

light

@rantav
totango



Spotify™

WHO AM I?

An aerial photograph of a coastal landscape during sunset or sunrise. The horizon is visible in the distance, where the warm light of the sun meets a sky filled with scattered clouds. In the foreground, there's a mix of dark, shadowed land and lighter, possibly sandy or rocky, areas. A small, isolated bird is visible in the middle ground. The overall atmosphere is serene and contemplative.



WHO AM I?

- A developer
 - Google, Microsoft, Outbrain, Gigaspaces, Totango etc
 - FLOSS: Hector, flask-restful-swagger, meteor-migrations, monitoring...
 - Contributor to **Luigi**
- reversim.com
- [Gormim](http://Gormim.com)



We help developers find their dream job!

Curated weekly email of top companies seeking talented **software engineers in Israel**

Enter your email...

Subscribe



Nir Dremer

Nir builds products at the intersection of technology, business & psychology. He founded MediaBox and led products at Imperva (NYSE: IMPV) & BitBand (acquired by Motorola).



Ran Tavy

Ran is domain expert in web infrastructure and scaling. He founded SocialStudios and built products at Google, YouTube, Microsoft and Outbrain. Ran is also a blogger and a podcaster.

A nighttime satellite view of Earth from space, showing city lights and clouds.

WHAT IS LUIGI?

WHAT IS LUIGI?

- A **Workflow Engine**.
 - Who the fuck needs a workflow engine?
 - You do!!!
 - If you run hadoop (or other ETL jobs)
 - If you have dependencies b/w them (who doesn't?!)
 - If they fail (**s/if/when/**)
- Luigi doesn't replace Hadoop, Scalding, Pig, Hive, Redshift.
 - It **orchestrates** them



SCREENSHOTS

Luigi Task Visualiser

<https://pipeline.spotify.net/luigi/static/visualiser/index.html>

Failed Tasks

- WebplayerRequestTimeImporterMulti (1)
- WeeklyBrowseData (1)

Running Tasks

- ActivationDomino (0)
- AggregateClientCrashHockeyApp (10)
- CrashLogsToHdfs (15)
- DominoImportTask (1)
- ImpressionsAndUsers (0)
- OrganicMultiplier (2)
- RetentionDomino (10)
- UpdateLogAvroSchemas (3)
- UserUsageSummary (1)
- UTMAcquisitionShard (0)

AggregateClientCrashHockeyApp (test=False, date=2013-11-02)

AggregateWebPlayerUpgradeTest (1)

AndroidHockeyAppCrashes (10)

BandwidthConnectionJoin (1)

DominoImportTask (2)

DumpFslImage (1)

DumpMetadata (2)

DumpMetadata (date=2013-11-06, table=publishedaudio)

DumpMetadata (date=2013-11-06, table=audio)

DumpSuperviseStatus (0)

6:32:03 PM

Luigi Task Visualiser

[https://pipeline.spotify.net/luigi/static/visualiser/index.html#HockeyAppCrashesToPostgresMulti\(last_date=2013-11-06, days_back=15, days_back Strict=10, today=False\)](https://pipeline.spotify.net/luigi/static/visualiser/index.html#HockeyAppCrashesToPostgresMulti(last_date=2013-11-06, days_back=15, days_back Strict=10, today=False))

Luigi Task Status Active tasks

Dependency Graph

HockeyAppCrashesToPostgresMulti(last_date=2013-11-06, days_back=15, days_back Strict=10, today=False)

Dependency Graph

Legend: Failed (Red), Running (Blue), Pending (Yellow), Done (Green)

The dependency graph illustrates the relationships between various Luigi tasks. At the top center is the root task 'HockeyAppCrashesToPostgresMulti'. Numerous arrows point downwards from this central node to several intermediate nodes labeled 'CompleteDate' (yellow). These 'CompleteDate' nodes then have arrows pointing to more specific task nodes, such as 'AggregateClientCrashHockeyApp', 'AndroidHockeyAppCrashes', and 'DroidHockeyAppCrashes'. Some of these specific task nodes are shown in red (Failed), while others are green (Done). The graph is highly interconnected, with many tasks having multiple dependencies and being dependent on multiple other tasks.

HOW DO YOU ETL YOUR DATA?

- Hadoop
- Spark
- Redshift
- Postgres
- Ad-hoc java/python/ruby/go/...

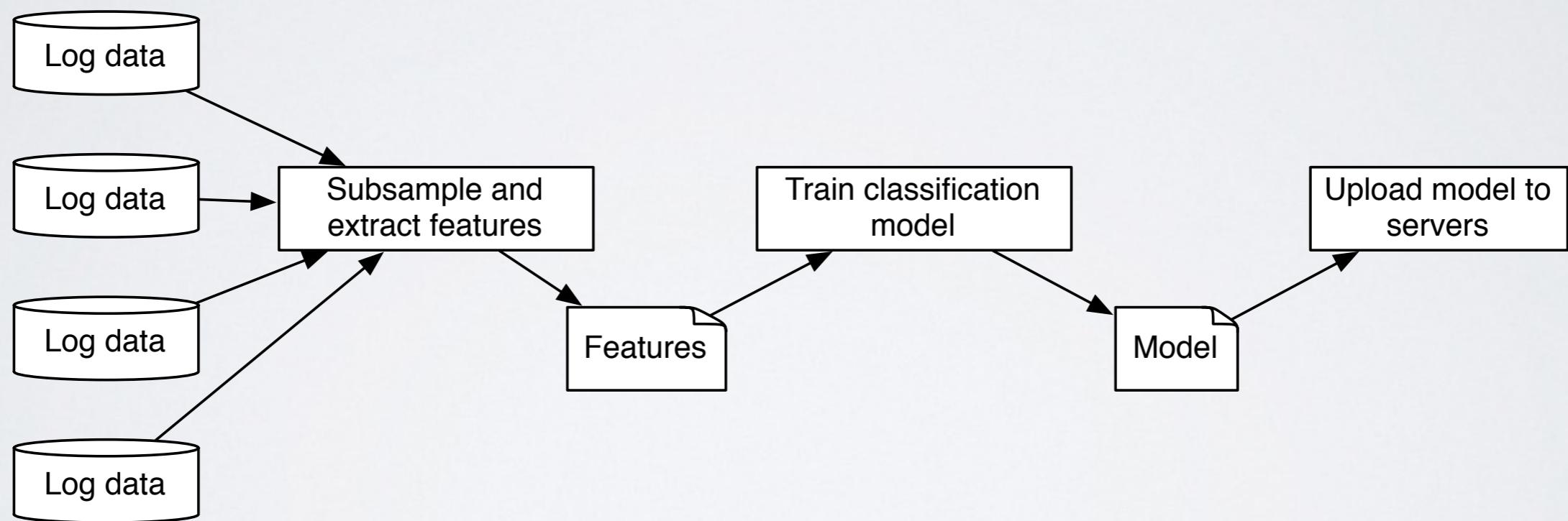
RUNNING ONE JOB IS EASY RUNNING MANY IS HARD

- 100s of concurrent jobs, 1000s Daily.
- Job dependencies.
 - E.g. first copy the file, then crunch it.
- Errors / retries
 - Idempotency
- Monitoring / Visuals

THE WRONG WAY TO DO IT



EXAMPLE WORKFLOW



THE CRON PHENOMENON

THE WRONG WAY TO DO IT

```
# m h dom mon dow user  command
0 1 * * * spotify-analytics /usr/bin/spotify_extract_features --date `date +%Y-%m-%d` # 1am
0 2 * * * spotify-analytics /usr/bin/spotify_train_model --date `date +%Y-%m-%d` # 2am
0 3 * * * spotify-analytics /usr/bin/spotify_update_model --date `date +%Y-%m-%d` # 3am
log files are available by 1am
this job finished by 2am
ugly!!!
uuu---F1 crontab      All L3      (Fundamental)-----
Wrote /Users/erikbern/pydata_pres/crontab
```

ENTER LUIGI



ENTER LUIGI

- Like Makefile - but in python
 - And - For data
- Integrates well with data targets
 - Hadoop, Spark, Databases
 - Atomic file/db operations
- Visualization
- CLI - really nice developer interface!



LUIGITASK

```
class MyTask(Task):
    def output(self):
        pass

    def requires(self):
        pass

    def run(self):
        pass
```

LUIGITASK

```
class AggregateArtists(Task):
    def output(self):
        return HdfsTarget("data/artist_streams.tsv")
    def requires(self):
        return Streams()
    def run(self):
        with self.input().open('r') as in_file:
            ... # read stuff from in_file
        with self.output().open('w') as out_file:
            ... # write stuff to out_file
    if __name__ == "__main__":
        luigi.run()
```

Store output in HDFS

Run - actual work

Read from input

Write to output

Run with __main__

RUN FROM THE CLI

Run on the command line:

```
$ python dataflow.py Aggreg
```

```
DEBUG: Checking if AggregateArtists() is complete
INFO: Scheduled AggregateArtists()
DEBUG: Checking if Streams() is complete
INFO: Done scheduling tasks
DEBUG: Asking scheduler for work...
DEBUG: Pending tasks: 1
INFO: [pid 74375] Running      AggregateArtists()
INFO: [pid 74375] Done          AggregateArtists()
DEBUG: Asking scheduler for work...
INFO: Done
INFO: There are no more tasks to run at this time
```

TASK PARAMETERS

```
class AggregateArtists(Task):  
    date = DateParameter()
```

```
$ python dataflow.py AggregateArtists --date 2013-03-05
```

AWESOME HADOOP (MR) SUPPORT

```
class AggregateArtists(luigi.hadoop.JobTask):
    def requires(self):
        return Streams()

    def output(self):
        return HdfsTarget("data/artist_streams.tsv")

    def mapper(self, line):
        timestamp, artist, track = line.split('\t')
        yield artist, 1

    def reducer(self, artist, streams):
        yield artist, sum(streams)
```

WEB UI

Luigi Task Status

Task List Dependency Graph

Upstream Failure

Failed Tasks

- + 1 AccountIndexDataLoaderTask
- + 3 ActiveListRealtimeTask
- + 1 AvgActivityLevelTask
- + 2 CreateUserDocumentTask
- + 1 DownloadSdrTask
- + 1 HadoopAccountAggregationTask
- + 1 HadoopUserAggregationTask
- + 2 UserDocumentsMergeTask

Pending Tasks

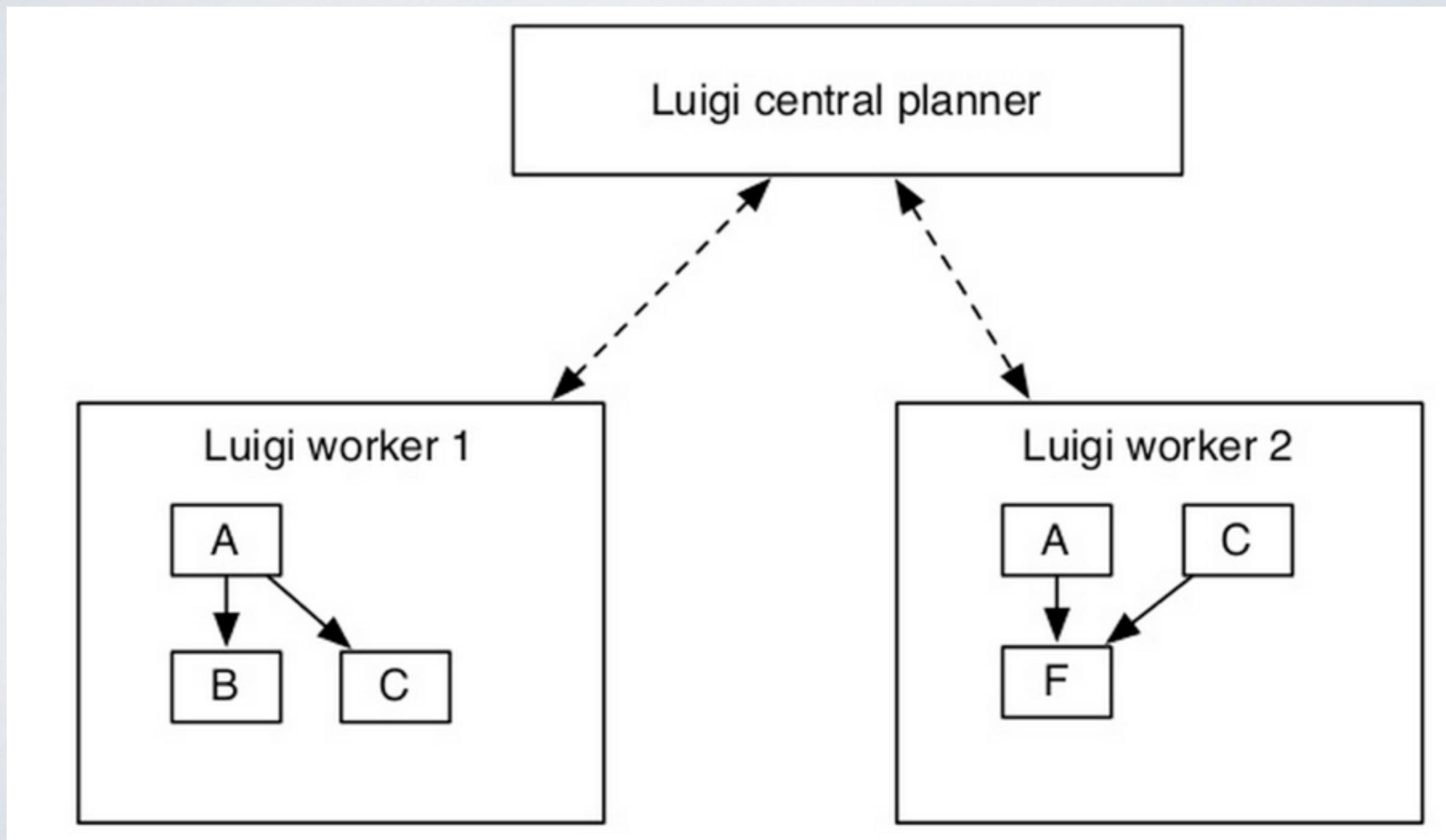
- + 1 AccountAgingTask
- + 3 AccountIndexDataLoaderTask
- AccountSecondIndexDataLoaderTask

AllDailyTask(service_id=██████████, day=2014-06-21, force_run=False, tz_string=-0600)

Dependency Graph

The dependency graph illustrates the relationships between tasks. The root node is AllDailyTask, which has numerous dependencies pointing to other tasks such as ActiveListOrganizationStateTask, ClearRealtimeAggregationsTablesTask, CreateAccountDocumentTask, PeriodicEndMarkerCommand, and several tasks under the UserEngagementScoreTa category.

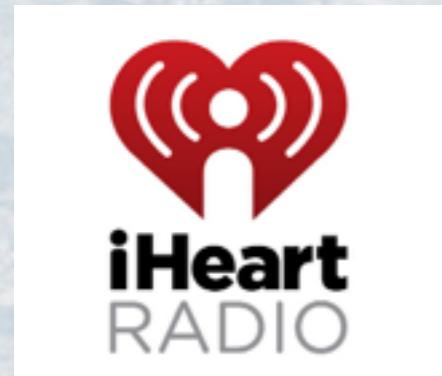
PROCESS SYNCHRONIZATION



USED BY



Spotify®



foursquare®



VisualDNA™



SEMI-DEEP DIVE

A photograph of a scuba diver swimming in the ocean. The diver is wearing a black wetsuit and fins, and is carrying a tank on their back. They are moving towards the left of the frame, creating a large cloud of bubbles behind them. The water is a deep blue color.

Programming for Luigi

LUIGI TASKS

- Implement 4 method:

def input(self) (optional)

def output(self)

def run(self)

def depends(self)

LUIGI TASKS

- Or extend one of the predefined tasks
 - **S3CopyToTable**
 - **RedshiftManifestTask**
 - **SparkJob**
 - **HiveQueryTask**
 - **HadoopJobTask**
 - ...

EXAMPLE LOCAL WORDCOUNT

```
class WordCount(luigi.Task):
    date_interval = luigi.DateIntervalParameter()

    def requires(self):
        return [InputText(date) for date in self.date_interval.dates()]

    def output(self):
        return luigi.LocalTarget('/var/tmp/text-count/%s' % self.date_interval)

    def run(self):
        count = {}
        for file in self.input():
            for line in file.open('r'):
                for word in line.strip().split():
                    count[word] = count.get(word, 0) + 1

        # output data
        f = self.output().open('w')
        for word, count in count.iteritems():
            f.write("%s\t%d\n" % (word, count))
        f.close()
```

EXAMPLE

HADOOP WORDCOUNT

```
class WordCount(luigi.hadoop.JobTask):
    date_interval = luigi.DateIntervalParameter()

    def requires(self):
        return [InputText(date) for date in self.date_interval.dates()]

    def output(self):
        return luigi.hdfs.HdfsTarget('/tmp/text-count/%s' % self.date_interval)

    def mapper(self, line):
        for word in line.strip().split():
            yield word, 1

    def reducer(self, key, values):
        yield key, sum(values)
```

LUIGITARGETS

- HDFS
- Local File
- Postgres / MySQL, Redshift, ElasticSearch
- ... Easy to extend

DEFINING A TARGET

- Implement:

```
def exists(self)
```

And optionally:

```
connect or open / close
```

EXAMPLE

MYSQL TARGET

```
class MySqlTarget(luigi.Target):

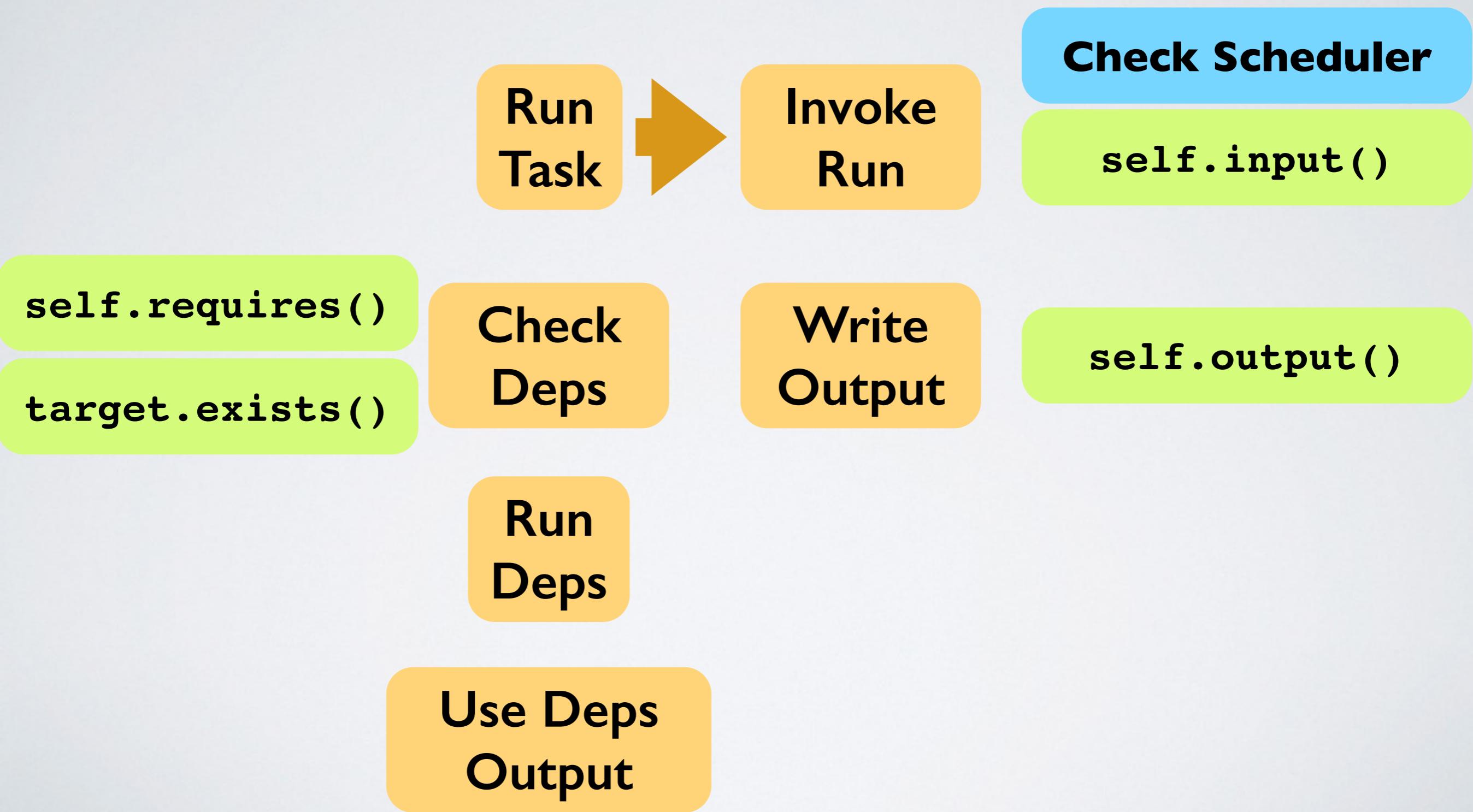
    def touch(self, connection=None):
        ...

    def exists(self, connection=None):
        cursor = connection.cursor()
        cursor.execute("""SELECT 1 FROM {marker_table}
                           WHERE update_id = %s
                           LIMIT 1""".format(marker_table=self.marker_table),
                      (self.update_id,))
        row = cursor.fetchone()
        return row is not None

    def connect(self, autocommit=False):
        ...

    def create_marker_table(self):
        ...
```

THE GRAND SCHEME



OPEN SOURCE

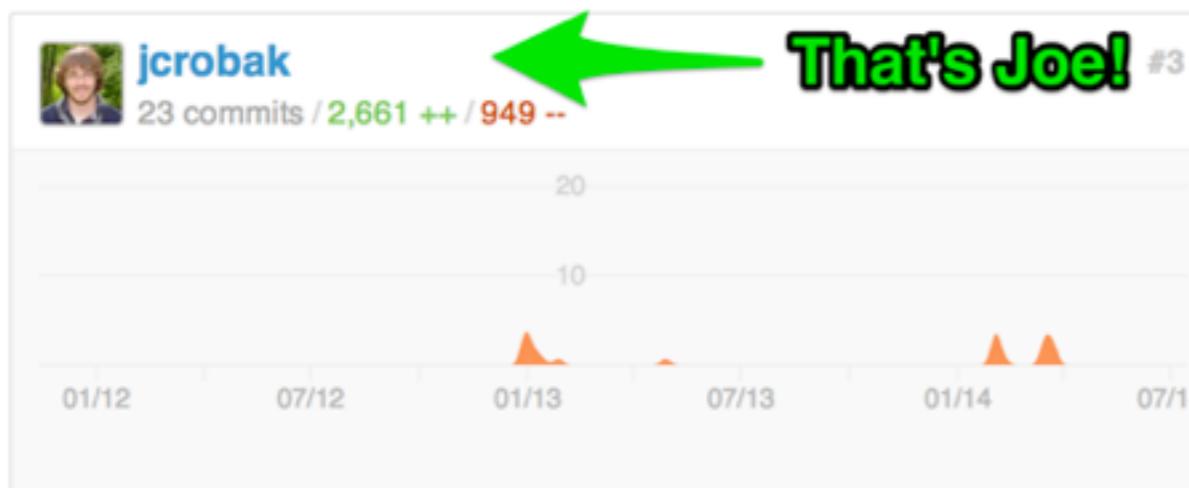


BY

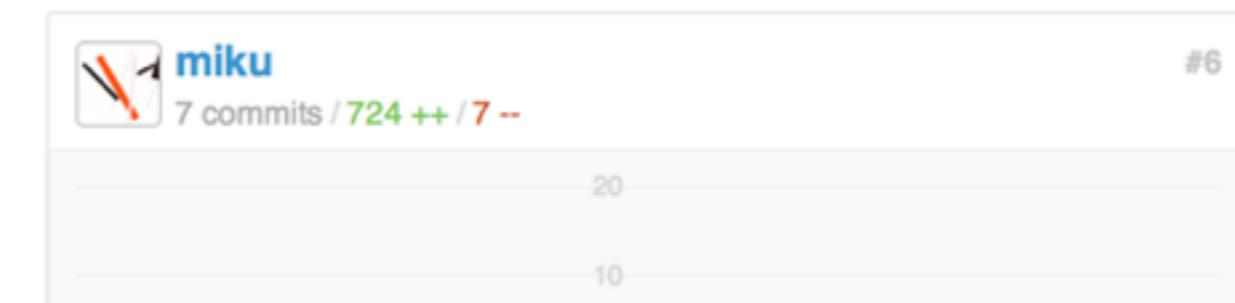
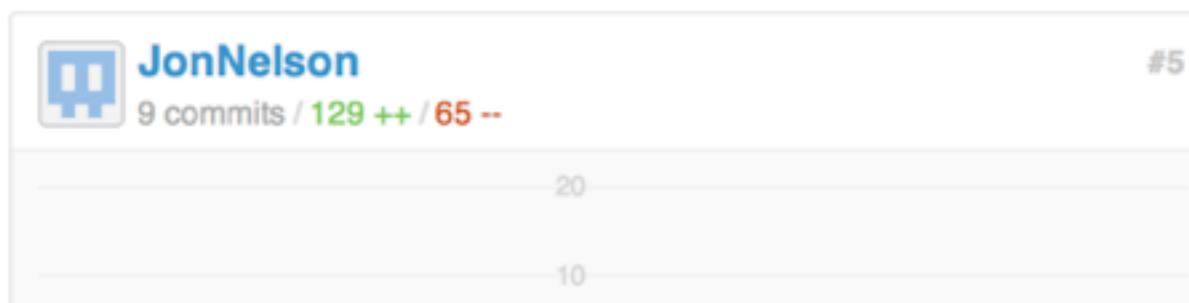
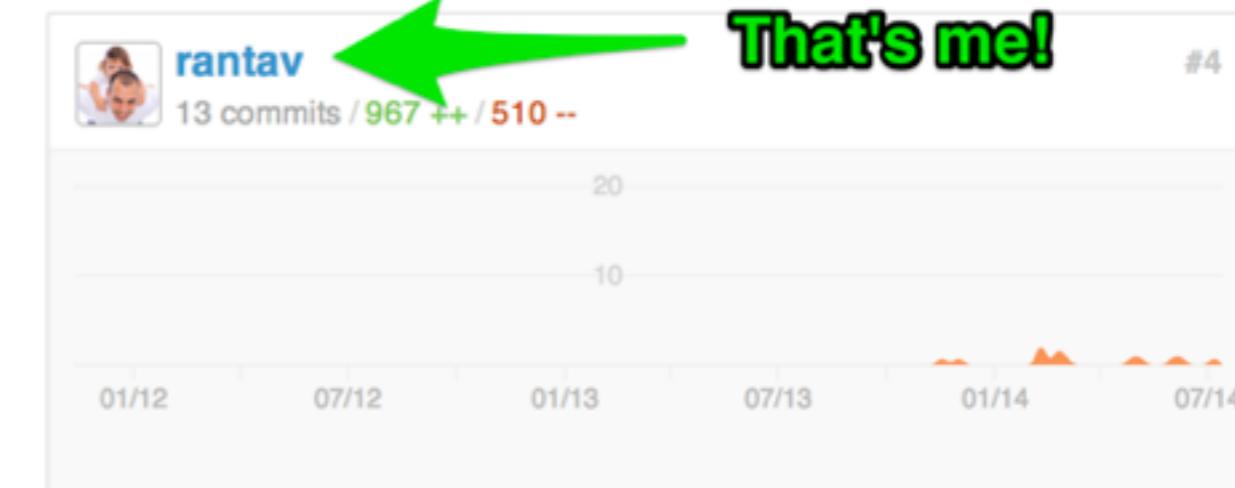




authors,
spotify



That's Joe!



authors,
spotify

That's me!

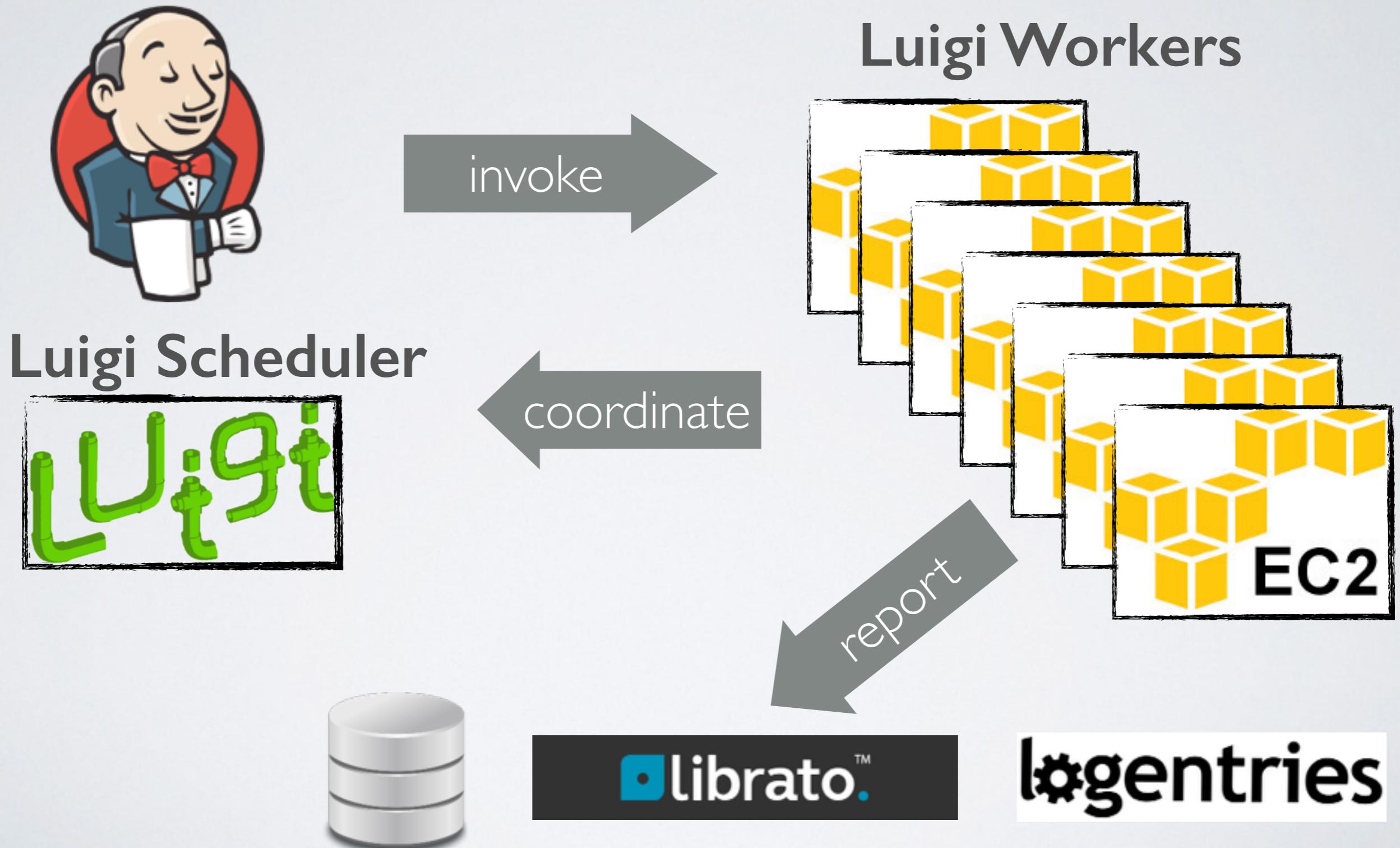
WHAT DID I DO?

- Add **Redshift** support
- Add **MySQL** support
- Various small features (improved **notifications**, **dep.py**, **historydb** etc)
- Various bug reports
 - And fixes!

LUIGI @ TOTANGO

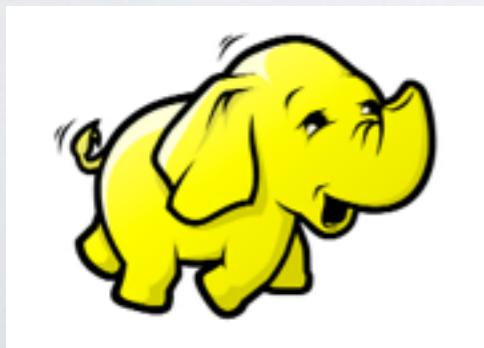
- Daily computation
- Hourly computation
- Ad-hoc data loading (for data analysis activities, to redshift)

TOTANGO'S SETUP



TOTANGO'S SETUP

Luigi Worker



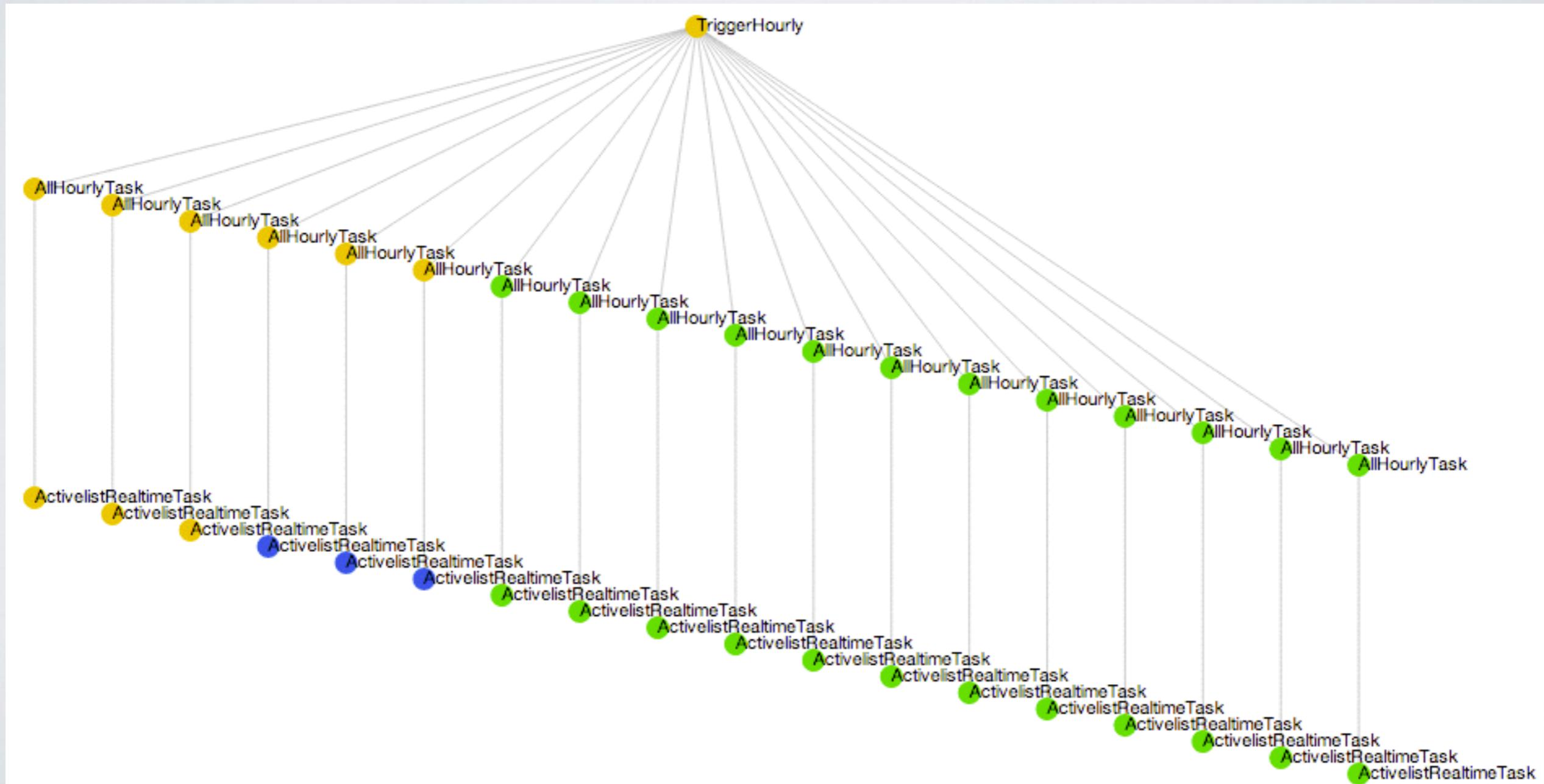
Spark



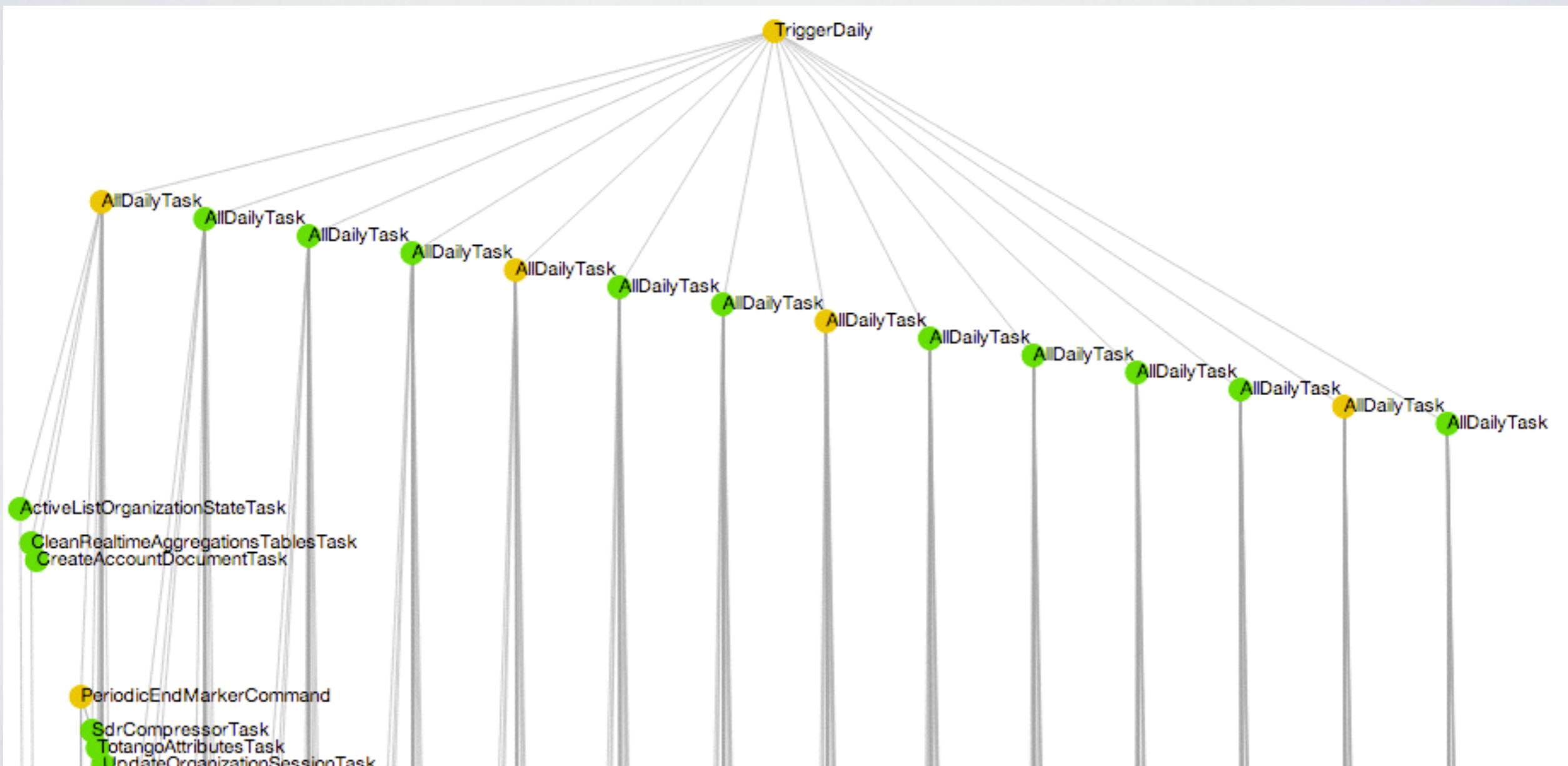
MOAR SCREENSHOTS



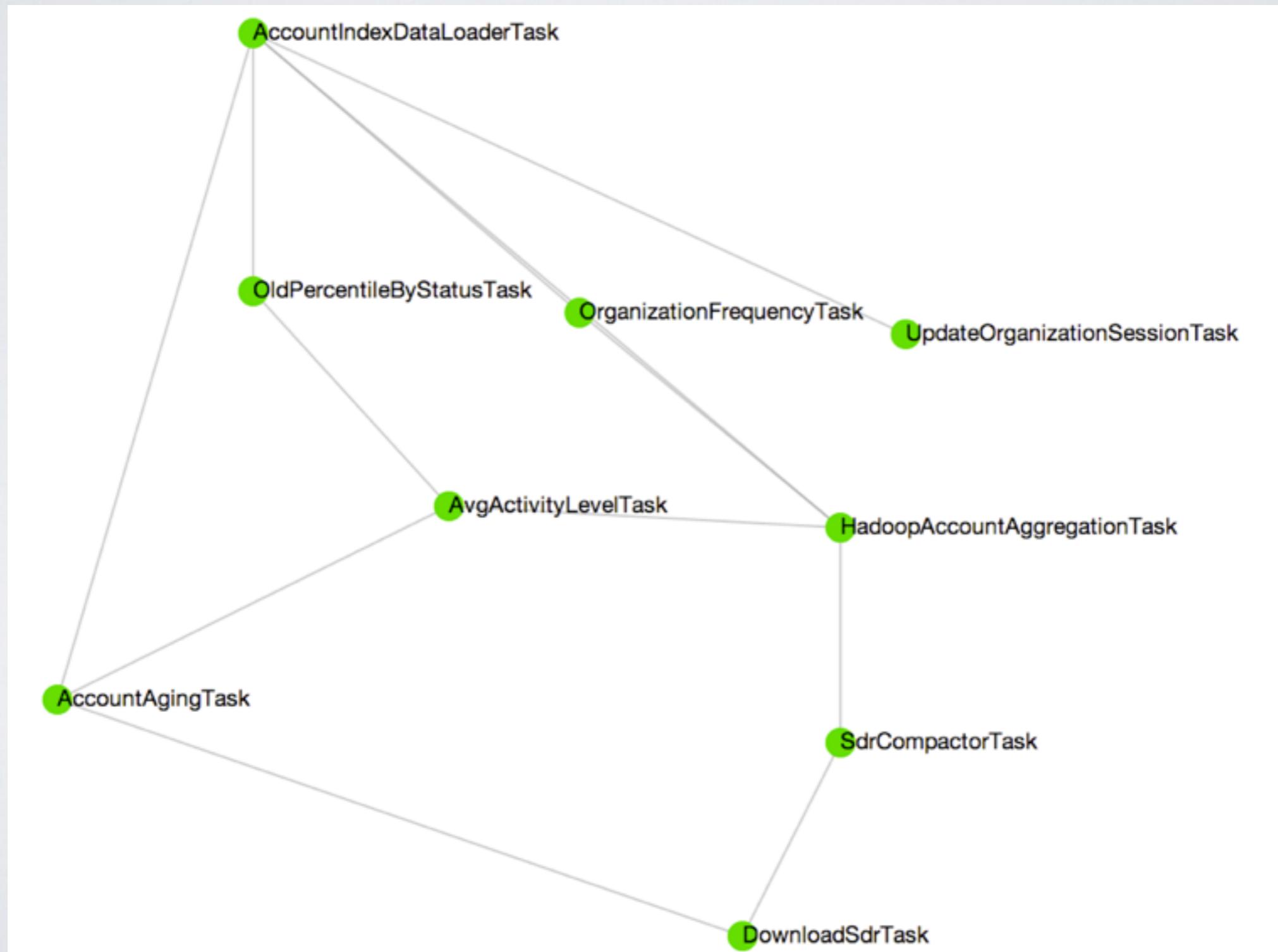
MOAR SCREENSHOTS



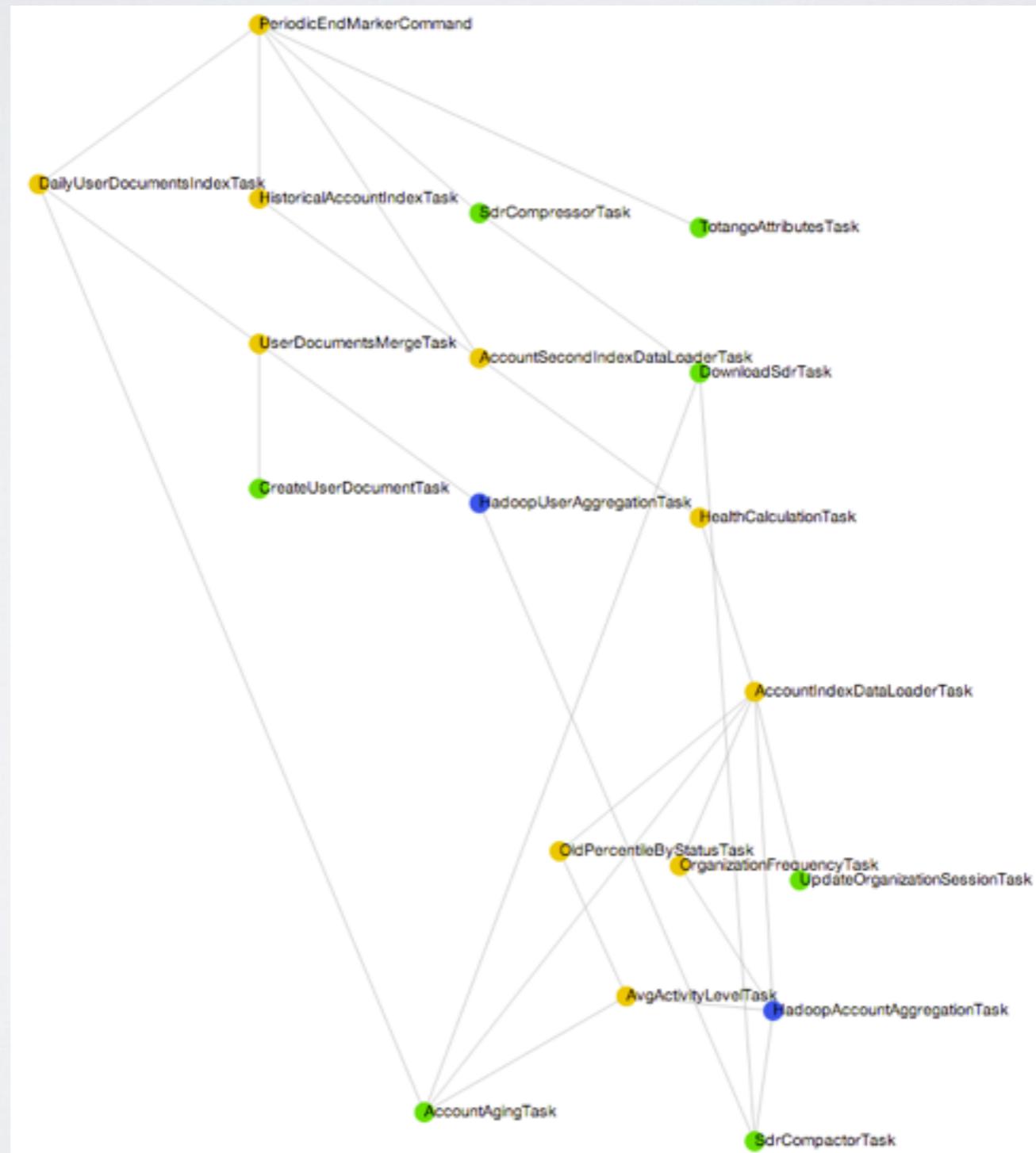
MOAR SCREENSHOTS



MOAR SCREENSHOTS



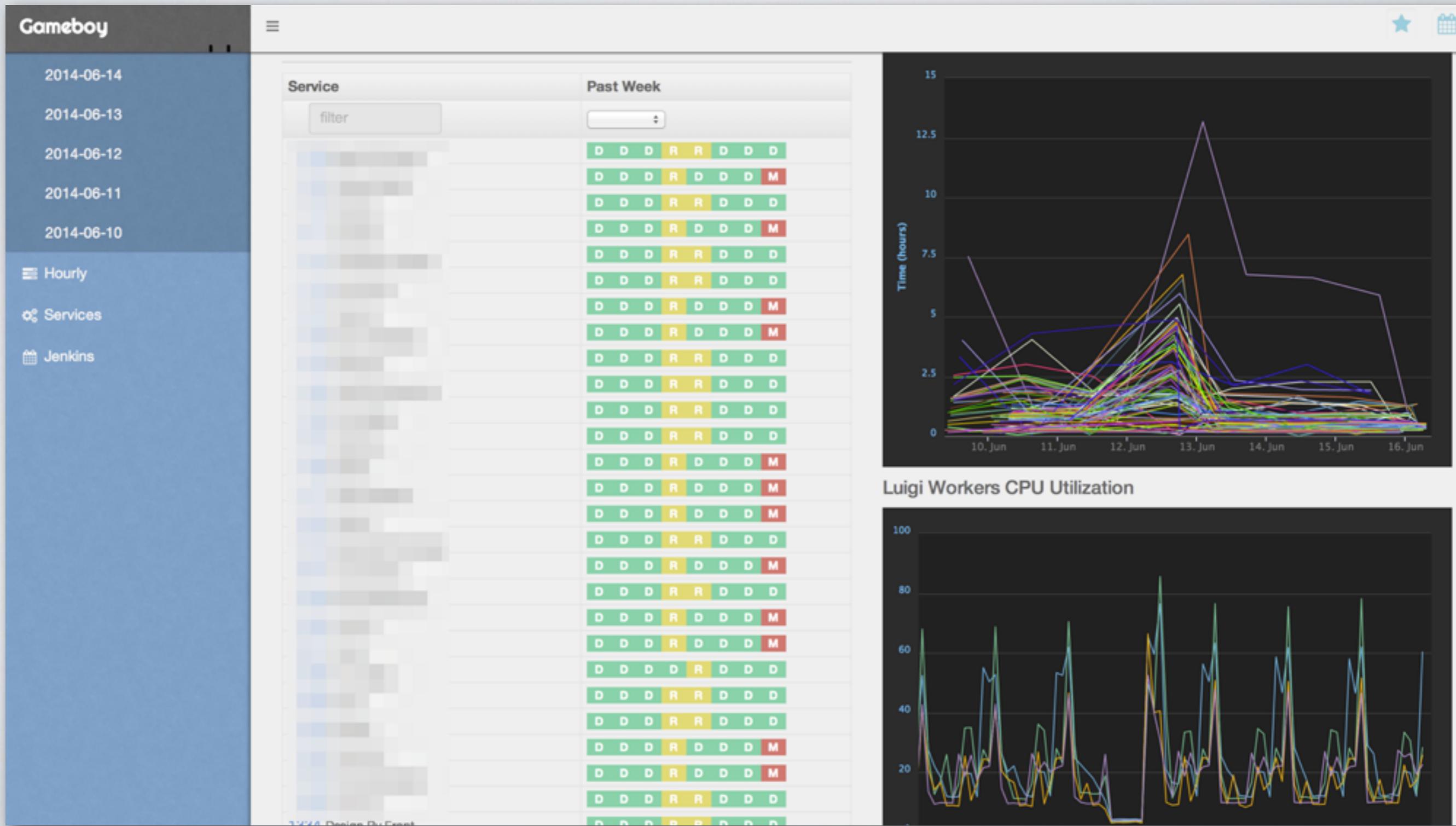
MOAR SCREENSHOTS



GAMEBOY!!!



AND... GAMEBOY



GAMEBOY

Gameboy  

 [Dashboard](#)  [Home](#) / [Dashboard](#)

 [Daily](#)  [Hourly](#)  [Services](#)  [Jenkins](#)

TODAY

130 (30 pending/running)
DAILY
96 services (96 in luigi)

2351 (56 pending/running)
HOURLY
96 services (96 in luigi)

0 **FAILED**

JENKINS QUEUE 0

PAST WEEK DAILIES

Service	Past Week
	D D D D D D D P
	D D D D D D D P
	D D D D D D D P
	M D D D D D D D
	M D D D D D D D
	D D D D D D D P
	D D D D D D D P
	M D D D D D D D
	D D D D D D D P
	M D D D D D D D
	M M M M D D D D
	M D D D D D D D
	M D D D D D D D
	M D D D D D D P
	D D D D D D D P

RUNNING/PENDING/FAILED TODAY

Task	Status	Service
DailyUserDocumentsIndexTask { "service_id": "1", "force_run": "False", "day": "2014-06-22"}	PENDING	1
PeriodicEndMarkerCommand { "service_id": "1", "force_run": "False", "day": "2014-06-22"}	PENDING	1
AllDailyTask { "service_id": "1", "force_run": "False", "day": "2014-06-22", "tz_string": "-0700"}	PENDING	1
HistoricalAccountIndexTask { "service_id": "1", "force_run": "False", "day": "2014-06-22"}	PENDING	1
AccountSecondIndexDataLoaderTask { "service_id": "1", "force_run": "False", "day": "2014-06-22"}	PENDING	1
HealthCalculationTask { "service_id": "1", "force_run": "False", "day": "2014-06-22"}	PENDING	1
AccountIndexDataLoaderTask	PENDING	1

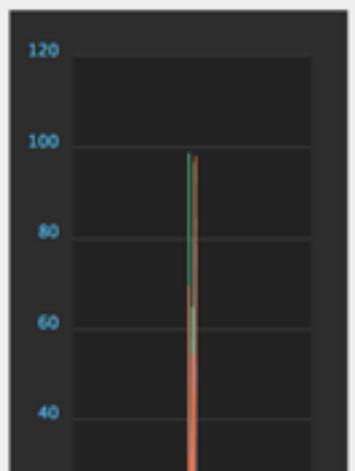
Gameboy Status

200	admin_db
200	history_db
200	jenkins
200	luigi
200	maint_db
200	status

Jenkins Status

enabled	daily-cron
enabled	daily-manual
enabled	hourly-cron

Luigi Workers CPU Utilization



GAMEBOY

Gameboy

Dashboard Daily All Dailies 2014-06-23 2014-06-22 2014-06-21 2014-06-20 2014-06-19 2014-06-18 2014-06-17 Hourly Services Jenkins

Home / Daily / Daily on 2014-06-21

DAILY FOR 2014-06-21

ALL JOBS

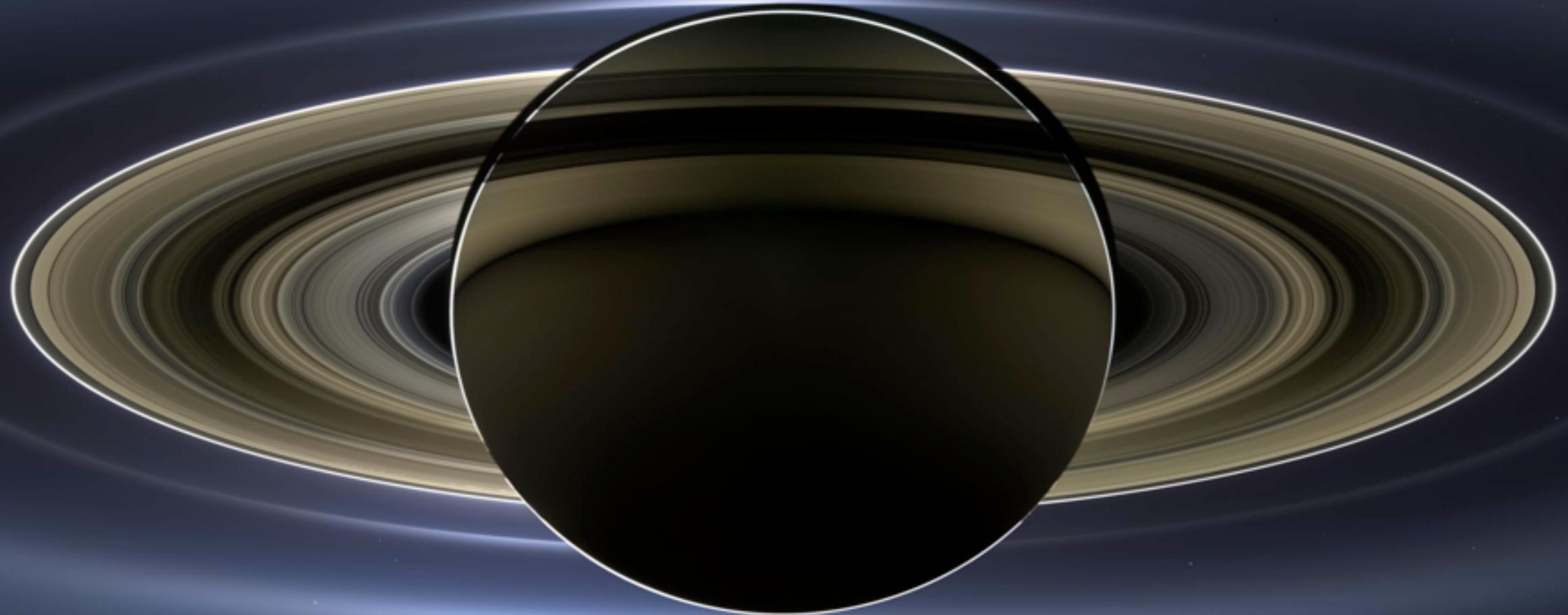
Service	Start	Last update	Duration	Status
	22nd - 07:01:56	22nd - 09:23:13	2 hours	DONE
	22nd - 07:02:05	22nd - 10:56:59	4 hours	DONE
	22nd - 07:01:41	22nd - 12:56:49	6 hours	DONE
	22nd - 12:31:40	22nd - 02:33:12	2 hours	DONE
	22nd - 04:01:45	22nd - 04:42:14	40 minutes	DONE
	22nd - 07:01:56	22nd - 09:19:24	2 hours	DONE
	22nd - 07:02:03	22nd - 10:06:33	3 hours	DONE
	22nd - 12:31:39	22nd - 01:14:05	42 minutes	DONE
	22nd - 07:01:54	22nd - 08:41:19	2 hours	DONE
	21st - 09:31:46	21st - 10:23:40	an hour	DONE
	21st - 09:31:44	21st - 10:25:55	an hour	DONE
	22nd - 04:01:42	22nd - 04:55:17	an hour	DONE
	21st - 10:31:40	21st - 11:35:10	an hour	DONE
	22nd - 07:02:01	22nd - 09:40:05	3 hours	DONE
	22nd - 06:01:41	22nd - 02:48:28	9 hours	DONE

GAMEBOY IS

- A Totango specific controller for Luigi
 - The transition process (to Luigi)
 - Provide **high level** overview
 - Manual **re-run** of tasks
 - Monitor **progress, performance, run times, queues, worker load** etc...
 - Implemented using **Flask** and **AngularJS**



WHAT ELSE IS OUT THERE?



WHAT ELSE IS OUT THERE?

- Oozie
- Azkaban
- AWS Data Pipeline
- Chronos
- spring-batch
- Dataswarm (facebook)
- River (outbrain internal)
- What's your favorite WF engine? (did you build one?)

REFS

- <https://github.com/spotify/luigi>
- Facebook's Dataswarm <https://www.youtube.com/watch?v=M0VCbhfQ3HQ>
- Outbrain's River <https://www.youtube.com/watch?v=EzsckTggDiM>