Using Amazon's Web Services From Ruby

Jonathan Younger

Does it scale?

Amazon Web Services

- SQS - Simple Queue Service
- EC2 - Elastic Compute Cloud
- S3 - Simple Storage Service
- SimpleDB

Ruby

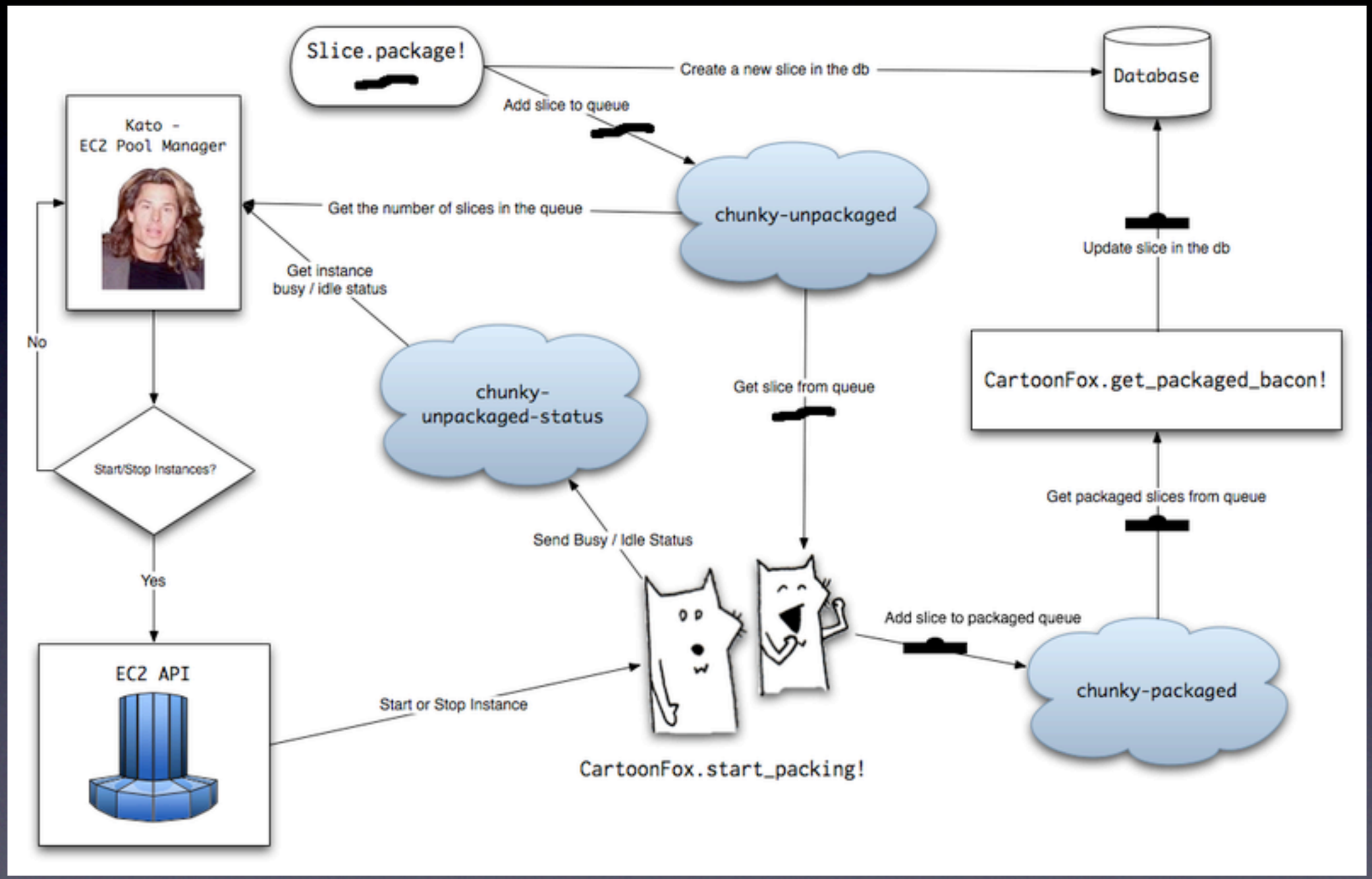
```
$ gem install right_aws
```

- RightScale RightAws::Sqs
- RightScale RightAws::Ec2

```
$ gem install kato
```

- Kato - EC2 Pool Manager

Makin' Bacon



MountainWest RubyConf 2008

RightAws::Sqs

```
require 'rubygems'
require 'right_aws'
RightAws::RightAWSParser.xml_lib = "libxml"
SQS = RightAws::Sqs.new("access_id", "access_key",
  :server => "localhost",
  :port => "4002",
  :protocol => "http"
)
```

SQS Queuing

```
queue = SQS.queue("name_of_the_queue")
queue.push "put a message in the queue"
queue.size # => 1

message = queue.receive
puts message.body # => "put a message in the queue"
# Remove the message from the queue
message.delete
```

Slice.package!

```
class Slice < Sequel::Model
  def self.package!
    slice = create(:queued_at => Time.now)
    queue = SQS.queue("chunky-unpackaged")
    queue.push(slice.id)
    slice
  end
end
```

RightAws::Ec2

```
require 'rubygems'
require 'right_aws'
RightAws::RightAWSParser.xml_lib = "libxml"
EC2 = RightAws::Ec2.new("access_id", "access_key",
  :server => "localhost",
  :port => "4002",
  :protocol => "http"
)
```

EC2 Instances

```
EC2.describe_instances # => { [instance data] }
EC2.run_instances("name_of_ami", minimum, maximum)
EC2.terminate_instances(["id_1", "id_2"])
```

Kato - Pool Manager

```
require 'rubygems'
require 'kato'

pool_supervisor = Kato::PoolSupervisor.new(config)
pool_supervisor.run
```

Kato Config

- Manage multiple pools
- Minimum number of instances
- Maximum number of instances
- Ramp up/down intervals

Fox.start_packing!

```
def start_packing!
  while $packing
    if message = unpackaged.receive
      send_status_update(true)
      packaged.push message.body
      message.delete
      send_status_update(false)
    end
    sleep 2
  end
end
```

Fox.get_packaged_bacon!

```
def self.get_packaged_bacon!
  packaged = SQS.queue("chunky-packaged")
  loop do
    if( message = packaged.receive) &&
    (slice = Slice[message.body.to_i])

    slice.set(:packaged_at => Time.now)
    message.delete
  end
  sleep 0.5
 end
end
```

Demo

Resources

- http://github.com/daikini/chunky_bacon
- http://aws.amazon.com
- http://rightaws.rubyforge.org