Getting Started with Anchor

The Anchor framework is a tool that simplifies the process of building Solana programs. Whether you're new to blockchain development or an experienced programmer, Anchor simplifies the process of writing, testing, and deploying Solana programs.

In this section, we'll walk through:

- Creating a new Anchor project
- Building and testing your program
- Deploying to Solana clusters
- Understanding the project file structure

Prerequisites#

For detailed installation instructions, visit theinstallation page.

Before you begin, ensure you have the following installed:

- Rust: The programming language for building Solana programs.
- Solana CLI: Command-line tool for Solana development.
- Anchor CLI: Command-line tool for the Anchor framework.

To verify Anchor CLI installation, open your terminal and run:

Terminal anchor --version Expected output:

Terminal anchor-cli 0.30.1

Getting Started#

This section covers the basic steps to create, build, and test your first local Anchor program.

Create a new Project#

To start a new project, use theanchor init command followed by your project's name. This command creates a new directory with the specified name and sets up a default program and test file.

Terminal anchor init my-program Navigate to the new project directory and open it in your code editor.

Terminal cd my-project The default Anchor program is located at/programs/my-project/src/lib.rs .

Default Program

The default Typescript test file is located at/tests/my-project.ts .

Default Test File

If you prefer Rust for testing, initialize your project with the--test-template rust flag.

anchor init --test-template rust my-program The Rust test file will be at/tests/src/test_initialize.rs .

Rust Test File

Build the Program#

Build the program by runninganchor build .

Terminal anchor build The compiled program will be at/target/deploy/my_project.so . The content of this file is what gets stored on the Solana network (as an executable account) when you deploy your program.

Test the Program#

To test the program, runanchor test .

Terminal anchor test By default, theAnchor.toml config file specifies thelocalnet cluster. When developing onlocalnet ,anchor test will automatically:

1. Start a local Solana validator
2. Build and deploy your program to the local cluster
3. Run the tests in thetests
4. folder
5. Stop the local Solana validator

Alternatively, you can manually start a local Solana validator and run tests against it. This is useful if you want to keep the validator running while you iterate on your program. It allows you to inspect accounts and transaction logs on theSolana Explorer while developing locally.

Open a new terminal and start a local Solana validator by running thesolana-test-validator command.

Terminal solana-test-validator In a separate terminal, run the tests against the local cluster. Use the--skip-local-validator flag to skip starting the local validator since it's already running.

Terminal anchor test --skip-local-validator

Deploy to Devnet#

By default, theAnchor.toml config file in an Anchor project specifies the localnet cluster.

Anchor.toml [toolchain]

[features] resolution = true skip-lint = false

[programs . localnet] my_program = "3ynNB373Q3VAzKp7m4x238po36hjAGFXFJB4ybN2iTyg"

[registry] url = "https://api.apr.dev"

[provider] cluster = "Localnet" wallet = "~/.config/solana/id.json"

[scripts] test = "yarn run ts-mocha -p ./tsconfig.json -t 1000000 tests/*.ts" To deploy your program to devnet, change thecluster value toDevnet . Note that this requires your wallet to have enough SOL on Devnet to cover deployment cost.

-cluster = "Localnet" +cluster = "Devnet" Anchor.toml [provider] cluster = "Devnet" wallet = "~/.config/solana/id.json" Now when you runanchor deploy , your program will be deployed to the devnet cluster. Theanchor test command will also use the cluster specified in theAnchor.toml file.

anchor deploy To deploy to mainnet, simply update theAnchor.toml file to specify the mainnet cluster.

Anchor.toml [provider] cluster = "Mainnet" wallet = "~/.config/solana/id.json"

Update the Program#

Solana programs can be updated by redeploying the program to the same program ID.

To update a program, simply make changes to your program's code and run theanchor build command to generated an updated.so file.

anchor build Then run theanchor deploy command to redeploy the updated program.

anchor deploy

Close the Program#

To reclaim the SOL allocated to a program account, you can close your Solana program.

To close a program, use thesolana program closecommand. For example:

solana program close 3ynNB373Q3VAzKp7m4x238po36hjAGFXFJB4ybN2iTyg --bypass-warning Note that once a program is closed, the program ID cannot be reused to deploy a new program.

Project File Structure#

Below is an overview of default file structure in an Anchor workspace:

. ├── .anchor │ └── program-logs ├── app ├── migrations ├── programs │ └── [project-name] │ └── src │ ├── lib.rs │ ├── Cargo.toml │ └── Xargo.toml ├── target │ ├── deploy │ │ └── [project-name]-keypair.json │ ├── idl │ │ └── [project-name].json │ └── types │ └── [project-name].ts ├── tests │ └── [project-name].ts ├── Anchor.toml ├── Cargo.toml └── package.json

Programs Folder#

The/programs folder contains your project's Anchor programs. A single workspace can contain multiple programs.

Tests Folder#

The/tests folder contains test files for your project. A default test file is created for you when you create your project.

Target Folder#

The/target folder contains build outputs. The main subfolders include:

- /deploy
- : Contains the keypair and program binary for your programs.
- /idl
- : Contains the JSON IDL for your programs.
- /types
- : Contains the TypeScript type for the IDL.

Anchor.toml File#

TheAnchor.toml file configures workspace settings for your project.

.anchor Folder#

Includes aprogram-logs file that contains transaction logs from the last run of test files.

App Folder#

The/app folder is an empty folder that can be optionally used for your frontend code.

Previous «Clusters & Endpoints Next Program Structure»