Setting up Environment

You need an environment to run contracts. You can either run your node locally or connect to an existing network. For easy testing, the Malaga testnet is live. You can use the testnet to deploy and run your contracts.

To verify that the testnet is currently active, make sure the following URLs are working for you:

- https://rpc.malaga-420.cosmwasm.com/status
- https://faucet.malaga-420.cosmwasm.com/status

The testnet has two native tokens set up -Andalucía (uand) for becoming a validator andMálaga (umlg) for paying fees.

Available Block Explorers:

- Big Dipper Block Explorer:https://block-explorer.malaga-420.cosmwasm.com

You can use the block explorer to explore transactions, addresses, validators and contracts.

When interacting with the testnet, you can either usewasmd which is a Go client, or the Node REPL. The Node REPL is recommended for contract operations, since JSON manipulation is not intuitive with the Shell/Go client.

Setting up wasmd Go CLI

Let's configure thewasmd executable, point it to the testnet, create a wallet and ask for tokens from faucet:

First source the network configuration for the Malaga testnet in the shell:

source

< (curl -sSL https://raw.githubusercontent.com/CosmWasm/testnets/master/malaga-420/defaults.env) Set up wallet addresses:

add wallets for testing

wasmd keys add wallet

wasmd keys add wallet2 info Running the command above will add two encrypted private keys to the wasmd keyring and display their attributes as follows:

- name: wallet type: local address: wasm1evvnsrte3rdghy506vu4c87x0s5wx0hpppqdd6 pubkey: '{"@type":"/cosmos.crypto.secp256k1.PubKey","key":"A2aKoMPLbUnXkN2zJF/q4lIH/34ybelQSRTg3d9Js86T"}' mnemonic: ""

Important write this mnemonic phrase in a safe place. It is the only way to recover your account if you ever forget your password.

table shell potato spike paddle very asthma raise glare noodle vibrant chuckle indicate spell perfect craft step net radio yellow minor deal blur hybrid You need some tokens in your address to interact with the network. If you are using local node you can skip this step. Requesting tokens from faucet:

JSON

(jq -n --arg addr (wasmd keys show -a wallet)

'{"denom":"umlg","address":addr}')

&&

curl -X POST --header "Content-Type: application/json" --data " JSON " https://faucet.malaga-420.cosmwasm.com/credit JSON = (jq -n --arg addr (wasmd keys show -a wallet2)

'{"denom":"umlg","address":addr}')

&&

curl -X POST --header "Content-Type: application/json" --data " JSON " https://faucet.malaga-420.cosmwasm.com/credit

Export wasmd Parameters

wasmd client requires to be further configured in order to interact with different testnets. Each testnet has its own endpoints and system parameters.

An effective way to configure wasmd is to define the following environment variables, making use of the network configuration parameters we sourced earlier.

If you intend to use wasmd as your preferred client, we recommend you to set up these variables. Otherwise you will have to type in node, chain id and gas-price details with every command you execute.

bash

export

NODE

"--node RPC " export

TXFLAG

" {NODE} --chain-id {CHAIN_ID} --gas-prices 0.25 {FEE_DENOM} --gas auto --gas-adjustment 1.3"

zsh

export

NODE

(--node RPC) export

TXFLAG

(NODE --chain-id CHAIN_ID --gas-prices 0.25 FEE_DENOM --gas auto --gas-adjustment 1.3) If the commands above throws an error, it means you are utilizing a different type of shell. If there are no errors, try executing the following command:

wasmd query bank total NODE A response similar to the one below means that you can now interact with the node you have configured.

pagination: next_key: null total: "2" supply: - amount: "10006916235913" denom: uand - amount: "10000000000000" denom: umlg You can check that your credit request from the faucet has been successful by checking the balance of your wallet addresses by running the following commands:

wasmd query bank balances (wasmd keys show -a wallet)

NODE wasmd query bank balances (wasmd keys show -a wallet2)

NODE You can explore the details about various other wasmd commands by running

wasmd help

Setting up the CosmJS CLI client

Beyond the standard CLI tooling,CosmJS was developed as a flexible TypeScript library, which runs in Node.js as well as in modern browsers. Among other capabilities, the library supports smart contract queries and transactions. Along with this library, the@cosmjs/cli was developed to act as a super-charged Node console. It supports the keywordawait , does type checking for helpful error messages, and preloads many CosmJS utilities. If you are comfortable with the Node console, you will probably find CosmJS CLI easier and more powerful than the wasmd Go CLI tooling.

To initialize a CosmJS CLI session,node.js 12+ andnpx must be installed first.

Run the following command to start the Node REPL:

npx @cosmjs/cli@^0.28.1 --init https://raw.githubusercontent.com/InterWasm/cw-plus-helpers/main/base.ts --init https://raw.githubusercontent.com/InterWasm/cw-plus-helpers/main/cw20-base.ts With that, you should observe the initialization of an interactive session.

Now, let us create a new wallet account, display the address and check the account balance.

// Create or load account const

[address , client]

=

await

useOptions (malagaOptions) . setup ("password" , ".new.key") ; // Display the wallet address client . getAccount (address) ; { address :

'wasm1llrnvtvhe5up6erf7mv7uh35efea567f5gejjr' , pubkey :

null , accountNumber :

53 , sequence :

0 } // Check the wallet balance client . getBalance (address , "umlg") {

denom :

'umlg' ,

amount :

'100000000'

} Previous Installing Prerequisites Next Downloading and Compiling a Contract * Setting up wasmd Go CLI * Export wasmd Parameters * Setting up the CosmJS CLI client