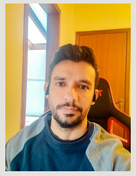
Bitup Coders 01

Rodando Nós e Explorando as Redes do Bitcoin

Mainnet, Testnet, Signet e Regtest na prática

Anfitrião:
Rafael Penna



Bitup Coders

🎯 Para quem é esta apresentação?

👥 Entusiastas do Bitcoin, com ou sem conhecimento técnico

🔧 Desenvolvedores e curiosos, que querem entender como o Bitcoin funciona por dentro

💡 Gente que acredita que estudar Bitcoin transforma perspectivas

💬 Um espaço aberto para compartilhar conhecimento, dúvidas e aprendizados

Sobre o Bitup Dev:

📍 Encontro colaborativo e descontraído

🤝 Ninguém aqui sabe tudo — estamos todos aprendendo

📚 A proposta é explorar o lado técnico do Bitcoin, sem medo

🧭 Nosso ponto de partida hoje:

Como funciona um nó Bitcoin na prática?

Como podemos interagir com as diferentes redes de forma segura e soberana?

Bitcoin é um protocolo, não um produto

📡 Um **protocolo** é um conjunto de regras que definem como participantes devem se comunicar.

🧩 No caso do Bitcoin, essas regras garantem:
- Segurança
- Consenso
- Validação de transações
- Emissão e escassez de moeda

🎯 “O Bitcoin não é uma empresa, nem uma aplicação. É uma rede descentralizada regida por software.”

O Nó é o guardião das regras

🖥️ Um Nó é um **software** que executa o protocolo Bitcoin em um dispositivo.

🛡️ Ele garante que todas as regras sejam seguidas de forma independente.

🔧 Funções de um Nó:

- 📥 Recebe blocos e transações
- 🔍 Valida tudo localmente
- 📤 Propaga dados para outros nós
- 📚 Mantém a blockchain
- ⚒️ Pode minerar blocos (se configurado)
- 🔌 Pode servir como API para apps e serviços

Rodar um Nó Bitcoin é para todos

Mas de jeitos diferentes…

🔻 **Usuário comum:**

“Quero participar da rede, mas sem complicação.”

🧑‍💼 **Alternativas amigáveis:**

✅ Bitcoin Core com interface gráfica (bitcoin-qt)
- Instalação simples
- Janela visual
- Sincronização automática

✅ Umbrel, MyNode, Start9, etc.
- Interface web
- Foco em soberania pessoal
- Ideal para Raspberry Pi ou PC

Rodar um Nó Bitcoin é para todos

Mas de jeitos diferentes…

🔺 **Desenvolvedor:**

“Quero controle total e acesso ao protocolo.”

👩‍💻 **Ferramentas:**

⚙️ Bitcoin Core (bitcoind + bitcoin-cli)
- Linha de comando
- bitcoin.conf
- RPC manual
- Total liberdade, mas exige conhecimento técnico

Nem todo nó é igual. Existem diferentes papéis

🟢 Full Node (Nó Completo)

🧠 Valida todas as regras do protocolo

📦 Baixa a blockchain inteira

🛡️ Não confia em ninguém — tudo é verificado localmente

🔁 Ajuda a rede retransmitindo blocos e transações

🟡 Light Node (SPV)

📬 Baixa só os headers dos blocos

🌳 Usa provas via Merkle Tree para validar transações

📲 Leve e ideal para celulares

⚠️ Menos privacidade e precisa confiar em um Full Node

🔵 Miner Node

⚙️ É um Full Node com software de mineração

🔍 Valida blocos e tenta encontrar o próximo hash válido

🚀 Quando encontra, propaga o bloco para a rede

🎯 “Agora que entendemos os papéis, vamos ver como interagir e rodar os Nós na prática.”

Instalação rápida do Bitcoin Core no Ubuntu

Passo a passo:

```
#Baixe e instale o Bitcoin Core
https://bitcoincore.org/en/download/
https://bitcoincore.org/bin/bitcoin-core-29.0/bitcoin-29.0-x86_64-linux-gnu.t

#Extraia e mova para /opt:
tar -xvf bitcoin-29.0-x86_64-linux-gnu.tar.gz
sudo mv bitcoin-29.0 /opt/bitcoin

#Adicione o executável ao PATH (opcional):
gedit ~/.bashrc
#Colocar no final do arquivo:
export PATH="/opt/bitcoin/bitcoin-29.0/bin:$PATH"
#Atualizar:
source ~/.bashrc

#Verifique a instalação:
bitcoind --version
bitcoin-cli --version
```

Instalação rápida do Bitcoin Core no Ubuntu

🧩 O que foi instalado?

Programa	Função
bitcoind	🛠️ Daemon principal: roda o nó, valida blocos, mantém a blockchain
bitcoin-cli	💬 Cliente de comandos RPC: envia instruções e consulta dados do bitcoind

💻 Agora no terminal:

```
$ bitcoind --version
$ bitcoin-cli --version
```

Duas formas de configurar o Bitcoin Core

🛠️ 1. Parâmetros direto no terminal
Passados junto com o comando **bitcoind** ou **bitcoin-cli**

```
bitcoind -testnet -daemon
bitcoin-cli -signet getblockcount
```

✅ Rápido, para testes
⚠️ Não fica salvo após reiniciar



Duas formas de configurar o Bitcoin Core

📄 2. Arquivo bitcoin.conf
Arquivo de configuração persistente (lido na inicialização)

📂 Local típico: dentro da pasta definida para o Nó e usada com -datadir

```
# Arquivo bitcoin.conf
regtest=1
rpcuser=usuario
rpcpassword=senha
prune=2000
```

✅ Boa organização
✅ Permite configurações duradouras
✅ Ideal para setups consistentes

Iniciar o bitcoind e escolher a rede certa

📁 1. Criar a pasta que irá armazenar os dados do nó

```
mkdir -p /home/user/bitcoin-no1
```

📝 2. Criar o bitcoin.conf dentro dessa pasta

Exemplo:

```
# Arquivo bitcoin.conf
regtest=1
rpcuser=usuario
rpcpassword=senha
```

▶️ 3. Iniciar o Nó

```
bitcoind -datadir=/home/user/bitcoin-no1 -daemon
```

🛠️ O daemon começa a rodar em segundo plano

Iniciar o bitcoind e escolher a rede certa

⏹️ 4. Parar o Nó

```
bitcoin-cli -datadir=/home/user/bitcoin-no1 stop
```

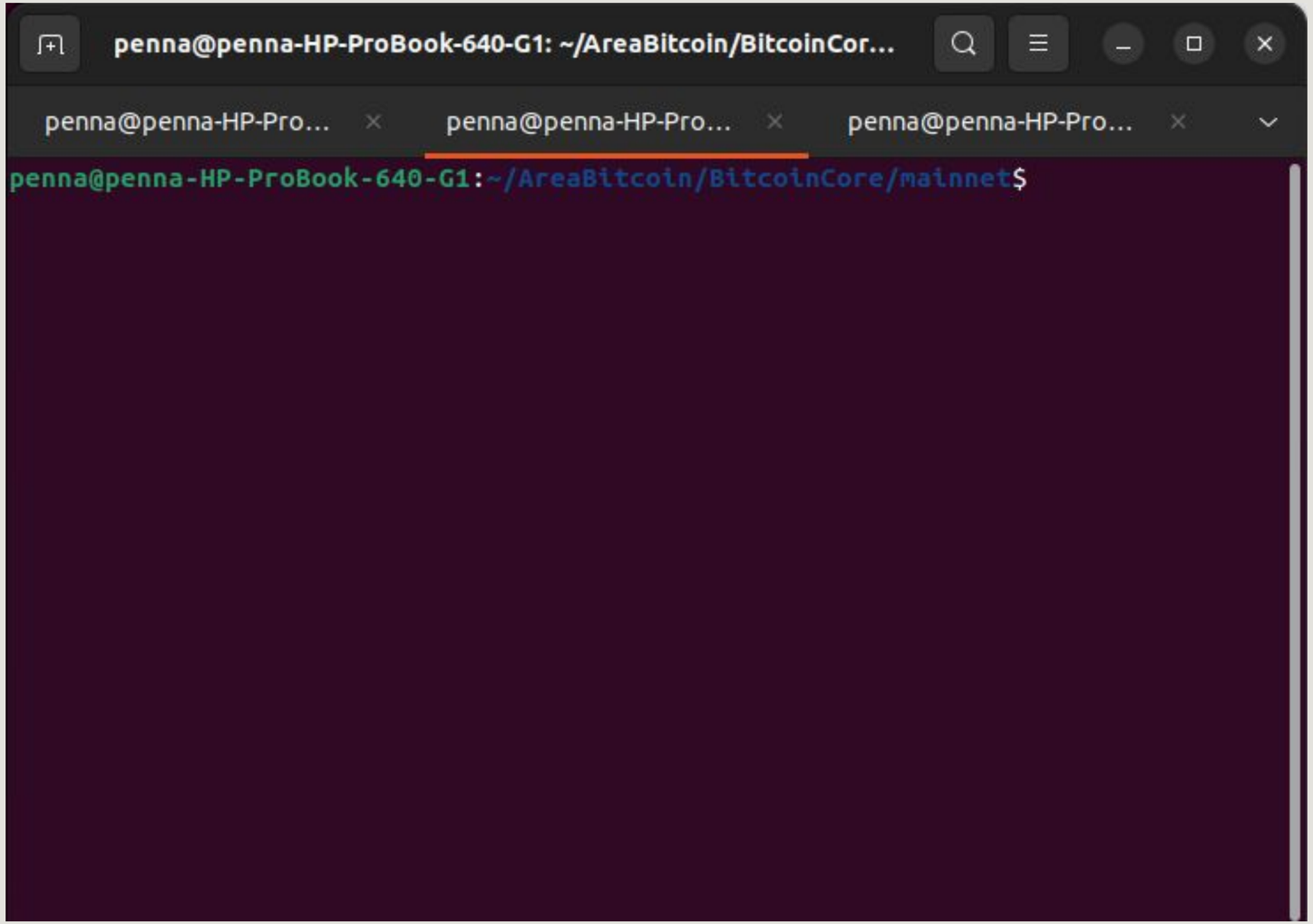
🌐 Modos de Operação

Modo	**Descrição**
mainnet	🌍 Rede principal real
testnet	🔬 Rede de testes pública
signet	🧪 Rede de testes controlada
regtest	🧰 Rede local para desenvolvedores

✅ Escolha a rede no bitcoin.conf com:
regtest=1 ou **testnet=1** ou **signet=1**
* se não colocar nenhuma das 3, então é mainnet

Rodando um Nó

💻 Vamos para o terminal



Rodando um Nó

💻 Com o bitcoind ativo, o Nó começa a trabalhar.

🔗 Conexões e sincronização (P2P)
- 📡 Conecta-se a outros nós automaticamente
- 📥 Recebe blocos e transações
- 📤 Propaga para outros nós

🧠 Validação e armazenamento
- ✅ Verifica regras de consenso
- 💾 Mantém e atualiza a blockchain (se for full node)

💬 Disponibiliza API RPC
- 📡 Interface de comunicação via bitcoin-cli
- ⚙️ Permite comandos para consultar, transacionar, criar carteiras, etc.

Rodando um Nó

🧪 Vamos testar!
Primeiro comando: consultar a blockchain:

bitcoin-cli -datadir=/home/user/bitcoin-no1 getblockchaininfo


```
{
  "chain": "main",
  "blocks": 668590,
  "headers": 907322,
  "bestblockhash":
"000000000000000000ce840d00833cf1770dba7046617870fe251cde058c224",
  "bits": "170d8457",
  "target": "0000000000000000000d8457000000000000000000000000000000000000000",
  "difficulty": 20823531150111.52,
  "time": 1612168054,
  "mediantime": 1612166182,
  "verificationprogress": 0.5014882382027116,
  "initialblockdownload": true,
  "chainwork": "0000000000000000000000000000000000000000018d66d4414d521eaadef2177",
  "size_on_disk": 874044842,
  "pruned": true,
  "pruneheight": 668130,
  "automatic_pruning": true,
  "prune_target_size": 2097152000,
  "warnings": [
  ]
}
```


Rodando um Nó

🧪 Vamos testar!
Primeiro comando: consultar a blockchain:

bitcoin-cli -datadir=/home/user/bitcoin-no1 getblockchaininfo


```
{
  "chain": "main",
  "blocks": 668590,
  "headers": 907322,
  "bestblockhash":
"000000000000000000ce840d00833cf1770dba7046617870fe251cde058c224",
  "bits": "170d8457",
  "target": "0000000000000000000d8457000000000000000000000000000000000000000",
  "difficulty": 20823531150111.52,
  "time": 1612168054,
  "mediantime": 1612166182,
  "verificationprogress": 0.5014882382027116,
  "initialblockdownload": true,
  "chainwork": "0000000000000000000000000000000000000000018d66d4414d521eaadef2177",
  "size_on_disk": 874044842,
  "pruned": true,
  "pruneheight": 668130,
  "automatic_pruning": true,
  "prune_target_size": 2097152000,
  "warnings": [
  ]
}
```


JSON

Rodando um Nó

🔍 Podemos acessar os campos individualmente com jq:

```
bitcoin-cli ... getblockchaininfo | jq '.blocks'
```

668590

🧩 Exemplos úteis:

Comando	**Retorno**
.chain	Nome da rede em uso
.verificationprogress	Progresso real da verificação
.pruned	Se está em modo prune
.size_on_disk	Tamanho atual da blockchain armazenada

Rodando um Nó

🔍 Com os dados estruturados podemos automatizar scripts.

📝 Script para diagnósticos no terminal:

```
DATADIR="."
INFO=$(bitcoin-cli -datadir="$DATADIR" getblockchaininfo)

BLOCKS=$(jq '.blocks' <<< "$INFO")
HEADERS=$(jq '.headers' <<< "$INFO")
VERIF_PROGRESS=$(jq '.verificationprogress' <<< "$INFO")

PERCENT_BLOCOS=$(echo "scale=2; 100 * $BLOCKS / $HEADERS" | bc)
PERCENT_VERIF=$(echo "scale=4; 100 * $VERIF_PROGRESS" | bc)

echo "⏱️ Atualizado em: $(date)"
echo "📦 Blocos verificados: $BLOCKS / $HEADERS (${PERCENT_BLOCOS}%)"
echo "🧠 Verificação real (peso computacional): ${PERCENT_VERIF}%"
```

Rodando um Nó

🔍 Com os dados estruturados podemos automatizar scripts.

📝 Script para diagnósticos no terminal:

```
⏱️ Atualizado em: sáb 26 jul 2025 18:09:06 -03
📦 Blocos verificados: 669817 / 907324 (73.82%)
🧠 Verificação real (peso computacional): 50.368734771217200%
```

Vamos agora ver as 4 redes!

Mainnet

🟠 O que é a Mainnet?

🌍 Rede oficial e principal do Bitcoin

🪙 Transações com BTC real e irreversível

✅ Regras de consenso rigorosas

🌐 Alta descentralização com milhares de nós e mineradores

⚠️ Cuidado total:

Tudo que acontece aqui tem consequências reais. As perdas são permanentes.

Mainnet

💾 Tamanho da blockchain (jul/2025):
📦 +906.000 blocos
💽 ~675 GB de dados

🪚 Usando prune para reduzir espaço:
prune=2000 # Armazena apenas 2 GB mais recentes

🔍 Mesmo com prune:
O nó ainda precisa verificar todos os blocos desde o gênesis na primeira sincronização.

🧠 E mais:
Também é necessário espaço extra para a UTXO (~10 GB)

Mainnet

⚙️ Configuração do **bitcoin.conf** para rodar na Mainnet:

📄 Exemplo típico:

```
rpcuser=teste
rpcpassword=teste
rpcallowip=127.0.0.1
fallbackfee=0.0001
prune=2000
```

🧩 O que cada linha faz:

Linha	Função
rpcuser	Nome de usuário para autenticar comandos via bitcoin-cli
rpcpassword	Senha correspondente
rpcallowip=127.0.0.1	Permite acesso RPC **apenas local**
fallbackfee=0.0001	Define uma taxa mínima padrão (usada quando nenhuma fee é definida)
prune=2000	**Limita o espaço em disco** usado pela blockchain a 2 GB

🔐 **Importante:**

Esse arquivo deve estar dentro da pasta usada com o -datadir, por exemplo:
/home/user/bitcoin-no1/bitcoin.conf

Mainnet

💻 Rodando o Nó

bitcoind -datadir="." -maxconnections=0 -daemon

*** maxconnections=0 não se conecta a Nós e não fica tentando baixar e validar blocos

🔐 Gerando endereços

bitcoin-cli -datadir="." createwallet "Carteira1"
bitcoin-cli -datadir="." -rpcwallet="Carteira1" getnewaddress "legacy" legacy
bitcoin-cli -datadir="." -rpcwallet="Carteira1" getnewaddress "p2sh" p2sh-segwit
bitcoin-cli -datadir="." -rpcwallet="Carteira1" getnewaddress "bech32" bech32
bitcoin-cli -datadir="." -rpcwallet="Carteira1" getnewaddress "taproot" bech32m

🔐 Endereços gerados:

📬 Endereço P2PKH (Legacy): 1HrkEH5sEq121METoQMVcJiSKN4XU7zEDs
📬 Endereço P2SH-SegWit: 3Eiznq7S9FUyHQCcJd3bxDCntqsaT6S6Pa
📬 Endereço Bech32 (P2WPKH): bc1qaz0pq6unr9esy59y6lry4fp2qhln89053edhg4
📬 Endereço Taproot (Bech32m): bc1pwn9lmspwg6fwm7x4k4fe3ek…4n0ewgqf99u4r

🔗 Vendo o Hash de um bloco:

bitcoin-cli -datadir="." getblockhash 50000
000000001aeae195809d120b5d66a39c83eb48792e068f8ea1fea19d84a4278a
bitcoin-cli -datadir="." getblockhash 500000
00000000000000000024fb37364cbf81fd49cc2d51c09c75c35433c3a1945d04



Testnet

🟦 O que é a Testnet?

🔬 Rede pública de testes do Bitcoin

🧪 Permite testar transações e comandos sem risco

🪙 Moedas sem valor real (tBTC), obtidas via faucet

🧠 Ideal para aprendizado, desenvolvimento e simulações

🌐 Funciona semelhantemente à Mainnet (com consenso real)

⚠️ Atenção:

❌ Pode ser instável: blocos órfãos e confirmações irregulares

❗ Alguns serviços de faucet podem demorar ou falhar

Testnet

💾 Tamanho da blockchain (jul/2025):

- 📦 +4.500.000 blocos
- 💽 ~170 GB de dados em disco

⚙️ Configuração do **bitcoin.conf** para rodar na Testnet:

📄 Exemplo típico:

```
testnet=1
rpcuser=teste
rpcpassword=teste
prune=2000
```

🧩 O que cada linha faz:

Linha	Função
testnet=1	Ativa o modo Testnet
rpcuser	Nome de usuário para autenticar comandos via bitcoin-cli
rpcpassword	Senha correspondente
prune=2000	**Limita o espaço em disco** usado pela blockchain a 2 GB

Testnet

💻 Rodando o Nó

bitcoind -datadir="." -maxconnections=0 -daemon

*** maxconnections=0 não se conecta a Nós e não fica tentando baixar e validar blocos

🔐 Gerando endereços

bitcoin-cli -datadir="." createwallet "Carteira1"
bitcoin-cli -datadir="." -rpcwallet="Carteira1" getnewaddress "legacy" legacy
bitcoin-cli -datadir="." -rpcwallet="Carteira1" getnewaddress "p2sh" p2sh-segwit
bitcoin-cli -datadir="." -rpcwallet="Carteira1" getnewaddress "bech32" bech32
bitcoin-cli -datadir="." -rpcwallet="Carteira1" getnewaddress "taproor" bech32m

🔐 Endereços gerados:

📬 Endereço P2PKH (Legacy): msdWCF131PcG3Y1NYasvv1HN67YUYFgAkv
📬 Endereço P2SH-SegWit: 2N6t7XUbx5fa7jriFwAwP4vBXFrtSYDvA6i
📬 Endereço Bech32 (P2WPKH): tb1qlts6nnnahxspz543p0kwssj8y7j5xcpm96es39
📬 Endereço Taproot (Bech32m): tb1p9jyg5vv76jgv8845ypftusyjwk2r43…mnzrf44s6ajtv7

🔗 Vendo o Hash de um bloco:

bitcoin-cli -datadir="." getblockhash 50000
000000001aeae195809d120b5d66a39c83eb48792e068f8ea1fea19d84a4278a
bitcoin-cli -datadir="." getblockhash 100000
00000000009e2958c15ff9290d571bf9459e93b19765c6801ddeccadbb160a1e

Signet

🟣 O que é a Signet?
🧪 Rede de testes controlada do Bitcoin

🔐 Mineração coordenada por validadores autorizados

🪙 Moedas sem valor real, obtidas via faucets específicos

🧠 Ideal para testes previsíveis, demonstrações e automações

⚙️ Mantém regras reais de consenso, como a Testnet,
mas com ambiente mais estável e confiável

⚠️ Atenção:
🔑 Blocos aceitos só se forem assinados por entidades da Signet

🌍 A sincronização inicial mais leve que a Testnet

Signet

🟣 Tamanho da Signet (jul/2025)

- 📦 ~260.000 blocos
- 💽 ~15 GB de dados em disco

⚙️ Configuração do **bitcoin.conf** para rodar na Signet:

📄 Exemplo típico:

```
signet=1
rpcuser=teste
rpcpassword=teste
prune=2000
```

🧩 O que cada linha faz:

Linha	Função
signet=1	Ativa o modo Signet
rpcuser	Nome de usuário para autenticar comandos via bitcoin-cli
rpcpassword	Senha correspondente
prune=2000	**Limita o espaço em disco** usado pela blockchain a 2 GB

Signet

💻 Rodando o Nó

```
bitcoind -datadir="." -daemon
```

🔐 Gerando endereços

```
bitcoin-cli -datadir="." createwallet "Carteira1"
bitcoin-cli -datadir="." -rpcwallet="Carteira1" getnewaddress "legacy" legacy
bitcoin-cli -datadir="." -rpcwallet="Carteira1" getnewaddress "p2sh" p2sh-segwit
bitcoin-cli -datadir="." -rpcwallet="Carteira1" getnewaddress "bech32" bech32
bitcoin-cli -datadir="." -rpcwallet="Carteira1" getnewaddress "taproot" bech32m
```

🔐 Endereços gerados:

📬 Endereço P2PKH (Legacy): mzBk5ozTWa4b2TiVs1dpgL6k2LMGyLkJrA
📬 Endereço P2SH-SegWit: 2N7co8e32V9KanbPkbaL5M8EBwTUFtdkjHt
📬 Endereço Bech32 (P2WPKH): tb1qxp8c04rfktnsqqepwx47tkhespz8gz0e94v3k7
📬 Endereço Taproot (Bech32m): tb1pz4hlfktrld5c79y2fqulr6dkx9skfx…elcfq2er9lzs0pgty9

🔗 Vendo o Hash de um bloco:

```
bitcoin-cli -datadir="." getblockhash 1
```

00000086d6b2636cb2a392d45edc4ec544a10024d30141c9adf4bfd9de533b53

```
bitcoin-cli -datadir="." getblockhash 250000
```

00000002d38fc984fa25a057930af276c00a001428bd68b8216f826d580a382f

Regtest

⚙️ O que é a Regtest?

🧰 Rede de testes local e privada do Bitcoin

🧠 Criada e controlada totalmente pelo desenvolvedor
(não se conecta com outros nós da internet)

⛏️ Mineração instantânea com um simples comando
generatetoaddress gera blocos sob demanda

📦 Não há sincronização de blocos (começa do bloco gênesis)

💻 Ideal para:

📜 Scripts automatizados

🧪 Testes rápidos e isolados

🏠 Desenvolvimento local

Regtest

📏 Tamanho da Regtest

📦 Começa com 0 blocos

💽 Consome espaço apenas conforme você minera

⚙️ Configuração do **bitcoin.conf** para rodar na Signet:

📄 Exemplo típico:

```
regtest=1
rpcuser=teste
rpcpassword=teste
```

🧩 O que cada linha faz:

Linha	Função
regtest=1	Ativa o modo Regtest
rpcuser	Nome de usuário para autenticar comandos via bitcoin-cli
rpcpassword	Senha correspondente

Regtest

💻 Rodando o Nó

```
bitcoind -datadir="." -daemon
```

🔐 Gerando endereços

```
bitcoin-cli -datadir="." createwallet "Carteira1"
bitcoin-cli -datadir="." -rpcwallet="Carteira1" getnewaddress "legacy" legacy
bitcoin-cli -datadir="." -rpcwallet="Carteira1" getnewaddress "p2sh" p2sh-segwit
bitcoin-cli -datadir="." -rpcwallet="Carteira1" getnewaddress "bech32" bech32
bitcoin-cli -datadir="." -rpcwallet="Carteira1" getnewaddress "taproot" bech32m
```

🔐 Endereços gerados:

📬 Endereço P2PKH (Legacy): mzqD7NKN4yGgN47FtU4QiCRhBCP3m4Ng9j
📬 Endereço P2SH-SegWit: 2MuZ2gcCsJQpm4vMDmWqziZnoFrfcuc1x7Q
📬 Endereço Bech32 (P2WPKH): bcrt1qwtxd64jx0sxee6pp09h0ygrcmfaca2x3xsj95s
📬 Endereço Taproot (Bech32m): bcrt1pgy2lc3clwv25xu5qhgr3…9vq2cct5vsheyvvs6x0v7p

🔗 Vendo o Hash de um bloco:

```
bitcoin-cli -datadir="." getblockhash 5
```

7442594c888e8cc02cda452423b469ff4d64842f7e9ae93ede38f5d0f2fec725

Regtest

⛏️ Minerando em Regtest

```
bitcoin-cli -datadir="." getblockchaininfo

bitcoin-cli -datadir="." generatetoaddress 100
bcrt1qwtxd64jx0sxee6pp09h0ygrcmfaca2x3xsj95s

bitcoin-cli -datadir="." getblockchaininfo

bitcoin-cli -datadir="." listunspent

bitcoin-cli -datadir="." generatetoaddress 1
bcrt1qwtxd64jx0sxee6pp09h0ygrcmfaca2x3xsj95s

bitcoin-cli -datadir="." listunspent

bitcoin-cli -datadir="." generatetoaddress 5 mzqD7NKN4yGgN47FtU4QiCRhBCP3m4Ng9j

bitcoin-cli -datadir="." listunspent
```

Obrigado!

“No início, o Bitcoin é só uma curiosidade. Com o tempo, vira estudo, depois reserva de valor… até que tudo o que você mais quer é trabalhar com isso, viver isso, fazer do Bitcoin seu caminho.”

Contato:
rapennas@gmail.com