**Write and deploy a smart contract in Rust**

Requirements and setup

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

Near CLI (npm install -g near-cli)

Near Testnet account (<https://wallet.testnet.near.org/>)

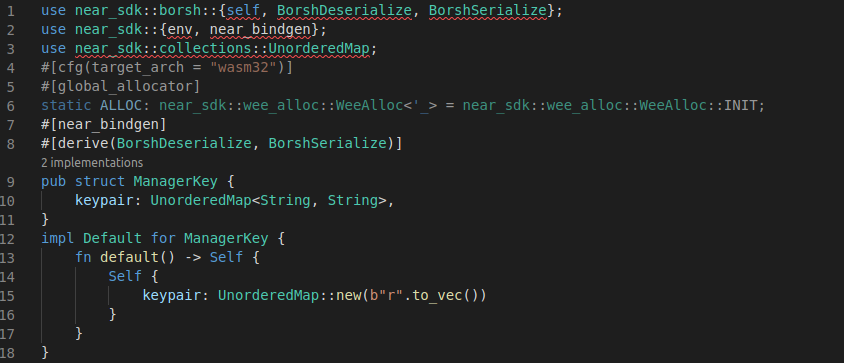
WASM (rustup target add wasm32-unknown-unknown)

Let’s create a directory called nearprotocol, then change into that directory and run the following command in the terminal: cargo init --lib

**Write the contract**

We will create a simple Create, Read, Update, Delete (CRUD) backend in Rust that utilizes the on-chain storage offered by NEAR

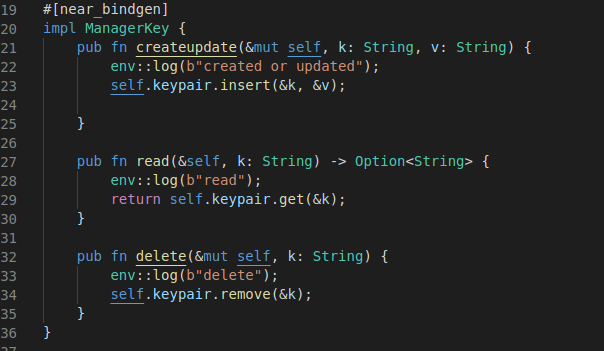
We can start by removing all the existing code in lib.rs and pasting in the following code snippet:



**Main struct and default implementation**

we will follow a pattern using one structure (struct) and an implementation (impl) associated with it

**core logic**

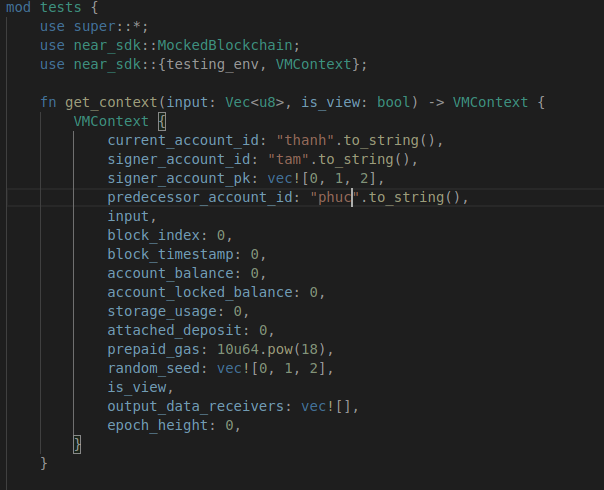


The first method createupdate is used to create or update particular pair

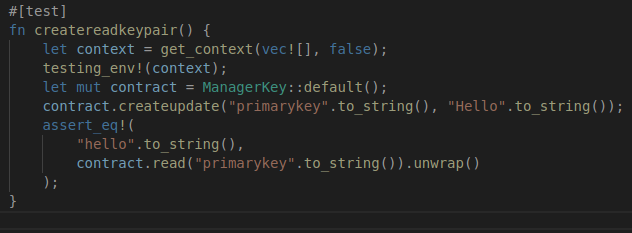
The next two methods are read and delete

**Testing the contract**

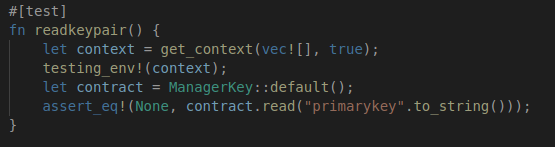
This is where we set up our testing environment with various parameters and a mocked blockchain. Read more about the Virtual Machine context and testing environment



We test createupdate method



The next we test read method



Now, we test our code : *cargo test -- --nocapture*

**Compiling the contract**

We will compile it into WebAssembly for deployment on NEAR

*env 'RUSTFLAGS=-C link-arg=-s'*

*cargo build --target wasm32-unknown-unknown –release*

**Deploying the contract**

We have to login into our account : near login

We will now have your Access Key stored locally in a hidden directory called

*cd ~/.near-credentials*

We create a new account belonging to our master account.

*near create-account abc.thaoluu16.testnet --masterAccount thaoluu16.testnet --initialBalance 10*

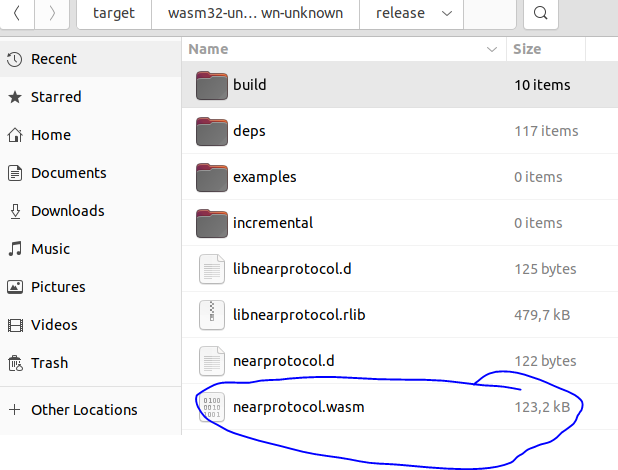
Now we can deploy our Rust contract to NEAR. Run the following command in the terminal:

*near deploy --wasmFile target/wasm32-unknown-unknown/release/nearprotocol.wasm --accountId abc.thaoluu16.testnet*

If there is an error during deployment, we need to check the size of the file WASM > 200k

we run the following command to reduce the size of wasm file then do it again

*RUSTFLAGS='-C link-arg=-s' cargo build --target wasm32-unknown-unknown –release*



**Interacting with the contract**

We'll create a key-value pair and then read it

*near call abc.thaoluu16.testnet createupdate '{"k": "first\_key", "v" : "1"}' --accountId abc.thaoluu16.testnet*

We will read the value of the first key:

*near view abc.thaoluu16.testnet read '{"k": "first\_key"}' --accountId abc.thaoluu16.testnet*

We will delete the key:

*near call abc.thaoluu16.testnet delete '{"k": "first\_key"}' --accountId abc.thaoluu16.testnet*