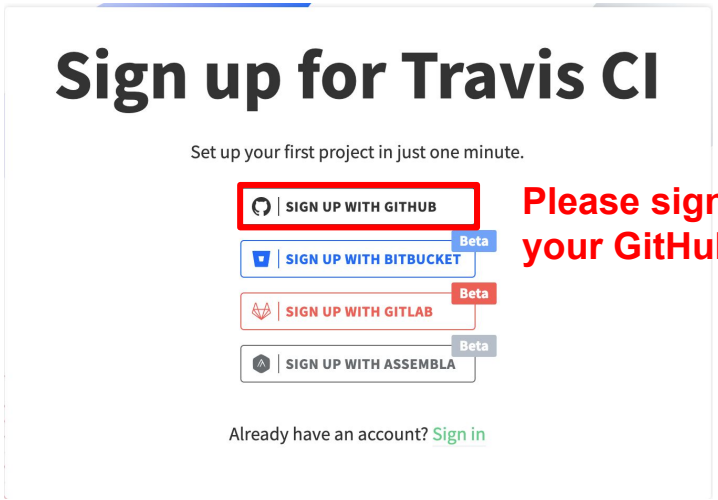
SWPP Practice Session #8

FE-BE Integration & CI Environment Setup

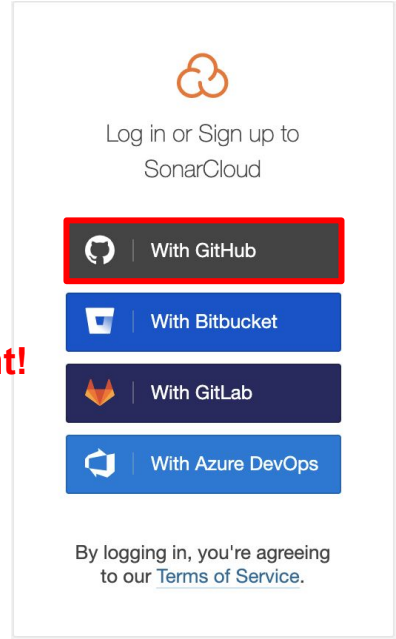
2022 Oct 26

Set Up

- Create your TravisCI account
  - https://app.travis-ci.com/signup
- Create your SonarCloud account
  - https://sonarcloud.io/sessions/new



Please sign-up with your GitHub account!



Set Up

- Please fork the repository for today's practice: https://github.com/swpp22fall-practice-sessions/swpp-p8-integration-practice
- Please Install travis cli:
  - For Ubuntu:
    - ```
      $ sudo apt install ruby-full
      $ gem install travis
      ```
  - For Mac:
    - `$ brew install travis`
    - If you are using Mac with an M1 chip and failed to install `travis`, the link below may help troubleshooting: https://blog.dalso.org/article/m1-mac-brew-error-cannot-install-in-homebrew-on-arm-processor-in-intel-default-prefix
      ```

Setup - docker container

- run docker container

```
$ docker run --rm -it \
     --ipc=host \
     --name "practice8" \
     -p 0.0.0.0:3000:3000 -p 0.0.0.0:8000:8000 \
     -v ${PWD}:/home \
     snuspl/swpp:practice8
```

Announcement

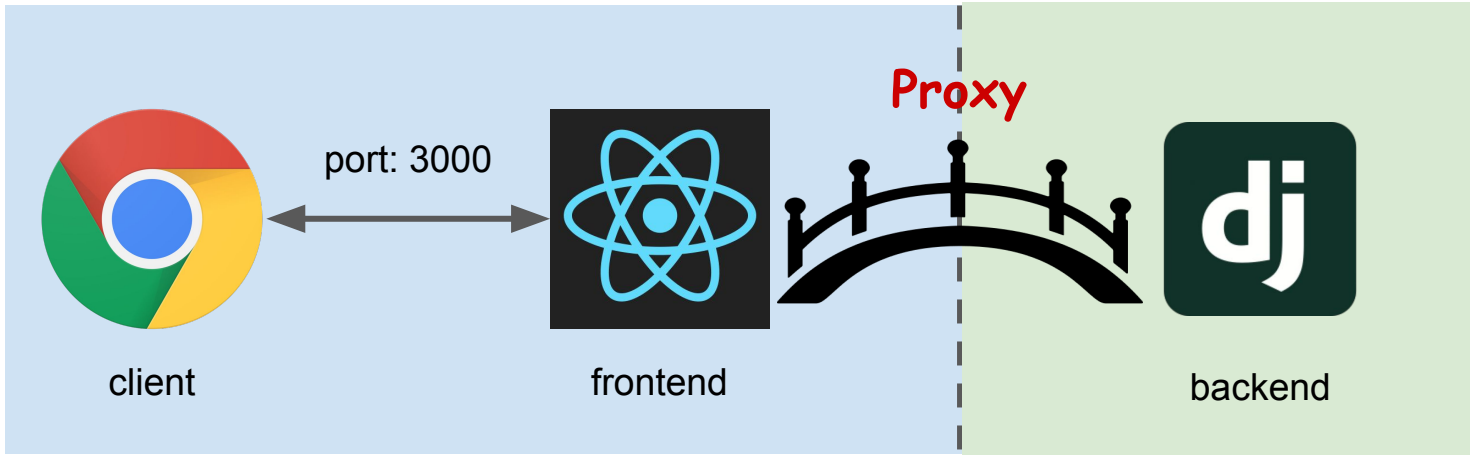
- Sprint #2
  - Don’t forget to submit the following documents.
    - (1) Design and planning (wiki, pdf)
    - (2) Sprint #2 backlog (pdf)
    - Due: **10/29(Sat) 6pm**
    - **Send the materials to swpp.22.staff@spl.snu.ac.kr**
- Homework 4
  - Due: **11/2(Wed) 6pm**
  - Don’t forget to add `**swpp-tas**` as your collaborator.

Today’s Tasks

- **One Individual Task**
  - To submit today’s practice assignment, fork the following repo.
    - https://github.com/swpp22fall-practice-sessions/swpp-p8-integration-practice
  - Implement today’s tasks and send PR.
  - Due: 10/27 (Thu) 9pm
- **One Team Task**
  - Setup Travis CI and SonarCloud and create a pull request in your team repo.
  - Set `jaewooMaeng` as your reviewer.
  - Due: 10/27 (Thu) 9pm

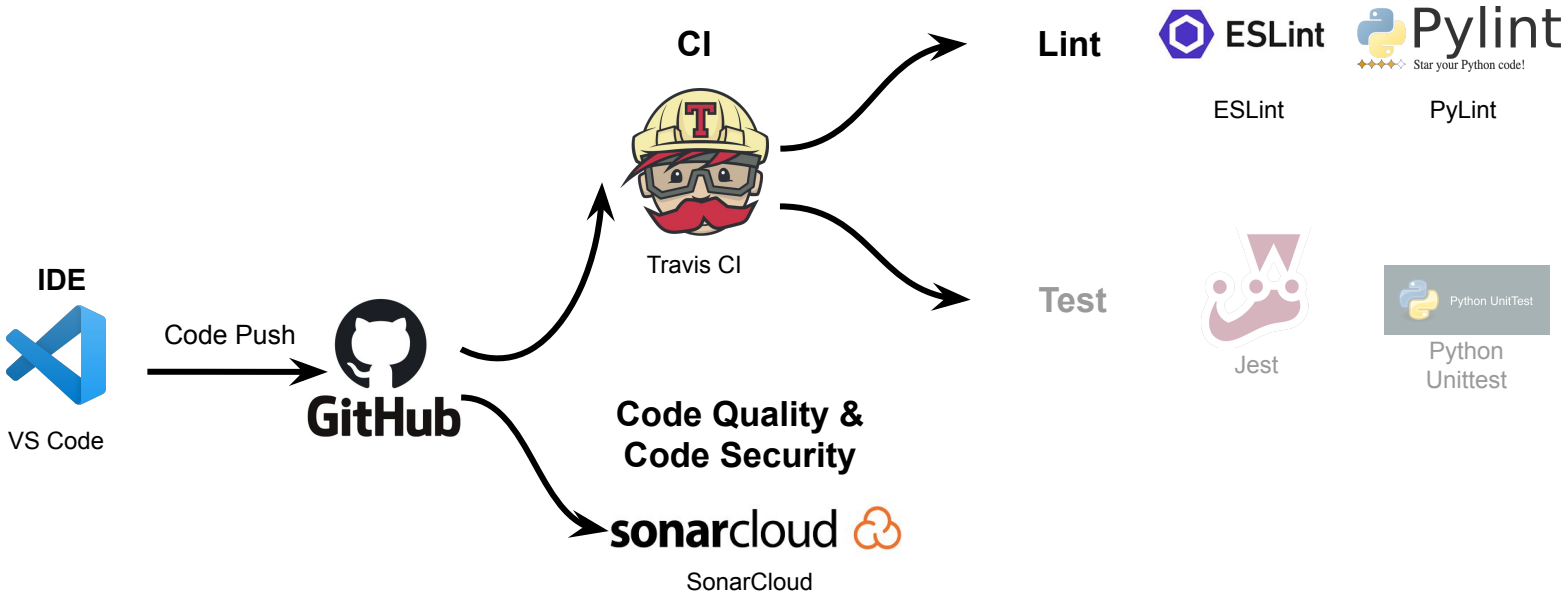
Today’s Contents

1. Frontend-Backend Integration
    a. Connect frontend and backend with Proxy server
    b. Pass CSRF tokens from frontend to backend



Today's Contents

2. Continuous Integration(CI) and code analysis tool setup
   a. Automate static code analysis using Lint tools through Travis CI
   b. Automate static code quality analysis with SonarCloud



Frontend-Backend Integration

Frontend-Backend Integration

What we will learn

1. How to connect frontend and backend through proxy server
2. How to handle CSRF token in frontend side.
    a. React / Axios
    b. React / Fetch
    c. React Form

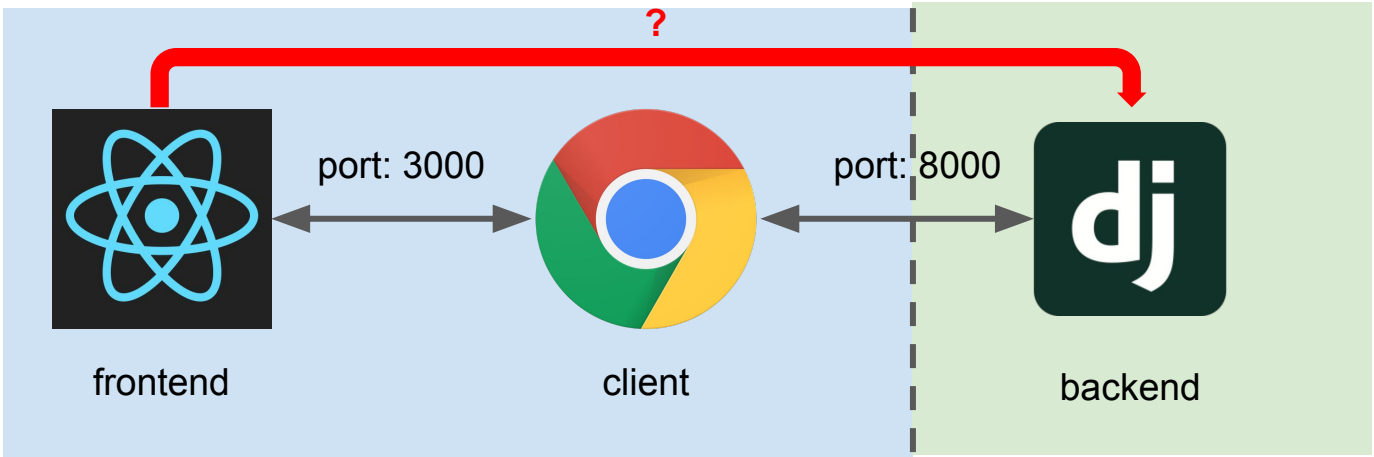
Frontend-Backend Integration

What we will learn

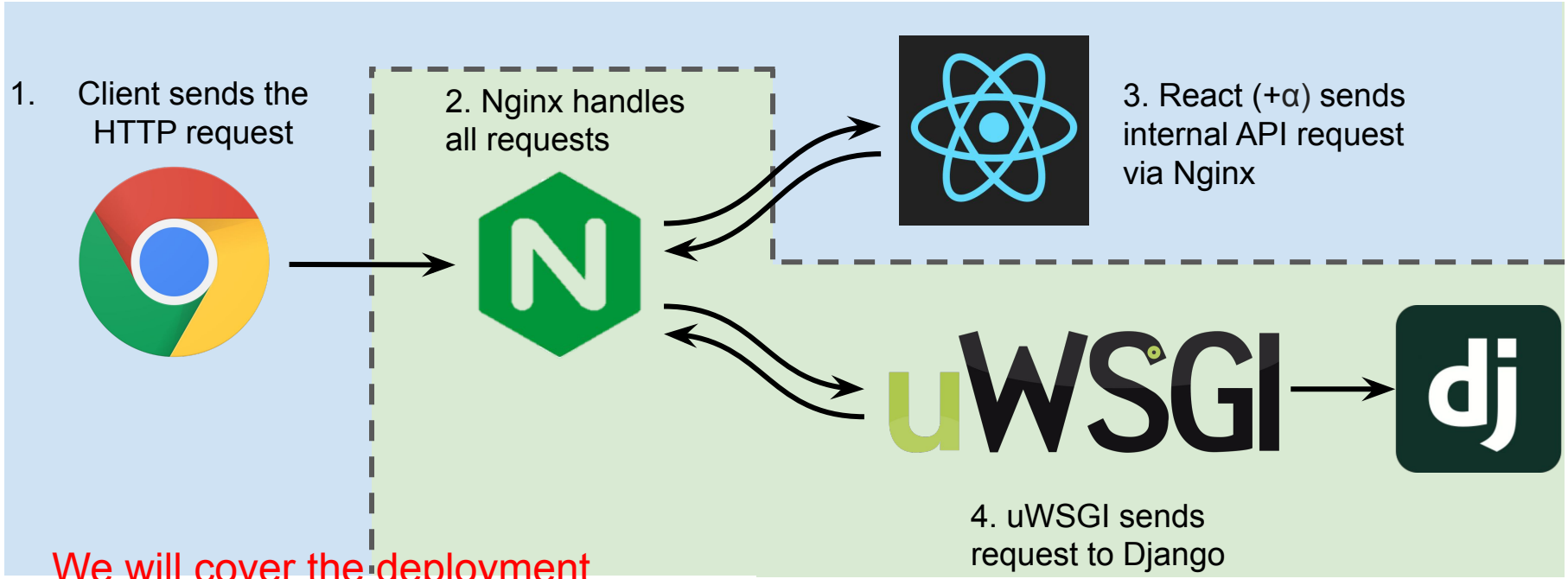
1. How to connect frontend and backend through proxy server
2. How to handle CSRF token in frontend side.
    a. React / Axios
    b. React / Fetch
    c. React Form

How to connect React and Django?

- Now you can make both frontend and backend of a website.
- How to use backend API in frontend side?
  - Django → port: 8000
  - React → port: 3000



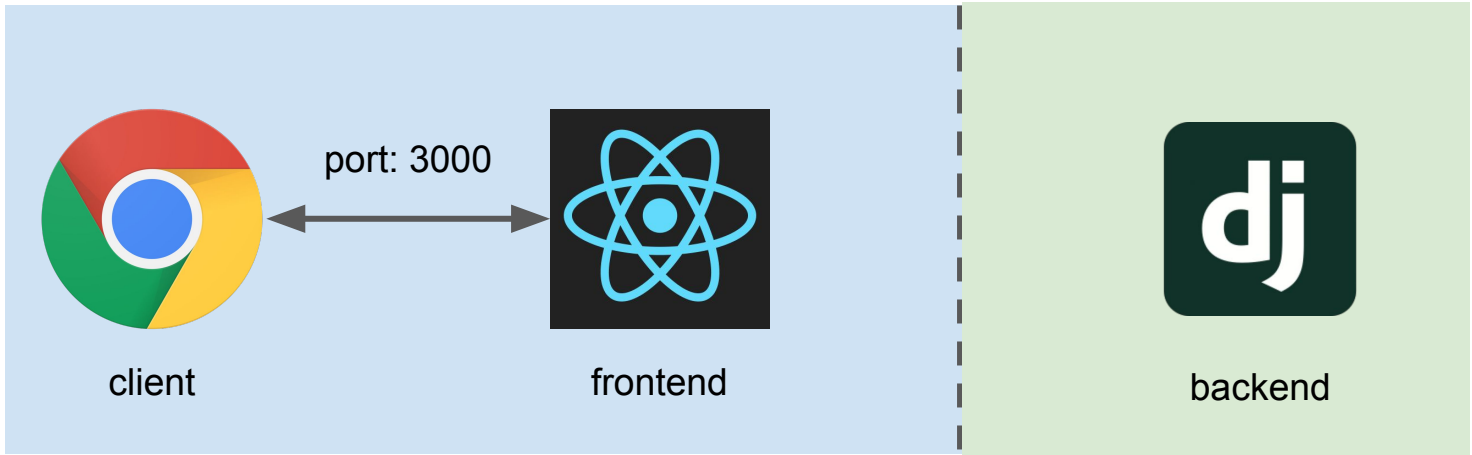
Deployment Layout



We will cover the deployment settings in the future lecture.

How to connect React and Django?

- Let’s try what happens when we access the frontend server without any connection.



How to connect React and Django?

- Let’s try what happens when we access the frontend server without any connection.

Terminal 1

```
cd frontend
yarn
yarn start
```

Terminal 2

```
cd backend
python manage.py makemigrations
python manage.py migrate
python manage.py runserver 0.0.0.0:8000
```

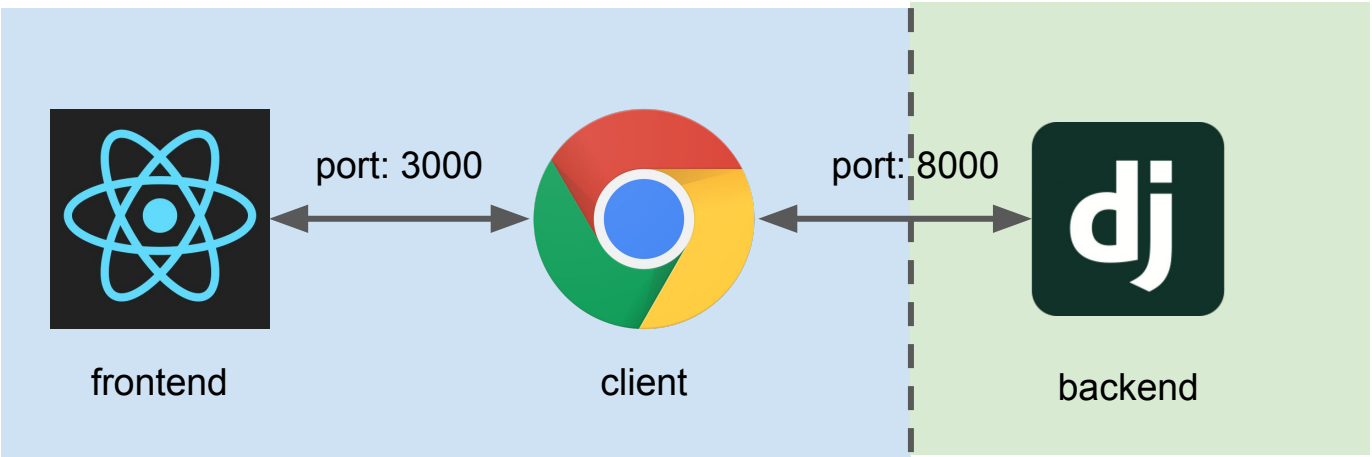
Use Proxy

- When you try to send a request (`/hero/info/`) with axios, it raise a 404 error.

- Add the following line at the end of `frontend/package.json`.
  - **`"proxy": "http://localhost:8000"`**
  - This will enable proxy server to bridge frontend and backend.
  - Run `yarn start` again when you add proxy settings in package.json.

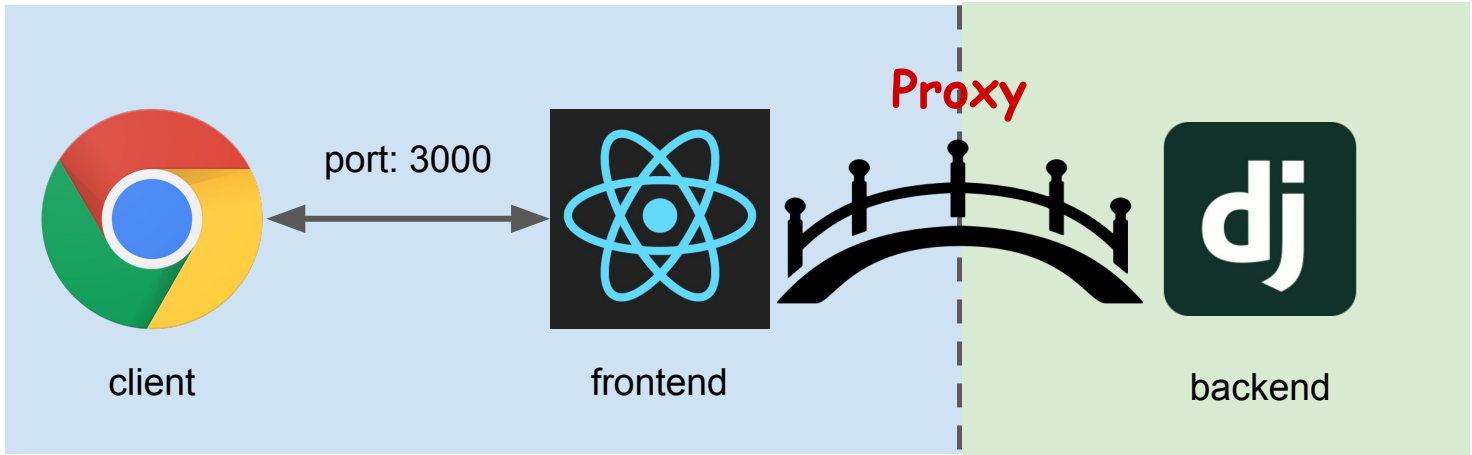
Use Proxy

- React frontend server cannot directly communicate with Django backend server.
  - `axios.get('hero/info/')`
  - `fetch('hero/info/')`



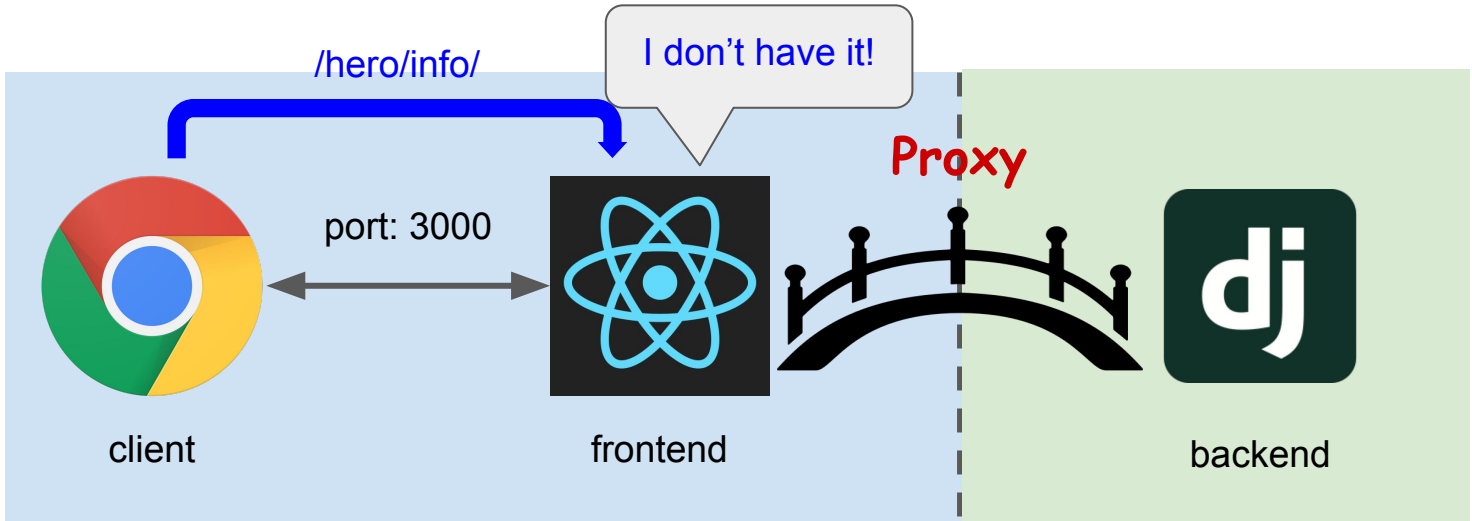
Use Proxy

- The web browser only communicates with a frontend server (`localhost:3000`).



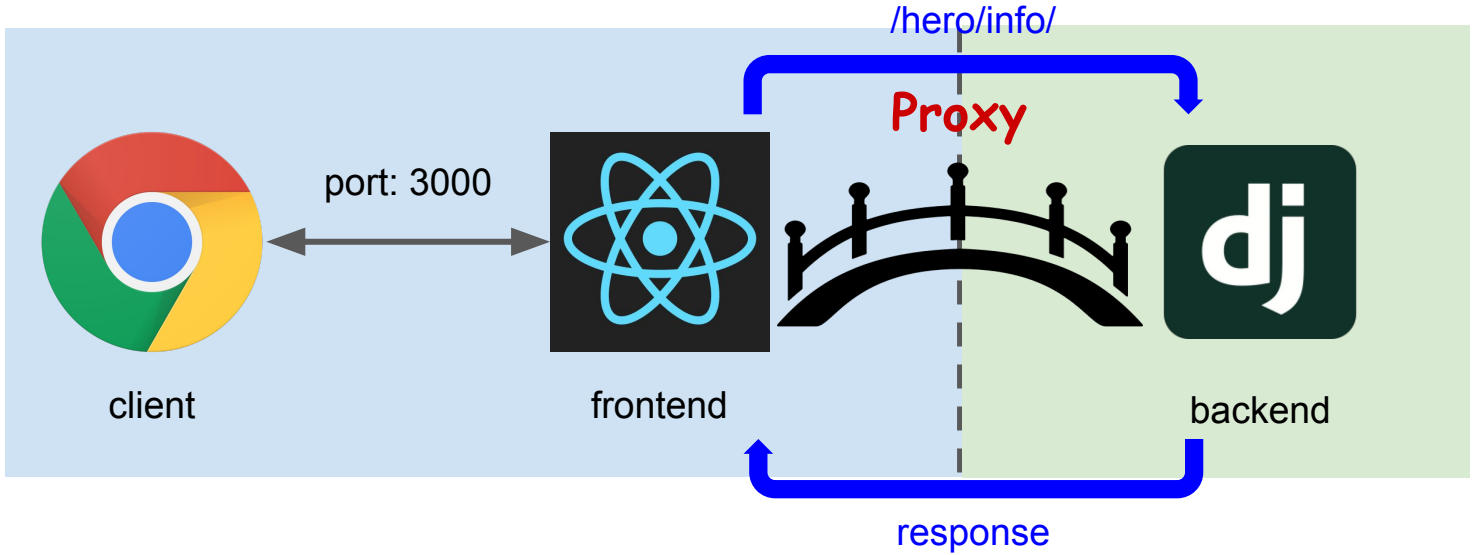
Use Proxy

- User sends a request (`localhost:3000/hero/info/`) to React server.
- React server cannot find the identical URL from itself.



Use Proxy

- Instead of raising an error instantly, React forwards the request to backend server. (Django)
- A response is returned from backend server to frontend.



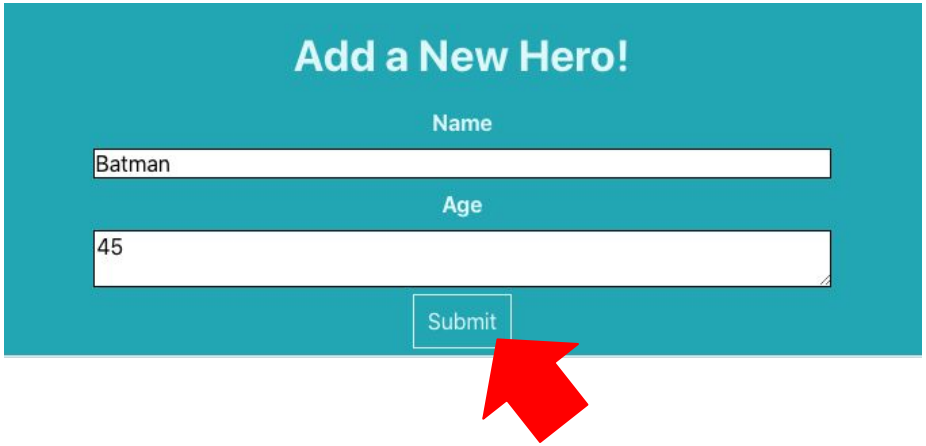
Frontend-Backend Integration

What we will learn

1. How to connect frontend and backend through proxy server
2. How to handle CSRF token in frontend side.
    a. React / Axios
    b. React / Fetch
    c. React Form

How to handle CSRF token?

- Now we have a proxy to bridge FE and BE.
- But we still have a problem :(
- When you try to send a POST request, 403 error occurs due to the absence of CSRF token.




```
Forbidden (CSRF cookie not set.): /hero/info/
[24/Oct/2020 16:24:13] "POST /hero/info/ HTTP/1.1" 403 2864
```

Review: CSRF token

- Whenever server responds to a request, it generates a new token.
  - Store it to session, and send it to client.
- Client send form(or request) with that CSRF token.
  - The token is embedded as a hidden property of the form.
- If server cannot find client’s CSRF token or provided the wrong token, just reject the request.

Handling CSRF Token: Django setting

- React renders component dynamically.
  - Form is dynamically added to the page.
- Django does not set the CSRF token cookie if the view is not rendering a template.
- We need to enforce a view to send the CSRF cookie.
  - Django provides a view decorator for this purpose:
  - `ensure_csrf_cookie(view)`

```
from django.views.decorators.csrf import ensure_csrf_cookie
```

```
@ensure_csrf_cookie
def hero_list(request):
    if request.method == 'GET':
        hero_all_list = [hero for hero in Hero.objects.all().values()]
        return JsonResponse(hero_all_list, safe=False)

    elif request.method == 'POST':
```

Handling CSRF Token: (1) React / Axios

- Attach CSRF token in the header of Axios call.
- You can automatically attach the CSRF token with the following code.

frontend/src/store/slices/hero.ts

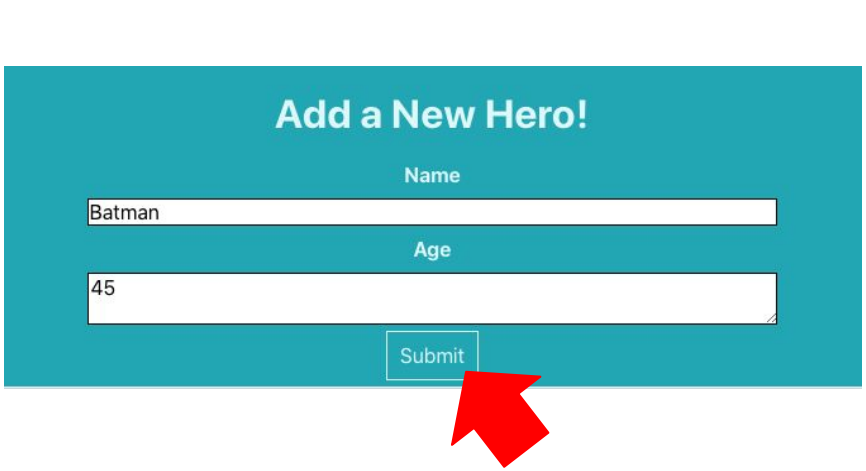
```
import axios from 'axios';

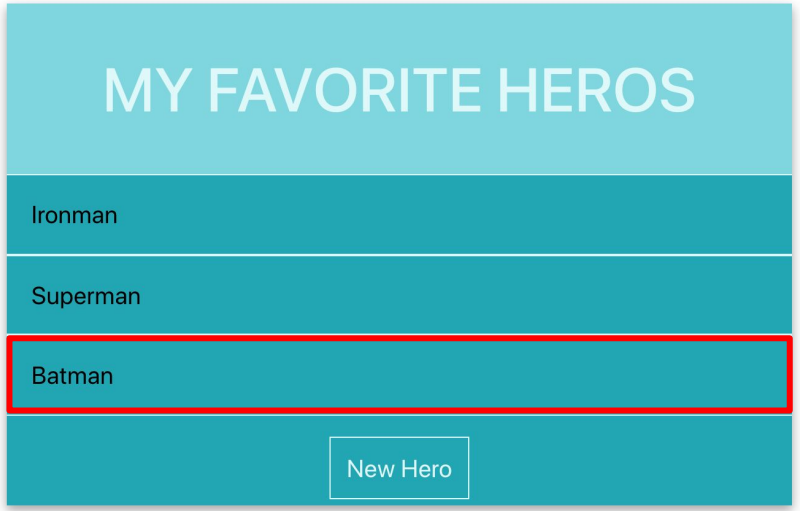
axios.defaults.xsrfCookieName = 'csrftoken';
axios.defaults.xsrfHeaderName = 'X-CSRFToken';
```

Handling CSRF Token: (1) React / Axios

- Now it works!

```
[26/Oct/2020 07:48:32] "POST /hero/info/ HTTP/1.1" 201 40
```





Handling CSRF Token: (2) React / Fetch

- Django docs provides a JS function to get the CSRF token form the cookie.
- Retrieve the CSRF token by calling `getCookie('csrftoken')`.

```
function getCookie(name) {
    let cookieValue = null;
    if (document.cookie && document.cookie !== '') {
        const cookies = document.cookie.split(';');
        for (let i = 0; i < cookies.length; i++) {
            const cookie = cookies[i].trim();
            if (cookie.substring(0, name.length + 1)
=== (name + '=')) {
                cookieValue =
decodeURIComponent(cookie.substring(name.length + 1));
                break;
            }
        }
    }
    return cookieValue;
}

const csrftoken = getCookie('csrftoken');
```

Handling CSRF Token: (2) React / Fetch

- Django docs provides a JS function to get the CSRF token form the cookie.
- Retrieve the CSRF token by calling `getCookie('csrftoken')`.
- Use the retrieved CSRF token when sending a request with fetch API.
  - Embed it to `X-CSRFToken` header.

Don't send csrftoken to another domain.

```
fetch(url, {
    credentials: 'include',
    method: 'POST',
    mode: 'same-origin',
    headers: {
        'Accept': 'application/json',
        'Content-Type': 'application/json',
        'X-CSRFToken': csrftoken
    },
    body: {
        // ...
    }
}).then(function(response) {
    // ...
});
```

retrieved csrftoken

Handling CSRF Token: (3) React Form

- Django template use `{ % csrf_token % }` tag to render a CSRF token inside a form. → React cannot use this tag.
- Need to create a frontend component to retrieve the CSRF token.

```
// csrftoken.js
import React from 'react';

const csrftoken = getCookie('csrftoken');

const CSRFToken = () => {
    return (
        <input type="hidden" name="csrfmiddlewaretoken" value={csrftoken} />
    );
};

export default CSRFToken;
```

Handling CSRF Token: (3) React Form

- Django template use `{ % csrf_token % }` tag to render a CSRF token inside a form. → React cannot use this tag.
- Need to create a frontend component to retrieve the CSRF token.

```
import React, { useState } from 'react';
import CSRFToken from './csrftoken';

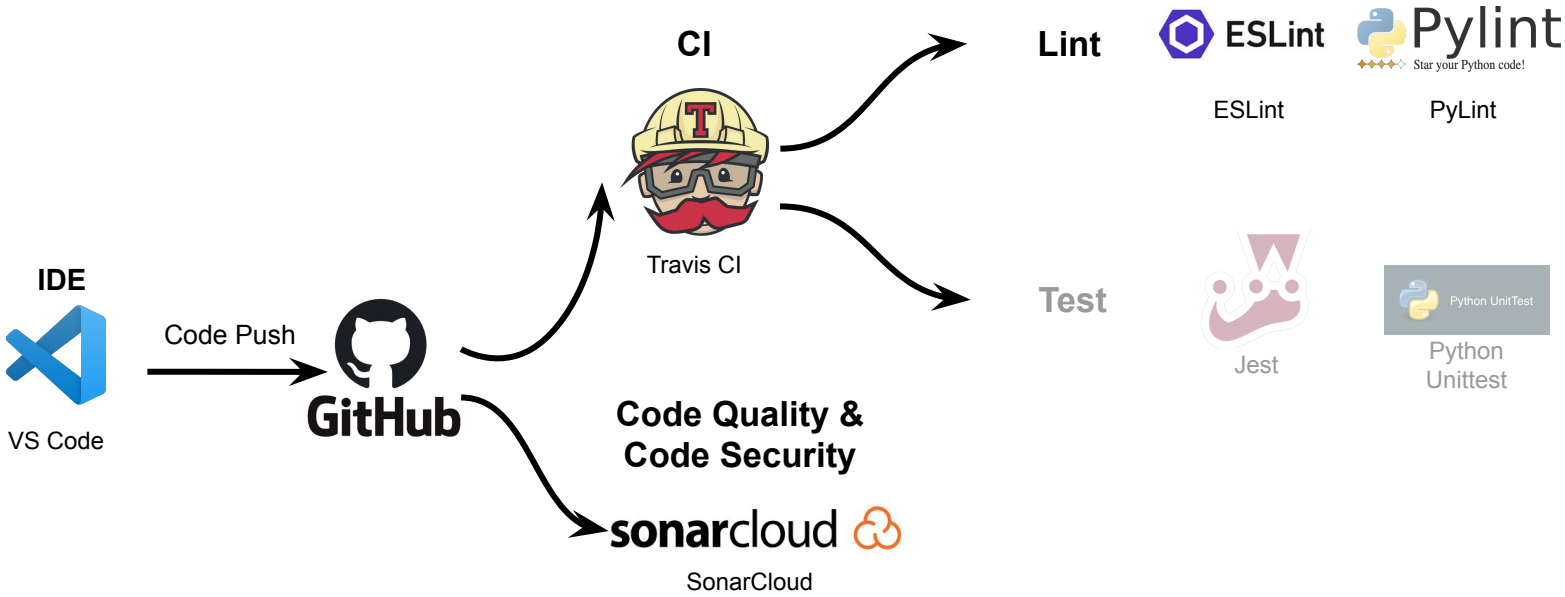
function ExampleForm() {
    return (
        <form action="/endpoint" method="post">
            <CSRFToken />
            <button type="submit">Send</button>
        </form>
    );
}
export default ExampleForm;
```

Today’s Task (Individual)

1. Set proxy server in your package.json
2. Apply Axios CSRF settings
3. Create a pull request
   - Due: 10/27 (Thu) 9pm

Today's Contents

2. Continuous Integration(CI) and code analysis tool setup
   a. Automate static code analysis using Lint tools through Travis CI
   b. Automate static code quality analysis with SonarCloud



Lint (ESLint + Pylint)

Lint

What we will learn

1. What is Linter? Purpose?
2. How to apply ESLint
3. How to apply PyLint

What is your coding style?

- Someone in your team may prefer snake_case, while you prefer camelCase.
- For the ease of maintenance and cooperation, we need a unified coding convention.
- Linter checks the coding style along with error-prone codes.

What is Linter?

- A static analysis tool that analyzes source code to flag programming errors, bugs, stylistic errors, and suspicious constructs.
- For our projects, we will use
  - ESLint for Javascript
  - PyLint for Python

Dynamic analysis is not everything

- **Dynamic analysis**: testing a program by executing data in real-time.
  - Unit test, integration test, acceptance test, etc.
- But as you may have realized…
  - You can achieve high coverage with incomprehensive test cases.
  - Even a good dynamic test does not guarantee soundness of your code.
- **Static analysis**: analysis is performed without executing program.
  - It shows potential problems of your code.
  - It also doesn’t always guarantee soundness of your code, but still very useful.

Dynamic analysis vs. Static analysis

Dynamic analysis

- **Can test developer's intent.**
  - Hey, `multiply(3,4)` should have returned 12, not 7!

```
int multiply(int a, int b) {
    return a + b;
}
```

- Only reports **actual** failure in execution.
- No false positive / Possibly many false negative
  - If unit test fails, the code is guaranteed to be wrong.
  - No failure in unit test does not guarantee soundness of your code.
- Takes quite a lot of effort.
  - Changing feature often requires changing test code.

Static analysis

- Cannot test developer's intent
  - No possible crash (e.g. segfault). It's fine!

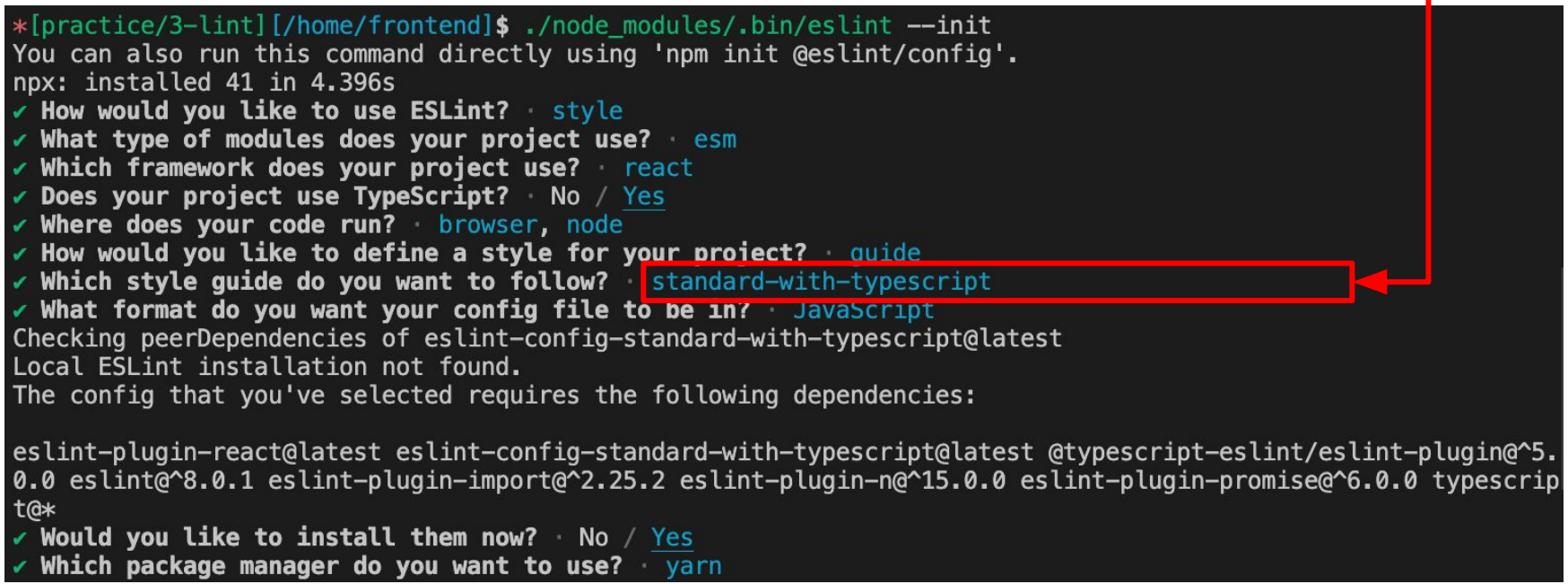
```
int multiply(int a, int b) {
    return a + b;
}
```

- Only reports **possible** failure in execution.
- Trade-off between false positive and false negative
  - You can configure the balance.
  - For example, you NEVER want a false negative in an autopilot program for aeroplanes.
- No extra effort
  - Changing feature does not require extra effort.

ESLint

- Install ESLint: `$ yarn add eslint --save-dev`
- Initialize ESLint: `$ ./node_modules/.bin/eslint --init`

Choose whatever you like



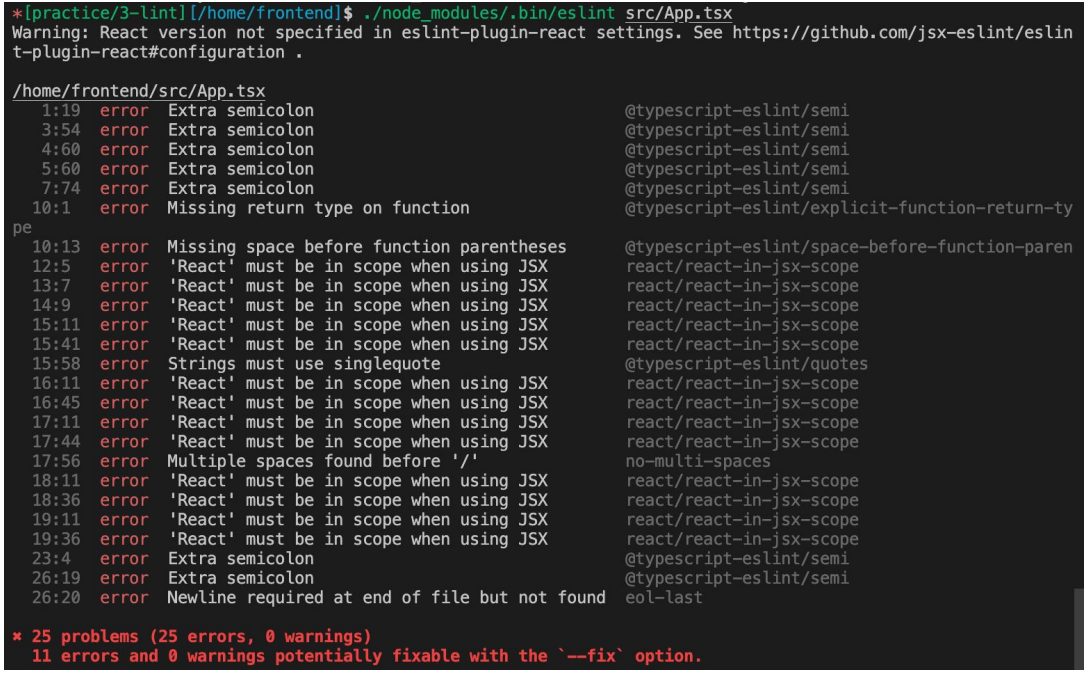
ESLint

- Open .eslint.js and add the highlighted configurations:

```
// project_root/frontend/.eslintrc.js
//...
  env: {
    //…
    jest: true  // This is for jest
  },
  extends: [
    'plugin:react/recommended', // This is for React
    'standard-with-typescript'
  ],
  parserOptions: {
    //…
    project: ['./tsconfig.json' ] // This is for Typescript
  }
//...
```

ESLint

- Run ESLint: `$ ./node_modules/.bin/eslint yourfile.tsx (.js)`
- You can see ESLint complain about your syntax, coding style, etc. :)



PyLint

- Install PyLint: `$ sudo apt install pylint`
  - Or `$ pip install pylint`
- Move to your project directory: `$ cd your/python/project`
- Run PyLint: `$ pylint **/*.py`
- You can config your pylint setting in `.pylintrc`.

Continuous Integration (CI)

Continuous Integration (CI)

What we will learn

1. What is CI?
2. Setting up Travis CI for your project

One common situation you will encounter...

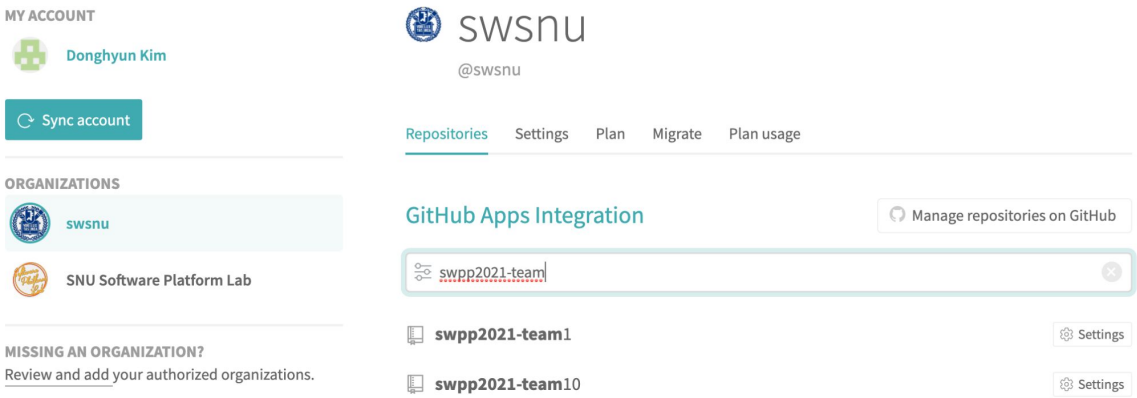
- You push your implementation to the remote repository after you double-check that it builds well, works well, etc.
- … but the build fails in the other teammates' machines!
- You: "Strange! It worked on my machine!"
- Your teammates:

Continuous Integration

- Keep merging your code frequently into the public repository
  - Can be done by Github Pull Request, etc...
- Continuous Integration (CI) service automatically builds and runs tests for you in a separate environment (not on your laptop!)
  - Build failure
  - Test case failure
  - Bad / suspicious code, etc.
- There are several free CI services (for public repo)
  - Travis CI
  - CircleCI
  - github actions

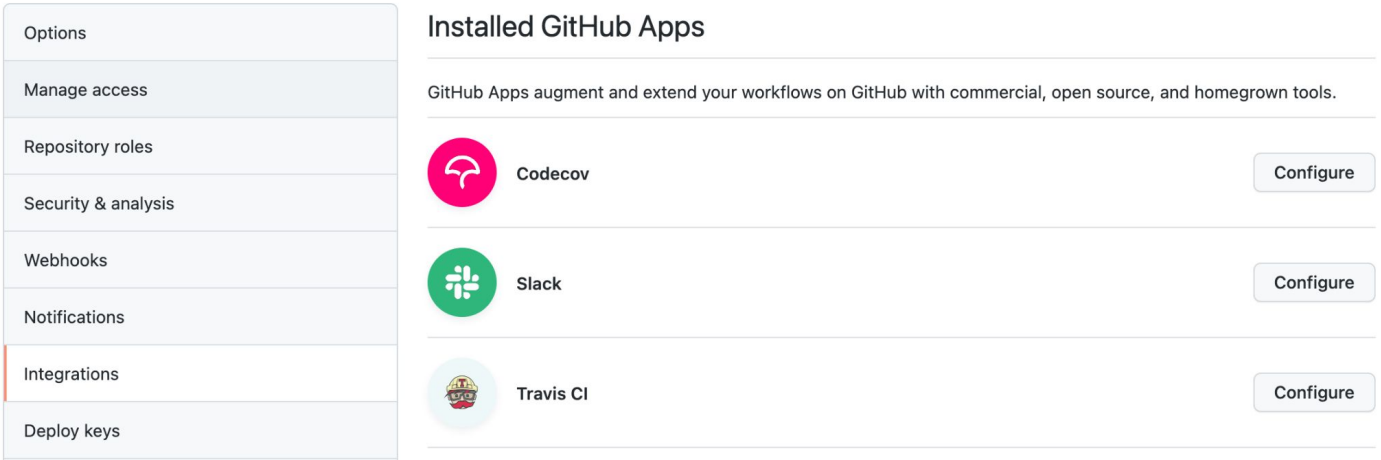
Setting up Travis CI

- https://travis-ci.com/
- Sign in with your Github account.
- You should be a member of the swsnu organization.
  - https://app.travis-ci.com/account/repositories
  - Let TAs know if the organization is not visible in Travis settings.



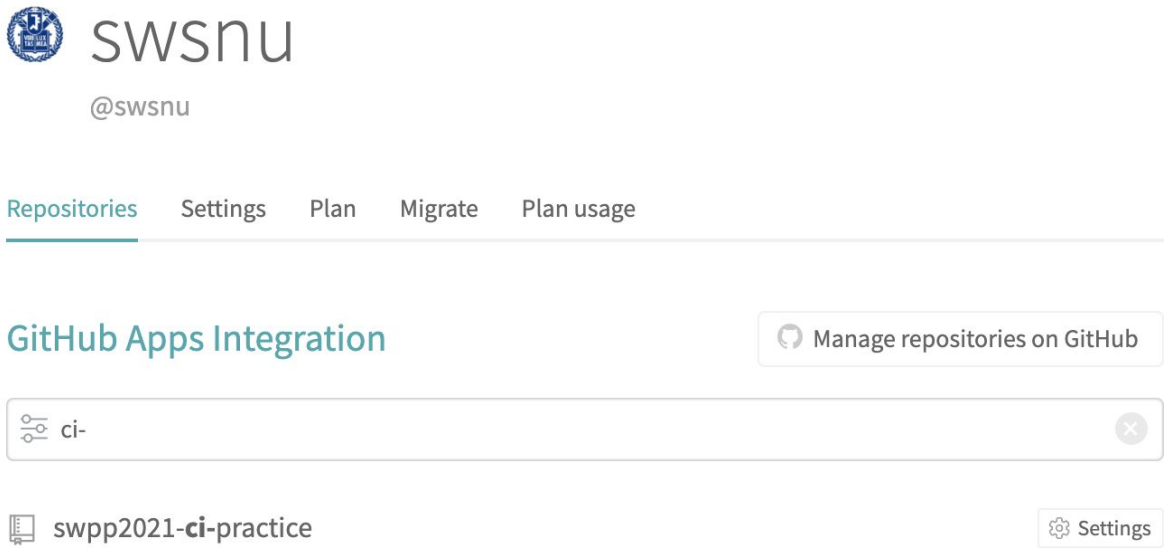
Lint + CI Demo

- Check if Travis CI is installed in the repository
  - GitHub repo settings -> Integrations



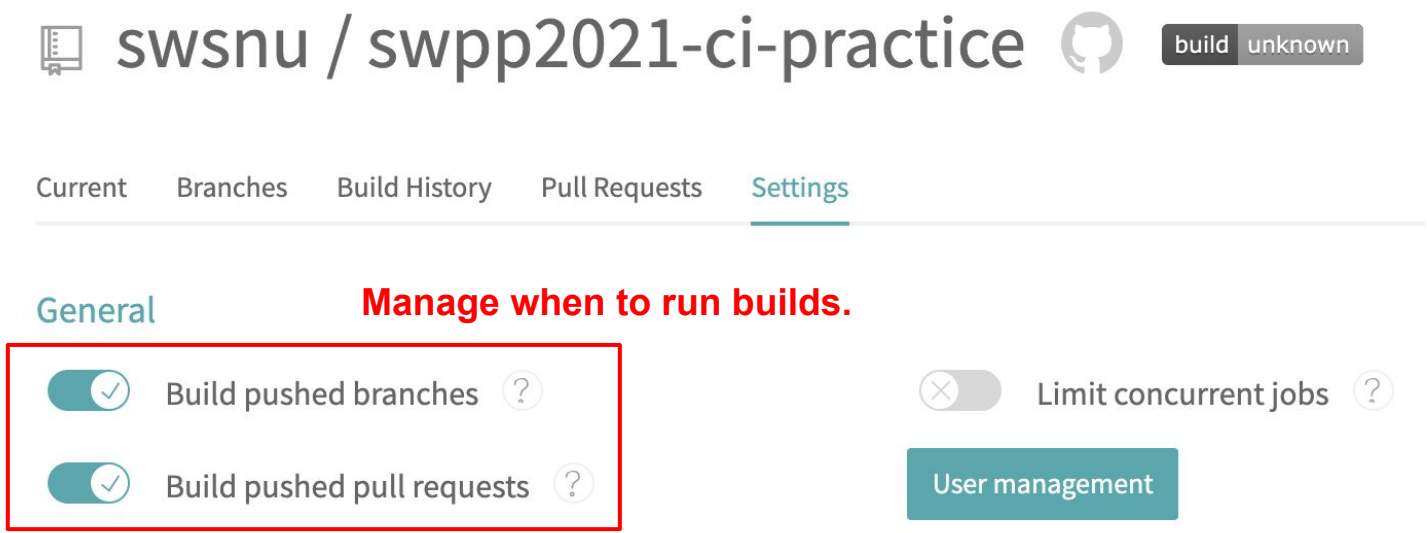
Lint + CI Demo

- Check if Travis CI can see your repository
  - Search for your team repo and check if you can access the Settings tab.



Lint + CI Demo

- Check if Travis CI can see your repository
  - Search for your team repo and check if you can access the Settings tab.



Lint + CI Demo

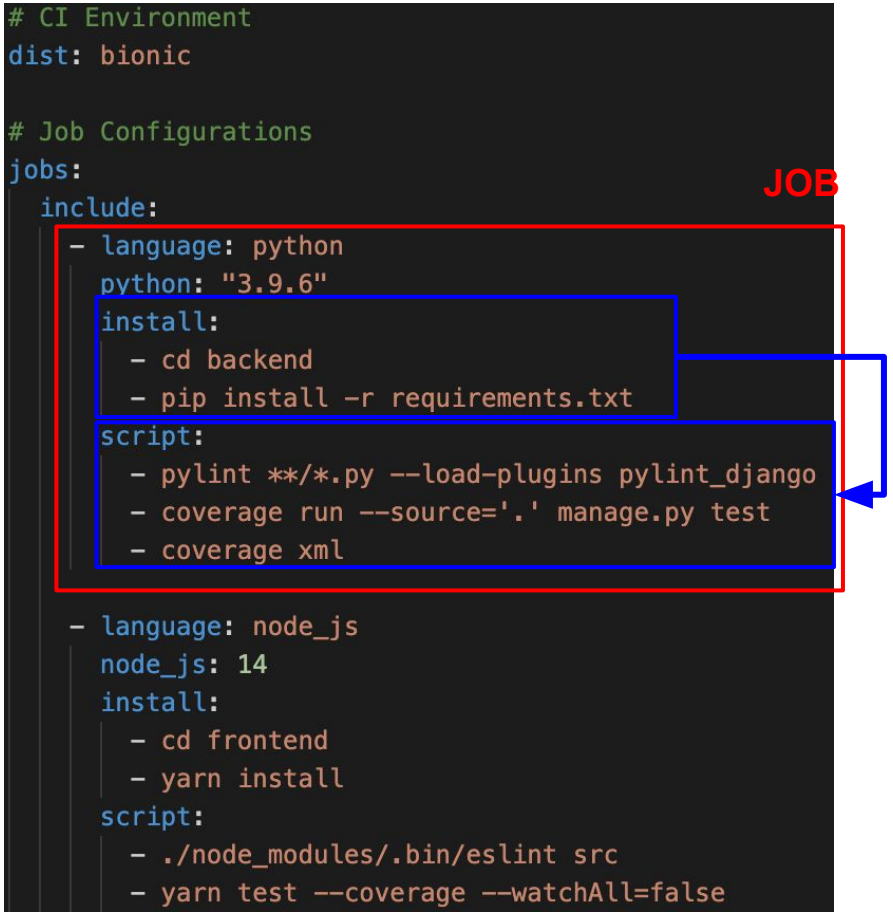
- Suppose we have a new React app and Django backend.
  - swpp2021-integration-practice:travis
  - We add backend dependencies in requirements.txt

```
$ cd backend
$ echo django==4.1.2 > requirements.txt
$ echo coverage==6.5 >> requirements.txt
$ echo pylint==2.15.5 >> requirements.txt
$ echo pylint-django==2.5.3 >> requirements.txt
```

```
$ tree -L 2
.
├── backend
│   ├── backend
│   ├── manage.py
│   └── requirements.txt
└── frontend
    ├── README.md
    ├── node_modules
    ├── package.json
    ├── public
    ├── src
    └── yarn.lock
```

Create travis config file

- File name should be `.travis.yml`
- Indentation is important!
- Build Lifecycle
  - install -> script
  - You can specify more phases such as before_install, after_script.
  - More on Build Lifecycle

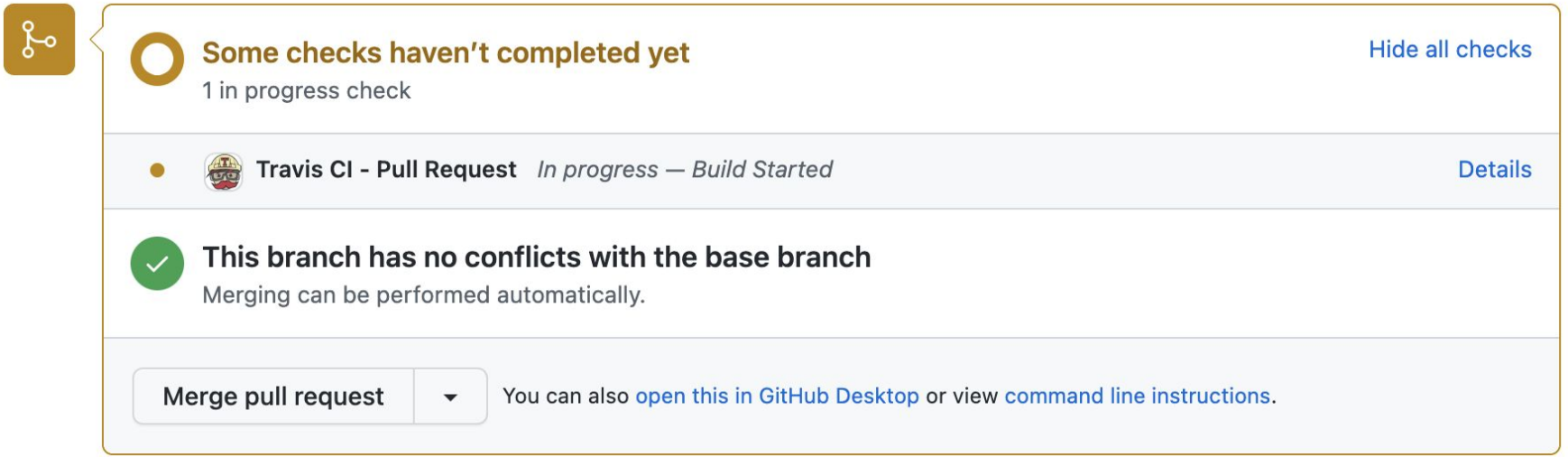


CI Demo

- YAML format
  - http://yaml.org/
- For more build configurations
  - https://docs.travis-ci.com/user/customizing-the-build/

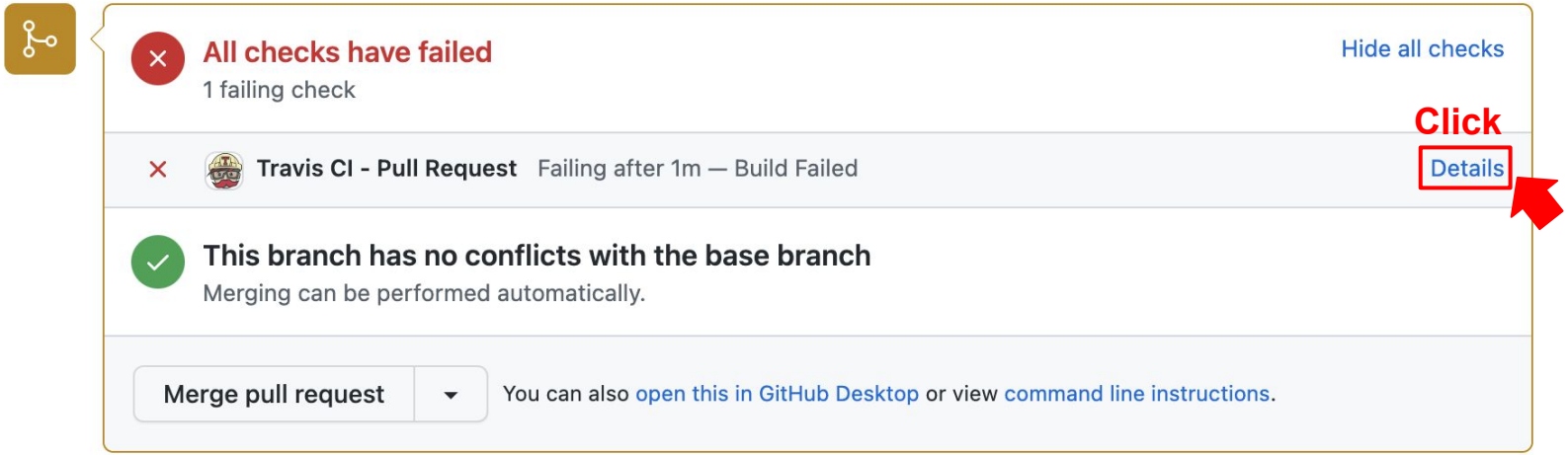
CI Demo

- Now, let’s see if our CI works.
- Push your changes and create a new pull request.
- And you’ll see your build is on progress



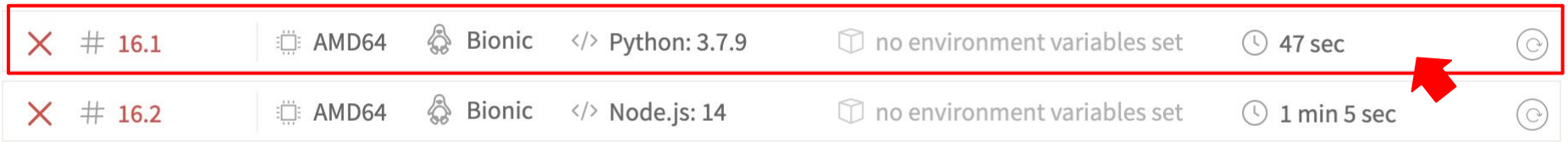
CI Demo

- Your build fails if one of your `script` (defined in .travis.yml) fails.
  - ESLint or PyLint error
  - Failed unit test, etc.



CI Demo

- Detail of your test build



CI Demo

- Detail of your test build

This build failed due to Pylint error

```
272 ************* Module hero.views
273 hero/views.py:51:9: C0303: Trailing whitespace (trailing-whitespace)
274 hero/views.py:1:0: C0114: Missing module docstring (missing-module-docstring)
275 hero/views.py:10:0: C0116: Missing function or method docstring (missing-function-docstring)
276 hero/views.py:14:0: C0103: Argument name "id" doesn't conform to snake_case naming style (invalid-name)
277 hero/views.py:14:0: C0116: Missing function or method docstring (missing-function-docstring)
278 hero/views.py:14:23: W0622: Redefining built-in 'id' (redefined-builtin)
279 hero/views.py:15:4: R1705: Unnecessary "elif" after "return" (no-else-return)
280 hero/views.py:24:8: C0103: Variable name "e" doesn't conform to snake_case naming style (invalid-name)
281 hero/views.py:24:8: W0612: Unused variable 'e' (unused-variable)
282 hero/views.py:36:0: C0116: Missing function or method docstring (missing-function-docstring)
283 hero/views.py:37:4: R1705: Unnecessary "elif" after "return" (no-else-return)
284 hero/views.py:38:0: R1721: Unnecessary use of a comprehension (unnecessary-comprehension)
285 hero/views.py:45:8: C0103: Variable name "e" doesn't conform to snake_case naming style (invalid-name)
286 hero/views.py:45:8: W0612: Unused variable 'e' (unused-variable)
287 hero/views.py:5:0: C0411: standard import "import json" should be placed before "from django.http import
    HttpResponse, JsonResponse, HttpResponseBadRequest" (wrong-import-order)
288 hero/views.py:6:0: C0411: standard import "from json.decoder import JSONDecodeError" should be placed before "from
    django.http import HttpResponse, JsonResponse, HttpResponseBadRequest" (wrong-import-order)
289
290 --------------------------------
291 Your code has been rated at 6.88/10
292
293 The command "pylint **/*.py --load-plugins pylint_django" exited with 28.
```

Setup CI on your team repository

- Please activate Travis CI on your team repository.
  - This is also included in our today's task!
- Check out this doc for your later sprint.
  - https://docs.travis-ci.com/

SonarCloud

SonarCloud

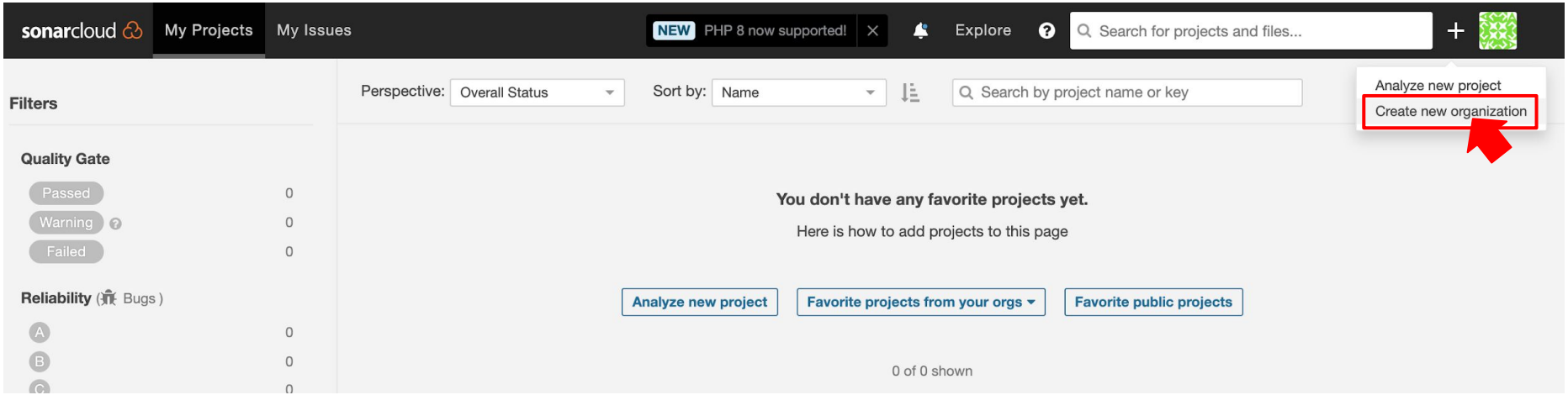
What we will learn

- What is SonarCloud?
- Setting up SonarCloud for your team project

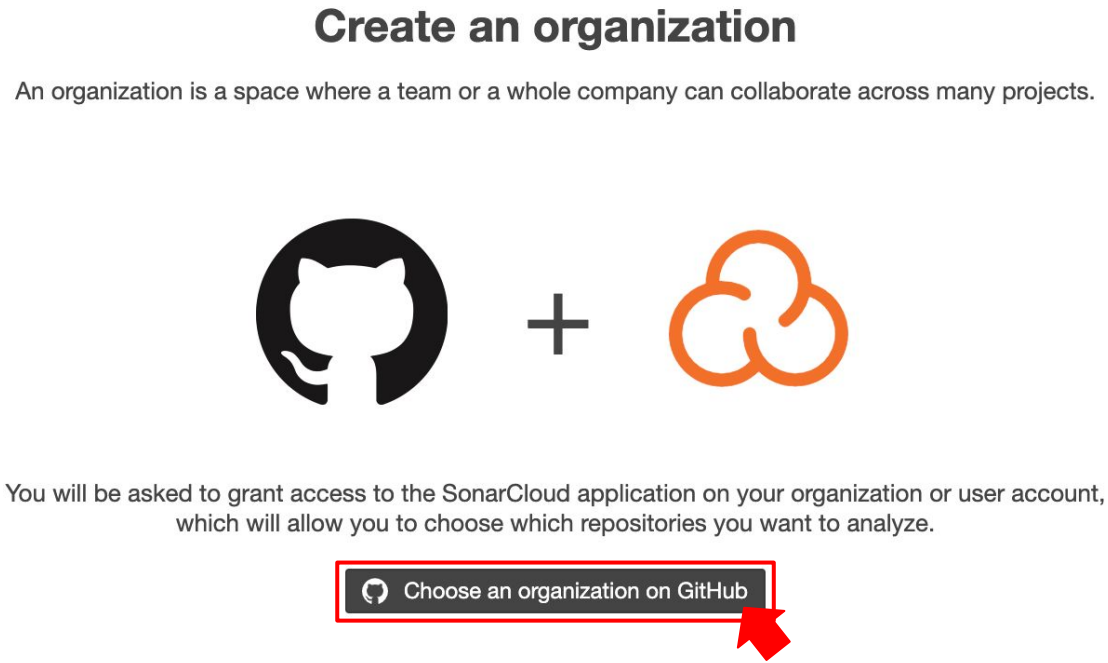
SonarCloud

- Continuous code quality static analysis tool.
  - bugs, vulnerabilities, code smell
- SonarCloud vs lint
  - SonarCloud highlights issues found on new code
  - SonarCloud supports 24 languages
  - SonarCloud integrates better with Github
- SonarCloud can be used an add-on of Travis CI
- Sign in with Github
  - https://sonarcloud.io/about

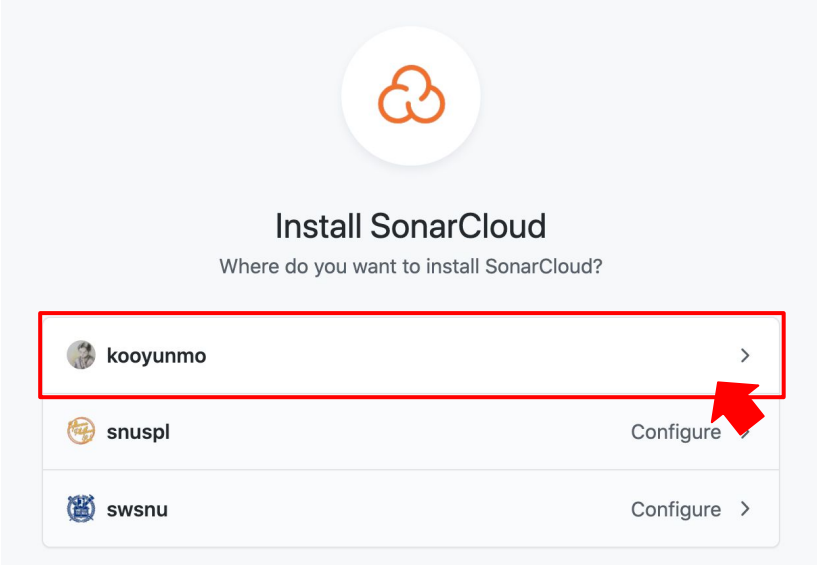
Create your SonarCloud Organization



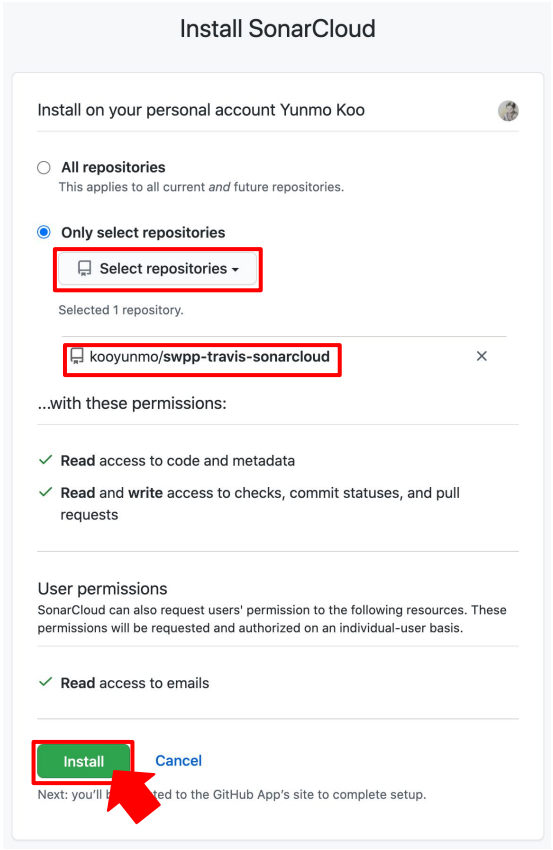
Create your SonarCloud Organization



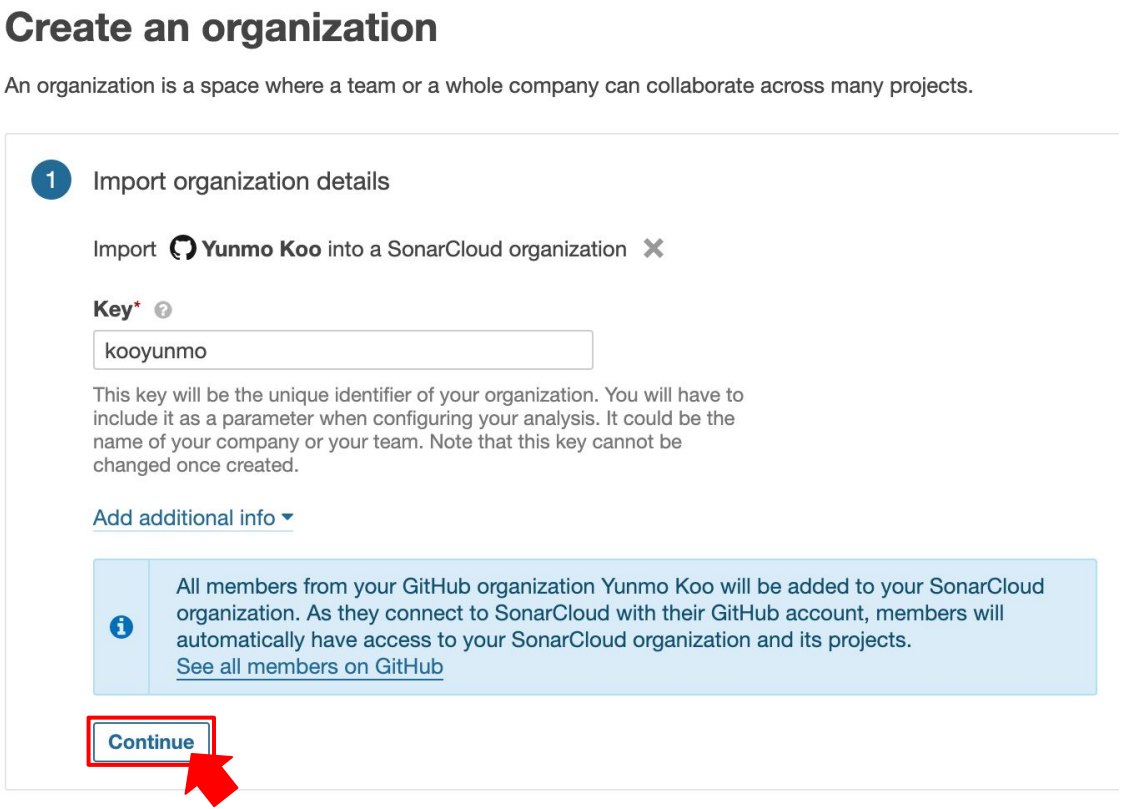
Create your SonarCloud Organization



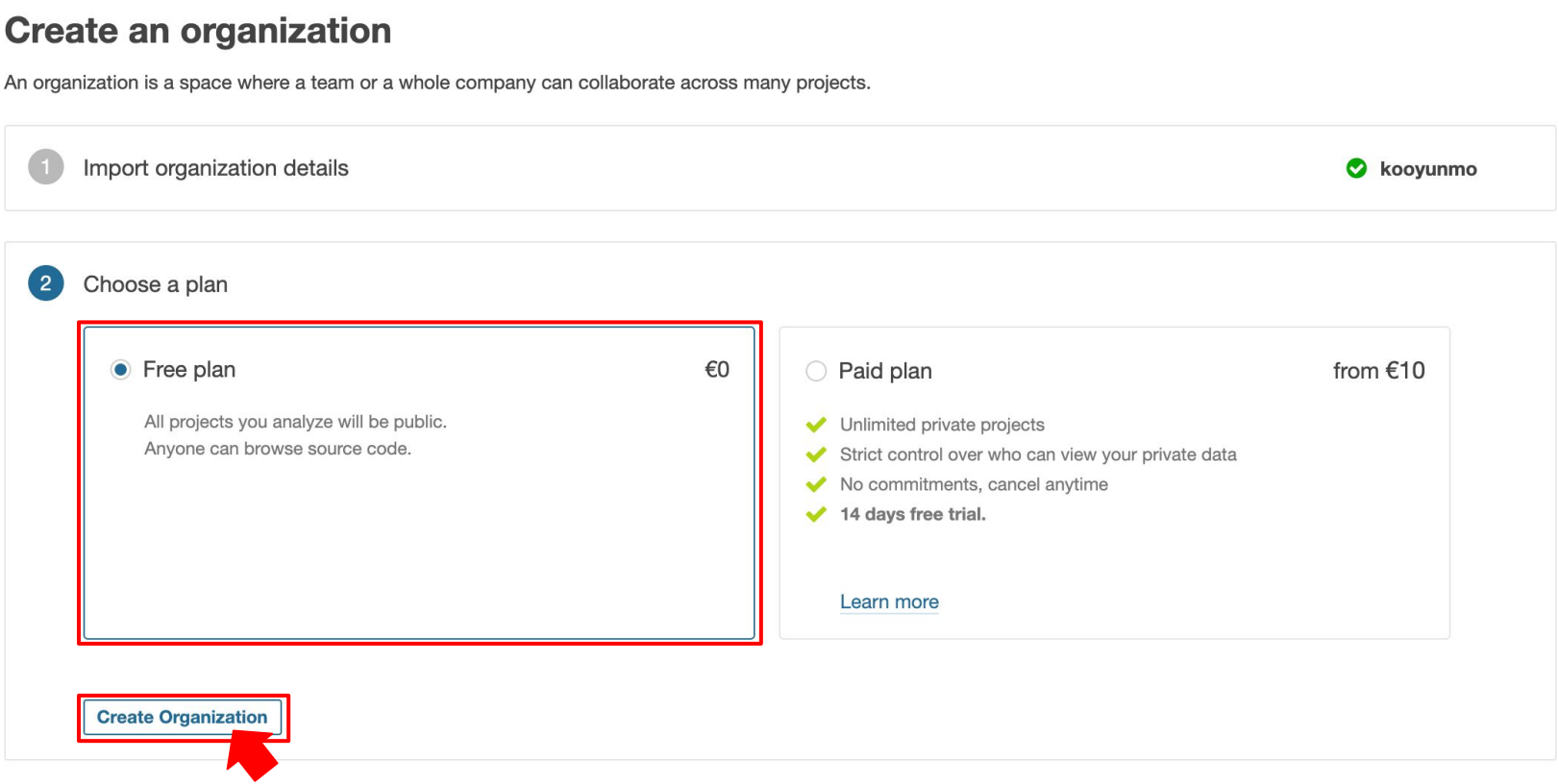
Create your SonarCloud Organization



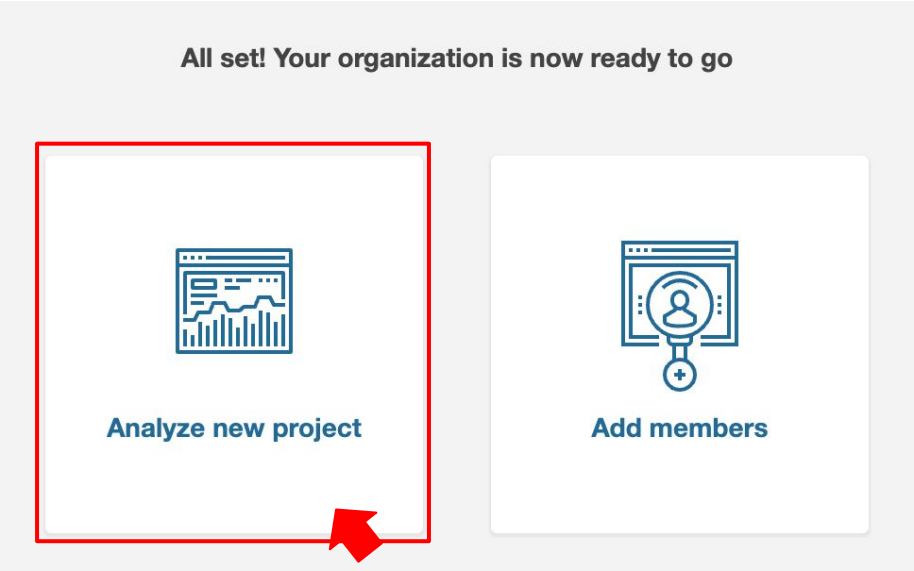
Create your SonarCloud Organization



Create your SonarCloud Organization

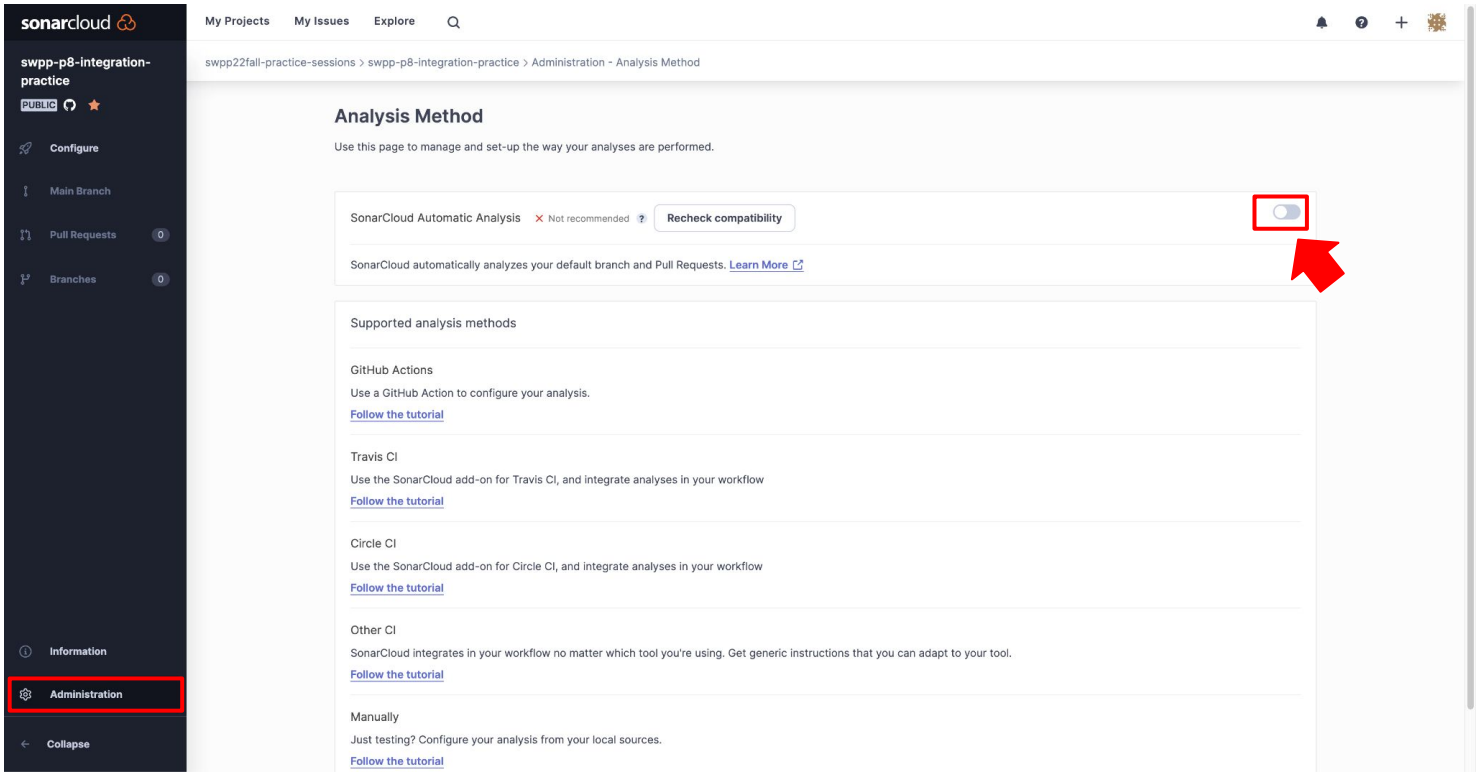


Create your SonarCloud Organization



Create your SonarCloud Organization

Create your SonarCloud Organization



your SonarCloud Organization

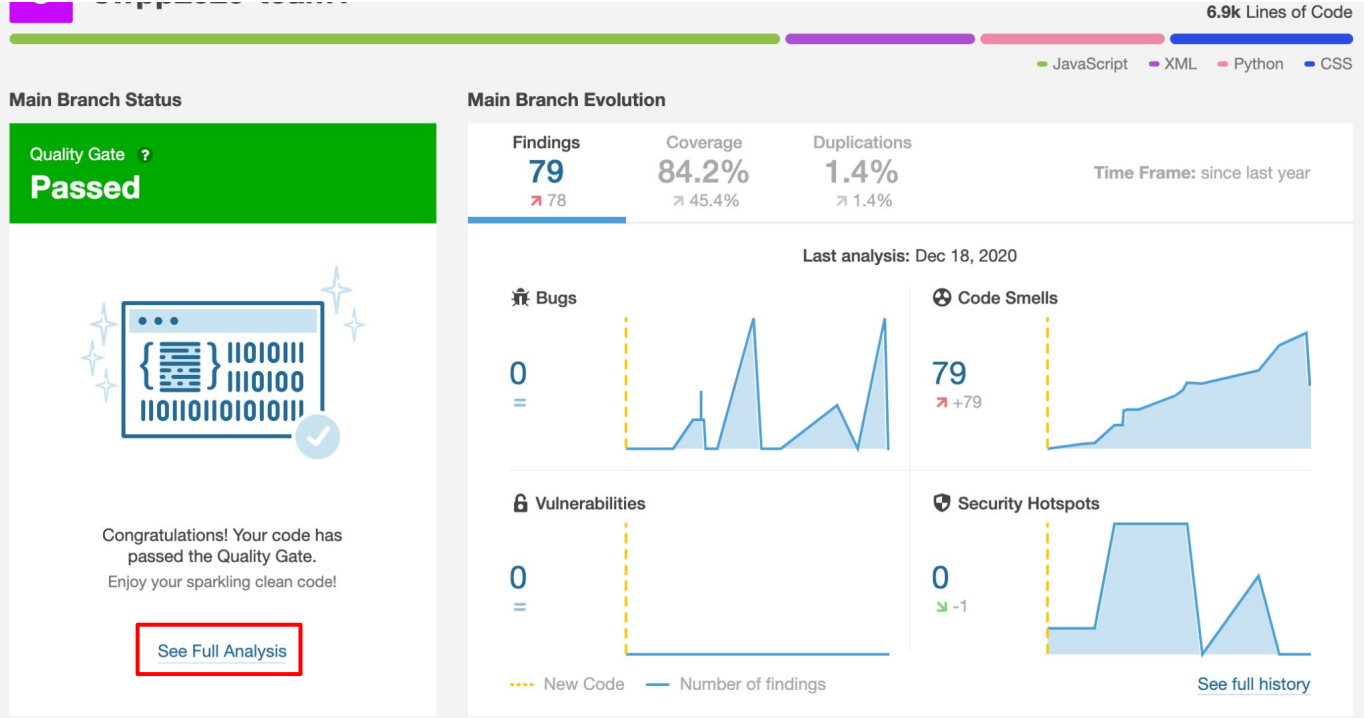
The SonarCloud projects for your team is in `swsnu`:

- https://sonarcloud.io/organizations/swsnu/projects

You can access your team sonarcloud page:

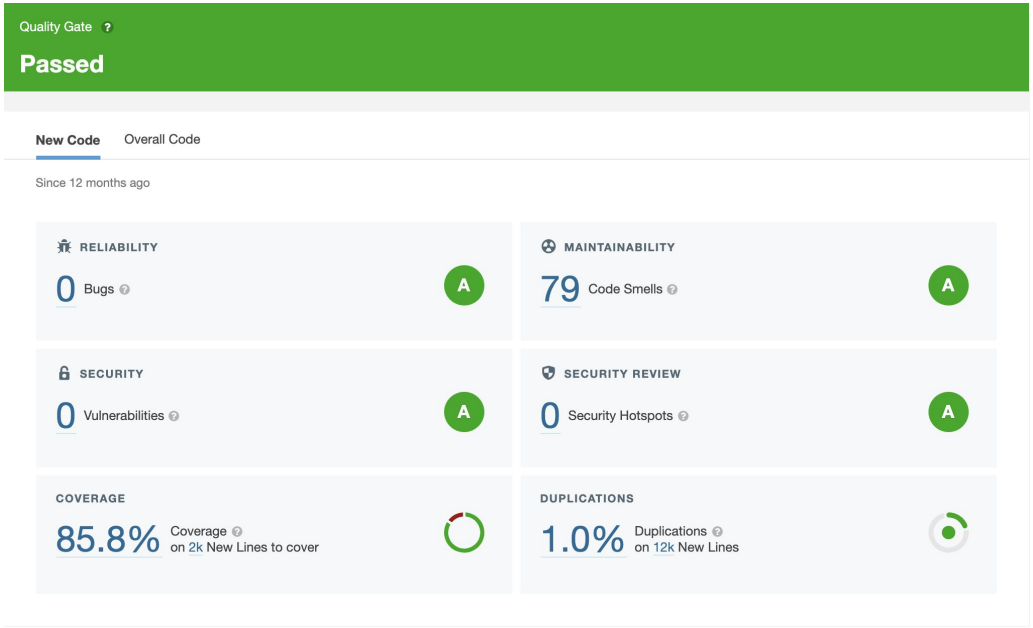
- https://sonarcloud.io/project/overview?id=swsnu_swppfall2022-teamX
- Replace `X` to your team number (1 ~ 20).

Code Analysis in SonarCloud



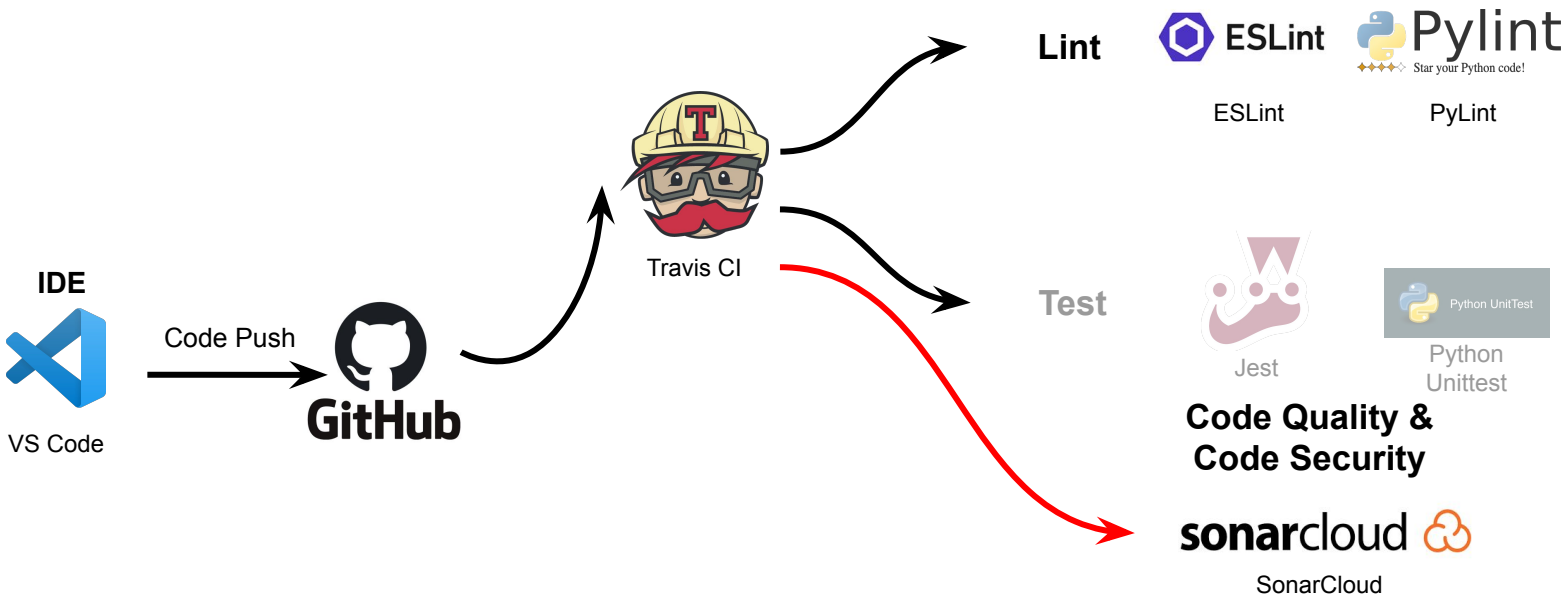
Code Analysis in SonarCloud

Should pass all the sections to pass the Quality Gate.

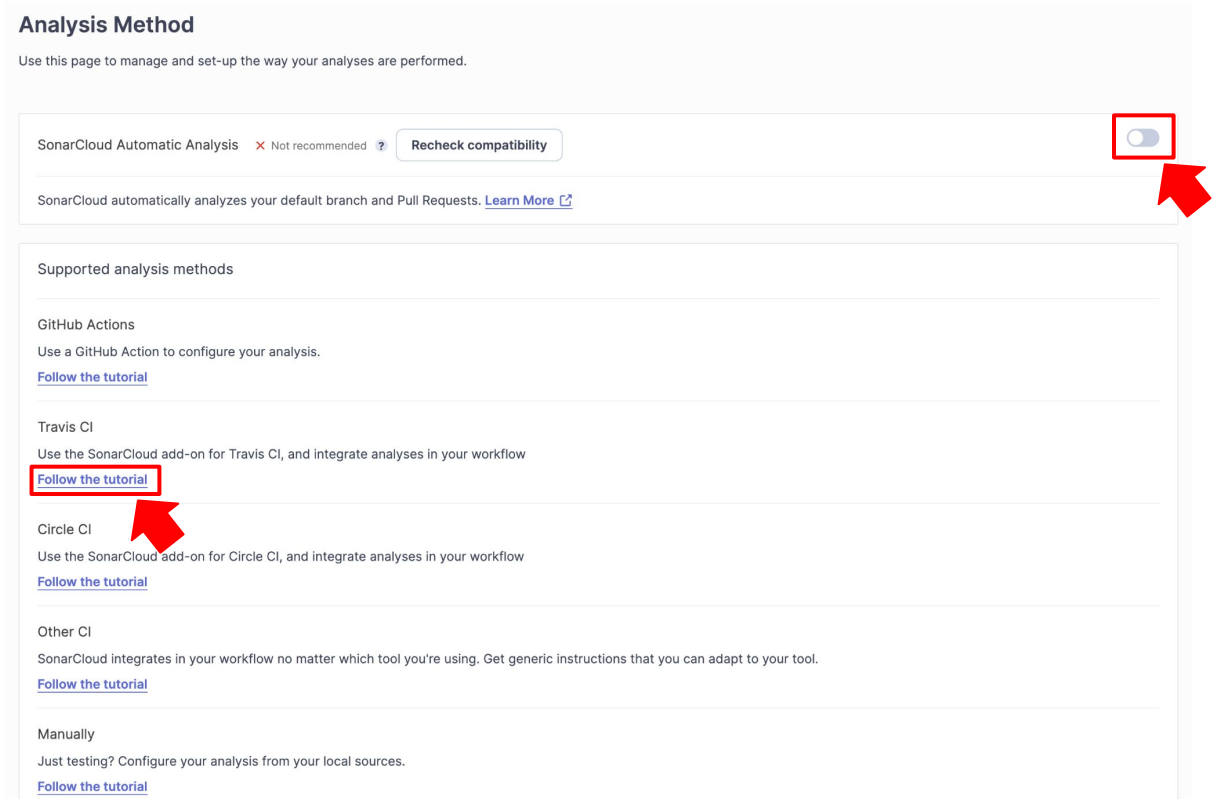


SonarCloud as an add-on of Travis CI

- Can trigger sonar-scanner with Travis CI
- Need connection between SonarCloud and Travis CI



Turn-off Automatic Analysis



Prepare Configuration Files

- Copy the output of `travis encrypt`
  - *token* is the token that you've just generated.
  - copy the result of encrypt command

```
$ sudo apt install ruby-full
$ gem install travis
```

for mac users:

```
$ brew install travis
```

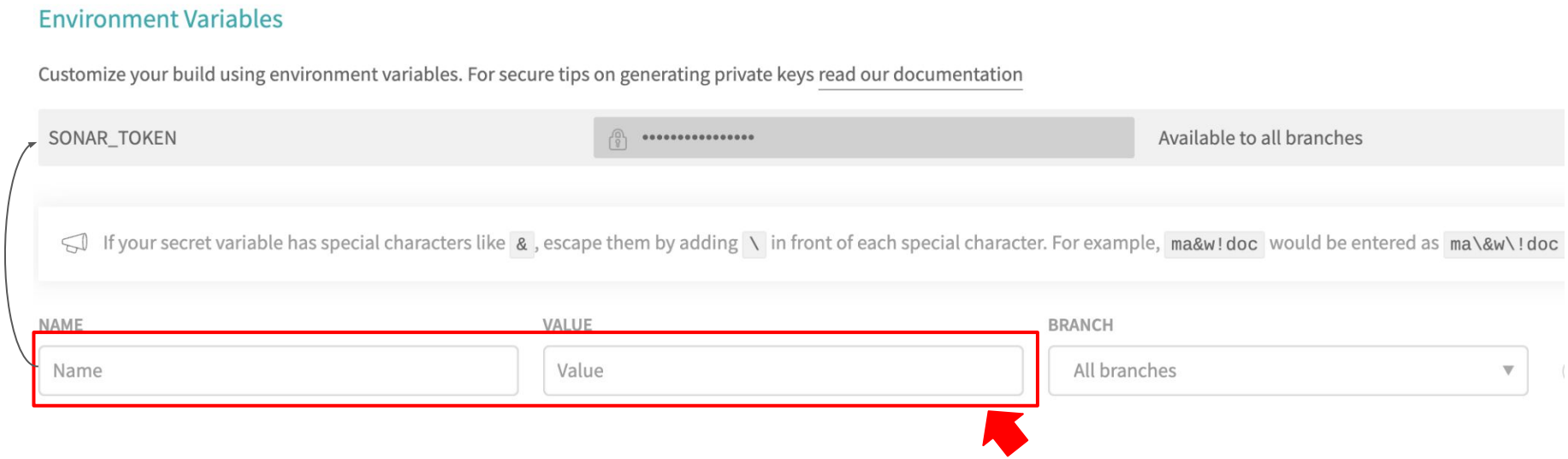
```
$ travis login --pro --github-token github_token
$ travis encrypt --pro token
```

```
Please add the following to your .travis.yml file:

  secure: "qerLSIOjhkQZrTtvUnkQMGSjQSS7v9TeghnbzQy2ipK/zbabHVBVDBSCAKEBROcP
CB3nlV5GWPnHHaB5P6PFrx9+WE5rLIpucqnAEVaKw+fww07e65YpG3gwH0D68DiX4jUve3fxI66
R5zpxSbJPb7+D6YAjYo8ZoAzwrqUsFHr5deeJ65Apg+0TsyGhJJ0bWCmreAx/1Ljrr0ZWJmQhEu
yidJfky7hviLT6qsiV6eHbOO72BYj1YFTsJoqqhsVkvyCQRJXLY7stNyXSs1yeX0/h1lYH9RZdk
1QJ79J2IwP6PE6A1HavbrgsrGzrCsFuaLN0lRqXdk0vvFiURunj61xcWFZCZoqajox1R7/iHRtZ
```

Copy this

Prepare Travis Repo



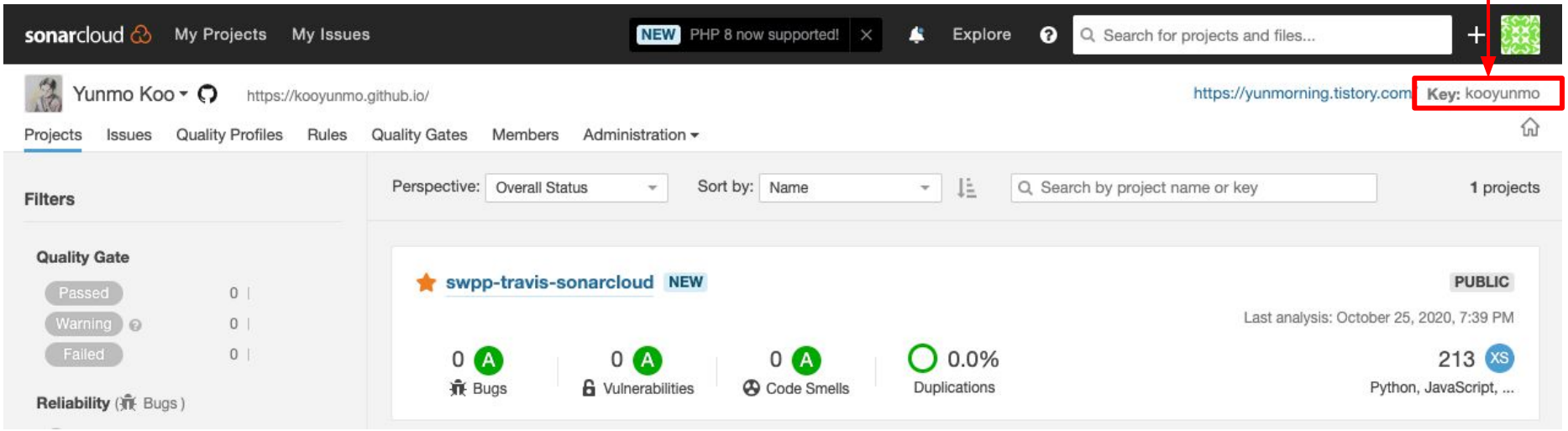
Prepare Configuration Files

- Open `.travis.yml` and add the highlighted lines.

```
# .travis.yml
#...
    node_js:
    - '10'
    addons:
      sonarcloud:
        organization: "your SonarCloud organization key (e.g. swsnu)"
        token:
          secure: "encrypted value that you just copied"
    before_install:
    #...
    script:
    - coverage xml
    - cd .. && sonar-scanner
```

Prepare Configuration Files

Your organization key



Prepare Configuration Files

- At the root of your project Create file with name: **sonar-project.properties**

```
sonar.projectKey=your-organization-key_your_project_name
sonar.organization=your-organization-key
sonar.projectName=your_project_name
sonar.projectVersion=1.0
sonar.sources=backend,frontend/src
sonar.tests=backend,frontend/src
sonar.exclusions=backend/**/migrations/*,frontend/src/index.tsx,frontend/src/serviceWorker.js
sonar.test.inclusions=backend/**/tests.py,frontend/src/**/*.test.js
sonar.python.coverage.reportPaths=backend/coverage.xml
sonar.javascript.lcov.reportPaths=frontend/coverage/lcov.info
```

Link to the example file

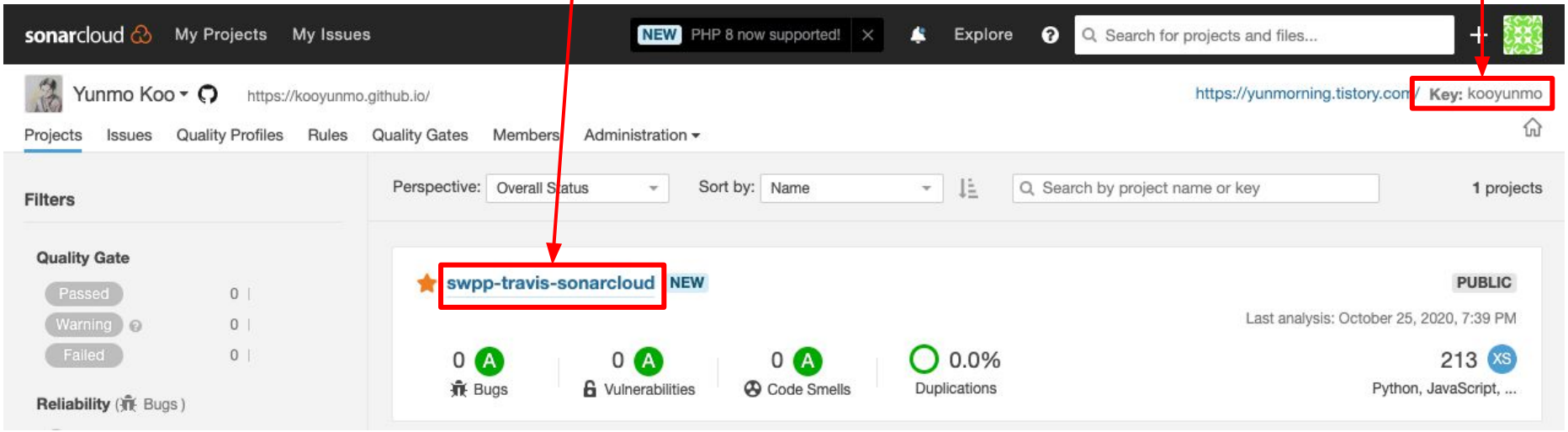
This has to be replaced with your SonarCloud organization key
- e.g. swsnu

These have to be replaced with your SonarCloud project name
- e.g. swppfall2022-teamX

Prepare Configuration Files

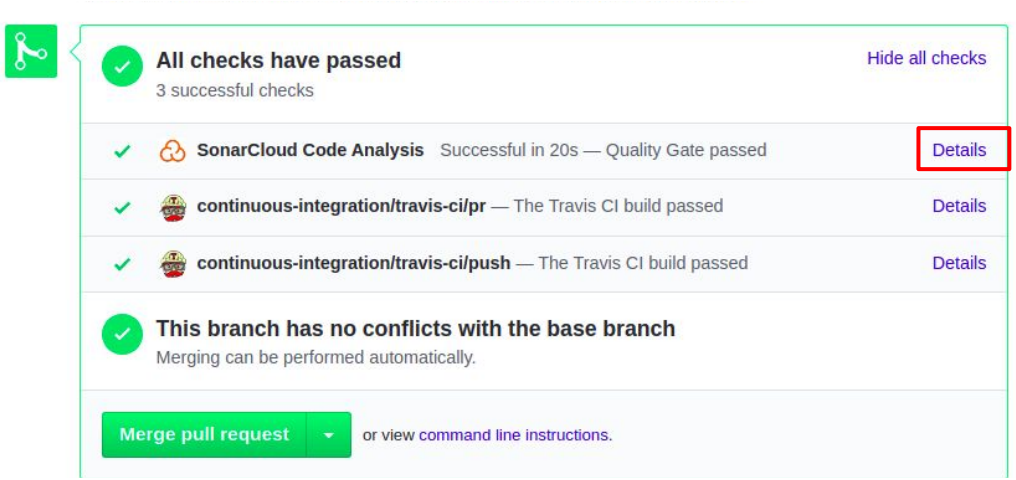
Your organization key

Your project name



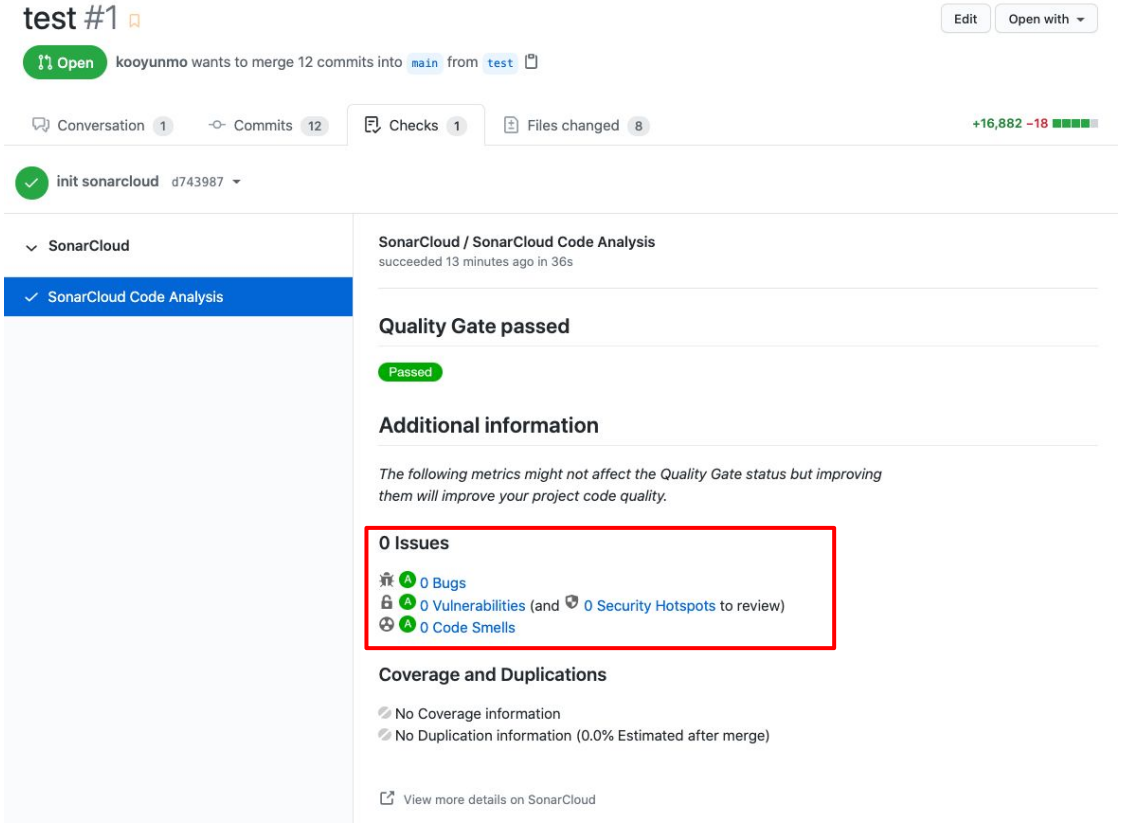
SonarCloud Demo

- Push all the changes and create a new pull request
- Travis CI will run sonar-scanner for you



Click

Hooray!


test #1
Edit
Open with
Open
kooyunmo wants to merge 12 commits into main from test
Conversation 1
Commits 12
Checks 1
Files changed 8
+16,882 −18
init sonarcloud d743987
SonarCloud
SonarCloud Code Analysis
SonarCloud / SonarCloud Code Analysis
succeeded 13 minutes ago in 36s
Quality Gate passed
Passed
Additional information
The following metrics might not affect the Quality Gate status but improving them will improve your project code quality.
0 Issues
0 Bugs
0 Vulnerabilities (and 0 Security Hotspots to review)
0 Code Smells
Coverage and Duplications
No Coverage information
No Duplication information (0.0% Estimated after merge)
View more details on SonarCloud

Add a badge to your team repo

- Copy and paste the following lines to your team repo `README.md`.
- Replace teamX to your team number.
  - `[![Build Status](https://travis-ci.com/swsnu/swppfall2022-teamX.svg?branch=main)](https://travis-ci.com/swsnu/swppfall2022-teamX)`
  - `[![Quality Gate Status](https://sonarcloud.io/api/project_badges/measure?project=swsnu_swppfall2022-teamX&metric=alert_status)](https://sonarcloud.io/dashboard?id=swsnu_swppfall2022-teamX)`
- You can attach status badges to your team repo README like this:

Today’s Task (Team)

- Adopt to your team project Travis CI with ESLint, PyLint, and SonarCloud.
- Add status badges to your team repo README.
- Make a pull request and set `jaewooMaeng` as your reviewer.
  - Files to be included: `.travis.yml`, `sonar-project.properties`, `README.md`
  - Due: 10/29 (Fri) 9pm
  - example: