TELEGRAM

Done! Congratulations on your new bot. You will find it at [t.me/HackIA24\_bot](https://t.me/HackIA24_bot) (https://t.me/HackIA24\_bot).

You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

Use this token to access the HTTP API:

--------------------------------------

Keep your token secure and store it safely, it can be used by anyone to control your bot.

For a description of the Bot API, see this page: <https://core.telegram.org/bots/api>

Créer une API sur Telegram consiste principalement à interagir avec l'API de Telegram pour créer un bot. Voici un guide étape par étape pour créer un bot Telegram :

### **Étape 1 : Créer un bot Telegram**

1. **Créer un compte Telegram** si vous n'en avez pas déjà un.
2. **Ouvrir une discussion avec BotFather**, l'outil de création de bots sur Telegram. Cherchez @BotFather dans la barre de recherche Telegram.
3. **Commencer une conversation avec BotFather** en cliquant sur "Start" ou en envoyant /start.
4. **Créer un nouveau bot** en envoyant la commande /newbot.
5. **Suivre les instructions** pour nommer votre bot et obtenir un jeton d'API (token). Ce token est crucial pour accéder à l'API de Telegram.

### **Étape 2 : Configurer l'environnement de développement**

Pour interagir avec l'API de Telegram, vous pouvez utiliser plusieurs langages de programmation. Dans cet exemple, nous utiliserons Python avec la bibliothèque python-telegram-bot.

1. **Installer Python** sur votre machine, si ce n'est pas déjà fait.
2. **Installer la bibliothèque python-telegram-bot**. Vous pouvez le faire en utilisant pip :  
   sh

pip install python-telegram-bot

### **Étape 3 : Écrire le code du bot**

Voici un exemple de script Python simple pour créer un bot qui répond à des messages :

python

import logging

from telegram import Update

from telegram.ext import Updater, CommandHandler, MessageHandler, Filters, CallbackContext

# Activer le logging

logging.basicConfig(

format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',

level=logging.INFO

)

logger = logging.getLogger(\_\_name\_\_)

# Fonction pour démarrer le bot

def start(update: Update, context: CallbackContext) -> None:

update.message.reply\_text('Bonjour! Je suis votre bot. Comment puis-je vous aider?')

# Fonction pour répondre aux messages

def echo(update: Update, context: CallbackContext) -> None:

update.message.reply\_text(update.message.text)

def main() -> None:

# Remplacer 'YOUR\_TOKEN' par le token de votre bot

updater = Updater("YOUR\_TOKEN")

dispatcher = updater.dispatcher

# Ajouter des gestionnaires de commandes et de messages

dispatcher.add\_handler(CommandHandler("start", start))

dispatcher.add\_handler(MessageHandler(Filters.text & ~Filters.command, echo))

# Démarrer le bot

updater.start\_polling()

# Laisser le bot tourner jusqu'à ce que vous interrompiez le script (Ctrl+C)

updater.idle()

if \_\_name\_\_ == '\_\_main\_\_':

main()

### **Étape 4 : Exécuter le bot**

Enregistrez le script dans un fichier, par exemple bot.py, et exécutez-le :

sh

python bot.py

Votre bot est maintenant en ligne et répondra aux messages que vous lui envoyez sur Telegram.

### **Étape 5 : Héberger le bot (optionnel)**

Pour garder votre bot en ligne 24/7, vous pouvez le déployer sur un serveur. Voici quelques options :

* **Heroku** : Un service cloud gratuit et simple d'utilisation.
* **AWS Lambda** : Pour des fonctions serverless.
* **Google Cloud Platform** : Pour des déploiements plus complexes.

### **Étape 6 : Ajouter des fonctionnalités**

Vous pouvez ajouter des fonctionnalités supplémentaires à votre bot en ajoutant plus de gestionnaires (handlers) pour différentes commandes ou types de messages. Par exemple, pour ajouter une commande /help :

python

def help\_command(update: Update, context: CallbackContext) -> None:

update.message.reply\_text('Voici les commandes disponibles : /start, /help')

dispatcher.add\_handler(CommandHandler("help", help\_command))