Job Queues with Gearman

Michael Moussa

About Me

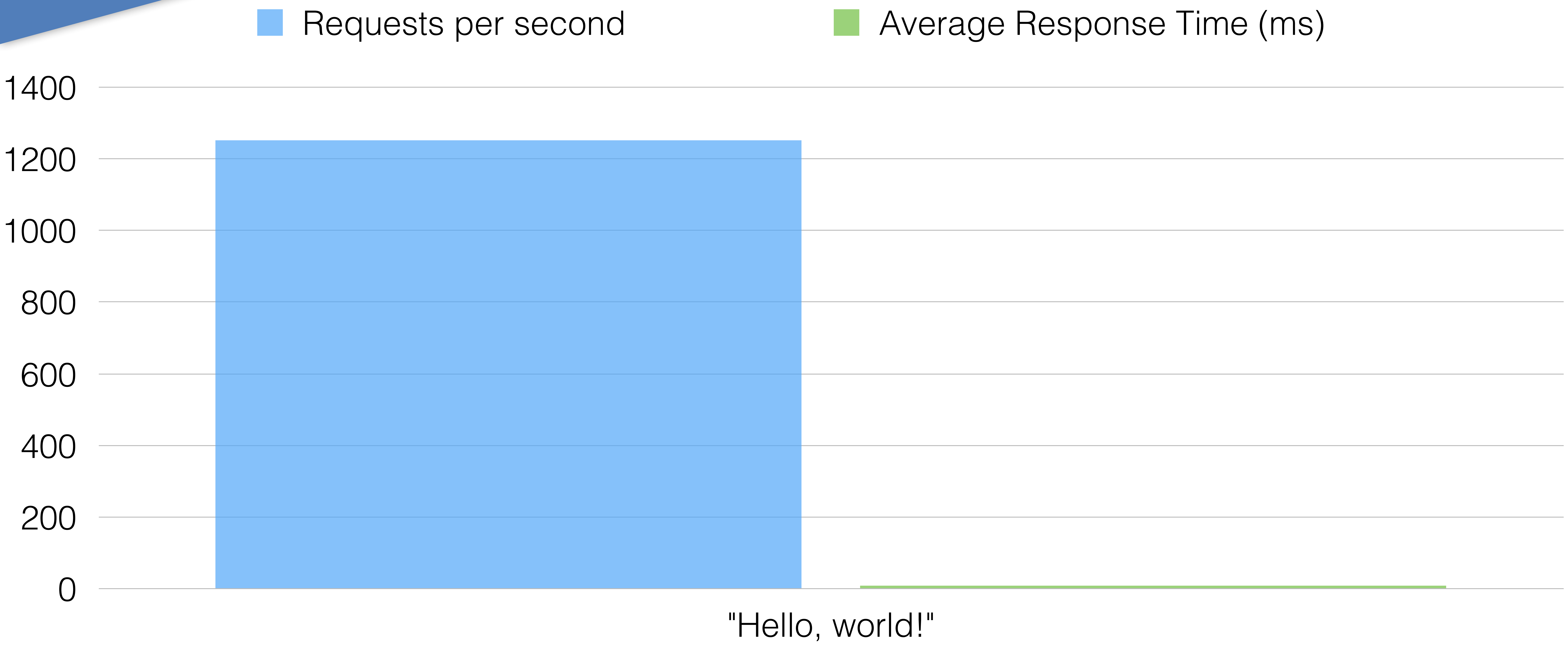
- Web application developer for 17 years
- Solutions Architect at rackspace® the #1 managed cloud company
- Zend Expressive maintainer & general open source contributor

Why job queues?

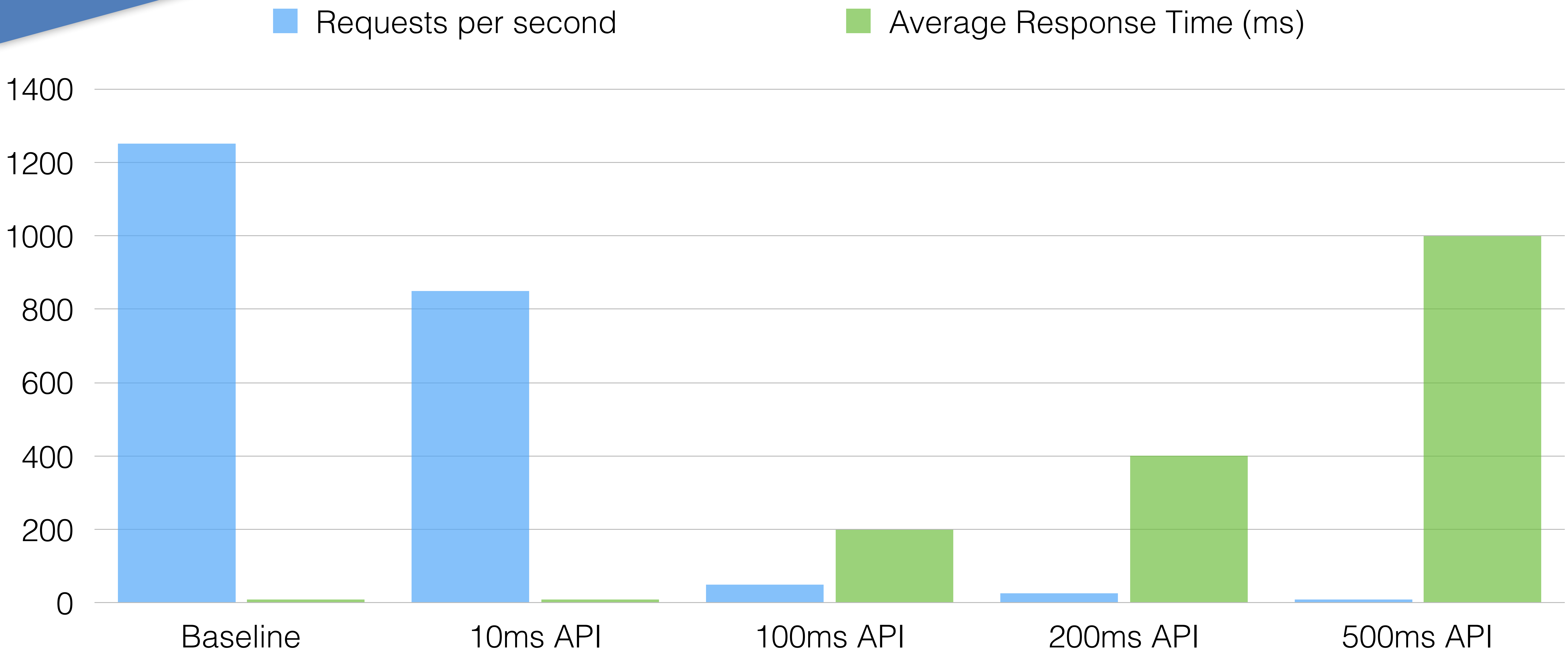
Benchmarks

- 1 CPU / 512MB RAM droplets running CentOS 7
- PHP 7.0, NGINX, php-fpm
- Siege (https://github.com/JoeDog/siege)
  - `siege -b -t 180s -c 10 http://example.com/example.php`
- Separate droplets for Siege / web / Gearman workers
- Don't worry too much about the specific numbers

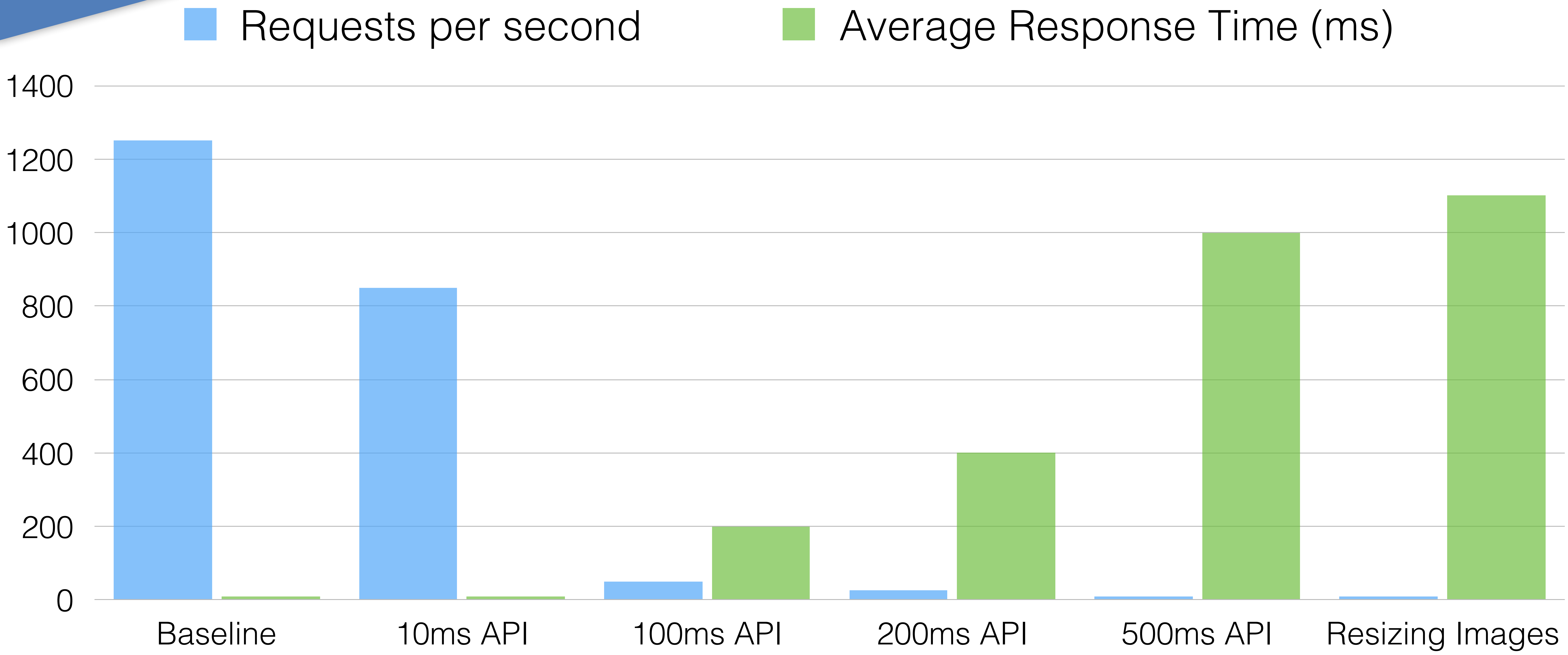
```
echo 'Hello, world!';
```


Requests per second
Average Response Time (ms)
1400
1200
1000
800
600
400
200
0
"Hello, world!"

```php
$client = new \GuzzleHttp\Client();
$response = $client->post(
    'https://api.example.com/something',
    ['json' => ['foo' => 'bar']]
);

echo (string) $response->getBody();
```


Requests per second
Average Response Time (ms)
1400
1200
1000
800
600
400
200
0
Baseline
10ms API
100ms API
200ms API
500ms API

```php
$imagePath = '/path/to/some-image.jpg';
list($width, $height) = getimagesize($imagePath);
$newWidth = $width * 0.5;
$newHeight = $height * 0.5;
$resizedImage = imagecreatetruecolor($newWidth, $newHeight);

imagecopyresized(
    $resizedImage, imagecreatefromjpeg($imagePath), 0, 0, 0, 0,
    $newWidth, $newWidth, $width, $height
);

imagejpeg($resizedImage, tempnam(__DIR__ . '/resized/', ''));
```

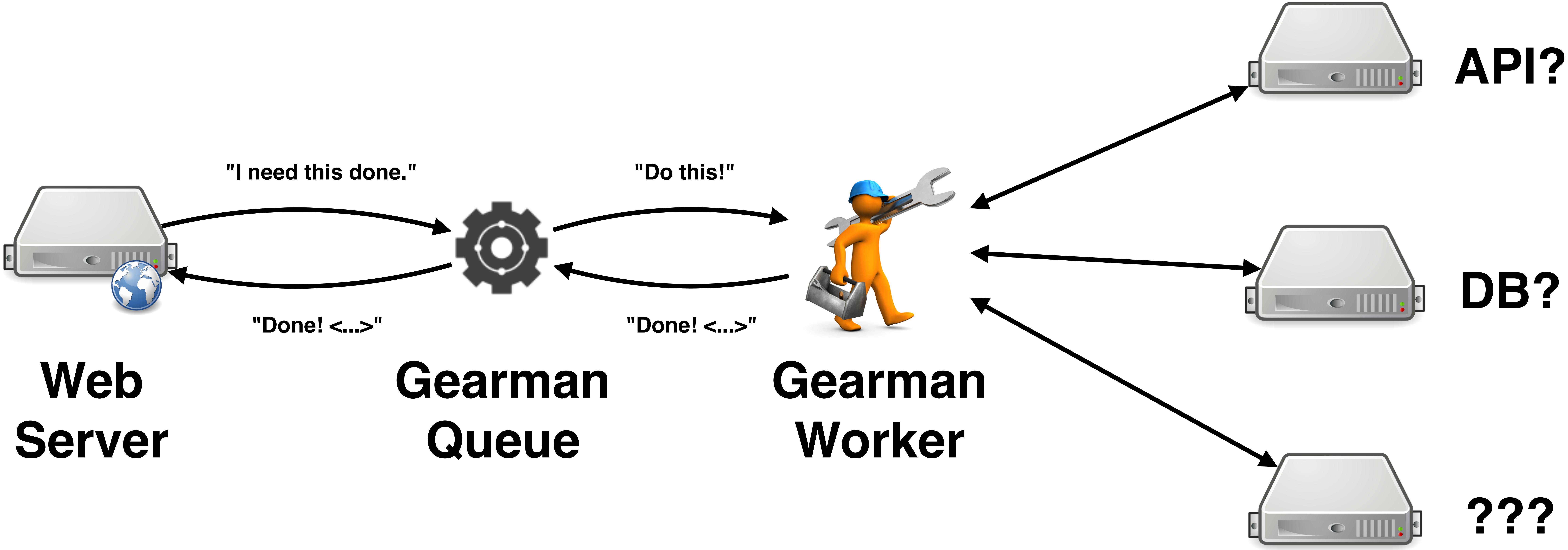

Requests per second
Average Response Time (ms)
1400
1200
1000
800
600
400
200
0
Baseline
10ms API
100ms API
200ms API
500ms API
Resizing Images

Gearman

"Gearman is a generic application framework for farming out work to other machines or processes."

https://php.net/manual/en/intro.gearman.php

http://gearman.org/getting-started/


"I need this done."
"Do this!"
"Done! <...>"
"Done! <...>"
Web Server
Gearman Queue
Gearman Worker
API?
DB?
???

- Workers don't read jobs from the server - the server assigns them to workers
    - No duplicate / simultaneous handling of the same job
    - Queue knows when a job is done, so you don't have to remove it
- Your worker(s) don't have to be written in PHP
- Jobs can be synchronous or asynchronous
- Jobs can be a group of tasks meant to be run in parallel

Installation

Gearman Server

Ubuntu: `apt-get install gearman-job-server`

RHEL / CentOS: `yum install gearmand`

PHP Extension

PHP 5.6: `pecl install gearman`

PHP 7.0 Ubuntu:

```
add-apt-repository -y ppa:ondrej/php
apt-get update -y
apt-get install -y php-gearman
```

PHP 7.0 RHEL / CentOS: `yum install php70-php-gearman`

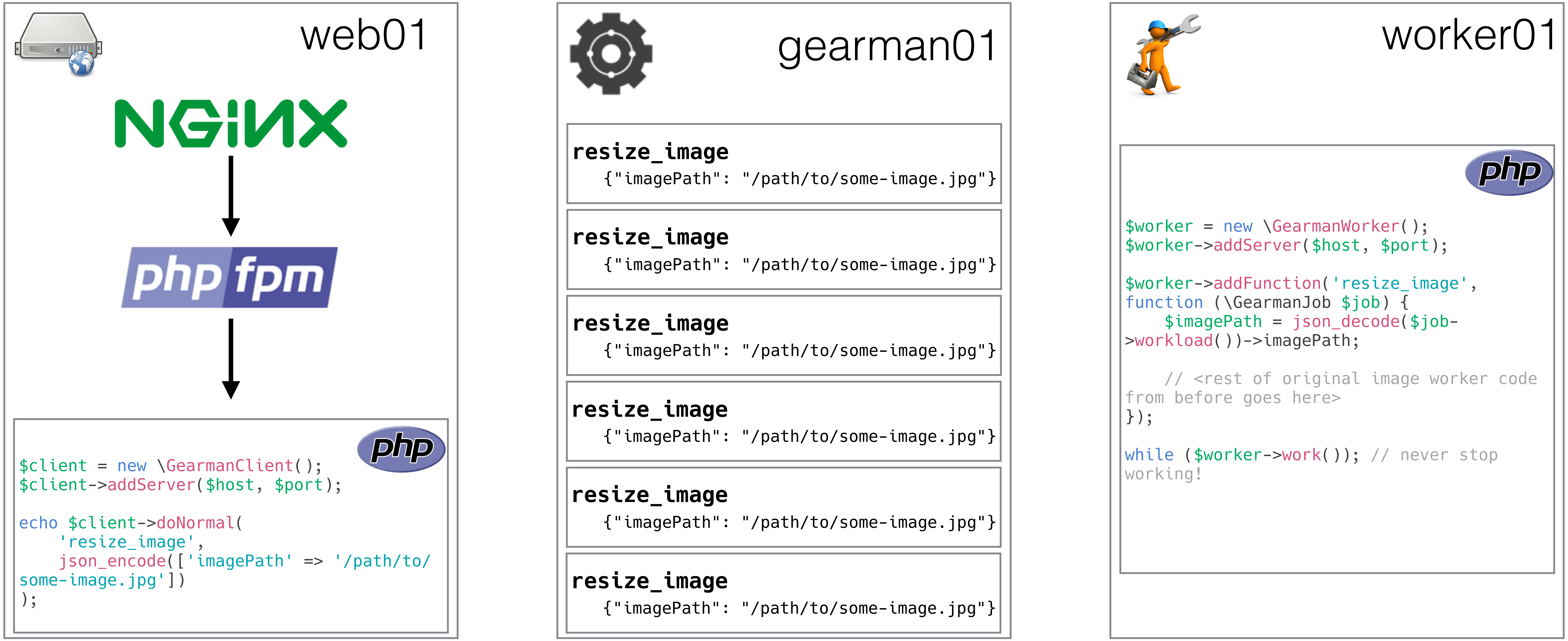
- https://rpms.remirepo.net/wizard/

GearmanWorker

```
$worker = new \GearmanWorker();
$worker->addServer($host, $port);

$worker->addFunction('resize_image', function (\GearmanJob $job)
{
    $imagePath = json_decode($job->workload())->imagePath;

    // <rest of original image worker code from before goes here>
});

while ($worker->work()); // never stop working!
```

GearmanClient

```
$client = new \GearmanClient();
$client->addServer($host, $port);

$client->doNormal(
    'resize_image',
    json_encode(['imagePath' => '/path/to/some-image.jpg'])
);
```



web01

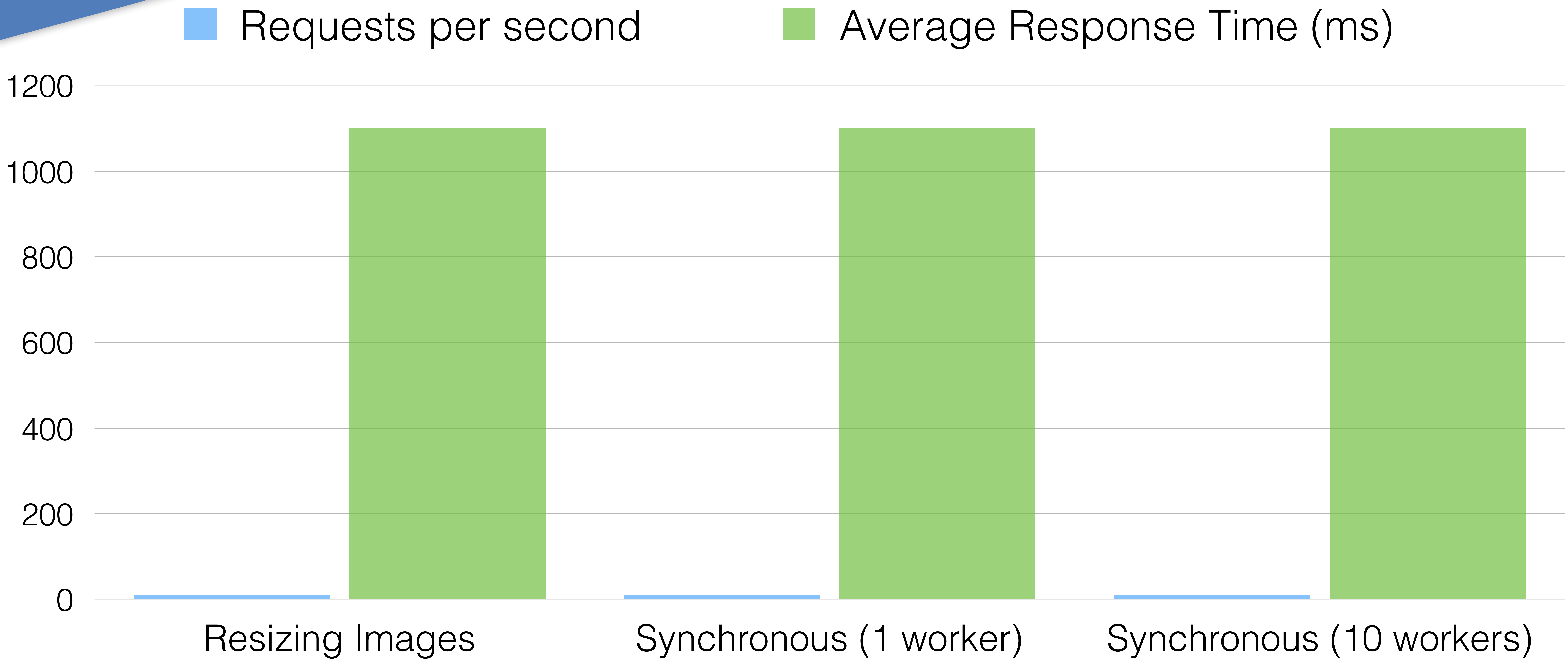
```
$client = new \GearmanClient();
$client->addServer($host, $port);

echo $client->doNormal(
    'resize_image',
    json_encode(['imagePath' => '/path/to/
some-image.jpg'])
);
```

gearman01

```
resize_image
   {"imagePath": "/path/to/some-image.jpg"}
resize_image
   {"imagePath": "/path/to/some-image.jpg"}
resize_image
   {"imagePath": "/path/to/some-image.jpg"}
resize_image
   {"imagePath": "/path/to/some-image.jpg"}
resize_image
   {"imagePath": "/path/to/some-image.jpg"}
resize_image
   {"imagePath": "/path/to/some-image.jpg"}
```

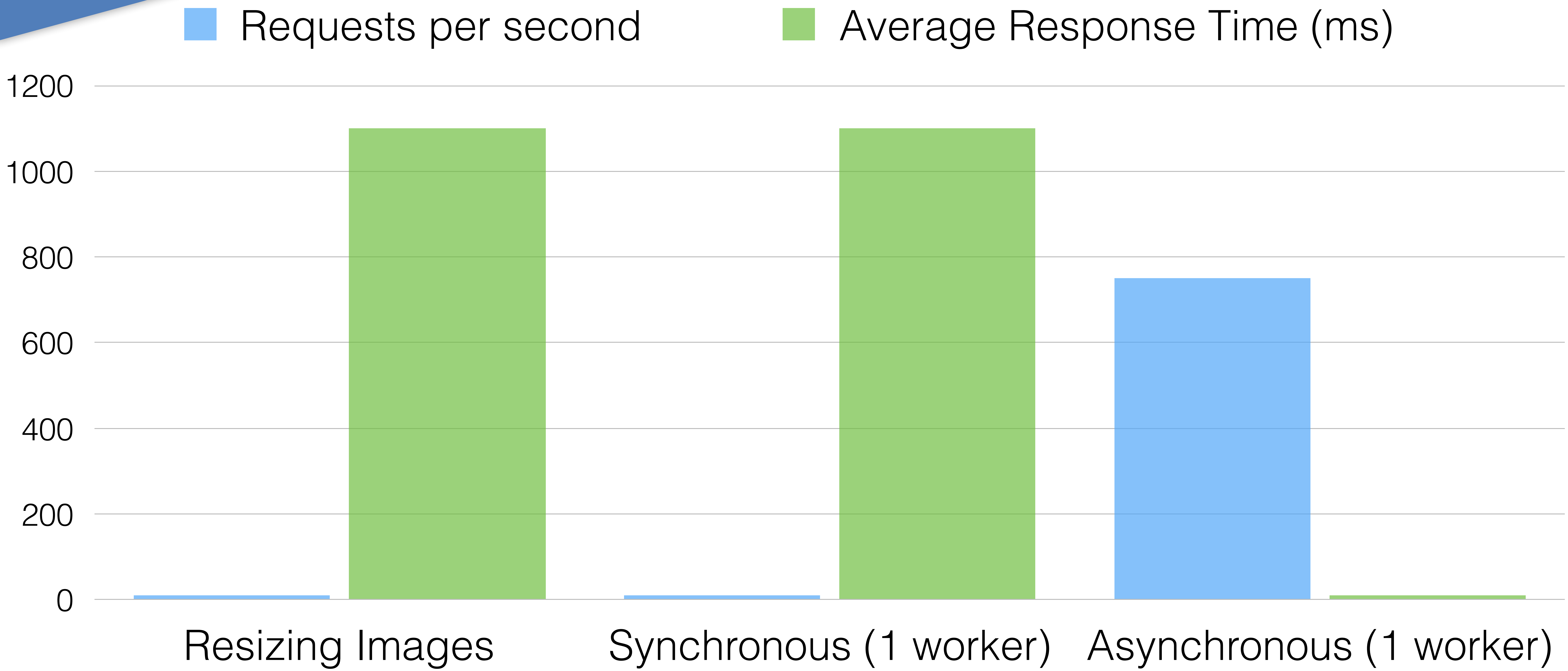
worker01

```
$worker = new \GearmanWorker();
$worker->addServer($host, $port);

$worker->addFunction('resize_image',
function (\GearmanJob $job) {
    $imagePath = json_decode($job-
>workload())->imagePath;

    // <rest of original image worker code
from before goes here>
});

while ($worker->work()); // never stop
working!
```


🟦 Requests per second
🟩 Average Response Time (ms)
1200
1000
800
600
400
200
0
Resizing Images
Synchronous (1 worker)
Synchronous (10 workers)

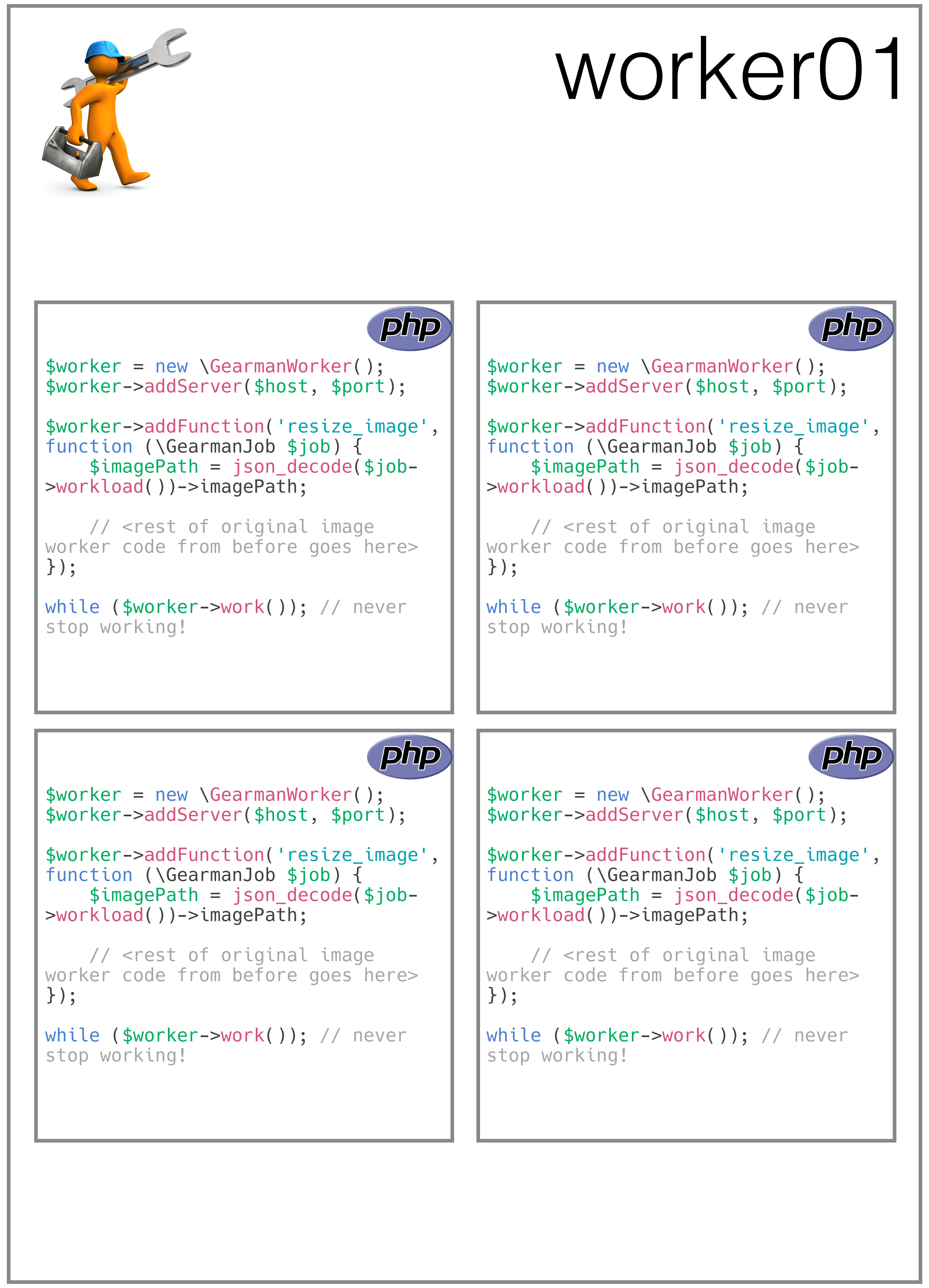
GearmanClient

```
$client = new \GearmanClient();
$client->addServer($host, $port);

~~$client->doNormal(~~
$client->doBackground(
    'resize_image',
    json_encode(['imagePath' => '/path/to/some-image.jpg'])
);
```


Requests per second
Average Response Time (ms)
1200
1000
800
600
400
200
0
Resizing Images
Synchronous (1 worker)
Asynchronous (1 worker)

worker01

```
$worker = new \GearmanWorker();
$worker->addServer($host, $port);

$worker->addFunction('resize_image',
function (\GearmanJob $job) {
    $imagePath = json_decode($job-
>workload())->imagePath;

    // <rest of original image worker code
from before goes here>
});

while ($worker->work()); // never stop
working!
```



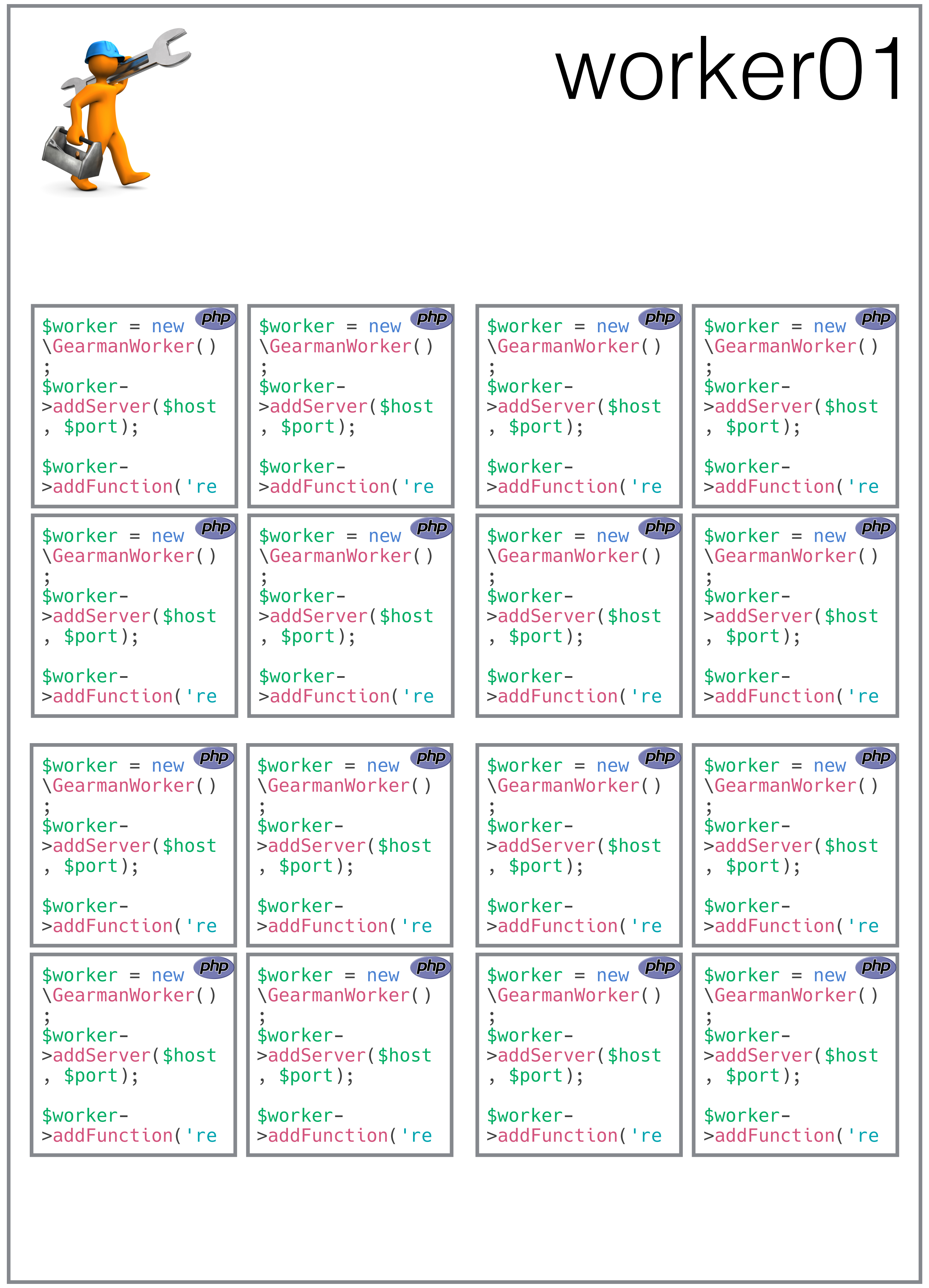
worker01

```php
$worker = new \GearmanWorker();
$worker->addServer($host, $port);

$worker->addFunction('resize_image',
function (\GearmanJob $job) {
    $imagePath = json_decode($job-
>workload())->imagePath;

    // <rest of original image
worker code from before goes here>
});

while ($worker->work()); // never
stop working!
```

```php
$worker = new \GearmanWorker();
$worker->addServer($host, $port);

$worker->addFunction('resize_image',
function (\GearmanJob $job) {
    $imagePath = json_decode($job-
>workload())->imagePath;

    // <rest of original image
worker code from before goes here>
});

while ($worker->work()); // never
stop working!
```

```php
$worker = new \GearmanWorker();
$worker->addServer($host, $port);

$worker->addFunction('resize_image',
function (\GearmanJob $job) {
    $imagePath = json_decode($job-
>workload())->imagePath;

    // <rest of original image
worker code from before goes here>
});

while ($worker->work()); // never
stop working!
```

```php
$worker = new \GearmanWorker();
$worker->addServer($host, $port);

$worker->addFunction('resize_image',
function (\GearmanJob $job) {
    $imagePath = json_decode($job-
>workload())->imagePath;

    // <rest of original image
worker code from before goes here>
});

while ($worker->work()); // never
stop working!
```



worker01

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

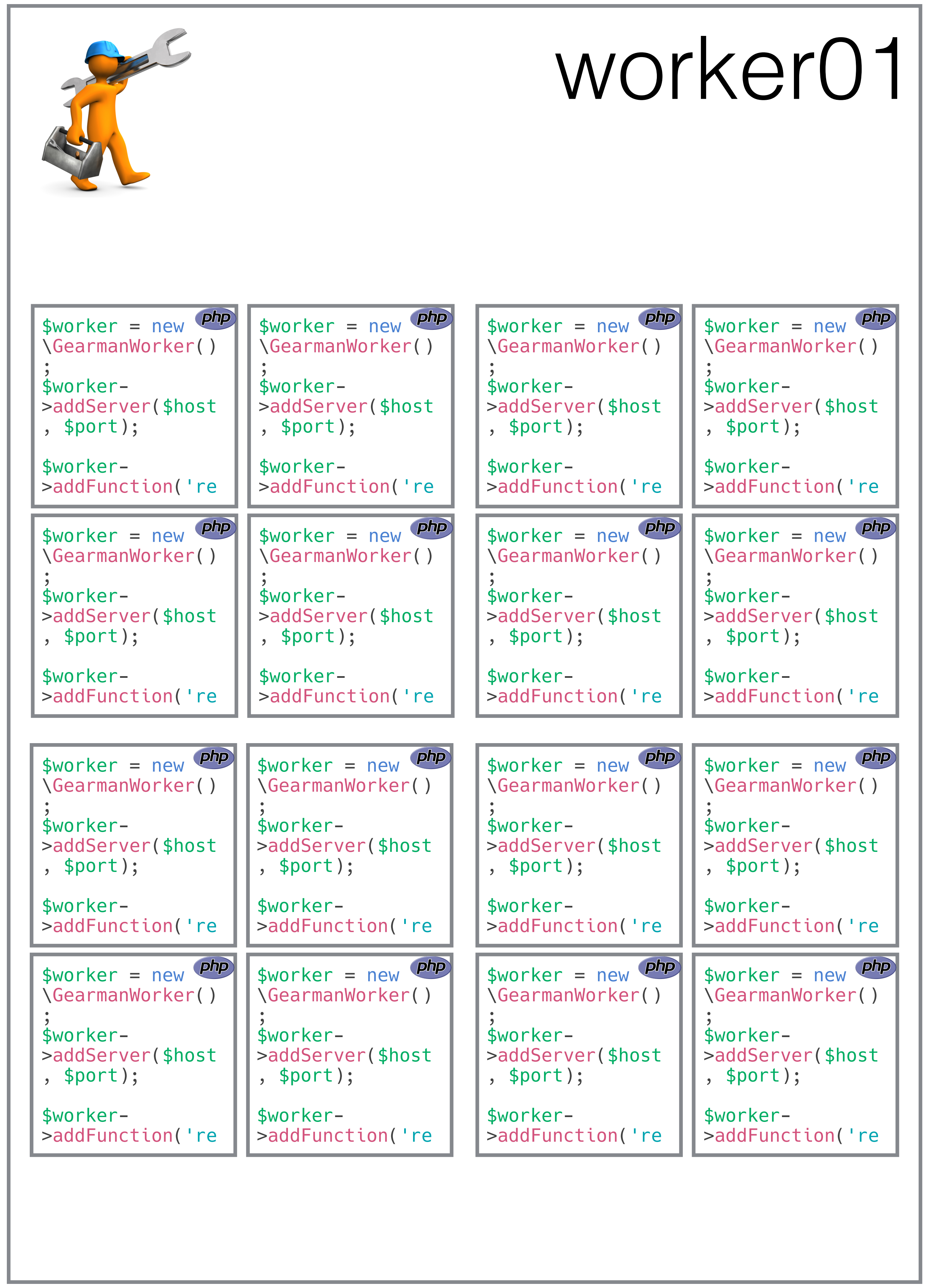
```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```



worker01

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

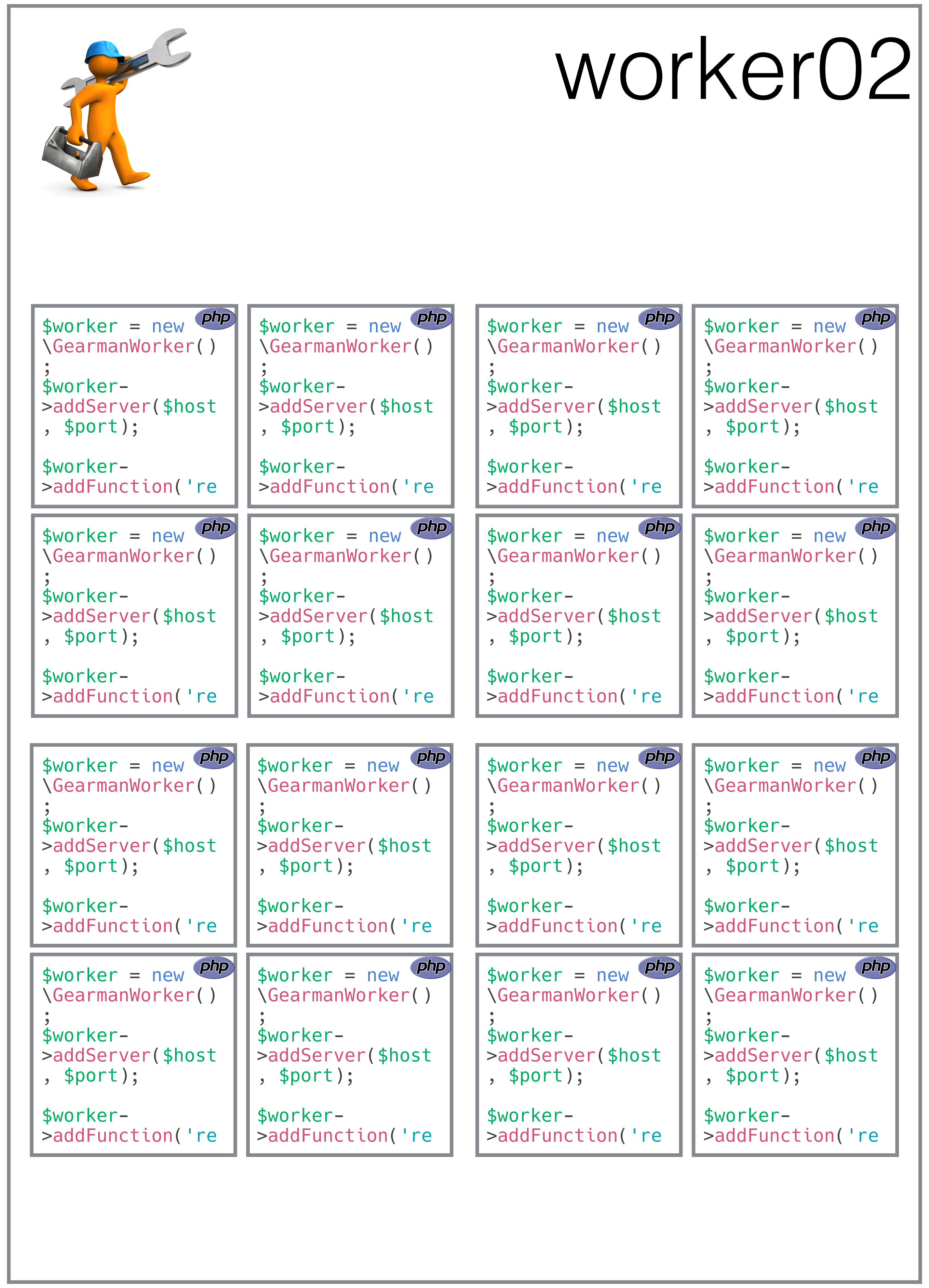
```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```



worker02

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

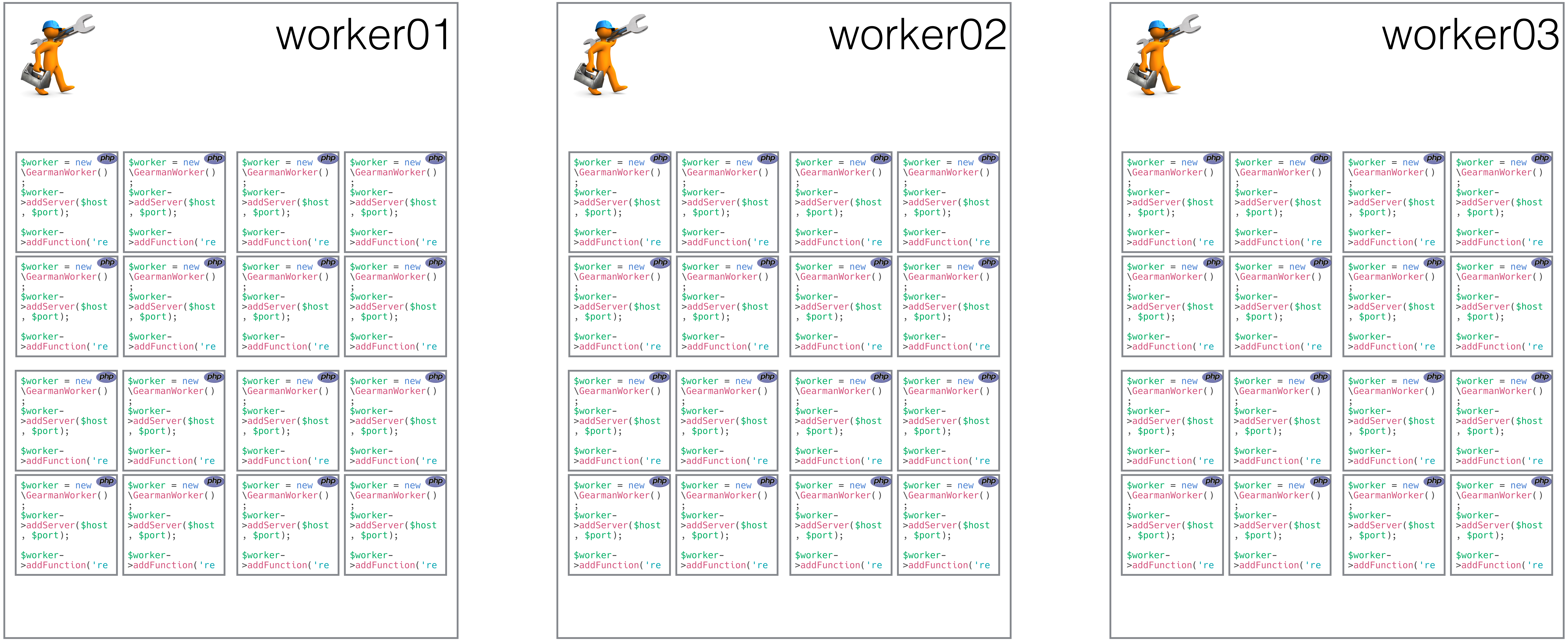
```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```



worker01

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

worker02

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

worker03

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

```
$worker = new
\GearmanWorker()
;
$worker-
>addServer($host
, $port);

$worker-
>addFunction('re
```

Worker management

Supervisor
hang on to your processes

"Supervisor is a client/server system that allows its users to monitor and control a number of processes on UNIX-like operating systems."

http://supervisord.org

Supervisor

```
# /etc/supervisord.d/resize-image-workers.ini
[program:resize-images-worker]
command=/usr/bin/php70 /path/to/image-resize-worker.php
process_name=%(program_name)s_%(process_num)02d
numprocs=5
```

```
$ ps -ef | grep image-resize-worker
root      2662  2661  0 14:32 ?        00:00:00 /usr/bin/php70 /path/to/image-resize-worker.php
root      2663  2661  0 14:32 ?        00:00:00 /usr/bin/php70 /path/to/image-resize-worker.php
root      2664  2661  0 14:32 ?        00:00:00 /usr/bin/php70 /path/to/image-resize-worker.php
root      2665  2661  0 14:32 ?        00:00:00 /usr/bin/php70 /path/to/image-resize-worker.php
root      2666  2661  0 14:32 ?        00:00:00 /usr/bin/php70 /path/to/image-resize-worker.php
```

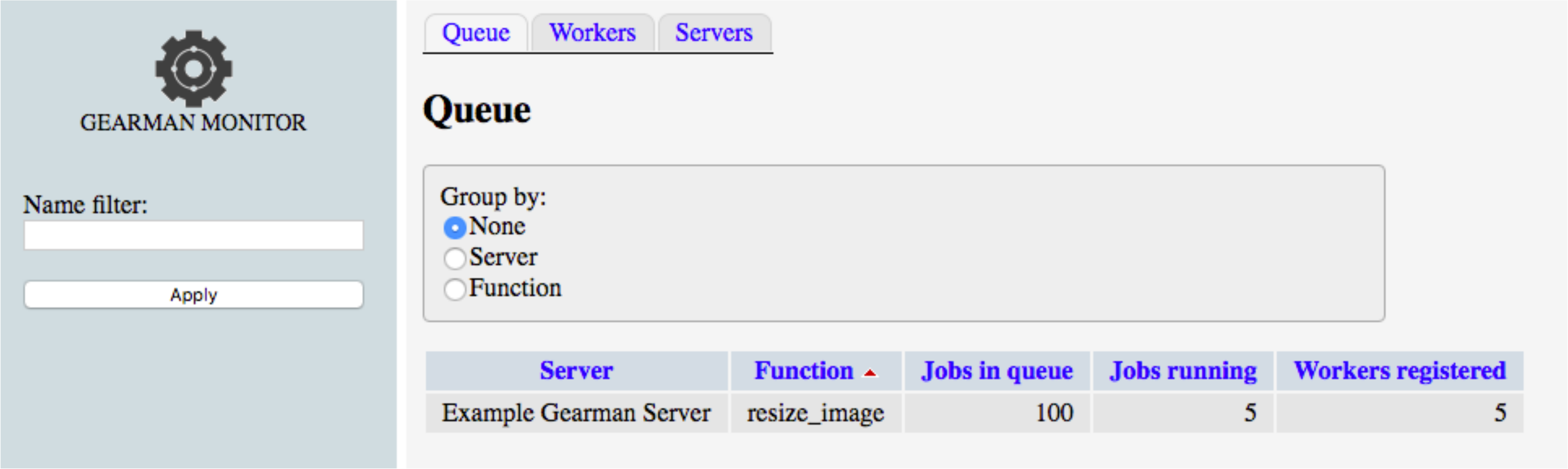
Queue management

```
$ gearadmin --status
resize_image  100  5  5
```

- **resize_image** - the function name
- **100** - number of jobs in queue
- **5** - number of jobs currently running
- **5** - number of capable workers

Gearman Monitor

https://github.com/yugene/Gearman-Monitor



Name filter:

Queue

Group by:
● None
○ Server
○ Function

Server	Function ▲	Jobs in queue	Jobs running	Workers registered
Example Gearman Server	resize_image	100	5	5

GearmanTask

```
$client = new \GearmanClient();
$client->addServer($host, $port);

foreach ([0.25, 0.5, 0.75 as $size]) {
    $client->addTask(
        'resize_image',
        json_encode([
            'imagePath' => '/path/to/some-image.jpg',
            'size' => $size
        ])
    );
}

$client->runTasks();
```

Task Priorities

	Foreground	Background
Low	`addTaskLow(...)`	`addTaskLowBackground(...)`
Normal	`addTask(...)`	`addTaskBackground(...)`
High	`addTaskHigh(...)`	`addTaskHighBackground(...)`

Job Priorities

	Foreground	Background
Low	`doLow(...)`	`doLowBackground(...)`
Normal	`doNormal(...)`	`doBackground(...)`
High	`doHigh(...)`	`doHighBackground(...)`

Communication

CLIENT

```
$client->set<__________>Callback(function (\GearmanTask $task) {})
```

- Complete, Created, Data, Exception, Fail, Status, Warning, Workload

WORKER

```
$job->send<__________>()
```

- Complete, Data, Exception, Fail, Status, Warning

```php
// worker.php

$worker->addFunction(
    'resize_image',
    function (\GearmanJob $job) {
        // do resizing here

        return $job->sendComplete(
            '/path/to/resized-image<#>'
          );
    }
);
```

```php
// client.php

$newImagePaths = [];

$client->setCompleteCallback(function (\GearmanTask $task) use (&$newImagePaths) {
    $newImagePaths[] = $task->data();
});

$client->addTask('resize_image', ...);
$client->addTask('resize_image', ...);
$client->addTask('resize_image', ...);

$client->runTasks();

var_dump($newImagePaths);
```

```
// client.php

$newImagePaths = [];

$client->setCompleteCallback(function (\GearmanTask $task) use (&$newImagePaths) {
    $newImagePaths[] = $task->data();
});

$client->addTask('resize_image', ...);
$client->addTask('resize_image', ...);
$client->addTask('resize_image', ...);

$client->runTasks();

var_dump($newImagePaths);
```

```
array(3) {
  [0]=>
  string(23) "/path/to/resized-image1"
  [1]=>
  string(23) "/path/to/resized-image2"
  [2]=>
  string(23) "/path/to/resized-image3"
}
```

Tips / Tricks / Pitfalls / Etc

$unique

```
$unique = md5($functionName . '|' . $workload));

$client->doNormal($functionName, $workload, $unique);
```

- No matter how many separate clients queue the same workload, the `$unique` identifier ensures there will only be one worked on at a time
- Foreground jobs will wait for the result from the currently active worker
- Background jobs will be discarded

Persistence

Persistence

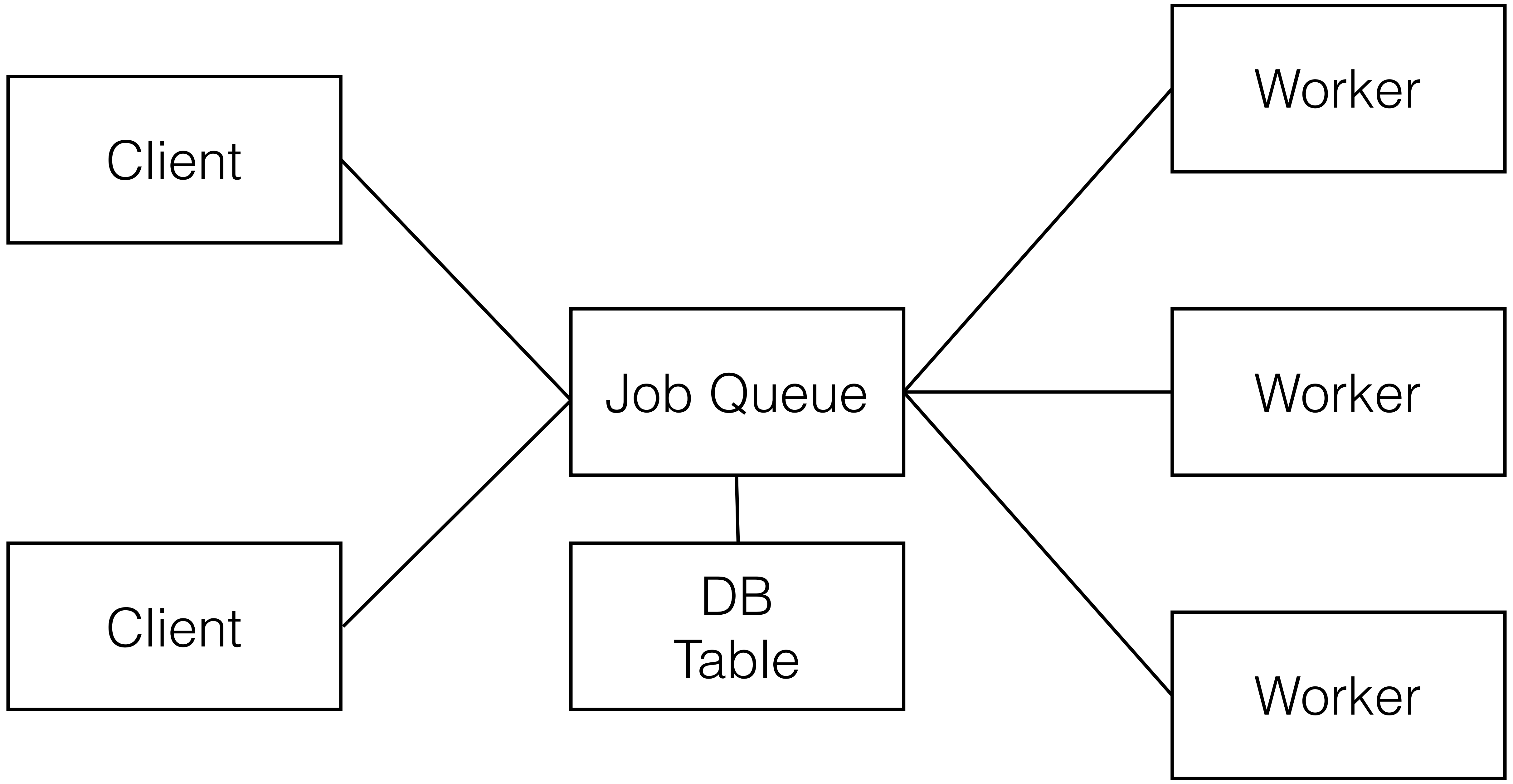
```
CREATE TABLE `gearman_queue` (
  `unique_key` varchar(64) DEFAULT NULL,
  `function_name` varchar(255) DEFAULT NULL,
  `priority` int(11) DEFAULT NULL,
  `data` longblob,
  `when_to_run` bigint(20) DEFAULT NULL,
  UNIQUE KEY `unique_key` (
    `unique_key`,
    `function_name`
  )
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Persistence

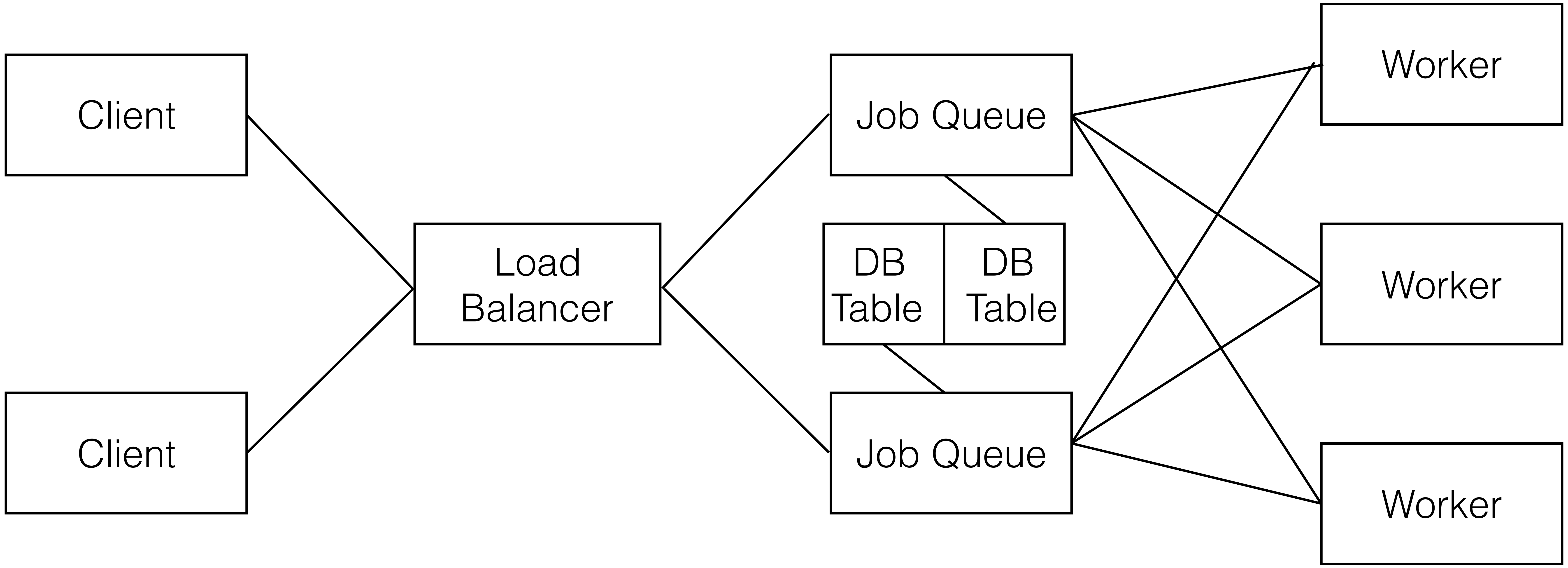
```sql
CREATE TABLE `gearman_queue` (
  `unique_key` varchar(64) DEFAULT NULL,
  `function_name` varchar(255) DEFAULT NULL,
  `priority` int(11) DEFAULT NULL,
  `data` longblob,
  `when_to_run` bigint(20) DEFAULT NULL,
  UNIQUE KEY `unique_key` (
    `unique_key`,
    `function_name`
  )
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

```
--queue-type=MySQL \
--mysql-host=... \
--mysql-port=3306 \
--mysql-user=gearman \
--mysql-password=password \
--mysql-db=gearman \
--mysql-table=gearman_queue
```

High availability



High availability



Job size

"Gearman supports single messages up to 4GB in size!"

Job size

~~"Gearman supports single messages up to 4GB in size!"~~

Please, no.

Job size

~~"Gearman supports single messages up to 4GB in size!"~~

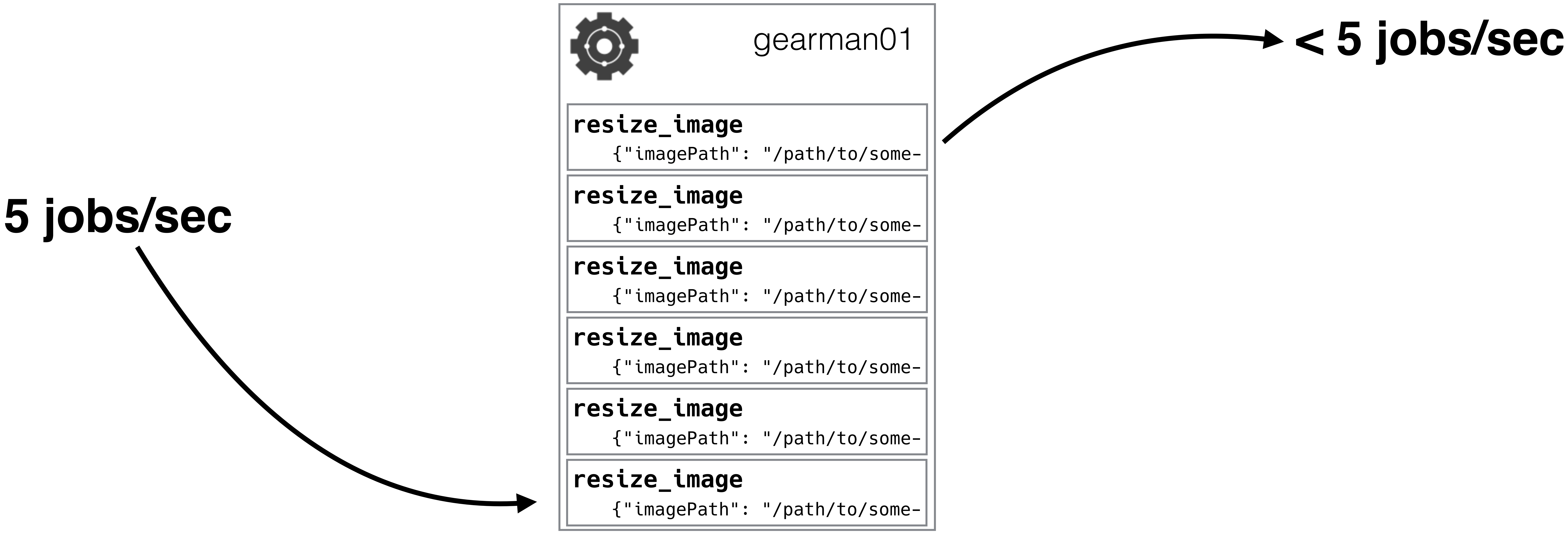
Please, no.

```
$client->doBackground(
    'do_stuff_with_huge_file',
    json_encode(['s3_url' => 'https://my-bucket.s3.amazonaws.com/
data.txt?AWSAccessKeyId=[...]&Expires=[...]&Signature=[...]'])
);
```

Capacity planning

- How many worker servers do you need?
- How many workers on each one?
- How do you know when you need more?

Do I need more?



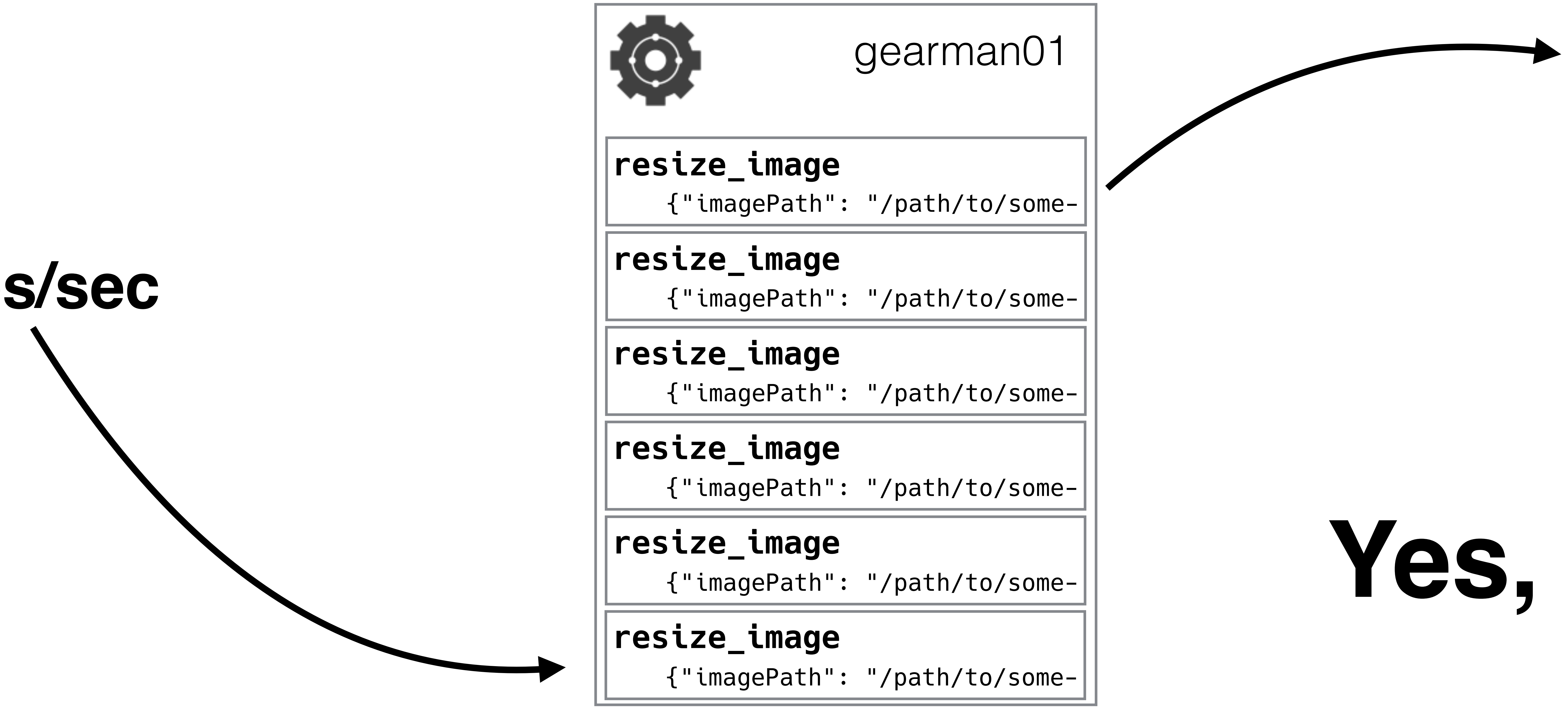
5 jobs/sec

gearman01

```
resize_image
    {"imagePath": "/path/to/some-
resize_image
    {"imagePath": "/path/to/some-
resize_image
    {"imagePath": "/path/to/some-
resize_image
    {"imagePath": "/path/to/some-
resize_image
    {"imagePath": "/path/to/some-
resize_image
    {"imagePath": "/path/to/some-
```

< 5 jobs/sec

Do I need more?



5 jobs/sec

gearman01

```
resize_image
    {"imagePath": "/path/to/some-
resize_image
    {"imagePath": "/path/to/some-
resize_image
    {"imagePath": "/path/to/some-
resize_image
    {"imagePath": "/path/to/some-
resize_image
    {"imagePath": "/path/to/some-
resize_image
    {"imagePath": "/path/to/some-
```

< 5 jobs/sec

Yes, you do!

How many workers?

How many workers?

"It's complicated"

How many workers?

"It's complicated"

- Memory

How many workers?

"It's complicated"

- Memory
- Network I/O

How many workers?

"It's complicated"

- Memory
- Network I/O
- Disk I/O

How many workers?

"It's complicated"

- Memory
- Network I/O
- Disk I/O
- CPU

How many workers?

"It's complicated"

- Memory
- Network I/O
- Disk I/O
- CPU
- Load Average

How many servers?

How many servers?

"More than you need!"

How many servers?

"More than you need!"

- Redundancy

How many servers?

"More than you need!"

- Redundancy
- Peak loads

How many servers?

"More than you need!"

- Redundancy
- Peak loads
- Unexpected new peaks

How many servers?

"More than you need!"

- Redundancy
- Peak loads
- Unexpected new peaks
- Deployment restarts

Who's first?

```
$client->doLowBackground('resize_user_avatar', '...');
$client->doHighBackground('send_welcome_email', '...');
$client->doBackground('other_stuff', '...');
```

Who's first?

```
$client->doLowBackground('resize_user_avatar', '...');
$client->doHighBackground('send_welcome_email', '...');
$client->doBackground('other_stuff', '...');
```

"It depends"

Order matters!

Competing jobs will be serviced in the order the worker functions were added

```
$worker->addFunction('resize_user_avatar', '...');
$worker->addFunction('send_welcome_email', '...');
$worker->addFunction('other_stuff', '...');
```

vs

```
$worker->addFunction('other_stuff', '...');
$worker->addFunction('send_welcome_email', '...');
$worker->addFunction('resize_user_avatar', '...');
```

How about now?

```php
// worker.php

$worker->addFunction('one', '...');
$worker->addFunction('two', '...');
$worker->addFunction('three', '...');


// client.php

for ($i = 0; $i < 5; $i++) {
    $client->doBackground('one', '...');
    $client->doBackground('two', '...');
    $client->doBackground('three', '...');
}
```

How about now?

```php
// worker.php

$worker->addFunction('one', '...');
$worker->addFunction('two', '...');
$worker->addFunction('three', '...');


// client.php

for ($i = 0; $i < 5; $i++) {
    $client->doBackground('one', '...');
    $client->doBackground('two', '...');
    $client->doBackground('three', '...');
}
```

```
one
one
one
one
one
two
two
two
two
two
three
three
three
three
three
```

Solution

- Forget about `$functionName` (***function*** underlined)
- Think of it more like `$queueName` (***queue*** underlined)
- All jobs will end up in the same place and then get routed
- You only need to do this if your workers are servicing more than one function, and your queue receives a lot of traffic

```
// worker.php

$worker->addFunction('user_registration', function (\GearmanJob $job) {
    $workload = json_decode($job->workload());

    switch ($workload['function']) {
        case 'resize_user_avatar':
            return resizeUserAvatar($workload['data']);
        case 'send_welcome_email':
            return sendWelcomeEmail($workload['data']);
        case 'other_stuff':
            return otherStuff($workload['data']);
    }
});
```

Now who's first?

```php
// client.php

$client->doLowBackground(
    'user_registration',
    json_encode(['function' => 'resize_user_avatar', 'data' => [...]])
);
$client->doHighBackground(
    'user_registration',
    json_encode(['function' => 'send_welcome_email', 'data' => [...]])
);
$client->doBackground(
    'user_registration',
    json_encode(['function' => 'other_stuff', 'data' => [...]])
);
```

Recap

Recap

- Offload non-critical or expensive tasks to another server using a queue

Recap

- Offload non-critical or expensive tasks to another server using a queue
- Choose wisely between sync and async, depending on your use case

Recap

- Offload non-critical or expensive tasks to another server using a queue
- Choose wisely between sync and async, depending on your use case
- Fill your worker servers with workers until they can't take anymore, then add more worker servers as needed

Recap

- Offload non-critical or expensive tasks to another server using a queue
- Choose wisely between sync and async, depending on your use case
- Fill your worker servers with workers until they can't take anymore, then add more worker servers as needed
- Most of what we just covered, at a conceptual level, is applicable to all sorts of queueing systems

Options!

- Amazon Simple Queue Service (SQS)
  - https://aws.amazon.com/sqs/
- RabbitMQ
  - https://www.rabbitmq.com/
- ZeroMQ
  - http://zeromq.org/
- beanstalkd
  - https://kr.github.io/beanstalkd/

https://joind.in/talk/895bd

(in case you missed it...)

Questions?

Thanks!