

CLOUD COMPUTING PROJECT

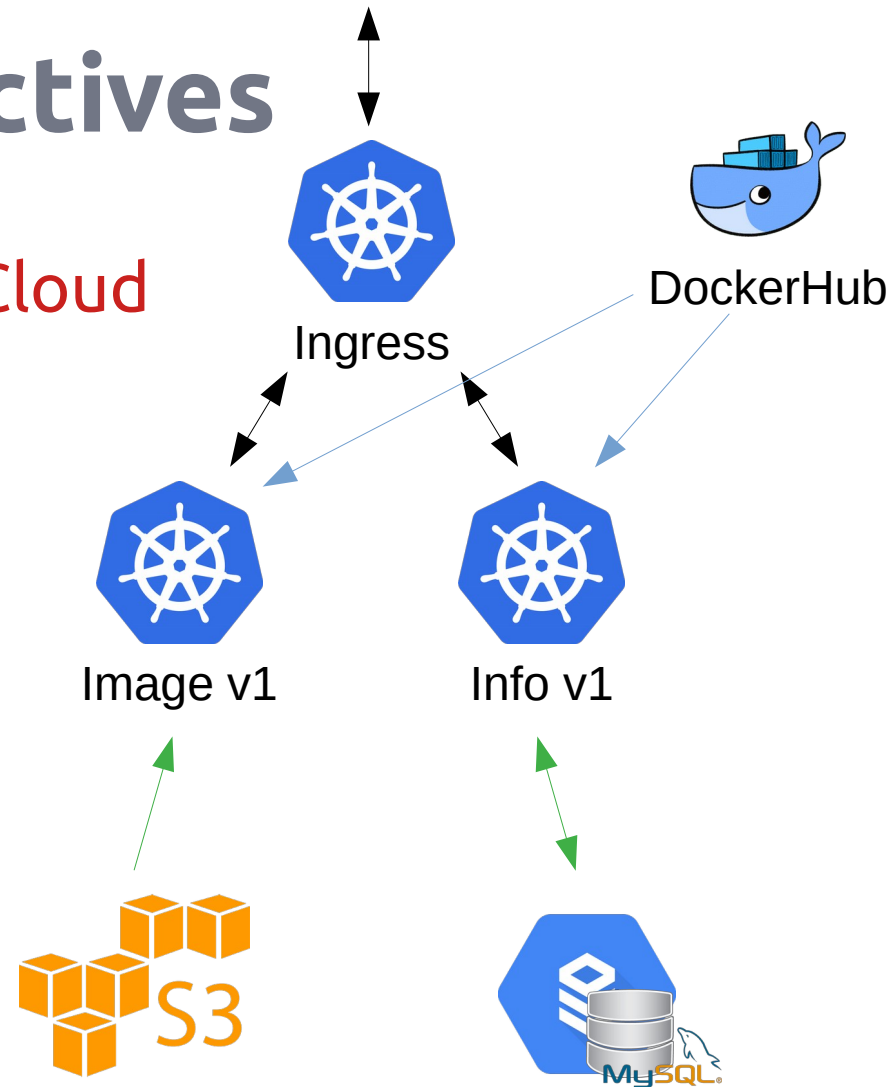
**WATCHES**

**WEBSERVICES**

**PART II**

# Part II - Objectives

- Deploy Watch Service in **Google Cloud**
- Leverage PaaS
  - Google GKE
  - Google CloudSQL
  - Docker Images repository
  - NoSQL DB (AWS S3)
- Automate the process
  - Code → Deployment



# Microservice - Image info v1

- New Image (proxy) microservice
  - One endpoint `/image/v1/get/{sku}`
    - Will transfer 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
  - OpenAPI description in git: `image_openapi_v1.yaml`

# Watches Images

- The images are stored in a AWS S3 bucket (Ireland)
  - Publicly accessible
    - You can access them from GKE
  - 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>
      - <https://s3-eu-west-1.amazonaws.com/cloudcomputing-2018/project1/images/CV201AP.FC6429.png>
  - Not all images will be available ! (~ 50% of SKUs have images)
    - Check for 200 or 404 HTTP codes

# DockerHub

- Docker/Kubernetes/AWS can use any Docker images repositories
  - 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
  - Images v1
- Publishing updated images should be part of your build process

# Google Cloud

- Create a Google Cloud Platform account
  - <https://cloud.google.com/compute/>
  - 300\$ / 1 year free credits + 50\$ educational credits
    - However these 2 billing accounts will not be combined
  - Create a new project
    - Associate with the right billing account
  - Create a budget
    - Can be global for all projects

## Billing

Billing account  
Billing Account for Education

- Overview
- Reports
- Cost table
- Cost breakdown
- Commitments
- Commitment analysis
- Budgets & alerts**
- Billing export
- Pricing
- Account management

## Create Budget

## 1 Scope

A budget enables you to track your actual spend against your planned spend.

Name \*

my budget

A budget can be scoped to focus on a specific set of resources.

Projects

All projects (2)

Services

All services (1684)

Labels ?

Select the key and value of the label you want to filter.

Credits

Selected credits are applied to the total cost. Budget tracks the total cost minus any applicable selected credits

☐ Discounts ?☐ Promotions and others ?

unckeck

NEXT

## 2 Amount

## 3 Actions

FINISH

CANCEL

## Cost trend

April 1, 2020 – April 30, 2021



Actual cost

→ View report

## Billing

Billing account  
Billing Account for Education ▼

- Overview
- Reports
- Cost table
- Cost breakdown
- Commitments
- Commitment analysis
- Budgets & alerts**
- Billing export
- Pricing
- Account management

## ← Create Budget

## ✓ Scope

## 2 Amount

Set a **monthly budget amount**. Budgets begin on the first of the month, and reset at the beginning of each month.

Budget type

Specified amount ▼

A fixed amount that your spend will be compared against.

Target amount

50

\$

NEXT

## 3 Actions

FINISH

CANCEL

## Cost trend

April 1, 2020 – April 30, 2021



● Actual cost

→ View report



# Google Cloud SDK

- Install the SDK
  - <https://cloud.google.com/sdk/docs/install>

\$ gcloud auth login

\$ gcloud components list

\$ gcloud components install minikube

\$ gcloud components install kubectl

...

# Kubernetes - GKE

- Create a minimal Kubernetes cluster
  - Autopilot or manually selecting computing resources
- Link your account to kubectl
  - <https://cloud.google.com/kubernetes-engine/docs/quickstart>  
\$ `gcloud container clusters get-credentials ...`
  - Use kubectl as you will use it locally with minikube !



Home



Billing



Kubernetes Engine



SQL



Storage



PRODUCTS



Marketplace



Billing



APIs &amp; Services



Support



IAM &amp; admin



Getting started



Security



COMPUTE

DASHBOARD

ACTIVITY

CUSTOMIZE



## Project info



Project name

kube-2019

Project ID

kube-2019-256711

Project number

938662733845



Go to project settings



## Resources



Compute Engine

1 instance



Storage

1 bucket



SQL

1 instance



## Trace



No trace data from the past 7 days



## Compute Engine



CPU (%)



instance/cpu/utilization: 0.204



Go to Compute Engine



## SQL



Storage used (bytes)



## Google Cloud Platform status



All services normal



Go to Cloud status dashboard



## Billing



Estimated charges

USD \$0.00

For the billing period Oct 1 – 23, 2019



View detailed charges



## Error Reporting



No sign of any errors. Have you set up Error Reporting?



Learn how to set up Error Reporting



## News



Use GKE usage metering to combat over-provisioning

[←](#) Create a Kubernetes cluster

## Clone an existing cluster

Select one of your existing clusters to populate fields



## Standard cluster

Continuous integration, web serving, backends. Best choice for further customization or if you are not sure what to choose.



## Your first cluster

Experimenting with Kubernetes Engine, deploying your first application. Affordable choice to get started. [→](#)



## CPU intensive applications

Web crawling or anything else that requires more CPU.



## Memory intensive applications

Databases, analytics, things like Hadoop, Spark, ETL or anything else that requires more memory.



## GPU Accelerated Computing

Machine learning, video transcoding, scientific computations or anything else that is compute-intensive and can utilize GPUs.



## Highly available

Most demanding availability requirements. Both the master and the nodes are replicated across multiple zones.

Name <sup>?</sup>Location type <sup>?</sup>☒ Zonal☐ RegionalZone <sup>?</sup>

## Master version

<sup>i</sup> Try the new Release Channels feature instead of managing the master version directly.

[Use Release Channels](#)

## Node pools

Node pools are separate instance groups running Kubernetes in a cluster. You may add node pools in different zones for higher availability, or add node pools of different type machines. To add a node pool, click Edit. [Learn more](#)

## pool-1

## Number of nodes

Machine configuration <sup>?</sup>

## Machine family

☒ General-purpose☐ Memory-optimized

Machine types for common workloads, optimized for cost and flexibility

## Generation

First

Powered by Skylake CPU platform or one of its predecessors

## Machine type



vCPU

1 shared core

Memory

1.7 GB

|                                  |                        |
|----------------------------------|------------------------|
| Cluster version                  | 1.14.6-gke.13 (latest) |
| Machine type                     | g1-small               |
| Autoscaling                      | Disabled               |
| Stackdriver Logging & Monitoring | Disabled               |
| Boot disk size                   | 30GB                   |

You will be billed for the 1 node (VM instance) in your cluster. [Compute Engine pricing](#)

- Clusters
- Workloads
- Services & Ingress
- Applications
- Configuration
- Storage

A Kubernetes cluster is a managed group of VM instances for running containerized applications. [Learn more](#)

Filter by label or name

| <input type="checkbox"/> | Name ^      | Location       | Cluster size | Total cores | Total memory | Notifications | Labels |   |
|--------------------------|-------------|----------------|--------------|-------------|--------------|---------------|--------|---|
| <input type="checkbox"/> | ✔ test-2019 | europe-west1-b | 1            | 1 vCPU      | 1.70 GB      |               |        | Connect   |

```
leo@t480s: ~  
gcloud container clusters get-credentials test-2019 --zone europe-west1-b --project kube-2019-256711  
  
Fetching cluster endpoint and auth data.  
kubeconfig entry generated for test-2019.  
→ kubectl cluster-info  
Kubernetes master is running at https://35.195.108.21  
GLBCDefaultBackend is running at https://35.195.108.21/api/v1/namespaces/kube-system/services/default-http-backend:http/proxy  
Heapster is running at https://35.195.108.21/api/v1/namespaces/kube-system/services/heapster/proxy  
KubeDNS is running at https://35.195.108.21/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy  
Metrics-server is running at https://35.195.108.21/api/v1/namespaces/kube-system/services/https:metrics-server:/proxy  
  
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.  
→
```

- Kubernetes Engine
- Clusters
- Workloads
- Services & Ingress
- Applications
- Configuration
- Storage

### ✔ test-2019

Details Storage Nodes

#### Nodes

Filter nodes

| Name ^                             | Status  | CPU requested | CPU allocatable | Memory requested | Memory allocatable |
|------------------------------------|---------|---------------|-----------------|------------------|--------------------|
| gke-test-2019-pool-1-bb16ecb7-zf6b | ✔ Ready | 629 mCPU      | 940 mCPU        | 634.8 MB         | 1.22 GB            |

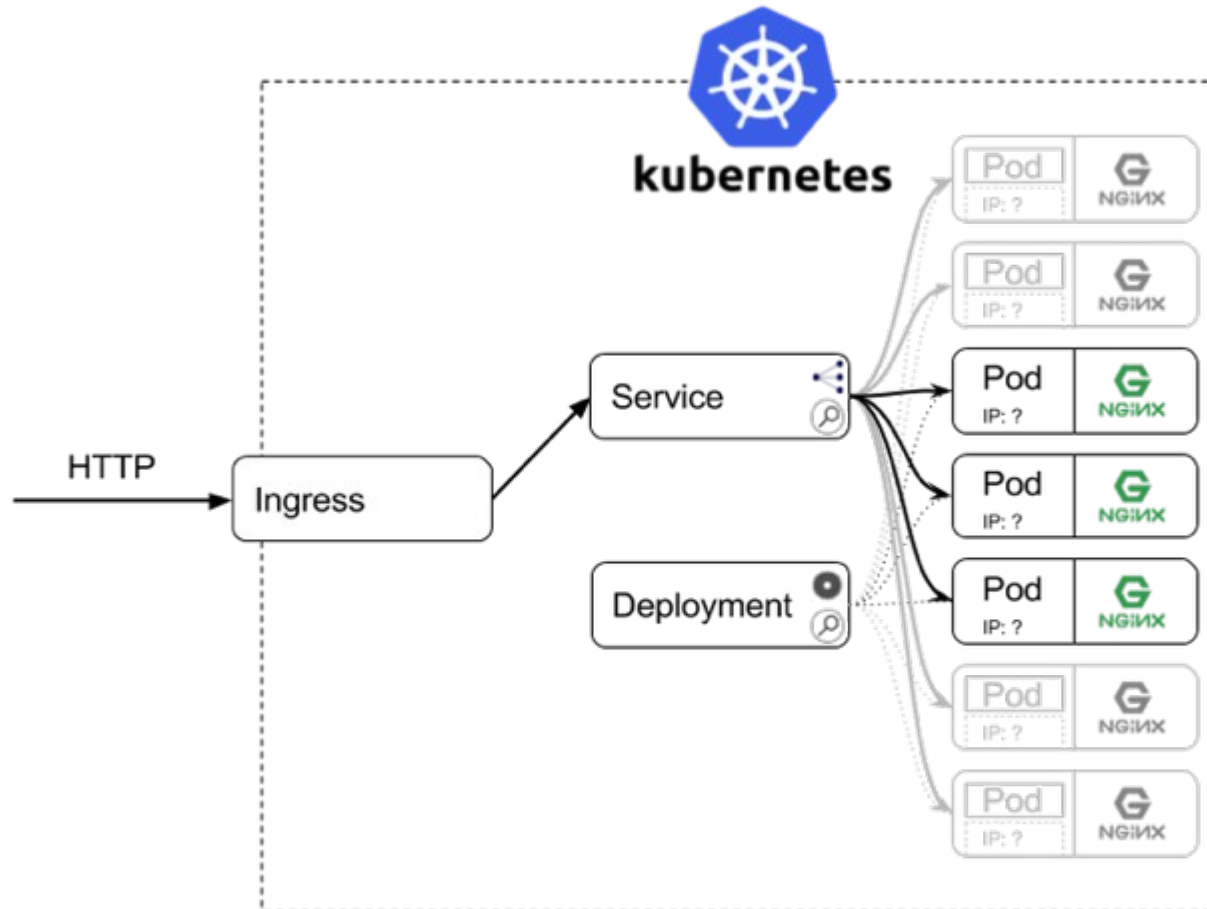
# CloudSQL

- CloudSQL
  - Managed MySQL by Google
- The SQL database will be used by info services v1
  - Deploy (manually) the same database as in part I
  - Or using command line:
    - <https://cloud.google.com/sql/docs/mysql/connect-admin-ip>
- Access from Kubernetes
  - Public access from everywhere (not recommended in production)
    - <https://cloud.google.com/sql/docs/mysql/configure-ip>
  - Using Cloud SQL Auth proxy and Secrets
    - <https://cloud.google.com/sql/docs/mysql/connect-kubernetes-engine>

# Kubernetes Architecture

- Objects to create
    - **Deployments**
      - One deployment for each docker image (→ 2)
      - 2 replicas (pods)
    - **Services**
      - One service for each image deployment (→ 2)
      - Service will load balance between the pods of the corresponding deployment
    - **Ingress**
      - Will route to the right service depending on the path
- ⇒ **all.yaml**
- Multiple object descriptions can be grouped in one file, use '\n---\n' as separator

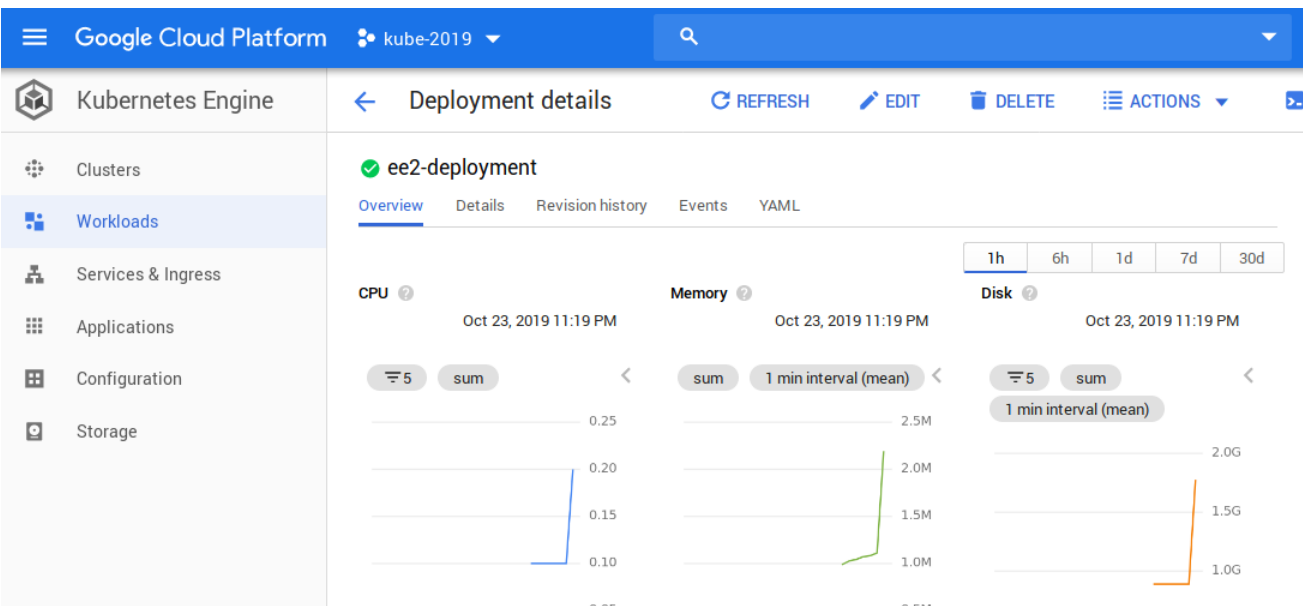
# Kubernetes Architecture #2





# Kubernetes - Ingress controller

- Use ingress controller to route to your microservices
  - `/info/v1/*` → Service info
  - `/image/v1/*` → Service image
- Use **HTTP** (HTTPS not mandatory)
- Documentation
  - <https://cloud.google.com/kubernetes-engine/docs/concepts/ingress>
  - <https://cloud.google.com/kubernetes-engine/docs/tutorials/http-balancer>
- Note: you may have to wait 10-15 minutes before the ingress controller is completely deployed



Google Cloud Platform kube-2019

### Kubernetes Engine

#### Services & Ingress

REFRESH CREATE INGRESS DELETE

Kubernetes services Brokered services BETA Ingresses

Services are sets of Pods with a network endpoint that can be used for discovery and load balancing. Ingresses are collections of rules for routing external HTTP(S) traffic to Services.

Is system object : False Filter resources

| Name        | Status | Type          | Endpoints         | Pods  | Namespace | Cluster   |
|-------------|--------|---------------|-------------------|-------|-----------|-----------|
| ee2-ingress | Ok     | Ingress       | 35.201.69.5/      | 0 / 0 | default   | test-2019 |
| ee2-service | Ok     | Load balancer | 35.195.207.231:80 | 2 / 2 | default   | test-2019 |

# Deliverables

- `/info/`
  - Put inside that directory the service developed in part I ( and all related files)
- `/image/`
  - Create the new image service, with the same files/structure than in part I
    - Dockerfile, server.py, requirements.txt, run.sh, ...
- `/all.yaml`
  - Kubernetes: Deployments + Services + Ingress
- `/build.sh`
  - (Re)build the 2 docker service images
- `/deploy.sh`
  - Publish/update the 2 docker images to DockerHub (built by `build.sh`)
  - Kubernetes (initial) deployment and rolling upgrade
- `/README`
  - Instructions to launch the service with an existing CloudSQL instance (where and how to set ENV vars or secrets to run the service)

Once this part is finished, tag your commit with '`part2`'

# Notes

- Start creating/testing the new image service as described in project Part I, test, build and publish as Docker image
  - Try to expose (only) that service locally with Minikube
- When possible use Minikube for testing
  - When you are not using GKE (for a few days or weeks), it could be better to destroy it and create a new one only when you need it
    - Minimal cluster  $\Rightarrow$  ~ 25\$/week
- CloudSQL deployment has not to be automated
  - But the how to access it has to be clearly explained in README (in a way that assistants will be able to deploy your services with their own instance of the DB)

# Documentation

- Minikube
  - <https://github.com/kubernetes/minikube>
  - <https://kubernetes.io/docs/tasks/tools/install-minikube/>
  - <https://kubernetes.io/docs/setup/learning-environment/minikube/>
- Kubernetes Cheat Sheet
  - <https://linuxacademy.com/blog/containers/kubernetes-cheat-sheet/>