
Google™
11
I/O

Putting Task Queues To Work

Vivek Sahasranaman and Nicholas Verne
May 10, 2011

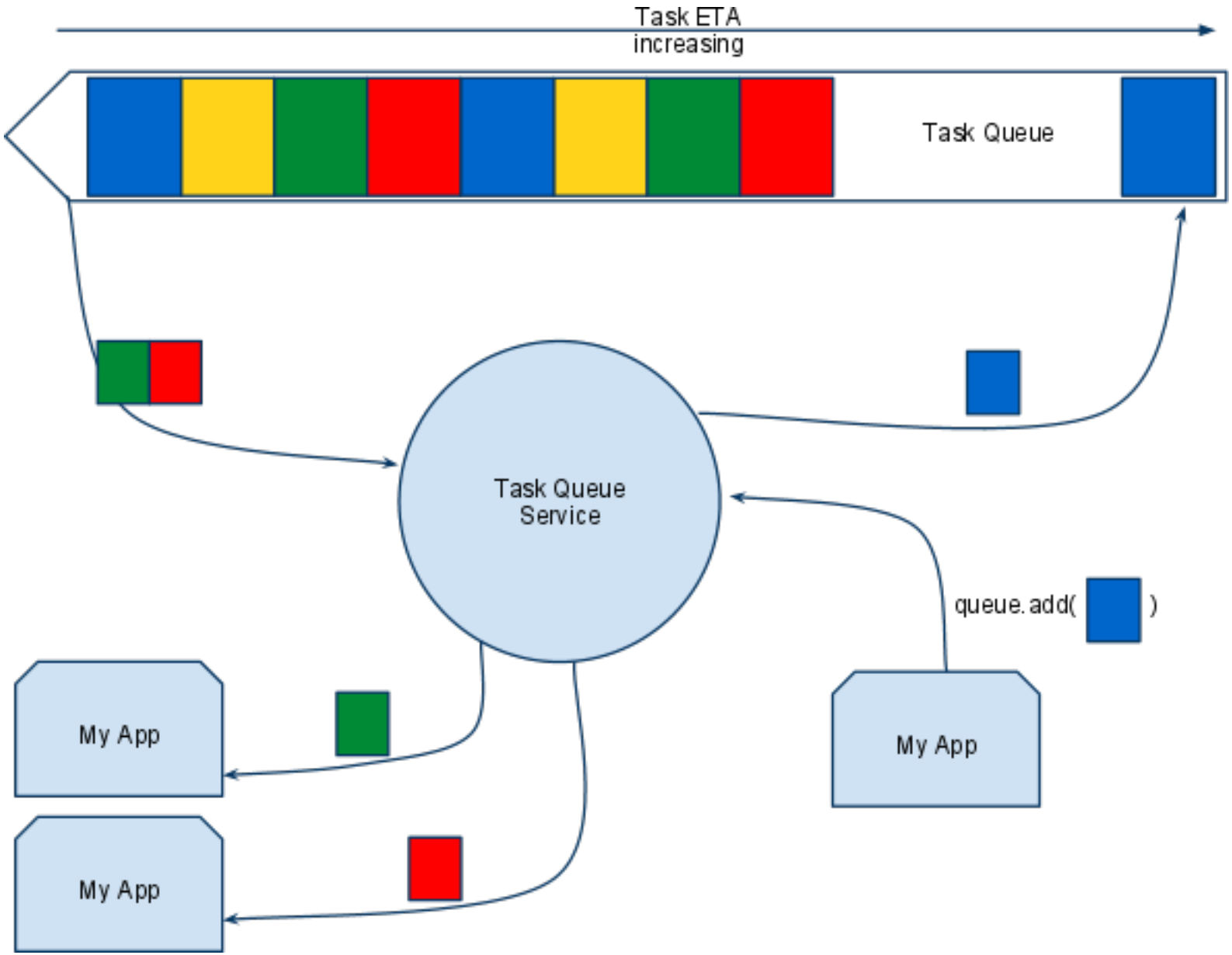
Feedback: **http://goo.gl/sAXZh**

Task Queues in Google App Engine

Some History

- App Engine - easy to write, manage, and scale web apps
- But ... user requests are limited to 30 seconds
- Task Queues to the rescue! - introduced as a Labs (experimental) feature in 2009
- Task Queues are good for computation that users need not wait for, e.g.:
  - datastore updates
  - urlfetch calls

Task Queues - Push



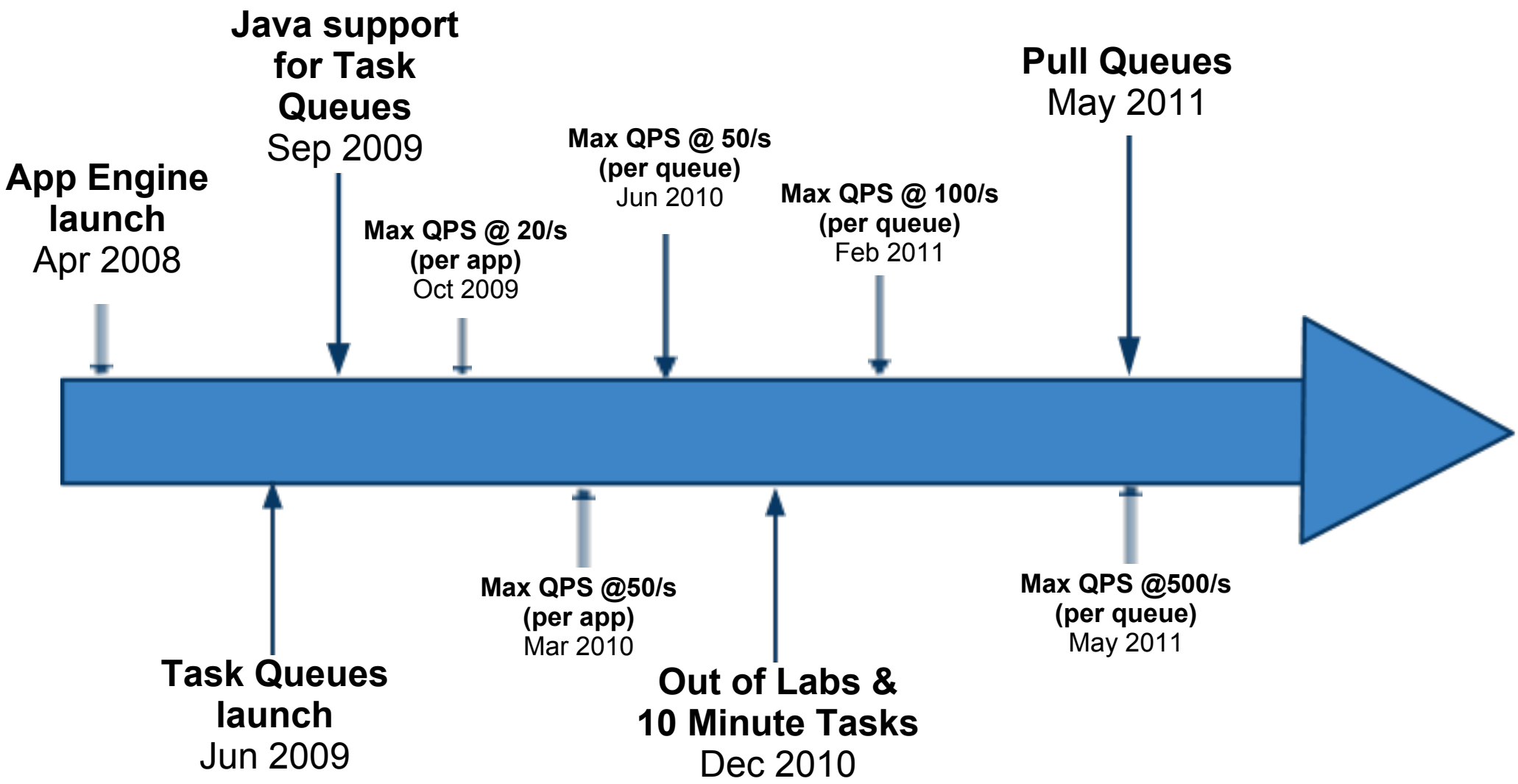
Task Queues in Google App Engine

Push Queues

- Developers define Queues using *queue.yaml* (Python) or *queue.xml* (Java)
  - Specify name, rate, etc
- Tasks are added to queues via API called from App Engine code.
  - *taskqueue.add(url='/foo')*
  - *q.add(TaskOptions.Builder.withUrl("foo"));*
- Task Queue pushes tasks to apps as HTTP requests.
  - Traffic from queue is limited to specified rate
  - App Engine scales instances depending on traffic
  - Task Queue system retries failed tasks

Task Queues in Google App Engine

Some More History



Introducing Pull Queues

Task Queues in Google App Engine

Pull Queues (New)

Basics

- Tasks are just data
- Workers lease tasks from the queue
- Workers delete tasks when processing is finished
- Developers need to control how many workers are available to process tasks

Worker May Crash!

- Task is not lost
- Some other worker can lease task after old lease expires

Crash and Crash Again!

- Queue's retry limit can prevent endless worker crashes

Pull Queues

New API Features (Python)

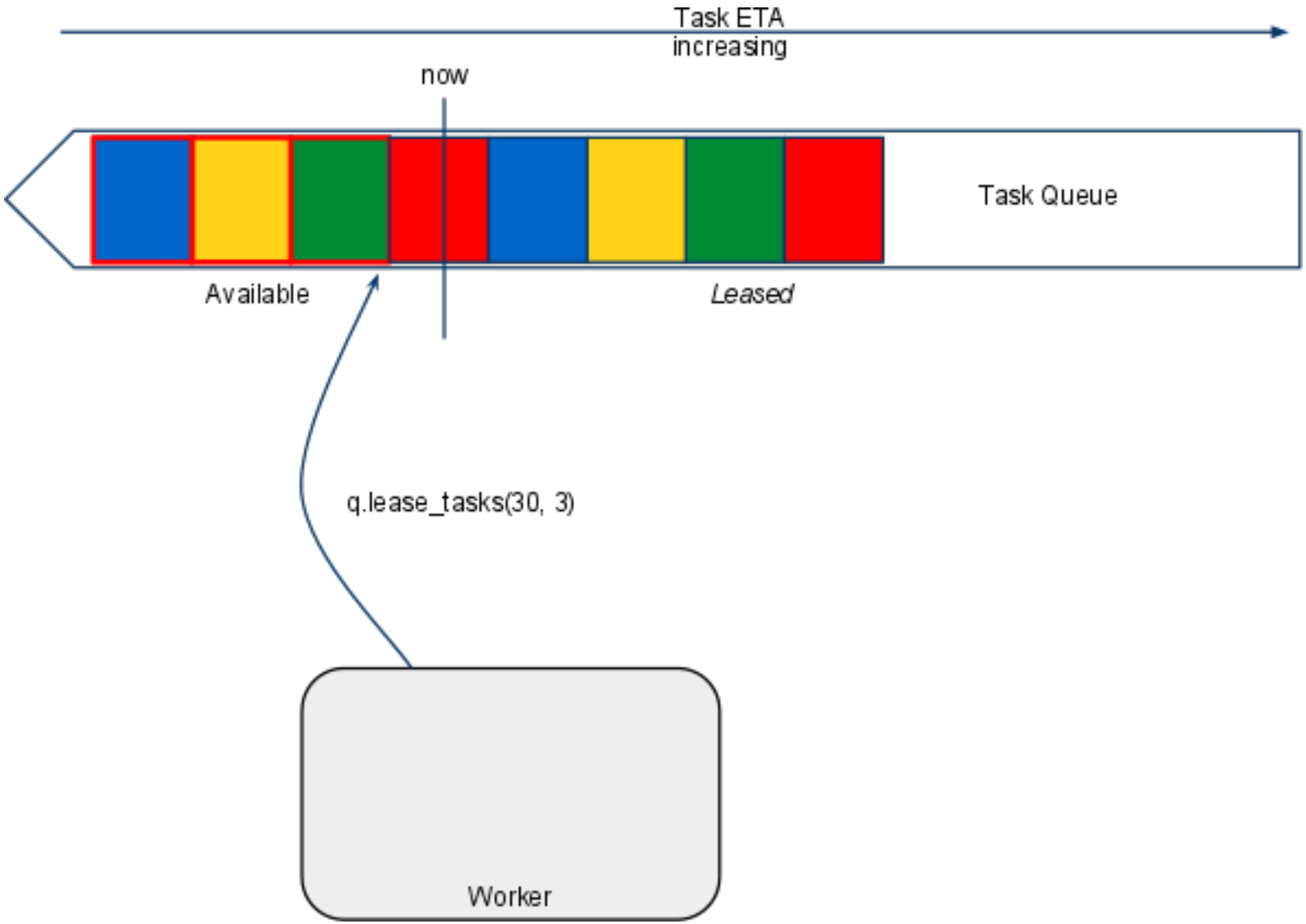
- Specify *mode: pull* in queue.yaml
- Add tasks with *method='pull'*
  - *q.add(taskqueue.Task(payload='hello', method='pull'))*
- Leasing tasks
  - *tasks = q.lease_tasks(30, 3)* # lease expires in 30 seconds,
    # maximum of 3 tasks returned
- Deleting tasks
  - *q.delete_tasks(tasks)*

Pull Queues

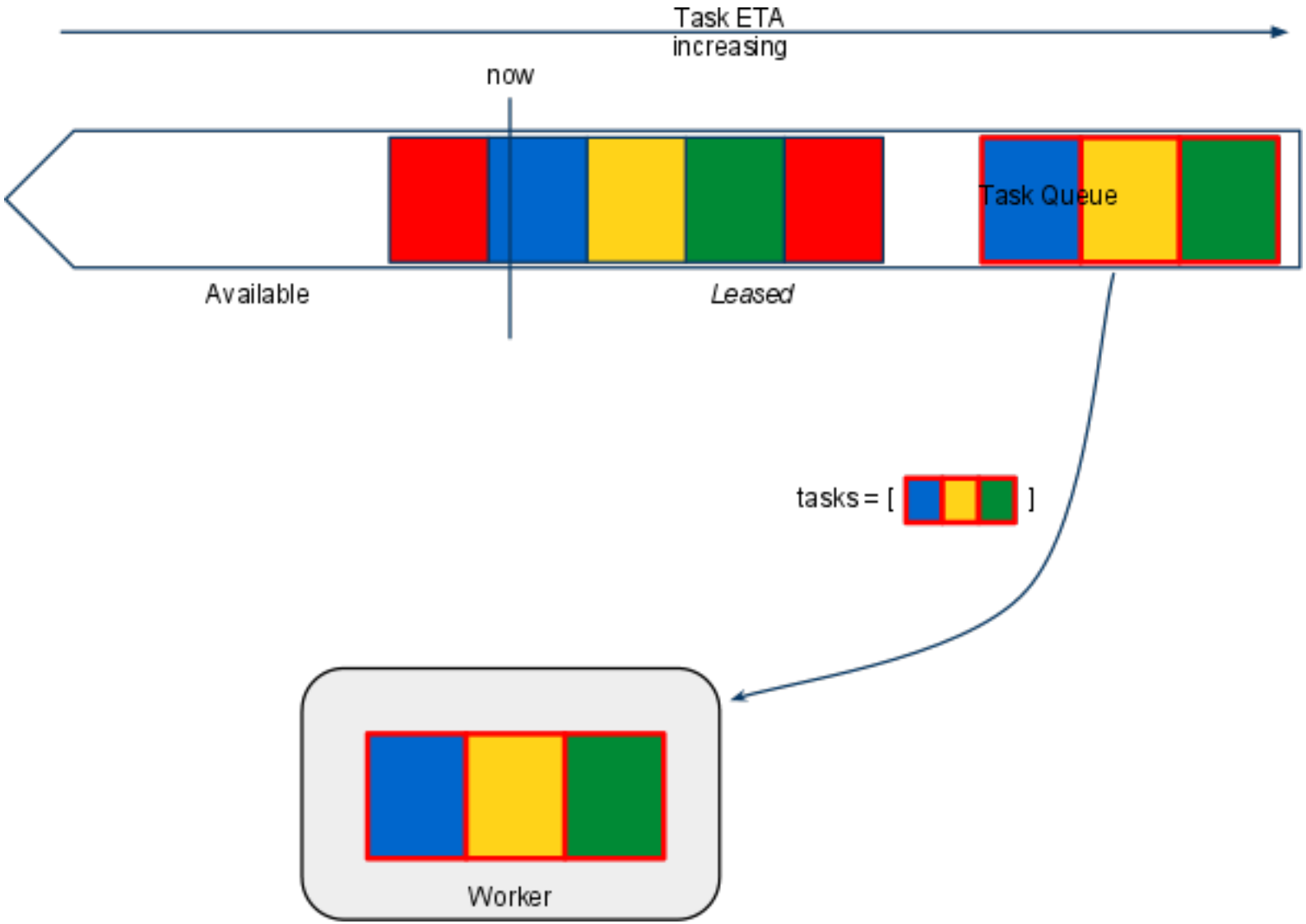
New API Features (Java)

- Specify *<mode>pull</mode>* in queue.xml
- Add tasks with *TaskOptions.Method.PULL*
  - *q.add(TaskOptions.Builder.withMethod(*
    TaskOptions.Method.PULL).payload('hello'));
- Leasing tasks
  - *List<TaskHandle> tasks =*
    q.leaseTasks(30, TimeUnit.SECONDS, 3);
- Deleting tasks
  - *for (TaskHandle t: tasks) {*
    q.deleteTask(t);
    }

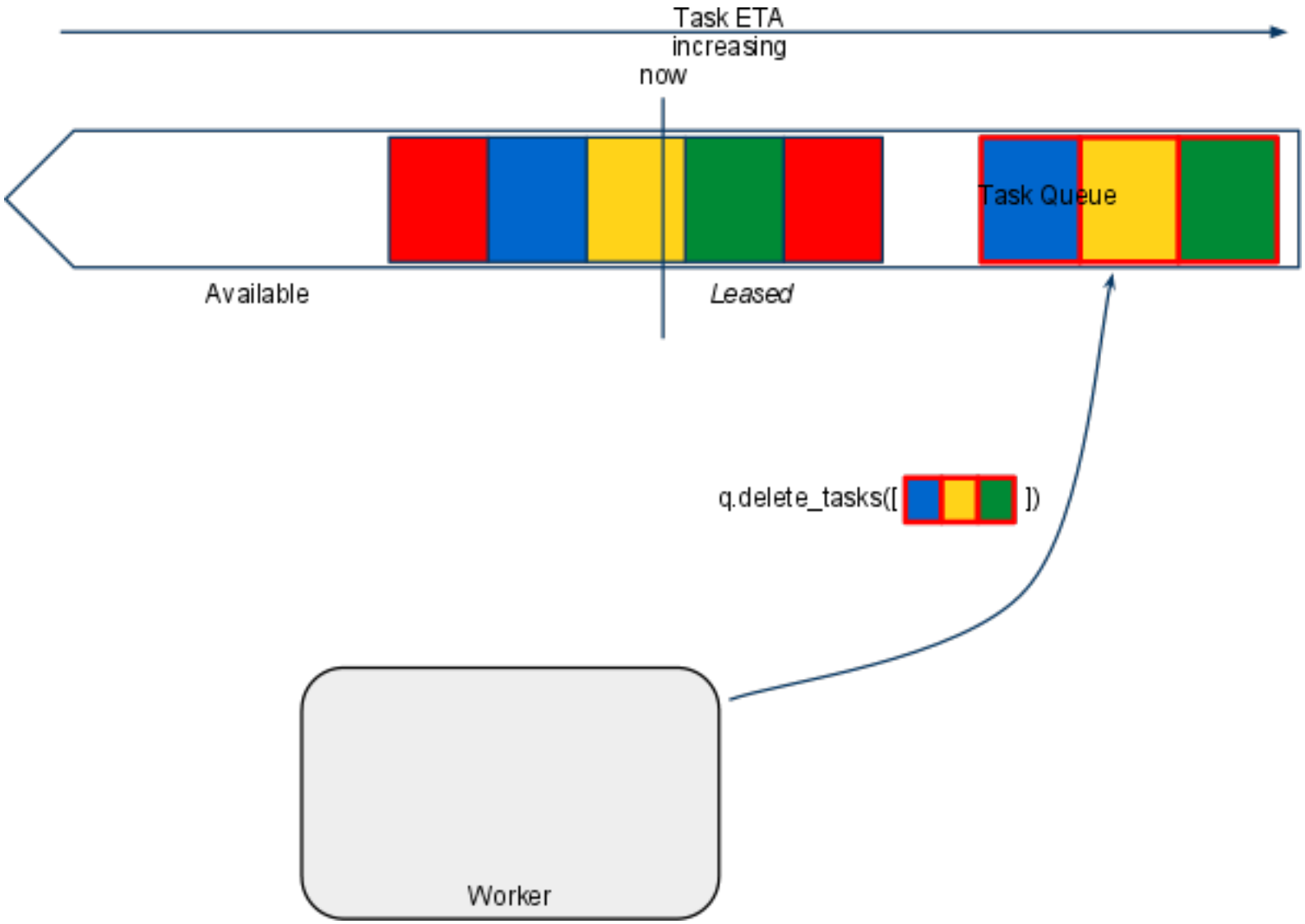
Leasing 3 tasks



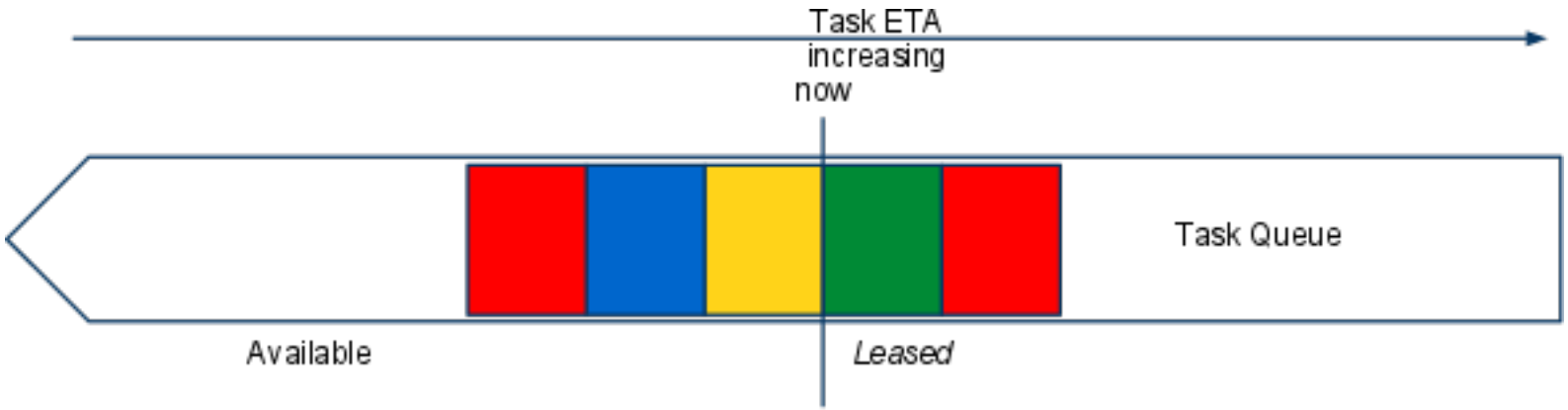
Leasing 3 tasks

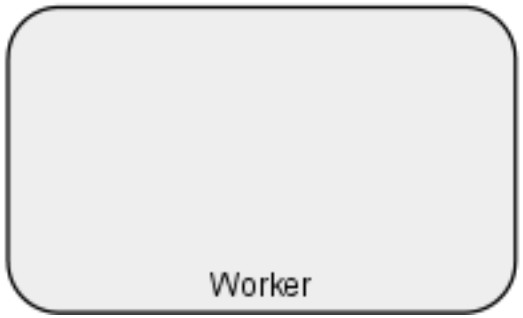


Leasing 3 tasks



Leasing 3 tasks





Example - Rough Tallying in Voterlator

Pull Queues in Action

- Tasks are votes
- Tallies to be stored in datastore
- Reduce Datastore traffic by accumulating many votes prior to writing results
- http://voterlator.appspot.com

Example - Rough Tallying in Voterlator

App and Queues

app.yaml

```
application: voterlator
version: 1
runtime: python
api_version: 1

handlers:
- url: .*
  script: main.py
```

queue.yaml

```
total_storage_limit: 1G

queue:
- name: votes
  mode: pull
```

Example - Rough Tallying in Voterlator

Handlers

```
application = webapp.WSGIApplication([
    ('/', VoteHandler),
    ('/tally', TallyHandler),
    ], debug=True)

def main():
  run_wsgi_app(application)


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

Example - Rough Tallying in Voterlator

Enqueueing Votes

```
class VoteHandler(webapp.RequestHandler):
"""Handles adding of vote tasks."""
def get(self):
  """Displays the voting form."""
 ...
def post(self):
   """Adds to the votes queue if ugliest is valid."""
   ugliest = self.request.get('ugliest')
   if ugliest and ugliest in LANGUAGES:
     q = taskqueue.Queue('votes')
     q.add(taskqueue.Task(payload=ugliest, method='PULL'))
   self.redirect('/')
```

Example - Rough Tallying in Voterlator

Configuring Workers

cron.yaml

```
cron:
- description: Vote tallying worker
  url: /tally
  schedule: every 1 minutes
```

Example - Rough Tallying in Voterlator

Tally Data - one tally for each language

```
class Tally(db.Model):
"""Simple counting model."""
count = db.IntegerProperty(default=0, required=True)

@classmethod
def increment_by(cls, key_name, count):
  """Increases a Tally's count. Should be run in a transaction."""
  tally = cls.get_by_key_name(key_name)
  if tally is None:
    tally = cls(key_name=key_name)
  tally.count += count
  tally.put()
```

Example - Rough Tallying in Voterlator

Storing Tallies

```
def store_tallies(self, tallies):
  """Updates the tallies in datastore."""
  for key_name, count in tallies.iteritems():
    db.run_in_transaction(Tally.increment_by, key_name, count)
```

Example - Rough Tallying in Voterlator

Tallying Votes

```
class TallyHandler(webapp.RequestHandler):
  def post(self):
    """Leases vote tasks, accumulates tallies and stores them."""
    q = taskqueue.Queue('votes')
    # Keep leasing tasks in a loop.
    while True:
      tasks = q.lease_tasks(300, 1000)
      if not tasks:
        return
      # accumulate tallies in memory
      tallies = {}
      for t in tasks:
        tallies[t.payload] = tallies.get(t.payload, 0) + 1
      self.store_tallies(tallies)
      q.delete_tasks(tasks)
```

Example - Rough Tallying in Voterlator

Advantages of this approach

- Less load on Datastore than if we used one transaction/vote
  - Potentially 1/1000th datastore puts

Introducing Task Queue REST API

Task Queue REST API

Workers Outside Google App Engine

- Workers can be anywhere on the Internet
  - Home machines, hosted machines, VMs
  - Allows developers to run custom binaries, image processing packages etc
- Queues and Tasks are Resources
- Workers can lease/delete tasks using REST verbs
- Auth:
  - Set ACLs on who can access API from *queue.yaml*
  - REST API uses OAuth, verifies ACLs

Task Queue REST API

Getting Started

- Specify acl in queue.yaml/queue.xml in AppEngine
- REST API available at URL:
  https://www.googleapis.com/taskqueue/myappid/taskqueues/myqueue
- Use the Google apiclient libraries to talk to the API
- Samples to get started from: Google API Samples
  - *gtaskqueue* : Run one command against the API and see result
  - *gtaskqueue_puller* : Continuously lease tasks from the queue, execute a binary and delete task

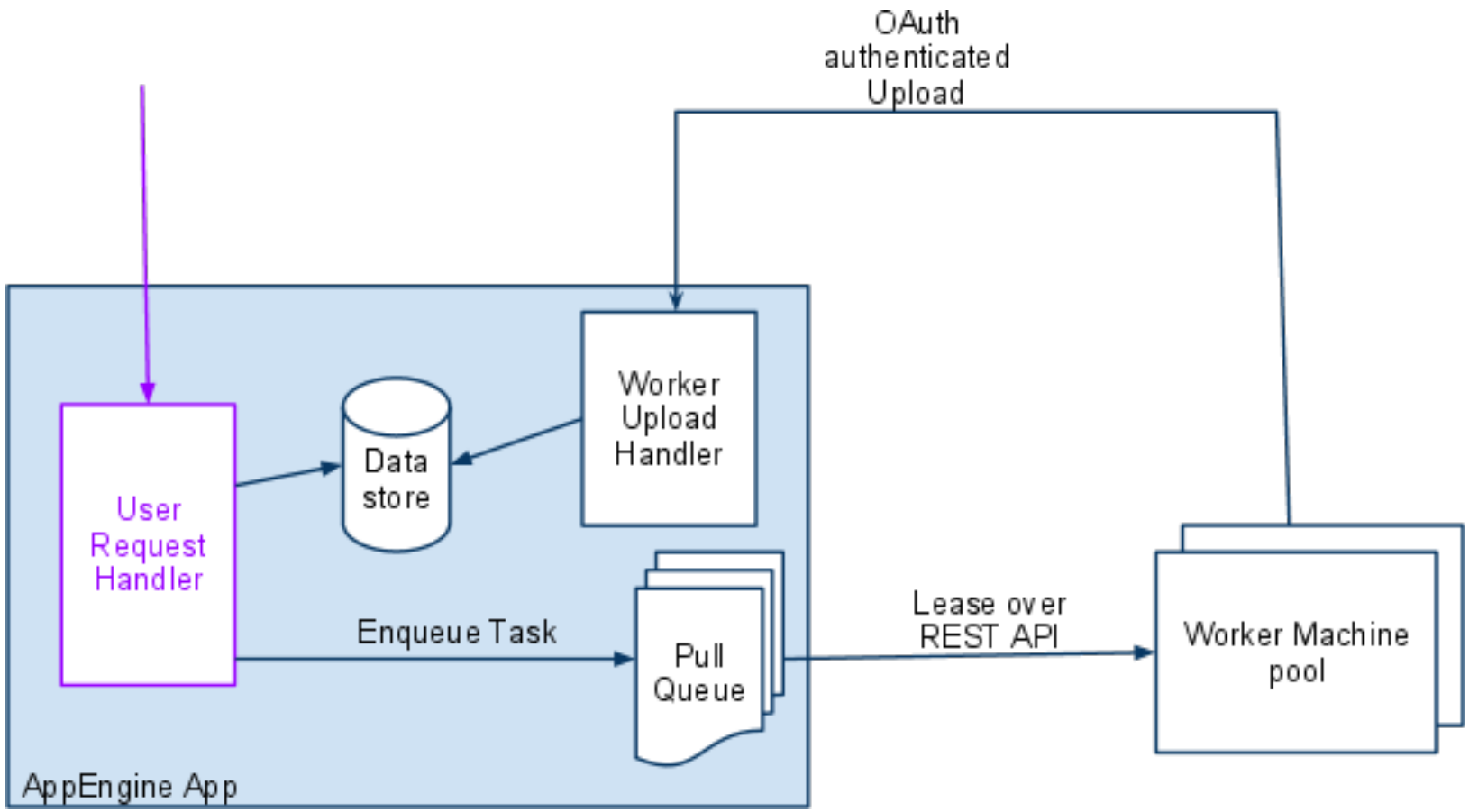
Example - Image Converter

Working Outside Google App Engine

- Task contains one photo in the payload
  - Added to App Engine queues
- Workers running in VMs modify the photo
- Results posted to a special handler on the appengine app

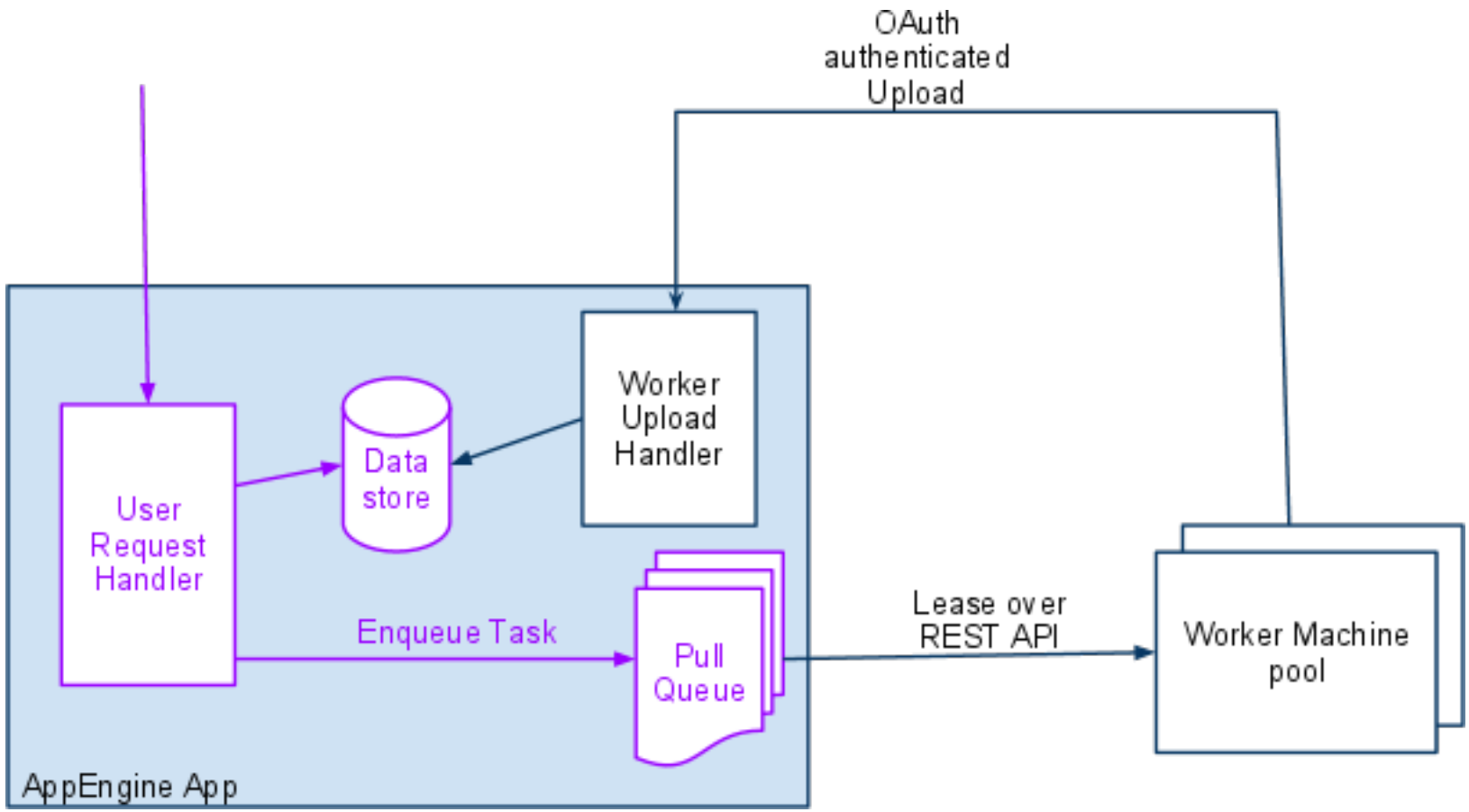
Example - Image Converter

Working Outside Google App Engine



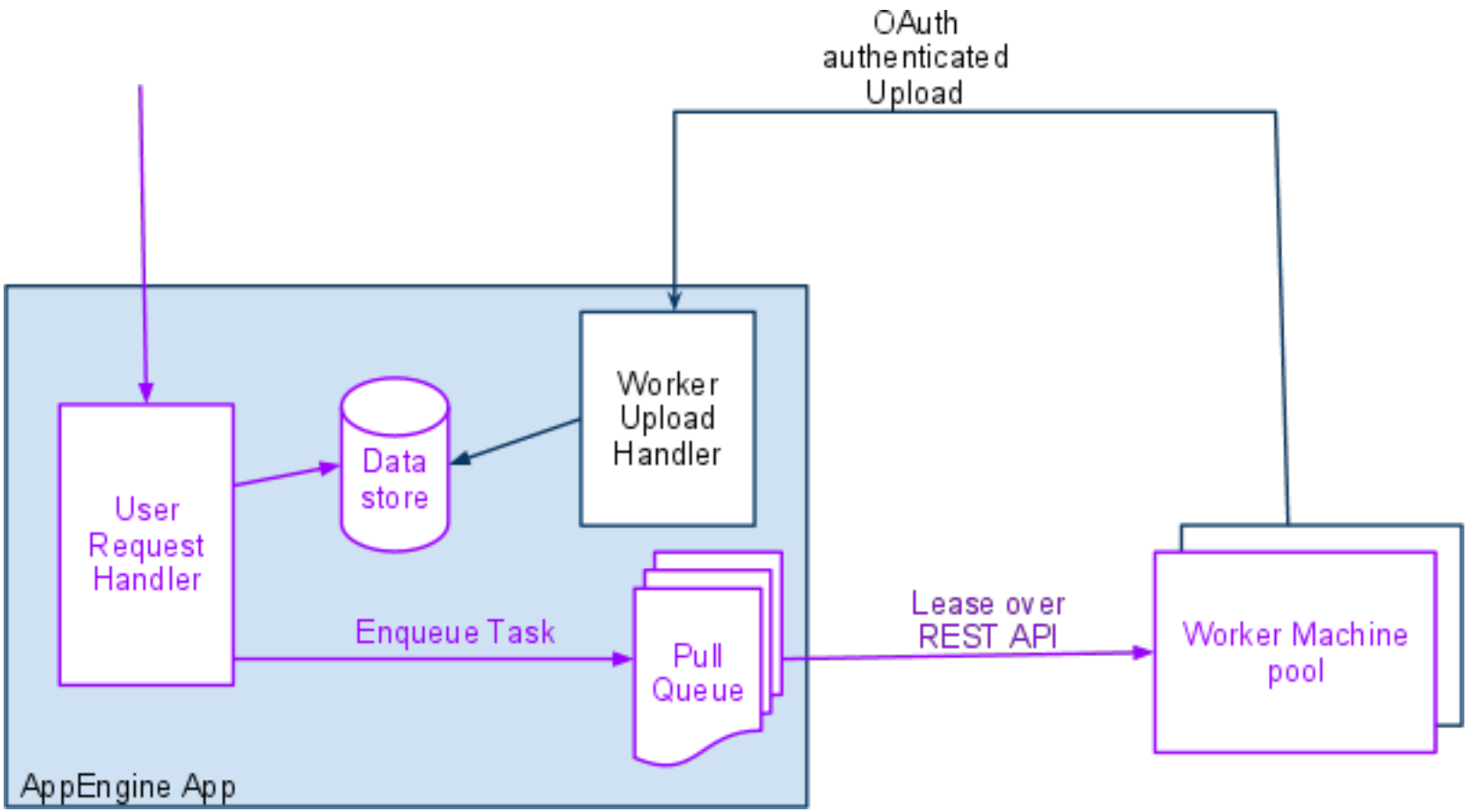
Example - Image Converter

Working Outside Google App Engine



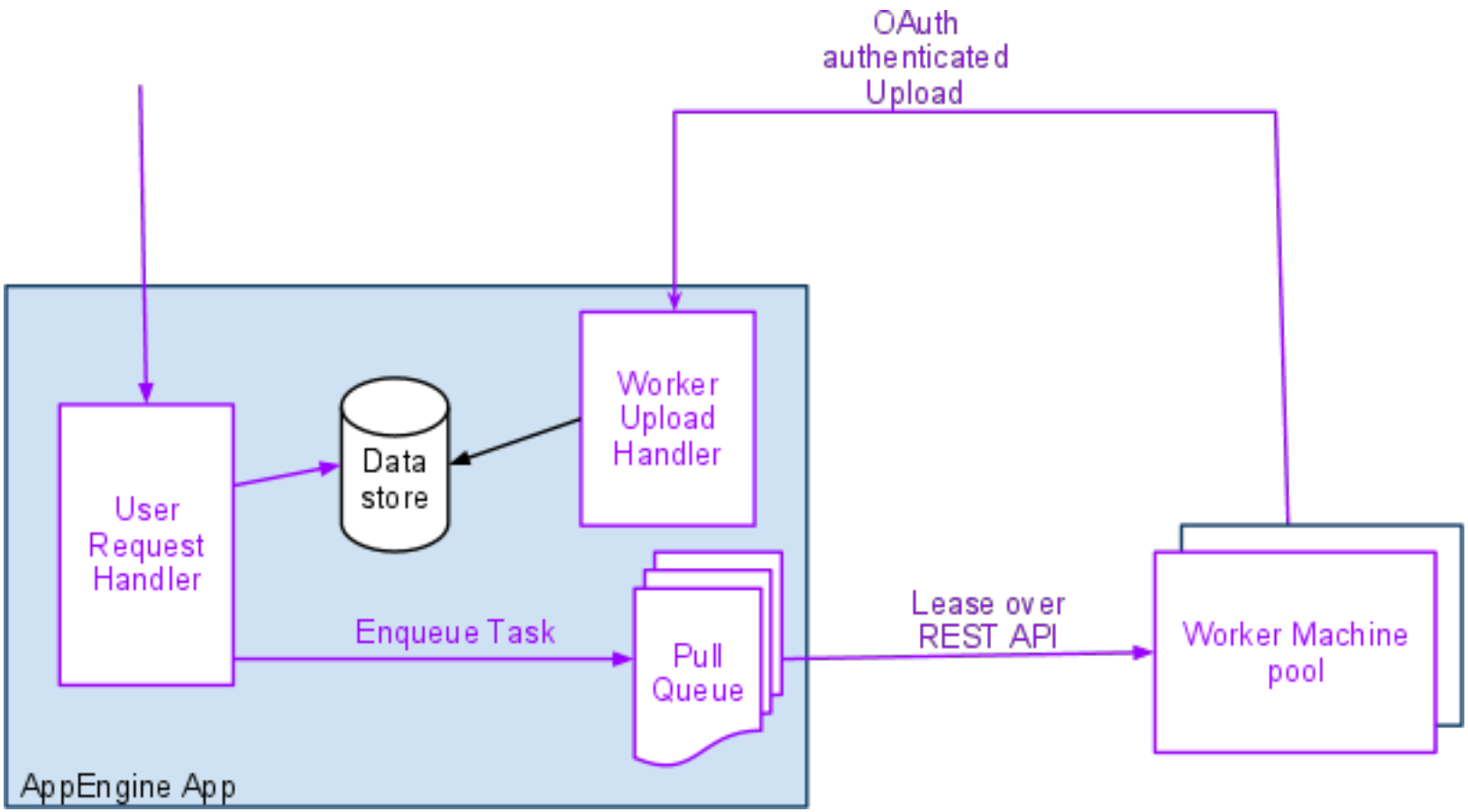
Example - Image Converter

Working Outside Google App Engine



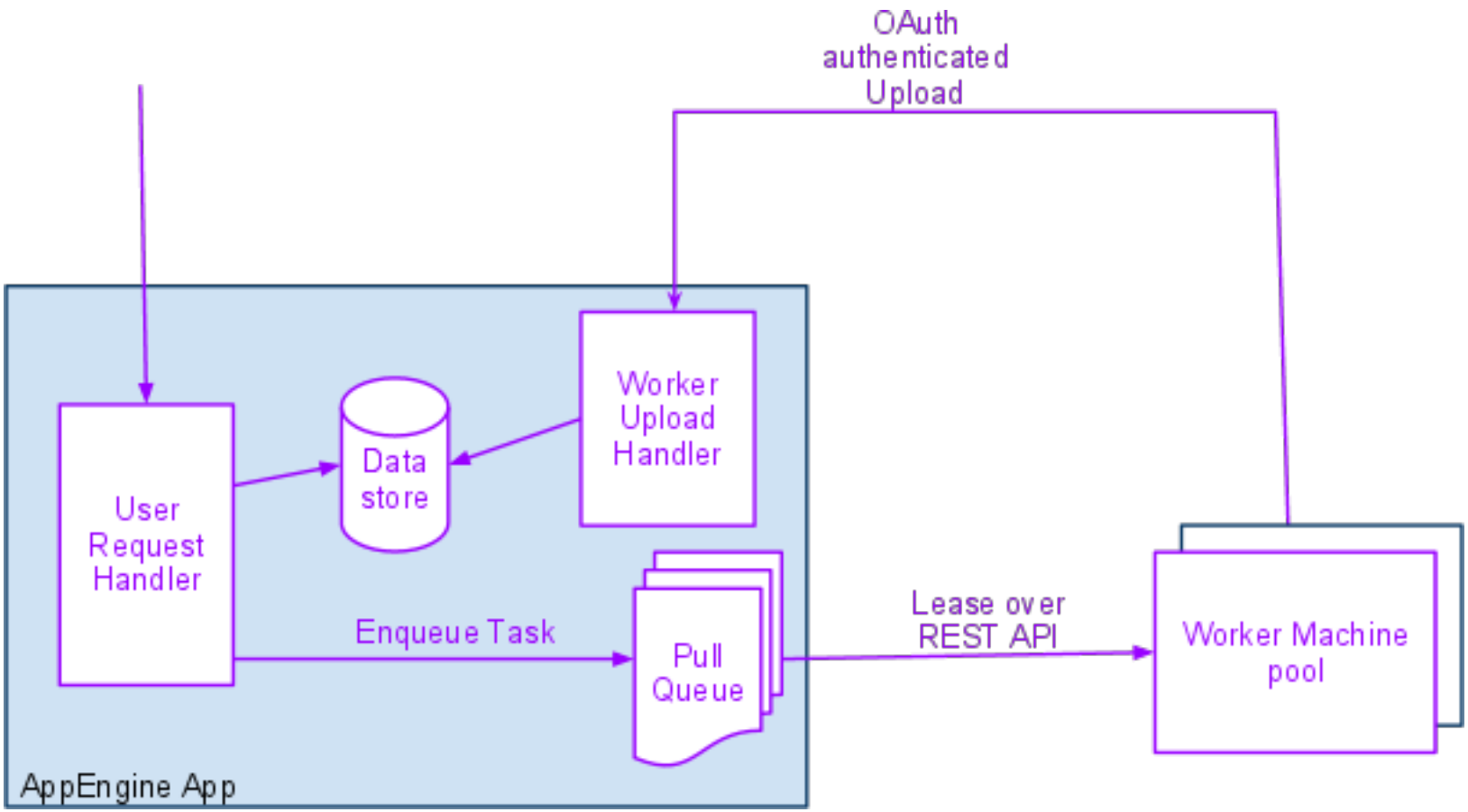
Example - Image Converter

Working Outside Google App Engine



Example - Image Converter

Working Outside Google App Engine



Example - Image Converter

Adding Tasks in Google App Engine

- Create app with queue.yaml

```
queue:
- name: photos
  mode: pull
  acl:
  - user_email: svivek@google.com
  - user_email: nverne@google.com
```

- Write App Engine code to insert tasks into queue as before
- Assume photos are the payload in the task

Example - Image Converter

Calling the API : API Client (Python)

- Google API Client library is available in Python, Java etc.
- In Python, can use discovery to build methods for an API on the fly :

```
from apiclient.discovery import build
# discovery URL template
discovery_uri =
    'https://www.googleapis.com/discovery/v0.2beta1' +
    '/describe/{api}/{apiVersion}'

# Now build API by filling in "api" and "version"
self.task_api = build('taskqueue',
                      service_version='v1beta1',
                      http=http,
                      discoveryServiceUrl=discovery_uri)
```

Example - Image Converter

Verify things work : use gtaskqueue

- First get an OAuth token scoped to taskqueue-api using an OAuth dance
- Store token in a file for future API calls
- Run one command on your queue and see the result in JSON

```
gtaskqueue leasetask --project_name=s~imageconvertdemo --
taskqueue_name=imageconvert --lease_secs=30 --num_tasks=10
```

Example - Image Converter

Worker code (Python)

```
while True:
# Build the API request
request = task_api.lease(project="myappid",
            taskqueue="photos",
                    leaseSecs=120,
                    numTasks=5,
                    body={})
  # Call the API over REST, get JSON response
  response = request.execute()
  for task in response.get('items'):
    ... exec('convert -annotate')
    ... post output to image-flipper.appspot.com/taskdata
    tasks_api.delete(project="myappid",
                taskqueue="photos",
                task=task.id)
```

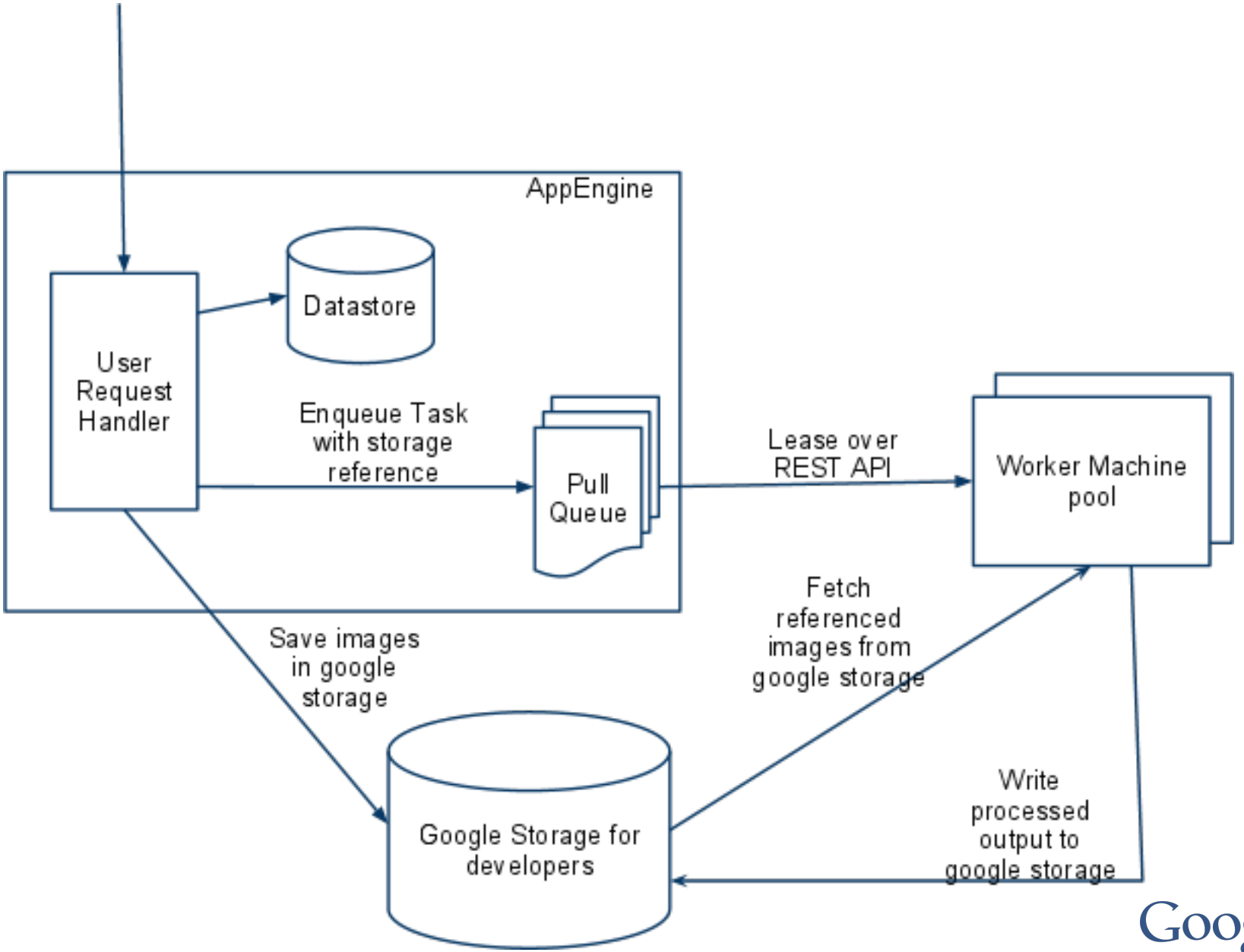
Example - Photo Stitcher

Using a reference to pass large amount of data

- Tasks are references to sets of photos, originals in Google Storage for Developers(GS)
  - Added to App Engine queues
- Workers running in VMs stitch photos referenced in tasks
- Results updated on GS file

Example - Photo Stitcher

Using a reference to pass large amount of data



Gotchas

Things you need to worry about with pull queues

- Scaling:
  - You need to scale up and down the workers on your own
  - Can use *queue.get()* call to get queue statistics
- Choosing a lease:
  - Choose close to worst possible time a worker can take
- Idempotent workers:
  - If the task overruns the lease, another worker will be able to lease it out
  - Could cause task to execute more than once
- Posting back to AppEngine from workers:
  - Write a handler in your AppEngine app that workers call
  - Use Google Storage for Developers/GAE blobstore
  - See samples

Links and Resources

- REST API Samples:
  - Python / Java
    http://code.google.com/p/google-api-{python,java}-client
- AppEngine application examples :
  - Voterlator, Imageconvert:
    http://code.google.com/p/google-app-engine-samples/
- Documentation:
  - Python / Java
    http://code.google.com/appengine/docs/{python,java}/taskqueue/
- API Explorer:
  - http://code.google.com/apis/explorer

Take it for a run and let us know what you think!

Feedback on this talk: **http://goo.gl/sAXZh**

Questions?