

TDD with REST APIs





Somkiat
Home

Somkiat Puisungnoen

Update Info 1

View Activity Log 10+

...

Timeline

About

Friends 3,138

Photos

More ▾

When did you work at Opendream?

×

...

22 Pending Items

Intro

Software Craftsmanship

Software Practitioner at สยามชำนานุกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Post

Photo/Video

Live Video

Life Event

What's on your mind?

Public ▾

Post

Somkiat Puisungnoen

15 mins · Bangkok · 🌐 ▾

Java and Bigdata

...



Facebook interface for the page **somkiat.cc**. The top navigation bar includes the Facebook logo, a search bar, and the page name **Somkiat** with a **Home** link. Icons for friends, messages, and a help menu are also present.

The main navigation menu includes **Page** (selected), **Messages**, **Notifications** (3), **Insights**, **Publishing Tools**, **Settings**, and **Help**.

The page cover image shows a man in a white Superman t-shirt with "SOMKIAT.CC" printed on it, posing against a white wall. The profile picture is a smaller version of the same image.

Page information: **somkiat.cc**, **@somkiat.cc**.

Left sidebar menu: **Home** (selected), **Posts**, **Videos**, **Photos**.

Below the cover image, there are buttons for **Liked**, **Following**, **Share**, and a menu icon. A blue button labeled **+ Add a Button** is also visible.

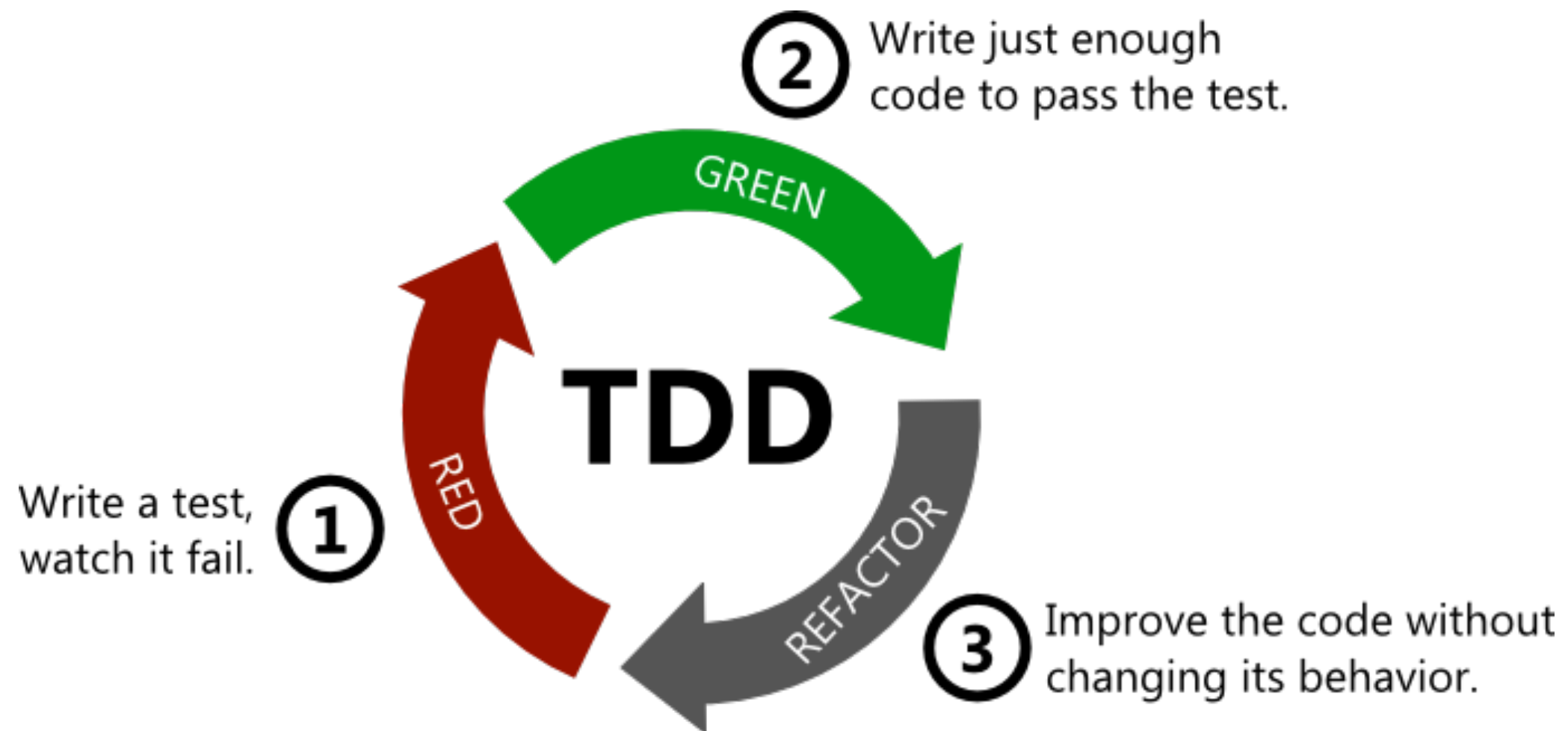
A blue call-to-action box on the right side of the cover image says: **Help people take action on this Page.**



Develop REST APIs with Node.js



Test-Driven Development



Why we need to test ?

Help you to catch bugs

Develop features faster

Enforce modularity of your project



**But,
It's take time to learning and
practice !!**



Goals

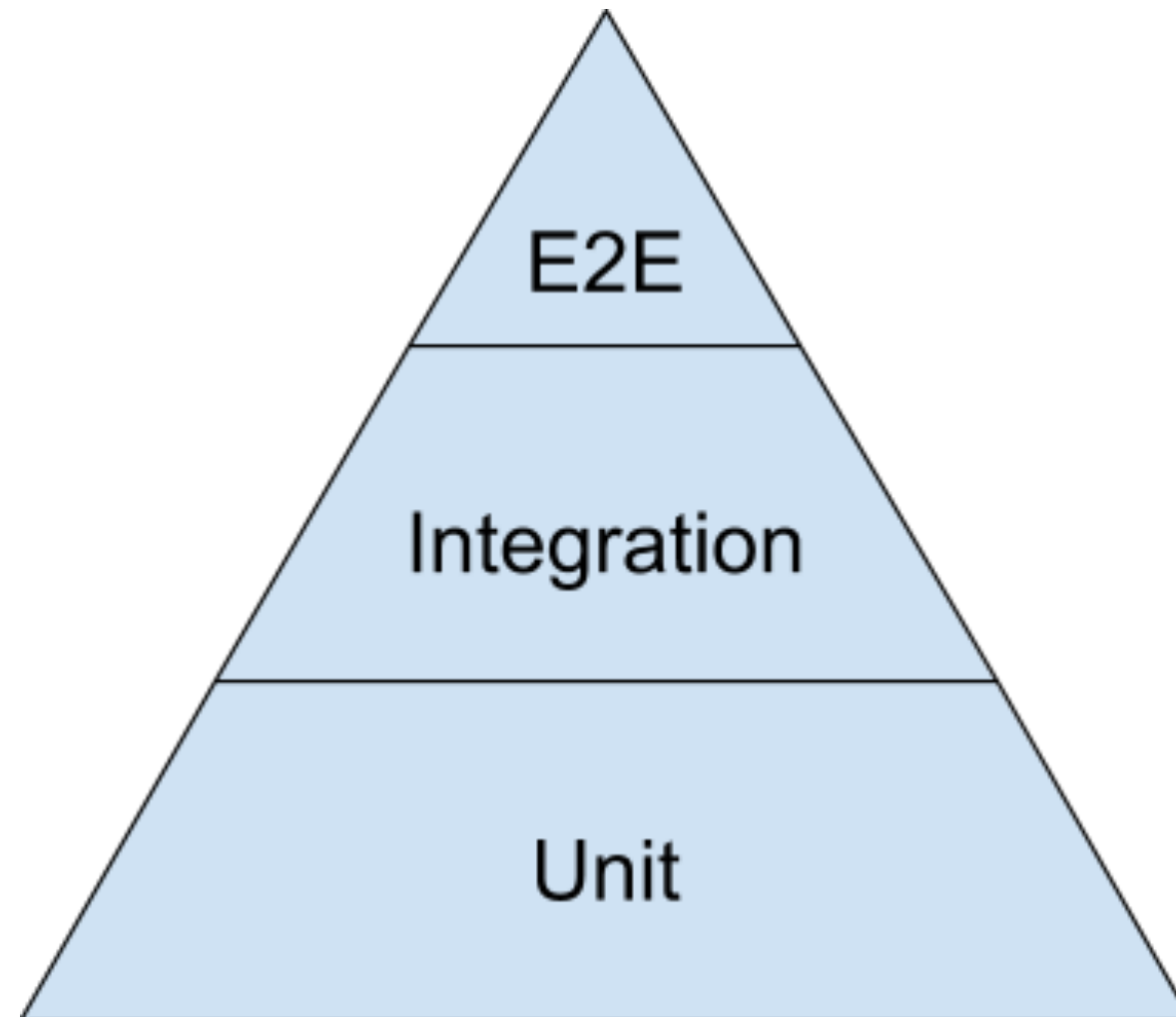
How to **THINK** when and where
you should test



Type of testing



Testing Pyramid



Workshop with Node.js



koa

next generation web framework for node.js



Resources

<http://koa.js.com/>

<http://www.chaijs.com/>

<https://github.com/chaijs/chai-http>



Let's start



Preparing Environments

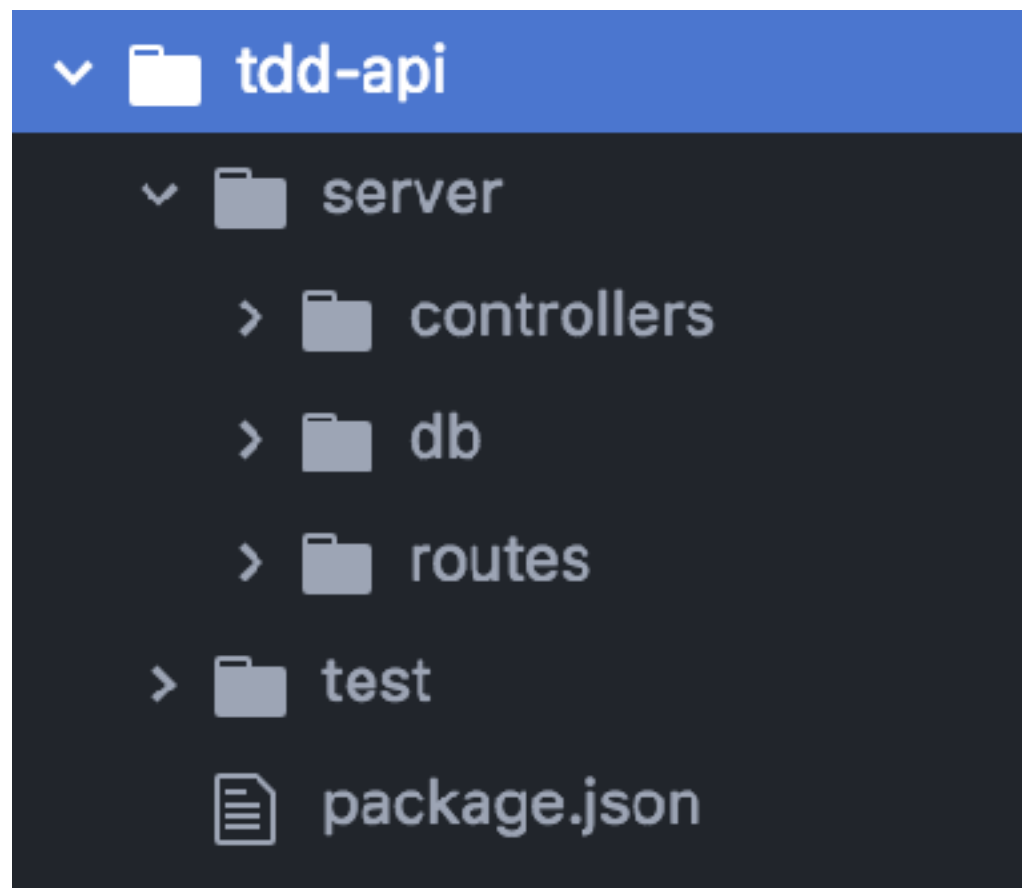


Preparing Environments

1. APIs project structure
2. Data Store/Database
3. Testing



1. Create project structure



Project structure

Folder Name	Description
server	Keep production code
test	Keep test code



Project structure of Server

Folder Name	Description
routes	Define routing of APIs
controllers	Main controller of each APIs
db	Manage data from Database



Create project with Node.js

```
$mkdir demo-tdd
```

```
$cd demo-tdd
```

```
$npm init
```

```
$mkdir -p server/{routes,controllers,db} test
```



2. Setup Database



Sequelize



Resources

<https://github.com/mapbox/node-sqlite3>

<http://knexjs.org/>



Setup Database with KNEX.JS

```
$npm i sqlite3 --save-dev
```

```
$npm i knex
```



Setup Database with KNEX.JS

`$node_modules/knex/bin/cli.js init`

Creating file knexfile.js

```
test: {  
  client: 'sqlite3',  
  connection: {  
    filename: './test.sqlite3'  
  },  
  useNullAsDefault: true,  
  migrations: {  
    directory: path.join(BASE_PATH, "migrations")  
  },  
  seeds: {  
    directory: path.join(BASE_PATH, "seeds")  
  }  
},
```



Database Migration

\$node_modules/knex/bin/cli.js migrate:make beer --env test

\$node_modules/knex/bin/cli.js migrate:latest --env test

```
exports.up = function(knex, Promise) {  
  return knex.schema.createTable("beer", table => {  
    table.increments();  
    table.string("name").nullable().unique();  
    table.text("description").nullable();  
    table.string("brand").nullable();  
  });  
};  
  
exports.down = function(knex, Promise) {  
  return knex.schema.dropTable("beer");  
};
```



Seeding data for Testing

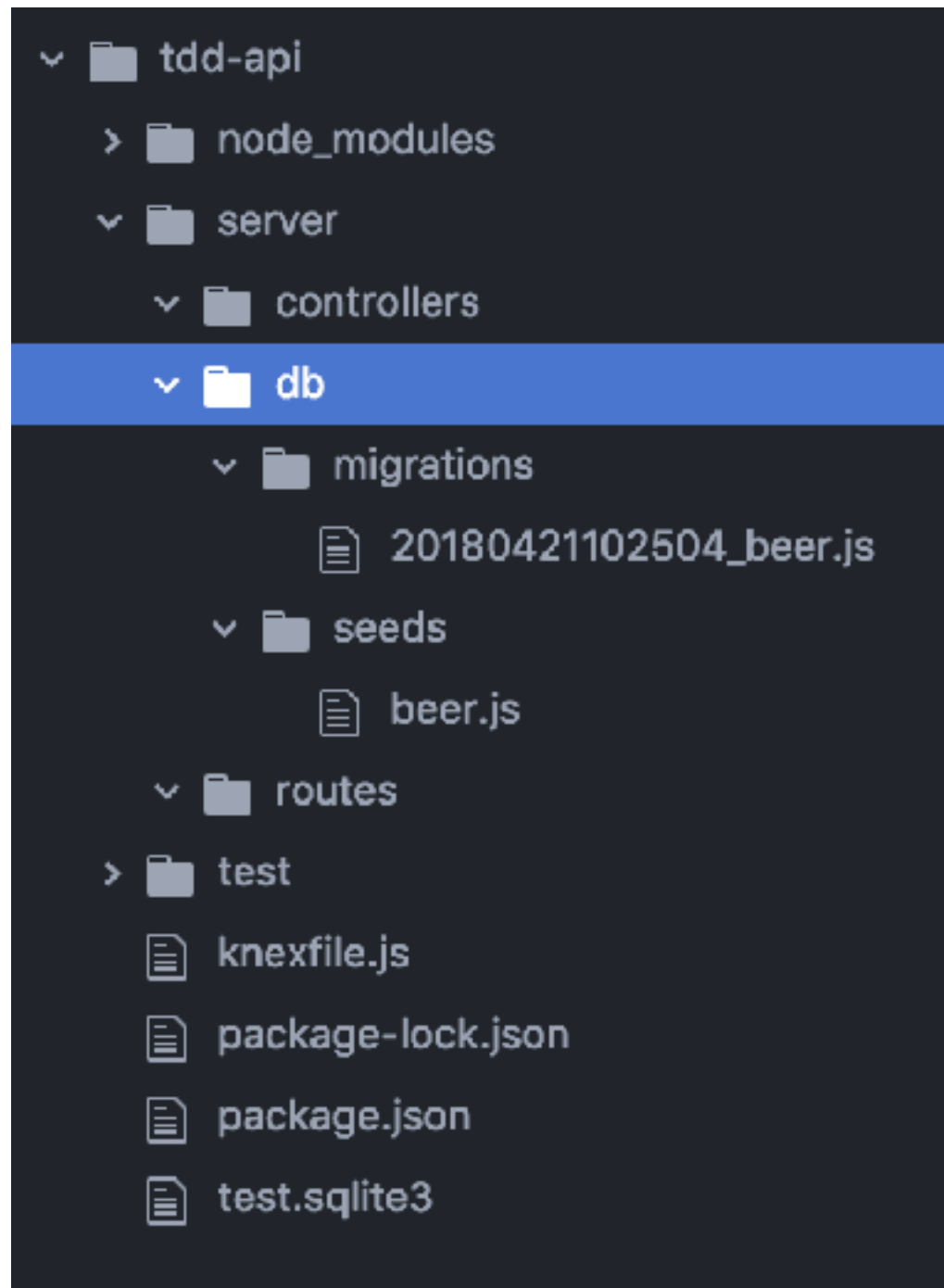
`$node_modules/knex/bin/cli.js seed:make beer --env test`

`$node_modules/knex/bin/cli.js seed:run --env test`

```
exports.seed = function(knex, Promise) {  
  // Deletes ALL existing entries  
  return knex('beer').del()  
    .then(function () {  
    // Inserts seed entries  
    return knex('beer').insert([  
      {id: 1, name: 'Another One', description: 'IPA', brand: 'IPA'},  
      {id: 2, name: 'Midway IPA', description: 'IPA', brand: 'IPA'},  
      {id: 3, name: 'All Day IPA', description: 'IPA', brand: 'IPA'}  
    ]);  
  });  
};
```



Project structure



3. Setup Testing



Testing Libraries

Mocha

Chai

Chai HTTP



Setup testing libraries

```
$npm i mocha chai chai-http --save-dev  
$mocha
```

```
0 passing (2ms)
```



Config package.json

\$npm test

```
{  
  "scripts": {  
    "test": "mocha --exit"  
  }  
}
```



Ready to start ...



Write first test case ?



First test case ?



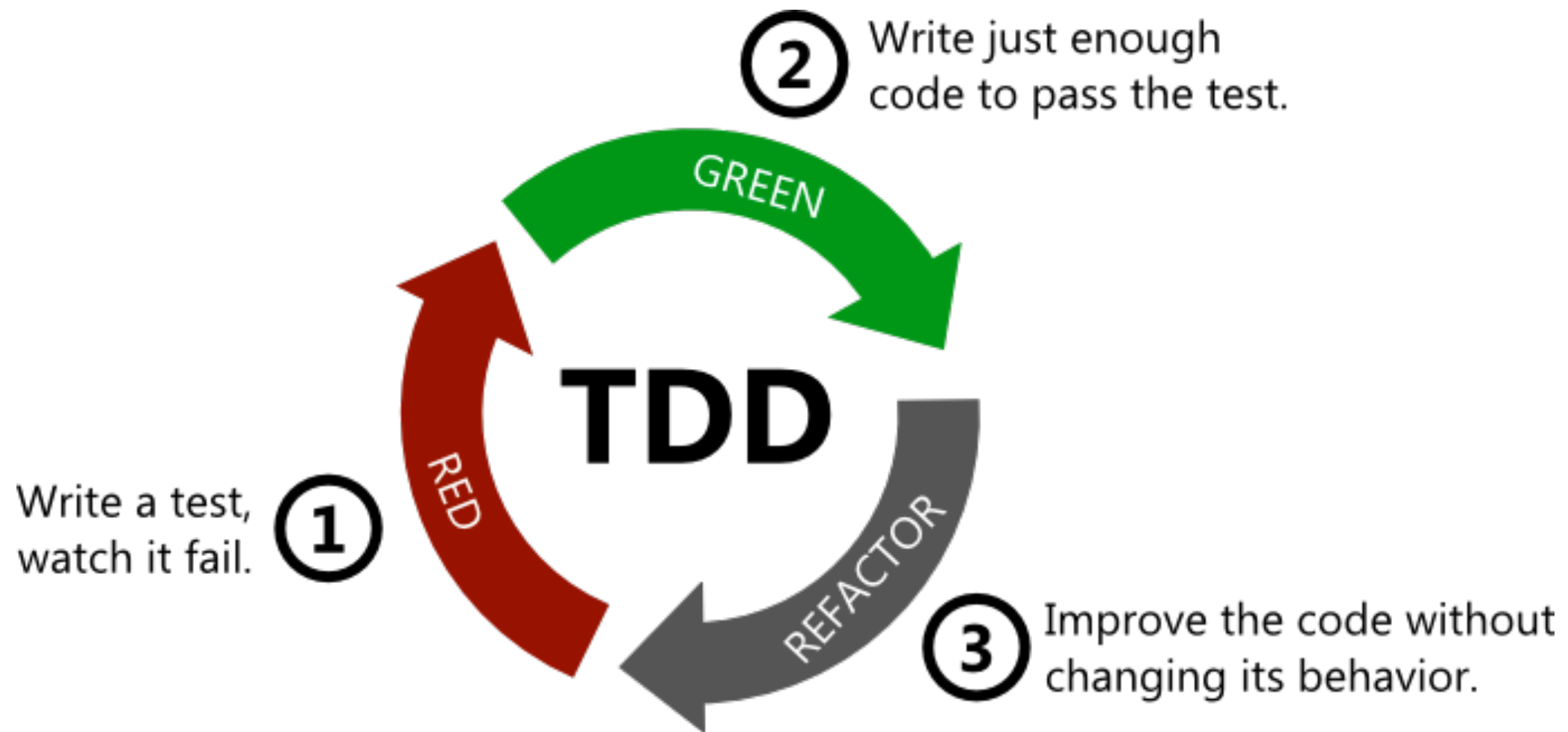
Acceptance Test

Input	Expected Result
GET /api/beer	HTTP Code = 200 Content Type = "application/json" Size of beer = 3 Beer ลำดับที่ 1 = Another One Beer ลำดับที่ 2 = Midway IPA Beer ลำดับที่ 3 = All Day IPA



Write first test case





Testing Libraries

Create file `/test/beer.list.test.js`



Configuration for test

```
// Configure the environment
const env = process.env.NODE_ENV || "test";
const config = require("../knexfile")[env];
const server = require("../server/index");
const knex = require("knex")(config);
```

```
// Assertion library
const chai = require("chai");
const should = chai.should();
const chaiHttp = require("chai-http");
chai.use(chaiHttp);
```

```
// For test
const PATH = "/api/beer";
```



Configuration for test

```
describe("routes of beer", () => {  
  
  beforeEach(() => {  
    return knex.migrate  
      .rollback()  
      .then(() => {  
        return knex.migrate.latest();  
      })  
      .then(() => {  
        return knex.seed.run();  
      });  
  });  
  
  afterEach(() => {  
    return knex.migrate.rollback();  
  });  
  
});
```



First test

```
describe("routes of beer", () => {  
  // First test case  
  describe(`Try to GET ${PATH}`, () => {  
    it("should return all beer from database", done => {  
      chai  
        .request(server)  
        .get(`${PATH}`)  
        .end((err, res) => {  
          should.not.exist(err);  
          res.status.should.eql(200);  
          res.type.should.eql("application/json");  
          res.body.data.length.should.eql(3);  
          res.body.data[0].should.include.keys("id", "name", "description", "brand");  
          done();  
        });  
    });  
  });  
});
```



Run test with RED

\$npm test

```
> tdd-api@1.0.0 test /Users/somkiat/data/coaching/SEAL/demo/tdd-api
> mocha
```

```
module.js:545
  throw err;
  ^
```

```
Error: Cannot find module '../server/index'
    at Function.Module._resolveFilename (module.js:543:15)
    at Function.Module.load (module.js:470:25)
    at Module.require (module.js:593:17)
    at require (internal/module.js:11:18)
    at Object.<anonymous> (/Users/somkiat/data/coaching/SEAL/demo/tdd-api/test/1
4:16)
    at Module._compile (module.js:649:30)
    at Object.Module._extensions.js (module.js:660:10)
    at Module.load (module.js:561:32)
    at tryModuleLoad (module.js:501:12)
    at Function.Module.load (module.js:493:3)
```

FAIL



Create file /server/index.js

\$npm test

routes of beer

Try to GET /api/beer

1) should return all beer from database

0 passing (127ms)

1 failing

1) routes of beer

Try to GET /api/beer

should return all beer from database:

TypeError: app.address is not a function

at serverAddress (node_modules/chai-http/lib/request.js:276:18)

at new Test (node_modules/chai-http/lib/request.js:265:53)

at Object.obj.(anonymous function) [as get] (node_modules/chai-http/lib/request.js:265:53)

at Context.done (test/beer.list.test.js:36:10)

FAIL



Try to develop your API ...



koa

next generation web framework for node.js

<http://koa.js.com/>



Resources

<http://koa.js.com/>

<https://github.com/alexmingoia/koa-router>



Install dependencies

```
$npm i koa koa-router
```



Step to develop /api/beer

1. Start API Server in file **index.js**
2. Create beer router in folder routes
3. Create beer controller



1. index.js

```
const Koa = require('koa');  
const beerRoutes = require('./routes/beer.routes');  
const app = new Koa();  
const PORT = process.env.PORT || 8081;  
  
app.use(beerRoutes.routes());  
  
const server = app.listen(PORT).on('error', err => {  
  console.error(err);  
});  
module.exports = server;
```



2. beer.routes.js

```
const Router = require('koa-router');
const router = new Router();
const beerController = require('../controllers/beerController');
const BASE_URL = '/api/beer';

router.get(BASE_URL, beerController.index);

module.exports = router;
```



3. beerController.js

```
const env = process.env.NODE_ENV || 'test';
const config = require('../../knexfile')[env];
const knex = require('knex')(config);

const index = async ctx => {
  try {
    const beers = await knex('beer').select();
    ctx.body = {
      data: beers,
    };
  } catch (error) {
    console.error(error);
  }
};

module.exports = {index};
```



routes of beer

Try to GET /api/beer

✓ should return all beer from database (49ms)

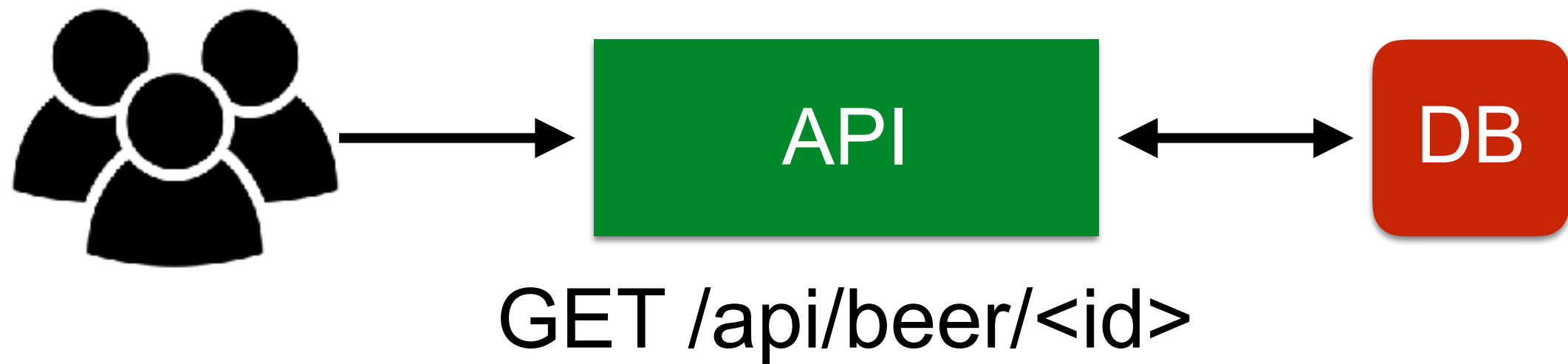
1 passing (153ms)



Add next test case ?



Next test case ?



Acceptance Test

Input	Expected Result
GET /api/beer/1	HTTP Code = 200 Content Type = "application/json" name = Another One description = IPA
GET /api/beer/100	HTTP Code = 404 Content Type = "application/json"



Case :: found

// Second test case

```
describe(`Try to GET ${PATH}/:id`, () => {
```

```
  it("should return a single beer", done => {
```

```
    chai
```

```
      .request(server)
```

```
      .get(`${PATH}/1`)
```

```
      .end((err, res) => {
```

```
        should.not.exist(err);
```

```
        res.status.should.eql(200);
```

```
        res.type.should.eql("application/json");
```

```
        res.body.data.length.should.eql(1);
```

```
        res.body.data[0].should.include.keys("id", "name", "description",
```

```
"brand");
```

```
        done();
```

```
      });
```

```
    });
```

```
  });
```



Run your test

\$npm test

routes of beer

Try to GET /api/beer

✓ should return all beer from database

Try to GET /api/beer/:id

1) should return a single beer

1 passing (198ms)

1 failing

1) routes of beer

Try to GET /api/beer/:id

should return a single beer:

Uncaught AssertionError: expected 404 to deeply equal 200

+ expected - actual

-404

+200

at chai.request.get.end (test/beer.list.test.js:57:29)

at Test.Request.callback (/node_modules/superagent/lib/node/index.js:706:17)

FAIL



Try to develop your API ...



routes of beer

Try to GET /api/beer

✓ should return all beer from database

Try to GET /api/beer/:id

✓ should return a single beer

2 passing (211ms)



Case :: beer not found

```
describe(`Try to GET ${PATH}/:id`, () => {  
  it("should return error when beer does not existed in database", done => {  
    chai  
      .request(server)  
      .get(`${PATH}/100`)  
      .end((err, res) => {  
        should.not.exist(err);  
        res.status.should.eql(404);  
        res.type.should.eql("application/json");  
        res.body.error.should.eql("Beer does not exists");  
        done();  
      });  
  });  
});
```



Run your test

\$npm test

routes of beer

Try to GET /api/beer

✓ should return all beer from database

Try to GET /api/beer/:id

1) should return error when beer does not existed in database

1 passing (197ms)

1 failing

1) routes of beer

Try to GET /api/beer/:id

should return error when beer does not existed in database:

Uncaught AssertionError: expected null to exist

at Object.should.exist (node_modules/chai/lib/chai/interface/sho

at chai.request.get.end (test/beer.list.test.js:56:18)

at Test.Request.callback (node_modules/superagent/lib/node/index

at IncomingMessage.parser (node_modules/superagent/lib/node/inde

at endReadableNT (_stream_readable.js:1101:12)

FAIL



Try to develop your API ...



routes of beer

Try to GET /api/beer

✓ should return all beer from database

Try to GET /api/beer/:id

✓ should return a single beer

✓ should return error when beer does not existed in database

3 passing (211ms)



Coding standard



Coding standard



Resources

<https://eslint.org/>

<https://github.com/prettier/prettier>



Install eslint

```
$npm i --save-dev eslint
```

```
$npm i --save-dev eslint-config-strongloop
```



Config package.json

\$npm run pretest

```
{  
  "scripts": {  
    "pretest": "eslint --ignore-path .gitignore .",  
    "test": "mocha --exit"  
  }  
}
```



Config .eslintrc.json

```
{  
  "extends": "strongloop",  
  "parserOptions": {  
    "ecmaVersion": 8  
  },  
  "rules": {  
    "max-len": [2, 120, 8]  
  }  
}
```



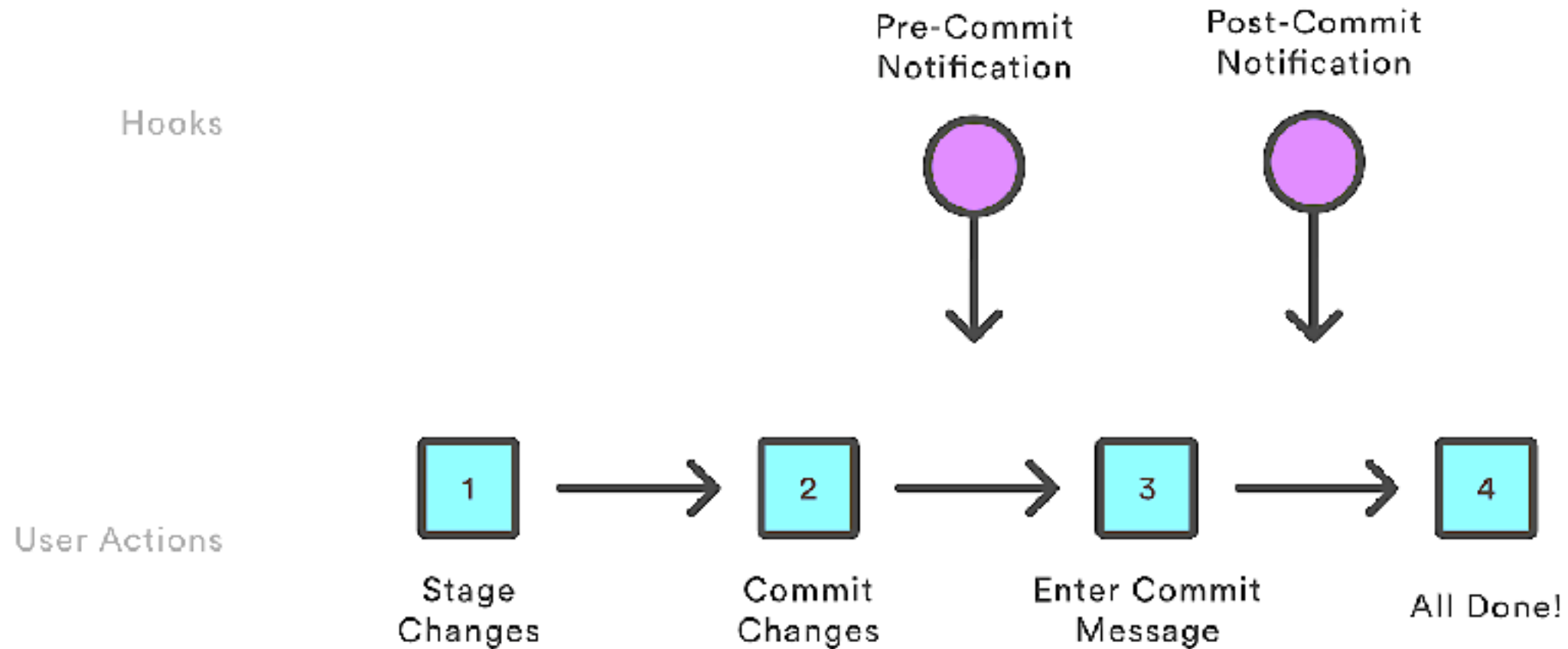
Run eslint

```
$npm run pretest
```

```
$npm run pretest -- --fix
```





Add to Git Hooks



Code coverage



Istanbul

Tutorials Advanced Features Contributing FAQ  GitHub

JavaScript test coverage made simple.

How Istanbul works

Istanbul instruments your ES5 and ES2015+ JavaScript code with line counters, so that you can track how well your unit-tests exercise your codebase.

The [nyc](#) command-line-client for Istanbul works well with most JavaScript testing frameworks: [tap](#), [mocha](#), [AVA](#), etc.

Features

- First class support of ES6/ES2015+ using [babel-plugin-istanbul](#).
- A collection of reporters, providing both terminal and HTML output:

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
All files	98.92	94.36	99.49	100	
yargs	99.17	93.95	100	100	
index.js	100	100	100	100	
yargs.js	99.15	93.86	100	100	
yargs/lib	98.7	94.33	99.87	100	

Quick Start

Adding coverage to your mocha tests could not be easier

```
$ npm install --save-dev nyc
```

Now, simply place the command `nyc` in front of your existing test command, for example:

```
{
  "scripts": {
    "test": "nyc mocha"
  }
}
```

<https://github.com/istanbuljs/nyc>



Use via nyc

```
$npm i --save-dev nyc
```

```
$npm i -g nyc
```

```
$nyc mocha
```



Use via nyc

\$nyc report

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	95.74	100	87.5	95.74	
tdd-api	100	100	100	100	
knexfile.js	100	100	100	100	
tdd-api/server	87.5	100	0	87.5	
index.js	87.5	100	0	87.5	8
tdd-api/server/controllers	94.44	100	100	94.44	
beerController.js	94.44	100	100	94.44	13
tdd-api/server/db/migrations	100	100	100	100	
20180421102504_beer.js	100	100	100	100	
tdd-api/server/db/seeds	100	100	100	100	
beer.js	100	100	100	100	
tdd-api/server/routes	100	100	100	100	
beer.routes.js	100	100	100	100	

<https://github.com/istanbuljs/nyc>



Use via nyc

\$nyc report --reporter=html

All files

95.74% Statements 45/47 100% Branches 6/6 87.5% Functions 7/8 95.74% Lines 45/47

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File		Statements	Branches	Functions	Lines				
tdd-api	<div><div></div></div>	100%	3/3	100%	0/0	100%	0/0	100%	3/3
tdd-api/server	<div><div></div></div>	87.5%	7/8	100%	2/2	0%	0/1	87.5%	7/8
tdd-api/server/controllers	<div><div></div></div>	94.44%	17/18	100%	4/4	100%	2/2	94.44%	17/18
tdd-api/server/db/migrations	<div><div></div></div>	100%	8/8	100%	0/0	100%	3/3	100%	8/8
tdd-api/server/db/seeds	<div><div></div></div>	100%	3/3	100%	0/0	100%	2/2	100%	3/3
tdd-api/server/routes	<div><div></div></div>	100%	7/7	100%	0/0	100%	0/0	100%	7/7



Use via nyc

[All files](#) / [tdd-api/server/controllers](#) beerController.js

94.44% Statements 17/18 100% Branches 4/4 100% Functions 2/2 94.44% Lines 17/18

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

```
1 'use strict';
2 1x const env = process.env.NODE_ENV || 'test';
3 1x const config = require('../../knexfile')(env);
4 1x const knex = require('knex')(config);
5
6 1x const index = async ctx => {
7 1x   try {
8 1x     const beers = await knex('beer').select();
9 1x     ctx.body = {
10     data: beers,
11   };
12   } catch (error) {
13     console.error(error);
14   }
15 };
16
17 1x const get = async ctx => {
18 2x   try {
19 2x     const {id} = ctx.params;
20 2x     const beer = await knex('beer')
21       .select()
22       .where({id});
23 2x     if (!beer.length) {
24 1x       throw new Error('Beer does not exists');
25     }
26 1x     ctx.body = {
27       data: beer,
28     };
29   } catch (error) {
30 1x     ctx.status = 404;
31 1x     ctx.body = {
32       error: error.message,
33     };
34   }
```



Config package.json

\$npm test

```
{  
  "scripts": {  
    "pretest": "eslint --ignore-path .gitignore .",  
    "test": "nyc mocha --exit"  
  }  
}
```



Automated Testing ?



Continuous Integration



Pipeline at Bitbucket

The screenshot shows the Bitbucket Pipelines interface for the repository 'demo-node'. The left sidebar contains navigation links: Overview, Source, Commits, Branches, Pull requests, Pipelines (selected), Deployments, Downloads, Boards, and Settings. The main content area displays a table of pipeline runs.

Pipeline	Status	Started	Duration
#3 Merge branch 'master' of bitbucket.org:somkiat/demo-node somkiat puisungnoen de7ef05 master	Successful	41 minutes ago	39 sec
#2 bitbucket-pipelines.yml edited online with Bitbucket somkiat puisungnoen 0b8075c master	Stopped	an hour ago	4 min 16 sec
#1 Initial Bitbucket Pipelines configuration somkiat puisungnoen 5675262 master	Failed	an hour ago	43 sec



Pipeline at Bitbucket

somkiat puisungnoen / demo-node / Pipelines

Pipeline #3

✓ Successful

Rerun

🕒 39 sec • 42 minutes ago

👤 Push by [somkiat puisungnoen](#)

Commit

>

🔗 [de7ef05](#)

Pipeline

[View configuration](#)

✓ Step 1

39s

Logs

⬇ Download raw ↗ ⋮

Build + Add a service or database

> Build setup	3s	✓
> npm install	14s	✓
> npm run pretest	1s	✓
> npm test	2s	✓
> Build teardown	13s	✓



Try to practice ...



<https://github.com/up1/workshop-tdd-api-with-nodejs/>

