JS QA automation framework for API

Table of Contents

1 Test automation theory

Lots of super boring stuff should be presented over here!

2 Linux basics

ls, cd, mkdir, cp, mv, rm, cat, nano, grep, regExp

3 Linux basics practice

ls, cd, mkdir, cp, mv, rm, cat, nano, grep, regExp

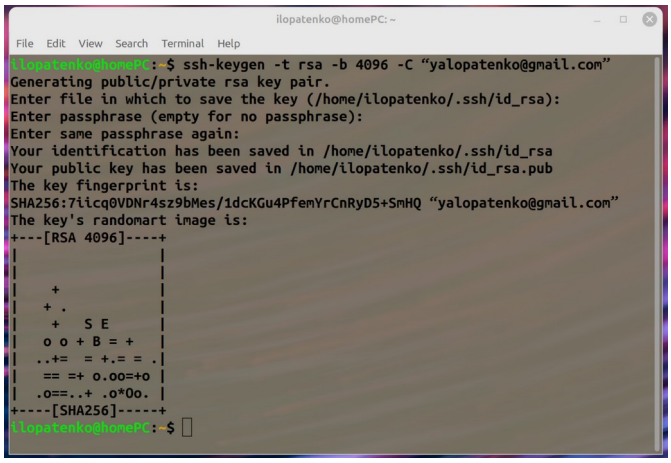
4 SSH keys

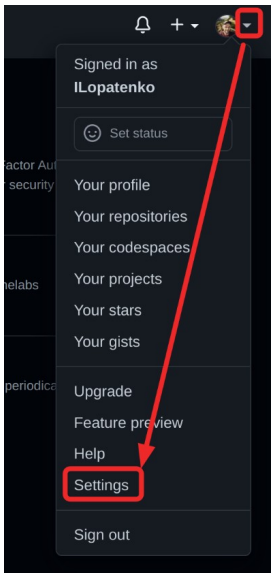
Open an terminal

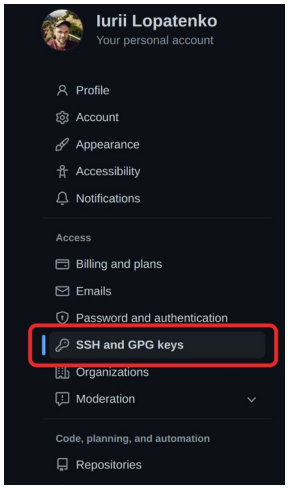
CD to a user home directory

Run next command:

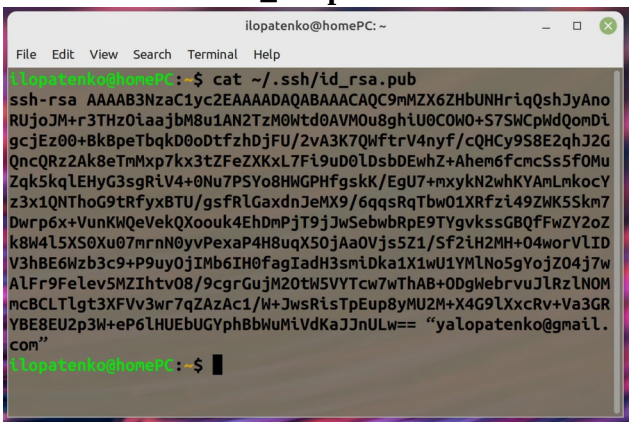
CLI=> **ssh-keygen -t rsa -b 4096 -C "ENTER_HERE_YOUR_EMAIL_OR_WHATEVER"**



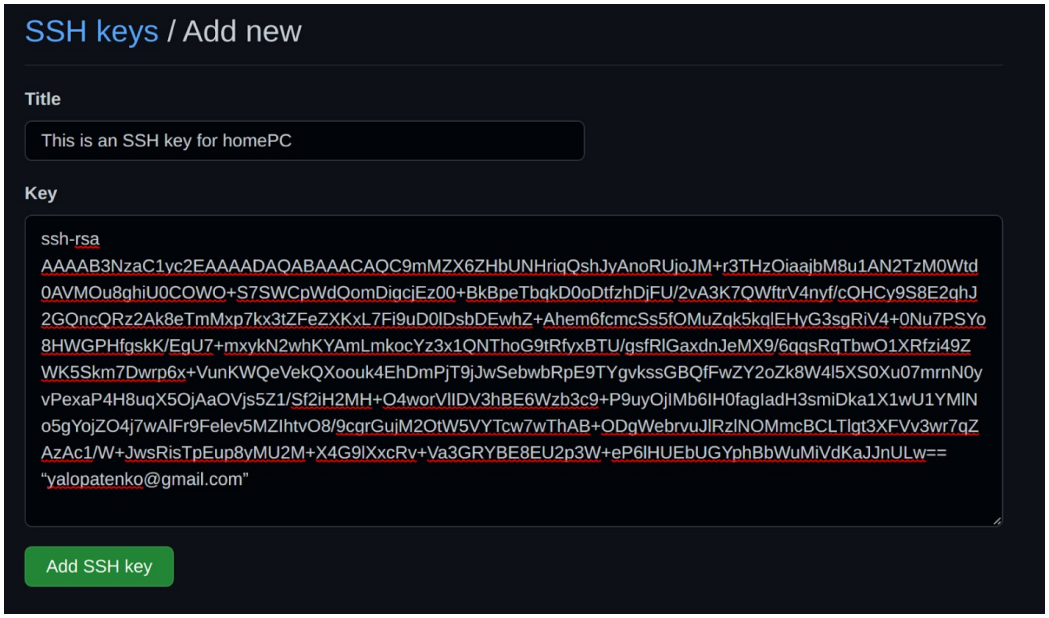


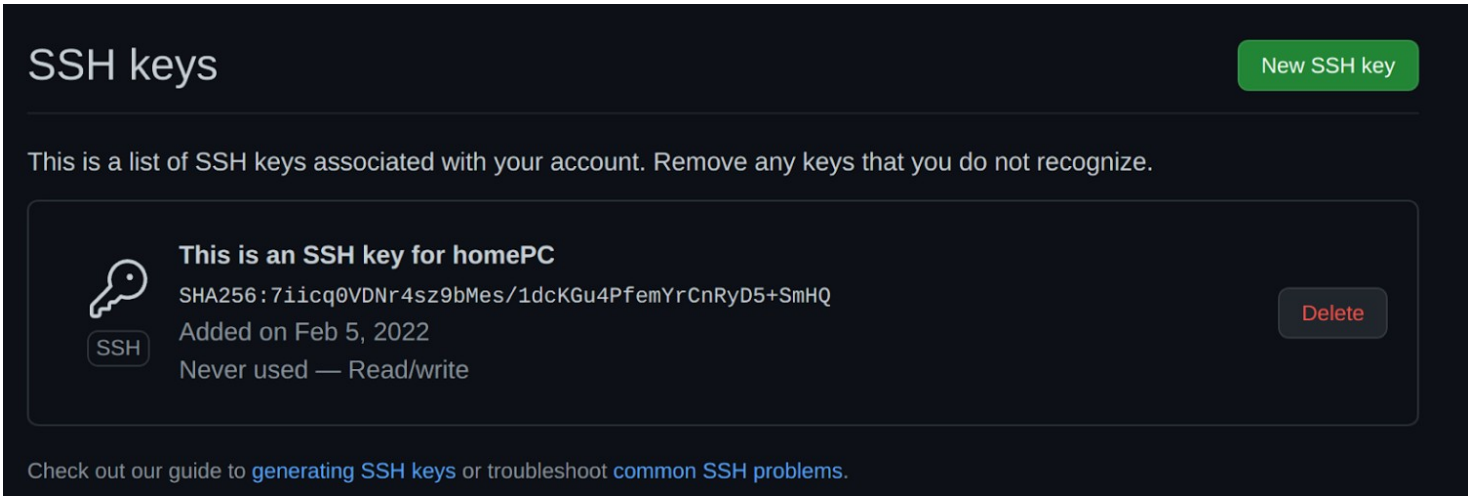


Go to gitHub. Go to settings https://github.com/settings/profile
Go to SSH and GPG keys https://github.com/settings/keys
Copy SSH key to buffer:
CLI=> **cat ~/.ssh/id_rsa.pub**



Click the button **Add SSH** and Insert an SSH key from buffer and click the button **Add SSH key** and confirm (re enter your gitHub password).





5 Test project setup

Create a new gitHub repository for a new project (JS automation framework for API testing)

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * **Repository name** *

ILopatenko / QAA-API-framework

Great repository names are short and memorable. Need inspiration? How about **solid-system**?

Description (optional)

JS test automation framework for API

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

- [x] **Add a README file**
This is where you can write a long description for your project. Learn more.

- [x] **Add .gitignore**
Choose which files not to track from a list of templates. Learn more.

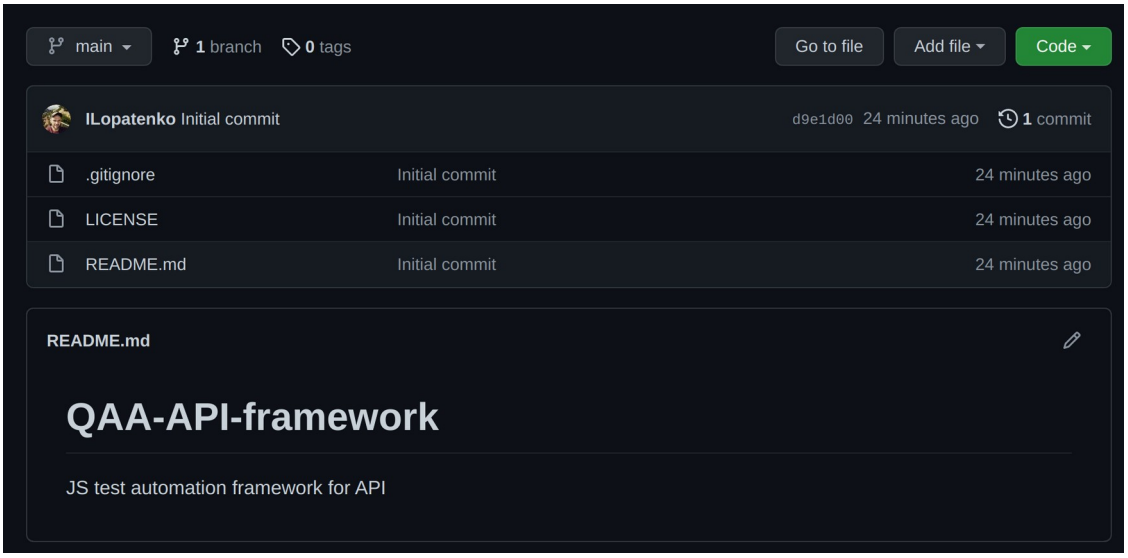
.gitignore template: Node

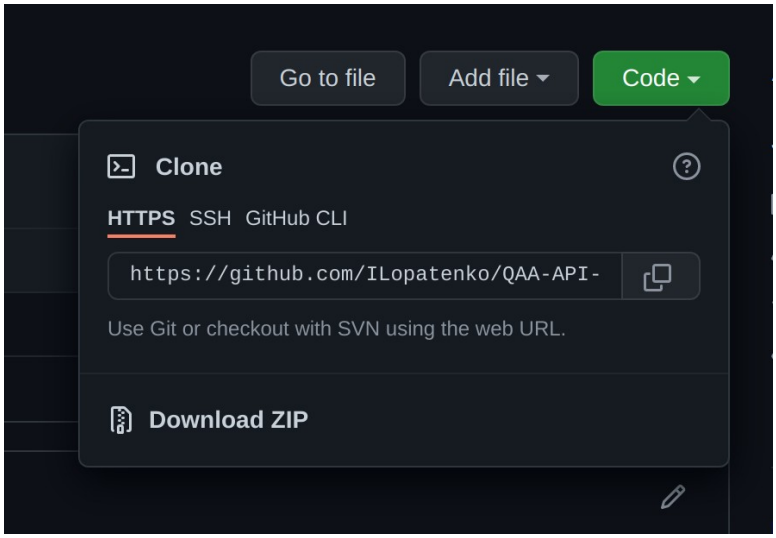
- [x] **Choose a license**
A license tells others what they can and can't do with your code. Learn more.

License: MIT License

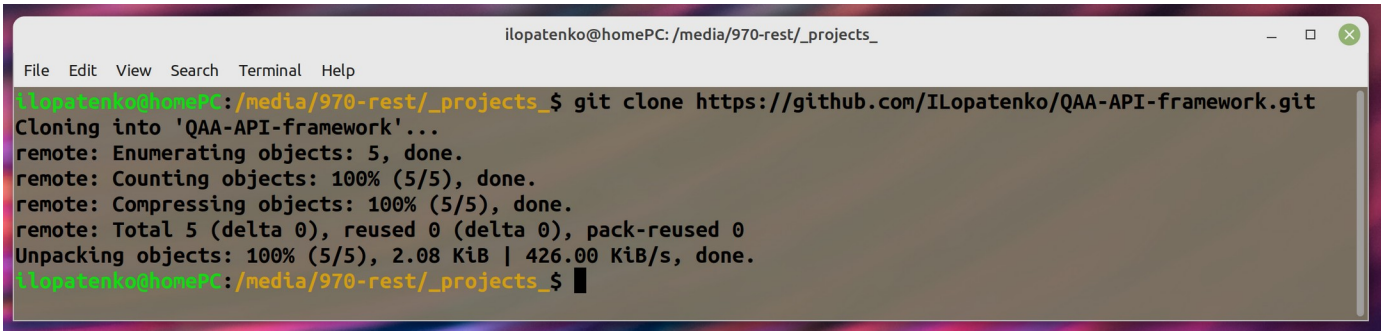
This will set `main` as the default branch. Change the default name in your settings.

Create repository

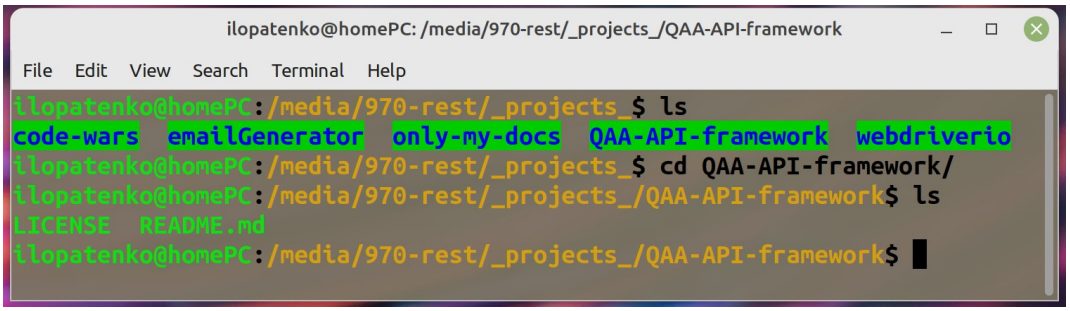




Copy your new repository's URL
Open terminal and **CD** to a directory where you want to keep your framework and clone it
CLI=> **git clone** https://github.com/ILopatenko/QAA-API-framework.git



CD into your project's folder



Initialize a new NPM project
CLI=> **npm init -y**

```
ilopatenko@homePC: /media/970-rest/_projects_/QAA-API-framework
File  Edit  View  Search  Terminal  Help
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ npm init -y
Wrote to /media/970-rest/_projects_/QAA-API-framework/package.json:

{
  "name": "qaa-api-framework",
  "version": "1.0.0",
  "description": "JS test automation framework for API",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/ILopatenko/QAA-API-framework.git"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/ILopatenko/QAA-API-framework/issues"
  },
  "homepage": "https://github.com/ILopatenko/QAA-API-framework#readme"
}


ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

6 Syntax for common test runners / Work with a new branch. Install BABEL. Pull request. First tests

https://dev.to/bormando/babel-setup-for-rest-api-tests-1dhf
Create a new branch babel-setup (and switch to it)
CLI=> **git checkout -B babel-setup**

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ git checkout -B babel-setup
Switched to a new branch 'babel-setup'
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

Install Babel
CLI=> **npm i -D @babel/cli @babel/core @babel/plugin-transform-runtime @babel/preset-env @babel/register**

Create a new file **.babelrc** in the root directory of your project
CLI=> **touch .babelrc**

And add this text

```
{
  "presets": ["@babel/preset-env"],
  "plugins": [
    ["@babel/transform-runtime"]
  ]
}
```

CLI=> **nano .babelrc**

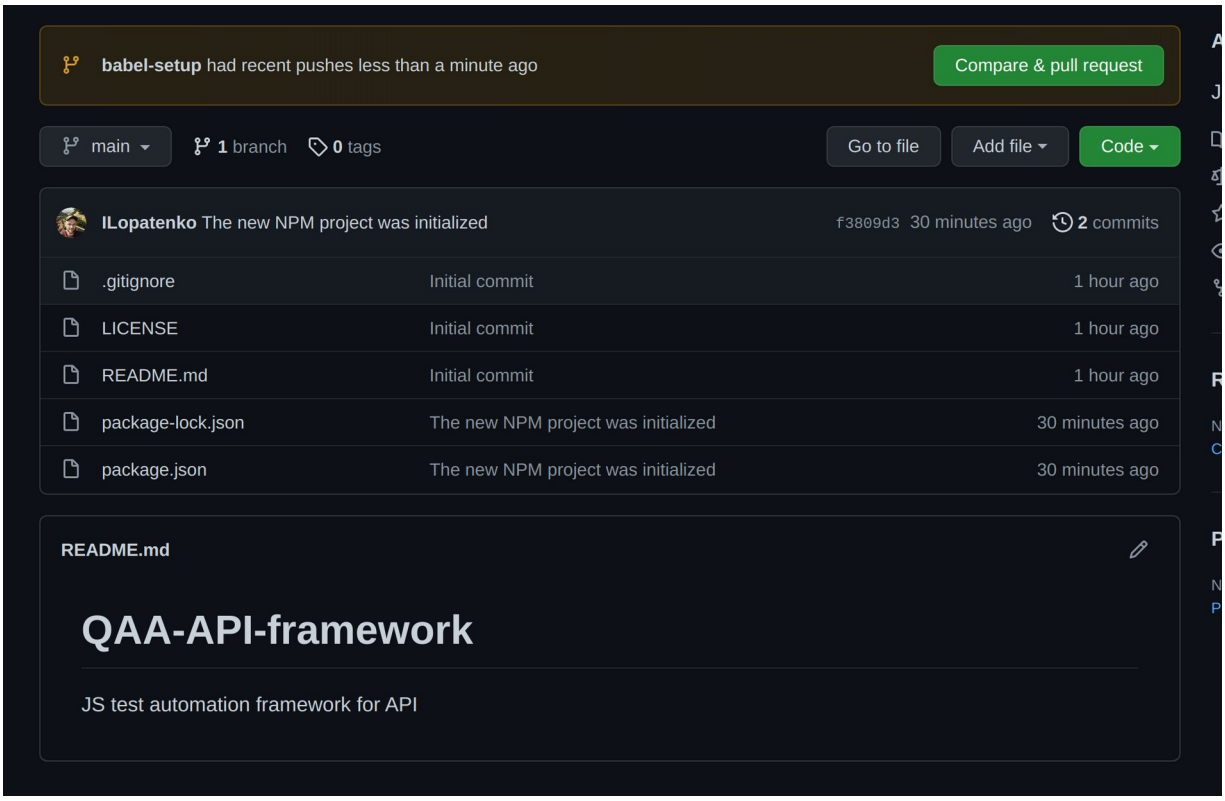
```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

  GNU nano 4.8
{
  "presets": ["@babel/preset-env"],
  "plugins": [
    ["@babel/transform-runtime"]
  ]
}
```

Now we can add all the changes to a STAGE. Create a new commit and push it to the repository.
CLI=> **git add .**
CLI=> **git commit -m 'Added Babel'**
CLI=> **git push --set-upstream origin babel-setup**

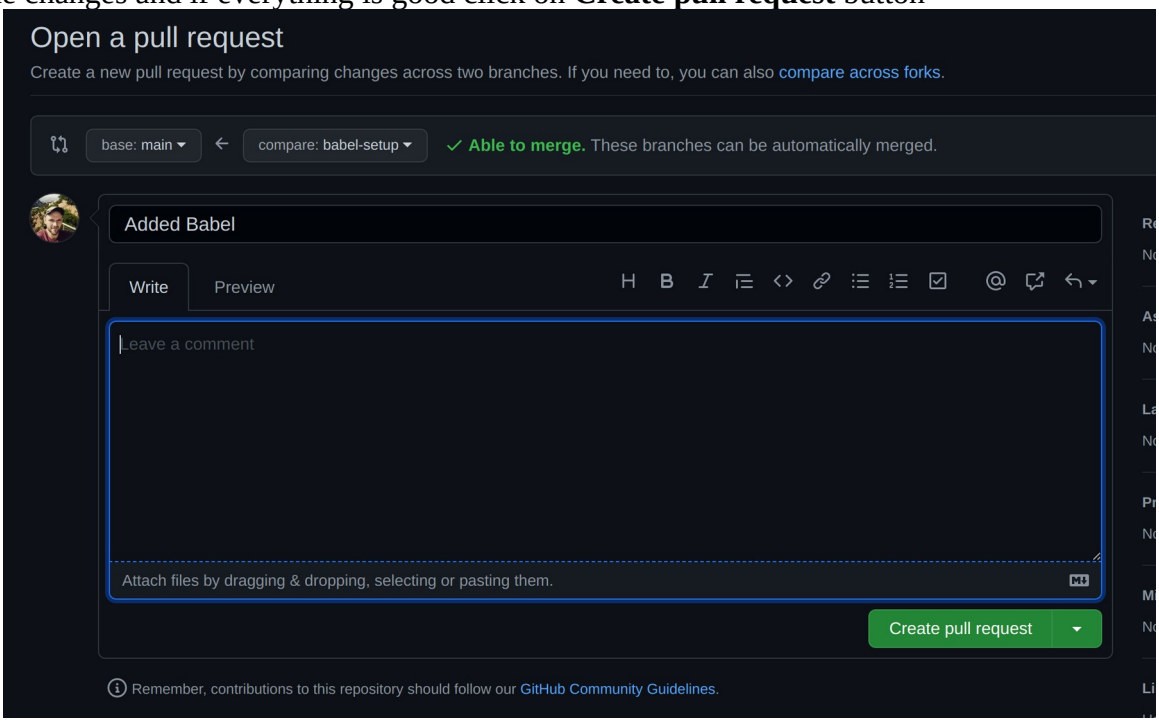
```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ git push --set-upstream origin babel-setup
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 44.15 KiB | 5.52 MiB/s, done.
Total 5 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'babel-setup' on GitHub by visiting:
remote:      https://github.com/ILopatenko/QAA-API-framework/pull/new/babel-setup
remote:
To https://github.com/ILopatenko/QAA-API-framework.git
 * [new branch]      babel-setup -> babel-setup
Branch 'babel-setup' set up to track remote branch 'babel-setup' from 'origin'.
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

Go to your gitHub repository

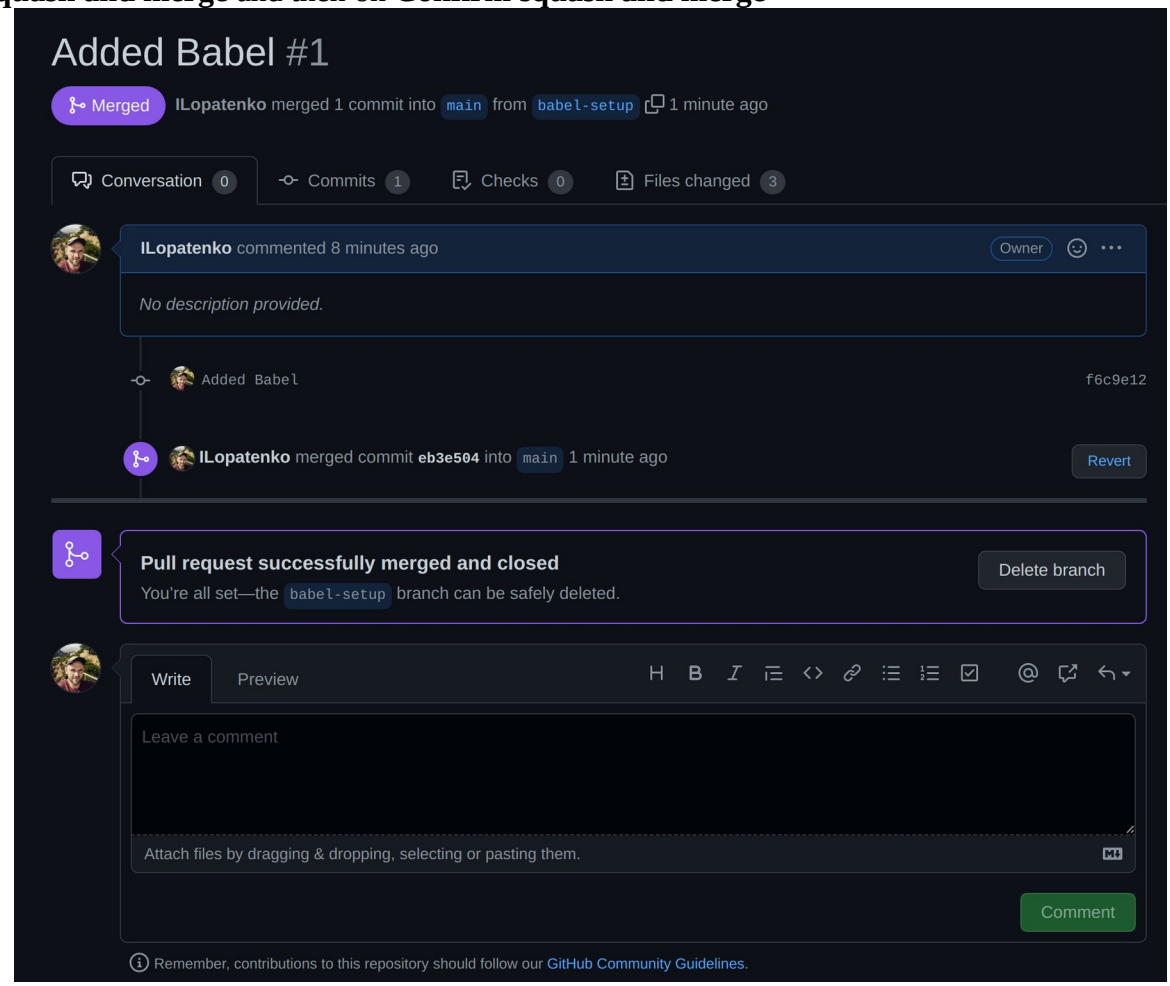


Click on **Compare & pull request** button

Check all the changes and if everything is good click on **Create pull request** button



Change default merge options to **Squash and merge**
Click on **Squash and merge** and then on **Confirm squash and merge**



Now we can delete branch **babel-setup** by click on **Delete branch** button
Change a branch to **main** in your IDE

```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

Pull all the changes to our local main branch from gitHub
CLI=> **git pull**

```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ git pull
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 1), reused 4 (delta 1), pack-reused 0
Unpacking objects: 100% (5/5), 44.52 KiB | 337.00 KiB/s, done.
From https://github.com/ILopatenko/QAA-API-framework
   f3809d3..eb3e504  main       -> origin/main
Updating f3809d3..eb3e504
Fast-forward
 .babelrc          |    6 +
 package-lock.json | 4866 ++++++++++++++++++++++++++++++++++++++++++++++++++++
 package.json      |    9 +-
 3 files changed, 4878 insertions(+), 3 deletions(-)
 create mode 100644 .babelrc
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

Create a new branch **first-tests**
CLI=> **git checkout -B first-tests**
Install Mocha and Chai packajes
CLI=> **npm i -D mocha chai**

```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ npm i -D mocha chai

added 62 packages, and audited 278 packages in 2s

31 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

Create a new directory **specs**
CLI=> **mkdir specs**
Create a new file **example.spec.js** in directory **specs**
CLI=> **touch specs/example.spec.js**
Open **specs/example.spec.js** with IDE import **Chai** and **Mocha** and start writing first tests

```
JS example.spec.js U ×
specs > JS example.spec.js > ...
1  import { expect } from 'chai';
2
3  describe('Math functions', () => {
4    it('A + B = C', () => {
5      const a = 4;
6      const b = 7;
7      const c = a + b;
8      expect(c).to.eq(11);
9    });
10 });
11
```

Open **package.json** file and make some changes in test script:

"scripts": {
  "test": "**npx mocha --config .mocharc.js**"
 },

Create a new file **.mocharc.js**
CLI=> **touch .mocharc.js**
And add next text

module.exports = {
 require: '@babel/register',
 spec: 'specs/**/*.spec.js',
};
Run a first test
CLI=> **npm run test**

```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ npm run test

> qaa-api-framework@1.0.0 test
> npx mocha --config .mocharc.js



  Math functions
    ✔ A + B = C


  1 passing (3ms)

ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

Create a new commit, push it to gitHub, Create a pull request, merge pull request and pull all the changes to a local computer

7 Syntax for HTTP client / First API tests

7.1 Installation Supertest

For this lesson I'm going to create a new branch in my project: first-api-tests and switch to the new branch

CLI=> **git checkout -b first-api-tests**

```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ git checkout -b first-api-tests
Switched to a new branch 'first-api-tests'
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

I'm going to use SUPERTEST as an HTTP client. Install the supertest as a DEV dependency.

CLI=> **npm i -D supertest**

```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ npm i -D supertest

added 28 packages, and audited 306 packages in 1s

37 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

7.2 Creating a new specification's file (spec file) for authorization tests

Delete example spec **example.spec.js**
Create a new file **auth.spec.js** inside **/QAA-API-framework/specs/**
CLI=> **touch specs/auth.spec.js**
Open this new file in VSCode and add next content:

```
//Imports block
import { expect } from 'chai';
import supertest from 'supertest';
//
//describe is a test suit that can contain another test sub suits or test cases
describe('Auth', () => {
  //Create a request object
  const request = supertest('https://paysis.herokuapp.com');

  //it is a test case with a couple of assertions
  it('Successful login (happy path, positive tests)', () => {
    //Use inner methods to add all the parameters to a request using chaining methods
    request
      //Setup a request method - POST and an endpoint - /auth
      .post('/auth')
      //Setup payload - object with 2 keys - login and password (and their values)
      .send({ login: 'adminius', password: 'supers3cret' })
      //Tests block that can handle error or response
      .end((err, res) => {
        //Test that status code in response equals to 200
        expect(res.statusCode).to.eq(200);
        //Test that response has body, body has token key and that token key is not undefined
        expect(res.body.token).not.to.be.undefined;
      })});
```

```
//it is a test case with a couple of assertions
it('Unsuccessful login (unhappy path, negative tests)', () => {
  //Use inner methods to add all the parameters to a request using chaining methods
  request
    //Setup a request method - POST and an endpoint - /auth
    .post('/auth')
    //Setup payload - object with 2 keys - login and password (and their values)
    .send({ login: 'wrongLogin', password: 'wrongPassword' })
    //Tests block that can handle error or response
    .end((err, res) => {
      //Test that status code in response equals to 404
      expect(res.statusCode).to.eq(404);
      //Test that error message equals to 'Wrong login or password.'
      expect(res.body.message).to.eq('Wrong login or password.')})})});
```

```
JS auth.spec.js U ×
specs > JS auth.spec.js > ...
1  //Imports block
2  import { expect } from 'chai';
3  import supertest from 'supertest';
4  //
5  //describe is a test suit that can contain another test sub suits or test cases
6  describe('Auth', () => {
7    //Create a request object
8    const request = supertest('https://paysis.herokuapp.com');
9
10   //it is a test case with a couple of assertions
11   it('Successfull login (happy path, positive tests)', () => {
12     //Use inner methods to add all the parameters to a request using chaining methods
13     request
14       //Setup a request method - POST and an endpoint - /auth
15       .post('/auth')
16       //Setup payload - object with 2 keys - login and password (and their values)
17       .send({ login: 'adminius', password: 'supers3cret' })
18       //Tests block that can handle error or response
19       .end((err, res) => {
20         //Test that status code in response equals to 200
21         expect(res.statusCode).to.eq(200);
22         //Test that response has body, body has token key and that token key is not undefined
23         expect(res.body.token).not.to.be.undefined;
24       });
25   });
26
27   //
28   //it is a test case with a couple of assertions
29   it('Unsuccessfull login (unhappy path, negative tests)', () => {
30     //Use inner methods to add all the parameters to a request using chaining methods
31     request
32       //Setup a request method - POST and an endpoint - /auth
33       .post('/auth')
34       //Setup payload - object with 2 keys - login and password (and their values)
35       .send({ login: 'wrongLogin', password: 'wrongPassword' })
36       //Tests block that can handle error or response
37       .end((err, res) => {
38         //Test that status code in response equals to 404
39         expect(res.statusCode).to.eq(404);
40         //Test that error message equals to 'Wrong login or password.'
41         expect(res.body.message).to.eq('Wrong login or password.');
42       });
43   });
44 });
45
```

Run all the specs with 1 command:
CLI=> **npm run test**

```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ npm run test

> qaa-api-framework@1.0.0 test
> npx mocha --config .mocharc.js


  Auth
    ✔ Successfull login (happy path, positive tests)
    ✔ Unsuccessfull login (unhappy path, negative tests)


  2 passing (16ms)

ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

So far so good. Make a commit, push it to gitHub, create a pull request, merge it. Pull all the changes.

8 Environment variables

For this lesson I'm going to create a new branch in my project: **env-vars** and switch to the new branch

CLI=> **git checkout -b env-vars**

8.1 Installation dotenv npm package

Dotenv is a zero-dependency module that loads environment variables from a .env file into process.env.

CLI=> **npm i -D dotenv**
Create a new file **.env** in root project's folder to store all the global variables
CLI=> **touch .env**

Add **.env** file to **.gitignore** file

```
.gitignore
.gitignore
69  .yarn-integrity
70
71  # dotenv environment variables file
72  .env
73  .env.test
```

8.1 Work with environment variables

Add all the variables (with their values) to **.env** file:

```
.env
.env
1  BASE_URL="https://paysis.herokuapp.com"
2  LOGIN="adminius"
3  PASSWORD="supers3cret"
```

Now I'm going to import dotenv package to my specs and get rid of all hard coded values inside all the specs.

```
JS auth.spec.js M
specs > JS auth.spec.js > describe('Auth') callback
You, seconds ago | 2 authors (Iurii Lopatenko and others)
1  //Imports block
2  import { expect } from 'chai';
3  import supertest from 'supertest';
4  import 'dotenv/config';
5  //
6  //describe is a test suit that can contain another test sub suits or test cases
7  describe('Auth', () => {
8    //Create a request object
9    const request = supertest(process.env.BASE_URL);
10
11   //it is a test case with a couple of assertions
12   it('Successfull login (happy path, positive tests)', () => {
13     //Use inner methods to add all the parameters to a request using chaining methods
14     request
15       //Setup a request method - POST and an endpoint - /auth
16       .post('/auth')
17       //Setup payload - object with 2 keys - login and password (and their values)
18       .send({ login: process.env.LOGIN, password: process.env.PASSWORD })
19       //Tests block that can handle error or response
20       .end((err, res) => {
21         //Test that status code in response equals to 200
22         expect(res.statusCode).to.eq(200);
23         //Test that response has body, body has token key and that token key is not undefined
24         expect(res.body.token).not.to.be.undefined;
25       });
26   });
27
```

```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ npm run test

> qaa-api-framework@1.0.0 test
> npx mocha --config .mocharc.js



  Auth
    ✔ Successfull login (happy path, positive tests)
    ✔ Unsuccessfull login (unhappy path, negative tests)


  2 passing (16ms)

ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

Make a commit, push, merge pull request, pull all the changes to local main branch.

9 Test runner (mocha) hooks

For this lesson I'm going to create a new branch in my project: **env-vars** and switch to the new branch

CLI=> **git checkout -b mocha-hooks**

9.1 Hooks

```
JS auth.spec.js M ×
specs > JS auth.spec.js > ...
You, seconds ago | 2 authors (You and others)
1  //Imports block
2  import { expect } from 'chai';
3  import supertest from 'supertest';
4  import 'dotenv/config';
5  //This describe is a main test suite
6  describe('Auth test suite', () => {
7    //Create a request object
8    const request = supertest(process.env.BASE_URL);
9    //Create a new variable to store a response from server
10   let result;
11   describe('Successful login sub suite (happy path, positive tests)', () => {
12     //BEFORE hook - will be runned 1st (before all other suits/tests)
13     //this hook will send a request to a server and save a response from a server to a request variable
14     before(async () => {
15       //Send async request
16       await request
17         //Setup a request method - POST and an endpoint - /auth
18         .post('/auth')
19         //Setup payload - object with 2 keys - login and password (and their values)
20         .send({ login: process.env.LOGIN, password: process.env.PASSWORD })
21         //Save a response from server to result variable
22         .then((res) => {
23           result = res;
24         });
25     });
26     it('Checking that response status code is 200', () => {
27       expect(result.statusCode).to.eq(200);
28     });
29     it('Checking that response contains an authorization token', () => {
30       expect(result.body.token).not.to.be.undefined;
31     });
32   });
33   describe('Unsuccessful login sub suite (unhappy path, negative tests)', () => {
34     //BEFORE hook - will be runned 1st (before all other suits/tests)
35     //this hook will send a request to a server and save a response from a server to a request variable
36     before(async () => {
37       //Send async request
38       await request
39         //Setup a request method - POST and an endpoint - /auth
40         .post('/auth')
41         //Setup payload - object with 2 keys - login and password (and their values)
42         .send({ login: 'tralala', password: 'trulala' })
43         //Save a response from server to result variable
44         .then((res) => {
45           result = res;
46         });
47     });
48     it('Checking that response status code is 404', () => {
49       expect(result.statusCode).to.eq(404);
50     });
51     it('Checking that error message is "Wrong login or password."', () => {
52       expect(result.body.message).to.eq('Wrong login or password.');
53     });
54   });
55 });
56
```

```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ npm run test

> qaa-api-framework@1.0.0 test
> npx mocha --config .mocharc.js



  Auth test suite
    Successful login sub suite (happy path, positive tests)
      ✔ Checking that response status code is 200
      ✔ Checking that response contains an authorization token
    Unsuccessful login sub suite (unhappy path, negative tests)
      ✔ Checking that response status code is 404
      ✔ Checking that error message is "Wrong login or password."


  4 passing (772ms)

ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ +
```

9.2 Global hooks

Create a new directory **config** and a new file **setup.js** inside to store global mocha hooks
CLI=> **mkdir config && touch config/setup.js**

I'm going to move successful log in action to a global Mocha hook (to be able so save a token from response to a .env variable)

```
setup.js U
config > setup.js > ...
import 'dotenv/config';
import supertest from 'supertest';

before(async () => {
  //Create a request object
  const request = await supertest(process.env.BASE_URL);
  await request
    //Setup a request method - POST and an endpoint - /auth
    .post('/auth')
    //Setup payload - object with 2 keys - login and password (and their values)
    .send({ login: process.env.LOGIN, password: process.env.PASSWORD })
    //Save a response from server to result variable
    .then((res) => {
      process.env['TOKEN'] = res.body.token;
    });
});
```

```
auth.spec.js M
specs > auth.spec.js > describe('Auth test suite') callback > describe('Unsuccessful login sub suite (unhappy path, negativ
You, seconds ago | 2 authors (You and others)
import { expect } from 'chai';
import supertest from 'supertest';
//This describe is a main test suite
describe('Auth test suite', () => {
  //Create a request object
  const request = supertest(process.env.BASE_URL);
  //Create a new variable to store a response from server
  let result;
  describe('Successful login sub suite (happy path, positive tests)', () => {
    //BEFORE hook - will be runned 1st (before all other suits/tests)
    before(async () => {
      //Send async request
      await request
        //Setup a request method - POST and an endpoint - /auth
        .post('/auth')
        //Setup payload - object with 2 keys - login and password (and their values)
        .send({ login: process.env.LOGIN, password: process.env.PASSWORD })
        //Save a response from server to result variable
        .then((res) => {
          result = res;
        });
    });
    it('Checking that response status code is 200', () => {
      expect(result.statusCode).to.eq(200);
    });
    it('Checking that response contains an authorization token', () => {
      expect(result.body.token).not.to.be.undefined;
    });
  });
  describe('Unsuccessful login sub suite (unhappy path, negative tests)', () => {
    //BEFORE hook - will be runned 1st (before all other suits/tests)
    before(async () => {
      //Send async request
      await request
        //Setup a request method - POST and an endpoint - /auth
        .post('/auth')
        //Setup payload - object with 2 keys - login and password (and their values)
        .send({ login: 'tralala', password: 'trulala' })
        //Save a response from server to result variable
        .then((res) => {
          result = res;
        })});
    it('Checking that response status code is 404', () => {
      expect(result.statusCode).to.eq(404)});
    it('Checking that error message is "Wrong login or password."', () => {
      expect(result.body.message).to.eq('Wrong login or password.')})})});
```

10 Wrappers (helpers) for API tests

11 API tests practice

12 Setting up a mock server

13 CI/CD theory

14 Integrating API tests with GitHub Actions