

Chapitre 6 : Les cloud IoT

Les plateformes IoT fournissent des ressources et des outils pour rendre une solution IoT plus simple à utiliser et à moindre coût pour les entreprises, les développeurs et les utilisateurs. Un cloud IoT est généralement une plateforme IoT en tant que service (PaaS). Ce type de solution permet de louer une infrastructure cloud et une plateforme IoT auprès d'un seul fournisseur de cloud computing. Une plateforme IoT contribue à faciliter la communication, le stockage du flux de données, la gestion des appareils (objet IOT) et les fonctionnalités applicatives.

Les plateformes IoT contribuent à :

- connecter et gérer le matériel(device), comme les capteurs et les dispositifs (les objets IoT),
- gérer différents protocoles de communication hardware et software(http, Coap, MQTT...),
- assurer la sécurité et l'authentification des dispositifs et des utilisateurs(Token, clés...),
- collecter, visualiser et analyser les données recueillies par les capteurs et les appareils avec des applications de dashboarding (Web/Mobile),
- intégrer tout ce qui précède avec les systèmes d'entreprise existants.

1. Top des plates-formes IoT

Le marché de l'IoT (Internet des objets) évolue à un rythme rapide. Le rapport affirme que la valeur marchande mondiale des plates-formes IoT atteindra 74,74 milliards de dollars d'ici 2023. La raison de cette croissance est l'énorme demande de plates-formes IoT, d'appareils IoT et d'autres composants.

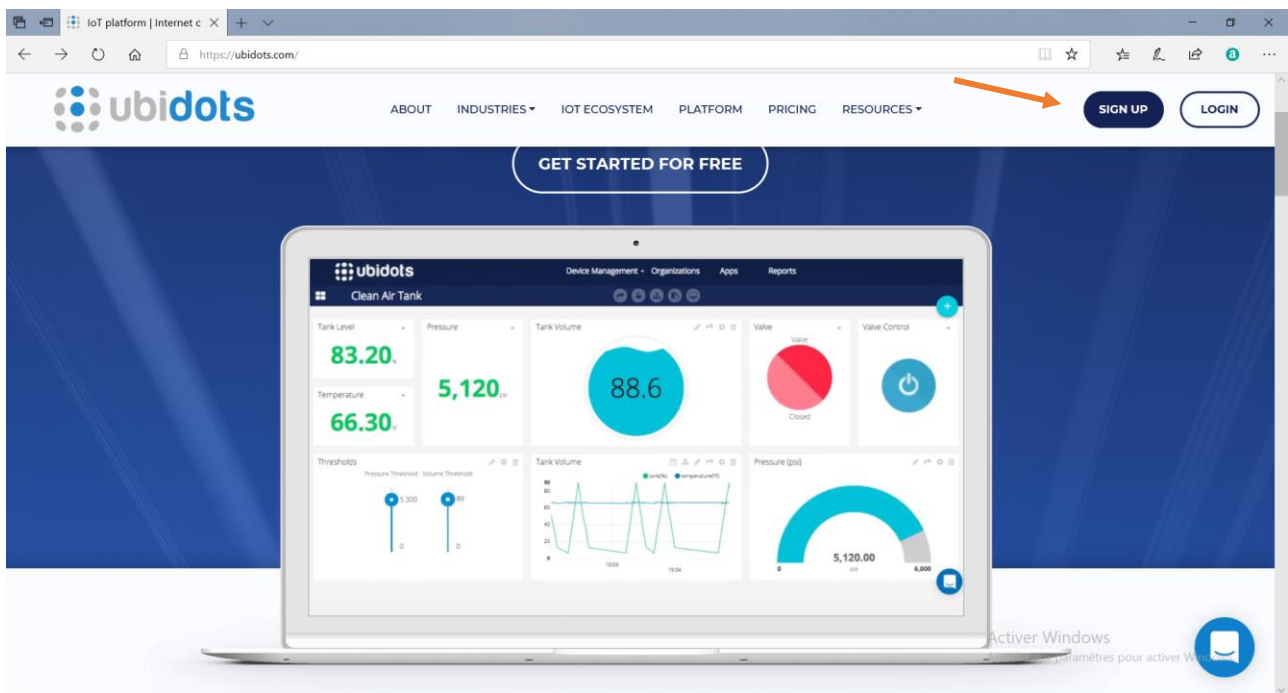
Un développeur IOT doit choisir l'une des meilleures plates-formes IoT parmi toutes celles qui sont présentes sur le marché en fonction des besoins de l'entreprise. Voici les plateformes IOT les plus connues :

- Thingworx 8 IoT platform
- AWS IoT platform
- Microsoft Azure IoT Suite
- Google Cloud's IoT platform
- PubNub
- IBM Watson IoT platform
- Cisco IoT Cloud connect
- Salesforce IoT cloud
- Kaa IoT platform
- Oracle IoT platform
- Thingspeak IoT platform
- Ubidots
- INTEL iot
- Things io

2. Cloud IOT Ubidots

Une fois la carte Raspberry est fonctionnelle on voulait bien ramener les données collectées depuis les capteurs: température, humidité... au service Cloud pour qu'elle soient accessibles à l'utilisateur final via une interface Web ou une application mobile.

On va utiliser un cloud orienté Internet Of Things appelé Ubidots voici son lien : <https://www.ubidots.com/>
<https://ubidots.com/stem>



Dans la suite on va voir les étapes nécessaires pour établir la communication entre la carte Raspberry et le Cloud Ubidots ainsi que l'utilisation de widget pour créer un tableau de bord(Dashboard) permettant d'afficher les données (température, humidité, position géographique) collectées par les capteurs d'une manière ergonomique.

Etape1 : S'inscrire au site

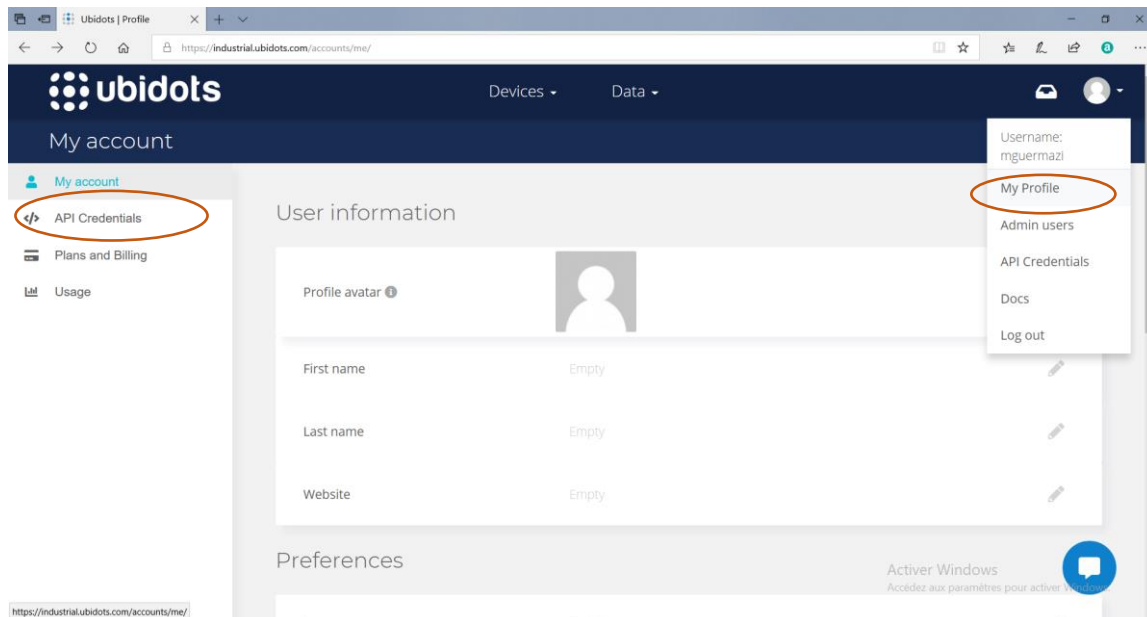
- cliquer sur Sign-Up et fournir les informations nécessaires
- choisir



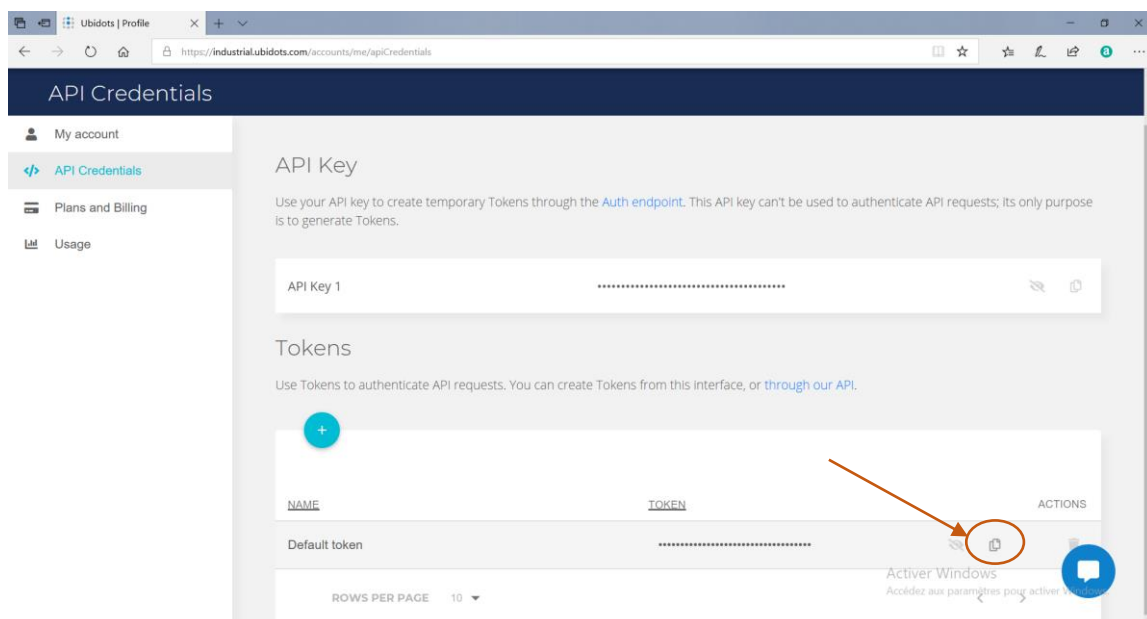
- fournir un nom de compte, un email et un mot de passe valide,
- cocher « My IOT Project is for personnel non-commercial use»
- cliquer sur SIGN UP FOR FREE

Etape2 : récupérer la valeur du jeton (token) qui permettra de se connecter au cloud

- Cliquer sur « My profile »
- Cliquer sur </> API Credentials



- Faire une copie de la valeur du jeton dans un autre endroit, par exemple au « Bloc-notes »



Etape 3 : Connecter la carte Raspberry Pi à Ubidots

À la suite de cette étape, on va envoyer des valeurs de données du Raspberry Pi à Ubidots.

Prérequis : Une Raspberry Pi (carte physique ou machine virtuelle) connectée à Internet + interpréteur python

- Mise à jour des référentiels et installation des bibliothèques et mise à jour des référentiels apt-get:
`$ sudo apt-get update`
`$ sudo apt-get upgrade`

Pour vérifier que python est installé sur la carte, depuis un terminal lancer la commande python.

3. Le protocole applicatif HTTP

Requests est une bibliothèque Python populaire qui simplifie les requêtes HTTP à partir de n'importe quel script python qui peut être exécuté dans le terminal de votre ordinateur ou tout périphérique Linux intégré tel que le Raspberry Pi.

Pour installer la bibliothèque, exécutez la commande ci-dessous dans le terminal Raspberry Pi:

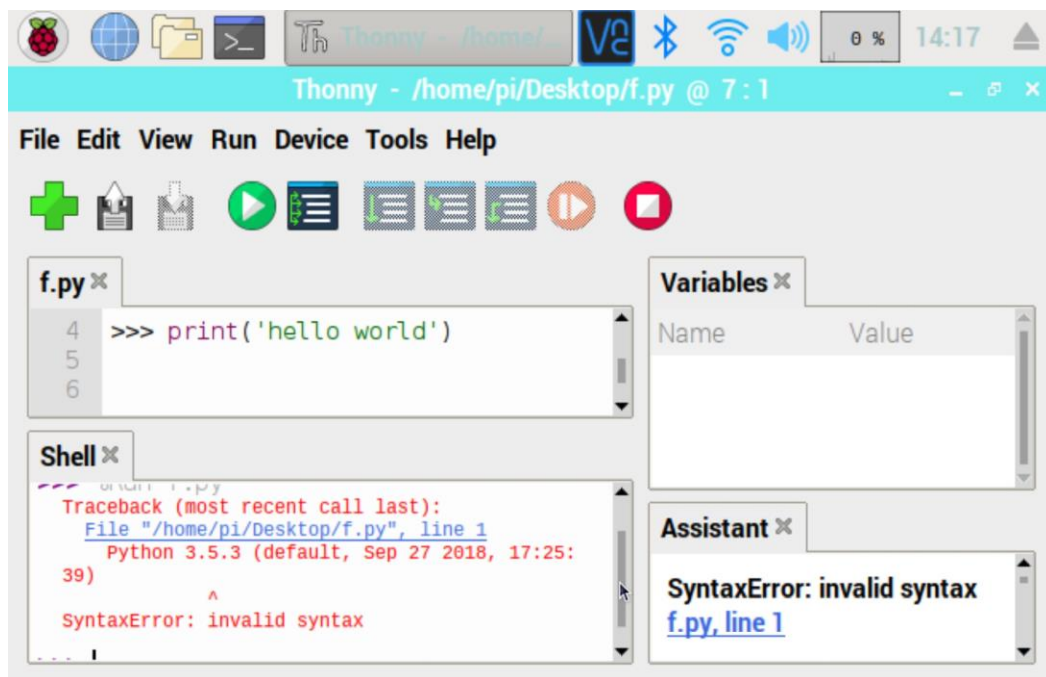
```
$ pip install requests
```

Envoyer des valeurs de données à Ubidots

Le script au-dessous envoie des valeurs de température d'humidité et de position géographique à Ubidots. Pour l'instant ces données sont générées dans un intervalle de valeurs par la carte raspberry, mais il peuvent être lues à partir d'un capteurs ou reçues d'un réseau de capteurs sans fils(RCSF).

Créons le script python à l'aide de l'éditeur de texte préféré. On peut utiliser tout simplement l'éditeur "nano" ou l'environnement de développement intégré **Thonny** pour Python.

```
$ nano PiToUbi.py
```



Alors utiliser le code suivant :

```
import time
import requests
import math
import random

TOKEN = "BBFF-rNZfXEmO0jCfrAQjfnVUukrLsS35OP" # Copier la valeur de ton TOKEN ici
DEVICE_LABEL = "Raspberry" # Label de la machine ou l'objet IOT qu'on va créer sur Ubidots
VARIABLE_LABEL_1 = "temperature" # Label de la 1ière variable capturée
VARIABLE_LABEL_2 = "humidite" # Label de la 2ième variable capturée
VARIABLE_LABEL_3 = "position" # Label de la 3ième variable capturée

def build_payload(variable_1, variable_2, variable_3):
    # Creates two random values for sending data
    value_1 = random.randint(30, 81) # calcul aléatoire de la température
    value_2 = random.randint(50, 71) # calcul aléatoire de l'humidité

    # Creates the gps coordinates
    lat = 34.757358 # valeur du latitude

    lng = 10.772115 # valeur du longitude

    payload = {variable_1: value_1,
                variable_2: value_2,
                variable_3: {"value": 34.10, "context": {"lat": lat, "lng": lng}}}
    # création du payload, c'est le champ Data du message à envoyer
    return payload

def post_request(payload):
    # Creates the headers for the HTTP requests
    url = "http://industrial.api.ubidots.com"
    url = "{}api/v1.6/devices/{}".format(url, DEVICE_LABEL)
    headers = {"X-Auth-Token": TOKEN, "Content-Type": "application/json"}

    # Makes the HTTP requests
    status = 400
    attempts = 0
    while status >= 400 and attempts <= 5:
        req = requests.post(url=url, headers=headers, json=payload)
        status = req.status_code
        attempts += 1
        time.sleep(1)
    # Processes results
    if status >= 400:
        print("[ERROR] Could not send data after 5 attempts, please check \
              your token credentials and internet connection")
    return False
```

```
print("[INFO] request made properly, your device is updated")
return True

def main():
    payload = build_payload(
        VARIABLE_LABEL_1, VARIABLE_LABEL_2, VARIABLE_LABEL_3)

    print("[INFO] Attempting to send data")
    post_request(payload)
    print("[INFO] finished")

if __name__ == '__main__':
    while (True):
        main()
        time.sleep(10)
```


Remarques :

- Copier la valeur de ton TOKEN dans la 6^{ème} ligne
- les lignes du code trop longs sont coupées avec le caractère \ , le reste se trouve sur la ligne suivante.
- Les commentaires débutent avec le caractère #

Etape4 : Exécuter le script python

Sur le terminal exécutez le script avec la commande :

```
$ python PiToUbi.py
```

Ou bien à partir du bouton Run  de Thonny

Etape5 : création du device Raspberry et des 3 variables

Une fois le script *PiToUbi.py* est exécuté, un Device(machine) **Raspberry** est créée automatiquement ainsi que les trois variables : température d'humidité et de position géographique.

Cliquer sur l'objet raspberry pour voir les valeurs des 3 variables. Vérifier que l'humidité et la température changent de valeurs chaque fois que la carte raspberry envoie les nouvelles données. Vous pouvez aussi rafraîchir la page web avec la touche F5.



3 Variables

<input type="checkbox"/>	Value	Name	Last updated ↓	
<input type="checkbox"/>	37	temperature	a day ago	⋮
<input type="checkbox"/>	34.1	position	a day ago	⋮
<input type="checkbox"/>	69	humidite	a day ago	⋮