fog

OR:

HOW I LEARNED TO STOP WORRYING AND LOVE THE CLOUD

geemus (Wesley Beary)

web: *github.com/geemus*

twitter: @geemus

employer and sponsor

web: *engineyard.com*

twitter: *@engineyard*

CLOUD

API driven on demand services

core: compute, dns, storage

also: *kvs, load balance, ...*

What?

on demand - only pay for what you actually use

flexible - add and remove resources in minutes (instead of weeks)

repeatable - code, test, repeat

resilient - build better systems with transient resources

Why Worry?

option overload - which provider/service should I use

expertise - each service has yet another knowledge silo

tools - vastly different API, quality, maintenance, etc

standards - slow progress and differing interpretations

Ruby cloud services

web: *github.com/geemus/fog*

twitter: @*fog*

Why?

portable - AWS, Bluebox, Brightbox, Google, Rackspace, Slicehost, Terremark, ...

powerful - compute, dns, storage, collections, models, mocks, requests, ...

established - 182k downloads, 1371 followers, 225 forks, 104 contributors, me, ...

Fog.mock! - faster, cheaper, simulated cloud behavior

Who?

libraries - carrierwave, chef, deckard, gaff, gemcutter, ...

products - DevStructure, Engine Yard, iSwifter, OpenFeint, RowFeeder, ...

Interactive Bit!

cloud

fog

What?

That's great and all but *I don't have a use case...*

uptime - because who wants a busted web site?

Setup

```
geymus ~ ⌘ gem install fog
```

or

```
geymus ~ ⌘ sudo gem install fog
```

Get Connected

```
1 credentials = {
2   :provider           => 'Rackspace',
3   :rackspace_api_key  => RACKSPACE_API_KEY,
4   :rackspace_username => RACKSPACE_USERNAME
5 }
6
7 # setup a connection to the service
8 compute = Fog::Compute.new(credentials)
```

Boot that Server

```
1 server_data = compute.create_server(
2   1,
3   49
4 ).body['server']
5
```

Boot that Server

```
server_data = compute.create_server(
  1,
  49
).body['server']

until compute.get_server_details(
  server_data['id']
).body['server']['status'] == 'ACTIVE'
end

```

Boot that Server

```
server_data = compute.create_server(
  1,
  49
).body['server']

until compute.get_server_details(
  server_data['id']
).body['server']['status'] == 'ACTIVE'
end

commands = [
  %{'mkdir .ssh'},
  %{'echo #{File.read('~/.ssh/id_rsa.pub')} >> ~/.ssh/authorized_keys'},
  %{passwd -l root},
]

Net::SSH.start(
  server_data['addresses'].first,
  'root',
  :password => server_data['password']
) do |ssh|
  commands.each do |command|
    ssh.open_channel do |ssh_channel|
      ssh_channel.request_pty
      ssh_channel.exec(%{bash -lc '#{command}'})
      ssh.loop
    end
  end
end
```

Worry!

arguments - what goes where, what does it mean?

portability - most of this will only work on Rackspace

disservice - back to square one, but with tools in hand

Bootstrap

```
server_attributes = {
  :image_id            => '49',
  :private_key_path => PRIVATE_KEY_PATH,
  :public_key_path  => PUBLIC_KEY_PATH
}

# boot server and setup ssh keys
server = compute.servers.bootstrap(server_attributes)
```

Servers?

```
1 compute.servers # list servers, same as #all
2
3 compute.servers.get(1234567890) # server by id
4
5 compute.servers.reload # update to latest
6
7 compute.servers.new(attributes) # local model
8
9 compute.servers.create(attributes) # remote model
```

ping

```
1 # ping target 10 times
2 ssh_results = server.ssh("ping -c 10 #{target}")
3 stdout = ssh_results.first.stdout
4
5 # parse result, last line is summary
6 # round-trip min/avg/max/stddev = A.A/B.B/C.C/D.D ms
7 stats = stdout.split("/n").last.split(' ')[-2]
8 min, avg, max, stddev = stats.split('/')
```

NOTE: most complex code was string parsing!?!

cleanup

```
1  # shutdown the server
2  server.destroy
3
4  # return the data as a hash
5  {
6    :min    => min,
7    :avg    => avg,
8    :max    => max,
9    :stddev => stddev
10 }
```

Next!

```
-server_data = compute.create_server(
+compute.import_key_pair(
+  'id_rsa.pub',
+  File.read('~/.ssh/id_rsa.pub')
+)
+
+compute.authorize_security_group_ingress(
+  'CidrIp'      => '0.0.0.0/0',
+  'FromPort'    => 22,
+  'IpProtocol'  => 'tcp',
+  'GroupName'   => 'default',
+  'ToPort'      => 22
+)
+
+server_data = compute.run_instances(
+  'ami-1a837773',
   1,
-  49
-).body['server']
+  1,
+  'InstanceType'  => 'm1.small',
+  'KeyName'       => 'id_rsa.pub',
+  'SecurityGroup' => 'default'
+).body['instancesSet'].first

-until compute.get_server_details(
-  server_data['id']
-).body['server']['status'] == 'ACTIVE'
+until compute.describe_instances(
+  'instance-id' => server_data['instanceId']
+).body['reservationSet'].first['instancesSet'].first['instanceState']['name'] == 'running'
 end

+sleep(300)
+
 Net::SSH.start(
-  server_data['addresses'].first,
-  'root',
-  :password => server_data['password']
+  server_data['ipAddress'],
+  'ubuntu',
+  :key_data => [File.read('~/.ssh/id_rsa')]
 ) do |ssh|
   commands = [
     %{'mkdir .ssh'},
```

geopinging v1

```
1 # specify a different provider
2 credentials = {
3   :provider              => 'AWS',
4   :aws_access_key_id     => AWS_ACCESS_KEY_ID,
5   :aws_secret_access_key => AWS_SECRET_ACCESS_KEY
6 }
7
8 server_attributes = {
9   :image_id              => 'ami-1a837773',
10  :private_key_path      => PRIVATE_KEY_PATH,
11  :public_key_path       => PUBLIC_KEY_PATH,
12  :username              => 'ubuntu'
13 }
```

geopinging v2

```
1 # specify a different aws region
2 # ['ap-southeast-1', 'eu-west-1', 'us-west-1]
3 credentials.merge!({
4   :region => 'eu-west-1'
5 })
```

geopinging v...

portable - ~~AWS~~, Bluebox, Brightbox, ~~Rackspace~~, Slicehost, Terremark, ...

lather, rinse, repeat

How?

That is awesome, but *how did you…*

exploring

```
geymus ~ ⌘ fog

  To run as 'default', add the following to ~/.fog

:default:
  :aws_access_key_id:     INTENTIONALLY_LEFT_BLANK
  :aws_secret_access_key: INTENTIONALLY_LEFT_BLANK
  :public_key_path:       INTENTIONALLY_LEFT_BLANK
  :private_key_path:      INTENTIONALLY_LEFT_BLANK
  :rackspace_api_key:     INTENTIONALLY_LEFT_BLANK
  :rackspace_username:    INTENTIONALLY_LEFT_BLANK
  ...
```

sign posts

```
geymus ~ ⌘ fog
  Welcome to fog interactive!
  :default credentials provide AWS and Rackspace
>> providers
[AWS, Rackspace]
>> Rackspace.collections
[:directories, :files, :flavors, :images, :servers]
>> Compute[:rackspace]
#<Fog::Compute::Rackspace ...>
>> Compute[:rackspace].requests
[:confirm_resized_server, ..., :update_server]
```

what are those?

provider => [AWS, Rackspace, Zerigo, ...]

service => [Compute, DNS, Storage, ...]

collection => [flavors, images, servers, ...]

model => [flavor, image, server, ...]

request => [describe_instances, run_instances, ...]

requests?

```
>> Compute[:rackspace].list_servers
#<Excon::Response:0x_______
@body = {
  "servers" => []
},
@headers = {
  "X-PURGE-KEY"=>"/______/servers",
  ...,
  "Connection"=>"keep-alive"
},
@status=200>
```

sanity check

```
>> Compute[:rackspace].servers.select {|s| s.ready?}
  <Fog::Compute::Rackspace::Servers
    filters={}
    []
  >
>> Compute[:aws].servers.select {|s| s.ready?}
  <Fog::Compute::AWS::Servers
    []
  >
>> exit
```

finding images

```
>> Compute[:rackspace].images.table([:id, :name])
  +----------+------------------------------------+
  | id       | name                               |
  +----------+------------------------------------+
  | 49       | Ubuntu 10.04 LTS (lucid)           |
  +----------+------------------------------------+
  ...
>> Compute[:aws].images # I use alestic.com listing
...
```

exploring...

It takes *forever!*

It's so expensive!

A warm welcome for *Fog.mock!*

Mocks!

```
geymus ~ ⌘ FOG_MOCK=true fog
```

or

```
require ‘fog’
Fog.mock!
```

simulation

Most functions *just work!*

Unimplemented mocks? **Errors** keep you on track.

Tests run against both, so it is either **consistent** or a **bug**.

Back to Business

I have a bunch of data *now what?*

storage - aggregating cloud data

Get Connected

```
credentials = {
  :provider              => 'AWS',
  :aws_access_key_id     => AWS_ACCESS_KEY_ID,
  :aws_secret_access_key => AWS_SECRET_ACCESS_KEY
}

# setup a connection to the service
storage = Fog::Storage.new(credentials)
```

directories

```
1 # create a directory
2 directory = storage.directories.create(
3   :key    => directory_name,
4   :public => true
5 )
```

files

```
1 # store the file
2 file = directory.files.create(
3   :body   => File.open(path),
4   :key    => name,
5   :public => true
6 )
7
8 # return the public url for the file
9 file.public_url
```

geostorage

```
1 # specify a different provider
2 credentials = {
3   :provider               => 'Rackspace',
4   :rackspace_api_key      => RACKSPACE_API_KEY,
5   :rackspace_username     => RACKSPACE_USERNAME
6 }
```

cleanup

```
geymus ~ ⌘ fog
...
>> dir = Storage[:aws].directories.get(DIRECTORY_NAME)
...
>> dir.files.each {|file| file.destroy}
...
>> dir.destroy
...
>> exit
```

geoaggregating

portable - ~~AWS~~, Google, Local, Rackspace

lather, rinse, repeat

Phase 3: Profit

I've got the data, but how do I *freemium?*

dns - make your cloud (premium) accessible

Get Connected

```
credentials = {
  :provider      => 'Zerigo',
  :zerigo_email => ZERIGO_EMAIL,
  :zerigo_token => ZERIGO_TOKEN
}

# setup a connection to the service
dns = Fog::DNS.new(credentials)
```

zones

```
1 # create a zone
2 zone = dns.zone.create(
3    :domain => domain_name,
4    :email  => "admin@#{domain_name}"
5 )
```

records

```
1 # create a record
2 record = zones.records.create(
3   :ip   => '1.2.3.4',
4   :name => "#{customer_name}.#{domain_name}",
5   :type => 'A'
6 )
```

cleanup

```
geymus ~ ⌘ fog
...
>> zone = DNS[:zerigo].zones.get(ZONE_ID)
...
>> zone.records.each {|record| record.destroy}
...
>> zone.destroy
...
>> exit
```

geofreemiuming

portable - AWS, Linode, Slicehost, ~~Zerigo~~

lather, rinse, repeat

Congratulations!

todo - copy/paste, push, deploy!

budgeting - find ways to spend your pile of money

geemus - likes coffee, bourbon, games, etc

retire - at your earliest convenience

Love!

knowledge - ~~exferins encoded in ruby~~

empowering - show the cloud who is boss

exciting - this is some cutting edge stuff!

Homework: Easy

follow *@fog* to hear about releases

follow *github.com/geemus/fog* to hear nitty gritty

proudly display *stickers* wherever hackers are found

ask *geemus* your remaining questions

play *games* with *geemus*

Homework: Normal

report issues at *github.com/geemus/fog/issues*

irc *#ruby-fog* on freenode

discuss *groups.google.com/group/ruby-fog*

write *blog posts*

give *lightning talks*

Homework: Hard

help make *fog.io* the cloud services resource for ruby

send *pull requests* fixing issues or adding features

proudly wear contributor-only *grey shirt* wherever hackers are found

Homework: Expert

help *maintain* the cloud services you depend on

become a *collaborator* by keeping informed and involved

proudly wear commit-only *black shirt* wherever hackers are found

Thanks! Questions?

(see also: *README*)

tutorial - *http://github.com/geemus/learn_fog*

examples - *http://gist.github.com/729992*

slides - *http://slidesha.re/hR8sP9*

repo - *http://github.com/geemus/fog*

bugs - *http://github.com/geemus/fog/issues*

@geemus - questions, comments, suggestions