Deploying Your First Solana Program

In this section, we'll build, deploy, and test a simple Solana program using the Anchor framework. By the end, you'll have deployed your first program to the Solana blockchain!

The purpose of this section is to familiarize you with the Solana Playground. We'll walk through a more detailed example in the PDA and CPI sections. For more details, refer to thePrograms on Solana page.

Create Anchor Project#

First, openhttps://beta.solpg.io in a new browser tab.

- Click the "Create a new project" button on the left-side panel.
- Enter a project name, select Anchor as the framework, then click the "Create"
- button.

New Project

You'll see a new project created with the program code in thesrc/lib.rs file.

lib.rs use anchor_lang :: prelude ::* ;

// This is your program's public key and it will update // automatically when you build the project. declare_id! ("11111111111111111111111111111111");

[program]

mod hello_anchor { use super ::* ; pub fn initialize (ctx : Context < Initialize

> , data : u64) -> Result <()> { ctx . accounts . new_account . data = data; msg! ("Changed data to: {}!" , data); // Message will show up in the tx logs Ok (()) } }

[derive(

Accounts)] pub struct Initialize <' info

> { // We must specify the space in order to initialize an account. // First 8 bytes are default account discriminator, // next 8 bytes come from NewAccount.data being type u64. // (u64 = 64 bits unsigned integer = 8 bytes)

[account(init, payer

= signer, space = 8 + 8)] pub new_account : Account <' info , NewAccount

> ,

[account(

mut)] pub signer : Signer <' info

> , pub system_program : Program <' info , System , }

[account]

pub struct NewAccount { data : u64 }

Explanation

Build and Deploy Program#

To build the program, simply runbuild in the terminal.

Terminal build Notice that the address indeclare_id!() has been updated. This is your program's on-chain address.

Output

Once the program is built, rundeploy in the terminal to deploy the program to the network (devnet by default). To deploy a program, SOL must be allocated to the on-chain account that stores the program.

Before deployment, ensure you have enough SOL. You can get devnet SOL by either runningsolana airdrop 5 in the Playground terminal or using theWeb Faucet .

Terminal deploy

Output

Alternatively, you can also use theBuild andDeploy buttons on the left-side panel.

Build and Deploy

Once the program is deployed, you can now invoke its instructions.

Test Program#

Included with the starter code is a test file found intests/anchor.test.ts . This file demonstrates how to invoke theinitialize instruction on the starter program from the client.

anchor.test.ts // No imports needed: web3, anchor, pg and more are globally available

describe ("Test" , () => { it ("initialize" , async () => { // Generate keypair for the new account const newAccountKp = new web3. Keypair ();

// Send transaction const data = new BN (42); const txHash = await pg.program.methods . initialize (data) . accounts ({ newAccount: newAccountKp.publicKey, signer: pg.wallet.publicKey, systemProgram: web3.SystemProgram.programId, }) . signers ([newAccountKp]) . rpc (); console. log (Use 'solana confirm -v { txHash }' to see the logs);

// Confirm transaction await pg.connection. confirmTransaction (txHash);

// Fetch the created account const newAccount = await pg.program.account.newAccount. fetch (newAccountKp.publicKey,);

console. log ("On-chain data is:" , newAccount.data. toString ());

// Check whether the data on-chain is equal to local 'data' assert (data. eq (newAccount.data)); }); }); To run the test file once the program is deployed, runtest in the terminal.

Terminal test You should see an output indicating that the test passed successfully.

Output

You can also use theTest button on the left-side panel.

Run Test

You can then view the transaction logs by running thesolana confirm -v command and specifying the transaction hash (signature) from the test output:

Terminal solana confirm -v [TxHash] For example:

Terminal solana confirm -v 3TewJtiUz1EgtT88pLJHvKFzqrzDNuHVi8CfD2mWmHEBAaMfC5NAaHdmr19qQYfTiBace6XUmADvR4Qrhe8gH5uc

Output

Alternatively, you can view the transaction details onSolanaFM orSolana Explorer by searching for the transaction signature (hash).

Info Reminder to update the cluster (network) connection on the Explorer you are using to match Solana Playground. Solana Playground's default cluster is devnet.

Close Program#

Lastly, the SOL allocated to the on-chain program can be fully recovered by closing the program.

You can close a program by running the following command and specifying the program address found indeclare_id!() :

Terminal solana program close [ProgramID] For example:

Terminal solana program close 2VvQ11q8xrn5tkPNyeraRsPaATdiPx8weLAD8aD4dn2r

Output

Explanation

Congratulations! You've just built and deployed your first Solana program using the Anchor framework!

Previous «Writing to Network Next Program Derived Address»