Introduction to Google App Engine with Python

Brian Lyttle (http://brianlyttle.com)
@brianly on Twitter

What is Google App Engine?

- Google call it “cloud development in a box”.
- It is a hosted development environment along the lines of Microsoft Azure.
- Targets users building internet applications, but that is changing.
- Built on the same scalable platform used for other Google services.
- Development is done locally in Python (or Java) on Windows, Mac OS X, or Linux.
- Push to the cloud and Google will manage everything.
- We’ll dig into the details shortly.

Who’s using App Engine?

socialwok lingospot Walk Score™

GIFTAG pick anything GigaPan caucho

CloudStatus powered by HYPERIC EZ asset.com BY STRONGTECH

- Case Studies
  - http://code.google.com/appengine/casestudies.html
- Most often GAE is used as a utility alongside another platform.
- GAE for Business will increase uptake. Eventually.

How much does it cost?

- It is free to get started, or run small applications.
  - ~5M page views per month
- Google post quotas and costs on their website
  - http://code.google.com/appengine/docs/quotas.html
- Once you reach certain limits Google will start charging you based on your usage.
- Usage costs can be expensive depending on your requirements, or alternative deployment platforms.
  - Choice of datastore can even impact costs
- Calculating costs can be tricky, but this is the case with all usage-based cloud providers and services.
  - You need to know your application, and the how the platform can be best used applied to your needs.
  - Cheaper features might have potential for higher lock-in.

Mapping use cases to App Engine

- You can’t really choose a framework or tool without some experience. Take this advice with a pinch of salt.
- Good
  - You want to learn a Python web development framework.
  - Managing servers is not something you want to do.
  - Your application needs to support a ridiculous number of users, or giant fluctuations in usage.
  - You want to quickly prototype a larger application that you may develop elsewhere.
- Maybe
  - An existing application is written in Python (or Java) and you want to create a web front end.
- Bad
  - Data sources for your application live inside your company.
  - Other data security or privacy requirements.
  - You need a version of Python newer than 2.5.
  - Your Python library requires a C extension and does not have a Python fallback.

Learning Python

- Zed Shaw's "Learn Python the Hard Way" is recommended.
  - http://learnpythonthehardway.org/index
- Many other resources are available:
  - http://diveintopython.org/toc/index.html
  - http://docs.python.org/tutorial/
- Focus on learning Python 2.x if you want to use the broadest range of libraries today.
  - Python 3.2 is the latest but newbies will have issues with libraries.
- GAE makes the choice easier. Use **CPython 2.5**.
- Get familiar with REPL-style development in the shell.

Development tools

- Python 2.5
  - Only this version is supported at present.
  - Free from http://www.python.org/ftp/python/
  - Get the latest 2.5.x release from http://www.python.org/ftp/python/2.5.5/
- Google App Engine SDK
  - Download the latest version and it will auto-update.
  - Free from http://code.google.com/appengine/downloads.html
- A suitable editor
  - Most (good) text editors provide syntax highlighting.
  - I like JetBrains' PyCharm, but ActiveState Komodo is also good. A lot of people like the Wingware IDE.
  - You could use the latest Python support for Visual Studio 2010 if you want but it does not have special support for App Engine.

Documentation and samples

- All docs can be downloaded as a ZIP
  - http://code.google.com/appengine/downloads.html - Download_the_Google_App_Engine_Documentation
- Documentation
  - http://code.google.com/appengine/docs/python/overview.html
- Sample Code
  - http://code.google.com/p/google-app-engine-samples/
  - http://code.google.com/p/rietveld/
  - https://github.com/search?langOverride=&language=python&q=google+app+engine&repo=&start_value=1&type=Repositories&x=0&y=0
  - http://brizzled.clapper.org/id/77/
- Videos
  - Google IO Conference on YouTtube.
  - PyCon: http://pycon.bliptv.com

Services provided by App Engine

- Services are accessed via “webapp”.
- Most have equivalents in the .NET world (shown in parentheses)
- Sites hosted at *.appspot.com
- Limitations:
  - SSL for custom domains
  - Quotas impacting cost are complex
  - Users all have Gmail emails by default
  - Python libraries which depend on C extensions cannot run on the platform.
  - Google-like search requires some effort.

Datastore (SQL Server)

Blobstore (File system)

Capabilities

Channel (comet?)

Images (ImageGlue.NET)

Mail (BCL)

Memcache (AppFabric)

Multitenancy (SQL Schemas)

OAuth (BCL)

Prospective Search

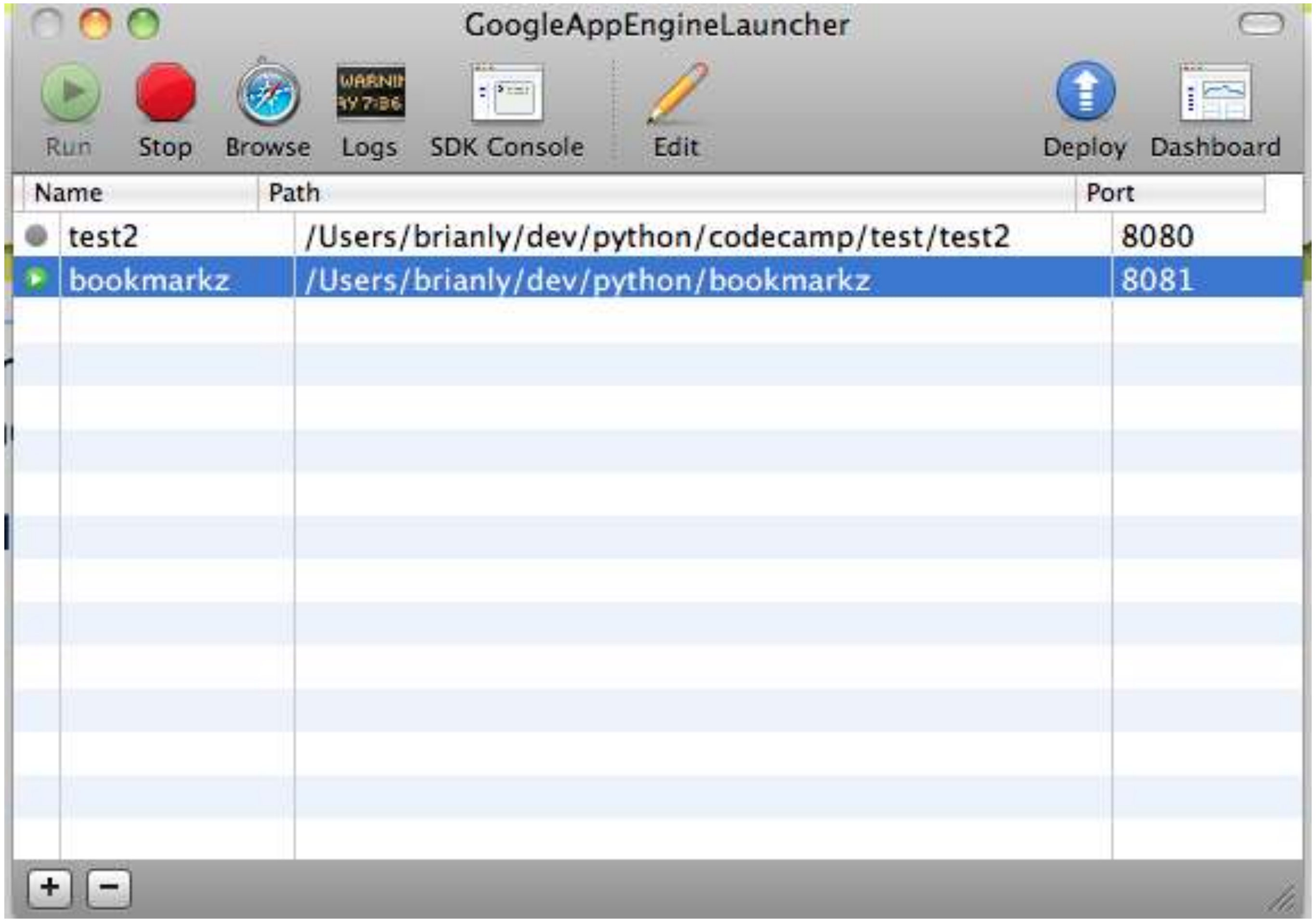
Task Queues (MSMQ)

URL Fetch (BCL)

Users (BCL)

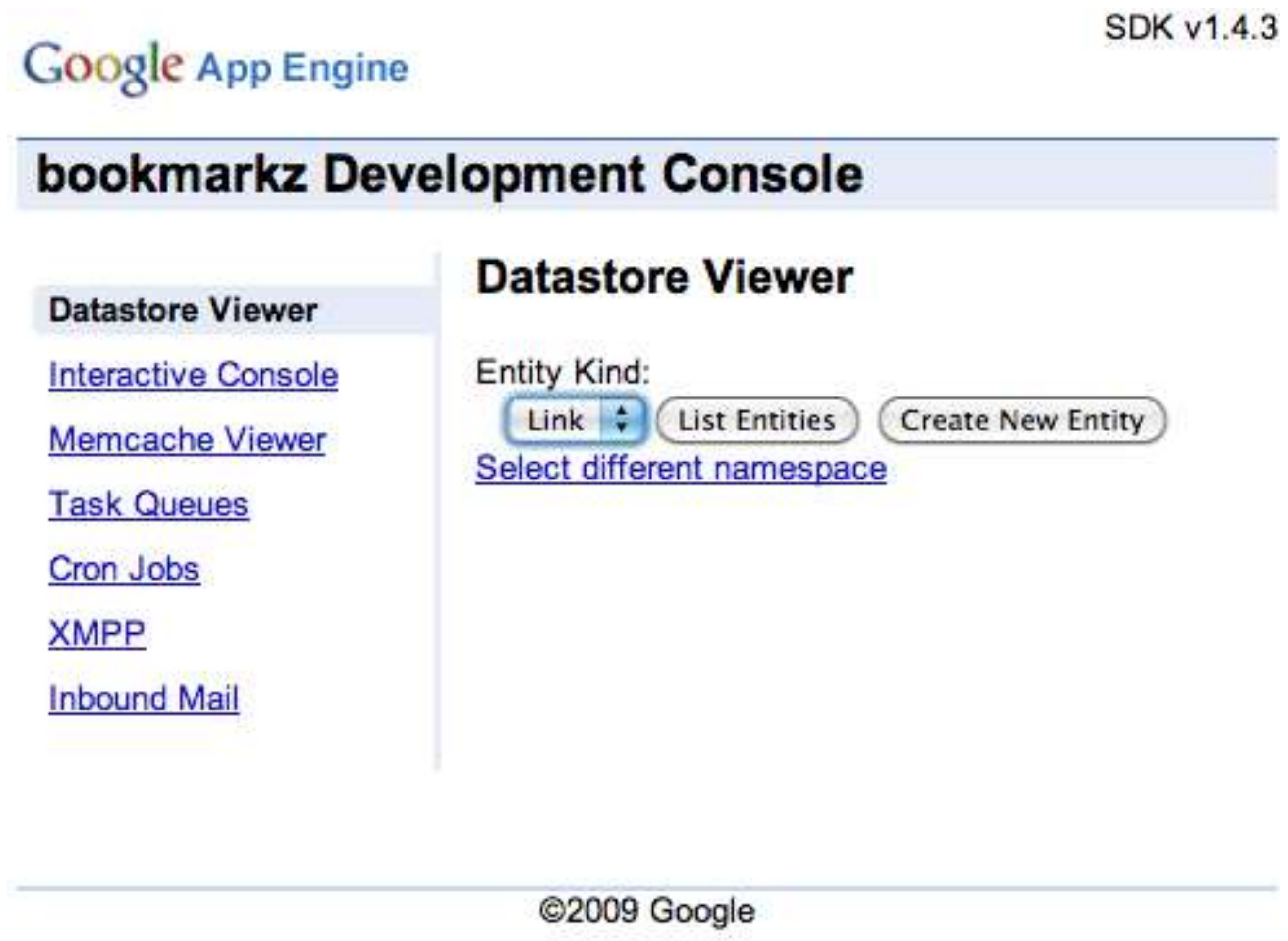
XMPP (Lync API)

App Engine Launcher



	Name	Path	Port
●	test2	/Users/brianly/dev/python/codecamp/test/test2	8080
▶	bookmarkz	/Users/brianly/dev/python/bookmarkz	8081

SDK Console


Google App Engine

SDK v1.4.3

bookmarkz Development Console

Datastore Viewer

Entity Kind: Link

©2009 Google

Basic App Engine demo

- Bookmarkz
  - Bookmark database
  - URL shortener
- Using the default “webapp” framework
- Source code is available from https://bitbucket.org/brianly/bookmarkz/src

Webapp framework

- Based on a basic framework called WebOb that runs atop WSGI.
  - WSGI is an interface to Python web servers.
- Whilst you have basic support for MVC, you don’t have the support provided by a full framework like ASP.NET MVC.
- Convention or configuration? Neither.
- For a lot of applications you should be considering a framework like Tipfy (more later).

Example handler

```
from google.appengine.ext import webapp
from google.appengine.ext.webapp.util import run_wsgi_app

class MainPage(webapp.RequestHandler):
    def get(self):
        self.response.headers['Content-Type'] = 'text/plain'
        self.response.out.write('Hello, webapp World!')

application = webapp.WSGIApplication(
                                     [('/', MainPage)],
                                     debug=True)


def main():
    run_wsgi_app(application)

if __name__ == "__main__":
    main()
```

Datastore

- You Entities based on Classes which derive from a special base class.
- Google provide a range of data types for Entity properties.
- Entities are stored in a system based on Big Table (Master/Slave) or Megastore (High Replication).
  - You choose this when you create your application.
- Choice of storage affects costs and can't be changed easily.
- Google recommend the High Replication datastore for better all round performance.
  - Fewer peaks and troughs compared to the Master/Slave store.

Working with Models

- Models are defined in Python in a similar fashion to other ORM tools.
- The challenges are a little different from your experiences using ORMs with relational databases.
- Data types
  - http://code.google.com/appengine/docs/python/datastore/typesandpropertyclasses.html

Definition

```
class Song(db.Model):
    author = db.UserProperty()
    composer = db.StringProperty()
    date = db.DateTimeProperty
             (auto_now_add=True)
    tags = db.StringListProperty()
    description = db.TextProperty()
```

Query

```
query = GqlQuery("SELECT * FROM
         Song WHERE composer
         = :composer",
         composer="Lennon, John")

query = db.Query(Song)
query.filter('composer=', id)
result = query.get()
```

More Models

- Expando and PolyModel.
- More information can be found in the SDK documentation

```
class Person(db.Expando):
    first_name = db.StringProperty()
    last_name = db.StringProperty()
    hobbies = db.StringListProperty()
```

```
class Contact(polymodel.PolyModel):
    phone_number = db.PhoneNumberProperty()
    address = db.PostalAddressProperty()
class Person(Contact):
    first_name = db.StringProperty()
    last_name = db.StringProperty()
    mobile_number = db.PhoneNumberProperty()
class Company(Contact):
    name = db.StringProperty()
    fax_number = db.PhoneNumberProperty()
```

Blobstore

- Limits apply to object size in the datastore.
- Store large objects e.g. images or documents in the Blobstore
- Exceptions might be small thumbnails where they are always returned with related fields.
- Use the BlobstoreUploadHandler class to make uploading blob objects less painful.
- BlobstoreDownloadHandler provides a way to specify byte ranges.
- BlobReader gives you a stream-like API.
- Plays well with the Images API.

URL Fetch

- Enables requests to HTTP and HTTPS resources.
- Use for calls to REST (and SOAP) web services.
- Supports calls to your intranet through the Google Secure Data Connector (SDC).

```
from google.appengine.api import urlfetch

result = urlfetch.fetch(url="http://www.corp.example.com/sales.csv",
                        headers={'use_intranet': 'yes'})
if result.status_code == 200:
  parseCSV(result.content)
```

- Download size is limited to 32 MB.

Task Queues

- Allows you to do work later such as send a notification, or update a 3rd party system.

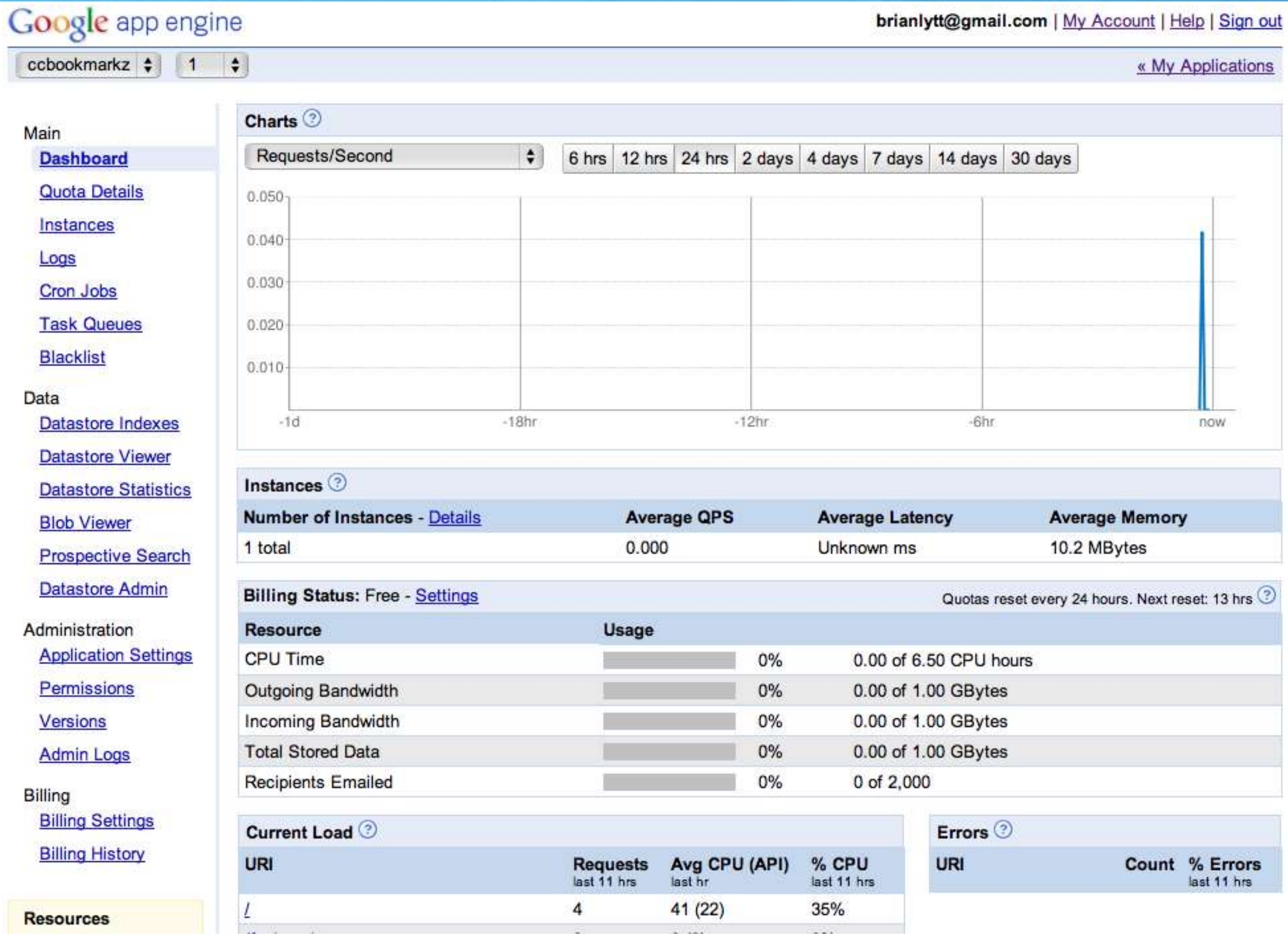
```
# Application queues a task
taskqueue.add(url='/worker', params={'key': key})

# Handler does the work
class CounterWorker(webapp.RequestHandler):
    def post(self): # should run at most 1/s
        key = self.request.get('key')
        def txn():
            counter = Counter.get_by_key_name(key)
            if counter is None:
                counter = Counter(key_name=key, count=1)
            else:
                counter.count += 1
            counter.put()
        db.run_in_transaction(txn)
```

Notes for Windows users

- Python works pretty well on Windows compared to other languages.
- Google's imaging library needs an extension called PIL.
- 64-bit versions of Python and the Python Imaging Library (PIL)
  - Make sure that the architectures of extensions match up.
  - Windows programs a 64-bit application cannot load a 32-bit DLL, and vice versa.
  - See my blog post on this topic.
- Windows Firewall and blocked ports
  - No server == no fun

App Engine Dashboard



Google app engine — brianlytt@gmail.com | My Account | Help | Sign out

ccbookmarkz | 1 — « My Applications

Main
- Dashboard
- Quota Details
- Instances
- Logs
- Cron Jobs
- Task Queues
- Blacklist

Data
- Datastore Indexes
- Datastore Viewer
- Datastore Statistics
- Blob Viewer
- Prospective Search
- Datastore Admin

Administration
- Application Settings
- Permissions
- Versions
- Admin Logs

Billing
- Billing Settings
- Billing History

Resources

Charts

Requests/Second — 6 hrs | 12 hrs | 24 hrs | 2 days | 4 days | 7 days | 14 days | 30 days

Instances

Number of Instances - Details	Average QPS	Average Latency	Average Memory
1 total	0.000	Unknown ms	10.2 MBytes

Billing Status: Free - Settings — Quotas reset every 24 hours. Next reset: 13 hrs

Resource	Usage		
CPU Time		0%	0.00 of 6.50 CPU hours
Outgoing Bandwidth		0%	0.00 of 1.00 GBytes
Incoming Bandwidth		0%	0.00 of 1.00 GBytes
Total Stored Data		0%	0.00 of 1.00 GBytes
Recipients Emailed		0%	0 of 2,000

Current Load

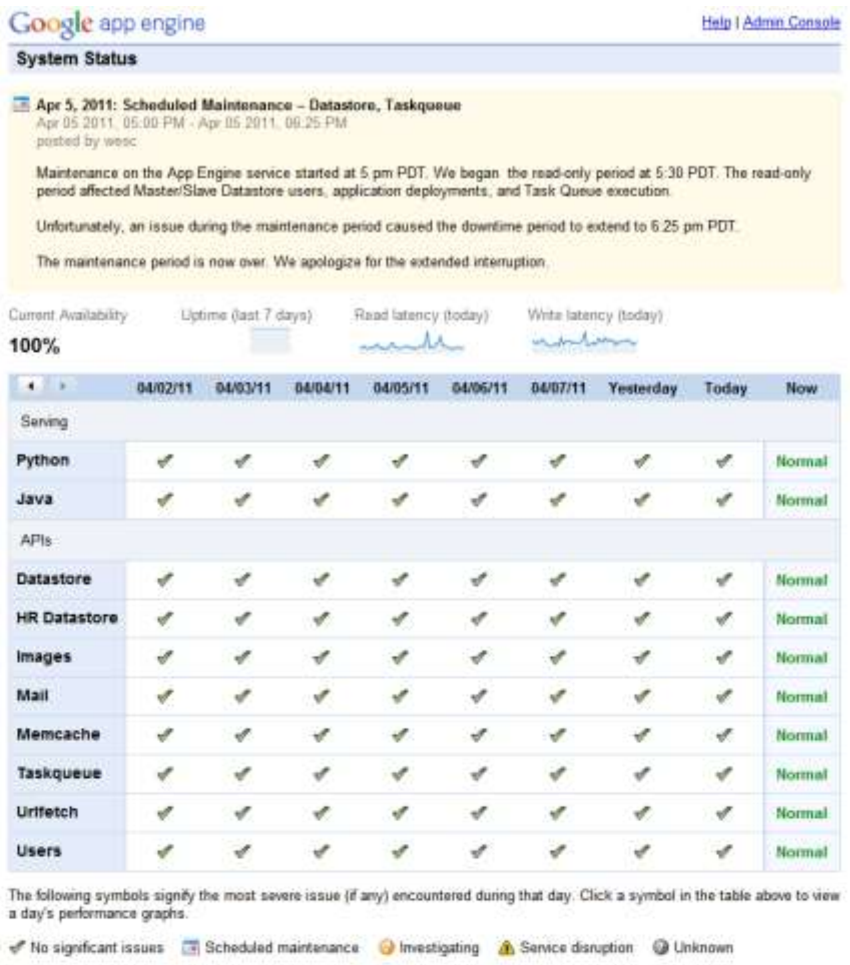
URI	Requests last 11 hrs	Avg CPU (API) last hr	% CPU last 11 hrs
/	4	41 (22)	35%

Errors

URI	Count	% Errors last 11 hrs

Health and uptime

- Check the status page if you experience issues.
- Situations
  - Read-only Datastore
  - “Regular” failures
- The Capabilities API is provided to enable you to handle maintenance periods.



Google app engine — Help | Admin Console

System Status

Apr 5, 2011: Scheduled Maintenance – Datastore, Taskqueue
Apr 05 2011, 05:00 PM - Apr 05 2011, 06:25 PM
posted by wesc

Maintenance on the App Engine service started at 5 pm PDT. We began the read-only period at 5:30 PDT. The read-only period affected Master/Slave Datastore users, application deployments, and Task Queue execution.

Unfortunately, an issue during the maintenance period caused the downtime period to extend to 6:25 pm PDT.

The maintenance period is now over. We apologize for the extended interruption.

Current Availability: **100%** — Uptime (last 7 days) — Read latency (today) — Write latency (today)

	04/02/11	04/03/11	04/04/11	04/05/11	04/06/11	04/07/11	Yesterday	Today	Now
Serving									
Python	✓	✓	✓	✓	✓	✓	✓	✓	Normal
Java	✓	✓	✓	✓	✓	✓	✓	✓	Normal
APIs									
Datastore	✓	✓	✓	✓	✓	✓	✓	✓	Normal
HR Datastore	✓	✓	✓	✓	✓	✓	✓	✓	Normal
Images	✓	✓	✓	✓	✓	✓	✓	✓	Normal
Mail	✓	✓	✓	✓	✓	✓	✓	✓	Normal
Memcache	✓	✓	✓	✓	✓	✓	✓	✓	Normal
Taskqueue	✓	✓	✓	✓	✓	✓	✓	✓	Normal
Urlfetch	✓	✓	✓	✓	✓	✓	✓	✓	Normal
Users	✓	✓	✓	✓	✓	✓	✓	✓	Normal

The following symbols signify the most severe issue (if any) encountered during that day. Click a symbol in the table above to view a day's performance graphs.

✓ No significant issues ▦ Scheduled maintenance ● Investigating ⚠ Service disruption ● Unknown

Better frameworks for App Engine

- The “webapp” framework is bare bones and based on WSGI.
- Any WSGI framework can run on App Engine.
- Kay
  - Components: Django, Werkzeug, Jinja2, babel
  - http://code.google.com/p/kay-framework/
- Tipfy
  - Custom framework with multiple template engines, and plugins.
  - http://www.tipfy.org/
- Django-nonrel
  - A NoSQL-optimized version of Django.
  - Wesley Chun’s from Google presented on this at PyCon.
  - http://www.allbuttonspressed.com/projects/django-nonrel

Run your own App Engine

- "Platform as a Service" can lock you in
  - Data lock-in is probably a bigger problem than frameworks.
- Possible solutions
  - Typhoon AE: http://code.google.com/p/typhoonae/
  - AppScale: http://sites.google.com/site/gaeasaframework/appscale
- Essentially these are a lot of pain to setup, and you are probably using the App Engine framework to solve the wrong problem.
- A better solution is to use a native Python framework:
  - Django: http://www.djangoproject.com/
  - Pyramid (Pylons): http://pylonsproject.org/
  - Flask: http://flask.pocoo.org/

Thanks!

- Don’t forget to provide provide feedback on my talk and others you attended today.
- Slides and code will be posted to http://brianlyttle.com shortly.