Kubernetes pour de vrai

Cedric Hauber - WeScale

wescale
Be a cloud expert

Par où commencer ?

1. Pré-requis :
   - Vagrant
   - VirtualBox
   - Git
2. Cloner le projet: https://github.com/WeScale/kubernetes-handson.git
3. cd step-01
4. ./run-cluster-vagrant.sh

Agenda

- Cloud 1.0 vs Cloud 2.0
- Kubernetes c'est quoi ?
- Par où commencer ?
- Les composants
- La Haute Disponibilité
- Et dans les nuages ?
- Et si on s'amusait un peu à casser tout ça
- Les Outils
- Conclusion

Cloud 1.0

- Des machines virtuelles, derriere des load balancers
- Chef, Puppet, Salt…
- Environnements lourds, consommateurs de resources
- Maitrise difficile de la charge et de la haute disponibilité

Cloud 2.0

- Des conteneurs légers, rapides
- Une architecture micro-services, stateless pour une montée en charge rapide
- Une abstraction complète des machines physiques

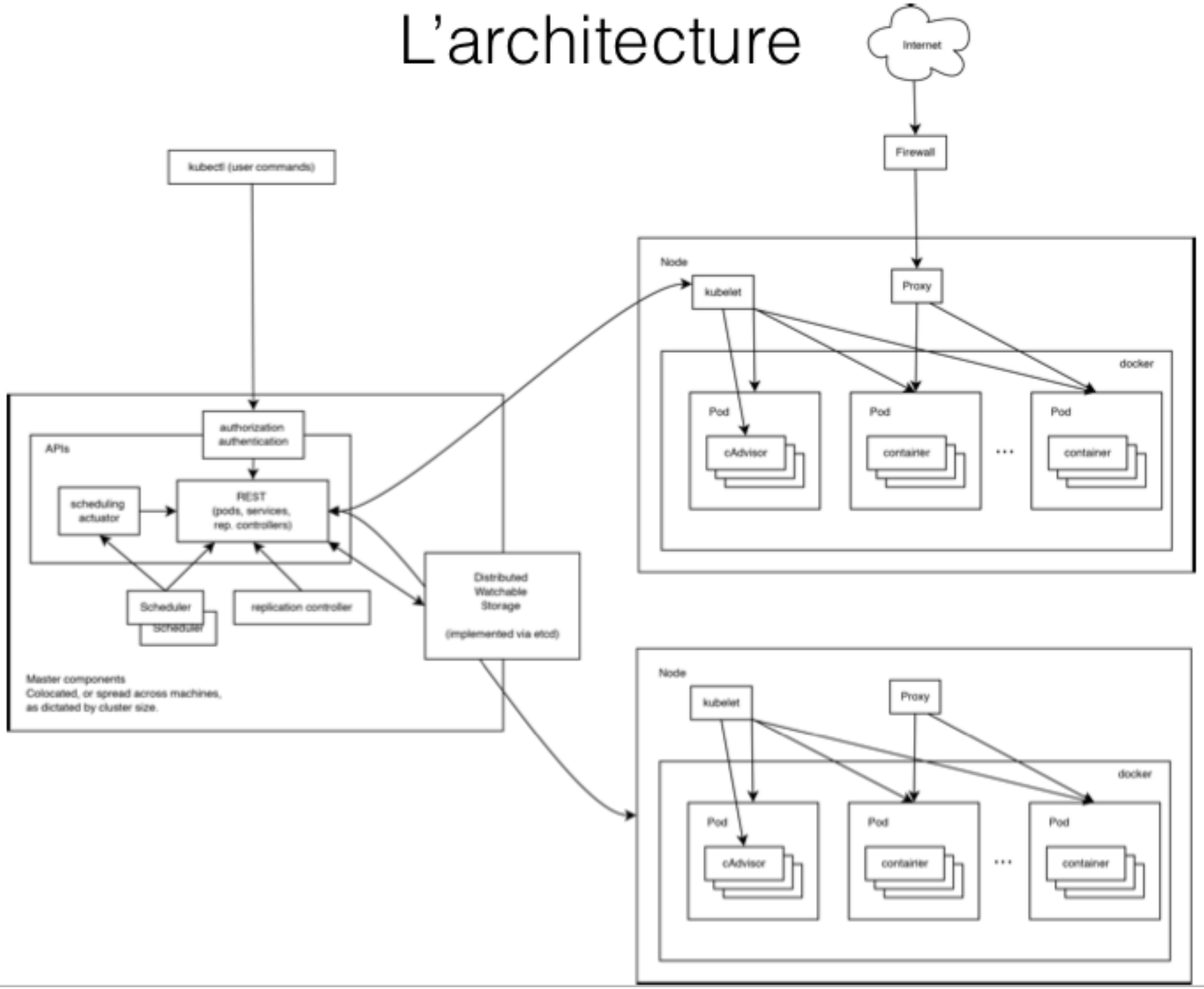
Kubernetes c'est quoi ?

- On dit /koo-ber-nay'-tace/
- Le « maitre de navigation »
- Ses origines: Le projet BORG
- Un orchestrateur de conteneurs Docker ou Rocket
- Il apporte une solution concrète à la gestion d'architectures micro-services

Les Composants

- etcd: Base de données distribuée
- kube-apiserver: Serveur REST
- kube-scheduler: Gestion des resources
- kube-controller-manager: Gestion des Controllers
- kubelet: Gestion d'un noeud
- kube-proxy: Proxy (pour les services)

L'architecture



Les Pods

- Brique élémentaire
- Lie plusieurs conteneurs entre eux
- A sa propre adresse IP

L’indispensable Hello World

1. kubectl create -f hello-world.yaml

2. Rendez-vous sur http://172.17.8.x

Quelques commandes :

- kubectl get pods : Liste les pods
- kubectl logs hello-world : Affiche les logs

hello-world.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: hello-world
spec:
  containers:
    - name: hello-world
      image: tutum/hello-world
      ports:
        - name: www
          containerPort: 80
          hostPort: 80
```

Les Labels

- Permettent d'identifier et de sélectionner des ressources
- Concernent toutes les resources
- C'est un élément central de Kubernetes

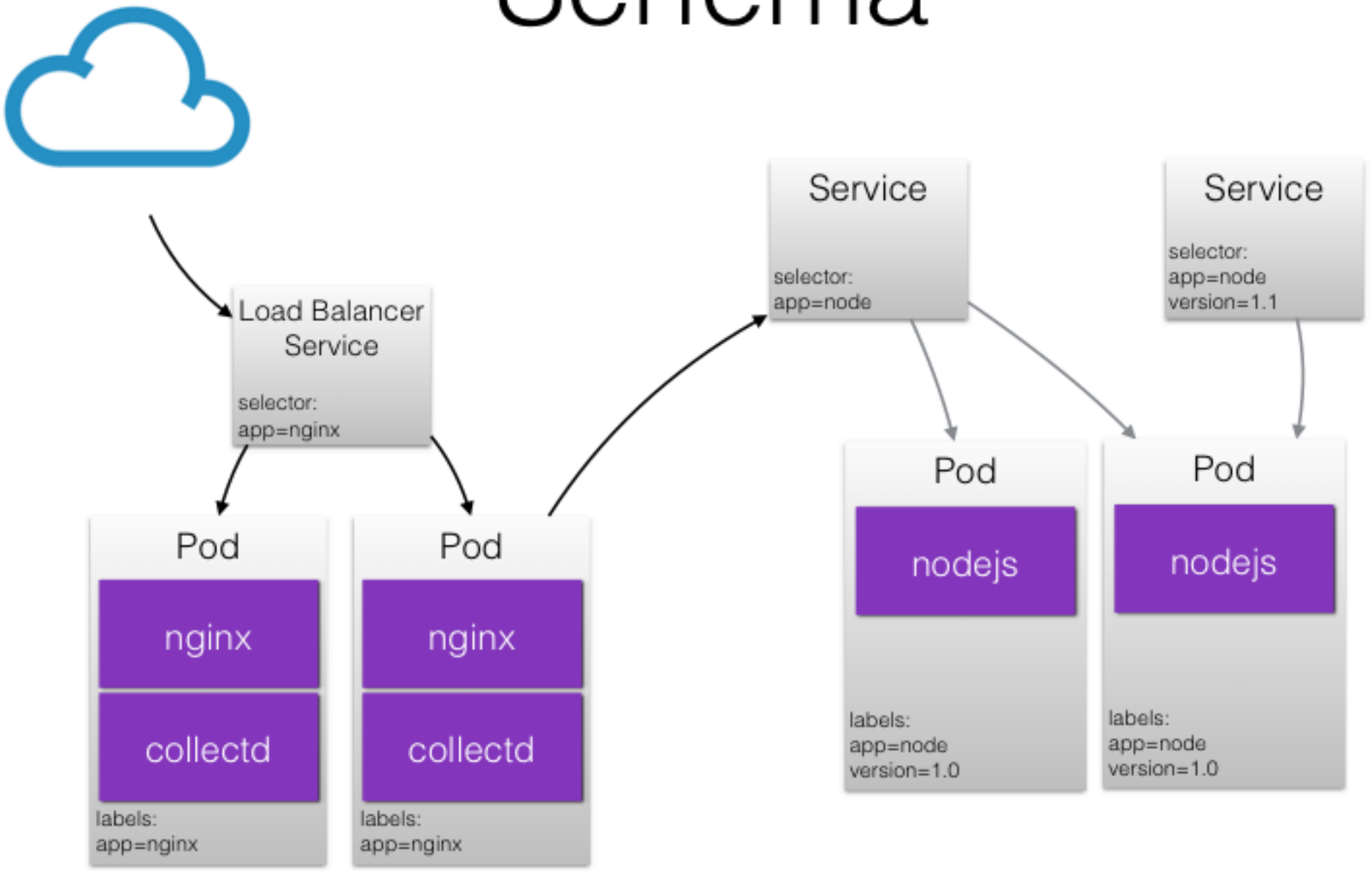
Les Services

- Jouent le rôle de point d'entrée vers un ou plusieurs Pods
- Sont accessibles depuis les autres conteneurs du même namespace
- Utilisent les labels pour découvrir les pods ciblés par le service
- Trois types de services :
    - ClusterIP
    - NodePort
    - LoadBalancer
    - Headless

Découverte de service

- Les services ne changent jamais d'ip
- Ils sont exposés grâce à des variables d'environnement
- Un addon, KubeDNS permet d'accéder aux services grâce au DNS interne de Kubernetes

Schema


Load Balancer Service
selector:
app=nginx
Service
selector:
app=node
Service
selector:
app=node
version=1.1
Pod
nginx
collectd
labels:
app=nginx
Pod
nginx
collectd
labels:
app=nginx
Pod
nodejs
labels:
app=node
version=1.0
Pod
nodejs
labels:
app=node
version=1.0

hello-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: hello-service
spec:
  type: NodePort
  selector:
    demo: hello
  ports:
    - protocol: "TCP"
      port: 80
      nodePort: 30000
```

Exemple

1. cd ../step-02
2. kubectl delete hello-world
3. kubectl create -f hello-world.yaml
4. kubectl create -f hello-service.yaml

Les Controllers

- Ils contrôlent la création / suppression des Pods
- Il en existe un seul type pour le moment, le ReplicationController
- Les ReplicationControllers s'assurent qu'un nombre bien précis de Pods fonctionnent continuellement
- C'est ce qui nous permet de « scaler »

Exemple

1. cd ../step-03
2. kubectl delete pod hello-world
3. kubectl create -f hello-world-controller.yaml

wescale

hello-world-controller.yaml

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: hello-world-controller
spec:
  replicas: 2
  selector:
    demo: hello
  template:
    metadata:
      name: hello-world
      labels:
        demo: hello
    spec:
      containers:
        - name: hello-world
          image: tutum/hello-world
          ports:
            - name: www
              containerPort: 80
```

wescale

Jouons un peu

- Lister les pods:

  kubectl get pods

- Supprimer un pod :

  kubectl delete pod *name*

- Scaler un ReplicationController :

  kubectl scale rc *name* —replicas=*nb*

wescale

Les Health Checks

- Plusieurs types de probes :
  - Readyless: Permet d'indiquer a Kubernetes que le container est prêt
  - Liveness: Permet de surveiller l'état de l'application
  - Exec: Execute une commande à l'intérieur d'un conteneur pour déterminer son état

wescale

Les Volumes

Les Volumes

- emptyDir : Un répertoire partagé disponible pendant toute la durée de vie d'un Pod
- hostPath : Un répertoire sur la machine hôte
- gcePersistentDisk : Un volume persistant Google Container Engine
- awsElasticBlockStore : Un volume persistant AWS
- nfs : Un partage NFS
- iscsi : Un volume iSCSI
- flocker : Un dataset Flocker
- glusterfs : Un volume glusterfs
- rbd : Un Rados block
- gitRepo : Un repository Git
- secret : Un secret Kubernetes
- persistentVolumeClaim : Une demande de mise à disposition d'un volume persistant Kubernetes

wescale

Déployons quelque chose

d'utile : Gitlab

- Un GitHub like gratuit
- Nécessite un serveur :
  - Redis
  - Postgresql
  - Omnibus

wescale

Le Réseau

Le Réseau

- Flannel
- Google Compute Engine routes
- OpenVSwitch
- L2 networks
- Weave
- Calico

wescale

Les Namespaces

Les Namespaces

- Permettent d'isoler les resources entre elles
- Ne partagent pas les services des autres namespaces
- Pour les lister : kubectl get namespaces

wescale

Les Extensions

Les Extensions

- HorizontalPodAutoscaler
- Job
- Scale
- Ingress

wescale

La Haute Disponibilité

La Haute Disponibilité

- Fondamental: Avoir un bon cluster Etcd
- Snapshot Etcd
- Plusieurs Master

wescale

Et dans les nuages ?

Et dans les nuages ?

- Google Compute Engine : Google Container Engine
- Amazon Web Services : Coreos-Kubernetes
- Microsoft Azure

wescale

Et si on s'amusait un peu à

casser tout ça !

- Lancez des Pods ou des ReplicationControllers et essayez de survivre à mon X-Wing !
- Seule convention: les Pods et les ReplicationControllers doivent avoir comme label: game: starwars
- Si vous voulez que votre nom s'affiche, pensez à rajouter le label player à vos resources
- Enfin, vos conteneurs doivent être dans l'état Running pour que votre vaisseau puisse correctement fonctionner…

wescale

Les outils

- Kubectl
- KubeUI
- Tectonic by CoreOS (in beta)

wescale

Conclusion

Kubernetes est la réponse de Google à une des problématiques que nous rencontrons avec les conteneurs Docker/Rocket :

Comment correctement les orchestrer ?

Cette solution n'a pas pour but de résoudre tous les problèmes mais fait ses preuves et a de beaux jours devant elle.

Elle a, entre autre, l'avantage d'être cloud-agnostic et donc de laisser une certaine liberté aux entreprises

Google le montre avec Google Compute Engine et de plus en plus de projets passent en production avec Kubernetes

Kube-Wars Project

https://github.com/cedbossneo/kube-stars.git