Workshop

Introduction to Google App Engine

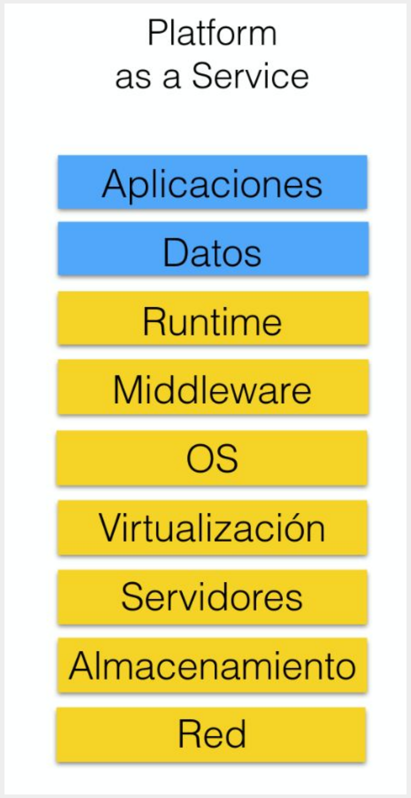
Alejandro Tatay
CTO Blinkfire Analytics
@alecdotico

Introduction to Google App Engine (GAE)

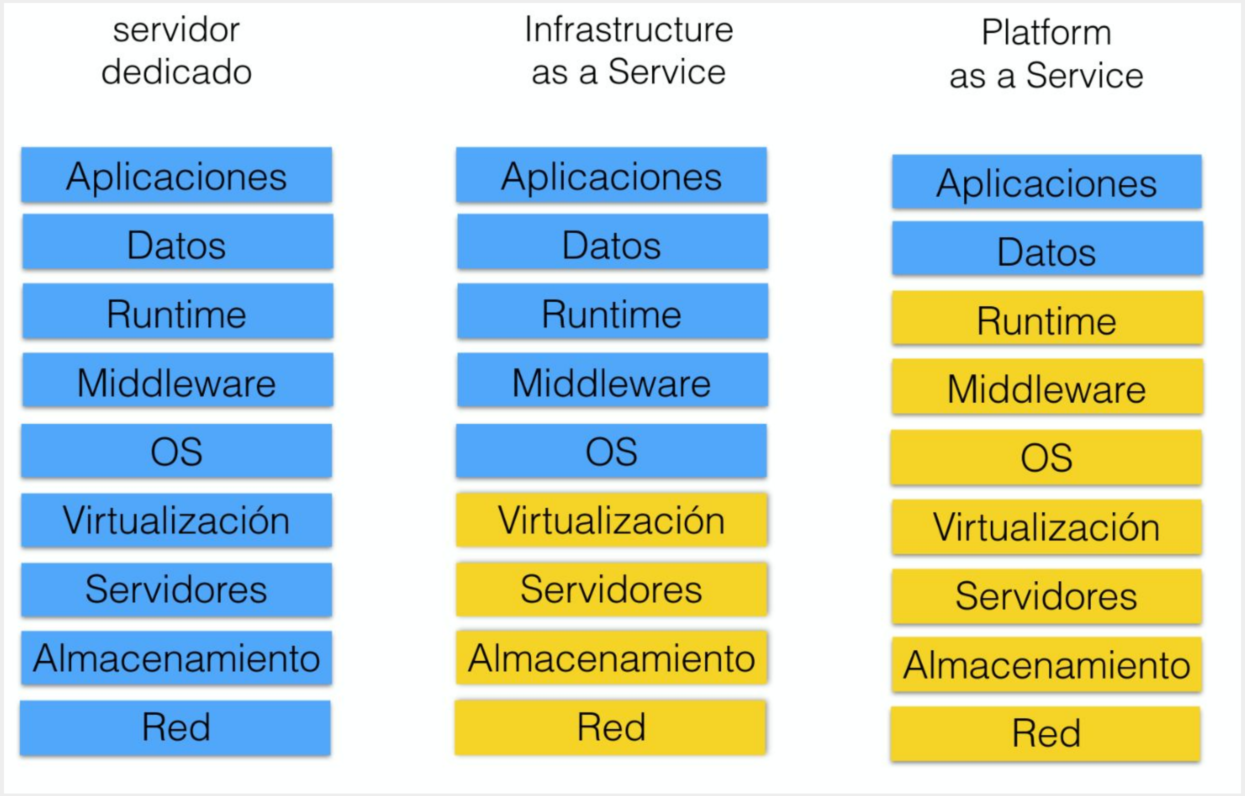
- Understanding Cloud Services
- Introduction to Google App Engine
- Workshop: Create and deploy your own GAE project

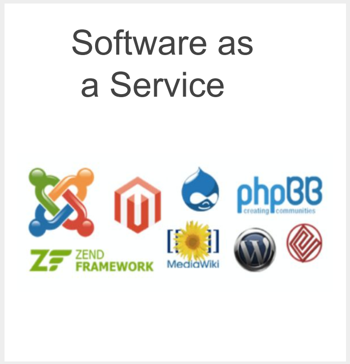
Platform as a Service (PaaS)

- Cloud Computing solution stack
- Consumer: software + configuration settings
- provider: automatized and transparent infrastructure



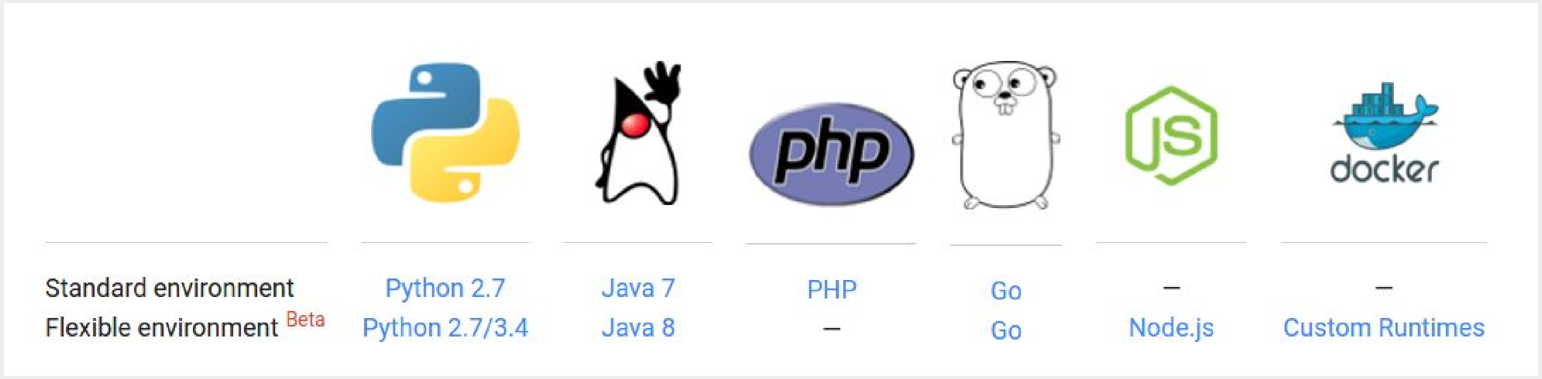
Cloud services: dedicated server, IaaS, PaaS, SaaS





PaaS: Google App Engine

- Web applications on Google's infrastructure
- Focus on your code: Easy to create, maintain and update
- Multiple storage options
- Powerful and familiar development tools
- Develop with popular frameworks and languages



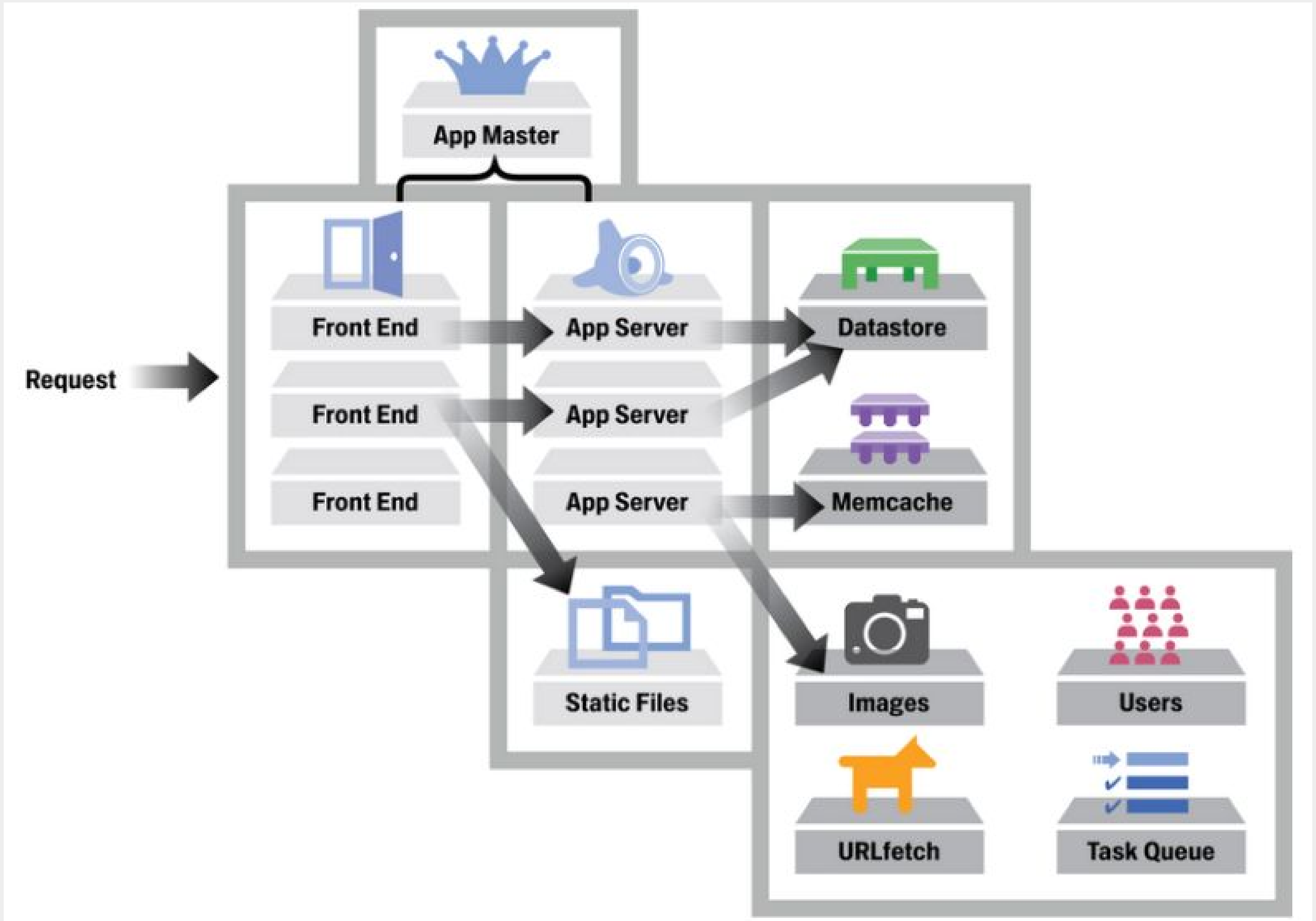
Standard environment	Python 2.7	Java 7	PHP	Go	–	–
Flexible environment Beta	Python 2.7/3.4	Java 8	–	Go	Node.js	Custom Runtimes

Google App Engine Services

App Engine applications scale automatically according to inbound traffic and workload. It natively supports:

- Load balancing
- Tasks, queues and cron jobs
- Authentication
- SQL and noSQL databases
- Distributed memcache
- Monitoring and logging
- Versioning
- Traffic splitting
- Security Scanner
- And much more...

Google App Engine Stack



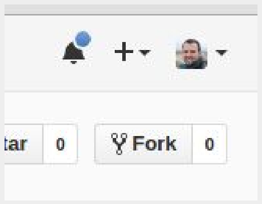
Pros & Cons

- Automatically and limitless scaling
- Easy and cheap (short term), free quotas
- agile, great fit for small projects
- Minimal management
- Google infrastructure and services
- Content Delivery Network (CDN)

- Limited environment (third-party libraries)
- Dependency on technology
- Lack of portability
- PaaS - lack of control: VMs, OS, ssh, file system, DevOps
- Pay per use of each service
- Expensive* (long term)

Let’s get to work!

- Download PyCharm Professional jetbrains.com/pycharm
- Github Log in/sign in github.com
- Fork repository github.com/alecdotico/blablastar



- Clone fork in local directory

```
git clone git@github.com:usuario/blablastar.git
```

- Google account accounts.google.com

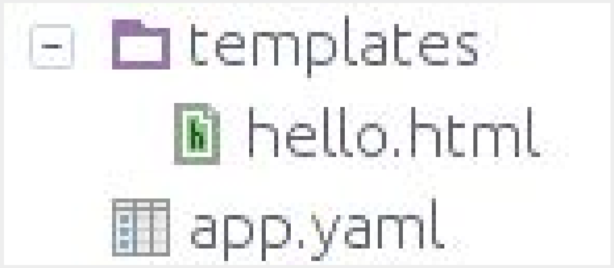
Hello World *in 10 lines only*

app.yaml

```
application: blablastar-alecdotico
version: '1'
runtime: python27
api_version: 1
threadsafe: true

handlers:
- url: /.*
  static_files: templates/hello.html
  upload: templates/hello.html
```

<html>Hello world</html>



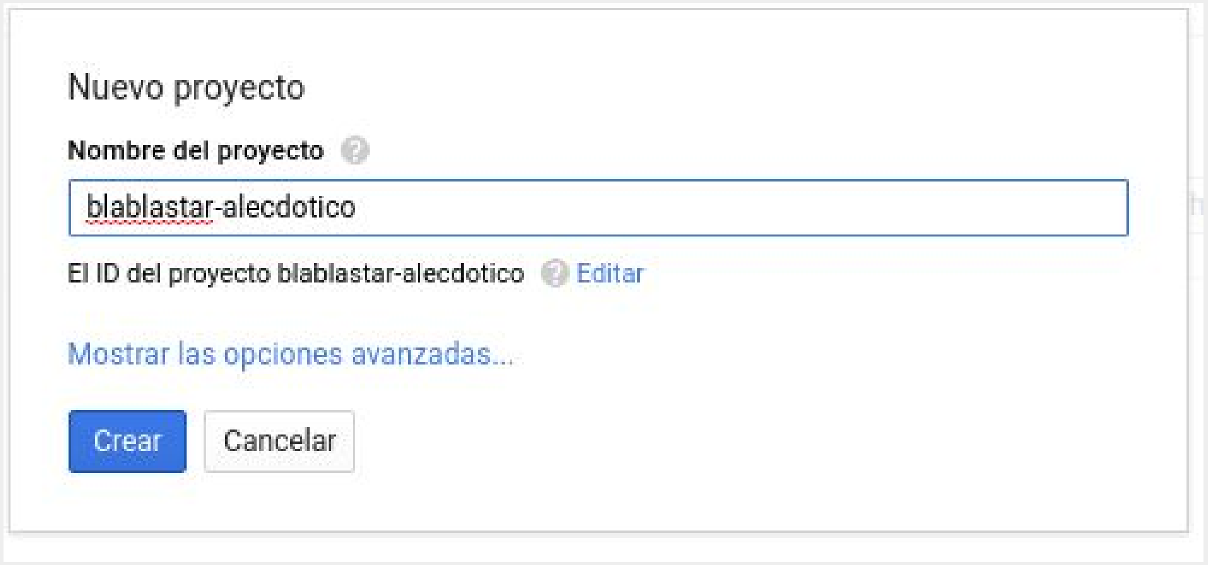
Run local GAE server

```
$ <GAE-SDK>/dev_appserver.py .
```

```
Starting module "default" running at: http://127.0.0.1:8080
Starting admin server at: http://localhost:8000
```

Deploy your project *in 10 lines only*

Create new project in https://console.cloud.google.com/appengine


Nuevo proyecto

Nombre del proyecto

blablastar-alecdotico

El ID del proyecto blablastar-alecdotico Editar

Deploy project in production `$ <GAE-SDK>/appcfg.py update . --no_cookies`

Deployed in https://<id-proyecto>.appspot.com

GAE Workshop: BlaBlaStar

BLABLASTAR

Save for your trips and ride-share through the galaxy, really cheap!

- web server: WSGI app
- Instances, modules, versioning
- Bootstrap
- noSQL datastore (NDB)
- Memcache
- Tasks, queues, cron jobs
- Picture uploading
- Unit Testing
- PubSub

WSGI web server

Web Server Gateway Interface using webapp2 library:

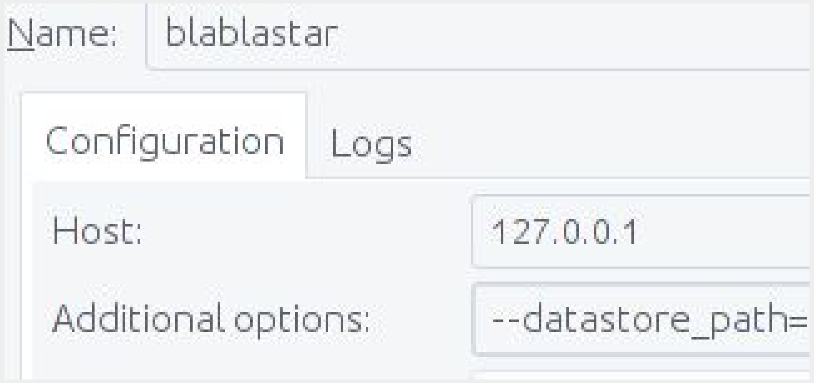
- RequestHandler: web controllers that implement HTTP methods
- WSGIApp: URL mapping with controllers
- Templates: generating dynamic and maintainable content

Move to the repository tag *step1-wsgi* to browse the example code and see how to implement a WSGI server in GAE with the following command:

```
$ git checkout step1-wsgi
```

GAE local configuration

Configure the datastore file, blobstore and search indexes in PyCharm: Run -> Edit Configurations -> Additional Options. Use any path in your filesystem.



Name: blablastar

Configuration | Logs

Host: 127.0.0.1

Additional options: --datastore_path=

```
--datastore_path=<path>/blablastar/datastore/datastore --
blobstore_path=<path>/blablastar/datastore/blobstore --
search_indexes_path=<path>/blablastar/datastore/searchidx
```

Storage

ndb datastore client mapping library on top of BigTable. NoSQL, highly scalable, transparent to the user

```
$ git checkout step2-ndb
```

- StarTrip model. Create and edit.
- Location a model for destinations. Create and edit
- StarTrip listing
- Filter by date, origin and destination

Proposed exercises:

1. Model Comments inside each trip, edition and deletion.
2. Model users and a recommending system: opinions and rating created by users after each trip

https://cloud.google.com/appengine/docs/python/ndb/

This is 2016! Bootstrap

Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.

```
$ git checkout step3-bootstrap
```

Advices:

1. Never write CSS from scratch. Rely on existing frameworks and/or preprocessors (SASS, LESS, ...)
2. Value the need of UX and UI in your product, it is a part of the development and requires both design and programming skills

http://getbootstrap.com/

Memcache

Caché service in memory, distributed, to speed up and save costs of frequently access data

```
$ git checkout step4-memcache
```

- Caching of locations

Proposed exercises:

1. Search caching by filters (date, origin, destiny, etc.)
2. Use ndb hooks on trip edition to delete cached searches from exercise 1

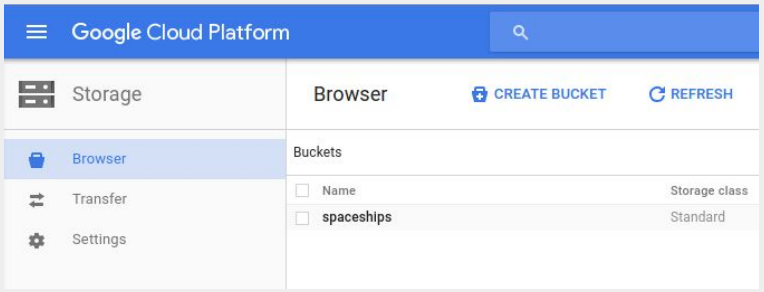
https://cloud.google.com/appengine/docs/python/memcache/

Blobs, images and CloudStorage

- GCS: library for uploading and serve files in buckets inside your application.
- Images API: picture manipulation service

```
$ git checkout step5-blobs-images
```

- Spaceship modelling, including a picture



Name	Storage class
spaceships	Standard

Requiere bucket *spaceships*

Exercises:

1. Attach a Spaceship to each StarTrip on the trip creation/edition
2. Add a *picture* field to Location model, uploaded on location edition. Display origin/destination pictures in a trip

https://cloud.google.com/appengine/docs/python/images/

Tasks and cron jobs

Task Queue: *tasks* queueing system, asynchronously executed by application modules

```
$ git checkout step6-tasks
```

- Task: job to calculate the most frequent origins and destinations in the last 5 days
- Cron job: run this task every day on a secondary module

Exercises:

1. StarTrip price suggestion engine: cron job that calculates the average price of a trip (origin-destiny) based on the latest 100 trips per permutation.
2. Calculate the most frequent destinations using a scalable algorithm that doesn't overflow the memory, for any number of trips and locations.

https://cloud.google.com/appengine/docs/python/taskqueue/

Unit Testing

GAE includes the unittest library and testbed module for stubbing

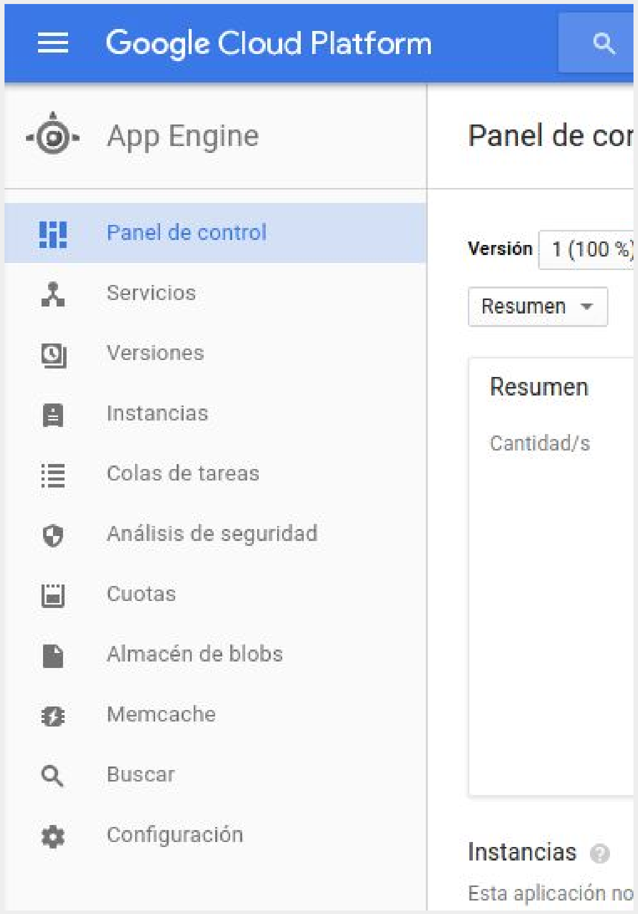
```
$ git checkout step7-unittest
```

- trip_exists: validate the correct creation of a trip with valid origin and destination.

Exercises:

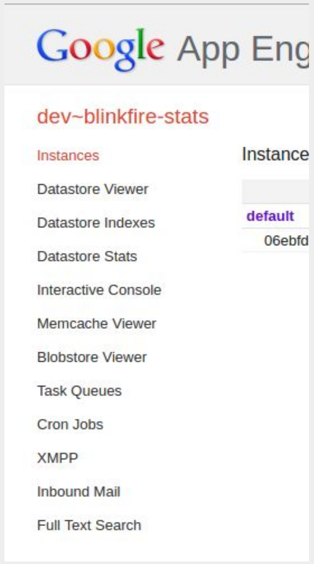
1. Validate the creation, edition and deletion of comments of a trip

Project Management: Developers Console



Application management console with access to all App Engine and Google Cloud features

Simplified version for local environments



```
Starting module "default" running at:
http://127.0.0.1:8080
Starting admin server at: http:
//localhost:8000
```

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

Appendix: exercises

1. Model a seat booking system for StarTrips
2. Model Galaxy routes, distances and costs. User will see a map, the approximate cost and the interstellar trip time on the trip edition.
3. The Galactic Empire demands its fees! Implement 3 types of routes: with tolls (safe and fast but expensive) and without (dangerous)
4. Trip edition: There are too many locations to be contained in a combobox. Implement locations with an autocomplete engine, using search API as document index
5. Model users with users API. Create a log in/out system with user dashboard with *my trips, my bookings, my messages, rating system* and *user profile*.
6. Implement a payment gateway to pay trip bookings through PayPal, adding a commission for BlaBlaStar on top of the price.
7. Back-office: create an administration module for BlaBlaStar team. Statistics, users, trip edition.
8. Implement a PubSub system that notifies (push) new trips to any user that is using the search
9. Upon user actions, add notification messages on the next screen, stored in the cookie session

BLABLASTAR

FLAMINGTEXT.COM

Share your journey to Alderaan, the soul of the Galaxy, and enjoy those beautiful mountains before they disappear!!

blinkfire analytics

That’s all, folks!

Questions, suggestions?

alex.tatay@blinkfire.com

Interested in Blinkfire? **Apply** **NOW**

jobs@blinkfire.com