Deploying Programs

Solana on-chain programs (otherwise known as "smart contracts") are stored in "executable" accounts on Solana. These accounts are identical to any other account but with the exception of:

- having the "executable" flag enabled, and
- the owner being assigned to a BPF loader

Besides those exceptions, they are governed by the same runtime rules as non-executable accounts, hold SOL tokens for rent fees, and store a data buffer which is managed by the BPF loader program. The latest BPF loader is called the "Upgradeable BPF Loader".

Overview of the Upgradeable BPF Loader#

State accounts#

The Upgradeable BPF loader program supports three different types of state accounts:

1. Program account
2. :
3. This is the main account of an on-chain program and its address is commonly
4. referred to as a "program id." Program id's are what transaction instructions
5. reference in order to invoke a program. Program accounts are immutable once
6. deployed, so you can think of them as a proxy account to the byte-code and
7. state stored in other accounts.
8. Program data account
9. :
10. This account is what stores the executable byte-code of an on-chain program.
11. When a program is upgraded, this account's data is updated with new
12. byte-code. In addition to byte-code, program data accounts are also
13. responsible for storing the slot when it was last modified and the address of
14. the sole account authorized to modify the account (this address can be
15. cleared to make a program immutable).
16. Buffer accounts
17. :
18. These accounts temporarily store byte-code while a program is being actively
19. deployed through a series of transactions. They also each store the address
20. of the sole account which is authorized to do writes.

Instructions#

The state accounts listed above can only be modified with one of the following instructions supported by the Upgradeable BPF Loader program:

1. Initialize buffer
2. :
3. Creates a buffer account and stores an authority address which is allowed to
4. modify the buffer.
5. Write
6. :
7. Writes byte-code at a specified byte offset inside a buffer account. Writes
8. are processed in small chunks due to a limitation of Solana transactions
9. having a maximum serialized size of 1232 bytes.
10. Deploy
11. :
12. Creates both a program account and a program data account. It fills the
13. program data account by copying the byte-code stored in a buffer account. If
14. the byte-code is valid, the program account will be set as executable,
15. allowing it to be invoked. If the byte-code is invalid, the instruction will
16. fail and all changes are reverted.
17. Upgrade
18. :
19. Fills an existing program data account by copying executable byte-code from a
20. buffer account. Similar to the deploy instruction, it will only succeed if
21. the byte-code is valid.
22. Set authority
23. :

24. Updates the authority of a program data or buffer account if the account's
25. current authority has signed the transaction being processed. If the
26. authority is deleted without replacement, it can never be set to a new
27. address and the account can never be closed.
28. Close
29. :
30. Clears the data of a program data account or buffer account and reclaims the
31. SOL used for the rent exemption deposit.

How solana program deploy works#

Deploying a program on Solana requires hundreds, if not thousands of transactions, due to the max size limit of 1232 bytes for Solana transactions. The Solana CLI takes care of this rapid firing of transactions with thesolana program deploy subcommand. The process can be broken down into the following 3 phases:

1. Buffer initialization
2. :
3. First, the CLI sends a transaction whichcreates a buffer account
4. large enough for the byte-code being deployed. It also invokes theinitialize buffer instruction
5. to set the buffer authority to restrict writes to the deployer's chosen
6. address.
7. Buffer writes
8. :
9. Once the buffer account is initialized, the CLIbreaks up the program byte-code
10. into ~1KB chunks andsends transactions at a rate of 100 transactions per second
11. to write each chunk withthe write buffer instruction
12. .
13. These transactions are sent directly to the current leader's transaction
14. processing (TPU) port and are processed in parallel with each other. Once all
15. transactions have been sent, the CLIpolls the RPC API with batches of transaction signatures
16. to ensure that every write was successful and confirmed.
17. Finalization
18. :
19. Once writes are completed, the CLIsends a final transaction
20. to eitherdeploy a new program
21. orupgrade an existing program
22. .
23. In either case, the byte-code written to the buffer account will be copied
24. into a program data account and verified.

Reclaim rent from program accounts#

The storage of data on the Solana blockchain requires the payment ofrent , including for the byte-code for on-chain programs. Therefore as you deploy more or larger programs, the amount of rent paid to remain rent-exempt will also become larger.

Using the current rent cost model configuration, a rent-exempt account requires a deposit of ~0.7 SOL per 100KB stored. These costs can have an outsized impact on developers who deploy their own programs sinceprogram accounts are among the largest we typically see on Solana.

Example of how much data is used for programs#

As a data point of the number of accounts and potential data stored on-chain, below is the distribution of the largest accounts (at least 100KB) at slot103,089,804 onmainnet-beta by assigned on-chain program:

1. Serum Dex v3
2. : 1798 accounts
3. Metaplex Candy Machine
4. : 1089 accounts
5. Serum Dex v2
6. : 864 accounts
7. Upgradeable BPF Program Loader
8. : 824 accounts
9. BPF Program Loader v2
10. : 191 accounts
11. BPF Program Loader v1
12. : 150 accounts

Reclaiming buffer accounts#

Buffer accounts are used by the Upgradeable BPF loader to temporarily store byte-code that is in the process of being deployed on-chain. This temporary buffer is required when upgrading programs because the currently deployed program's byte-code cannot be affected by an in-progress upgrade.

Unfortunately, deploys fail occasionally and instead of reusing the buffer account, developers might retry their deployment with a new buffer and not realize that they stored a good chunk of SOL in a forgotten buffer account from an earlier deploy.

As of slot103,089,804 onmainnet-beta there are 276 abandoned buffer accounts that could be reclaimed! Developers can check if they own any abandoned buffer accounts by using the Solana CLI:

solana program show --buffers --keypair ~/.config/solana/MY_KEYPAIR.json

Buffer Address | Authority | Balance 9vXW2c3qo6DrLHa1Pkya4Mw2BWZSRYs9aoyoP3g85wCA | 2nr1bHFT86W9tGnyvmYW4vcHKsQB3sVQfnddasz4kExM | 3.41076888 SOL And they can close those buffers to reclaim the SOL balance with the following command:

solana program close --buffers --keypair ~/.config/solana/MY_KEYPAIR.json

Fetch the owners of buffer accounts via RPC API#

The owners of all abandoned program deploy buffer accounts can be fetched via the RPC API:

curl http://api.mainnet-beta.solana.com -H "Content-Type: application/json" \ --data-binary @- << EOF | jq --raw-output '.result | .[] | .account.data[0]' { "jsonrpc":"2.0", "id":1, "method":"getProgramAccounts", "params":["BPFLoaderUpgradeab1e11111111111111111111111", { "dataSlice": {"offset": 5, "length": 32}, "filters": [{"memcmp": {"offset": 0, "bytes": "2UzHM"}}], "encoding": "base64" }] } EOF After re-encoding the base64 encoded keys into base58 and grouping by key, we see some accounts have over 10 buffer accounts they could close, yikes!

'BE3G2F5jKygsSNbPFKHHTxvKpuFXSumASeGweLcei6G3' => 10 buffer accounts
'EsQ179Q8ESroBnnmTDmWEV4rZLkRc3yck32PqMxypE5z' => 10 buffer accounts
'6KXtB89kAgzW7ApFzqhBg5tgnVinzP4NSXVqMAWnXcHs' => 12 buffer accounts
'FinVobfi4tbdMdfN9jhzUuDVqGXfcFnRGX57xHcTWLfW' => 15 buffer accounts
'TESAinbTL2eBLkWqyGA82y1RS6kArHvuYWfkL9dKkbs' => 42 buffer accounts

Reclaiming program data accounts#

You may now realize that program data accounts (the accounts that store the executable byte-code for an on-chain program) can also be closed.

Note: This doesnot mean thatprogram accounts can be closed (those are immutable and can never be reclaimed, but it's fine they're pretty small). It's also important to keep in mind that once program data accounts are deleted, they can never be recreated for an existing program. Therefore, the corresponding program (and its program id) for any closed program data account is effectively disabled forever and may not be re-deployed While it would be uncommon for developers to need to close program data accounts since they can be rewritten during upgrades, one potential scenario is that since program data accounts can't beresized . You may wish to deploy your program at a new address to accommodate larger executables.

The ability to reclaim program data account rent deposits also makes testing and experimentation on themainnet-beta cluster a lot less costly since you could reclaim everything except the transaction fees and a small amount of rent for the program account. Lastly, this could help developers recover most of their funds if they mistakenly deploy a program at an unintended address or on the wrong cluster.

To view the programs which are owned by your wallet address, you can run:

solana -V # must be 1.7.11 or higher! solana program show --programs --keypair ~/.config/solana/MY_KEYPAIR.json

Program Id | Slot | Authority | Balance CN5x9WEusU6pNH66G22SnspVx4cogWLqMfmb85Z3GW7N | 53796672 | 2nr1bHFT86W9tGnyvmYW4vcHKsQB3sVQfnddasz4kExM | 0.54397272 SOL To close those program data accounts and reclaim their SOL balance, you can run:

solana program close --programs --keypair ~/.config/solana/MY_KEYPAIR.json You might be concerned about this feature allowing malicious actors to close a program in a way that negatively impacts end users. While this is a valid concern in general, closing program data accounts doesn't make this any more exploitable than was already possible.

Even without the ability to close a program data account, any upgradeable program could be upgraded to a no-op implementation and then have its upgrade authority cleared to make it immutable forever. This new feature for closing program data accounts merely adds the ability to reclaim the rent deposit, disabling a program was already technically possible.