Google Developers

Orchestrating Google Compute Engine through Google App Engine

Adam Eijdenberg, Product Manager - Google Compute Engine
Alon Levi, Tech Lead/Manager - Google App Engine

Why should I care?

App
Engine

+

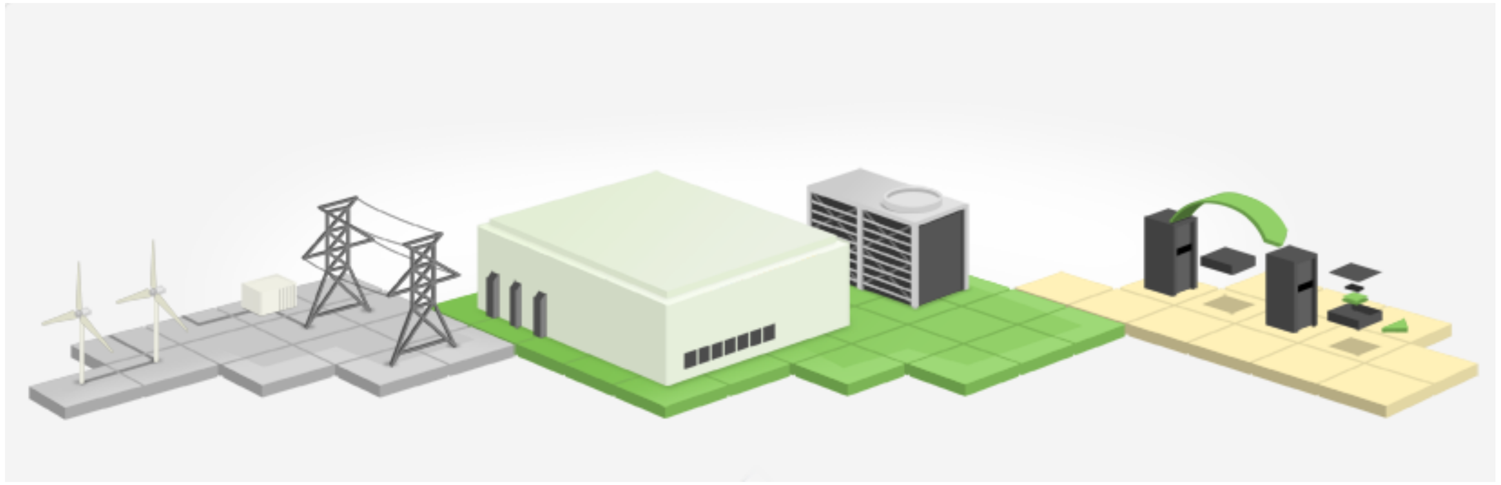
Compute
Engine

by Google™

Globally Distributed

Efficient

Secure



google.com/about/datacenters/

Google App Engine

Platform as a Service

Easy to write
Simple to scale
Trivial to manage

Google App Engine

When is it optimal?

When you don't want to worry about infrastructure

- Web UIs
- API Endpoints
- Workflows
- Managed Backends

Google Compute Engine

Infrastructure as a Service

High Performance
Easy Scalability
Great Value

Google Compute Engine

When is it optimal?

When you need low-level access or fine grained control

- Large Batch Workloads
- Native Code
- Off-the-Shelf OSS

Why to use them together

To get "unstuck" from PaaS

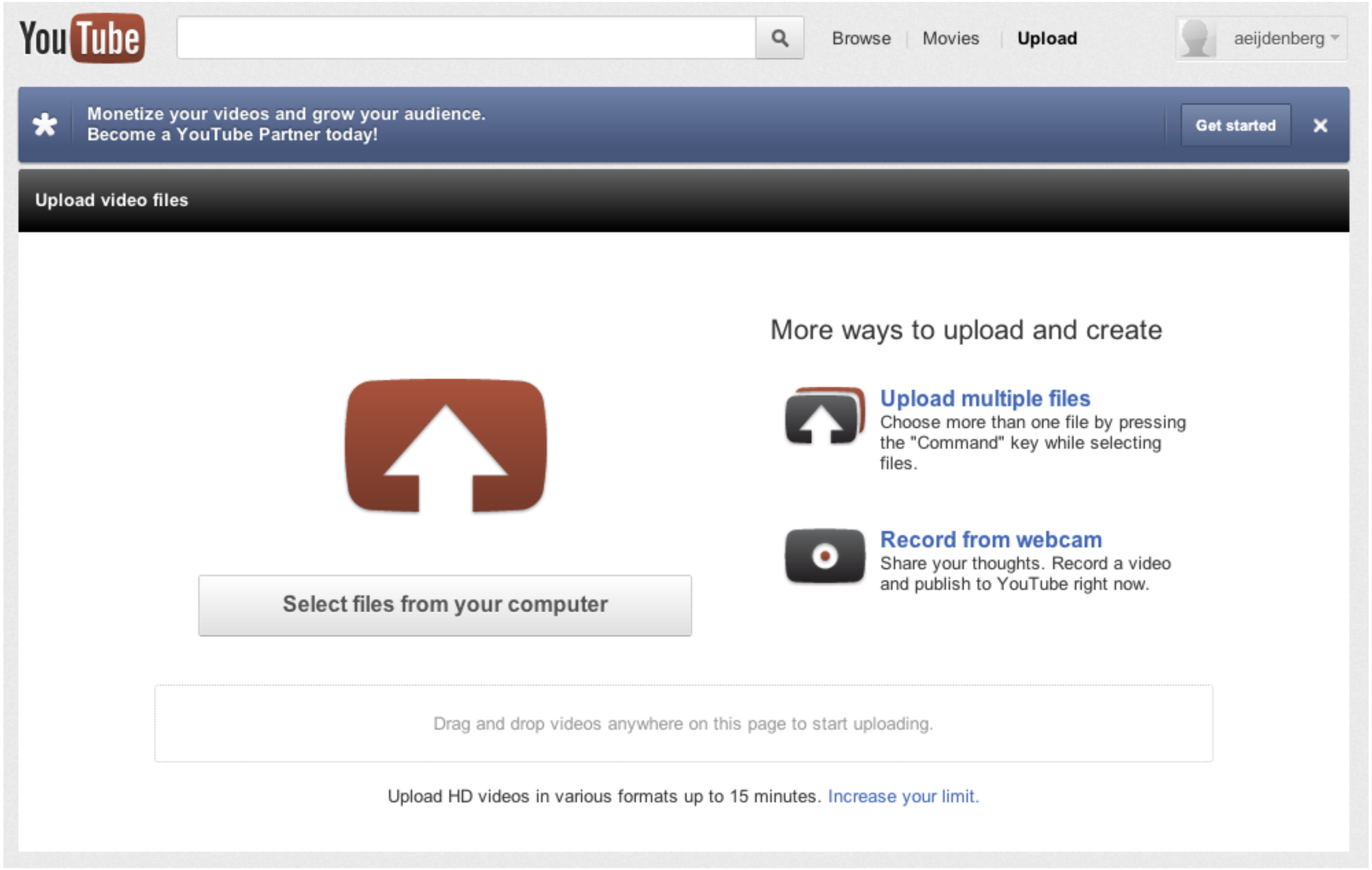
- Extend the power of your App Engine app

To simplify your IaaS systems

- Make your VM solution easier to manage or maintain

An Example App

Building a video sharing site



Monetize your videos and grow your audience.
Become a YouTube Partner today!

More ways to upload and create

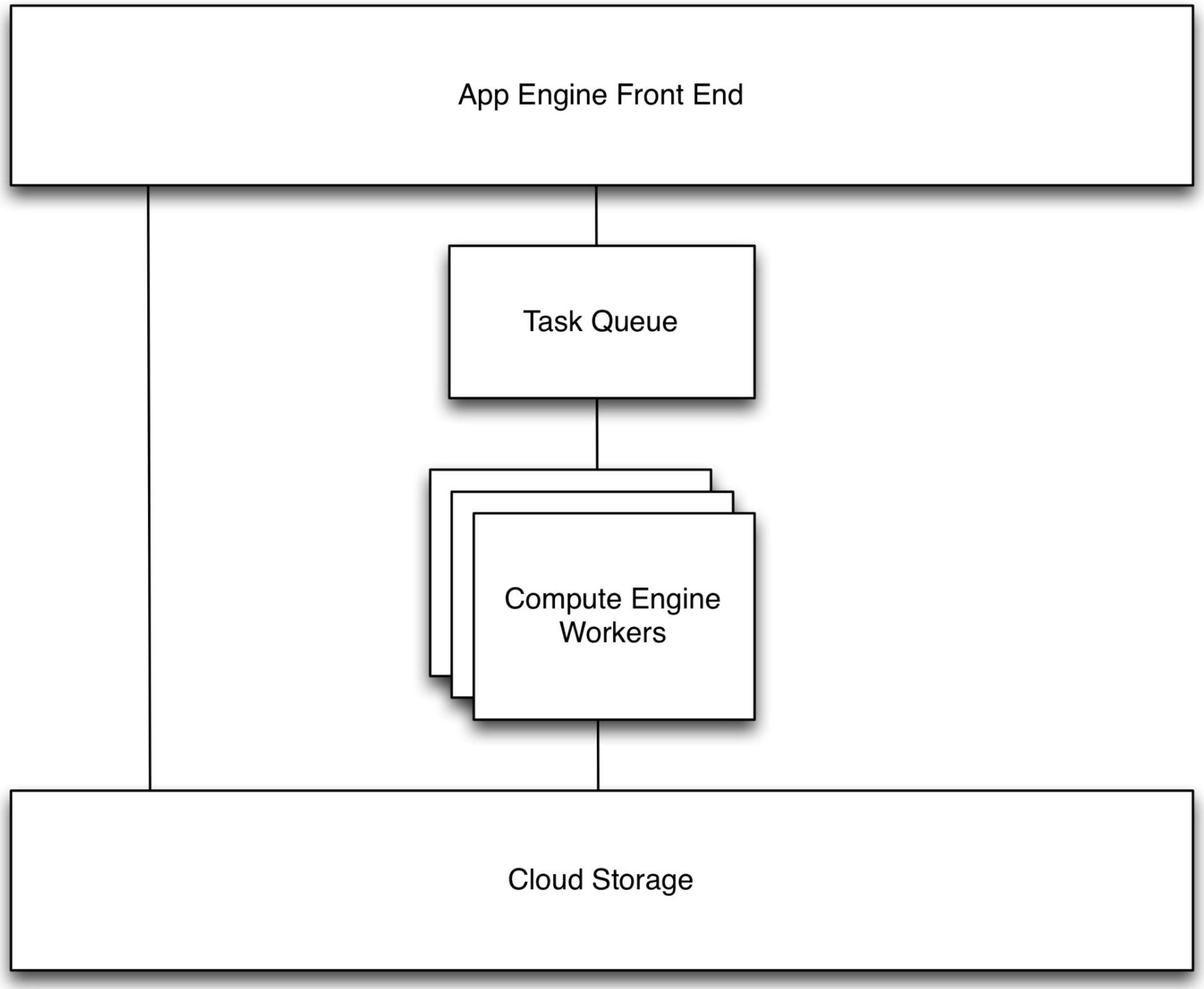
Upload multiple files
Choose more than one file by pressing the "Command" key while selecting files.

Record from webcam
Share your thoughts. Record a video and publish to YouTube right now.

Drag and drop videos anywhere on this page to start uploading.

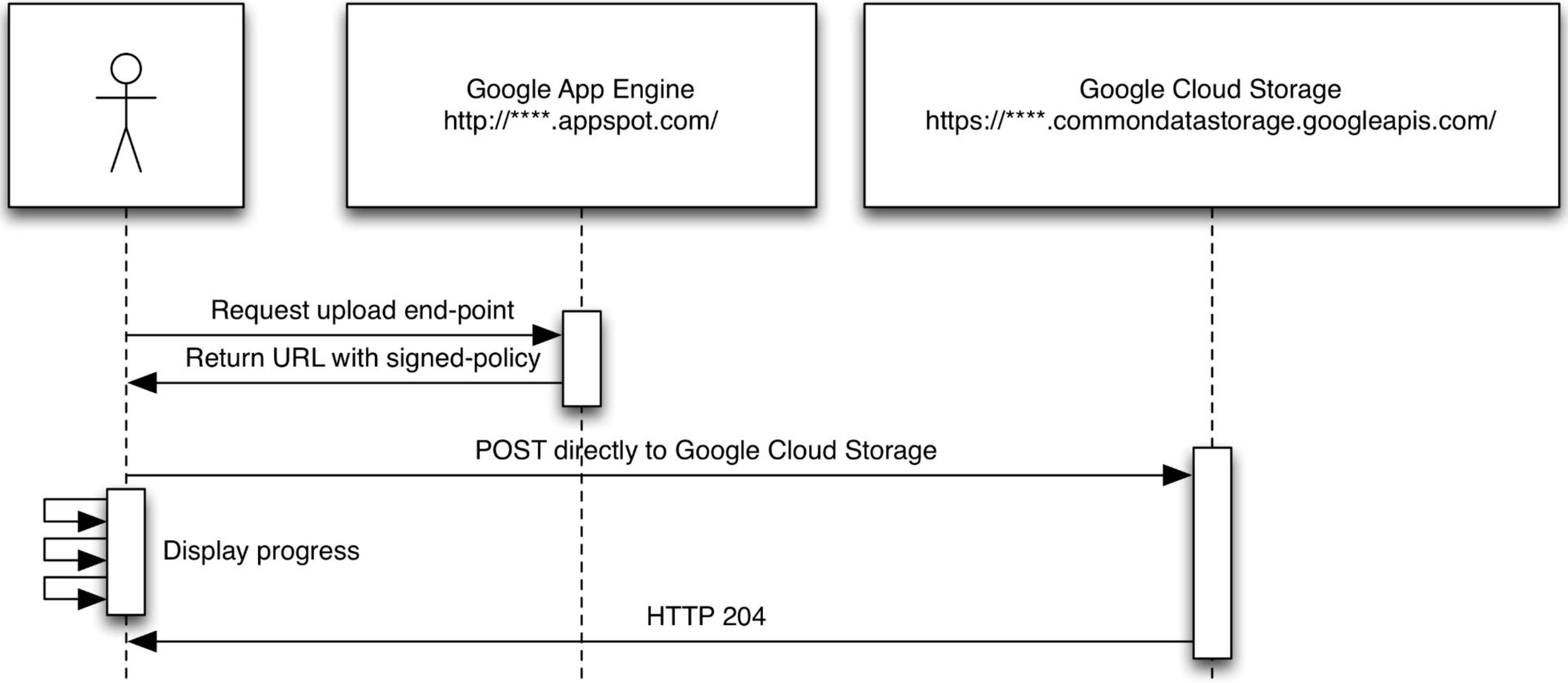
Upload HD videos in various formats up to 15 minutes. Increase your limit.

Architecture



Part 1 / Demo 1: Uploading

Uploading files


Google App Engine
http://****.appspot.com/
Google Cloud Storage
https://****.commondatastorage.googleapis.com/
Request upload end-point
Return URL with signed-policy
POST directly to Google Cloud Storage
Display progress
HTTP 204

Display file upload progress

Javascript

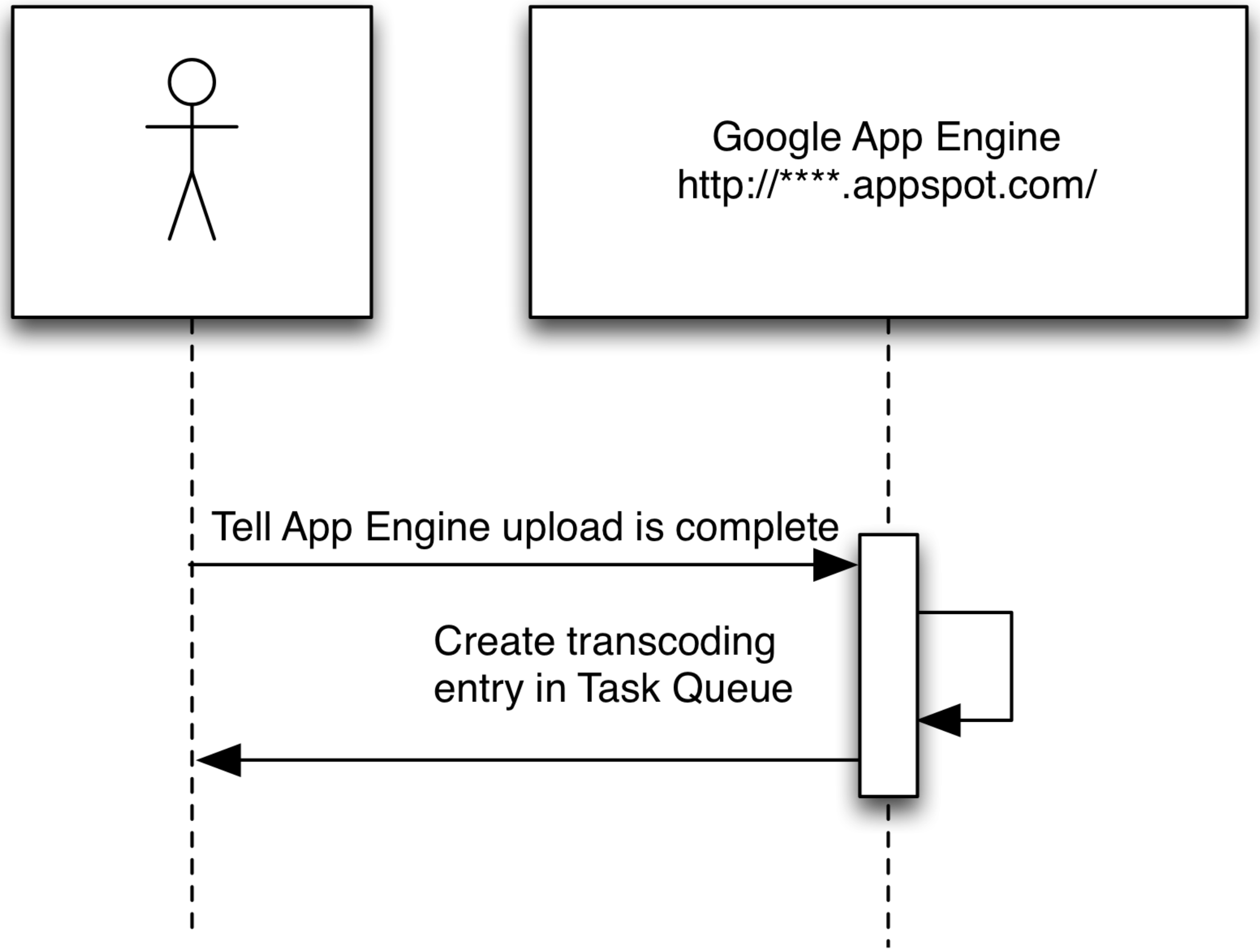
```
function startUpload(f) {
  $.ajax({url: '/getFileUploadEndpoint', cache: false, success: function (data) {
      var fd = new FormData();
      for (var n in data.params) { fd.append(n, data.params[n]); }
      fd.append('file', f);
      var xhr = new XMLHttpRequest();
      xhr.upload.addEventListener('progress', function (evt) { $('#progress').text(evt.loaded + ' / ' + evt.total); }, false);
      xhr.upload.addEventListener('load', function (evt) { $('#progress').text('Complete'); }, false);
      xhr.open(data.method, data.url);
      xhr.send(fd);
  }});
}
$('#droparea').bind('drop', function (evt) { evt.preventDefault(); startUpload(evt.originalEvent.dataTransfer.files[0]); });
```

Signing a storage policy

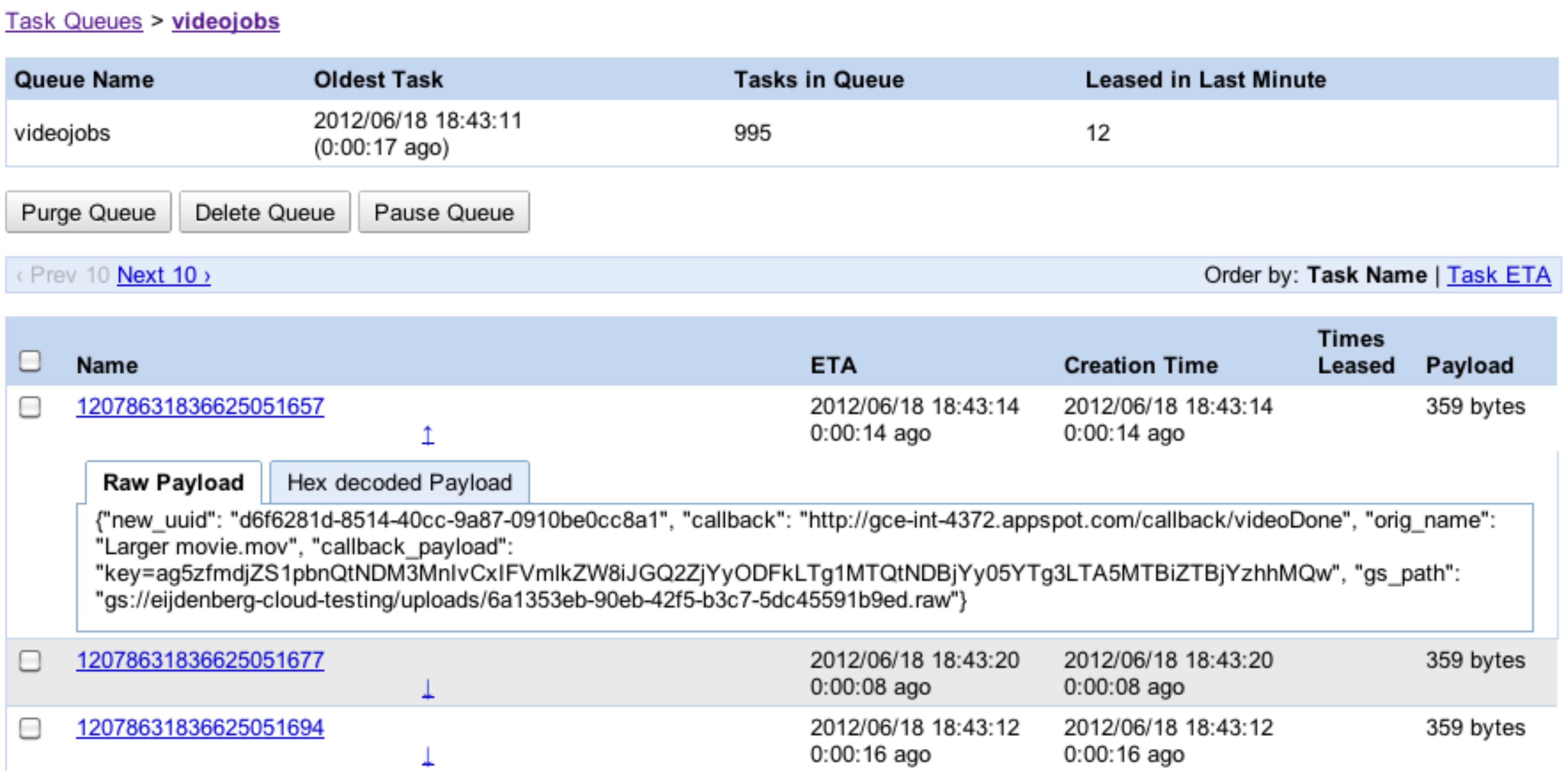
Python

```python
class FileUploadEndpointHandler(webapp2.RequestHandler):
  def get(self):
    expires = '%sZ' % ((datetime.datetime.utcnow() + datetime.timedelta(hours=1)).isoformat()[:19])
    fname = 'uploads/%s.raw' % str(uuid.uuid4())
    policy = base64.b64encode(json.dumps({'expiration': expires,
                                         'conditions': [{'bucket': BUCKET}, {'key': fname}]}))
    signed = base64.b64encode(app_identity.sign_blob(policy)[1])
    self.response.headers['Content-Type'] = 'application/json'
    self.response.write(json.dumps({
        'method': 'POST', 'url': 'https://%s.commondatastorage.googleapis.com/' % BUCKET,
        'params': {'key': fname, 'GoogleAccessId': app_identity.get_service_account_name(),
                   'signature': signed, 'policy': policy}
    }))
```

Create task queue entry


Google App Engine
http://****.appspot.com/
Tell App Engine upload is complete
Create transcoding entry in Task Queue

Task Queue in the admin console



Task Queues > videojobs

Queue Name	Oldest Task	Tasks in Queue	Leased in Last Minute
videojobs	2012/06/18 18:43:11 (0:00:17 ago)	995	12

Purge Queue | Delete Queue | Pause Queue

‹ Prev 10 Next 10 › Order by: **Task Name** | Task ETA

	Name	ETA	Creation Time	Times Leased	Payload
☐	12078631836625051657 ↥	2012/06/18 18:43:14 0:00:14 ago	2012/06/18 18:43:14 0:00:14 ago		359 bytes

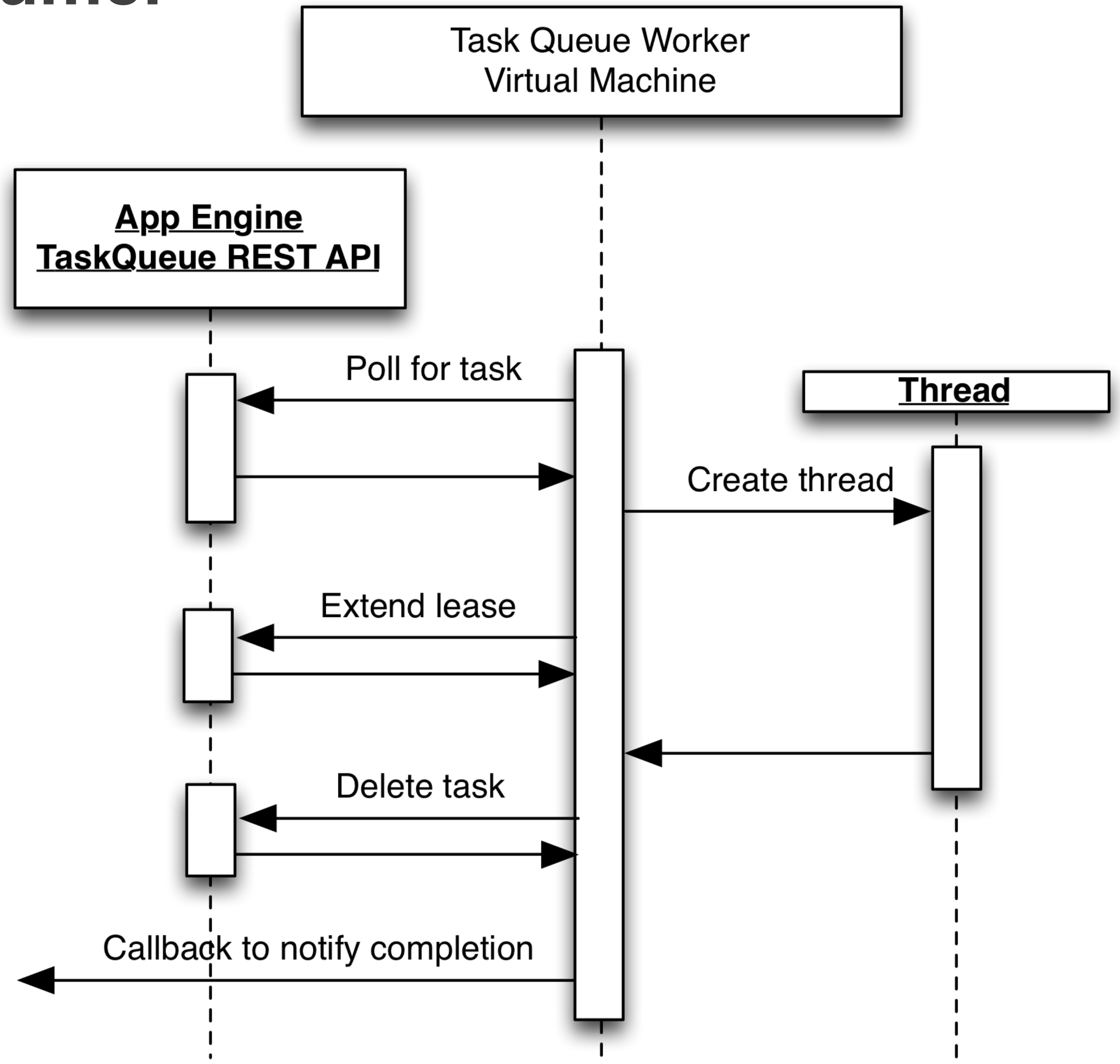
Raw Payload | Hex decoded Payload

```
{"new_uuid": "d6f6281d-8514-40cc-9a87-0910be0cc8a1", "callback": "http://gce-int-4372.appspot.com/callback/videoDone", "orig_name": "Larger movie.mov", "callback_payload": "key=ag5zfmdjZS1pbnQtNDM3MnlvCxIFVmlkZW8iJGQ2ZjYyODFkLTg1MTQtNDBjYy05YTg3LTA5MTBiZTBjYzhhMQw", "gs_path": "gs://eijdenberg-cloud-testing/uploads/6a1353eb-90eb-42f5-b3c7-5dc45591b9ed.raw"}
```

	Name	ETA	Creation Time	Times Leased	Payload
☐	12078631836625051677 ↧	2012/06/18 18:43:20 0:00:08 ago	2012/06/18 18:43:20 0:00:08 ago		359 bytes
☐	12078631836625051694 ↧	2012/06/18 18:43:12 0:00:16 ago	2012/06/18 18:43:12 0:00:16 ago		359 bytes

Part 2: On the VM

Task queue consumer

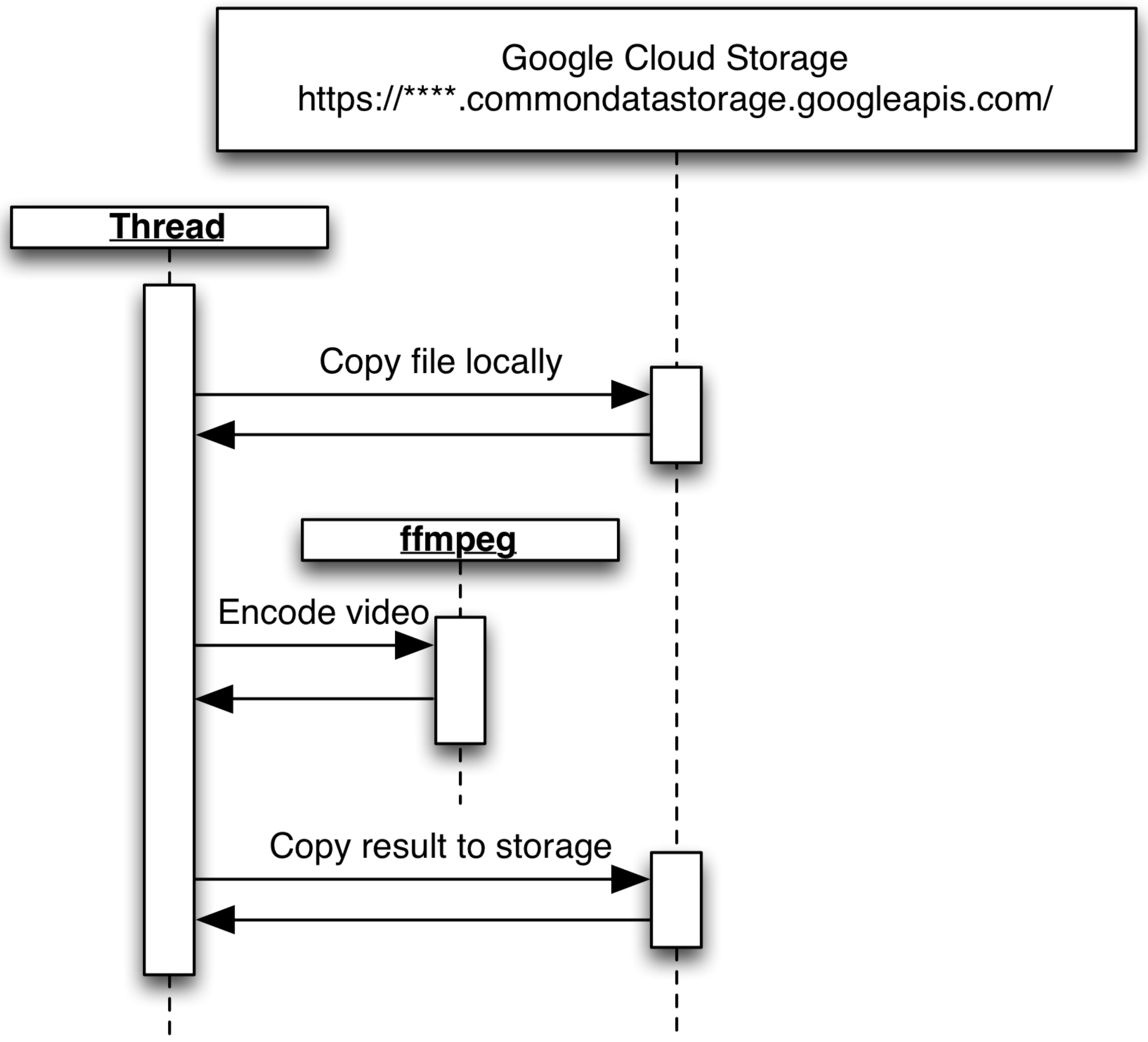


Task queue processing loop

Python

```
while True:
  result = service.tasks().lease(leaseSecs=30, taskqueue='jobs', project='s~project-name', numTasks=1, body='').execute()
  if 'items' in result:
    for item in result['items']:
      t = threading.Thread(target=PerformTask, args=(json.loads(base64.b64decode(item['payloadBase64'])),))
      t.start()
      while t.is_alive():
        t.join(20)
        if t.is_alive():
          item.update(service.tasks().update(taskqueue='jobs', project='s~project-name', newLeaseSeconds=30, body={
            'id': item['id'], 'kind': 'taskqueues#task', 'leaseTimestamp': item['leaseTimestamp'], 'queueName': 'jobs'}, task=item['id']))
      service.tasks().delete(taskqueue='jobs', project='s~project-name', task=item['id']).execute()
  else:
    time.sleep(10)
```

Workload


Google Cloud Storage
https://****.commondatastorage.googleapis.com/
Thread
Copy file locally
ffmpeg
Encode video
Copy result to storage

Transcoding the video

Python

```python
def PerformTask(task):
  orig_name = task['gs_path'].split('/')[-1][:-4]
  final_dest = '/'.join(task['gs_path'].split('/')[:-2] + ['rendered', '%s.mp4' % task['new_uuid']])
  tmp_src = '/tmp/%s.raw.%s' % orig_name
  tmp_dst = '/tmp/%s.%s.mp4' % orig_name
  subprocess.call(['/usr/bin/gsutil', 'cp', task['gs_path'], tmp_src])
  subprocess.call(['/usr/bin/ffmpeg', '-y', '-i', tmp_src, '-s', '432x320', ..., tmp_dst])
  subprocess.call(['/usr/bin/gsutil', 'cp', tmp_dst, final_dest])
  subprocess.call(['/usr/bin/gsutil', 'setacl', 'public-read', final_dest])
  os.remove(tmp_src)
  os.remove(tmp_dst)
  urllib.urlopen(task['callback'], task['callback_payload']).read()
```

Create VM and install software

Commands

```
$ gcutil addinstance --zone=us-east1-a --machine_type=n1-standard-8-d
    --service_account_scopes="https://www.googleapis.com/auth/devstorage.full_control,
                             https://www.googleapis.com/auth/taskqueue" transcoder
$ gcutil ssh transcoder
Welcome to Ubuntu 12.04 LTS (GNU/Linux 2.6.39-gcg-201203291735 x86_64)

$ sudo apt-get -y install ffmpeg screen
$ gsutil cp gs://<bucket name>/scripts/task_queue_reader.py .
$ screen -d -m python task_queue_reader.py
```

Special note: Application authentication and authorization

- Service accounts - created through Admin Console / API Access
  - Download private key (".p12"), distribute with code, sign with crypto library
  - Can be used anywhere

AND

- Provisioned service accounts - automatically created by App Engine and Compute Engine
  - `app_identity.get_access_token(...)`
  - `curl http://metadata/0.1/meta-data/service-accounts/default/acquir`
  - Can only be used within container

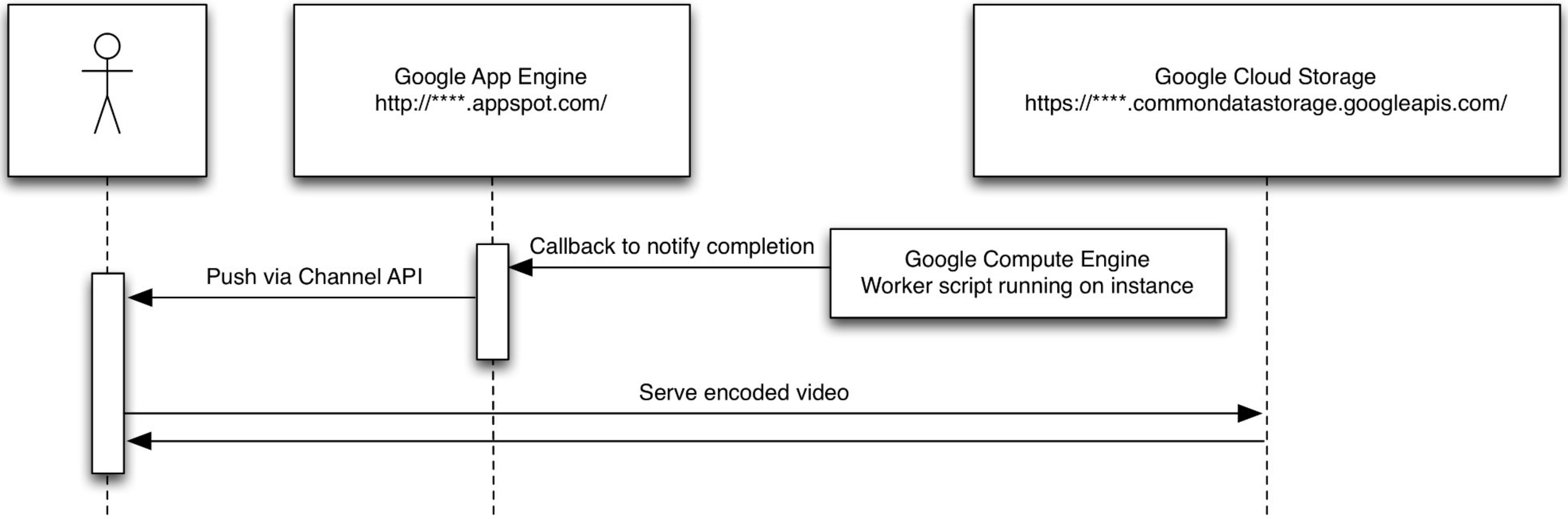
- Either method results in an email address that can be added to ACLs
  - queue.yaml
  - gsutil setacl ...
  - API Console / Team

Authentication from instance

Python

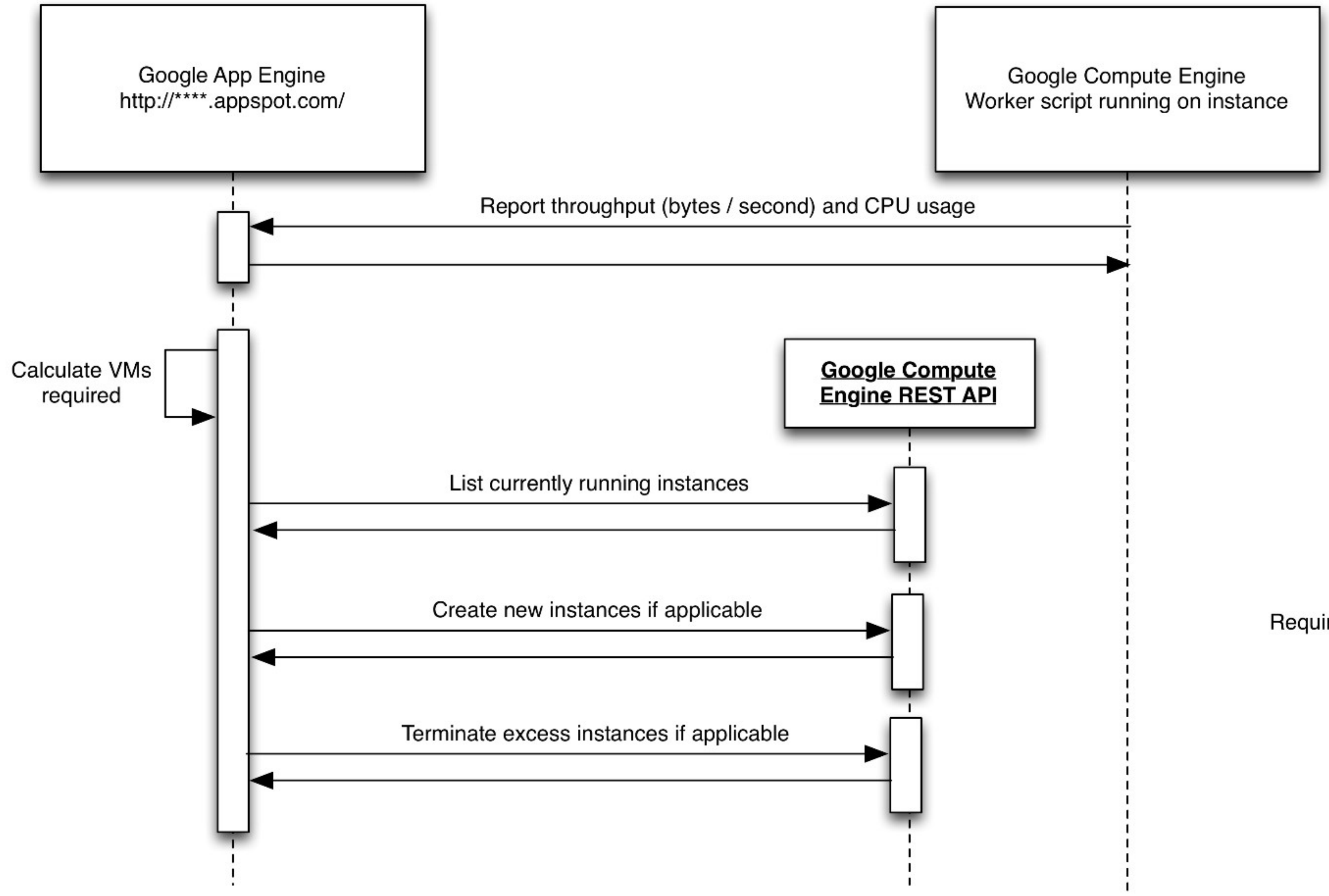
```
import httplib2, json, urllib
from oauth2client.client import AccessTokenCredentials
from apiclient.discovery import build

def FetchToken():
  return AccessTokenCredentials(json.loads(httplib2.Http().request(
      'http://metadata/0.1/meta-data/service-accounts/default/acquire?'
      + urllib.urlencode({'scopes': 'https://www.googleapis.com/auth/taskqueue'}
  ), method='POST', headers={'Content-Length': '0'})[1])['accessToken'], '').authorize(httplib2.Http())

service = build('taskqueue', 'v1beta2')
service.tasks().lease(...).execute(http=FetchToken())
```

Show result


Google App Engine
http://****.appspot.com/
Google Cloud Storage
https://****.commondatastorage.googleapis.com/
Callback to notify completion
Google Compute Engine
Worker script running on instance
Push via Channel API
Serve encoded video

Part 3: Scaling

Scaling


Google App Engine
http://****.appspot.com/
Google Compute Engine
Worker script running on instance
Report throughput (bytes / second) and CPU usage
Calculate VMs required
Google Compute Engine REST API
List currently running instances
Create new instances if applicable
Terminate excess instances if applicable

$$\text{Required VMs} = \frac{\dfrac{\text{total bytes video outstanding}}{\text{target seconds to process}}}{\text{average throughput (bytes/second) per VM}}$$

Demo 2: Scaling

Heartbeat

Python

```
import re, subprocess, sys, time, urllib

ENDPOINT = sys.argv[1]
INSTANCE_NAME = urllib.urlopen('http://metadata/0.1/meta-data/hostname').read().split('.')[0]
SPACE_SPLIT = re.compile(r'\s+')

while True:
  stdout, strerr = subprocess.Popen(['/usr/bin/mpstat', '1', '1'], stdout=subprocess.PIPE).communicate()
  data = dict(zip([l[1:] for l in SPACE_SPLIT.split(stdout.split('\n')[2])[2:]],
                                          SPACE_SPLIT.split(stdout.split('\n')[3])[2:]))
  data['hostname'] = INSTANCE_NAME
  urllib.urlopen(ENDPOINT, urllib.urlencode(data)).read()
  time.sleep(1)
```

Scaling VMs - Simplistic approach

Python

```
vm_throughputs = [i.throughput for i in Instance.all()]
average_throughput = sum(vm_throughputs) * 1.0 / len(vm_throughputs)

outstanding_bytes = sum(v.orig_size for v in Video.all().filter('status =', 0))

num_processes = (outstanding_bytes * 1.0 / TARGET_SECONDS) / average_throughput
desired_vms = int(math.ceil(num_processes * 1.0 / WORKERS_PER_VM))
required_vms = max(MIN_SERVERS, min(MAX_SERVERS, desired_vms))

if required_vms < len(vm_throughputs):
  CreateMoreVMs(required_vms - len(vm_throughputs))
elif required_vms > len(vm_throughputs):
  TerminateExcessVMs(len(vm_throughputs) - required_vms)
```

Creating a VM

Python

```
project_path = 'projects/' + project
service.instances().insert(project=project, body={
  'name': 'taskqueue-' + str(uuid.uuid4()), 'kind': 'compute#instance', 'image': project_path + '/images/video-enc-2012-06-14',
  'machineType': project_path + '/machine-types/n1-standard-8', 'zone': project_path + '/zones/us-east1-a',
  'networkInterfaces': [{'accessConfigs': [{'type': 'ONE_TO_ONE_NAT', 'name': 'External NAT'}],
                                  'network': project_path + '/networks/default'}],
  'serviceAccounts': [{'scopes': ['https://www.googleapis.com/auth/devstorage.full_control',
                                               'https://www.googleapis.com/auth/taskqueue'], 'email': 'default'}],
  'metadata': {'kind': 'compute#metadata', 'metadata': [{'key': 'startup-script', 'value': '''#!/bin/sh
      export APP=%s
      /usr/bin/screen -d -m /usr/bin/python /root/task_queue_reader.py $APP
      /usr/bin/screen -d -m /usr/bin/python /root/cpu_monitor.py http://$APP.appspot.com/callback/cpuReport''' % appname}]},
}).execute()
```

Demo 3: Throughput

Using App Engine AND Compute Engine

- Use Compute Engine to augment the capabilities of App Engine
- Use App Engine to manage Compute Engine VM instance creation/deletion
- Compute Engine is in limited preview. Sign-up, browse the documentation: http://cloud.google.com/

Reference URLs - File upload

- Create Google Cloud Storage project
  - https://developers.google.com/storage/docs/signup
- Create service account for the project
  - https://developers.google.com/console/help/#service_accounts
- Enable Cross-Origin Resource Sharing
  - https://developers.google.com/storage/docs/cross-origin
- Create App Engine app
  - https://appengine.google.com/
- Build AJAX call to create POST URL with signed policy to allow upload to a unique file path
  - https://developers.google.com/storage/docs/reference-methods#postobject
  - https://developers.google.com/storage/docs/reference-methods#policydocument
- Drag and drop using HTML5 DataTransfer object
  - http://www.w3.org/TR/html5/dnd.html#datatransfer

Reference URLs - Build first worker VM

- Enable Google Compute Engine on your Cloud Storage project
  - https://developers.google.com/compute/
- Create virtual machine instances and build task queue processor
  - https://developers.google.com/compute/docs/gcompute_setup
  - https://developers.google.com/appengine/docs/java/taskqueue/rest
- Enable service account and scope on virtual machine, add ACL to TaskQueue configuration
  - https://developers.google.com/compute/docs/authentication
  - https://developers.google.com/appengine/docs/python/config/queue#Defining_Pull_Queues
- Build App Engine call to create queue entry, and enable channel API for feedback to client
  - https://developers.google.com/appengine/docs/python/taskqueue/overview-pull
  - https://developers.google.com/appengine/docs/python/channel/overview
- Save virtual machine image and/or create startup-script
  - https://developers.google.com/compute/docs/images
  - https://developers.google.com/compute/docs/howtos/startupscript

Reference URLs - Scale horizontally

- Use Google Compute Engine REST API
    - https://developers.google.com/compute/docs/reference/v1beta12/instances
- Authenticate using either App Engine robot (may not suit domain hosted projects)
    - https://developers.google.com/appengine/docs/python/appidentity/overview
    - http://code.google.com/p/google-api-python-client/
- Create attractive graphics from Javascript
    - http://code.google.com/p/flot/
    - http://code.google.com/p/jgauge/

Questions?

Thank You!

Adam Eijdenberg - eijdenberg@google.com
Alon Levi - alevi@google.com

Google Developers