

# Projet OSF - Gitification -

*Open Source Frameworks 2013  
Yverdon*

*Dorian Gambin  
Geoffrey Papaux  
Vincent Grivel  
Vincent Pasquier*

*Prof. Olivier Liechti*

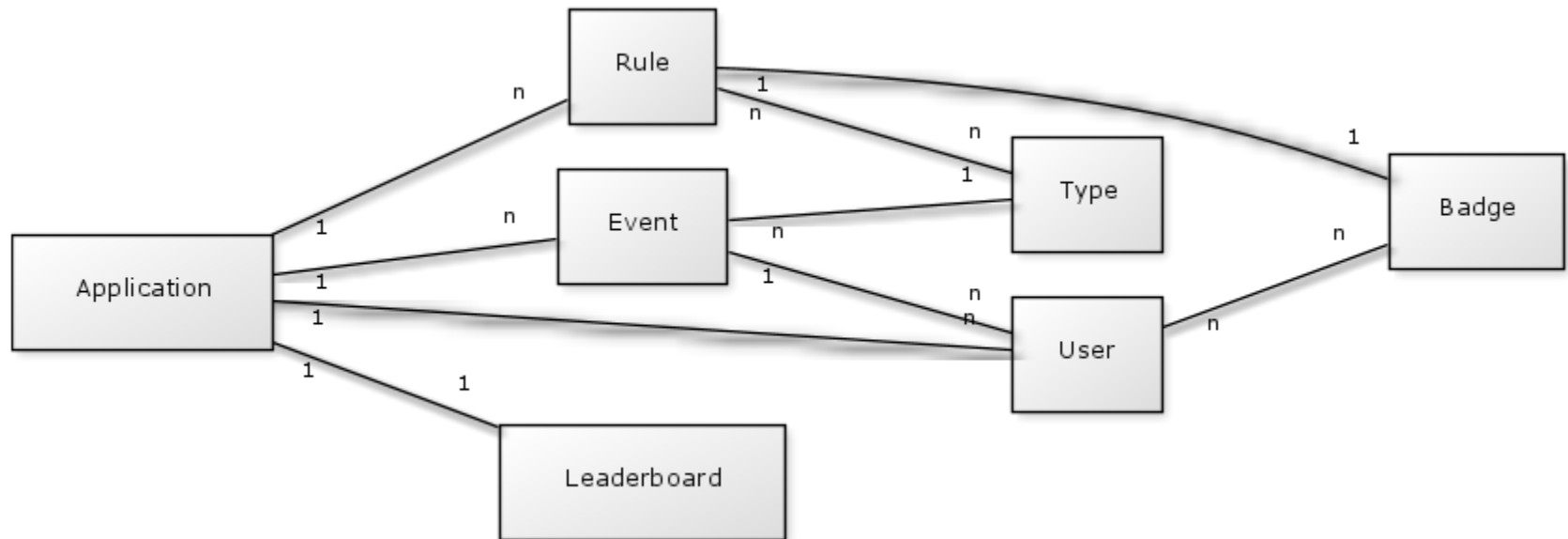
# Sommaire

- Roles
- Gitification : Concept
- JS-Stack
- View : Angular
- REST : Restify
- Database : Riak
- Continuous integration
- Benchmarks

# Roles

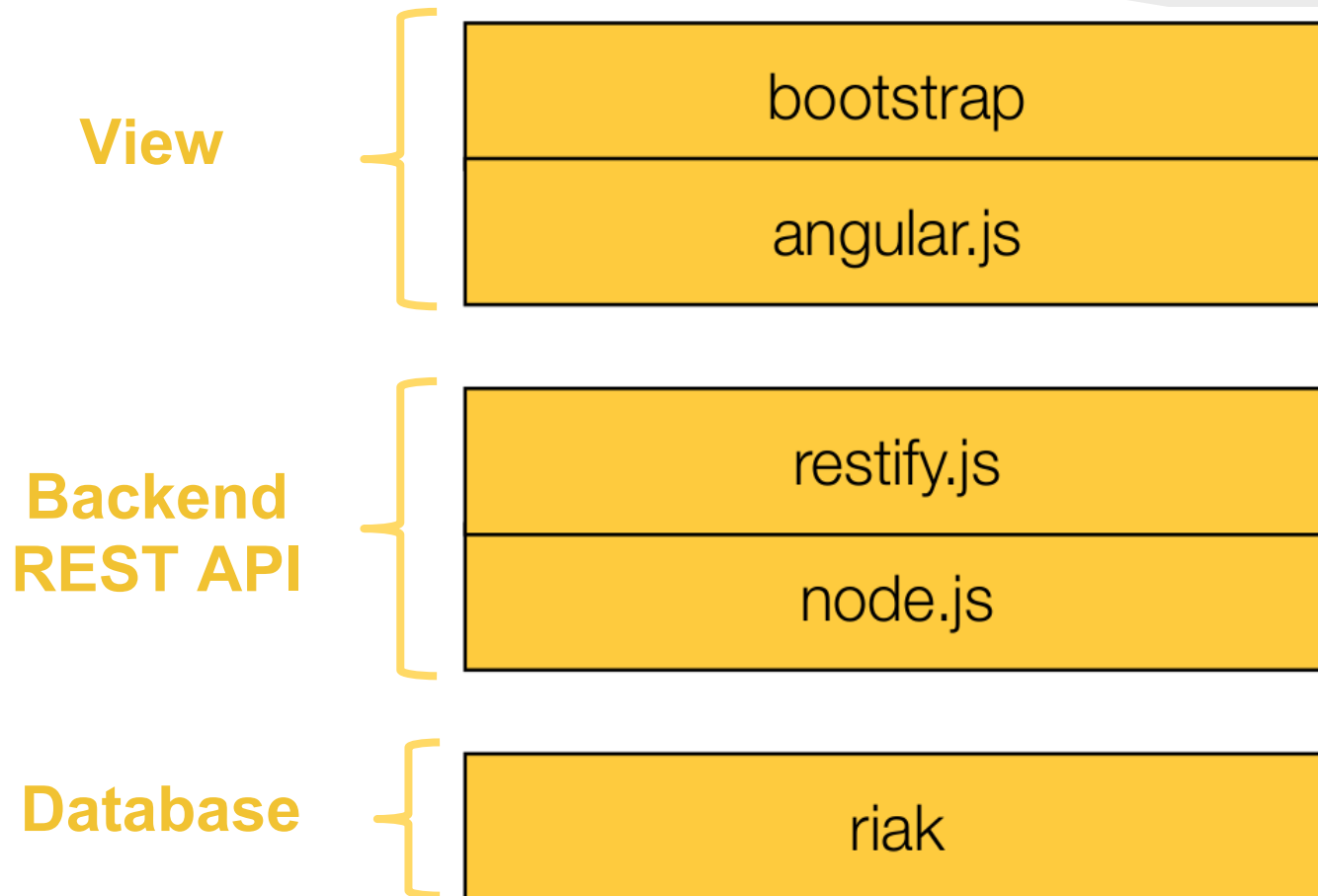
- Dorian:
  - Dev-server (db/riak, benchmarks)
- Geoffrey:
  - Dev-server (rest-api, tests, db/mysql)
- Vincent G.:
  - Dev-client (angular client)
- Vincent P.:
  - Devop + Dev-server

# Gitification: concept



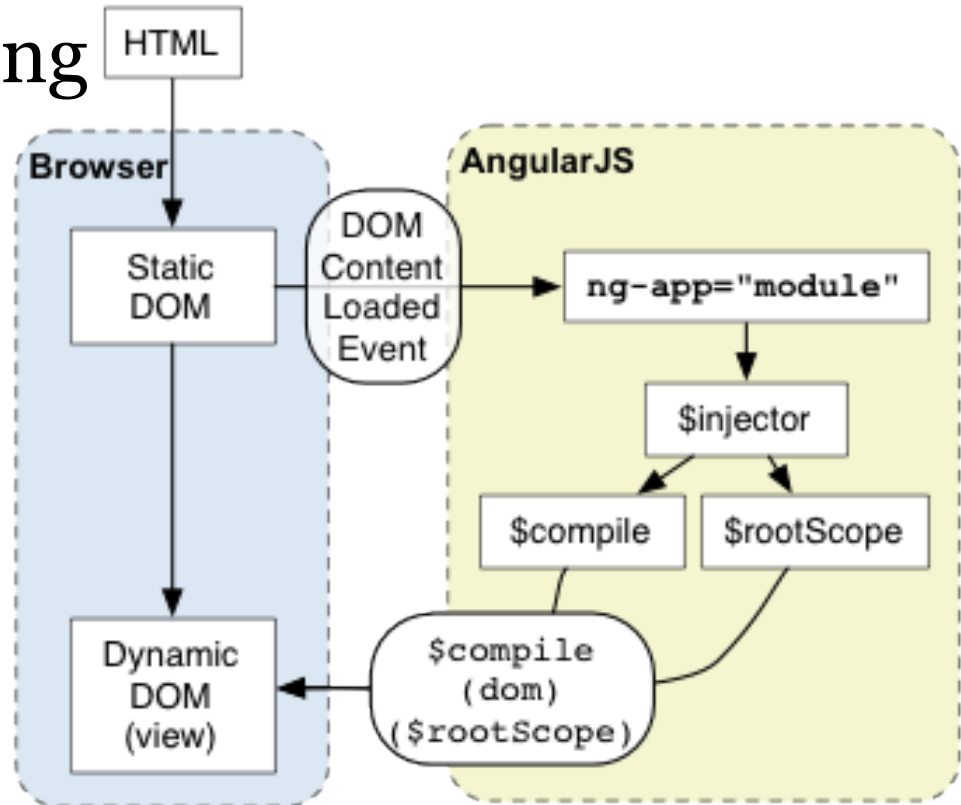
<http://yuml.me/167103be>

# Stack JS



# Angular

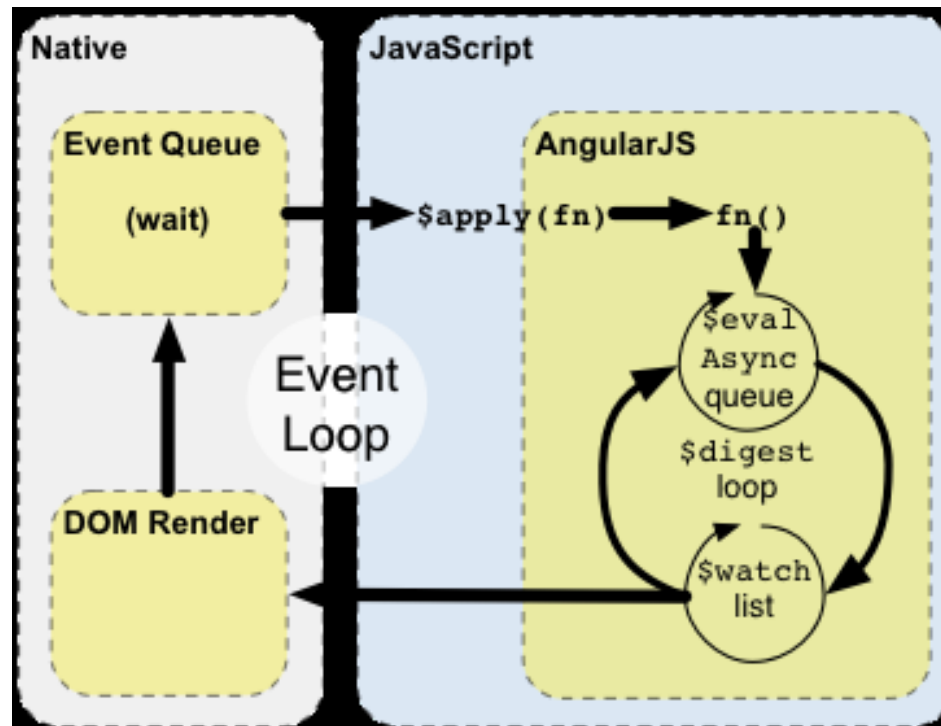
- Structural framework for dynamic web application
- Two way data binding
- Directives
- Partial views



# Angular : Two way data binding

```
<input ng-model="name">
```

```
<p>Hello {{name}}!</p>
```



<http://docs.angularjs.org/guide/concepts>

# Angular: Resource

```
var api = 'http://localhost\\:8080/api';
angular.module('gamificationServices', ['ngResource'], function ($provide) {
  $provide.factory('Badges', function ($resource) {
    return $resource(api + '/applications/:appId/badges/:badgeId', {
      appId: '@appId'
    }, {query: {method: 'GET', isArray: true},
      create: {method: 'POST', headers: {'Content-Type':
'application/json'}}},
      get: {method: 'GET', badgeId: '@badgeId'},
      update: {method: 'PUT', headers: {'Content-Type': 'application/json'},
        badgeId: '@badgeId'}
    });
  });

  $scope.allBadges = Badges.query({appId: 1});

  badge.$update({appId: 1, badgeId: badge.badge_id});
```



# Apiary



Must have for accordance

Developer specification

API mock

Code duplication

<http://docs.gitification.apiary.io/>

# Restify

- From the creator
  - "Node module to build **correct** REST web services"
- Heavily based on express, but
  - express targets browser application (templating, rendering, ...)
  - restify targets **strict** API services
- Pretty straight forward

```
var restify = require('restify');  
var server = restify.createServer();
```

```
server.get('/hello/:name', function (req, res, next) {  
  res.send('hi ' + req.params.name);  
});  
server.listen(8080);
```

# Restify : cool features (1)

- Versioned Routes (HTTP Header: accept-version)

```
function v1(req, res, next) {  
  res.send('hello: ' + req.params.name);  
}  
function v2(req, res, next) {  
  res.send({hello: req.params.name});  
}
```

```
server.get({path: '/hello/:name', version: '1.1.3'}, v1);  
server.get({path: '/hello/:name', version: '2.0.0'}, v2);
```

# Restify : cool features (2)

- Request parameter validation
  - Can be coupled with node-validator
  - And chained with restify

=> modify prototype of req parameter (`http.IncomingMessage`)

- Validation functions become accessible in restify 'req' object

```
function v1(req, res, next) {  
  req.check('email', '"email": must be valid email @').isEmail();  
  req.check('name', '"name": must be set').notNull();  
  req.check('year', '"lastname": must be a valid year').isInt();  
  var errors = req.validationErrors();  
  if (errors) { // send errors to the client }  
}
```

# Riak

API

REST API

Protocol Buffer API



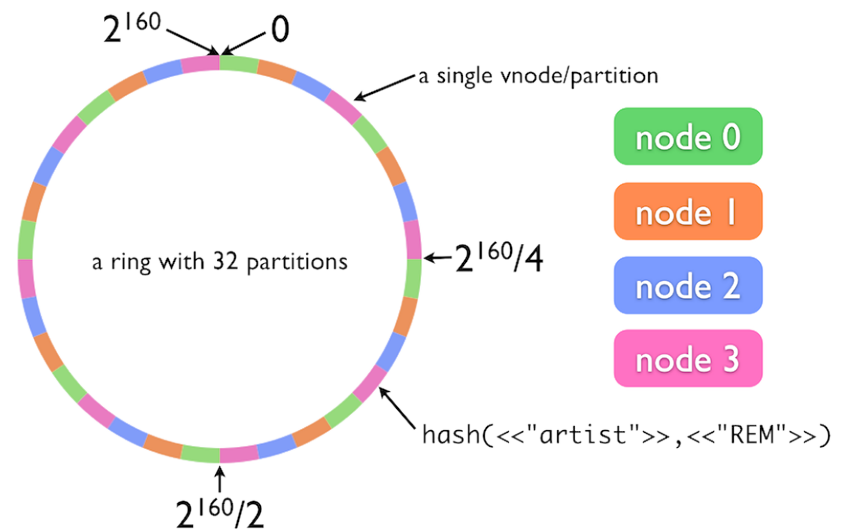
Multiple client libraries (C/C++, Java, Python, Ruby, ...)

Bucket, keys and values (NoSQL)

Clustering, 160-bit integer space

Replication

MapReduce, LinkWalking



# Riak-js

Riak client for Node.js

Asynchronous calls

One web-page documentation

Advanced documentation is the code on Github

The logo for Riak-js, featuring the text "riak-js" in a bold, lowercase, sans-serif font. The text is yellow and is set against a dark grey rectangular background.

# Riak-js: U mad ?

bucket = "blu", key = "dablu"

**1:** rc.exists(bucket, key, function (err, result, meta) { if(result) { //exists } });

key = undefined OR key = null

**2:** rc.exists(bucket, key, function (err, result, meta) { if(result) { //exists } });

1 and 2 does not check the existence of the same thing.

- 1 checks if the key "dablu" in bucket "blu" exists.
- 2 checks if the bucket "blu" exists.

Bug reason: // TODO: Correctly parse each element!



**Avoid any call with a variable undefined or null. Otherwise you are dependent to the underlying library.**

# Riak-js: Do You Even Spaghetti

```
359 exports.createRule = function (rule, callback) {
360   rule.rule_id = shortid.generate();
361
362   rc.exists(bPrefix + "badge" + rule.application_id, rule.badge_id, function (err, result/*, meta*/) {
363     if (!result) { callback.error(404, "Badge does not exist"); return; }
364     rc.exists(bPrefix + "eventType" + rule.application_id, rule.event_types[0].event_type, function (err, result/*, meta*/) {
365       if (!result) { callback.error(404, "Event type does not exist"); return; }
366       rc.get(bPrefix + "eventType" + rule.application_id, rule.event_types[0].event_type, function (err, eventtype, meta) {
367         if (magicCheck(callback, err, eventtype, meta)) { return; }
368         meta.links.push({ bucket: bPrefix + "rule" + rule.application_id, key: rule.rule_id, tag: 'hasRule' });
369         rc.save(bPrefix + "eventType" + eventtype.application_id, eventtype.type_id, eventtype, meta, function (err, result, meta) {
370           if (magicCheck(callback, err, result, meta)) { return; }
371           rc.save(bPrefix + "rule" + rule.application_id, rule.rule_id, rule, function (err, result, meta) {
372             if (magicCheck(callback, err, result, meta)) { return; }
373             callback.success(201, "Successfully created rule", rule);
374           });
375         });
376       });
377     });
378   });
379 };
380 };
```



# Riak-js: Do You Even Spaghetti

```
358 exports.createRule = function (rule, callback) {
359   rule.rule_id = shortid.generate();
360
361   async.series([
362     function (cb) {
363       rc.exists(bPrefix + "badge" + rule.application_id, rule.badge_id, function (err, result/*, meta*/) {
364         if (!result) { callback.error(404, "Badge does not exist"); cb("err"); }
365         cb();
366       });
367     },
368     function (cb) {
369       rc.exists(bPrefix + "eventType" + rule.application_id, rule.event_types[0].event_type, function (err, result/*, meta*/) {
370         if (!result) { callback.error(404, "Event type does not exist"); cb("err"); }
371         cb();
372       });
373     },
374     function (cb) {
375       rc.get(bPrefix + "eventType" + rule.application_id, rule.event_types[0].event_type, function (err, eventtype, meta) {
376         if (magicCheck(callback, err, eventtype, meta)) { cb("err"); }
377         meta.links.push({ bucket: bPrefix + "rule" + rule.application_id, key: rule.rule_id, tag: 'hasRule' });
378         rc.save(bPrefix + "eventType" + rule.application_id, eventtype.type_id, eventtype, meta, function (err, result, meta) {
379           if (magicCheck(callback, err, result, meta)) { cb("err"); }
380           cb();
381         });
382       });
383     },
384     function (cb) {
385       rc.save(bPrefix + "rule" + rule.application_id, rule.rule_id, rule, function (err, result, meta) {
386         if (magicCheck(callback, err, result, meta)) { cb("err"); }
387         cb();
388       });
389     }
390   ], function (err) {
391     if (err) { return; }
392     callback.success(201, "Successfully created rule", rule);
393   });
394 }
```

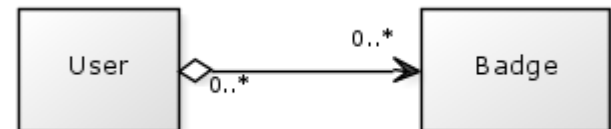


**When dealing with async calls, use a helper library to avoid spaghetti code.**

# Riak - link and link walking

```
rc.get(bucketUser, user_id, function (err, result, meta) {  
  link = { bucket: bucketBadge, key: badge_id, tag: "hasBadge" };  
  if (existsLink(meta.links, link)) { return;}  
  meta.links.push(link);  
  rc.save(bucketUser, user_id, result, meta);  
});
```

```
rc.walk(bucketUser, user_id, [{bucket: bucketBadge, tag: "hasBadge"}], function (err, result, meta) {  
  result[0].forEach(function (entry) {  
    badge = entry.data;  
  });  
});
```



# Riak - key filtering

```
rc.save(bucket, keypart1 + "&" + keypart2, value, function (err, result, meta) {  
});
```

```
rc.mapreduce.add(  
  {bucket: bucket,  
    "key_filters": [["and", [["tokenize", "&", 1], ["eq", part1]],  
    [["tokenize", "&", 2], ["eq", part2]]]]},  
  ).map('Riak.mapValuesJson').run(function (err, result, meta) { });
```



**Choose wisely your keys to avoid conflict during tokenize.**

# Grunt task-runner



- Automate tasks
  - compilation
  - tests
  - uglify
  - lint
  - ...
- Plugin based
  - 28 official plugins
  - 853 third-party plugins
- Configuration file: `Gruntfile.js`
- Development dependencies: `package.json`

# Gruntfile.js + dependencies

## Package.json

```
{
  ...
  "scripts": {
    "test": "grunt travis --verbose"
  },
  "devDependencies": {
    "grunt": "~0.4.1",
    "grunt-karma": "~0.3.0"
  },
  "dependencies": {
    "shortid": ">=1.0.9",
    "riak-js": ">= 0.9.0",
    "restify": ">= 2.2.0",
    "rewire": ">=1.1.2",
    "api-easy": "0.2.x",
    "validator": "1.1.0",
    "async": "0.2.x",
    "mysql": "2.0.0-alpha8"
  },
  ...
}
```

## Gruntfile.js

```
module.exports = function (grunt) {
  grunt.initConfig({
    karma: {
      unit: {
        configFile: 'karma.conf.js',
        singleRun: true
      }
    }
  });

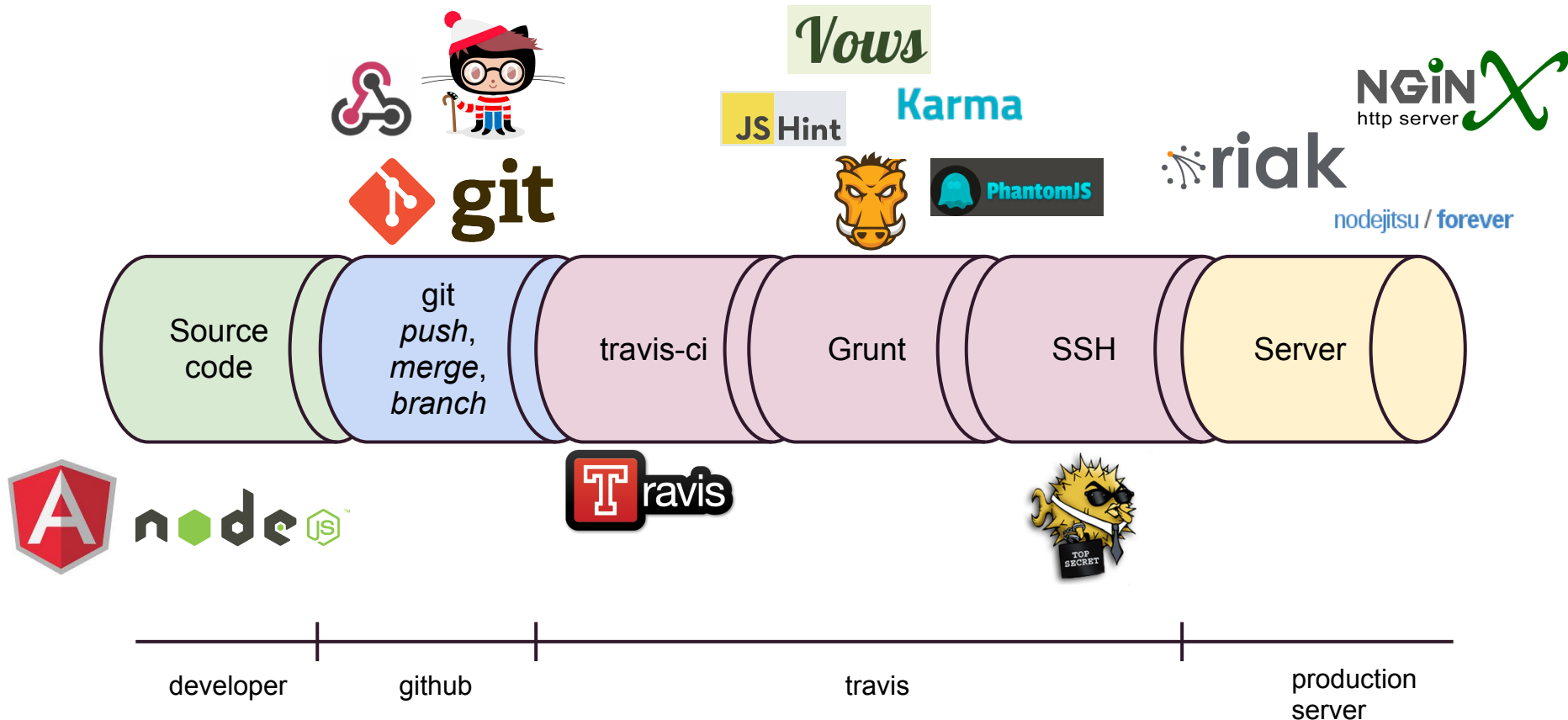
  grunt.loadNpmTasks('grunt-karma');
  grunt.registerTask('travis', [ 'karma' ]);
};
```

## Tilde version range

```
"~1.2.3" = ">=1.2.3 <1.3.0"
"~1.2" = ">=1.2.0 <1.3.0"
```

# Continuous integration

gitification build passing



# In action

- Check out our Travis-ci build report.
  - <https://travis-ci.org/Gitification/gitification-server/builds/7431861>
  - <https://travis-ci.org/Gitification/gitification-client/builds/7559785>

## Gitification/gitification-server

Gitification's project server


Current

Build History


Pull Requests

Branch Summary

Build #65



Build

 [65](#)

Commit

[69682ba \(master\)](#)

State

Passed

Compare

[516863f6cd14...69682ba08422](#)

Finished

about 19 hours ago

Author

[Perdiesk](#)

Duration

54 sec

Committer

[Perdiesk](#)

Message

FindAllUsers implementation

Config

NodeJs: 0.8

# Plato, JavaScript Source Analysis

- Metrics

- Source Line of Code (SLOC)
- Average Maintainability
- Estimated errors in implementation (Halstead measure)
- Lint errors
- Functions weight
  - Cyclomatic complexity
  - SLOC

- Plato on our server

- <http://ks25416.kimsufi.com/plato/>

```
$ plato -d plato -l .jshintrc -r controllers/*.js db/*.js helpers/*.js lib/*.js test/*.js
```



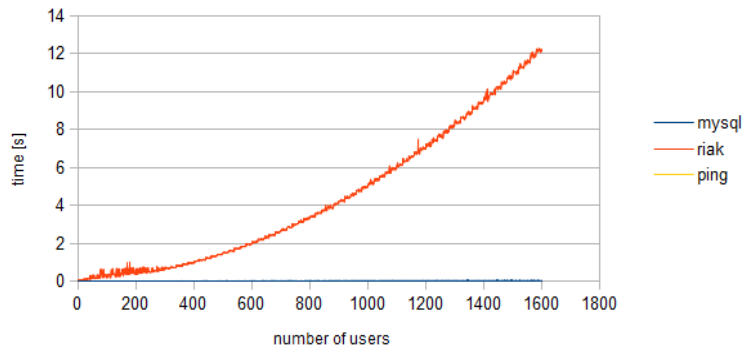
# Benchmark



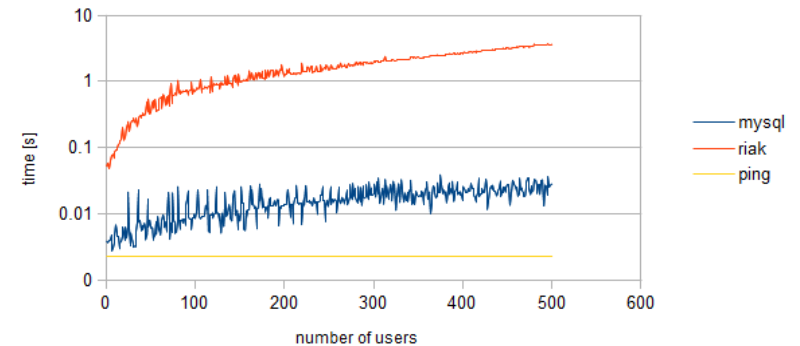
VS



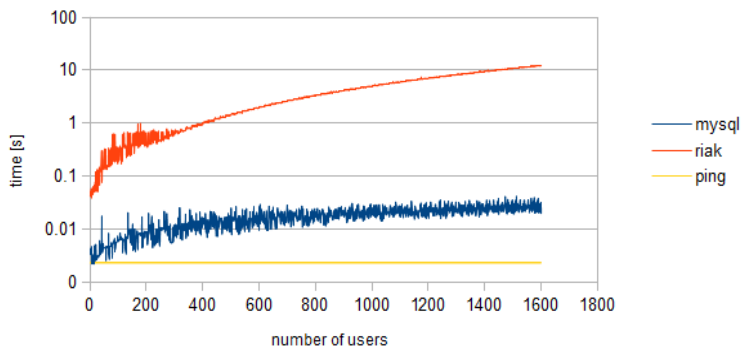
AllUsers (lin-lin)



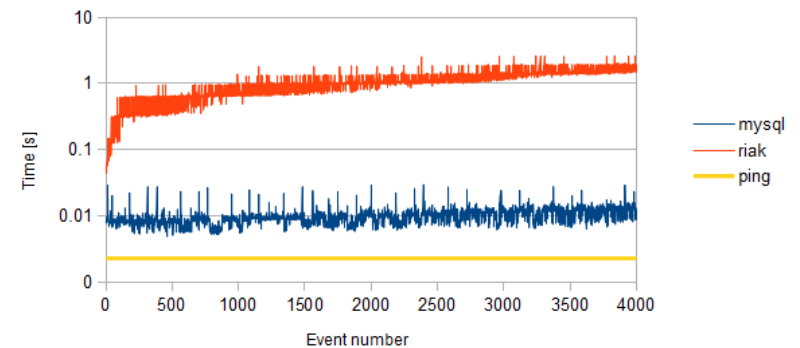
LeaderBoard (lin-lin)



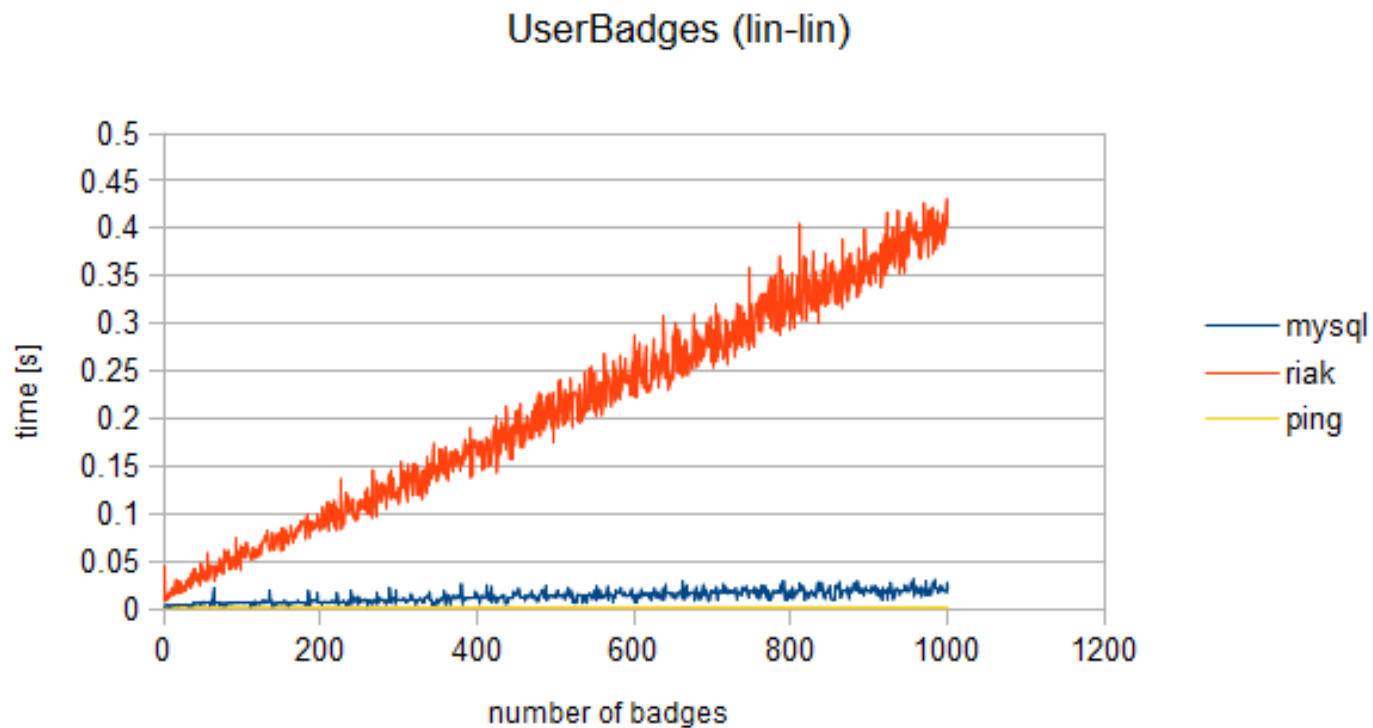
AllUsers (lin-log)



CreateEvent (lin-log)



# Benchmark

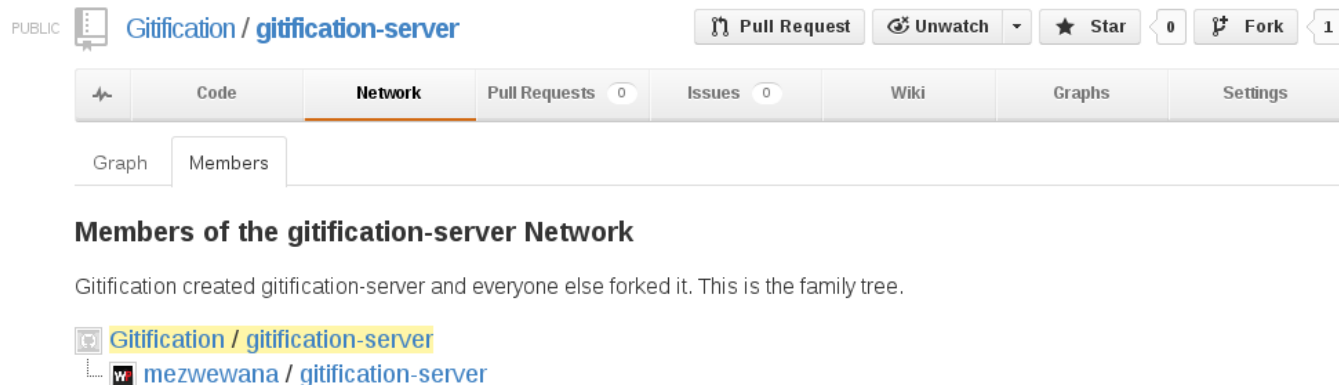


# Conclusion

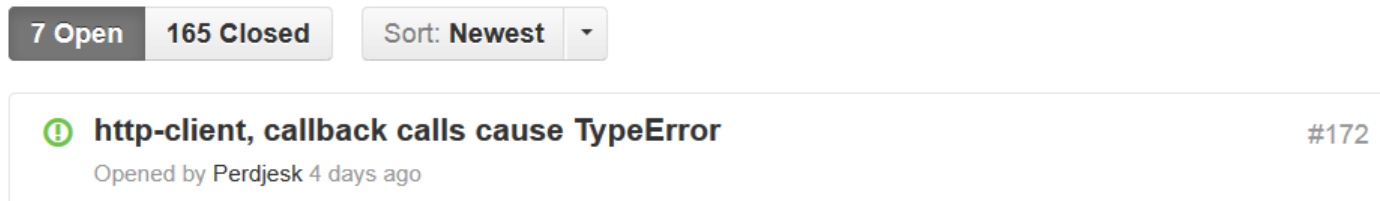
- NoSQL model designed with "relational mind"
  - No real aggregate !
  - Fair benchmarks require new model for db/riak
- Only one riak node, 5 nodes are recommended
- Please visit our wiki for more information
  - <https://github.com/Gitification/gitification/wiki>
- JavaScript environment in constant change

# Conclusion

## We got forked



## We participated by reporting an issue in riak-js



# Appendix: CLI tools

- Node Version Manager (nvm)

- `nvm install 0.8`

- `nvm use 0.8`

- Node Package Manager (npm)

- `npm (install|update|...) [-g] [module_name]`

- GruntJS

- `grunt [cible]`

- Travis

- `travis encrypt [something]`

- `travis status`

```
$ travis status  
build #65 passed
```

# Appendix: SSH Deploy with Travis

- Capistrano ? Shell !
  - Uses SSH RSA key auth to connect and deploy
  - Install on travis after each compile
- Homebrewed solution
  - Based on [gist:4242707](#)
    - Slice SSH private key in chunks
    - Encode it in base 64 to remove encoding hell
    - Encrypt each chunks with `travis encrypt`
    - Store each chunk in `.travis.yml` file
  - Add SSH key to travis virtual machine
  - Connect and deploy on server

