Prérequis Installation - Polymarket Trading Bot

Date : Janvier 2025 **Objectif :** Installer les outils nécessaires pour le développement du bot

---

1. Installation de Rust

Windows

1. **Télécharger rustup** (installateur officiel) :
   - Aller sur : https://rustup.rs/
   - Ou télécharger directement : https://win.rustup.rs/x86_64
2. **Exécuter l'installateur** :

```
# Télécharger et exécuter
Invoke-WebRequest -Uri https://win.rustup.rs/x86_64 -OutFile rustup-init.exe
.\rustup-init.exe
```

3. **Choisir l'installation par défaut** (option 1)
4. **Redémarrer le terminal** puis vérifier :

```
rustc --version
# Attendu : rustc 1.XX.X (XXXXXXXX YYYY-MM-DD)

cargo --version
# Attendu : cargo 1.XX.X (XXXXXXXX YYYY-MM-DD)
```

Prérequis Windows pour Rust

Rust nécessite les **Build Tools C++** de Visual Studio :

1. **Télécharger Visual Studio Build Tools** :
   - https://visualstudio.microsoft.com/visual-cpp-build-tools/
2. **Installer avec les composants** :
   - "Développement Desktop en C++"
   - Windows 10/11 SDK
   - MSVC v143 (ou plus récent)

---

2. Installation de Node.js (pour Tauri/Svelte)

Windows

1. **Télécharger Node.js LTS** :
   - https://nodejs.org/
   - Choisir la version **LTS** (Long Term Support)

2. **Vérifier l'installation** :

```
node --version
# Attendu : v20.X.X ou supérieur

npm --version
# Attendu : 10.X.X ou supérieur
```

---

3. Installation des outils Tauri

Prérequis système Tauri (Windows)

```
# WebView2 (généralement déjà installé sur Windows 10/11)
# Si besoin : https://developer.microsoft.com/microsoft-edge/webview2/
```

Installation CLI Tauri

```
# Installer le CLI Tauri globalement
cargo install tauri-cli

# Vérifier
cargo tauri --version
```

---

4. Outils de développement recommandés

IDE / Éditeur

- **VS Code** avec extensions :
  - `rust-analyzer` (Rust)
  - `Svelte for VS Code` (Svelte)
  - `Tauri` (Tauri)
  - `Even Better TOML` (Cargo.toml)
- **RustRover** (JetBrains) - Alternative complète

Installation extensions VS Code

```
code --install-extension rust-lang.rust-analyzer
code --install-extension svelte.svelte-vscode
code --install-extension tauri-apps.tauri-vscode
code --install-extension tamasfe.even-better-toml
```

---

5. Vérification finale

Après toutes les installations, exécuter ces commandes :

```
# Rust
rustc --version
cargo --version

# Node.js
node --version
npm --version

# Tauri CLI (après cargo install tauri-cli)
cargo tauri --version
```

Résultat attendu

```
rustc 1.75.0 (ou supérieur)
cargo 1.75.0 (ou supérieur)
v20.11.0 (ou supérieur)
10.2.0 (ou supérieur)
tauri-cli 2.0.0 (ou supérieur)
```

---

6. Configuration Git (optionnel mais recommandé)

```
git config --global user.name "Votre Nom"
git config --global user.email "votre@email.com"
```

---

7. Variables d'environnement

Créer un fichier `.env` à la racine du projet (sera créé automatiquement) :

```
# Polymarket API
POLY_API_KEY=votre_api_key
POLY_API_SECRET=votre_api_secret
POLY_API_PASSPHRASE=votre_passphrase

# Wallet
POLY_PRIVATE_KEY=0x...votre_cle_privee
POLY_PROXY_WALLET=0x...votre_proxy_wallet

# Configuration
RUST_LOG=info
```

Résumé des actions

Étape	Action	Temps estimé
1	Installer Visual Studio Build Tools	5-10 min

2	Installer Rust via rustup	5 min
3	Installer Node.js LTS	2 min
4	`cargo install tauri-cli`	5 min
5	Extensions VS Code (optionnel)	2 min

Total : ~20 minutes

En cas de problème

Erreur "linker not found"

→ Réinstaller Visual Studio Build Tools avec les composants C++

Erreur "cargo not found" après installation

→ Redémarrer le terminal ou le PC

Erreur WebView2

→ Télécharger manuellement : https://developer.microsoft.com/microsoft-edge/webview2/

Une fois tous les prérequis installés, relancez Claude Code pour continuer le développement.