# Polymarket-Kalshi Arbitrage Bot

Complete User Guide

Step-by-Step Documentation for Non-Technical Users

Chapter 1

# Getting Started

## Getting Started Guide

Welcome! This guide will help you set up and use the Polymarket-Kalshi Arbitrage Bot, even if you've never coded before.

### What This Bot Does

This bot automatically finds and executes risk-free arbitrage opportunities between two prediction market platforms: - Polymarket - A decentralized prediction market - Kalshi - A regulated prediction market

#### What is Arbitrage?

Arbitrage means buying and selling the same thing on different platforms to make a guaranteed profit. Here's a simple example:

- Polymarket: You can buy "YES" for a market at $0.40
- Kalshi: You can buy "NO" for the same market at $0.58
- Total cost: $0.98
- Guaranteed payout: $1.00 (one will always win)
- Your profit: $0.02 per contract (2% risk-free return!)

The bot automatically finds these opportunities and executes trades for you 24/7.

### What You'll Need

Before starting, make sure you have:

- ■ A computer (Windows, Mac, or Linux)

## Quick Overview

Here's what you'll do (don't worry, we'll guide you through each step):

- Install Rust (the programming language this bot uses)

## Next Steps

Ready to start? Follow these guides in order:

- Installation Guide - Installing Rust and setting up your computer

## Important Safety Notes

■■ Always start in DRY RUN mode - This lets you test the bot without risking real money. The bot is set to dry run by default, so your real money is safe until you're ready.

■■ Start with small amounts - Even when you go live, start with small positions to make sure everything works correctly.

■■ Monitor your bot - Check on it regularly, especially in the beginning.

---

Ready? Let's start with Installation!

Chapter 2

# Installation Guide

---

## Installation Guide

This guide will help you install everything you need to run the bot on your computer.

### Step 1: Install Rust

Rust is the programming language this bot is written in. Don't worry - you don't need to learn Rust to use the bot!

#### For Windows Users

- Download Rust installer:
- Go to: https://rustup.rs/
- Click the "DOWNLOAD RUSTUP-INIT.EXE" button
- Save the file to your Downloads folder

2. Run the installer: - Double-click rustup-init.exe in your Downloads folder - When prompted, press Enter to proceed with default installation - The installer will take a few minutes to download and install everything - When it says "Press the Enter key to continue", press Enter

3. Restart your computer: - Close all terminal/command prompt windows - Restart your computer (this is important!)

4. Verify installation: - Open PowerShell or Command Prompt - Type: rustc --version - You should see something like: rustc 1.75.0 (or higher) - If you see an error, try restarting your computer again

#### For Mac Users

- Open Terminal (Press Cmd + Space, type "Terminal", press Enter)

2. Run this command: curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh

3. Follow the prompts: - Press Enter when asked about default installation - Type 1 and press Enter to proceed

4. Restart Terminal: - Close and reopen Terminal, or run: source $HOME/.cargo/env

5. Verify installation: rustc --version - You should see a version number like rustc 1.75.0

### For Linux Users

- Open Terminal

2. Run this command: curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh

3. Follow the prompts: - Press Enter for default installation - Type 1 and press Enter

4. Reload your shell: source $HOME/.cargo/env

5. Verify installation: rustc --version

## Step 2: Download the Bot Code

### Using Git (Recommended)

If you have Git installed:

- Open Terminal/PowerShell/Command Prompt

2. Navigate to where you want to save the bot: cd Desktop (or wherever you want to save it)

3. Clone the repository: git clone https://github.com/terauss/prediction-market-arbitrage.git

4. Go into the folder: cd prediction-market-arbitrage

### Without Git (Download ZIP)

- Go to the GitHub repository:
- Visit: https://github.com/terauss/prediction-market-arbitrage
- Click the green "Code" button

- Click "Download ZIP"

2. Extract the ZIP file: - Find the downloaded ZIP file - Right-click and choose "Extract All" (Windows) or double-click (Mac/Linux) - Remember where you extracted it!

3. Open Terminal/PowerShell in that folder: - Windows: In the extracted folder, right-click in empty space, choose "Open in Terminal" or "Open PowerShell window here" - Mac: Right-click folder, choose "New Terminal at Folder" - Linux: Right-click folder, choose "Open Terminal Here"

## Step 3: Install dotenvx (For Running the Bot)

The bot needs a tool called dotenvx to read your configuration file.

### For Windows (PowerShell)

```
iwr https://dotenvx.sh/install.ps1 -useb | iex
```

If you get an error about execution policy, run this first: Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser Then try the install command again.

### For Mac/Linux

```
curl -fsSL https://dotenvx.sh/install.sh | sh
```

### Verify dotenvx Installation

Close and reopen your terminal, then run: dotenvx --version

You should see a version number.

## Step 4: Build the Bot

Now let's compile the bot code so it's ready to run:

- Make sure you're in the bot folder:

2. Build the bot: cargo build --release

3. Wait for it to finish: - This will take 5-15 minutes the first time (downloading dependencies) - You'll see lots of text scrolling - this is normal! - When you see "Finished release [optimized] target(s)", you're done!

## Troubleshooting Installation

### "cargo: command not found"

Windows: Restart your computer after installing Rust, then open a NEW PowerShell window.

Mac/Linux: Run source $HOME/.cargo/env or restart Terminal.

### Build errors or network issues

- Make sure you have internet connection
- Try again - sometimes it's just a temporary network issue
- If it keeps failing, check you have enough disk space (need at least 2GB free)

### "dotenvx: command not found"

- Close and reopen your terminal completely
- Make sure you ran the installation command correctly
- Try the verification command again

## What's Next?

Great! You've installed everything. Now you need to:

- Get your credentials - Get API keys from Kalshi and Polymarket

---

Ready? Let's get your API keys: Getting Your Credentials

Chapter 3

# Getting Your Credentials

## Getting Your Credentials

To use the bot, you need to give it permission to trade on your behalf using API keys. Think of API keys like passwords that allow the bot to access your accounts.

■■ IMPORTANT: Keep your API keys secret! Never share them with anyone or post them online. They give full access to your accounts.

### Part 1: Getting Kalshi Credentials

Kalshi requires two things: 1. An API Key ID (like a username) 2. A Private Key file (like a password file)

#### Step 1: Log into Kalshi

- Go to https://kalshi.com

#### Step 2: Create an API Key

- Go to Settings:
- Click on your profile/account icon (usually top right)
- Click "Settings" or "Account Settings"

2. Find API Keys section: - Look for a tab or section called "API Keys" or "Developer Settings" - Click on it

3. Create a new API key: - Click "Create New API Key" or "Generate API Key" button - You may be asked to give it a name (e.g., "Arbitrage Bot") - Make sure it has trading permissions enabled - Click "Create" or "Generate"

4. Save your API Key ID: - You'll see an API Key ID (looks like a long string of characters) - WRITE THIS DOWN or copy it - you'll need it later! - Example: AKIAIOSFODNN7EXAMPLE

5. Download your Private Key: - Click "Download Private Key" or similar button - This will download a .pem file (like kalshiprivatekey.pem) - SAVE THIS FILE SOMEWHERE SAFE - you'll need it in the next step - ■■ You can only download this once! Make sure you save it.

### Step 3: Save Your Kalshi Private Key

- Move the downloaded file to the bot folder:
- Find the .pem file you just downloaded
- Copy or move it to your bot folder (prediction-market-arbitrage)
- You can rename it to kalshiprivatekey.pem to make it easier

2. Note the full path to the file: - Windows example: C:\Users\YourName\Desktop\prediction-market-arbitrage\kalshiprivatekey.pem - Mac/Linux example: /Users/YourName/Desktop/prediction-market-arbitrage/kalshiprivatekey.pem - You'll need this path in the configuration step!

## Part 2: Getting Polymarket Credentials

Polymarket uses your Ethereum wallet as your account. You need: 1. Your Wallet Address (your account number) 2. Your Private Key (the password to your wallet)

### Step 1: Have a Wallet Ready

You need an Ethereum wallet that works on Polygon network. The most common options:

Option A: MetaMask (Recommended for beginners)

- Install MetaMask:
- Go to https://metamask.io
- Click "Download" and install the browser extension
- Create a new wallet or import an existing one
- ■■ SAVE YOUR SECRET RECOVERY PHRASE - write it down somewhere safe!

2. Add Polygon Network: - Open MetaMask - Click the network dropdown (usually says "Ethereum Mainnet") - Click "Add Network" or "Add a network manually" - Enter these details: - Network Name: Polygon - RPC URL: https://polygon-rpc.com - Chain ID: 137 - Currency Symbol: MATIC - Block Explorer: https://polygonscan.com - Click "Save"

Option B: Use Existing Wallet

If you already have a wallet, make sure it's set up for Polygon network.

## Step 2: Fund Your Wallet

- Get USDC on Polygon:
- Your Polymarket account needs USDC (USD Coin) on Polygon network
- You can bridge USDC from Ethereum to Polygon, or buy it directly on Polygon
- Make sure you have USDC, not just MATIC!

2. Check your balance: - In MetaMask, switch to Polygon network - You should see your USDC balance - You need at least some USDC to trade (start with a small amount for testing!)

## Step 3: Get Your Wallet Address

- In MetaMask:
- Click on your account name at the top (it shows "Account 1" or similar)
- Click to copy your address
- It starts with 0x followed by lots of letters and numbers
- Example: 0x742d35Cc6634C0532925a3b844Bc9e7595f0bEb
- WRITE THIS DOWN - this is your POLY_FUNDER!

## Step 4: Export Your Private Key

■■ WARNING: Your private key gives full access to your wallet. Never share it!

- In MetaMask:
- Click the three dots (menu) in the top right
- Click "Account Details"
- Click "Export Private Key"

- Enter your MetaMask password
- Click to reveal your private key
- It starts with 0x followed by 64 characters
- Example: 0x1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef
- COPY THIS CAREFULLY - you'll need it in the configuration step!

2. Save it securely: - Don't save it in a plain text file that others can access - Consider using a password manager - You'll paste it into the configuration file in the next step

## Quick Checklist

Before moving to configuration, make sure you have:

- ■ Kalshi API Key ID (copied somewhere safe)
- ■ Kalshi Private Key file (.pem file saved in your bot folder)
- ■ Polymarket Wallet Address (starts with 0x)
- ■ Polymarket Private Key (starts with 0x, 64 characters after)
- ■ Both accounts funded with money to trade

## Security Reminders

■ Never share your credentials with anyone ■ Don't commit your .env file to GitHub or share it online ■ Keep backups of your private keys in a safe place ■ If someone gets your private keys, they can steal your money!

## What's Next?

Now that you have all your credentials, let's set up the configuration file:

Configuration Setup →

Chapter 4

# Configuration Setup

---

## Configuration Setup Guide

Now we'll create a configuration file that tells the bot how to use your accounts. This file is called .env and contains all your settings in one place.

### Step 1: Create the .env File

- Navigate to your bot folder:

2. Create a new file called .env: - Windows: Right-click in the folder, choose "New > Text Document", rename it to .env (make sure to remove .txt extension) - Mac/Linux: In Terminal, run: touch .env

3. Open the .env file in a text editor: - Use Notepad (Windows), TextEdit (Mac), or any text editor - Not Microsoft Word! Use a plain text editor

### Step 2: Add Your Credentials

Copy and paste the following template into your .env file, then fill in YOUR actual values:

```bash
```

==========

## KALSHI CREDENTIALS

==========

KALSHIAPIKEYID=YOURKALSHIAPIKEYIDHERE
KALSHIPRIVATEKEYPATH=C:/full/path/to/kalshiprivate_key.pem

==========

## POLYMARKET CREDENTIALS

==========

POLYPRIVATEKEY=0xYOURPOLYMARKETPRIVATEKEYHERE
POLYFUNDER=0xYOURPOLYMARKETWALLETADDRESS_HERE

==========

## SYSTEM CONFIGURATION

==========

DRYRUN=1 RUSTLOG=info FORCEDISCOVERY=0 PRICELOGGING=0

==========

## TEST MODE (Leave as is for normal use)

==========

TESTARB=0 TESTARBTYPE=polyyeskalshino

==========

# CIRCUIT BREAKER SETTINGS (Risk Management)

==========

CBENABLED=true CBMAXPOSITIONPERMARKET=50000 CBMAXTOTALPOSITION=100000 CBMAXDAILYLOSS=500.0 CBMAXCONSECUTIVEERRORS=5 CBCOOLDOWNSECS=300 ```

## Step 3: Fill in Your Values

### Kalshi Credentials

- KALSHIAPIKEY_ID:
- Replace YOURKALSHIAPIKEYID_HERE with your actual Kalshi API Key ID
- Example: KALSHIAPIKEY_ID=AKIAIOSFODNN7EXAMPLE
- No spaces around the = sign!

2. KALSHIPRIVATEKEYPATH: - Replace with the full path to your .pem file - Windows examples: - KALSHIPRIVATEKEYPATH=C:\Users\John\Desktop\prediction-market-arbitrage\kalshiprivatekey.pem - Or: KALSHIPRIVATEKEYPATH=C:/Users/John/Desktop/prediction-market-arbitrage/kalshiprivatekey.pem - Mac/Linux examples: - KALSHIPRIVATEKEYPATH=/Users/john/Desktop/prediction-market-arbitrage/kalshiprivatekey.pem - Tip: You can also just put the filename if the file is in the same folder as .env: - KALSHIPRIVATEKEYPATH=kalshiprivate_key.pem

### Polymarket Credentials

- POLYPRIVATEKEY:
- Replace 0xYOURPOLYMARKETPRIVATEKEYHERE with your actual private key
- It should start with 0x and be 66 characters total
- Example: POLYPRIVATEKEY=0x1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef
- Keep the 0x at the beginning!

2. POLYFUNDER: - Replace 0xYOURPOLYMARKETWALLETADDRESSHERE with your wallet address - It should start with 0x and be 42 characters total - Example: POLYFUNDER=0x742d35Cc6634C0532925a3b844Bc9e7595f0bEb - Keep the 0x at the beginning!

## Step 4: Understand Configuration Options

### System Configuration

These settings control how the bot behaves:

| | Value | What It Does |
|---|---|---|
| | 1 = Safe mode 0 = Live trading | START WITH 1 - This makes the bot test without using real money. Only change to 0 when you're r |
| | info, debug, warn, error | How much detail to show in logs. info is good for most users. |
| Y | 0 = Use cache 1 = Refresh | Usually keep at 0. Set to 1 if markets aren't matching properly. |
| | 0 = Normal 1 = Verbose | Usually keep at 0. Set to 1 to see every price update (lots of output!). |

### Circuit Breaker Settings (Risk Management)

These protect you from losing too much money:

| Variable | Default | What It Does |
|---|---|---|
| CB_ENABLED | true | Turn circuit breaker on/off. Keep this true for safety! |
| CBMAXPOSITIONPERMARKET | 50000 | Maximum contracts to hold in any single market. Adjust based on your capital. |
| CBMAXTOTAL_POSITION | 100000 | Maximum total contracts across all markets. Adjust based on your capital. |
| CBMAXDAILY_LOSS | 500.0 | Maximum loss in dollars per day before bot stops. 500 = $500.00 |
| CBMAXCONSECUTIVE_ERRORS | 5 | How many errors before bot stops. Keep at 5. |
| CBCOOLDOWNSECS | 300 | How long to wait (seconds) after circuit breaker trips. 300 = 5 minutes. |

### Recommended Settings for Beginners

Start with these safe settings:

```
DRYRUN=1 CBMAXPOSITIONPERMARKET=100 CBMAXTOTALPOSITION=500 CBMAXDAILY_LOSS=100.0
```

This means: - ■ Bot runs in test mode (no real money) - ■ Max 100 contracts per market - ■ Max 500 contracts total - ■ Stops if you lose more than $100 in a day

## Step 5: Save and Verify

- Save the .env file:
- Make sure all your values are filled in correctly
- Double-check there are no extra spaces around the = signs
- Save the file

2. Verify the file exists: ```bash # Windows PowerShell Test-Path .env

# Mac/Linux ls -la .env `` Should return True` or show the file.

## Common Mistakes to Avoid

■ Don't put spaces around =: - Wrong: KALSHIAPIKEYID = ABC123 - Right: KALSHIAPIKEYID=ABC123

■ Don't use quotes (unless the value has spaces): - Wrong: KALSHIAPIKEYID="ABC123" - Right: KALSHIAPIKEYID=ABC123

■ Don't forget the 0x prefix for Polymarket values: - Wrong: POLYFUNDER=742d35Cc6634C0532925a3b844Bc9e7595f0bEb - Right: POLYFUNDER=0x742d35Cc6634C0532925a3b844Bc9e7595f0bEb

■ Don't commit .env to Git: - Make sure .env is in .gitignore (it should be by default) - Never upload your .env file to GitHub or share it!

## Example .env File (Fake Values)

Here's what a complete .env file might look like (with fake values):

```
```bash
```
```

## Kalshi

KALSHIAPIKEYID=AKIAIOSFODNN7EXAMPLE12345
KALSHIPRIVATEKEYPATH=kalshiprivatekey.pem

## Polymarket

POLYPRIVATEKEY=0x1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcde
f POLY_FUNDER=0x742d35Cc6634C0532925a3b844Bc9e7595f0bEb

## Settings

DRYRUN=1 RUSTLOG=info FORCEDISCOVERY=0 PRICELOGGING=0

## Circuit Breaker

CBENABLED=true CBMAXPOSITIONPERMARKET=100 CBMAXTOTALPOSITION=500
CBMAXDAILYLOSS=100.0 CBMAXCONSECUTIVEERRORS=5 CBCOOLDOWNSECS=300

## Test Mode

TESTARB=0 TESTARBTYPE=polyyeskalshino ```

### What's Next?

Perfect! Your configuration is set up. Now let's test it and run the bot:

Running the Bot →

Chapter 5

# Running the Bot

---

## Running the Bot Guide

Now that everything is set up, let's run your bot! This guide will show you how to start it and understand what it's doing.

### Before You Start

■ Checklist: - [ ] Rust is installed (rustc --version works) - [ ] Bot is built (cargo build --release completed successfully) - [ ] .env file is created with your credentials - [ ] DRY_RUN=1 in your .env file (we'll start in safe mode!) - [ ] You have funds in both Kalshi and Polymarket accounts

### Step 1: Make Sure You're in the Right Folder

Open Terminal/PowerShell and navigate to your bot folder:

```
cd prediction-market-arbitrage
```

(Or wherever you saved the bot)

### Step 2: Run the Bot (Dry Run Mode)

Start with dry run mode first! This lets you see what the bot would do without using real money.

```
dotenvx run -- cargo run --release
```

#### What Should Happen

You'll see output like this:

```
``` ■ Prediction Market Arbitrage System v2.0 Profit threshold: <0.5¢ (0.5% minimum profit) Monitored leagues: [] Mode: DRY RUN (set DRY_RUN=0 to execute) [KALSHI] API key loaded [POLYMARKET] Creating async client and deriving API credentials... [POLYMARKET] Client ready for 0x742d35Cc... ■ Loaded 1234 team code mappings ■ Market discovery... ■ Market discovery complete: - Matched market pairs: 45
```

The bot will then start monitoring markets and looking for arbitrage opportunities.

### Understanding the Output

Here's what each part means:

- ■ Prediction Market Arbitrage System v2.0 - Bot version
- Profit threshold - Minimum profit % needed to trade (0.5% default)
- Mode: DRY RUN - Safe mode, no real trades
- [KALSHI] API key loaded - Connected to Kalshi ✓
- [POLYMARKET] Client ready - Connected to Polymarket ✓
- Team code mappings - Markets matched between platforms
- Discovered market pairs - Markets ready to monitor

## Step 3: Let It Run

Once started, the bot will:

- Discover markets - Find matching markets between platforms

You'll see messages like:

```
[INFO] Monitoring 45 market pairs [INFO] Connected to Kalshi WebSocket [INFO] Connected to Polymarket WebSocket [DEBUG] Price update: LAL-GSW-YES @ 0.42 [DEBUG] Price update: LAL-GSW-NO @ 0.58
```

### What to Look For

When an arbitrage opportunity is found (in dry run mode), you'll see:

```
[INFO] ■ ARB OPPORTUNITY DETECTED (DRY RUN): Market: Lakers vs Warriors Type: polyyeskalshi_no Cost: $0.98 Profit: $0.02 (2.04%) Would execute: Buy Polymarket YES
```
```

```
@ $0.40, Buy Kalshi NO @ $0.58
```

In dry run mode, it shows what it would do but doesn't actually trade.

## Step 4: Run in Live Mode (When Ready)

■■ ONLY do this when you're confident everything works!

- Edit your .env file:
- Change DRYRUN=1 to DRYRUN=0

2. Save the file

3. Run the bot again: dotenvx run -- cargo run --release

4. Watch carefully: - You should see: Mode: LIVE EXECUTION - The bot will now actually place orders - Monitor it closely, especially at first!

### Or Run Live Mode Temporarily (Without Editing .env)

You can also override the setting for one run:

```
DRY_RUN=0 dotenvx run -- cargo run --release
```

This runs live mode once without changing your .env file.

## Step 5: Stop the Bot

To stop the bot:

- Press Ctrl + C (Windows/Linux) or Cmd + C (Mac)
- The bot will stop gracefully
- Any open positions will remain (you'll need to manage them manually)

## Common Command Examples

### Basic Dry Run (Safe Testing)

```
dotenvx run -- cargo run --release
```

### Dry Run with Verbose Logging

```
RUST_LOG=debug dotenvx run -- cargo run --release
```

### Live Trading (Real Money!)

```
DRY_RUN=0 dotenvx run -- cargo run --release
```

### Force Market Re-Discovery

If markets aren't matching, force refresh: FORCE_DISCOVERY=1 dotenvx run -- cargo run --release

### Live Trading with Custom Loss Limit

```
DRYRUN=0 CBMAXDAILYLOSS=1000.0 dotenvx run -- cargo run --release
```

## Understanding Bot Behavior

### When the Bot Finds Opportunities

The bot continuously monitors prices. When it finds an opportunity:

- Calculates profit - Checks if YES + NO < $1.00

### Expected Behavior

- Opportunities are rare - Don't expect trades every minute
- Opportunities are short-lived - They may disappear quickly
- Not all opportunities execute - Some may fill partially or not at all
- You need funds on both platforms - Bot can't trade without money!

## Monitoring Your Bot

### What to Monitor

- Connections - Make sure both WebSocket connections stay open

### Log Levels Explained

| Level | When Used | Example |
|---|---|---|
| error | Critical problems | Connection failures, auth errors |
| warn | Important warnings | Circuit breaker warnings, partial fills |
| info | Normal operations | Opportunities found, trades executed |
| debug | Detailed information | Every price update, order details |
| trace | Very detailed | Internal state, network packets |

Set RUSTLOG=info for normal use, or RUSTLOG=debug to see more details.

## Running the Bot 24/7

If you want the bot to run continuously:

### Option 1: Keep Terminal Open

- Just leave Terminal/PowerShell open
- Bot will run until you close it or it crashes

### Option 2: Use Screen (Linux/Mac)

```bash screen -S arbitrage-bot dotenvx run -- cargo run --release

# Press Ctrl+A then D to detach

# Reattach later with: screen -r arbitrage-bot
```

### Option 3: Use a Process Manager (Advanced)

- Windows: Run as a service or use Task Scheduler
- Linux: Use systemd or supervisor
- Mac: Use launchd

## What's Next?

Your bot should now be running! If you encounter any problems, check:

Troubleshooting Guide →

Or review the other guides: - Getting Started - Installation - Credentials - Configuration

Happy trading! ■

Chapter 6

# Troubleshooting Guide

---

## Troubleshooting Guide

Having problems? This guide covers common issues and how to fix them.

### Installation Problems

#### "cargo: command not found" or "rustc: command not found"

Problem: Rust isn't installed or not in your PATH.

Solution: 1. Windows: - Restart your computer (important!) - Open a NEW PowerShell window (not the old one) - Run: rustc --version - If still not working, try: $env:Path += ";$env:USERPROFILE\.cargo\bin" 2. Mac/Linux: - Run: source $HOME/.cargo/env - Or restart Terminal - Run: rustc --version

3. If still not working: - Reinstall Rust from https://rustup.rs - Make sure to restart your computer/terminal after installation

#### "dotenvx: command not found"

Problem: dotenvx isn't installed or not in PATH.

Solution: 1. Close and reopen Terminal/PowerShell completely

2. Reinstall dotenvx: - Windows: iwr https://dotenvx.sh/install.ps1 -useb | iex - Mac/Linux: curl -fsSL https://dotenvx.sh/install.sh | sh

3. Verify installation: dotenvx --version

4. If still not working: - Check your PATH: echo $PATH (Mac/Linux) or $env:Path (Windows) - Make sure ~/.cargo/bin (or equivalent) is in your PATH

### Build Errors: "could not compile..."

Problem: Code won't compile.

Solution: 1. Update Rust: rustup update

2. Clean and rebuild: cargo clean cargo build --release

3. Check your Rust version: rustc --version - Should be 1.75.0 or higher - If not, update: rustup update stable

4. Network issues: - Make sure you have internet connection - Try again later (might be temporary network issue) - Check firewall isn't blocking cargo

## Credential Problems

### "KALSHI_API_KEY_ID not set"

Problem: Can't find your Kalshi API key.

Solution: 1. Check your .env file exists: ls -la .env # Mac/Linux Test-Path .env # Windows

2. Check the variable name is correct: - Should be: KALSHIAPIKEYID=yourkey_here - No spaces around = - No quotes (unless value has spaces)

3. Make sure you're running with dotenvx: dotenvx run -- cargo run --release Not just cargo run --release!

### "Failed to read private key from..."

Problem: Can't find or read the Kalshi private key file.

Solution: 1. Check the file exists: ls -la kalshiprivatekey.pem # Mac/Linux dir kalshiprivatekey.pem # Windows

2. Check the path in .env: - If file is in same folder as .env, use: KALSHIPRIVATEKEYPATH=kalshiprivatekey.pem - Or use full path: KALSHIPRIVATEKEYPATH=C:/full/path/to/file.pem

3. Check file permissions (Mac/Linux): chmod 600 kalshiprivatekey.pem

4. Check file format: - File should be a .pem file from Kalshi - Open it in text editor - should start with -----BEGIN RSA PRIVATE KEY----- - Make sure it's not corrupted

### "POLY_PRIVATE_KEY not set" or "POLY_FUNDER not set"

Problem: Polymarket credentials missing or wrong.

Solution: 1. Check your .env file: - Make sure both POLYPRIVATEKEY and POLY_FUNDER are set - Values should start with 0x - Private key should be 66 characters (0x + 64 hex chars) - Wallet address should be 42 characters (0x + 40 hex chars)

2. Verify your wallet: - Check your MetaMask (or other wallet) - Make sure you copied the private key correctly - Make sure you copied the wallet address correctly

3. Common mistakes: - Missing 0x prefix - Extra spaces in the value - Quotes around the value (don't use quotes)

## Runtime Problems

### "Mode: DRY RUN" but I want live trading

Problem: Bot is running in test mode.

Solution: 1. Check your .env file: - Should have: DRYRUN=0 for live trading - DRYRUN=1 means test mode

2. Or override for one run: DRY_RUN=0 dotenvx run -- cargo run --release

### Bot connects but finds no market pairs

Problem: No markets matched between platforms.

Solution: 1. Force market re-discovery: FORCE_DISCOVERY=1 dotenvx run -- cargo run --release

2. Check which leagues are enabled: - Look at ENABLED_LEAGUES in src/config.rs - Empty array [] means all leagues - You can modify this to only specific leagues

3. Wait for markets to be available: - Markets only appear when there are upcoming games - Check both platforms manually to see if markets exist

4. Check your accounts have access: - Make sure your Kalshi account can access the markets - Make sure your Polymarket account is properly set up

## Bot finds opportunities but doesn't execute trades

Problem: Opportunities detected but no orders placed.

Solution: 1. Check if you're in DRYRUN mode: - If DRYRUN=1, bot won't place real orders - Change to DRY_RUN=0 for live trading

2. Check circuit breaker: - Look for circuit breaker messages in logs - If circuit breaker tripped, bot won't trade - Wait for cooldown period or reset manually

3. Check you have funds: - Make sure you have USDC on Polymarket - Make sure you have cash on Kalshi - Bot needs money to place orders!

4. Check position limits: - CBMAXPOSITIONPERMARKET might be too low - CBMAXTOTAL_POSITION might be too low - Increase these if needed

## WebSocket connection errors

Problem: Can't connect to Kalshi or Polymarket.

Solution: 1. Check internet connection: - Make sure you're online - Try pinging: ping google.com

2. Check firewall: - Windows Firewall might be blocking connections - Add exception for your terminal/cargo

3. Check if services are down: - Visit https://kalshi.com - does it work? - Visit https://polymarket.com - does it work? - Both platforms might be temporarily down

4. Wait and retry: - Sometimes temporary network issues - Wait a few minutes and try again

5. Check API key permissions: - Make sure your Kalshi API key has trading permissions - Make sure it hasn't been revoked

## "Circuit breaker tripped" errors

Problem: Bot stops trading due to circuit breaker.

Solution: 1. Check what triggered it: - Look at the error message - Common reasons: - Too many consecutive errors - Daily loss limit exceeded - Position limits exceeded

2. Wait for cooldown: - Bot will automatically retry after cooldown period - Default is 300 seconds (5 minutes)

3. Adjust limits if needed: - If limits are too strict, increase them in .env: - CBMAXDAILYLOSS=1000.0 (increase loss limit) - CBMAXPOSITIONPER_MARKET=100000 (increase position size)

4. Check for errors: - Look at recent error messages - Fix underlying issues before restarting

### Orders not filling

Problem: Orders placed but not executing.

Solution: 1. Opportunities disappear quickly: - This is normal - prices change fast - Bot tries to execute quickly but can't guarantee fills

2. Check order status: - Log in to Kalshi and Polymarket - Check if orders are pending - Cancel stale orders if needed

3. Check your balance: - Make sure you have enough funds - Orders need collateral

4. Check fees: - Kalshi has trading fees - Make sure profit is enough to cover fees

## Performance Issues

### Bot is slow or uses too much CPU

Solution: 1. Use release build (not debug): cargo build --release Always use --release for production!

2. Reduce logging: - Set RUSTLOG=info or RUSTLOG=warn - RUST_LOG=debug or trace is very verbose

3. Reduce price logging: - Set PRICE_LOGGING=0 - Price logging is very resource-intensive

### Bot uses too much memory

Solution: 1. Limit market discovery: - Set ENABLED_LEAGUES to specific leagues only - Fewer markets = less memory

2. Regular restarts: - Restart bot daily or weekly - Clears any memory leaks

## Still Having Problems?

If nothing here helps:

- Check the logs:
- Run with RUST_LOG=debug to see more details
- Look for [ERROR] messages
- Read the error messages carefully

2. Test components individually: - Test Kalshi API: Try accessing their website - Test Polymarket: Check your wallet connection - Test Rust: rustc --version should work

3. Check the code: - Make sure you're using the latest version - Pull latest changes: git pull - Rebuild: cargo clean && cargo build --release

4. Ask for help: - Create an issue on GitHub - Include: - Error messages - Your configuration (without credentials!) - What you were trying to do - System information (OS, Rust version)

## Quick Reference

### Common Commands

```bash
```

# Build the bot

cargo build --release

# Run in dry run mode

dotenvx run -- cargo run --release

## Run in live mode

DRY_RUN=0 dotenvx run -- cargo run --release

## Force market refresh

FORCE_DISCOVERY=1 dotenvx run -- cargo run --release

## Verbose logging

RUST_LOG=debug dotenvx run -- cargo run --release

## Clean build

cargo clean && cargo build --release ```

### Check Your Setup

```bash

## Check Rust

rustc --version

## Check dotenvx

dotenvx --version
```

## Check .env file exists

ls -la .env # Mac/Linux Test-Path .env # Windows

## Check credentials are set (without showing values)

Good luck! If you're still stuck, check the other guides or ask for help.