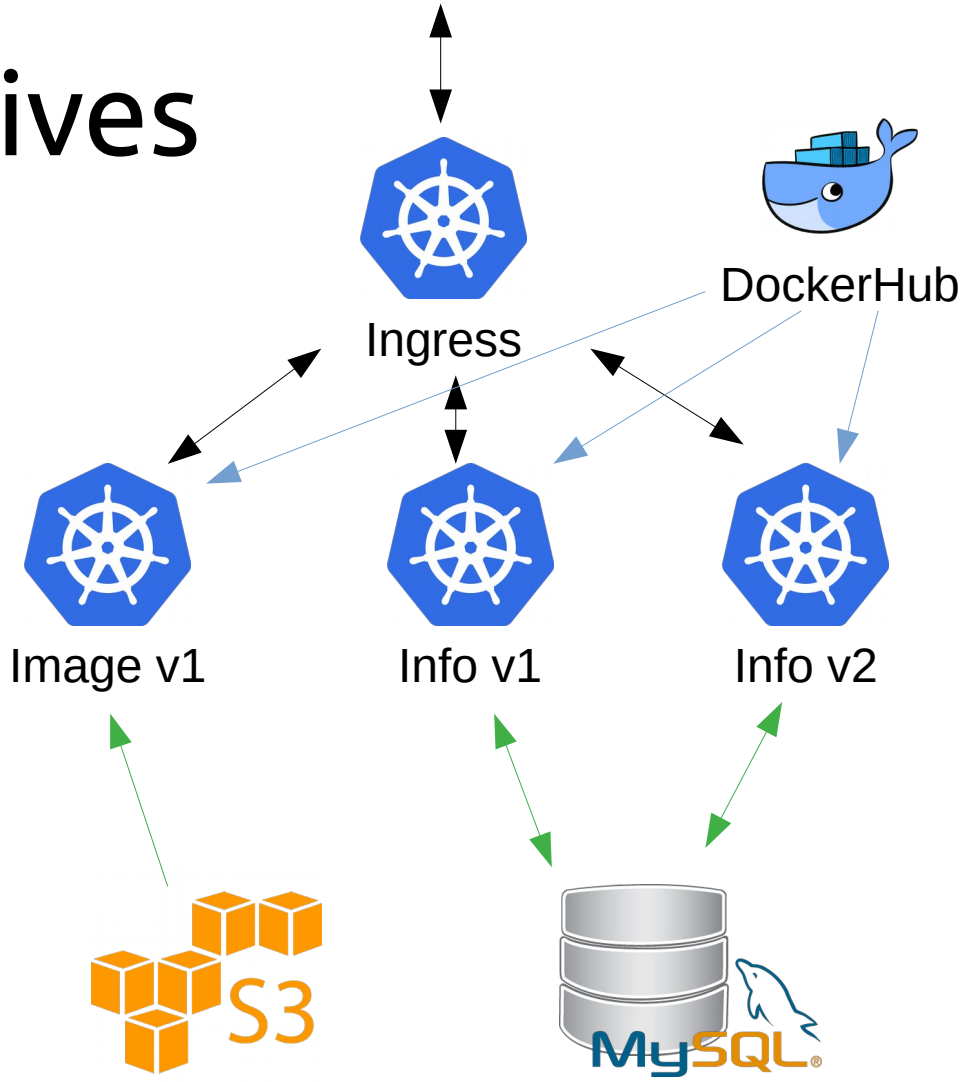
CLOUD COMPUTING

PROJECT 1 - Part II

Lorenzo Leonini
lorenzo.leonini@unine.ch

Objectives

- Deploy an API in the cloud
  - Using kubernetes !
- Leverage PaaS
  - Docker Images repository
  - SQL DB
  - NoSQL DB (S3)
  - Container orchestration
- Automate the process
  - Code => Deployment



Microservice - Watch info v2

- Compared to v1, minor changes + new search semantic
  - See README in git
  - New microservice
    - Duplicate v1 and apply the changes
    - Or decide between v1 and v2 at launch via env parameter
- Keep v1
  - Backward compatibility

Microservice - Image info v1

- New microservice
- Very simple (image proxy)
  - One endpoint /image/v1/get/{sku}
    - Will transfert an image from a NoSQL repo (AWS S3) to the user
      - Fixed mime type (image/png)
      - No image processing (ATM ;-)
    - Set proper caching headers and ETag
      - Lifetime 1h

Watches images

- The images are stored in a AWS S3 bucket (localized in Ireland)
  - Publicly accessible
  - Image format is transparent PNG (max 1024x1024)
  - https://s3-eu-west-1.amazonaws.com/cloudcomputing-2018/project1/images/<SKU>.png
    - https://s3-eu-west-1.amazonaws.com/cloudcomputing-2018/project1/images/CAC1111.BA0850.png
  - You can access them from GKE
    - Not in the same datacenter will add a slight delay
  - Not all images will be available ! (~ 50% of SKUs have images)
    - Check for 200 or 404 HTTP codes

Git repository

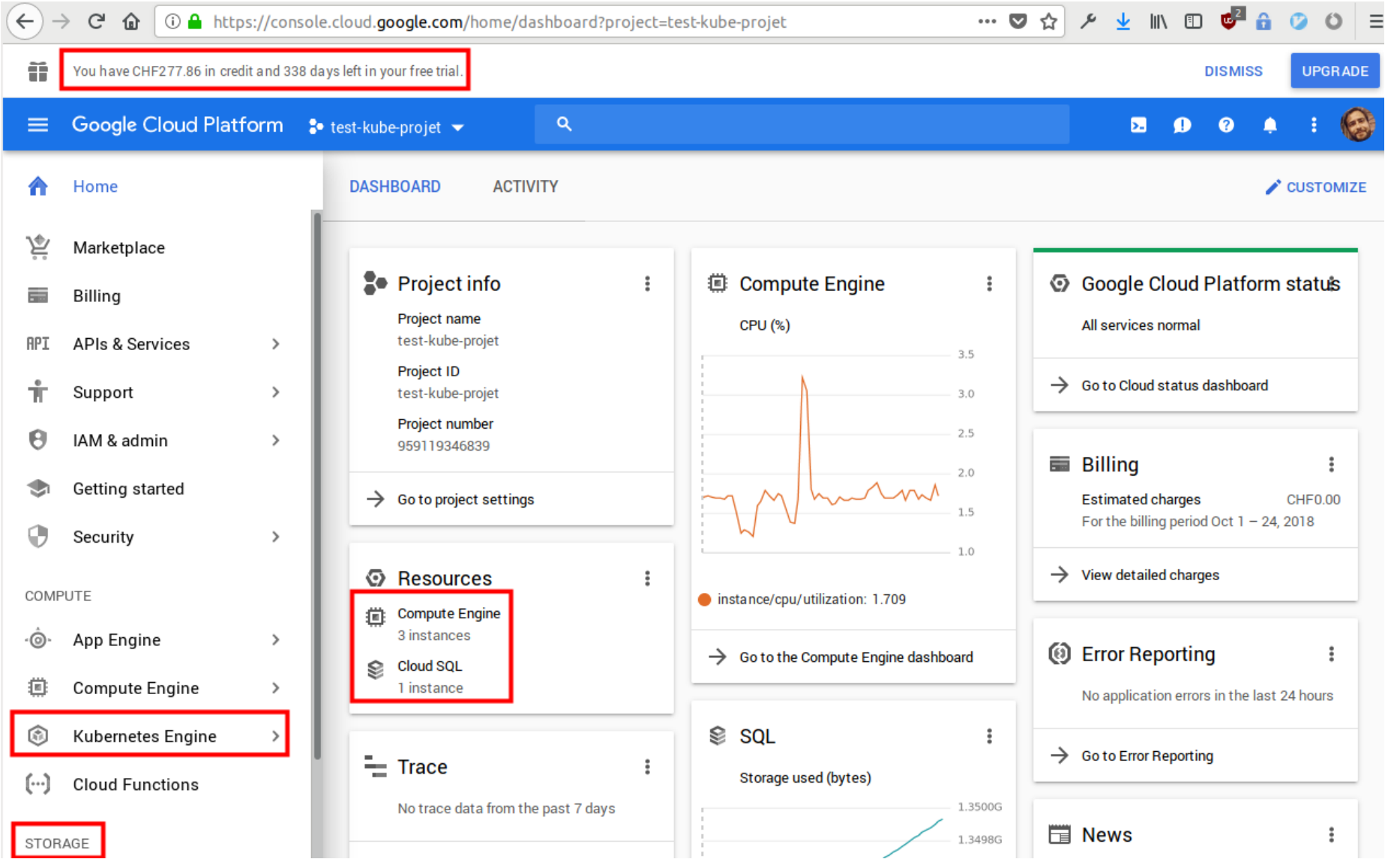
- Renaming
  - openapi.yaml to info_openapi_v1.yaml
  - swagger.yaml to info_swagger_v1.yaml
- Added
  - info_openapi_v2.yaml
  - image_openapi_v1.yaml
    - There is not anymore a swagger version
  - README to summarize changes in the repo

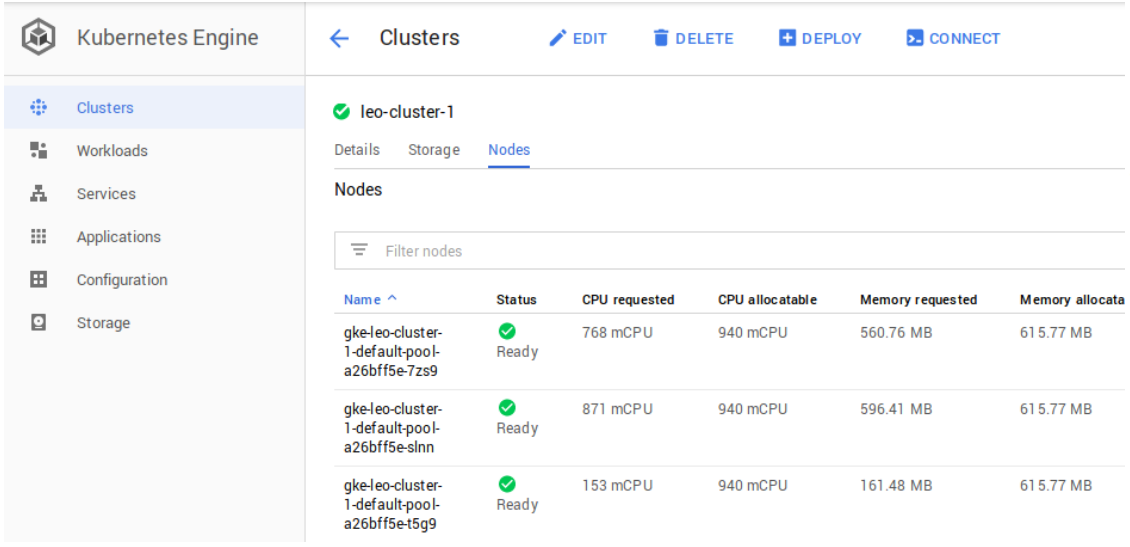
DockerHub

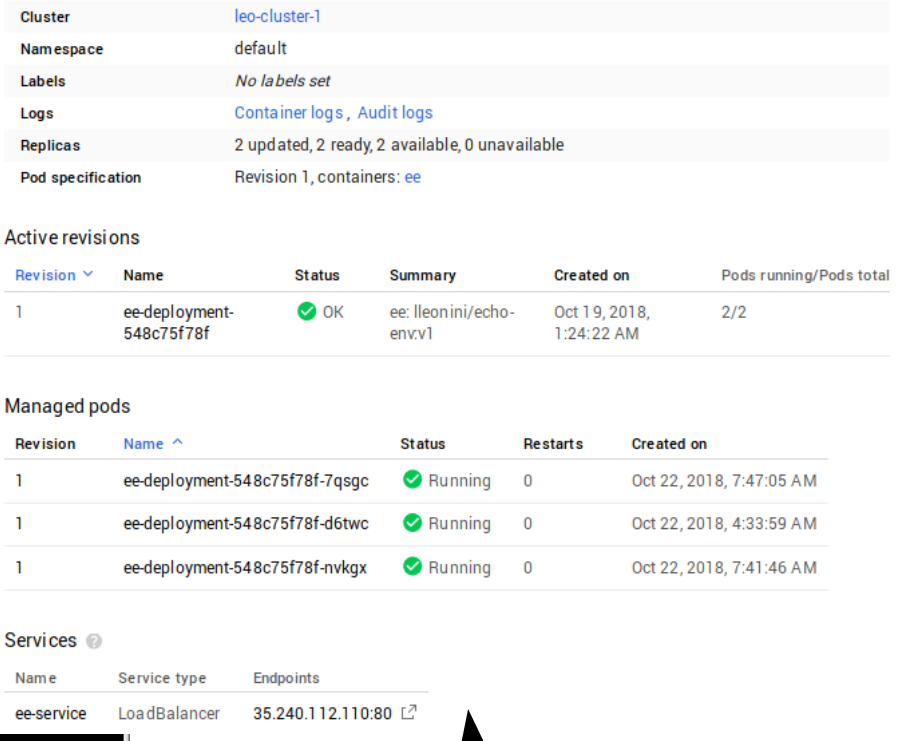
- Docker/Kubernetes/AWS can use any Docker images repositories
  - But DockerHub is the default
  - If you publish your images publicly on DockerHub, you will not have to manage complex access to a particular repo (and no authentication to manage)
  - Create a free account
- Microservices images to publish
  - Infos v1
  - Infos v2
  - Images v1
- Publishing updated images should be part of your build process

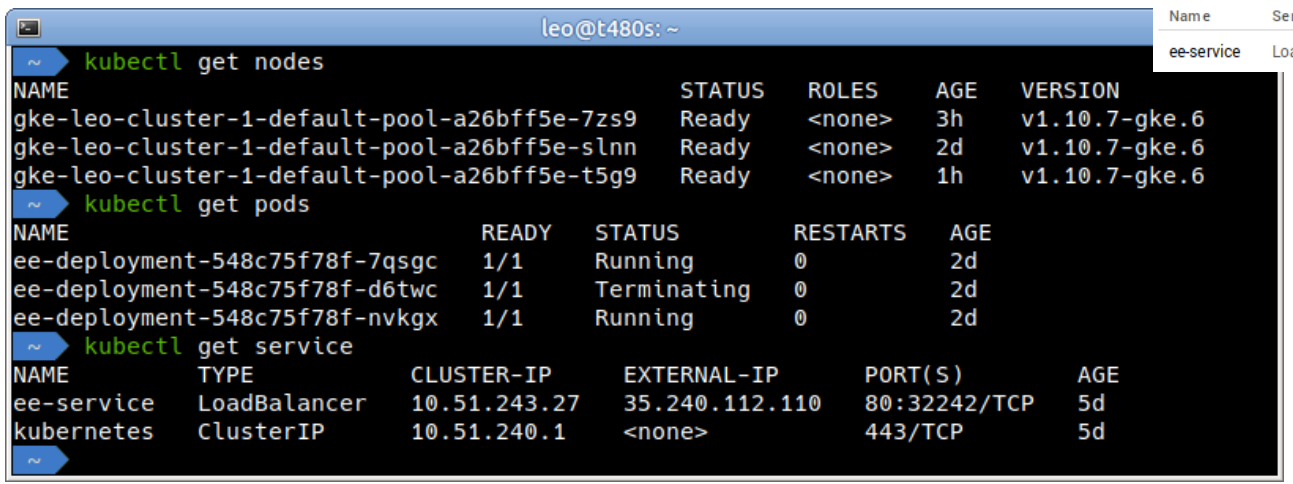
Kubernetes - GKE

- Create a Google Compute/Kubernetes Engine account
  - https://cloud.google.com/compute/
  - 300$ / 1 year free credits
  - Create a kubernetes cluster in Europe West (close to AWS S3 for images)
  - Link your account to kubectl
    - https://cloud.google.com/kubernetes-engine/docs/quickstart
    - $ gcloud container clusters get-credentials <cluster> --zone europe-west1-b --project <project>
    - Then, use kubectl as you will use it locally with minikube !
- On AWS, kubernetes also exists but is expensive (~200$ / month with mininal config) and much complicated to deploy
  - Kubernetes is a technology from Google and it is much more integrated in GCE


https://console.cloud.google.com/home/dashboard?project=test-kube-projet
You have CHF277.86 in credit and 338 days left in your free trial.
DISMISS
UPGRADE
Google Cloud Platform
test-kube-projet
Home
Marketplace
Billing
APIs & Services
Support
IAM & admin
Getting started
Security
COMPUTE
App Engine
Compute Engine
Kubernetes Engine
Cloud Functions
STORAGE
DASHBOARD
ACTIVITY
CUSTOMIZE
Project info
Project name
test-kube-projet
Project ID
test-kube-projet
Project number
959119346839
Go to project settings
Resources
Compute Engine
3 instances
Cloud SQL
1 instance
Trace
No trace data from the past 7 days
Compute Engine
CPU (%)
3.5
3.0
2.5
2.0
1.5
1.0
instance/cpu/utilization: 1.709
Go to the Compute Engine dashboard
SQL
Storage used (bytes)
1.3500G
1.3498G
Google Cloud Platform status
All services normal
Go to Cloud status dashboard
Billing
Estimated charges
CHF0.00
For the billing period Oct 1 – 24, 2018
View detailed charges
Error Reporting
No application errors in the last 24 hours
Go to Error Reporting
News







Connect GCE frontend with command line kubectl

SQL Database

- The SQL database must be **shared** between info services v1 and v2
- 2 possibilities (you can decide)
  - 1. Use a cloud managed MySQL (in GCE or in AWS)
    - Open firewall to access it from kubernetes pods
  - 2. Use kubernetes with MySQL image and a persistent volume
    - https://kubernetes.io/docs/tasks/run-application/run-single-instance-stateful-application/
      - Can be more difficult to initialize the DB
- Bonus
  - (re-) Initialize the DB automatically with deployments

Kubernetes - Ingress controller

- Use ingress controller to route to your microservices
  - /watch/v1/*
  - /watch/v2/*
  - /image/v1/*
- Ask for a static IP address
  - https://cloud.google.com/kubernetes-engine/docs/tutorials/http-balancer

Kubernetes - Deployment

- Create 3 services (info v1, v2 and image)
  - => 3 deployment + replicaSets
    - Permits scalling instances
    - Permits rolling upgrade
- Create Ingress controller to route to these 3 services
- Bonus
  - Container probes
    - Add a liveness check (HTTP query returning 200)
      - https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/

Deliverables - Delay - Grading

- Deliverables
  - **IP/DNS to test your API !**
  - Short documentation (README)
    - How to (re-)deploy
  - Script(s) to build & deploy
    - => rebuild 3 service images
    - => update 3 images to DockerHub
    - => kubernetes rolling upgrade
  - Kubernetes deployment file(s) yaml
    - Deployment instructions
    - Multiple object descriptions can be grouped in one file, use '\n---\n' as separator
- Delay: 3 weeks
- Grading
  - Try to **automate** the process as much as possible
    - Code changes to "in production" must be as simple as possible
  - Bonus: each + 0.5 points