

TP CI/CD avec Gitlab



Un client vous demande d'automatiser les test de son application grâce a Gitlab-ci

Vous trouverez le projet sur le git suivant :
<https://github.com/jakerella/node-unit-tests>

Il s'agit d'une application NodeJs

I) Télécharger le projet sous forme de ZIP (Attention a ne pas cloner le projet)

II) Modifier les test suivant :

```
5 var i18n = require( '../app/i18n' );
6
7
8 describe('i18n', function() {
9   it('should return correct translation if available', function() {
10
11     assert( i18n('hello'), 'hello' );
12     assert( i18n('hello', 'en'), 'hello' );
13     assert( i18n('hello', 'es'), 'hola' );
14     assert( i18n('beer', 'es'), 'cervesa' );
15
16   });
17
18   it('should return the input on a missing phrase', function() {
19
20     assert.equal( i18n('foobar'), 'foobar' );
21     assert.equal( i18n('foobar', 'es'), 'foobar' );
22     assert.equal( i18n('beer', 'jp'), 'beer' );
23
24   });
25
26   it('should handle mixed case', function() {
27
28     assert.equal(i18n('Beer'), 'cervesa');
29
30   });
31 });
32
```

```
5 ↓
6 describe('i18n', function() {↓
7   it('should return correct translation if available', function() {↓
8
9     assert(i18n('hello'), 'hello');↓
10    assert(i18n('hello', 'en'), 'hello');↓
11    assert(i18n('hello', 'es'), 'hola');↓
12    assert(i18n('beer', 'es'), 'cervesa');↓
13
14  });↓
15
16   it('should return the input on a missing phrase', function() {↓
17
18     assert.equal(i18n('foobar'), 'foobar');↓
19     assert.equal(i18n('foobar', 'es'), 'foobar');↓
20     assert.equal(i18n('beer', 'jp'), 'beer');↓
21
22   });↓
23
24   it('should handle mixed case', function() {↓
25
26     assert.equal(i18n('Beer'), 'Beer');↓
27
28   });↓
29 });[EOF]
```

Modifier les test suivant :

```

JS test-i18n.js  JS test-text-mock.js
test > JS test-text-mock.js > describe('text module with mock') callback > afterEach() callback
14 |     require.cache[ require.resolve('../app/i18n') ].exports = function() {
15 |         return 'foo';
16 |     };
17 | });
18 |
19 | afterEach(function() {
20 |     require.cache[ require.resolve('../app/i18n') ].exports = i18nOriginal;
21 | });
22 |
23 | it('should return correct translations from a file', function( done ) {
24 |
25 |     var translate = require('../app/text');
26 |
27 |     translate( __dirname + '/phrases.json', 'es', function(err, data) {
28 |
29 |         if (err) { return done( err ); }
30 |
31 |         assert.equal( data.length, 3 );
32 |         assert.deepEqual( data, ['foo', 'foo', 'foo'] );
33 |
34 |         done();
35 |
36 |     } );
37 |
38 | });
39 |

```

```

    require.cache[require.resolve('../app/i18n')].exports = i18nOriginal;
});

it('should return correct translations from a file', function(done) {
    var translate = require('../app/text');

    translate(__dirname + '/phrases.json', 'es', function(err, data) {
        if (err) { return done(err); }

        assert.equal(data.length, 3);
        assert.deepEqual(data, ['hola', 'cerveza', 'I like beer']);

        done();
    });
});
EOF

```

III) Modifier le fichier package.json :

```
JS test-i18n.js  {} package.json X
{} package.json > {} scripts > abc test
1  {↓
2    "name": "unit-tests",↓
3    "version": "0.1.0",↓
4    "private": true,↓
5    "description": "A testbed for unit testing Node",↓
6    "main": "server.js",↓
7    "scripts": {↓
8      "test": "./node_modules/mocha/bin/mocha test/"↓
9    },↓
10   "author": "Jordan Kasper",↓
11   "license": "MIT",↓
12   "devDependencies": {↓
13     "chai": "^3.4.1",↓
14     "mocha": "^2.3.3",↓
15     "sinon": "^1.17.2",↓
16     "superagent": "^1.4.0"↓
17   }↓
18 }↓
19 [EOF]
```

```
JS test-i18n.js  {} package.json X
{} package.json > ...
1  {↓
2    "name": "unit-tests",↓
3    "version": "0.1.0",↓
4    "private": true,↓
5    "description": "A testbed for unit testing Node",↓
6    "main": "server.js",↓
7    "scripts": {↓
8      |  "test": "node node_modules/mocha/bin/mocha"↓
9    },↓
10   "author": "Jordan Kasper",↓
11   "license": "MIT",↓
12   "devDependencies": {↓
13     |  "chai": "^3.4.1",↓
14     |  "mocha": "^2.3.3",↓
15     |  "sinon": "^1.17.2",↓
16     |  "superagent": "^1.4.0"↓
17   }↓
18 |[EOF]
```

IV) 1 –Installer node sur votre machine ou faire un docker

2- lancer « npm i » dans la console en vous trouvant a la racine de votre projet (ou commande run)

3 – Puis lancer la commande : « npm test »

```
oceane@LAPTOP-S69BUJTC MINGW64 ~/Desktop/test/node-unit-tests-master
$ npm test

> unit-tests@0.1.0 test C:\Users\oceane\Desktop\test\node-unit-tests-master
> node node_modules/mocha/bin/mocha
```

```
i18n
  ✓ should return correct translation if available
  ✓ should return the input on a missing phrase
  ✓ should handle mixed case
```

```
server
  ✓ should return Hello World at /
  ✓ should return a 404 at /foo
```

```
team module
  ✓ should return members from mock
```

```
text module with mock
  ✓ should return correct translations from a file
```

```
text module
  ✓ should return correct translations from a file
```

```
8 passing (59ms)
```


Vous devriez avoir la même réponse qu'ici si vous avez modifié correctement les test

- V) 1- Créer / connecter vous a votre compte GitLab
- 2 - Créer un nouveau projet
- 3- pusher le code de l'application sur le git

ocean > integration-continue > Repository

master integration-continue / +

History Find file Web IDE

 Delete Dockerfile
ocean authored just now 72835362

Name	Last commit	Last update
app	inital commit	4 days ago
test	inital commit	4 days ago
.gitignore	inital commit	4 days ago
README.md	inital commit	4 days ago
package-lock.json	inital commit	4 days ago
package.json	inital commit	4 days ago

README.md

Node Unit Test Examples

This is a repo of small examples of testing Node.js modules and applications

VI) 1 –Créer a la racine du projet un fichier nommer .gitlab-ci.yml
(ATTENTION : ne pas oublier le point)



! .gitlab-ci.yml

2- Déclarer une image node(dernière version) qui servira de base pour notre conteneur dans gitlab-ci

3 – Instancier un cache pour y stocker le répertoires node_modules qui contient nos dépendances (pour éviter de le retélécharger a chaque fois qu'on lance un test on le stock dans le cache)

4 – On définit le processus de test que gitlab-ci devra exécuter

5 – le script de test devra lancer les commandes suivantes :

➤ npm install

➤ npm test

6 – Ensuite pusher gitlab-ci.yml sur git

Vous pouvez vous rendre dans la section CI/CD > pipelines



#81110540

latest



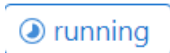
master d4537ac1



init commit



Vous devriez avoir quelques chose qui ressemble a ceci.

Job #290883889 triggered just now by  oceane

```
Running with gitlab-runner 12.2.0 (a987417a)
  on docker-auto-scale 72989761
  ▼ Using Docker executor with image node:latest ...
  Pulling docker image node:latest ...
```

```
...
```