

# Setting Up Solana Development Environment and Deploying a "Hello World" Program

By the end of this practice, you will:

- 1. Set up the necessary tools for Solana development.
- 2. Write a minimal "Hello World" program in Rust for Solana.
- 3. Deploy the program to a local validator.

#### **Step 1: Environment Setup (15 minutes)**

#### **Instructions**

- 1. Install the Solana CLI:
  - o Follow the official installation guide.
  - Verify your installation by running:

```
solana --version
```

- 2. Install Rust and Cargo:
  - Visit <a href="https://rustup.rs/">https://rustup.rs/</a> to install Rust.
  - o Confirm your installation with:

```
rustc --version cargo --version
```

- 3. Set up the Solana local validator:
  - Start a local validator with:

```
solana-test-validator
```

 $\circ\,$  Open a new terminal and verify the validator is running:

```
solana cluster-version
```

### Challenge 1

- Identify which folder stores the ledger files for the local validator.
- Clear the ledger and restart the validator if it's corrupted.

#### **Bonus Challenge**

• Modify the starting token balance of your validator by adjusting the configuration.

### Step 2: Writing the "Hello World" Program (10 minutes)

### **Instructions**

1. Create a new Solana program project:

```
cargo new --lib hello_world
cd hello_world
```

2. Add Solana Program Crate:

In Cargo.toml, add the following dependencies:

```
[dependencies]
solana-program = "1.16.0"
```

- 3. Write your "Hello World" program:
  - Replace the content of src/lib.rs with:

```
use solana_program::{
    account_info::AccountInfo,
    entrypoint,
    entrypoint::ProgramResult,
    msg,
    pubkey::Pubkey,
};

entrypoint!(process_instruction);

fn process_instruction(
    _program_id: &Pubkey,
    _accounts: &[AccountInfo],
    _instruction_data: &[u8],
) -> ProgramResult {
    msg!("Hello, Solana World!");
    Ok(())
}
```

4. Build the program:

```
cargo build-bpf --release
```

# **Challenge 2**

• Modify the program to accept and display input data (e.g., a custom message passed during the call).

# **Step 3: Deploying the Program (15 minutes)**

#### Instructions

- 1. Generate a keypair for your program in a json format.
- 2. Deploy your program to the local validator.
- 3. Verify the deployment by checking the program id, and looking for the "Program is successfully deployed" message.

### Challenge 3

• Call your program using the Solana CLI and verify its output in the validator logs.

#### **Bonus Challenge**

• Write a client script in Python or JavaScript (using Solana Web3.js) to send a transaction that triggers your program.

#### Resources

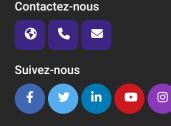
- 1. Solana CLI Documentation: Solana CLI Guide
- 2. Rust Programming Language: Rust Official Documentation
- 3. Solana Program Development: Solana Labs Cookbook
- 4. Solana Logs: Logging and Debugging

Activité précédente

Chapitre 07 : Introduction à Solana

Aller à...

Activité suivante
Chapitre 08 : Les Tokens



🛛 😉 Contacter l'assistance du site

Connecté sous le nom « <u>Ibrar Hamouda</u> » (<u>Déconnexion</u>)