Resque and Redis

https://github.com/igal/resque_and_redis_eg

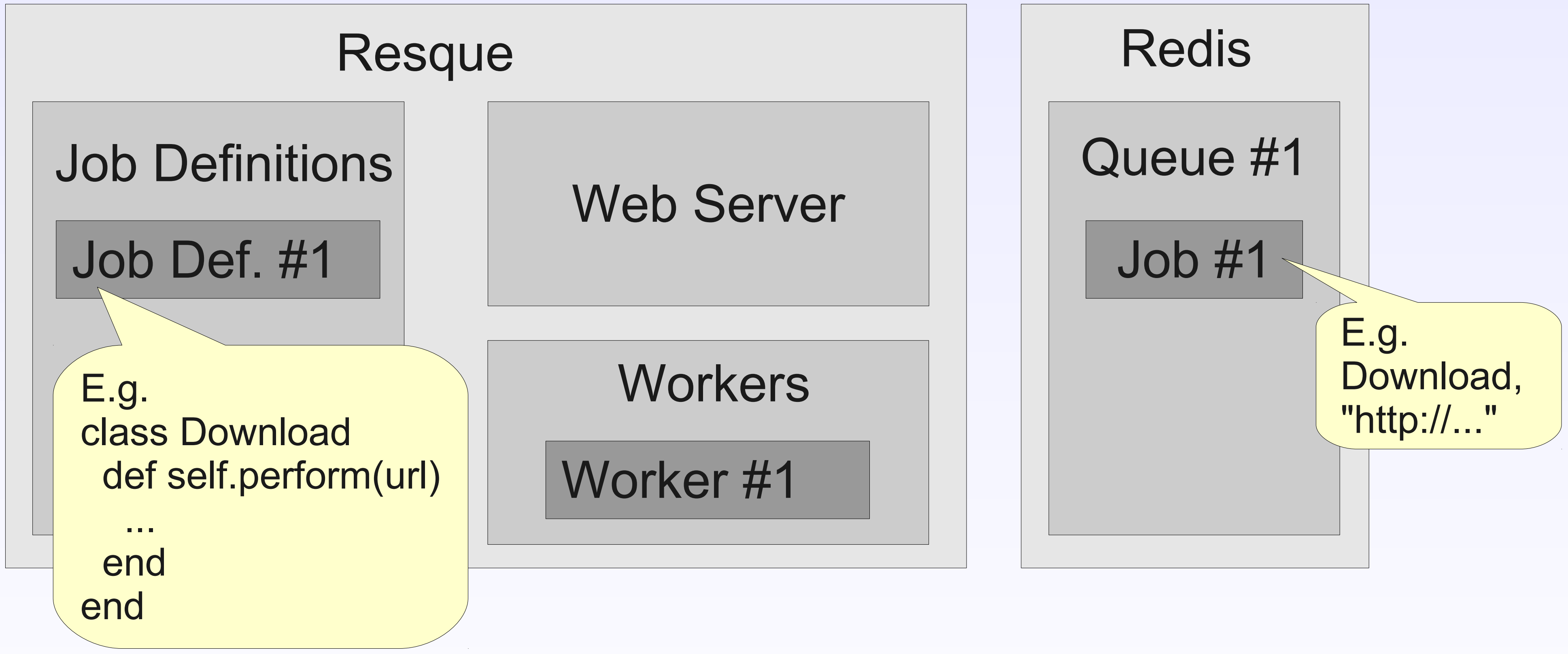
Igal Koshevoy, Pragmaticraft
Business-Technology Consultant
igal@pragmaticraft.com

Resque

"Resque is a Redis-backed Ruby library for creating background jobs, placing those jobs on multiple queues, and processing them later."

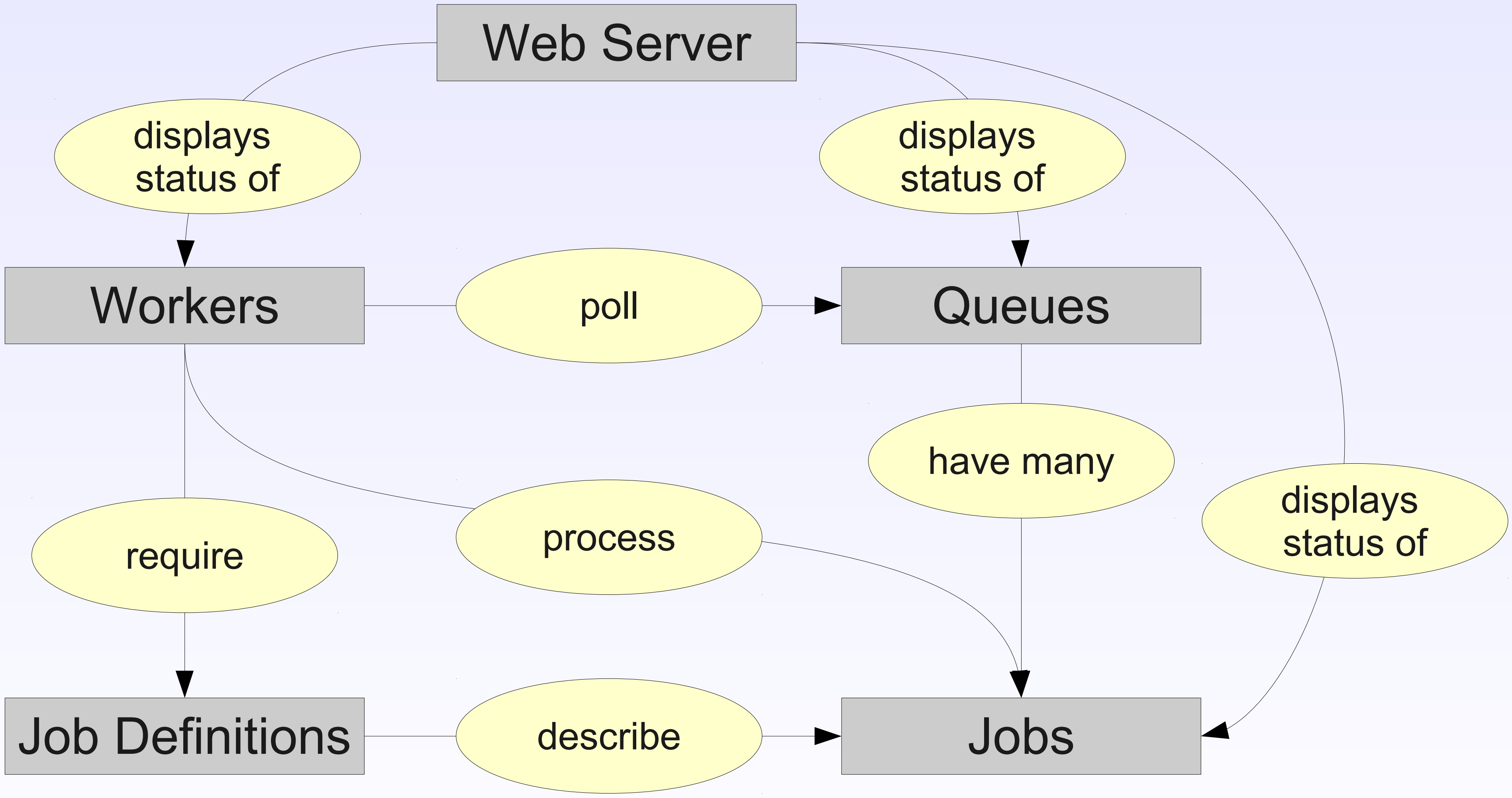
https://github.com/defunkt/resque

Architecture



```
E.g.
class Download
  def self.perform(url)
    ...
  end
end
```

Architecture



Defining simple jobs

```
require 'logger'

class Greeter
  LOG = Logger.new('log/greeter.log')

  @queue = 'greeter'

  def self.perform(entity)
    LOG.info("Hello #{entity}")
  end
end
```

Defining jobs that create jobs

```
require 'logger'
require 'rubygems'
require 'resque'

class Ping
  LOG = Logger.new('log/ping.log')

  @queue = 'pingpong'

  def self.perform(counter=0)
    sleep 1
    LOG.info("Ping #{counter}")
    Resque.enqueue(Pong, counter+1)
  end
end
```

```
class Pong
  LOG = Logger.new('log/pong.log')

  @queue = 'pingpong'

  def self.perform(counter=0)
    sleep 1
    LOG.info("Pong #{counter}")
    Resque.enqueue(Ping, counter+1)
  end
end
```

Running jobs with workers

```
require 'lib/greeter'
require 'lib/pingpong'

require 'rubygems'
require 'resque'
require 'resque/tasks'

desc "Run a worker"
task :work do
  ENV['VERBOSE'] = '1' unless ENV['VERBOSE']
  ENV['QUEUE'] = '*' unless ENV['QUEUE']
  Rake::Task['resque:work'].invoke
end

desc 'Run workers as threads'
task :workers do
  ENV['VERBOSE'] = '1' unless ENV['VERBOSE']
  ENV['QUEUE'] = '*' unless ENV['QUEUE']
  ENV['COUNT'] = '10' unless ENV['COUNT']
  Rake::Task['resque:workers'].invoke
end
```

Displaying state with web server

Pros

- Easy to learn
- Easy to setup
- Easy to use
- Lightweight compared to database polling
- Useful state display
- Useful error handling and retry

Cons

- WTF: Can't reserve jobs – they'll be lost if the machine running them crashes or disconnects
- No central control, all workers are independent
- Can't mix worker implementations
- Job and its arguments must be serialized to at most 1MB of JSON
- Poor design and coding style: poor separation of concerns, no model classes, do-everything helper class, excessively clever views, code duplication between views and helpers, many unnecessary operations, poor internal documentation, etc.

Alternatives

- Queues and Ruby-specific workers
  - DJ (DelayedJob): relies on ActiveRecord
- Queues with multi-language workers
  - beanstalkd: easy, simple and fast
  - Qpid
  - RabbitMQ
  - ActiveMQ
- Low-level
  - ZeroMQ + Google Protocol Buffers

Redis

"Redis is an open source, advanced key-value store. It is often referred to as a data structure server since keys can contain strings, hashes, lists, sets and sorted sets."

http://redis.io/

Examples

github.com/igal/resque_and_redis_eg

- See code in "lib/redis_eg.rb"
- Run code with "rake redis_eg"

Pros

- Easy setup
- Fast and efficient
- Support for basic structured data
- Multi-action atomic operations
- Publish/subscribe operations
- Blocking pop operations
- Simple replication

Cons

- WTF: No reservations or other complex transactions (e.g. begin transaction, pop value, process it, push value, commit transaction)
- WTF: Unreliable persistence
- Everything must fit into memory, else use experimental and slow virtual memory
- Missing operations: no unique identifier generator, no cursors for efficient pagination, can't remove item by list index, no alternatives to RPOPLPUSH, etc.
- No built-in clustering, sharding or failover

Questions & Answers

https://github.com/igal/resque_and_redis_eg

Igal Koshevoy
Biz-Tech Consultant
igal@pragmaticraft.com
@igalko