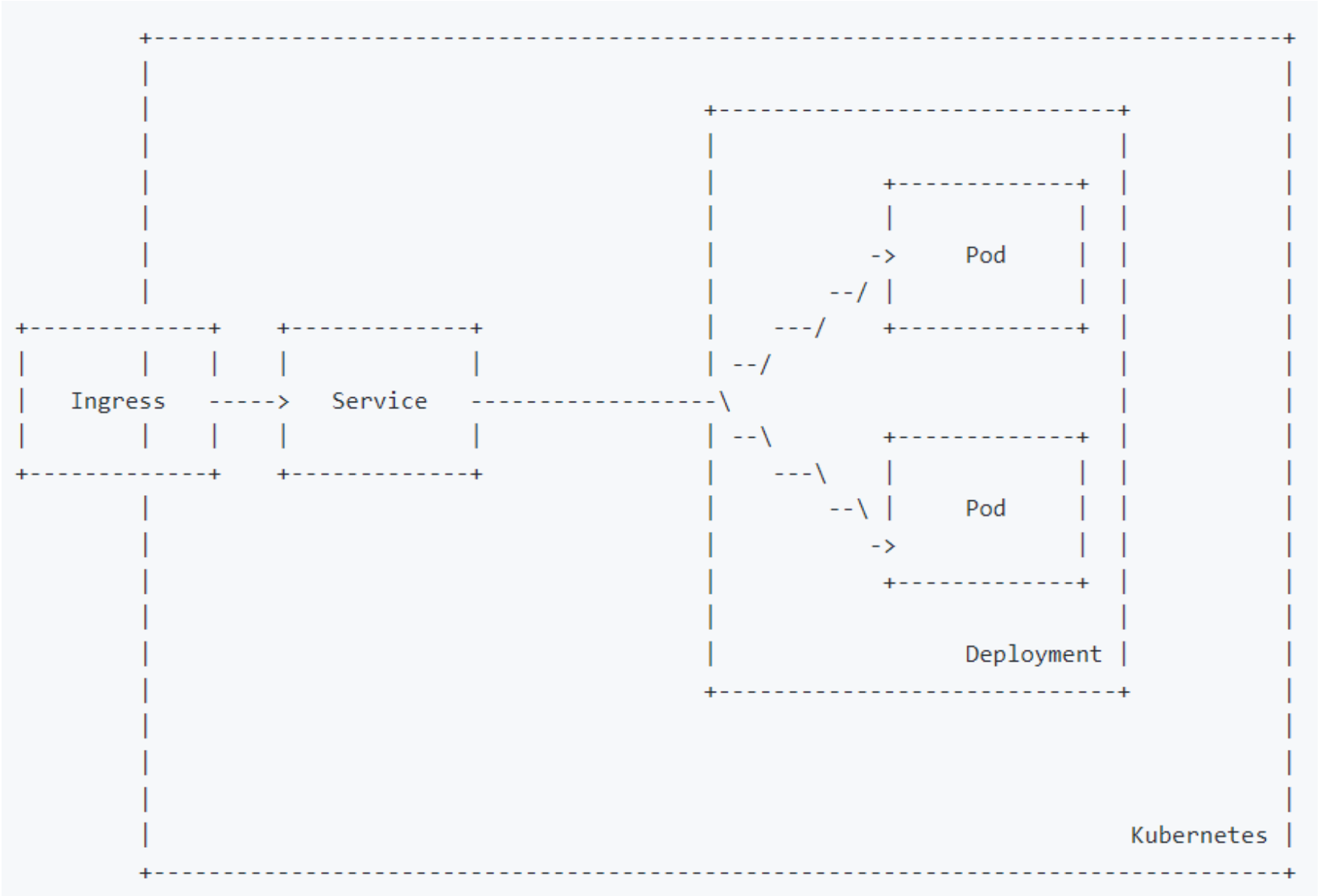
Esercizio: Creiamo un Servizio

```yaml
apiVersion: v1
kind: Service
metadata:
  name: simple-internal-service
spec:
  ports:
  - port: 80
    targetPort: 9876
  selector:
    app: simple-deployment
```

Accediamo al Servizio

- Per lanciare il servizio basta usare il comando apply.
- Una volta fatto tramite la bash creata precedentemente eseguiamo il comando «*nslookup NOME_SERVIZIO*» per ottenere l'IP.
- Eseguiamo ora curl all'URL «*IP_SERVIZIO/health*» (ricordatevi la porta specificata nel manifesto).
- Cosa accade se proviamo ad accedere al servizio al di fuori del cluster?

Global Overview



Esercizio: Accediamo al Servizio dall'esterno

- Un servizio permette di disaccoppiare l'accesso ad un deployment rispetto alle repliche specifiche. Per rendere però il servizio accessibile esternamente abbiamo bisogno di un «Ingress» da connettere al servizio.
- Prima di tutto dobbiamo installare alcuni addons per minikube con «*minikube addons enable ingress*» e «*minikube addons enable ingress-dns*».
- Esercizio: create un manifesto «*kind:Ingress*» associato al nostro servizio.

Esercizio: Accediamo al Servizio dall'esterno

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple-ingress
  annotations:
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
spec:
  backend:
    serviceName: simple-internal-service
    servicePort: 80
```

Esercizio: Accediamo al Servizio dall'esterno

- Ora non resta che lanciare l'Ingress con apply e ottenere l'IP del nostro cluster con «minikube ip».
- Adesso possiamo accedere al nostro servizio e quindi al nostro deployment (che gestisce le repliche del nostro Pod) dall'esterno all'IP del cluster e con la porta che abbiamo specificato nel manifesto.

Esercizio

Partendo dalle due immagini

- https://hub.docker.com/r/yeasy/simple-web (listens on port **80**)
- https://hub.docker.com/r/scottyc/webapp (listens on port **3000**)

scrivere

- un file **pod.yaml** per configurare il pod che gestisce le due immagini
- un file **service.yaml** per esporre il pod come servizio raggiungibile

Per eliminare tutto digitare «kubectl delete ingress,service,deployment,rs,pod --all» e «minikube delete --all» per eliminare il cluster minikube.

Documentazione

- https://kubernetes.io/docs/tasks/configure-pod-container/static-pod/
- https://kubernetes.io/docs/concepts/workloads/pods/
- https://kubernetes.io/docs/concepts/services-networking/service/

More details

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