Bonjour tout le monde

http://stage2.discimus.fr/guacamole/#/
student / Pa$$w0rd $ et 0 (zéro)
Cyrille : st1
Didier : st2
Jean-Sebastien : st3
Patrice : st4
Alexandre : st5
Carole : st6

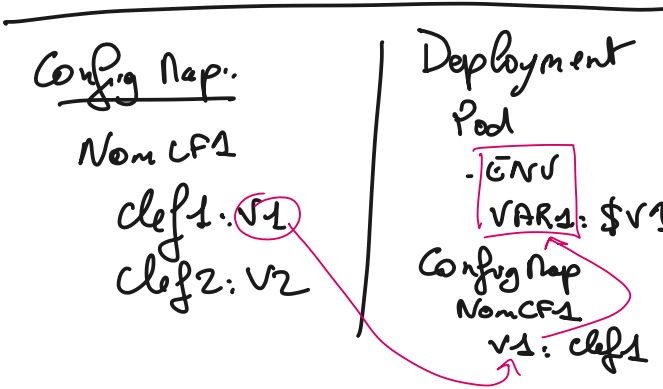
Vous travaillerez tous dans un projet qui portera le nom st1, st2, ...

Donner les droits d'executer des conteneurs root dans votre projet :
oc adm policy add-scc-to-user anyuid -z default -n <votre namespace>

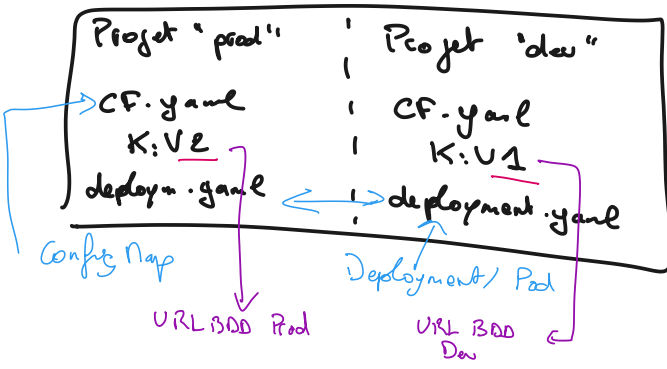
Guacamole copier/coller :
Appui sur Ctrl+Shift+alt et coller et ctrl shift alt, et coller

Préparation de votre environnement :
oc project st6 # change votre projet par défaut
oc adm policy add-scc-to-user anyuid -z default -n st6 # vous donne les droits root pour les conteneurs

Cluster Openshift en ligne :
https://learn.openshift.com/



1 Cluster pour env Prod & Dev



---

gitlab

PRJ_SRC
- Res K8s

Helm

Jenkins

Jenkins

Pipe →

Local

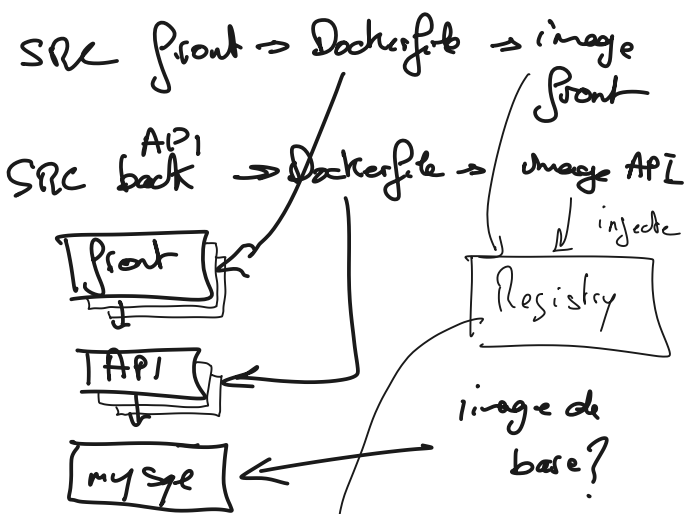
Jenkinsfile

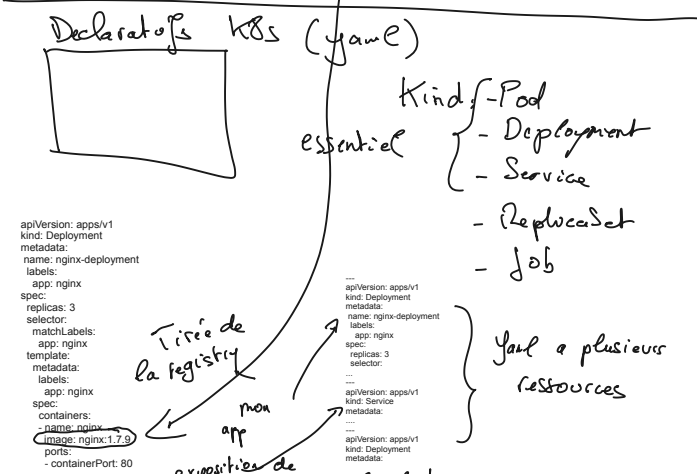
Dockerfile

SRC/ app s/ app2/

K8s/ : yaml

Res K8s






```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
...
---
apiVersion: apps/v1
kind: Service
metadata:
...
---
apiVersion: apps/v1
kind: Deployment
metadata:
```

Etat des lieux :

git (intégration)

images

Registry

Paquets Helm

Repo Helm

artifactory

Procédure fichier pour créer une chart Helm

Déploiement K8S :

oc apply -f http://

oc apply //

// //

etc etc ....

- Créer la chart

Helm install <produit>

apiVersion: The chart API version (required)
name: The name of the chart (required)
version: A SemVer 2 version (required)
kubeVersion: A SemVer range of compatible Kubernetes versions (optional)
description: A single-sentence description of this project (optional)
type: The type of the chart (optional)
keywords:
  - A list of keywords about this project (optional)
home: The URL of this projects home page (optional)
sources:
  - A list of URLs to source code for this project (optional)
dependencies: # A list of the chart requirements (optional)
  - name: The name of the chart (nginx)
    version: The version of the chart ("1.2.3")
    repository: (optional) The repository URL ("https://example.com/charts") or alias ("@repo-name")
    condition: (optional) A yaml path that resolves to a boolean, used for enabling/disabling charts (e.g. subchart1.enabled)
    tags: # (optional)
      - Tags can be used to group charts for enabling/disabling together
    import-values: # (optional)
      - ImportValues holds the mapping of source values to parent key to be imported. Each item can be a string or pair of child/parent sublist items.
    alias: (optional) Alias to be used for the chart. Useful when you have to add the same chart multiple times
maintainers: # (optional)
  - name: The maintainers name (required for each maintainer)
    email: The maintainers email (optional for each maintainer)
    url: A URL for the maintainer (optional for each maintainer)
icon: A URL to an SVG or PNG image to be used as an icon (optional).
appVersion: The version of the app that this contains (optional). Needn't be SemVer. Quotes recommended.
deprecated: Whether this chart is deprecated (optional, boolean)
annotations:
  example: A list of annotations keyed by name (optional).

HelloWorld en Helm :

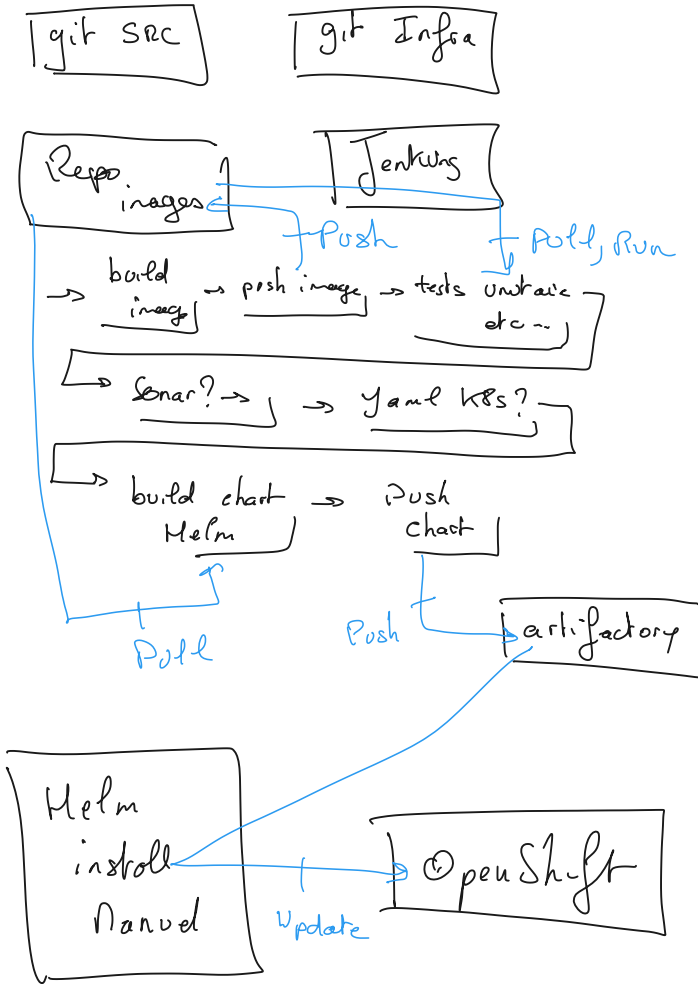
Chart.yaml:

```
apiVersion: v1
name: helloworld
description: helloworld chart that runs on kubernetes
version: 1.0.0
keywords:
  - helloworld
  - kubernetes
home: https://github.com/stakater-charts/helloworld
maintainers:
- name: Stakater
  email: stakater@aurorasolutions.io
```

Values.yaml :

```
kubernetes:
  host: https://kubernetes.default

helloworld:
  namespace: default
  image:
    name: tutum/hello-world
    tag: latest
    pullPolicy: IfNotPresent
  service:
    ingressClass: internal-ingress
```

helm install --name helloworld chartmuseum/helloworld


git SRC
git Infra
Repo images
Jenkins
Push
Poll, Run
build image
push image
tests unitaire etc ...
Sonar?
yaml K8s?
build chart Helm
Push chart
Poll
Push
artifactory
Helm install Manuel
OpenShift
Update

Helm

Jenkinsfile

helloworld/

Helm
- Chart.yaml
- Values.yaml
- templates/
  - Deployment.yaml
  - Service.yaml

} K8s

Chart.yaml :

```
apiVersion: v1
name: helloworld
description: helloworld chart that runs on kubernetes
version: 1.0.0
keywords:
  - helloworld
  - kubernetes
home: https://github.com/stakater-charts/helloworld
maintainers:
- name: Stakater
  email: stakater@aurorasolutions.io
```

Values.yaml :

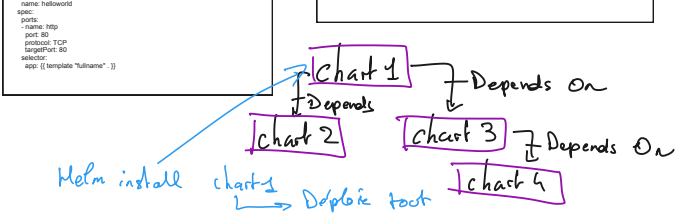
```
kubernetes:
  host: https://kubernetes.default

helloworld:
  namespace: default
  image:
    name: tutum/hello-world
    tag: latest
    pullPolicy: IfNotPresent
  service:
    ingressClass: internal-ingress
```

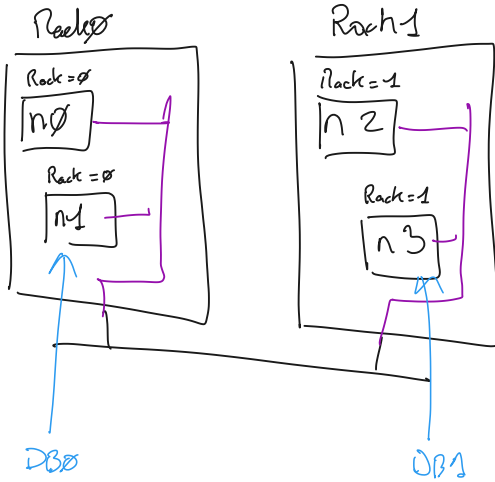
Deployment.yaml :

```
---
apiVersion: v1
kind: List
items:
- apiVersion: extensions/v1beta1
  kind: Deployment
  metadata:
    labels:
      app: {{ template "fullname" . }}
      chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
      release: {{ .Release.Name | quote }}
      heritage: {{ .Release.Service | quote }}
      version: {{ .Chart.Version }}
    name: helloworld
  spec:
    replicas: 1
    selector:
      matchLabels:
        app: {{ template "fullname" . }}
    template:
      metadata:
        labels:
          provider: fabric8
          app: {{ template "fullname" . }}
          chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
          release: {{ .Release.Name | quote }}
          heritage: {{ .Release.Service | quote }}
          version: 1.0.0
      spec:
        containers:
        - image: "{{ .Values.helloworld.image.name }}:{{ .Values.helloworld.image.tag }}"
          imagePullPolicy: {{ .Values.helloworld.image.pullPolicy }}
          livenessProbe:
            initialDelaySeconds: 60
            tcpSocket:
              port: 80
          name: helloworld
          readinessProbe:
            httpGet:
              path: /
              port: 80
            initialDelaySeconds: 5
          securityContext:
            privileged: false
```

Service.yaml :

```
---
apiVersion: v1
kind: List
items:
- apiVersion: v1
  kind: Service
  metadata:
    annotations:
      fabric8.io/ingress.annotations: |-
        ingress.kubernetes.io/force-ssl-redirect: false
        kubernetes.io/ingress.class: {{ .Values.helloworld.service.ingressClass }}
    labels:
      expose: "true"
      app: {{ template "fullname" . }}
      chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
      release: {{ .Release.Name | quote }}
      heritage: {{ .Release.Service | quote }}
      version: 1.0.0
    name: helloworld
  spec:
    ports:
    - name: http
      port: 80
      protocol: TCP
      targetPort: 80
    selector:
      app: {{ template "fullname" . }}
```





Dimensions.. Région

Zone → Data Center

Salle

Rack

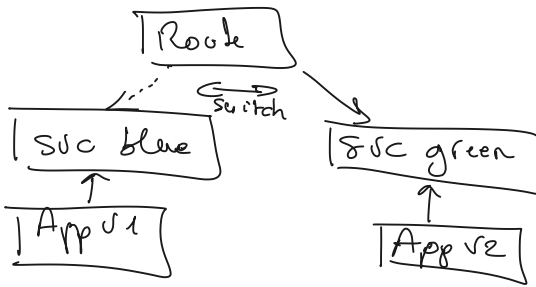
node = 1 Physique

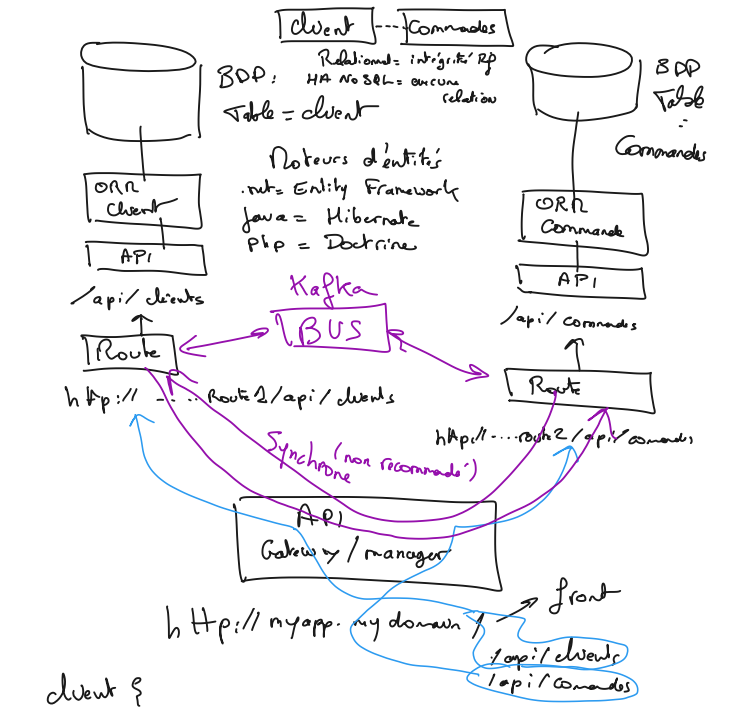
cluster = ensemble logique de machines (Virtuelle / Physique)

peut être déployé entre Régions

Types de Ressources — GPU? SSD? etc...

```
apiVersion: v1
kind: Pod
metadata:
 labels:
    test: liveness
  name: liveness-exec
spec:
  containers:
  - name: liveness
    image: k8s.gcr.io/busybox
    args:
    - /bin/sh
    - -c
    - touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600
    livenessProbe:
      exec:
        command:
        - cat
        - /tmp/healthy
      initialDelaySeconds: 5
      periodSeconds: 5
```





client {
id : 1
nom : apki'
commandes : {
{ id : 10001,
prix : 1000 },
{ id : 10009,
prix : 1500 }...

10001
10009
→ 10102

Synchrone → | - cohérent
| - Dépendant

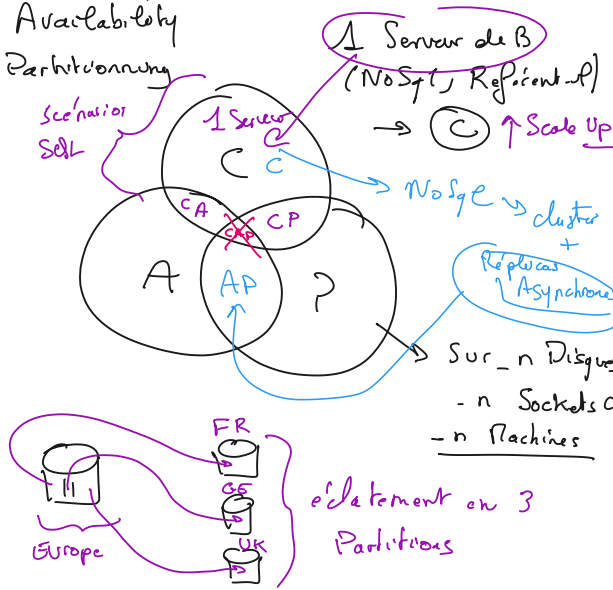
Asynchrone → Disponible
Découplé

Théorème de CAP

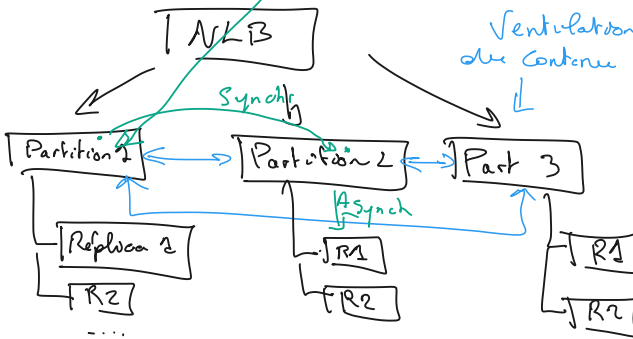
C : Consistency (cohérence)

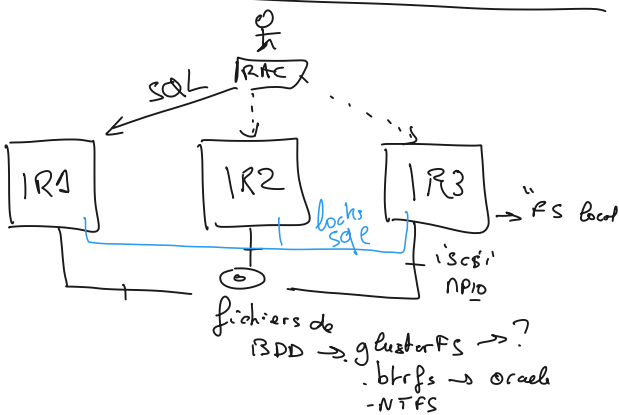
A : Availability

P : Partitionnement



Haute dispo NoSql





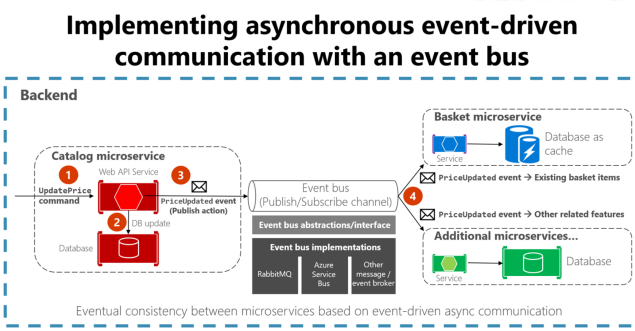
Oracle RAC = - Cohérence garantie
- HAUTE Dispo
- Partition de Serveurs, pas de stockage

Actuellement :

A Heuvidre au Nax la cohérence avec AP (HA) = utiliser le MVCC

Multi Version Concurrency Check_

L'utilisation via un Bus



CI/CD :
Continuous Integration : l'essentiel du pipeline
Continuous Delivery : - Livraison
Continous Deployment : - Déploiement

Livraison : mettre à disposition dans le repo (Image)
Déploiement : mise en prod automatique (dépendant des tests)

Un cluster OpenShift = n Projets
= 0..n déploiements de OpenShift Service Mesh (généralement un seul)
= 1..n Projets "intégrés dans le mesh"
(ces projets ne sont pas développés spécialement pour le mesh)
= 1 Mesh peut gérer n projets
1 Projet "OpenShift" eq "namespaces" K8s = 1 Micro Service
= 1 Projet de Dev (en général)

Alternative : n microservices = n Projets de Dev = 1 Projet OpenShift (les regrouper en Apps)

car 1 Projet = par ex. le front (UI), le Middle (API), le back (La BDD)

Tuto d'installation d'un ServiceMesh OpenShift v2 :
https://docs.openshift.com/container-platform/4.8/service_mesh/v2x/installing-ossm.html#installing-ossm