BITCOIN NFT'S
ORDINALS

Presented by: Rana Ahsan Ansar

Introduction to Ordinals Protocol

- BTC Ordinals were introduced as a method of generating Bitcoin NFTs by attaching information to individual satoshi.
- Ordinals Protocol is a system, In which each satoshi is linked by a serial number, it allow users to make individual satoshi unique by attaching extra data to them. This process is known as “Inscription”.
- Ordinal inscriptions are digital assets, similar to NFTs, inscribed on a satoshi in the Bitcoin network. All this data in minted on blockchain. This process has been made possible because of Taproot soft fork launched on November 14, 2021. Ordinal protocol allows the tracking and transfer of individual satoshis using ordinal wallet.

Explanation with example

Amuse that we have a table without serial number represent data coming from local node

Satoshi	Data
1.000000001 Sat	NFT Hash
1.000000002 Sat	NFT Hash
1.000000003 Sat	NFT Hash
1.000000004 Sat	NFT Hash

Explanation with example

After applying ordinal protocol each satoshi is represent by a unique primary key.

Ordinal Number	Satoshi	Data
0	1.000000001 Sat	NFT Hash
1	1.000000002 Sat	NFT Hash
2	1.000000003 Sat	NFT Hash
3	1.000000004 Sat	NFT Hash

Representations of Ordinal Numbers

Integer notation

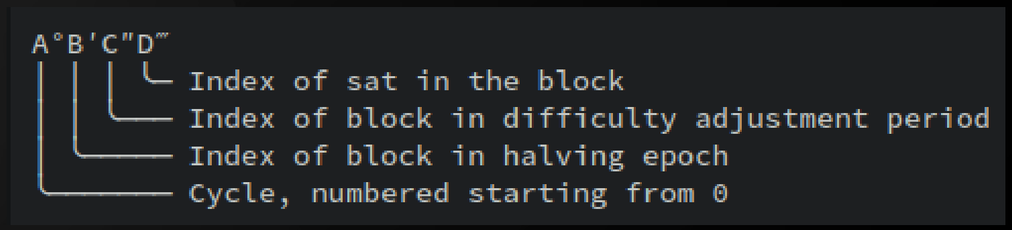
2099994106992659 The ordinal number, assigned according to the order in which the satoshi was mined.

Decimal notation

3891094.16797 The first number is the block height in which the satoshi was mined, the second the offset of the satoshi within the block.

Degree notation

3°111094′214″16797‴



Percentile notation

99.99971949060254% . The satoshi's position in Bitcoin's supply, expressed as a percentage.

Name

An encoding of the ordinal number using the characters a through z.

Methods to create Ordinal Inscriptions

Method 1

By using Ordinal-compatible wallets or online services. For example: Ordinals Wallet, Unisat wallet and Gama service.
This method is for beginners and non-technical Users

Method 2

By running a full node and sync ordinal server with the local BTC Validate node
This method is for developers

Steps for creating ordinal inscription using BTC full node

Steps

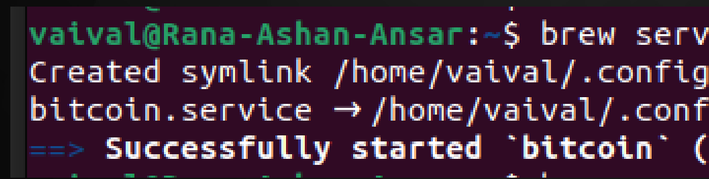
- *Install bitcoin core*
- *Sync the Bitcoin node with main net*
- *Install Ord server. Using Cargo*
- *Sync the Ord server with Bitcoin node*
- *Create ordinal wallet*
- *Send Bitcoins in to ordinal wallet*
- *Create your Inscriptions using cli command.*

Installation of Bitcoin node

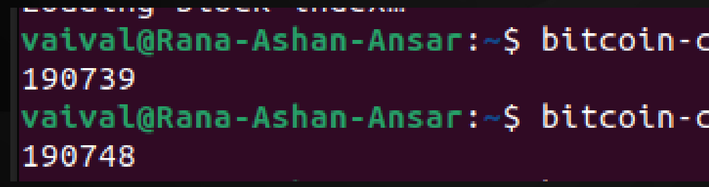
There are two method to run a bitcoin Validate node

- Using Bitcoin core (User Interface Client)
- Using HomeBrew Tool. (Command Line Interface)
- HomeBrew is used to install all packages & dependencies for bitcoin-core

Step for Installing Bitcoin using Brew

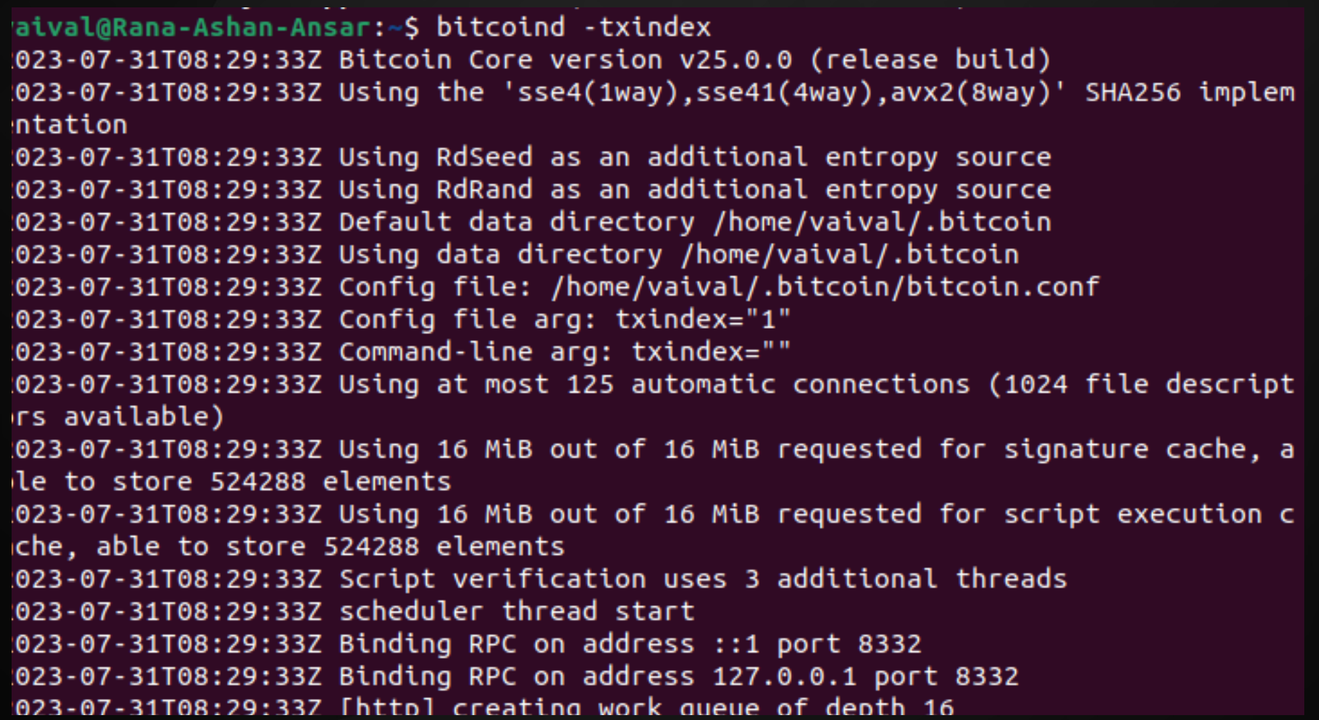
- Install HomeBrew by using command available on http://brew.sh
- Install bitcoin using "brew insall bitcoin"
- Start bitcoin node using "brew services start bitcoin"
- 

- Wait for complete download and check Block count

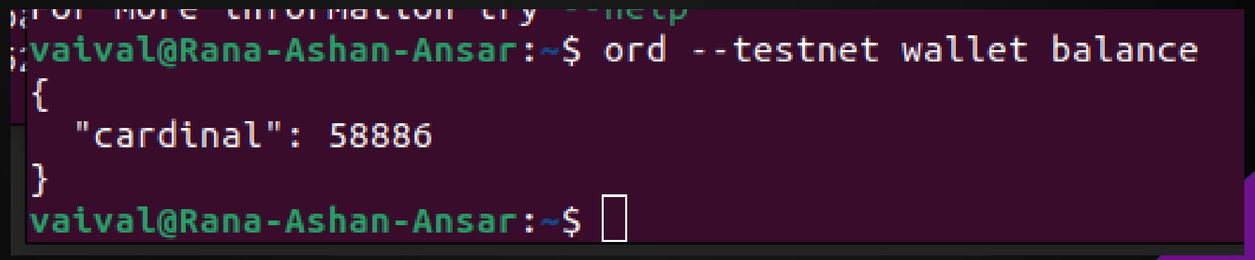


Start bitcoin with txindex

- After downloaded complete node, sync the ordinal server with blocks using command "bitcoind -txindex"
-



Next steps after running bitcoin node

- *Install cargo with rust*
- *Install ord using cargo “ cargo install ord ”*
- *create a new ord wallet using “ord wallet create”*
- *check ord wallet balance, It will automatically sync blocks with ordinal numbers.*
- 

Continue

- *Transfer some bitcoins from your main wallet to Ord wallet*
- *Create your NFT or FT file.*
- *Create inscriptions using "ord wallet inscribe --fee-rate FEE_RATE FILE_PATH"*

BRC-20

BRC-20 tokens use same process of “inscription” to add a short piece of JSON code to create tokens.

```
{
  "p": "brc-20",
  "op": "deploy",
  "tick": "ordi",
  "max": "21000000",
  "lim": "1000"
}
```

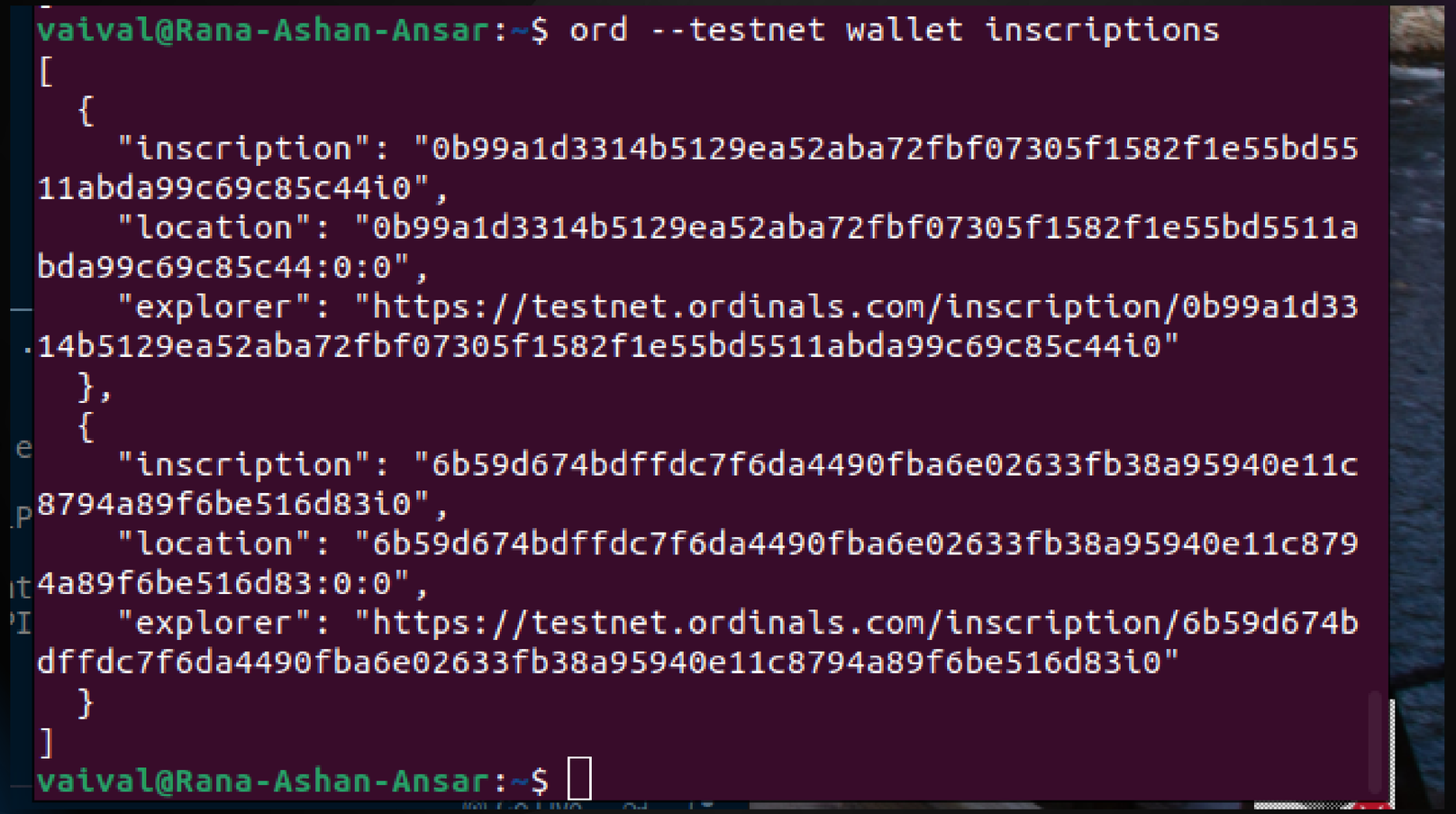
Key	Required ?	Description
p	Yes	Protocol: Helps other systems identify and process brc-20 events
op	Yes	Operation: Type of event (Deploy, Mint, Transfer)
tick	Yes	Ticker: 4 letter identifier of the brc-20
max	Yes	Max supply: set max supply of the brc-20
lim	No	Mint limit: If letting users mint to themsleves, limit per ordinal
dec	No	Decimals: set decimal precision, default to 18

BRC-20 Continue

- **Deploy a Token Contract.** The BRC-20 token contract is a JSON file that defines the four-letter name, maximum supply, decimals and mint limit. You can deploy your token by inscribing the JSON file onto a satoshi using a wallet that supports ordinals and inscriptions. This creates the first set of tokens of your BRC-20 token and assigns it to your wallet address.
- **Mint More Tokens.** To mint more of your new BRC-20 token, you can inscribe another JSON file onto another satoshi using the same process. The JSON file will be slightly shorter since you don’t have to specify the same information from the initial deployment.
- **Transfer Tokens.** To transfer tokens of your BRC-20 token to another wallet address, the inscription process is the same as above. This JSON file should contain the number of tokens you want to transfer, the name of the token contract you deployed in the first step and the recipient’s address. This should decrease the balance of your wallet address and increase the balance of the recipient’s wallet address.

BRC-20 Continue

Here is screen short of my minted Inscription on testnet



Track Inscription using Block Explorer

Now you can track your inscription using Ordinal Block Explorer

Block 801038

hash
00000000000000000000448daec378fe243a5c204fafdbe6adf216c28b8f08156

target
00000000000000000000000000000000000000000000261050000000000000000000

timestamp
2023-07-31 07:56:45 UTC

size
1734226

weight
3997813

previous blockhash
00000000000000000000054c703803dc982bfb6f56d0e590ade5a25ed49a4b3fbc

prev next

2451 Transactions

- 0642066a3a7ad225604ded9d1d499ad4ed9d90b9612fce929b6cf5fcb09578fa
- 7bda689b537a7aff7219eeb57b12c91e70817164f1c81d6c682723818b634e50
- 11da9361401aed1777c1bb251095507f1e20195280342d3350e44b73c8f4005e
- dc89c6b4957f3312f8932e70ee57bc9f1dfd6556aeee2c08e7ca014c7397e2cc
- cf8e097b0eff0d7e4150c4c9c7a101c596b05d66e845c08fe74728910787f4c0
- 1ec32a66f3d1de1e1528b6dacf959ab8fbc17dfa77035235e72c234e46e535b3
- 869296c087df99883c5f44e1286a80ad6553410ce5ff1c7eb6a6617654752cc4
- dfb68b766c8b679624d9086b073a75213982929a0de1d5d4fc2b348cb730fb95
- 9a0160da4c581f41850695c26c49efa902adbfbd6cc97fc256fe0096b8a71e58
- ba8d11ba36c39cad3e3e013db24ffc8cbe10a4de59e5f72535bc0b06420276ae
- 93a7c63b9ef9dbab8bfbd9df82e5bd8339860fa1e0546740f11f8e10a519e646
- 06dd485bc924a3696292ab5661d74cbbcdbc258845c5bec897aae863f2877ee8

Bitcoin Node API

- Bitcoin-core allow us to create API using “bitcoin-core” library. Using this API we can create our own block explorer.
- Before building API we need to add a configuration file (bitcoin.conf) in Bitcoin directory and declare RPC User and RPC password in it. e.g. (-rpcuser=”NAME” -rpcpassword=”12345678”)
- I push my example API in Final project directory on git-hub.
- Git-repository: https://github.com/ranaahsanansar/BlcokchainTraning