Madara Quick Start Guide

This is the quickest and easiest way that I've found to get a Madara node up and running and then deploy contracts locally.

Required Software

Please make sure that you have the following installed

- rust
- asdf
- starkli 0.1.20 this page also shows you how to install via `asdf` to manage different versions
- scarb also recommended install with `asdf`
- node v.20.5.1 (can manage with either asdf or nvm)

Repos

You'll clone these 2 repos

- Madara
- Karnot/Madara

Clone the Madara Repo

```
git clone git@github.com:keep-starknet-strange/madara.git
```

Starting Madara

Once cloned, from inside the `madara` folder of this newly cloned repo run the following command.
Replace the `/User/desmo/madara-state` with the location where you want to store your Madara state: `/home/user/directory`

> Note
>
> These commands are different than the commands given in the Madara repo and allow us to save the node's state locally.

```
cargo run --release -- setup --from-local ./configs/ --chain=dev --base-path=/Users/desmo/madara-state
```

```
cargo run --release -- --alice --base-path /Users/desmo/madara-state --rpc-cors=all --chain=dev
```

This starts the chain and is pointing at the location of where we stored the state you should replace the again with what you used in the previous command.

If successful you should get an output similar to this:

```
    Finished release [optimized] target(s) in 1.80s
     Running `target/release/madara --alice --base-path /Users/desmo/madara-state --rpc-cors=all --chain=dev`
2024-03-06 17:54:44 Madara initialized w/o DA layer
2024-03-06 17:54:44 Madara initialized w/o settlement layer
2024-03-06 17:54:44 Madara Node
2024-03-06 17:54:44 ✌  version 0.7.0-5e68494fe9d
```

```
2024-03-06 17:54:44 ❤  by Abdelhamid Bakhta <@keep-starknet-strange>:Substrate DevHub
<https://github.com/substrate-developer-hub>, 2017-2024
2024-03-06 17:54:44 📋 Chain specification: Development
2024-03-06 17:54:44 🏷  Node name: Alice
2024-03-06 17:54:44 👤 Role: AUTHORITY
2024-03-06 17:54:44 💾 Database: RocksDb at /Users/desmo/madara-state/chains/dev/db/full
2024-03-06 17:54:44 🧪 Using the following development accounts:
2024-03-06 17:54:44 🧪 NO VALIDATE with address: 0x1 and no pk
2024-03-06 17:54:44 🧪 ARGENT with address: 0x2 and pk:
0xc1cf1490de1352865301bb8705143f3ef938f97fdf892f1090dcb5ac7bcd1d
2024-03-06 17:54:44 🧪 OZ with address: 0x3 and pk:
0xc1cf1490de1352865301bb8705143f3ef938f97fdf892f1090dcb5ac7bcd1d
2024-03-06 17:54:44 🧪 CAIRO 1 with address: 0x4 and no pk
2024-03-06 17:54:44 Using default protocol ID "sup" because none is configured in the chain specs
2024-03-06 17:54:44 🏷  Local node identity is:
12D3KooWLngNFrmtGSXsdWV8RG1aKbNs7dHrt8NfBmt9TCtHc4CY
```

Creating our Contract

Now that we've successfully launched our node we'll build a contract, it can be any contract but let's start with a simple contract from the Starknet by Example a counter-contract in which we'll make some simple modifications.

Scarb Commands

Let's start by creating our project using scarb

```
scarb new counter_contract
```

then go into your `counter_contract` and open your newly created project with your favorite editor.

Modifying `scarb.toml`

We need to add a couple of lines to our `toml` file to make it compatible with our Madara node.

```
[package]
name = "counter_contract"
version = "0.1.0"

# See more keys and their definitions at
https://docs.swmansion.com/scarb/docs/reference/manifest.html

[dependencies]
starknet = ">=2.1.0"

[[target.starknet-contract]]
casm = true
```

Modifying `lib.rs`

If you're running your own `hello_world` (or different) contract you don't have to do this but if you're following along we need to modify the counter contract by removing and adding small parts of code.

```
#[starknet::interface]
// remove pub modifier
trait ISimpleCounter<TContractState> {
    fn get_current_count(self: @TContractState) -> u128;
    fn increment(ref self: TContractState);
```

```
    fn decrement(ref self: TContractState);
}

#[starknet::contract]
// remove pub modifier
mod SimpleCounter {
    #[storage]
    struct Storage {
        // Counter variable
        counter: u128,
    }

    #[constructor]
    // remove the `init_valule` parameter
    fn constructor(ref self: ContractState) {
        // Store initial value
        self.counter.write(1); // add `1` here
    }

        // this macro is important to make sure tha we can access
        // these functions externally
    #[external(v0)]
    impl SimpleCounter of super::ISimpleCounter<ContractState> {
        fn get_current_count(self: @ContractState) -> u128 {
            return self.counter.read();
        }

        fn increment(ref self: ContractState) {
            // Store counter value + 1
            let counter = self.counter.read() + 1;
            self.counter.write(counter);
        }
        fn decrement(ref self: ContractState) {
            // Store counter value - 1
            let counter = self.counter.read() - 1;
            self.counter.write(counter);
        }
    }
}
```

In the comments in the code you can see the modifications we've made, but to make it explicit we've:

1. Remove the `pub` modifier in the `trait ISimpleCounter` and `mod SimpleCounter`
2. Remove the `init_value` parameter from the `fn constructor()` and in `self.counter.write()`
3. Change the macro above the `impl SimpleCounter` to `#[external(v0)]`

Building our `lib.rs`

Once the contract is ready you can simply build the contract using

```
scarb build
```

If all goes well you should have 3 new files in your newly generated `target/dev` folder

```
counter_contract.starknet_artifacts.json
counter_contract_SimpleCounter.sierra.json #ABI file
```

```
counter_contract_SimpleCounter.casm.json
```

With these contracts exported we can move on to the next steps which are declaring and deploying and for this we need the *Karnot: Madara get Started* repo.

Setting up Karnot/madara-get-started

We start by cloning the repo, you probably want to do this outside of your `madara` node directory

```
git clone git@github.com:karnotxyz/madara-get-started.git
```

and

```
npm i
```

> Note
>
> You should to be running Node v20.5.1 - so make sure you're running that version before running the Node commands by using either `asdf` or `nvm`

Cleaning the Contracts Folder

In the `madara-get-started/contracts` folder you have some pre-built contracts

```
➜  madara-get-started git:(main) ls ./contracts/*
./contracts/ERC20.json
./contracts/OpenZeppelinAccountCairoOne.sierra.json
./contracts/OpenZeppelinAccountCairoOne.casm.json
```

remove the contracts from your contracts folder

```
rm ./contracts/*
```

Now you need to go back to your `counter_contract` project and get get the contracts you built in the `target/dev` folder and put these contracts:

```
counter_contract.starknet_artifacts.json
counter_contract_SimpleCounter.sierra.json
counter_contract_SimpleCounter.casm.json
```

into the `madara-get-started/contracts` folder

Making Changes to Madara

Before we try and declare and deploy let's make sure we have our environment variables set by running on your terminal:

```
export STARKNET_RPC="http://localhost:9944/"
```

You can also set-up a more permanent solution but for now this works.
Confirm that all is correct by running

```
echo $STARKNET_RPC
```

You should get the `http://localhost:9944/` back.

Changing the ChainId

Now that we've confirmed that we're set the correct `STARNKET_RPC` we need to go into our `madera` directory and open up the `pallets.rs` file which can be found at `madara/crates/runtime/src/pallets.rs`

Here we're going to change the `SN_GOERLI_CHAIN_ID` to `MADARA_CHAIN_ID` you can search, but I found them on line `13` and line `164`

```
// line 13
pub use mp_chain_id::MADARA_CHAIN_ID;

...
// line 164
pub const ChainId: Felt252Wrapper = MADARA_CHAIN_ID;
```

Save this file and close it now we're ready to re-launch the node

Go ahead and stop your previous Madara node using ctrl+C and from inside of your `madara` folder run the same command we ran earlier to start the node:

```
cargo run --release -- --alice --base-path /Users/desmo/madara-state --rpc-cors=all --chain=dev
```

It should pick up from where your node left off but now with the correct chain ID.

Declaring and Deploying the Contracts

Now we need to go back to your `madara-get-started` folder and and run the scripts with node. Remember to be on the correct version of `v.20.5.1` and run:

```
node scripts/declare.js ./contracts/counter_contract_SimpleCounter.sierra.json
./contracts/counter_contract_SimpleCounter.casm.json
```

the `.seirra.json` and `.casm.json` files should be the name of your contract name, if you're following along using the `counter_contract` example it should be the same as above and you should be able to run the scripts just the same, but if for whatever reason they're in a different location you can use the command with the correct locations

```
node scripts/declare.js <path_to_sierra> <path_to_casm>
```

If that works you should see something like this

```
This is the declare result -  {
  transaction_hash: '0x4d9495073404c0ed2fd42bec07fc5c91d32005fedc20702d6f90612a69d78ce',
  class_hash: '0x5c9b0add4b036cef0eef4fa593f29da8554c162186be99c563c3ef10d31628d'
}
```

Once that is done we should be able to deploy the contract with

```
node scripts/deploy.js ./contracts/counter_contract_SimpleCounter.sierra.json
```

If successful we should get something like this:

```
This is the deploy result -  {
  transaction_hash: '0x46803b2cb6d9b20902a04f6f23db165846e90a6f6096a5435198773c78912a2',
  contract_address: [
    '0x18a91f3352b686058894699654e470dfa65f20da5bc08317fb1a6aa3faaee78'
  ]
}
```

Contract Interaction

Now we should be able to interact with our contract using `starkli`

```
starkli call 0x18a91f3352b686058894699654e470dfa65f20da5bc08317fb1a6aa3faaee78 get_current_count
```

and that should output

```
[
    "0x0000000000000000000000000000000000000000000000000000000000000001"
]
```

and this shows our "counter" working as we have `1` as our output which is the `1` we passed through on the initial value

```
fn constructor(ref self: ContractState) {
        // Store initial value
        self.counter.write(1); // add `1` here
    }
```

Now all we have to do is deploy some more sophisticated contracts