

# Developing on-chain programs

Developers can write and deploy their own programs to the Solana blockchain. While developing these "on-chain" programs can seem cumbersome, the entire process can be broadly summarized into a few key steps.

## On-chain program development lifecycle#

1. Setup your development environment
2. Write your program
3. Compile the program
4. Generate the program's public address
5. Deploy the program

### 1. Setup your development environment#

The most robust way of getting started with Solana development, is [installing the Solana CLI](#) tools on your local computer. This will allow you to have the most powerful development environment.

Some developers may also opt for using [Solana Playground](#) , a browser based IDE. It will let you write, build, and deploy on-chain programs. All from your browser. No installation needed.

### 2. Write your program#

Writing Solana programs is most commonly done so using the Rust language. These Rust programs are effectively the same as creating a traditional [Rust library](#) .

You can read more about other [supported languages](#) below.

### 3. Compile the program#

Once the program is written, it must be compiled down to [Berkley Packet Filter](#) byte-code that will then be deployed to the blockchain.

### 4. Generate the program's public address#

Using the [Solana CLI](#) , the developer will generate a new unique [Keypair](#) for the new program. The public address (aka [Pubkey](#) ) from this Keypair will be used on-chain as the program's public address (aka [programId](#) ).

### 5. Deploying the program#

Then again using the CLI, the compiled program can be deployed to the selected blockchain cluster by creating many transactions containing the program's byte-code. Due to the transaction memory size limitations, each transaction effectively sends small chunks of the program to the blockchain in a rapid-fire manner.

Once the entire program has been sent to the blockchain, a final transaction is sent to write all of the buffered byte-code to the program's data account. This either mark the new program as [executable](#) , or complete the process to upgrade an existing program (if it already existed).

## Support languages#

Solana programs are typically written in the [Rust language](#) , but [C/C++](#) are also supported.

There are also various community driven efforts to enable writing on-chain programs using other languages, including:

- Python via [Seahorse](#)
- (that acts as a wrapper the Rust
- based Anchor framework)

## Example programs#

You can also explore the [Program Examples](#) for examples of on-chain programs.

## Limitations#

As you dive deeper into program development, it is important to understand some of the important limitations associated

with on-chain programs.

Read more details on the [Limitations](#) page

## Frequently asked questions#

Discover many of the [frequently asked questions](#) other developers have about writing/understanding Solana programs.