CHRONOS

A DISTRIBUTED, FAULT-TOLERANT & HIGHLY AVAILABLE JOB ORCHESTRATION FRAMEWORK FOR MESOS

Florian Leibert | @flo | flo@mesosphe.re

MY BACKGROUND

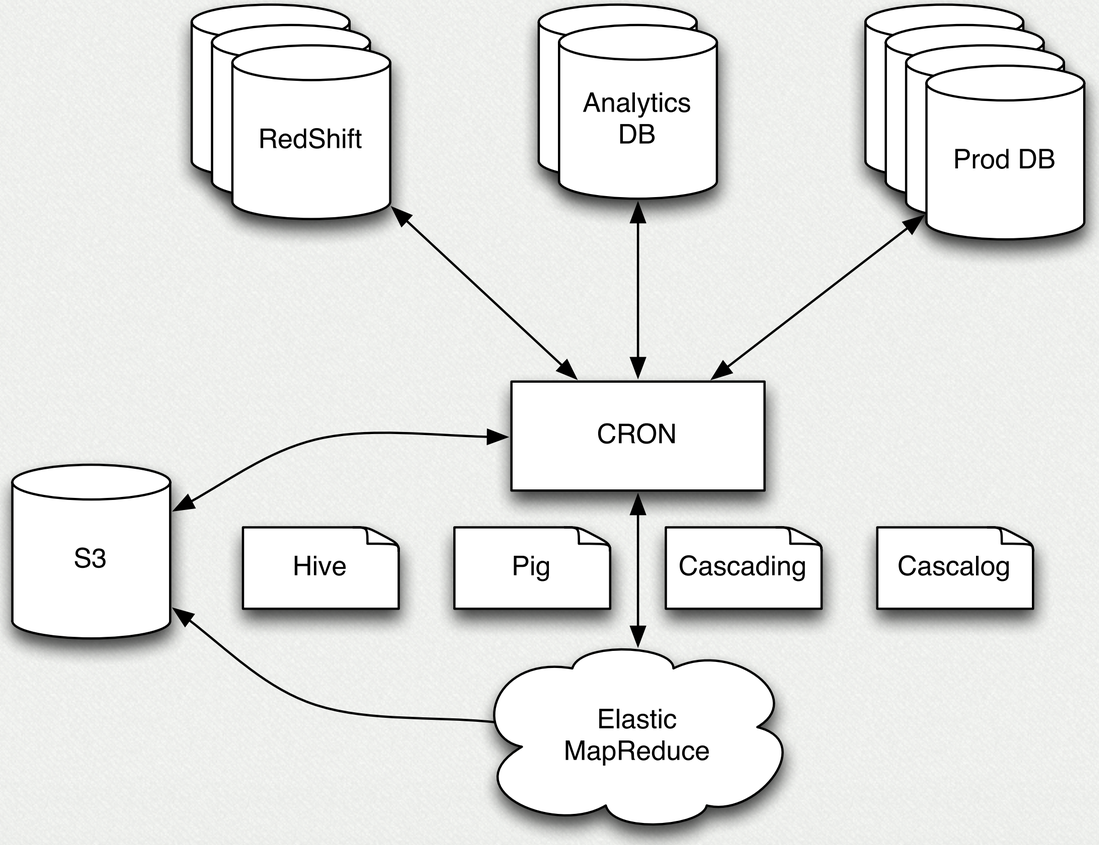
@FLO
— TWITTER, AIRBNB, MESOSPHERE —

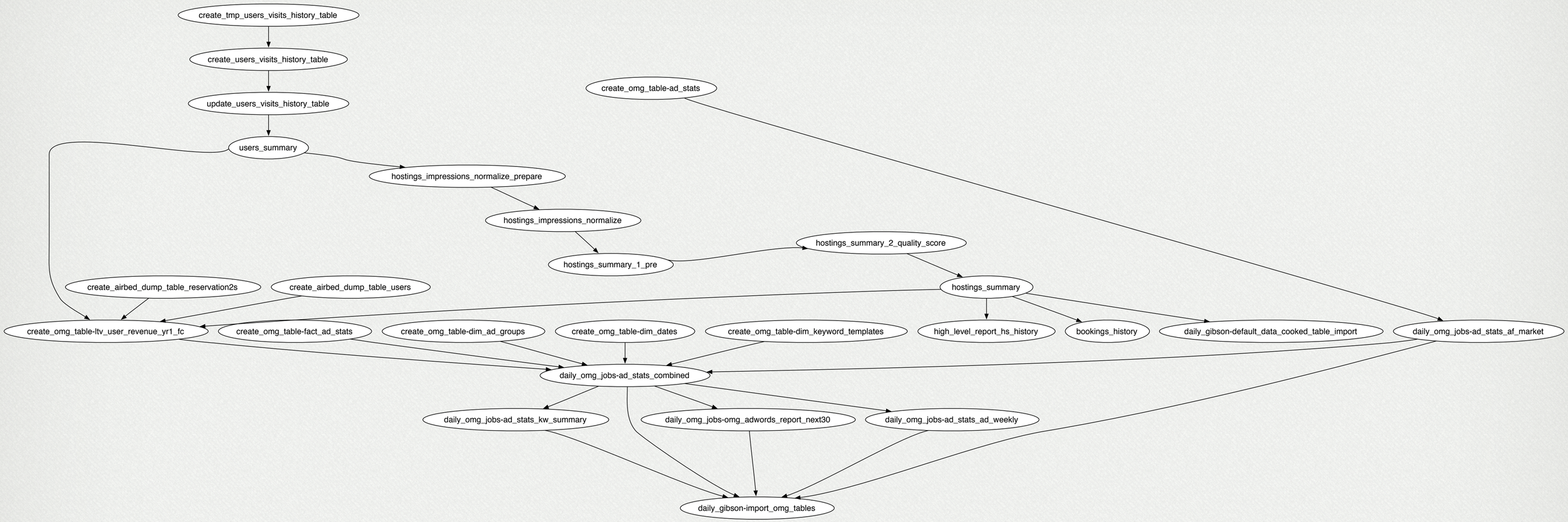
- Built user search, user store, and recommendation system at **Twitter**
- Lead data infrastructure at Airbnb, built logging system, ETL pipeline, Chronos

STRUCTURE

- Motivation
- Requirements
- Building Chronos
- Mesos Integration / Primitives
- Deploying & Running Chronos
- Marathon
- Q&A

DATA INFRASTRUCTURE @ AIRBNB (PRE CHRONOS)




create_tmp_users_visits_history_table
create_users_visits_history_table
update_users_visits_history_table
users_summary
hostings_impressions_normalize_prepare
hostings_impressions_normalize
hostings_summary_1_pre
hostings_summary_2_quality_score
hostings_summary
create_omg_table-ad_stats
create_airbed_dump_table_reservation2s
create_airbed_dump_table_users
create_omg_table-ltv_user_revenue_yr1_fc
create_omg_table-fact_ad_stats
create_omg_table-dim_ad_groups
create_omg_table-dim_dates
create_omg_table-dim_keyword_templates
high_level_report_hs_history
bookings_history
daily_gibson-default_data_cooked_table_import
daily_omg_jobs-ad_stats_af_market
daily_omg_jobs-ad_stats_combined
daily_omg_jobs-ad_stats_kw_summary
daily_omg_jobs-omg_adwords_report_next30
daily_omg_jobs-ad_stats_ad_weekly
daily_gibson-import_omg_tables

PROBLEMS

- 10,000+ of lines of bash
- “sleep 300”
- AWS: unreliable network & APIs
- single cron-node, DB import / export (I/O heavy workloads)
- debugging is really hard (pre-chronos, spent > 12hrs / week debugging jobs)

DATA INFRASTRUCTURE @ AIRBNB (PRE CHRONOS)

```
50 23 * * *  ubuntu      sudo find /tmp/* -name "combined*-*Z"  -mtime +7 -exec rm {} \;
59 23 * * *  anonymouse  cd /srv/emr && . ./aws.profile && ./pig/drivers/cronjobs_prepare_data.sh 2>&1 | logger -i -p user.info -t datachef
13 01 * * *  anonymouse  cd /srv/emr && . ./aws.profile && ./pig/drivers/cronjobs_search.sh 2>&1 | logger -i -p user.info -t datachef
00 23 * * *  anonymouse  cd /srv/emr && . ./aws.profile && ./pig/drivers/cronjobs_misc.sh 2>&1 | logger -i -p user.info -t datachef
00 01 * * *  anonymouse  cd /srv/emr && . ./aws.profile && ./pig/drivers/join_search_request.sh  2>&1 | logger -i -p user.info -t datachef

30 08 * * *  anonymouse  cd /srv/emr && . ./aws.profile && ./pig/drivers/adexploder/google_adword_event_gclid.sh 2>&1 | logger -i -p user.info -t datachef
45 08 * * *  anonymouse  cd /srv/emr && . ./aws.profile && ./pig/drivers/affiliate_events/affiliate_events_import.sh  2>&1 | logger -i -p user.info -t datachef

#00 07 * * *  anonymouse  cd /srv/emr && . ./aws.profile && ./hive/crons daily_jobs 2>&1 | logger -i -p user.info -t datachef_hive_crons
00 14 * * *  anonymouse  cd /srv/emr && . ./aws.profile && bash -x ./import_mysql/import_tables.bash import_production_tables 2>&1 | logger -i -p user.info -t datachef
00 17 * * *  anonymouse  cd /srv/emr && . ./aws.profile && bash -x ./import_mysql/import_tables.bash import_cooked_tables 2>&1 | logger -i -p user.info -t datachef_imp

00 12 * * *  anonymouse  cd /srv/emr && . ./aws.profile && ./pig/drivers/cronjobs_prepare_data_omg.sh 2>&1 | logger -i -p user.info -t datachef
00 13 * * *  anonymouse  cd /srv/emr && . ./aws.profile && ./hive/crons daily_omg_jobs 2>&1 | logger -i -p user.info -t datachef
00 18 * * *  anonymouse  cd /srv/emr && . ./aws.profile && bash -x ./import_mysql/import_tables.bash import_default_omg_tables_without_schema_from_data_cooked 2>&1 | l
00 18 * * *  anonymouse  cd /srv/emr && . ./aws.profile && ruby ./import_mysql/import_open_graph_metrics_to_river.rb 2>&1 | logger -i -p user.info -t datachef_import_c

00 16 * * *  anonymouse  cd /srv/emr && . ./aws.profile && ./hive/crons daily_history_jobs 2>&1 | logger -i -p user.info -t datachef_hive_crons
00 17 * * *  anonymouse  cd /srv/emr && . ./aws.profile && ./hive/crons roambi_data_complete 2>&1 | logger -i -p user.info -t datachef_hive_crons

00 03 * * 0  anonymouse  cd /srv/emr && . ./aws.profile && bash -x ./util/create_job_flow.bash production | logger -i -p user.info -t datachef_create_jobflow
```

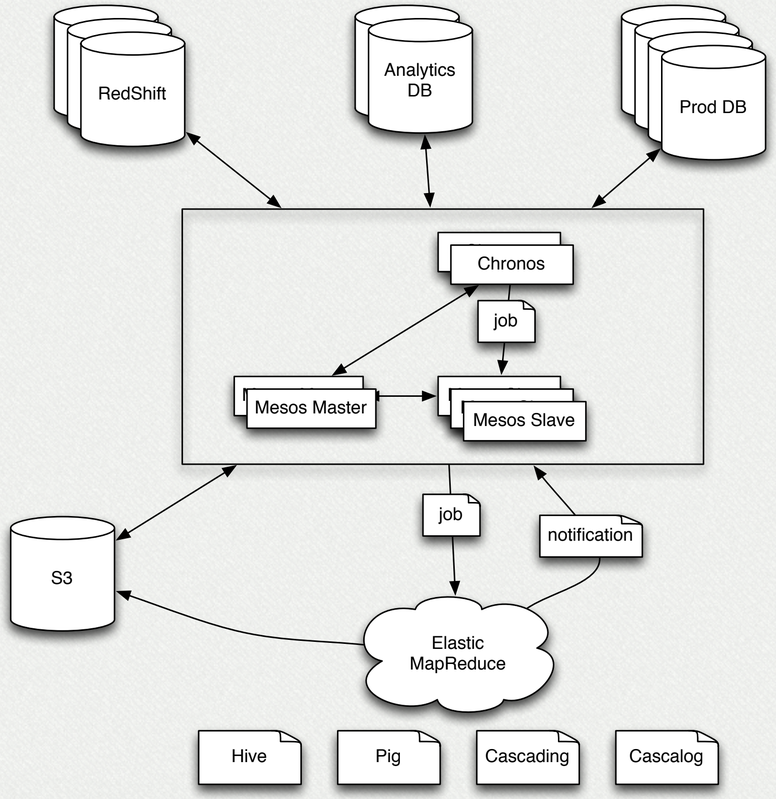
REQUIREMENTS

- existing tools too heavyweight
- express and visualize dependencies (easier debugging)
- retries / fault tolerance, HA
- distributed (many db export/import jobs use a lot of I/O)
- asynchronous jobs
- raw bash scheduler, schedule Hadoop & other (rake) jobs
- goal: start one command line in the cluster and ensure it finishes

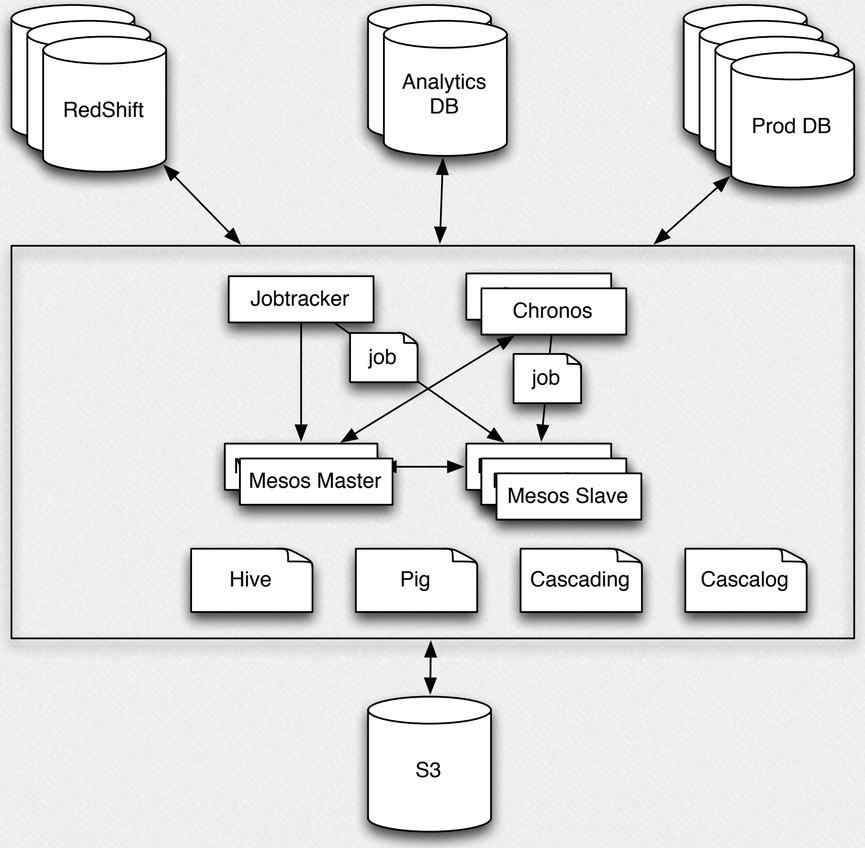
BUILDING CHRONOS

- 3K lines of scala
- iso8601 expressions (R10/2013-02-28T14:00:00Z/P1D)
- REST API
- HA
- Mesos as the kernel / Chronos leverages primitives
- Distributed / elastic
- No network programming!

DATA INFRASTRUCTURE @ AIRBNB (POST CHRONOS)



DATA INFRASTRUCTURE @ AIRBNB (HADOOP ON MESOS)



MESOS INTEGRATION: API

```
class ExampleScheduler extends Scheduler {

  def registered(driver: SchedulerDriver, frameworkId: FrameworkID, info: MasterInfo) {}

  def reregistered(driver: SchedulerDriver, info: MasterInfo) {}

  def resourceOffers(driver: SchedulerDriver, offers: util.List[Offer]) {}

  def offerRescinded(driver: SchedulerDriver, id: OfferID) {}

  def statusUpdate(driver: SchedulerDriver, status: TaskStatus) {}

  def frameworkMessage(driver: SchedulerDriver, executor: ExecutorID, slave: SlaveID, data: Array[Byte]) {}

  def disconnected(driver: SchedulerDriver) {}

  def slaveLost(driver: SchedulerDriver, slave: SlaveID) {}

  def executorLost(driver: SchedulerDriver, executor: ExecutorID, slave: SlaveID, status: Int) {}

  def error(driver: SchedulerDriver, error: String) {}
}
```

MESOS INTEGRATION: API

```
class ExampleScheduler extends Scheduler {

  def registered(driver: SchedulerDriver, frameworkId: FrameworkID, info: MasterInfo) {}

  def reregistered(driver: SchedulerDriver, info: MasterInfo) {}

  def resourceOffers(driver: SchedulerDriver, offers: util.List[Offer]) {}

  def offerRescinded(driver: SchedulerDriver, id: OfferID) {}

  def statusUpdate(driver: SchedulerDriver, status: TaskStatus) {}

  def frameworkMessage(driver: SchedulerDriver, executor: ExecutorID, slave: SlaveID, data: Array[Byte]) {}

  def disconnected(driver: SchedulerDriver) {}

  def slaveLost(driver: SchedulerDriver, slave: SlaveID) {}

  def executorLost(driver: SchedulerDriver, executor: ExecutorID, slave: SlaveID, status: Int) {}

  def error(driver: SchedulerDriver, error: String) {}
}
```

Learn more: "https://github.com/florianleibert/mesos-getting-started" (forked from Tobi Knaup's mesos-tutorial)

MESOS INTEGRATION: API

```
def resourceOffers(schedulerDriver: SchedulerDriver, offers: java.util.List[Offer]) {
  log.info("Received resource offers\n")
  import scala.collection.JavaConverters._
  def getNextTask(offers: List[Offer]) {
    taskManager.getTask match {
      case Some((x, j)) => {
        (offers.toIterator.map {
          offer => buildTask(x, j, offer) }.find(_._1)) match {
          case Some((sufficient, taskBuilder, offer)) =>
            processTask(x, j, offer, taskBuilder)
            getNextTask(offers.filter(x => x.getId != offer.getId))
          case _ =>
            log.warning("No sufficient offers found for task '%s', will append to queue".format(x))
            offers.foreach ( offer => mesosDriver.get().declineOffer(offer.getId) )

            /* Put the task back into the queue */
            taskManager.enqueue(x)
        }
      }
      case _ => {
        log.info("No tasks scheduled! Declining offers")
        offers.foreach ( offer => mesosDriver.get().declineOffer(offer.getId) )
      }
    }
  }
  getNextTask(offers.asScala.toList)
}
```

MESOS PRIMITIVES: STATE VARIABLE

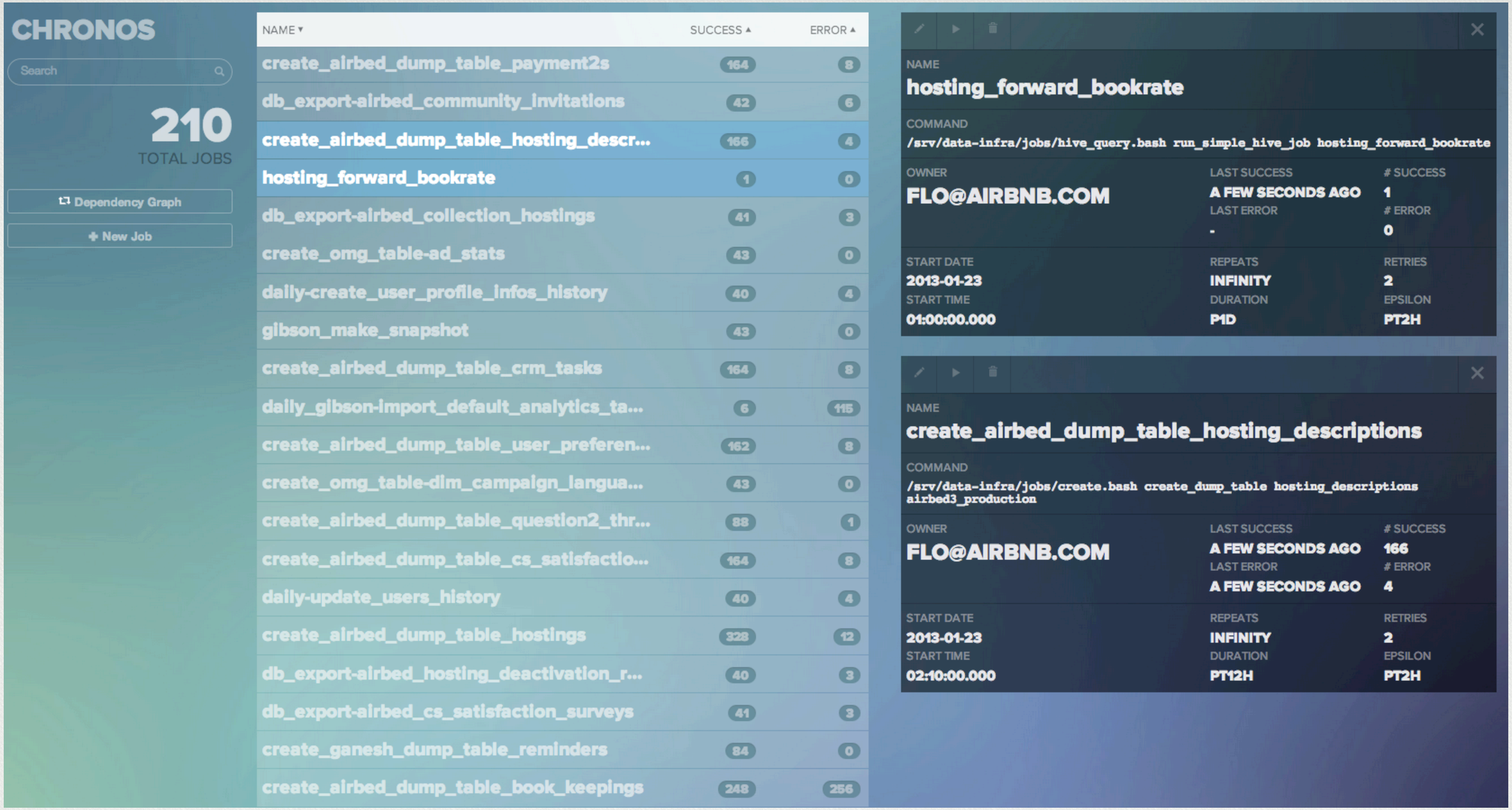
```
public class Variable {
  private long __variable;

  protected Variable() { /* compiled code */ }

  public native byte[] value();

  public native org.apache.mesos.state.Variable mutate(byte[] bytes);

  protected native void finalize();
}
```

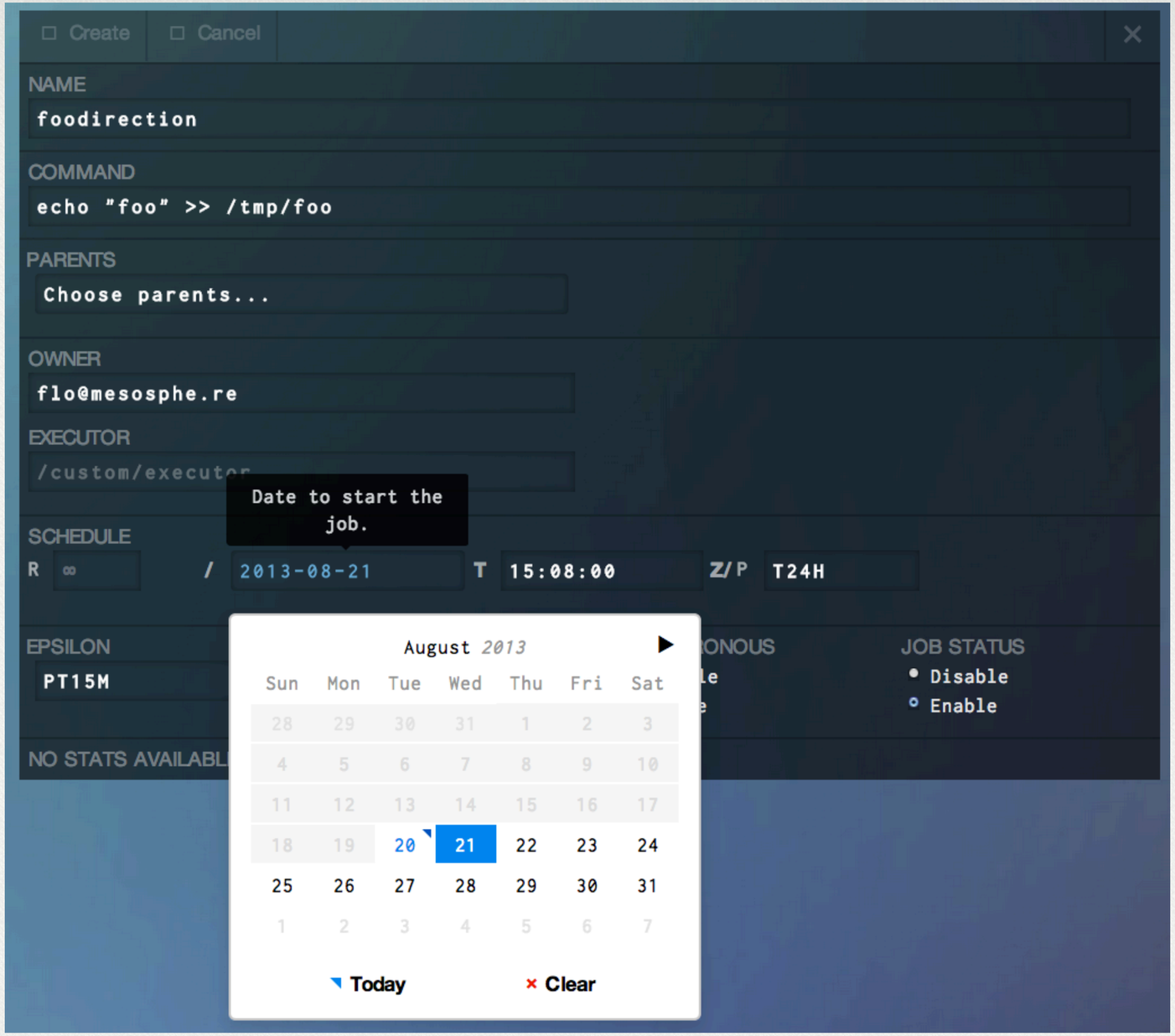
MESOS PRIMITIVES: STATE

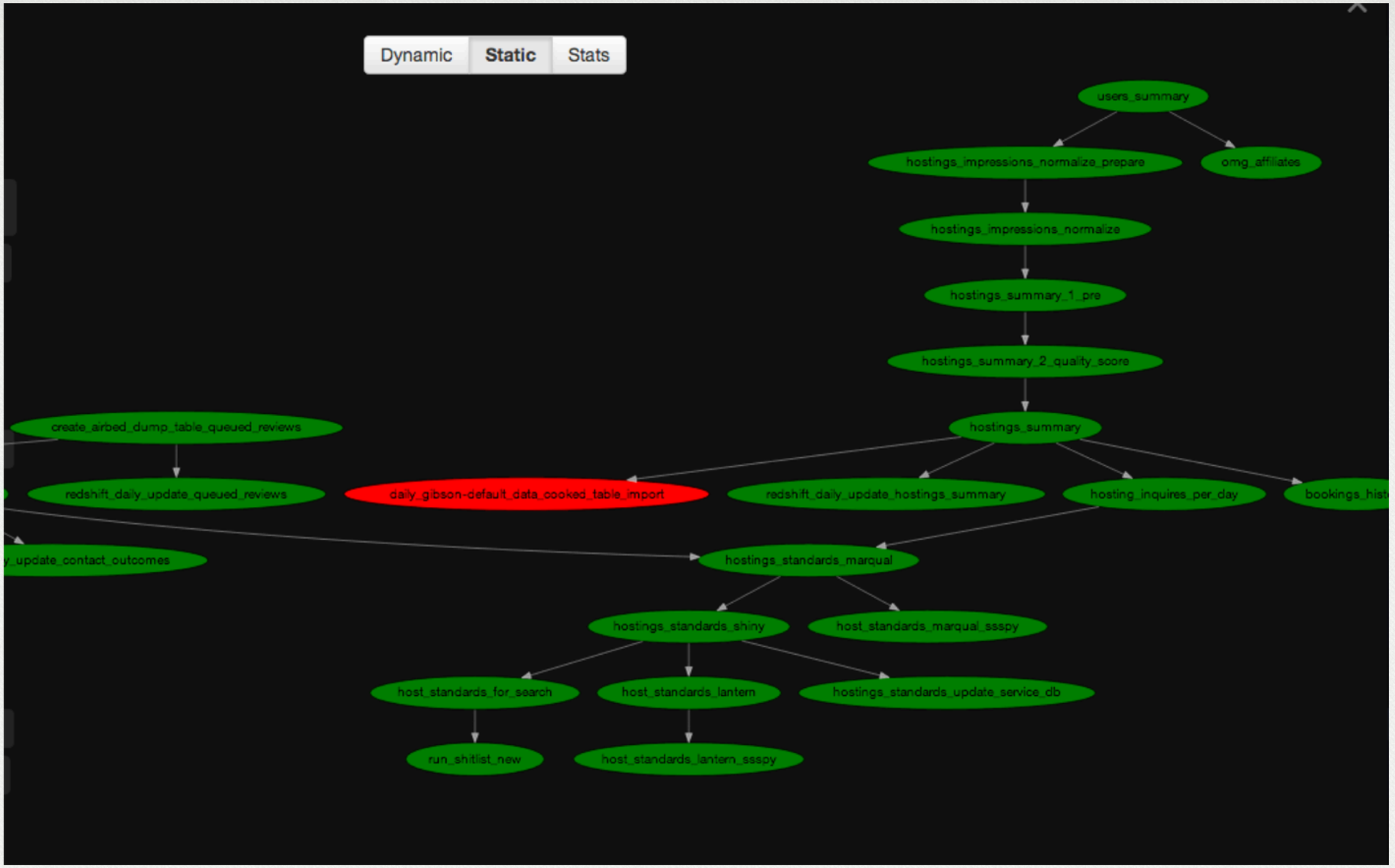
```
public interface State {
  java.util.concurrent.Future<org.apache.mesos.state.Variable> fetch(java.lang.String s);

  java.util.concurrent.Future<org.apache.mesos.state.Variable> store(org.apache.mesos.state.Variable variable);

  java.util.concurrent.Future<java.lang.Boolean> expunge(org.apache.mesos.state.Variable variable);

  java.util.concurrent.Future<java.util.Iterator<java.lang.String>> names();
}
```

MESOS PRIMITIVES: USING STATE

```
def getJobs: Iterator[BaseJob] = {
¬
  import scala.collection.JavaConversions._¬
¬
  state.names.get.filter(_.startsWith(jobPrefix))¬
    .map({¬
    x: String => JobUtils.fromBytes(state.fetch(x).get.value)¬
  })¬
}
```


CHRONOS
Search
210
TOTAL JOBS
Dependency Graph
New Job
NAME
SUCCESS
ERROR
create_airbed_dump_table_payment2s 164 8
db_export-airbed_community_invitations 42 6
create_airbed_dump_table_hosting_descr... 166 4
hosting_forward_bookrate 1 0
db_export-airbed_collection_hostings 41 3
create_omg_table-ad_stats 43 0
daily-create_user_profile_infos_history 40 4
gibson_make_snapshot 43 0
create_airbed_dump_table_crm_tasks 164 8
daily_gibson-import_default_analytics_ta... 6 115
create_airbed_dump_table_user_preferen... 162 8
create_omg_table-dim_campaign_langua... 43 0
create_airbed_dump_table_question2_thr... 88 1
create_airbed_dump_table_cs_satisfactio... 164 8
daily-update_users_history 40 4
create_airbed_dump_table_hostings 328 12
db_export-airbed_hosting_deactivation_r... 40 3
db_export-airbed_cs_satisfaction_surveys 41 3
create_ganesh_dump_table_reminders 84 0
create_airbed_dump_table_book_keepings 248 256
NAME
hosting_forward_bookrate
COMMAND
/srv/data-infra/jobs/hive_query.bash run_simple_hive_job hosting_forward_bookrate
OWNER
FLO@AIRBNB.COM
LAST SUCCESS
A FEW SECONDS AGO
SUCCESS
1
LAST ERROR
-
ERROR
0
START DATE
2013-01-23
REPEATS
INFINITY
RETRIES
2
START TIME
01:00:00.000
DURATION
P1D
EPSILON
PT2H
NAME
create_airbed_dump_table_hosting_descriptions
COMMAND
/srv/data-infra/jobs/create.bash create_dump_table hosting_descriptions airbed3_production
OWNER
FLO@AIRBNB.COM
LAST SUCCESS
A FEW SECONDS AGO
SUCCESS
166
LAST ERROR
A FEW SECONDS AGO
ERROR
4
START DATE
2013-01-23
REPEATS
INFINITY
RETRIES
2
START TIME
02:10:00.000
DURATION
PT12H
EPSILON
PT2H


Create
Cancel
NAME
foodirection
COMMAND
echo "foo" >> /tmp/foo
PARENTS
Choose parents...
OWNER
flo@mesosphe.re
EXECUTOR
/custom/executor
Date to start the job.
SCHEDULE
R ∞ / 2013-08-21 T 15:08:00 Z/P T24H
EPSILON
PT15M
JOB STATUS
Disable
Enable
NO STATS AVAILABL
August 2013
Sun Mon Tue Wed Thu Fri Sat
28 29 30 31 1 2 3
4 5 6 7 8 9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
1 2 3 4 5 6 7
Today
Clear


Dynamic
Static
Stats
users_summary
hostings_impressions_normalize_prepare
omg_affiliates
hostings_impressions_normalize
hostings_summary_1_pre
hostings_summary_2_quality_score
hostings_summary
create_airbed_dump_table_queued_reviews
redshift_daily_update_queued_reviews
daily_gibson-default_data_cooked_table_import
redshift_daily_update_hostings_summary
hosting_inquires_per_day
bookings_hist
y_update_contact_outcomes
hostings_standards_marqual
hostings_standards_shiny
host_standards_marqual_ssspy
host_standards_for_search
host_standards_lantern
hostings_standards_update_service_db
run_shitlist_new
host_standards_lantern_ssspy

CHRONOS

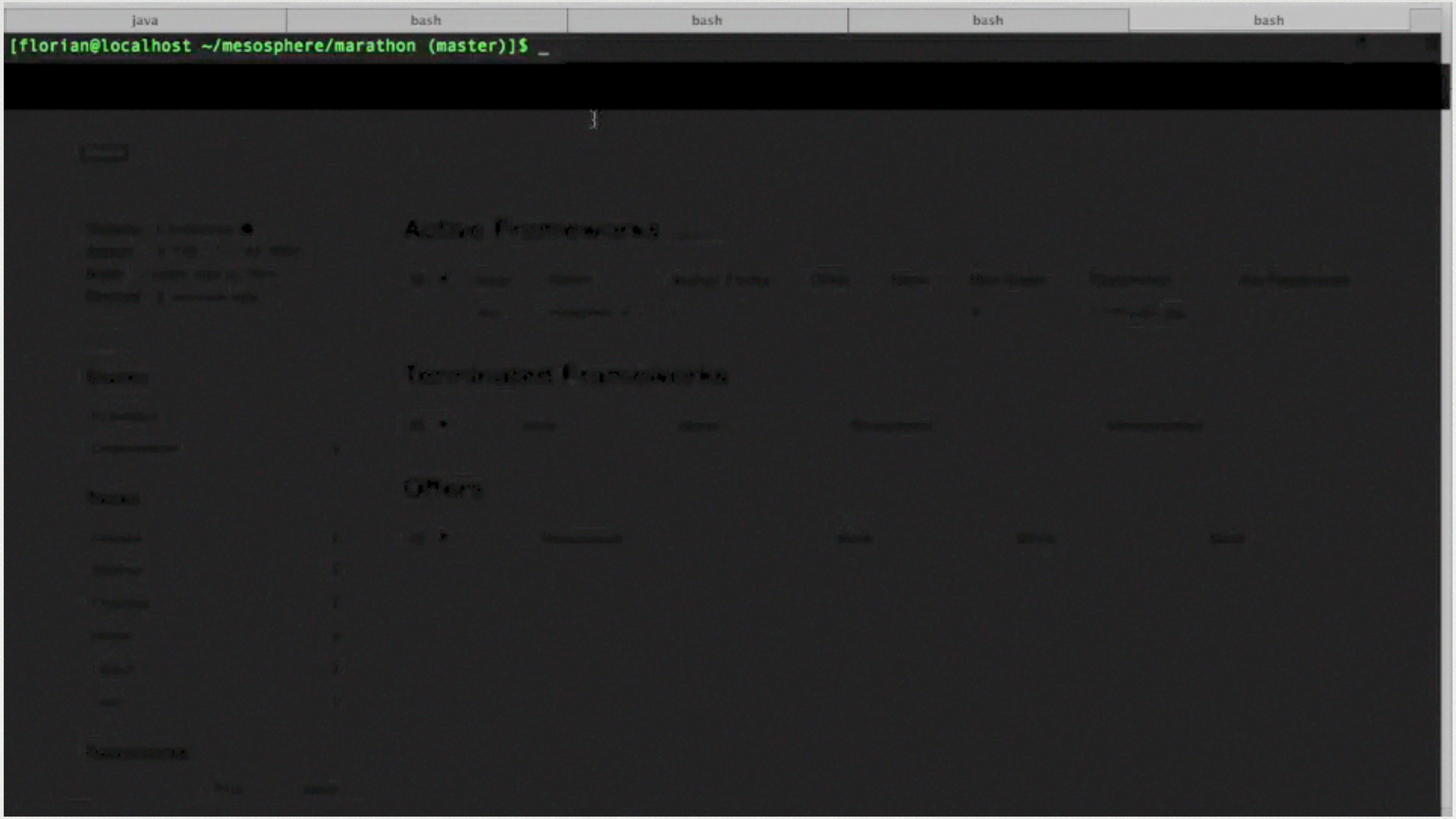
```
},
{
    "async": false,
    "command": "/srv/data-infra/jobs/hive_query.bash run_simple_hive_job ltv_user_revenue_yr1_fc",
    "epsilon": "PT15M",
    "errorCount": 0,
    "executor": "/srv/mesos/utils/async-executor.arx",
    "executorFlags": "",
    "lastError": "",
    "lastSuccess": "2013-02-27T21:29:40.159Z",
    "name": "create_omg_table-ltv_user_revenue_yr1_fc",
    "owner": "MATT@AIRBNB.COM",
    "parents": [
        "hostings_summary",
        "users_summary",
        "create_airbed_dump_table_reservation2s",
        "create_airbed_dump_table_users"
    ],
    "retries": 2,
    "successCount": 40
},
```

DEPLOY & RUN CHRONOS

- https://github.com/airbnb/chronos
- `mvn package`
- Local mode (In-process Zookeeper / no persistence!)
- ZooKeeper & Mesos required!
- Modify configuration (e.g. adjust ZK
- java -cp ./target/chronos-1.0.jar com.airbnb.scheduler.Main server ./config/sample_scheduler.yml

MARATHON

- Marathon is a framework for long running services (upstart for the DC)
- built on top of Mesos
- will be open-sourced soon (contact me if you want early access)
- minimal framework, supports elastic scaling / handles failures
- starts a command line n-times in your cluster & keeps it running!
- REST API & command line client
- HA
- start Chronos via Marathon?


java
bash
bash
bash
bash
[florian@localhost ~/mesosphere/marathon (master)]$ _

MARATHON/_

chronos

CMD: ./chronos/bin/demo ./chronos/config/nomail.yml ./chronos/target/chronos-1.0-SNAPSHOT.jar

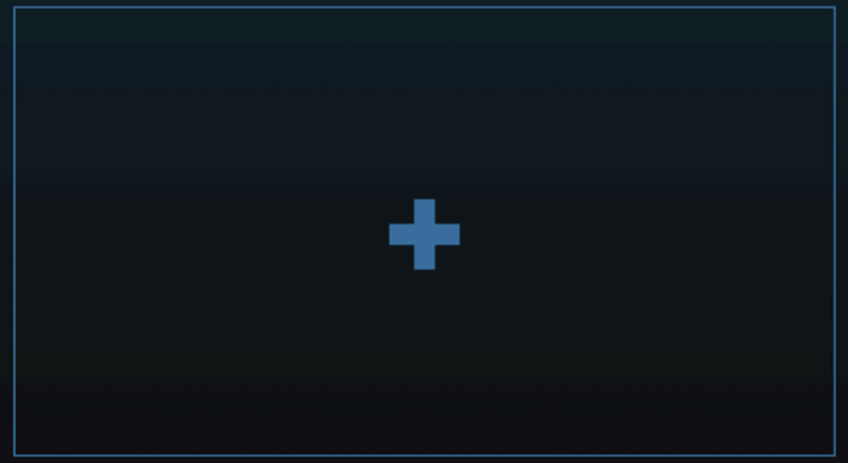
URIs: 1

Memory: 1024

CPU: 1

Instances: 1

SCALE | STOP



CHRONOS USERS

box

airbnb

vimeo

QUESTIONS?