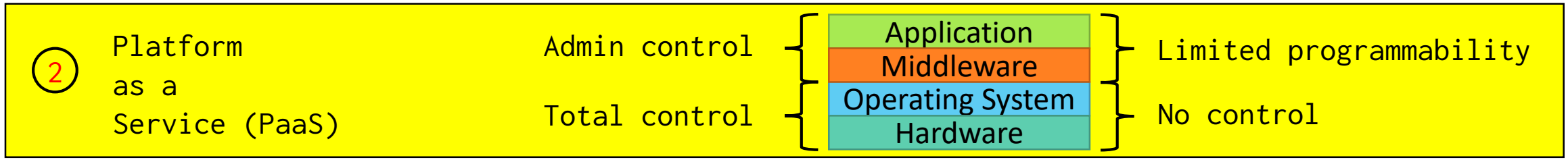
Google App Engine

Ing. Stabili Dario
dario.stabili@unimore.it
http://weblab.ing.unimo.it/people/stabili

Overview

Google App Engine

Platform as a Service (PaaS)



- The user (developer) does not see **hardware** nor **OS**
- He can user **services** through (usually proprietary) **APIs**

Focus on coding! (but **beware of lock-in!**)

- <u>Examples:</u>
  - Google App Engine (GAE)
  - Amazon Elastic Beanstalk
  - Heroku

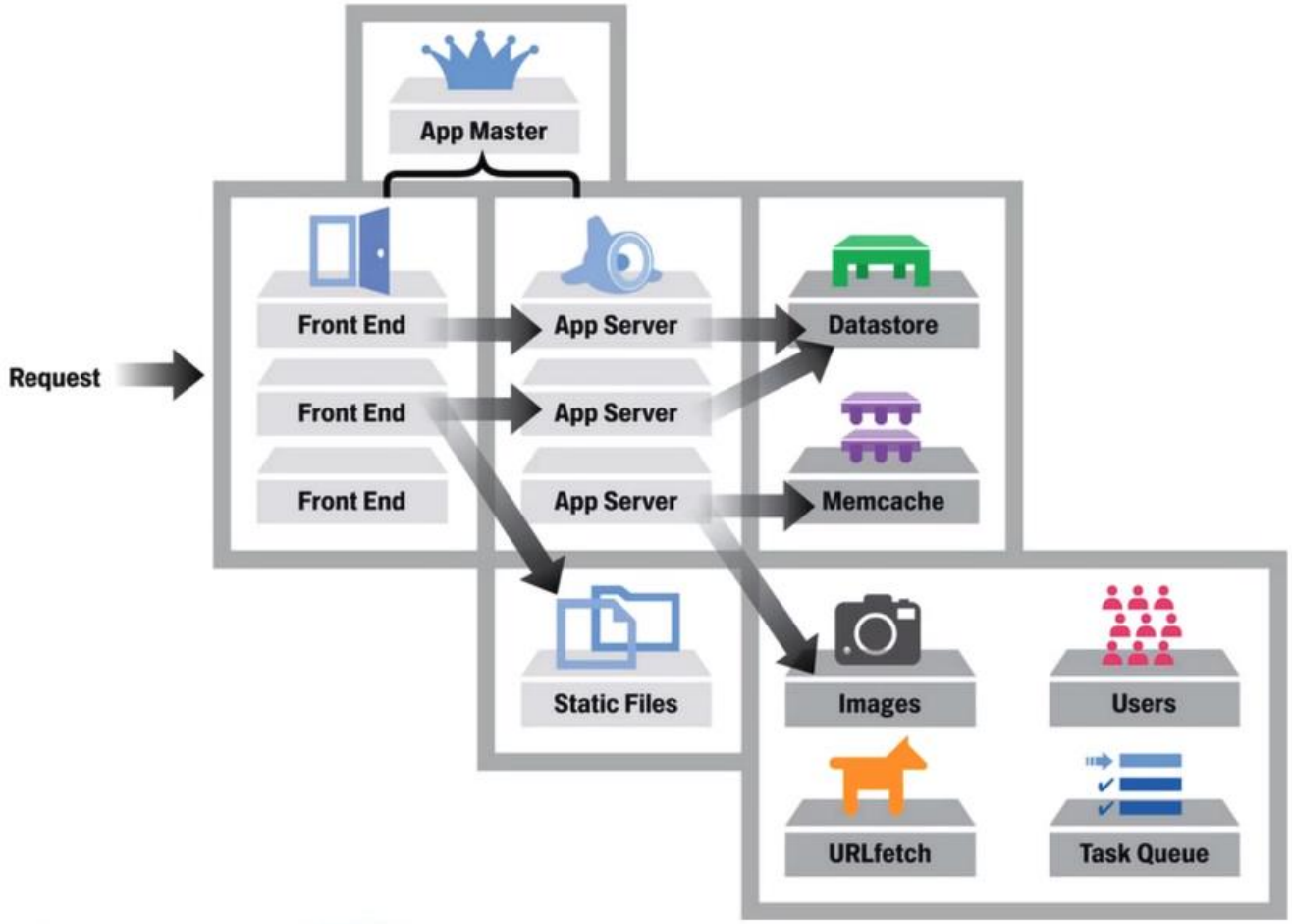
Google App Engine

"A Platform as a Service (**PaaS**) cloud computing platform for developing and hosting web applications in Google-managed data centers"

https://developers.google.com/appengine

- Supported languages:
  - **Python**
  - Java
  - PHP
  - Go

GAE Architecture



Source: https://www.youtube.com/watch?v=QJp6hmASstQ

The point of view of the developer

- Hardware and network characteristics are hidden
  - No networking facilities -> Network APIs (e.g. URL fetcher)
  - No storage of files -> Storage APIs (e.g. Cloud Storage, Memcache, Blobstore)
  - High level service APIs (e.g. MapReduce, Mail)
- Aware of the resource management strategies of the provider (e.g. fast on-line responses)
- Aware of the performance service parameters (e.g. Pending latencies)

Free Tier

Google App Engine

GAE costs: Free tier

- GAE provides a "free tier" of resources
- When resource usage exceeds, the user starts paying resources.
- Pricing changes according to the deployment zone of the application

  https://cloud.google.com/appengine/pricing

GAE costs: Free tier

- GAE provides a "free tier" of resources
- When reso esources.
- Pricing cha e application
  https://clo

Attenzione: Si consiglia di utilizzare un account Google non collegato a carte di credito per utilizzare la versione gratuita di Google App Engine senza rischio di spiacevoli addebiti

Deploy a Flask application on Google App Engine

Exercise

Outline

- Test a Flask application locally
- Create Google App Engine (GAE) project
- Migrate the Flask application to GAE
- Test the GAE application locally
- Deploy the GAE application on Google services

- Additional functionalities on GAE
  - Serve static files
  - Serve multiple applications

Requirements

- **Having a Google account**
  - Register at http://appengine.google.com
  - **Never** insert credit cards for this class
    - Do **not** activate trial account for google cloud platform
- **Python 2.7**
- **GAE Python SDK**

GAE Python SDK

- **Download Google Python SDK**
  - https://cloud.google.com/appengine/docs/standard/python/download
  - Follow instructions on site

GAE Python SDK - Commands

- We'll use two commands from the SDK:
  - **dev_appserver.py**
    - Starts a development app server
    - Test application locally
  - **gcloud app deploy**
    - Deploy applications **online**

Start - Hello World Flask application

- Directory structure:

```
<project-directory>
    - <package-directory>
    - requirements.txt
    - [venv]
    - [setup.py]
    - [MANIFEST.in, README.txt, …]
```

Migrating from Flask to GAE

- Directory structure:

```
<project-directory>
    - <package-directory>
    - requirements.txt
    - app.yaml
    - appengine_config.py
    - lib/
    - [venv]
    - [setup.py]
    - [MANIFEST.in, README.txt, …]
```

app.yaml

- Main configuration file for the GAE application
- YAML = Yet Another Markup Language

Environment configuration

Application configuration

```
runtime: python27
api_version: 1
threadsafe: true

handlers:
- url: /.*
  script: app.flask_app.app
```

app.yaml

```
runtime: python27
api_version: 1
threadsafe: true

handlers:
- url: /.*
  script: app.flask_app.app
```

- This code runs in the **`python27`** runtime env, API version **`1`**
- The application is **`threadsafe`** so the same instance can handle several simultaneous requests.
- Every request to a URL whose pattern matches the regular expression **`/.*`** should be handled by the **`app`** object in the **`app.flask_app`** python module

appengine_config.py

To deploy your application to GAE using third-party libraries you need to install them to your application directory.

Create a directory to store the third-party libraries:

```
$ mkdir lib
```

Install the third-party libraries in the folder

```
$ pip install -t lib -r requirements.txt
```

Create the `appengine_config.py` file in the application directory

```
# -*- coding: utf-8 -*-
from google.appengine.ext import vendor

vendor.add('lib')
```

Run your application locally

```
$ <gcloud-sdk>/dev_appserver.py app.yaml
```

- Where <gcloud-sdk> is the PATH of the gcloud SDK (add it to your PATH environment)

Open a browser at `http://localhost:8080`

Deploy the application on the cloud

Initialize the gcloud tool first

```
$ gcloud init
```

Deploy the application on the GAE service

```
$ gcloud app deploy
```

Test your application on the cloud using the URL provided by GAE

Extra: Serve static files

- Configure your application to serve static files such as css, images, resources, …
- Create a folder called `static` in the project directory and add it to the `app.yaml`

```
runtime: python27
api_version: 1
threadsafe: true

handlers:
- url: /static
  static_dir: static
- url: /.*
  scrpt: app.flask_app.app
```

Extra: Specify different handlers

Change your `app.yaml`

```
runtime: python27
api_version: 1
threadsafe: true

handlers:
- url: /static
  static_dir: static
- url: /greeting/.*
  script: app.flask_template_file.app
- url: /.*
  scrpt: app.flask_app.app
```

Test the handlers with the browser

Final directory structure

```
<project-directory>
        - app/
                - flask_app.py
                - flask_template_file.py
                - template/
                        - greeting.html
        - static/
                - file.txt
        - lib/
                - …
        - app.yaml
        - appengine_config.py
        - requirements.txt
        - [venv]
```

TODO

Sa fom?