Solana Command Line Interface (CLI)

Wallet Manager Cheatsheet

> Hi, i 'm Kenneth and i will guide you through this cheatsheet for an easy way to access the Solana Technology. I designed it to be procedural with a set of personal interventions providing metaphors for real-life cases. However, it can also be viewed by topic if needed.

Read Time: 9 m

Table of contents

- JavaScript

§ Token

1. Setup
2. Create
3. Account
    - Balance
4. Mint
    - Limit Supply
5. Send

---

§ Prerequisites

> Every nice work needs a nice setup.

0. System

🪟 ***Windows*** *Standard Procedure*
We will use the ***Linux*** 🐧 subsystem for ***Windows*** 🪟, meanwhile ***macOS*** 🍎 can skip here and native ***Linux*** 🐧 here.

Install

```
wsl --install
```

Run subsystem

```
ubuntu
```

Dependencies

```
# Install Curl
sudo apt-get install curl
```

Update

```
sudo apt-get update && sudo apt-get upgrade && sudo apt-get install -y pkg-config build-essential libudev-dev libssl-dev
```

Restart Ubuntu Terminal after completing the system setup.

1. Install NVM

> **NVM (Node Version Manager)** is a tool that allows you to manage multiple versions of **Node.js** a server-side JavaScript runtime environment that enables the execution of JavaScript code outside the browser, used for building scalable and high-performance server-side applications.

Install

```
# Install NVM
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh |
bash
```

Version

```
# Test if nvm exists - this will return "nvm" and not a version number if
working correctly!
command -v nvm
```

Update

```
# Install the latest version of Node.js
nvm install --lts
```

2. Rust 🦀

> The **Rust** programming language used to write smart contracts on Solana, known for its safety, speed, and memory management without garbage collection, making it ideal for blockchain development.

Install

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

Version

```
rustup --version
```

-

```
rustc --version
```

-

```
cargo --version
```

3. Solana

> The most powerful Blockchain technology according my modest opinion

Install

```
sh -c "$(curl -sSfL https://release.solana.com/v1.9.5/install)"
```

Version

```
solana --version
```

Local Host Setup

```
solana config set --url localhost
solana config get
```

Result

```
Config File: /Users/your-name/.config/solana/cli/config.yml
RPC URL: http://localhost:8899
WebSocket URL: ws://localhost:8900/ (computed)
Keypair Path: /Users/your-name/.config/solana/id.json
Commitment: confirmed
```

> Later we will setup the `Keypair Path` which now is set to default `/Users/your-name/.config/solana/id.json`

4. Mocha ☕

> **Mocha** is a JavaScript testing framework that facilitates running and organizing unit and integration tests for Node.js applications.

Install

```
npm install -g mocha
```

Version

```
mocha --version
```

5. Anchor ⚓

> **Anchor** is a development framework for the Solana blockchain that simplifies the creation, testing, and deployment of smart contracts.

Yarn - JavaScript Dependency

Install JS dependency 🍎

```
brew install yarn
```

or

Install JS dependency 🐧

```
npm install --global yarn
```

then

Install

```
cargo install --git https://github.com/project-serum/anchor anchor-cli --
locked
```

Version

```
anchor --version
```

---

§ Wallet

> Creating a wallet is the first step as a base to interact with the network.

1. Setup

Folder

```
cd path/to/wallet/folder
mkdir mywallet
```

Node package manager

```
npm init -y
```

Install package needed to interact with the blockchain network with JavaScript

```
npm install --save @solana/web3.js
```

or

Generate a setup file like i did to run like

```
node setup.js
```

You can find the code here setup.js file.

2. Generate Key Pair

> In a nutshell u should consider the Key pair as another way to say Wallet.

For this we will use the Solana CLI which creates a formatted key for you, giving the you the possibility to add a passphrase to lock secure you credentials.

> If you don't need it just press `Enter` when asked, at your own risk.

```
solana-keygen new --outfile /path/to/your/new-keypair.json --force
```

Specifying the path for the keygen will allow you to have multiple wallets linked to every project folder, otherwise it will save the key in `/home/user/.config/solana/id.json` as default.

This is concept can be applied to the Local Host Setup that we mentioned earlier which you can choose to leave it as a default or set it to a personalized wallet like so

```
solana config set --keypair /path/to/other/location/wallet.json
```

3. Derive Public Key

> This will return your public key derived from the private to be extra sure.

From path

```
solana-keygen pubkey /path/to/your/new-keypair.json
```

From default `/home/user/.config/solana/id.json`

```
solana address
```

4. Wallet.js

You can find the code here with a comprehensive set of comments for better understanding of dynamics.

To run functionality

```
node path/to/wallet.js
```

5. Balance

> Keeping track of balance as they thought me in Banking, Finance and Financial Markets university is the art key to success.

From path

```
solana balance /path/to/your/new-keypair.json --url devnet
```

*From default `/home/user/.config/solana/id.json

```
solana balance --url devnet
```

6. Network

You can check the status of your wallet at https://explorer.solana.com/ by adding your public key to the search bar.

The Solana Network splits in four accessible branches:

- **Localnet**
  Ambiente di rete locale per testare applicazioni Solana senza connettersi alla rete pubblica.

```
solana balance --url http://localhost:8899

solana balance --url localnet
```

- **Devnet**
  Rete di sviluppo pubblica di Solana per testare applicazioni in un ambiente simile a quello reale ma con fondi non reali.

```
solana balance --url https://api.devnet.solana.com

solana balance --url localnet
```

- **Testnet**
  Rete di prova di Solana che simula condizioni di Mainnet con token non reali per testare nuove funzionalità e aggiornamenti.

```
solana balance --url https://api.testnet.solana.com

solana balance --url testnet
```

- **Mainnet**
  La rete principale di Solana dove avvengono le transazioni reali e le operazioni con token veri.

```
solana balance --url https://api.mainnet-beta.solana.com

solana balance --url mainnet
```

> You can find a way to set the kind of net through JavaScript from line `90` in the `wallet.js` file here.

---

§ Test Validator

> The way to simulate your machine as a "mining" node, or more accurately, a Solana Validator Node.

The CLI functionality will use the credentials specified in the config file set by:

`solana config set --keypair /path/to/other/location/wallet.json`

or, if not specified, it will default to:

`/home/user/.config/solana/id.json`

When you run the command, it will create a folder with all the necessary tools and data in the directory from which you execute:

```
solana-test-validator
```

---

§ Airdrop

> No, we're not dropping bombs... Or, are we?

Airdrop SOL

```
# pubkey = 09j9hete82ec92h981uev433y5mb67b45q0
solana airdrop 2 09j9hete82ec92h981uev433y5mb67b45q0 --url devnet
```

JavaScript

You can find the code here with a comprehensive set of comments for better understanding of dynamics.

> I personally suggest to read the README.md file.

---

§ Token

> Yes, this may be the bomb type. Make sure to create something big 🚀

1. Setup

After you got set up with Rust and the Solana CLI there is one other set up to be done and it is the Token CLI. This will allow us to manage our tokens.

Install

```
cargo install spl-token-cli
```

Version

```
spl-token --version
```

2. Create

> By creating a token we mean "create the entity", designing it with its structural features, much like every bill is designed before release to the public in material, drawings and security elements.

Create

```
spl-token create-token --url devnet
```

- Result

```
Creating token wo8qhct874htovn824hoq87rytovn7tmhcqo847y

Signature:28c37rb872r3c0b378y...

# Token Address ID: wo8qhct874htovn824hoq87rytovn7tmhcqo847y
# Operation ID: 28c37rb872r3c0b378y...
```

The Token **Address** ID serves as a unique identifier in the blockchain world.

> The token **Address** is similar to what we call the **Codice Fiscale** in Italy, the **Steueridentifikationsnummer** in Germany, or the **Social Security Number** in the US.

> While you can choose any name for yourself, these identifiers are crucial for government entities to track and manage individuals. Much like we need to do with our tokens.

3. Account

In order to store tokens you need an account set for each of them. Every account has its own token type, so that's where the *token address* comes handy.

> This means that every wallet can have multiple accounts storing different types of tokens like pockets in your real wallet for every type of banknotes.

Create an account in your wallet

```
# We paste the Token Address for reference
spl-token create-account wo8qhct874htovn824hoq87rytovn7tmhcqo847y --url
devnet
```

- Result

```
Creating account eufq2he8u37rcyb987yv02984m1p270erx9b8237yx

Signature: 19rg8r0cn328ynf...
```

```
# Account Address ID: eufq2he8u37rcyb987yv02984m1p270erx9b8237yx
# Operation ID: 19rg8r0cn328ynf...
```

> Account **Address**? Easy, as said for the token Address same thing for the Account.

- ## Balance

Retrieve Balance of Token in wallet account

```
# We paste the Token Address again for reference

spl-token balance wo8qhct874htovn824hoq87rytovn7tmhcqo847y --url devnet
```

4. Mint

> Minting a token means to create copy of it much like the governments prints money creating inflation and poverty by reducing the value of the printed asset, so be careful.

Add 1000 token to previously created account in wallet

```
# We paste the Token Address again for reference

spl-token mint wo8qhct874htovn824hoq87rytovn7tmhcqo847y 1000 --url devnet
```

Only the creator of the token has minting privileges

- ## Supply
  Retrieve Circulating Supply of Minted Token in the Network

```
# We paste the Token Address again for reference

spl-token supply wo8qhct874htovn824hoq87rytovn7tmhcqo847y --url devnet
```

- ## Inflationary Limit
  The spl-token CLI let us remove all powers to mint new tokens permanently to secure their value against inflation.

  Impose Limit

  ```

  ```

We paste the Token Address again for reference

spl-token authorize wo8qhct874htovn824hoq87rytovn7tmhcqo847y mint --disable --url devent
```

- **Burn Tokens**
  The ability to remove your personal tokens from circulation.

*Remove 1000 Personal Supply From Our Account*

```
# We paste the Account Address for reference

spl-token burn eufq2he8u37rcyb987yv02984m1p270erx9b8237yx 991 --url devnet
```

5. Send

Giving that the receiving end, **on the same network**, has already a wallet with the dedicated account; we can send but we don't use the account address but the *Wallet Public Address*.

> which is our **Public Key**!

Send to Receiving Wallet Using its Public Key

```
# We paste first the Token Address and then the Public Key of receiver

spl-token transfer wo8qhct874htovn824hoq87rytovn7tmhcqo847y 9
udhuh89237o93nn2978hf9feqoh8uhoeueou --url devnet
```

> Unfunded accounts of a wallet may give some warning messages so follow instruction if you encounter any of those.

Author: Kenneth Boldrini
```