

Test, intégration continue & Documentation

Travail en équipe



Sommaire

1. Test & intégration continue
2. Documenter son projet



1. Test & intégration continue

1. Test & intégration continue

Les tests (1/3)

- Lorsqu'on code, une partie importante de son travail consiste à tester ce qu'on a codé, afin de vérifier que notre projet fonctionne dans les différents cas que l'on a imaginé
- Les tests peuvent être de différentes natures. Pour commencer ils peuvent être soit :
 - Manuels, c'est-à-dire effectué en personne, en interagissant directement avec le projet afin de valider qu'il fonctionne. C'est une méthode qui est longue et forcément sujette à l'erreur humaine
 - Automatisés, c'est un script de test qui va réaliser automatiquement les interactions avec le projet, et renvoyer un message si le test est réussi ou non

1. Test & intégration continue

Les tests (2/3)

- Ensuite, il existe de nombreux types de tests, on a par exemple :
 - Les tests unitaires, sont des tests bas niveau, c'est-à-dire proche du code source du projet. Ils consistent à tester une fonction, une classe ou un module de façon individuelle
 - Les tests d'intégration, consistent à vérifier que les modules de votre projet fonctionnent entre eux. Par exemple si on a un module « utilisateur » et un module « gestion de droit » pour les permissions des utilisateurs, un test d'intégration serait de vérifier qu'ils fonctionnent bien ensemble

1. Test & intégration continue

Les tests (3/3)

- Ensuite, il existe de nombreux types de tests, on a par exemple :
 - Les tests fonctionnels, consistent à vérifier le bon fonctionnement du projet du point de vue de l'utilisateur. C'est-à-dire qu'on va non seulement vérifier que le système fonctionne, mais qu'il répond bien au cahier des charges. Par exemple un test fonctionnel consistera à vérifier qu'un formulaire envoie les bonnes données au serveur, et ce que ces données soient traitées et stockées correctement
 - Les tests de performances, consistent à évaluer la performance du système sous une charge de trafic spécifique. On testera par exemple si le projet a la capacité de gérer un grand nombre d'utilisateurs en simultané

1. Test & intégration continue

Pourquoi les tests sont importants

- On peut dire que les tests ont une place importante dans le processus de développement logiciel. Plus le projet sera gros plus il sera essentiel de s'en servir, et cela pour plusieurs raisons :
 - L'assurance qualité (QA), garantir la qualité et la fiabilité du code
 - La détection d'erreurs, les tests vont faciliter la détection de bugs en gérant des cas qui n'auraient pas été testés autrement
 - La maintenabilité, avoir une validation en permanence de son code permet de faire des mises à jour de façon plus sereine
 - Repérer l'origine du problème, grâce aux différents types de tests il sera plus simple de savoir d'où vient le problème lorsqu'il se présente

1. Test & intégration continue

Pourquoi les tests sont importants

- On peut dire que les tests ont une place importante dans le processus de développement logiciel. Plus le projet sera gros plus il sera essentiel de s'en servir, et cela pour plusieurs raisons :
 - L'assurance qualité (QA), garantir la qualité et la fiabilité du code
 - La détection d'erreurs, les tests vont faciliter la détection de bugs en gérant des cas qui n'auraient pas été testés autrement
 - La maintenabilité, avoir une validation en permanence de son code permet de faire des mises à jour de façon plus sereine
 - Repérer l'origine du problème, grâce aux différents types de tests il sera plus simple de savoir d'où vient le problème lorsqu'il se présente

1. Test & intégration continue

Pourquoi les tests sont importants

- En résumé les tests vont garantir :
 - La qualité
 - La stabilité
 - La conformité
 - Les performances

1. Test & intégration continue

Comment réaliser des tests

- Chaque langage de programmation a des outils pour réaliser des tests, par exemple pour les tests unitaires :
 - Python utilise unittest
 - Javascript utilise Jest
 - PHP utilise PHPUnit

```
import unittest

class TestSum(unittest.TestCase):
    def test_list_int(self):

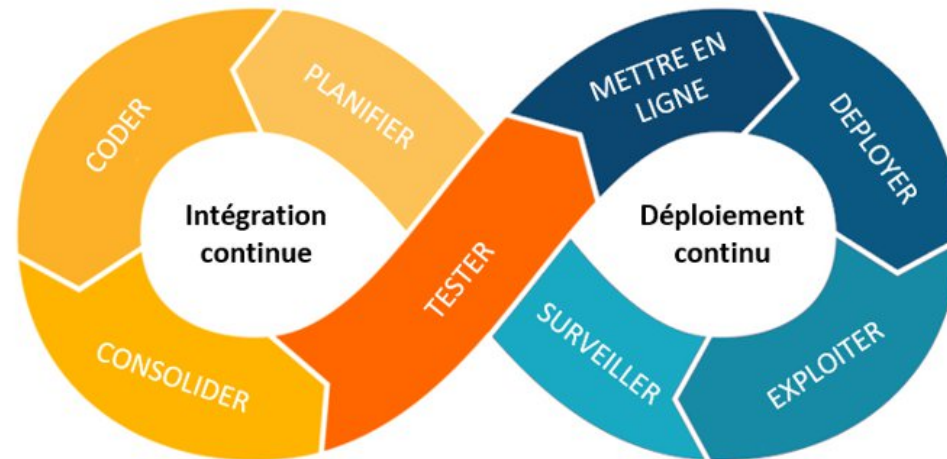
        data = [2, 4, 6]
        result = sum(data)
        self.assertEqual(result, 12)

if __name__ == '__main__':
    unittest.main()
```

1. Test & intégration continue

L'intégration continue

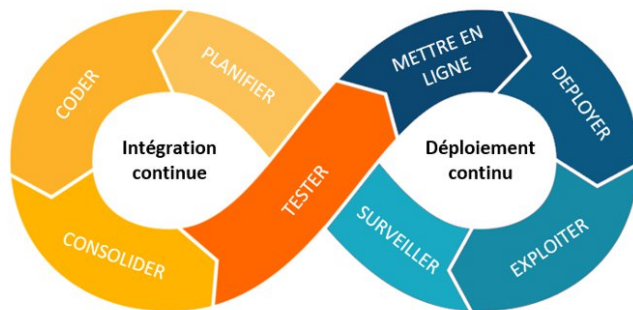
- Les tests automatisés s'intègrent dans une logique plus globale que l'on appelle l'intégration continue, aussi appelé CI (Continuous Integration).
Le nom complet est CI/CD (Continuous Integration / Continuous Delivery)



1. Test & intégration continue

L'intégration continue - CI : Continuous Integration

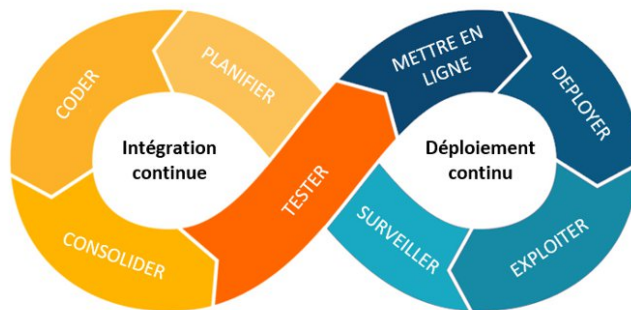
- « L'intégration continue est un ensemble de pratiques utilisées en génie logiciel consistant à vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression dans l'application développée. [...] Le principal but de cette pratique est de détecter les problèmes d'intégration au plus tôt lors du développement. De plus, elle permet d'automatiser l'exécution des suites de tests et de voir l'évolution du développement du logiciel. »



1. Test & intégration continue

L'intégration continue - CD : Continuous Delivery

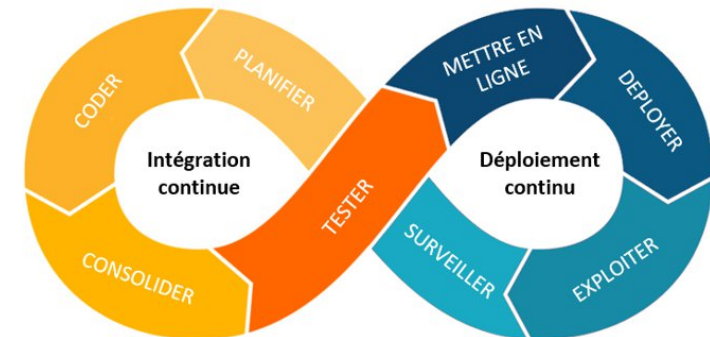
- "Le déploiement continu est une approche d'ingénierie logicielle dans laquelle les équipes produisent des logiciels dans des cycles courts, ce qui permet de le mettre à disposition à n'importe quel moment. Le but est de construire, tester et diffuser un logiciel plus rapidement. L'approche aide à réduire le coût, le temps et les risques associés à la livraison de changement en adoptant une approche plus incrémentale des modifications en production. Un processus simple et répétable de déploiement est un élément clé."



1. Test & intégration continue

L'intégration continue – Détail de chaque étapes

- Planifier (Plan) : définir les objectifs, les fonctionnalités et l'organisation
- Coder (Code) : réaliser les fonctionnalités selon les spécifications
- Consolider (Build) : fusion du travail de chacun via Git par exemple
- Tester (Test) : test du code à chaque commit
- Mettre en ligne (Release) : une fois les tests validés, création d'une release
- Déployer (Deploy) : une fois testée, la release est déployée en production
- Exploiter (Operate) & Surveiller (Monitor) : une fois en ligne, le service sera surveillé afin de détecter des problèmes ou erreurs possibles et prévoir des améliorations



1. Test & intégration continue

L'intégration continue – comment la mettre en place

- Github et Gitlab proposent tous les deux une solution pour mettre en place un pipeline CI/CD

Choose a workflow

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.

Skip this and [set up a workflow yourself](#) →

Q Search workflows

Suggested for this repository

Docker image
By GitHub Actions
Build a Docker image to deploy, run, or push to a registry.
[Configure](#) Dockerfile

Deno
By GitHub Actions
Test your Deno project
[Configure](#) JavaScript

Grunt
By GitHub Actions
Build a NodeJS project with npm and grunt.
[Configure](#) JavaScript

Jekyll
By GitHub Actions
Package a Jekyll site using the jekyll/builder Docker image.
[Configure](#) HTML

Gulp
By GitHub Actions
Build a NodeJS project with npm and gulp.
[Configure](#) JavaScript

Webpack
By GitHub Actions
Build a NodeJS project with npm and webpack.
[Configure](#) JavaScript

Continuous integration

[View all](#)

Publish Docker Container
By GitHub Actions
Build, test and push Docker image to GitHub Packages.
[Configure](#) Dockerfile

Node.js
By GitHub Actions
Build and test a Node.js project with npm.
[Configure](#) JavaScript

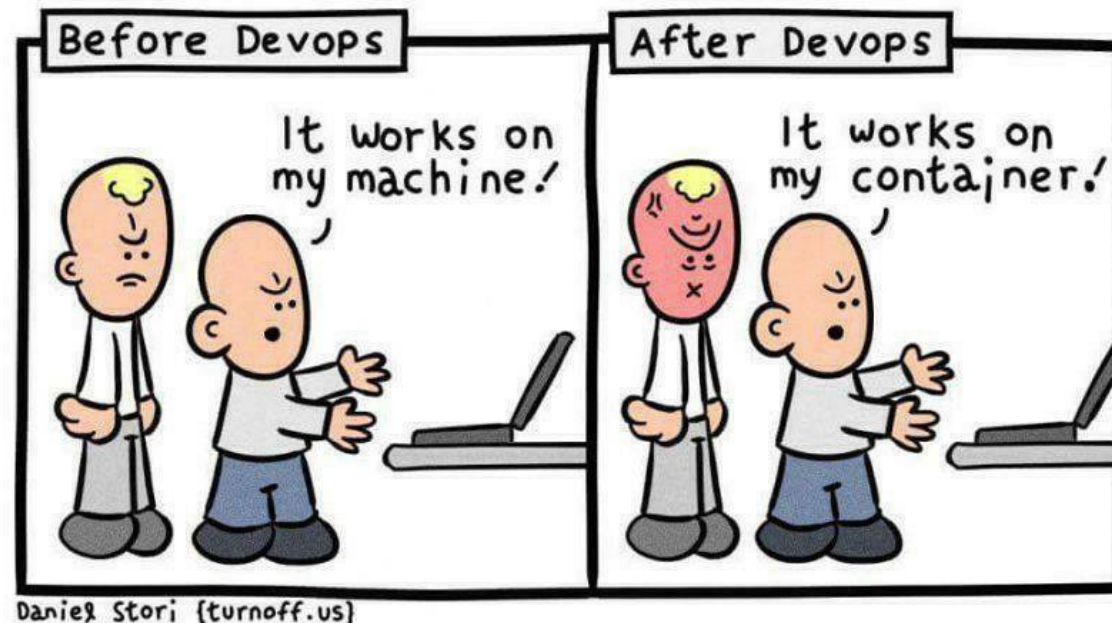
Publish Node.js Package to GitHub Packages
By GitHub Actions
Publishes a Node.js package to GitHub Packages.
[Configure](#) JavaScript

Publish Node.js Package
By GitHub Actions
Publishes a Node.js package to npm.
[Configure](#) JavaScript

1. Test & intégration continue

L'intégration continue

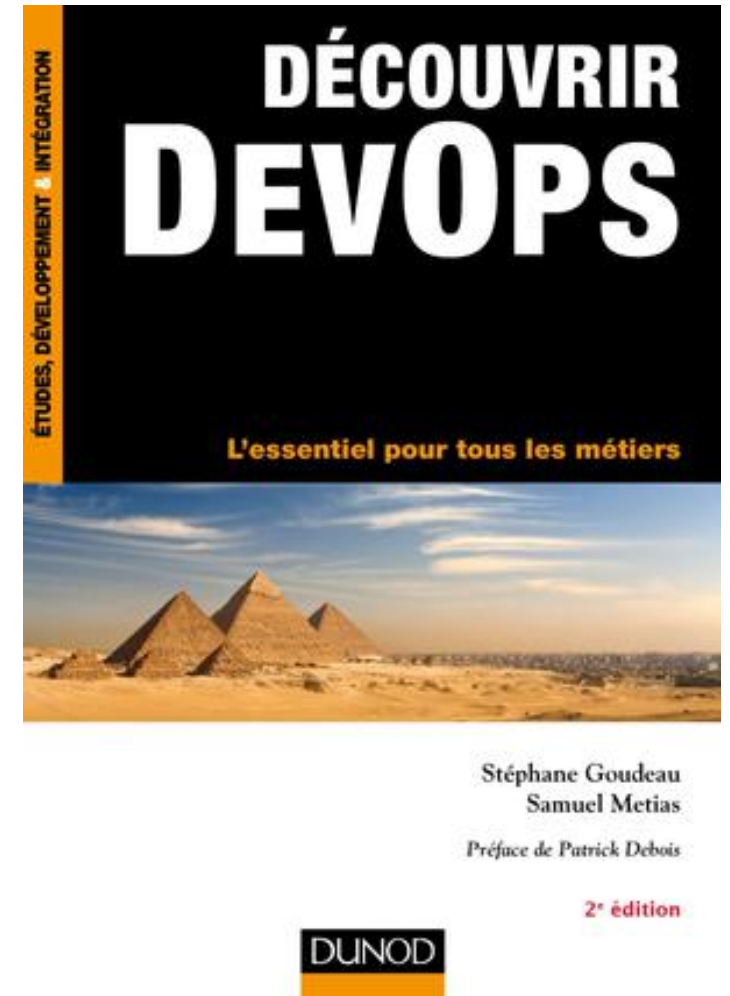
- Le but de tout cela est d'avoir des processus à suivre qui permettront d'avoir un système plus facilement reproductible, et d'éviter le fameux « ça marche sur ma machine »



1. Test & intégration continue

Pour aller plus loin

- La logique CI/CD appartient au métier de DevOps que nous avons déjà abordé au Chapitre 1
- Découvrir DevOps, l'essentiel pour tous les métiers
 - <https://ionis.scholarvox.com/catalog/book/docid/88859262>



1. Test & intégration continue

Pour aller plus loin

- Stackshare propose de partager les stacks techniques des startups tech. C'est un bon moyen de faire de la veille technologique et de comprendre le fonctionnement des services que nous utilisons au quotidien :
 - <https://stackshare.io/stacks>



Instagram

Instagram is an online photo-sharing and social networking service that enables its users to take pictures, apply digital filters to them, and share them on a variety ...

Stack

Members

Application and Data (18)



JavaScript



Python



React



Java



NGINX



PostgreSQL



Redis



Django



GraphQL



React Native



Redux



Objective-C



Memcached



Cassandra



Gunicorn



Immutable.js



Gearman



ReasonML

DevOps (8)



Webpack



Babel



Sentry



Jest



HAProxy



Fabric



Flow (JS)



Nuclide

1. Test & intégration continue

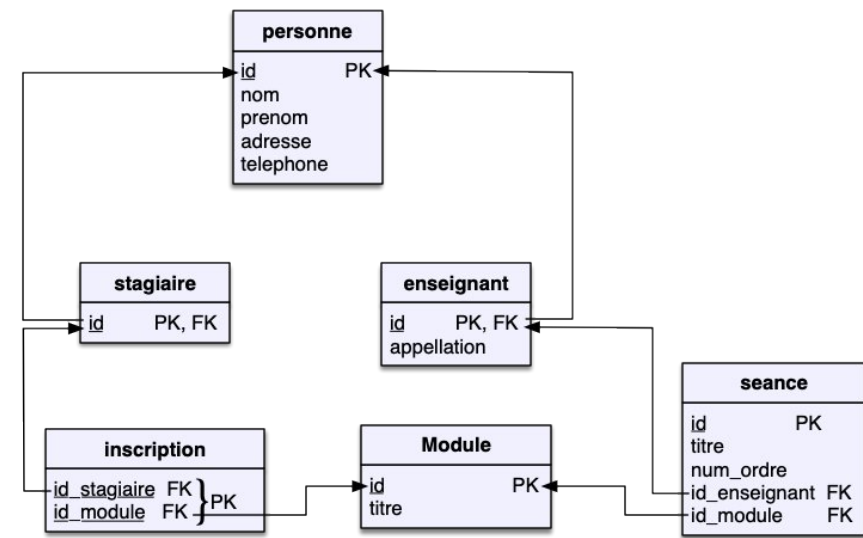


2. Documenter son projet

2. Documenter son projet

Qu'est-ce qu'une documentation de projet

- Quand on parle d'un projet logiciel, il existe différents types de documentation :
 - La documentation technique : c'est la documentation globale du projet, qui détaille l'architecture du système, les diagrammes de classes, les diagrammes de bases de données, etc.



2. Documenter son projet

Qu'est-ce qu'une documentation de projet

- Quand on parle d'un projet logiciel, il existe différents types de documentation :
 - La documentation de code : c'est la documentation du code, qui va expliquer le rôle des différentes fonctions, les normes de codes, etc.

PEP 8 – Style Guide for Python Code

Author: Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Alyssa Coghlan <ncoghlan at gmail.com>

Status: [Active](#)

Type: [Process](#)

Created: 05-Jul-2001

Post-History: 05-Jul-2001, 01-Aug-2013

▼ Table of Contents

- [Introduction](#)
- [A Foolish Consistency is the Hobgoblin of Little Minds](#)
- [Code Lay-out](#)
 - [Indentation](#)
 - [Tabs or Spaces?](#)
 - [Maximum Line Length](#)
 - [Should a Line Break Before or After a Binary Operator?](#)
 - [Blank Lines](#)

Format

```
>>> arrow.utcnow().format('YYYY-MM-DD HH:mm:ss ZZ')
'2013-05-07 05:23:16 -00:00'
```

Convert

Convert from UTC to other timezones by name or tzinfo:

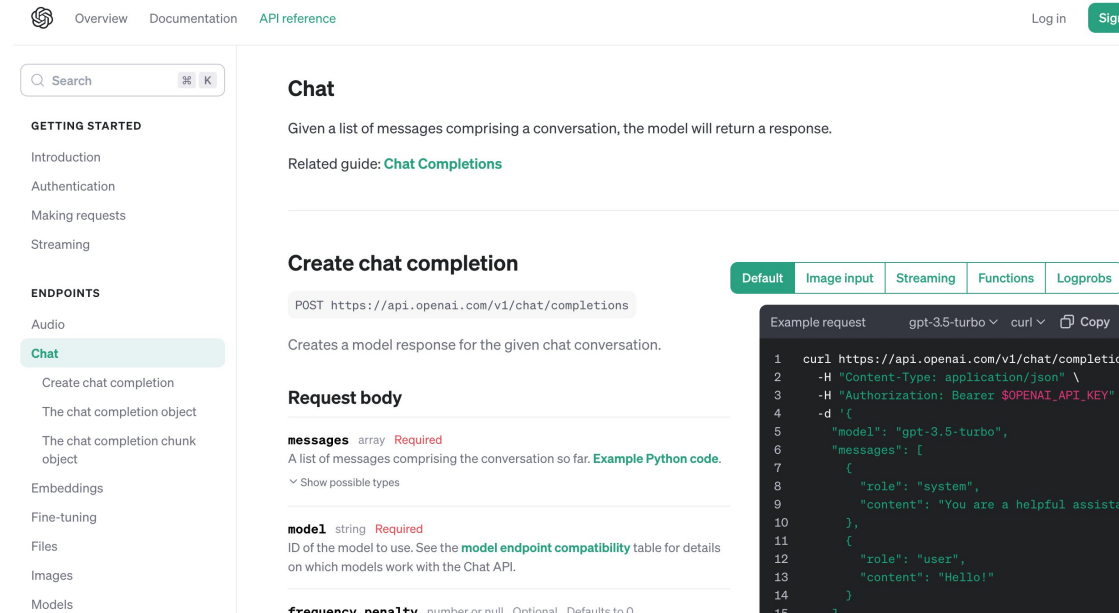
```
>>> utc = arrow.utcnow()
>>> utc
<Arrow [2013-05-07T05:24:11.823627+00:00]>

>>> utc.to('US/Pacific')
<Arrow [2013-05-06T22:24:11.823627-07:00]>
```

2. Documenter son projet

Qu'est-ce qu'une documentation de projet

- Quand on parle d'un projet logiciel, il existe différents types de documentation :
 - La documentation API : si le projet a une API, il est important d'avoir une documentation qui liste les endpoints et comment les utiliser (exemple avec OpenAI)



The screenshot shows the OpenAI API documentation page for the Chat endpoint. The page is titled "Chat" and describes the endpoint as "Given a list of messages comprising a conversation, the model will return a response." It includes a "Related guide: Chat Completions" link. The "Create chat completion" section shows the endpoint "POST https://api.openai.com/v1/chat/completions" and its purpose: "Creates a model response for the given chat conversation." The "Request body" section lists the required parameters: "messages" (array, required), "model" (string, required), and "frequency_penalty" (number or null, optional). An example request is shown in a code block, demonstrating a curl command with headers for content type and authorization, and a JSON body with a system message and a user message.

Overview Documentation **API reference** Log in Sign up

Search 36 K

GETTING STARTED

- Introduction
- Authentication
- Making requests
- Streaming

ENDPOINTS

- Audio
- Chat**
- Create chat completion
- The chat completion object
- The chat completion chunk object
- Embeddings
- Fine-tuning
- Files
- Images
- Models

Chat

Given a list of messages comprising a conversation, the model will return a response.

Related guide: [Chat Completions](#)

Create chat completion

POST https://api.openai.com/v1/chat/completions

Creates a model response for the given chat conversation.

Request body

messages array **Required**
A list of messages comprising the conversation so far. [Example Python code.](#)
Show possible types

model string **Required**
ID of the model to use. See the [model endpoint compatibility](#) table for details on which models work with the Chat API.

frequency_penalty number or null. Optional. Defaults to 0.

Default Image input Streaming Functions Logprobs

Example request gpt-3.5-turbo curl Copy

```
1 curl https://api.openai.com/v1/chat/completions
2 -H "Content-Type: application/json" \
3 -H "Authorization: Bearer $OPENAI_API_KEY" \
4 -d '{
5   "model": "gpt-3.5-turbo",
6   "messages": [
7     {
8       "role": "system",
9       "content": "You are a helpful assistant"
10    },
11    {
12      "role": "user",
13      "content": "Hello!"
14    }
15  ]
16 }
```

Qu'est-ce qu'une documentation de projet

- | | A | B | C | D | E | F | G | H | I | J | K |
|----|-----------------|--|-----------------------|---|--------------------------|------------------|---|--|---|---|---|
| 1 | Test Case ID | BU_001 | Test Case Description | Test the Login Functionality in Banking | | | | | | | |
| 2 | Created By | Mark | Reviewed By | Bill | Version | 2.1 | | | | | |
| 3 | | | | | | | | | | | |
| 4 | QA Tester's Log | Review comments from Bill incorporated in version 2.1 | | | | | | | | | |
| 5 | | | | | | | | | | | |
| 6 | Tester's Name | Mark | Date Tested | 1-Jan-2025 | Test Case (Pass/Fail/Not | Pass | | | | | |
| 7 | | | | | | | | | | | |
| 8 | S # | Prerequisites: | | | S # | Test Data | | | | | |
| 9 | 1 | Access to Chrome Browser | | | 1 | UserId = mg12345 | | | | | |
| 10 | 2 | | | | 2 | Pass = df12@434c | | | | | |
| 11 | 3 | | | | 3 | | | | | | |
| 12 | 4 | | | | 4 | | | | | | |
| 13 | | | | | | | | | | | |
| 14 | Test Scenario | Verify on entering valid userid and password, the customer can login | | | | | | | | | |
| 15 | | | | | | | | | | | |
| 16 | Step # | Step Details | | Expected Results | | Actual Results | | Pass / Fail / Not executed / Suspended | | | |
| 17 | | | | | | | | | | | |
| 18 | 1 | Navigate to
http://demo.guru99.com | | Site should open | | As Expected | | Pass | | | |
| 19 | 2 | Enter UserID & Password | | Credential can be entered | | As Expected | | Pass | | | |
| 20 | 3 | Click Submit | | Customer is logged in | | As Expected | | Pass | | | |
| 21 | 4 | | | | | | | | | | |
| 22 | | | | | | | | | | | |

2. Documenter son projet

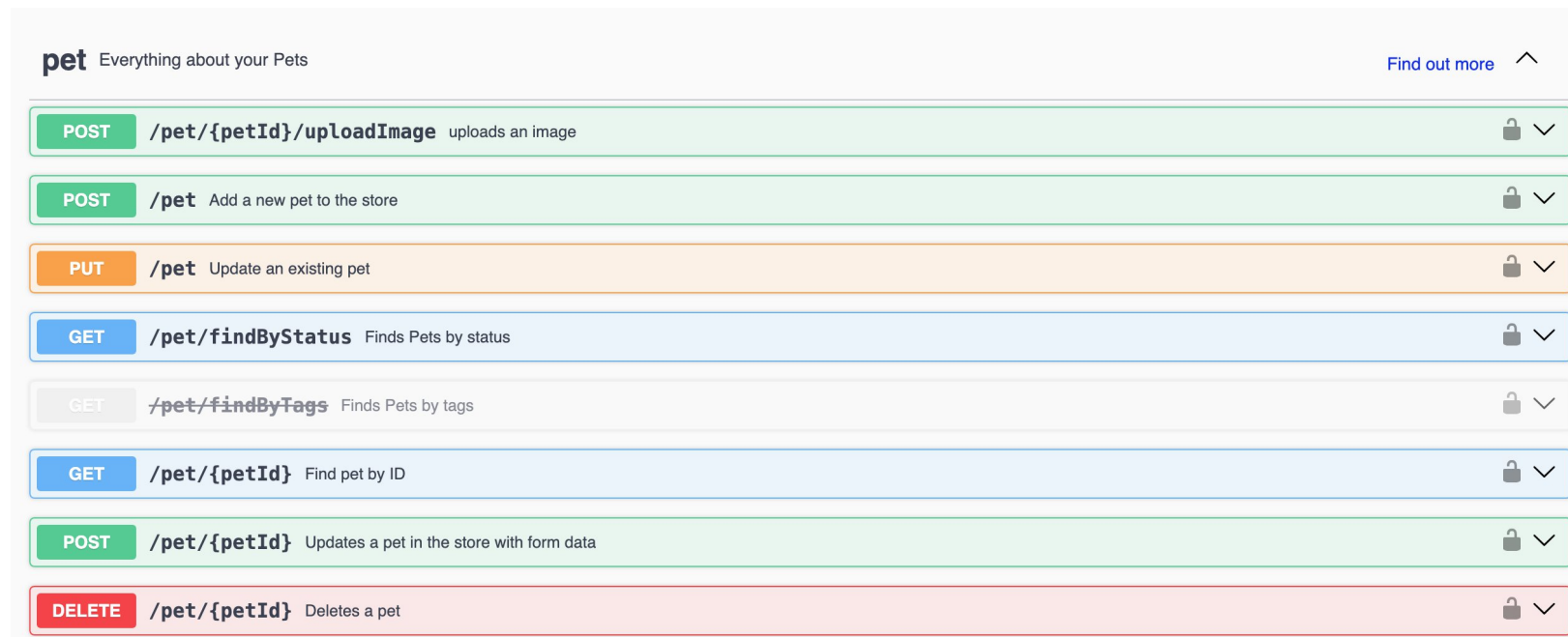
Qu'est-ce qu'une documentation de projet

- Comme toujours, chaque entreprise a sa façon de travailler qui lui est propre par rapport à la politique de documentation de projet.
- Ce qu'il faut retenir c'est que plus un projet sera gros, plus la documentation aura son importance.
- Elle permettra le partage de connaissances, et donc de faciliter la maintenabilité du système et son évolution. Et également, elle permettra d'avoir un historique le jour il y aura besoin de comprendre pourquoi le système a été réalisé d'une certaine façon

2. Documenter son projet

Les outils pour réaliser une documentation de projet – en ligne

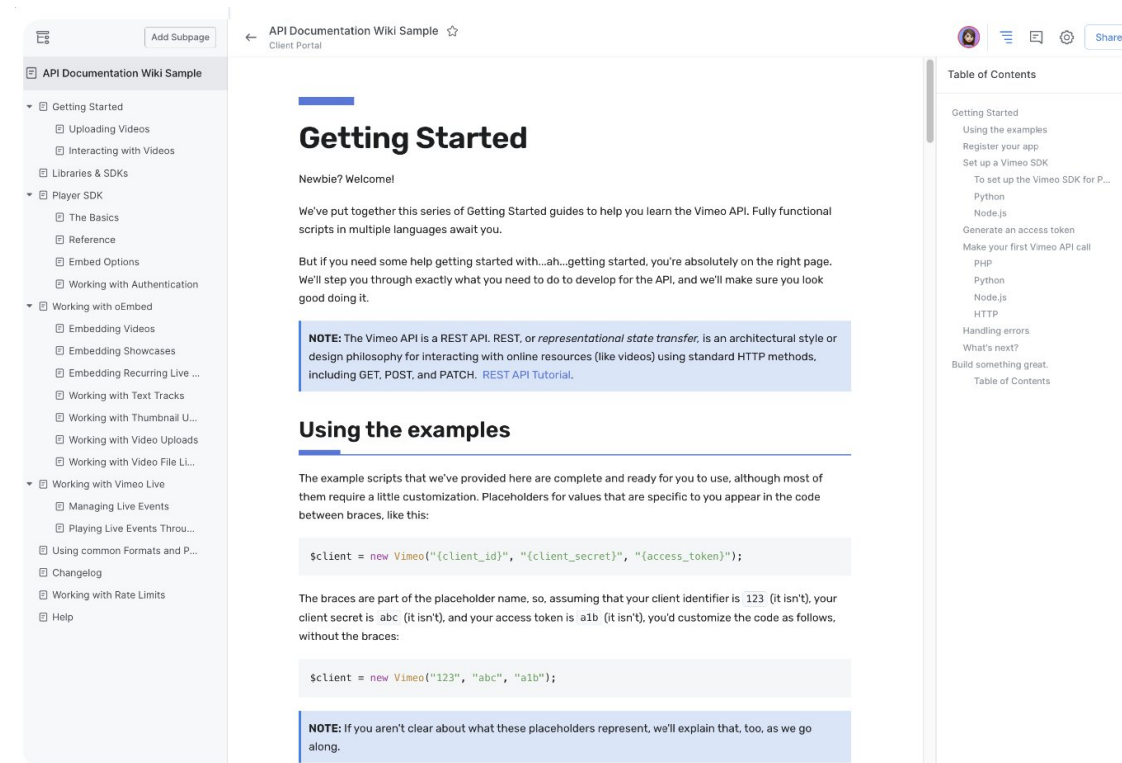
- Les outils en lignes qui permettent de réaliser une documentation :
 - Swagger, permet de réaliser une documentation d'API très facilement



2. Documenter son projet

Les outils pour réaliser une documentation de projet – en ligne

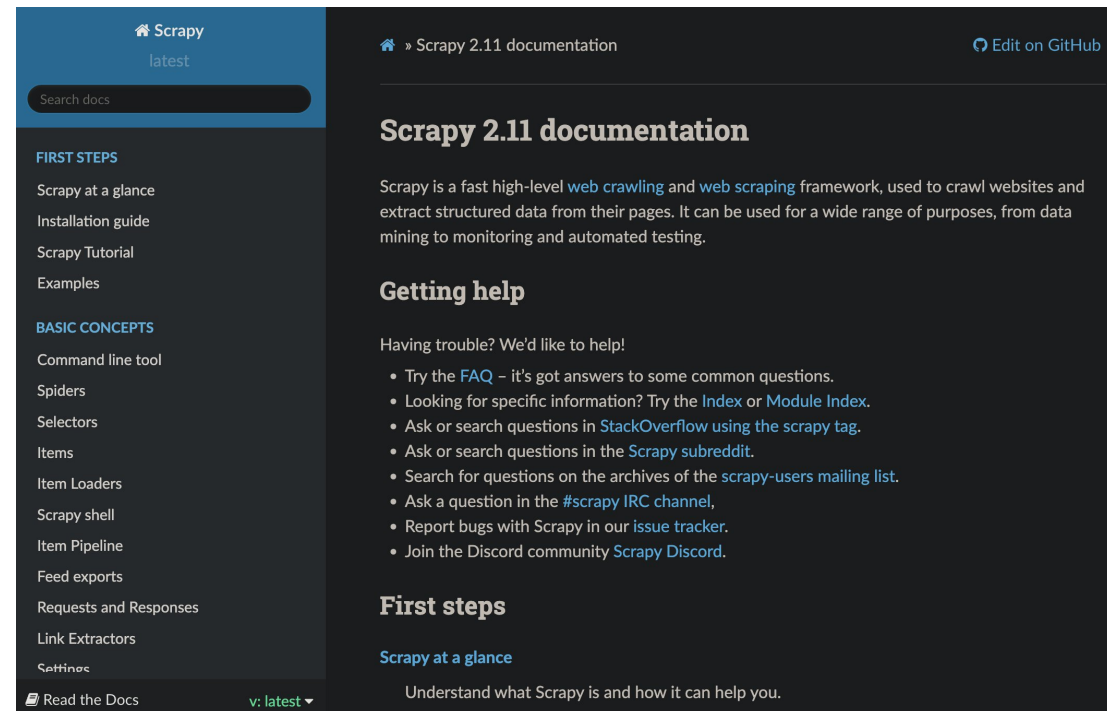
- Les outils en lignes qui permettent de réaliser une documentation :
 - Bit.ai, permet de réaliser une documentation technique et wiki



2. Documenter son projet

Les outils pour réaliser une documentation de projet – en ligne

- Les outils en lignes qui permettent de réaliser une documentation :
 - Read the docs, spécialisé sur la documentation technique



2. Documenter son projet

Les outils pour réaliser une documentation de projet – les librairies

- Comme pour les tests, il existe des librairies permettant de réaliser sa documentation technique avec chaque langage. Par exemple :
 - Avec Python on peut utiliser Sphinx
 - Avec Javascript on peut utiliser JSDoc
 - Avec PHP on peut utiliser phpDocumentor



2. Documenter son projet

Pour aller plus loin

- Apprendre en lisant des documentations techniques :
 - <https://google.github.io/styleguide/>
 - <https://github.com/readthedocs-examples/awesome-read-the-docs>
 - <https://docs.python.org/3/>

2. Documenter son projet



