Visual Studio Code setup

cc32d9 / Europechain

Installation steps

- OpenSSH Client
- Key-based SSH authentication
- Install Visual Studio Code, with SSH extension

OpenSSH client on Win10

- Start → Apps&Features → Optional features
- If OpenSSH Client is not installed, “Add a feature” → find and install OpenSSH Client
- Start → cmd

Create SSH key and upload to remote server

```
# this command will generate a private SSH key.
# It asks for file location (press ENTER)
# Then it asks for passphrase. You can leave it empty for training purpose
# In production environment, give it a solid passphrase and keep it in a safe place

ssh-keygen

# this created two files in .ssh directory in your home folder:
#  id_rsa       (your private key)
#  id_rsa.pub   (your public key)
# It is safe to share your public key, but you must keep the private key safely.

# this makes a directory on remote server
ssh std010@myserver mkdir .ssh

# this copies your public key to remote server
scp .ssh/id_rsa.pub std010@myserver:.ssh/authorized_keys

# try logging in to remote server. It should not ask you a password
# You can use Ctrl-D or “exit” to logout
ssh std010@myserver
```

Install VSCode

- Download and install VSCode: https://code.visualstudio.com/
- In VSCode, File → Preferences → Extensions, install:
  - **C/C++** by Microsoft
  - **Remote Development** by Microsoft
  - **Remote - SSH** by Microsoft

SSH connection in VSCode

- Remote Explorer icon appeared after SSH extension is installed (terminal with ><)
- Remote explorer → SSH Targets → **+**
- "Enter SSH Connection Command" →
  - **ssh std010@myserver**
- "Select SSH configuration file to update" →
  - **C:\Users\USER\.ssh\config**
- in SSH Targets window, select your server and click “connect” button

Get a copy of token contract

```
# in CMD window,

ssh std010@myserver

cd build
git clone https://github.com/EOSIO/eosio.contracts.git
cp -a eosio.contracts/contracts/eosio.token/ .
mv eosio.token/ mytoken
rm mytoken/CMakeLists.txt

# now you have a copy of eosio.token contract in “build/mytoken” directory
```

Set up VSCode workspace

- File → Open Folder → /home/USER/build/mytoken/
- File → Save workspace as... → /home/USER/build/mytoken/mytoken

Configurte Intellisense

- Open eosio.token.cpp in editor.
- the **#include** statement shows warnings. Click the yellow bulb → *Edit includePath settings*
  - C standard: c11
  - C++ standard: c++17

Configurte Intellisense (2)

- Include Path (assuming CDT 1.6.3 is installed):

```
${workspaceFolder}/**

/usr/opt/eosio.cdt/1.6.3/include/eosiolib/core

/usr/opt/eosio.cdt/1.6.3/include/eosiolib/contracts
```

VCScode is ready

- When navigating definitions, such as “contract” in *eosio.token.hpp*, the editor should display where the term or class is defined.
- Try editing (or adding spaces) the source files and saving.

Compile the contract

```
# in CMD window,

# probably you still have it open already
ssh std010@myserver

# this changes you to home directory
cd

# go to project directory and compile the sources
cd build/mytoken/
eosio-cpp -I include/ src/eosio.token.cpp

# this will show two new files: eosio.token.abi and eosio.token.wasm
ls -l
```

Uploading the contract

```
$ alias jcleos='cleos -v -u http://jungle2.cryptolions.io'

$ jcleos set contract training1111 ./ eosio.token.wasm eosio.token.abi
```

Testing the contract

```
# create a token with symbol DONKEY on our token contract
# with maximum supply of 100k and precision of 2 decimals
$ jcleos push action training1111 create '["training1111", "100000.00 DONKEY"]' -p
training1111@active

# issue 10 tokens to ourselves
$ jcleos push action training1111 issue '["training1111", "10.00 DONKEY", "training"]' -p
training1111@active

# check balance
$ jcleos get currency balance training1111 training1111 DONKEY
10.00 DONKEY

# send 10 tokens to another account
$ jcleos push action training1111 transfer '["training1111", "cc32dninexxx", "10.00
DONKEY", "training"]' -p training1111@active

# check balance
$ jcleos get currency balance training1111 training1111 DONKEY
0.00 DONKEY
$ jcleos get currency balance training1111 cc32dninexxx DONKEY
10.00 DONKEY
```

Token contract allows multiple currencies

```
$ jcleos push action training1111 create '["training1111", "1000000 MONKEY"]' -p
training1111@active

$ jcleos push action training1111 issue '["training1111", "10 MONKEY", "training"]' -p
training1111@active

# check balance
$ jcleos get currency balance training1111 training1111 MONKEY
10 MONKEY
```