**Deploying a smart contract on testnet (Sepolia)**

**Deploying our Contract to Testnet or Live Network with Foundry and Alchemy**

Hi, everyone! Are you curious about what your contract would look like on a testnet or a live network? If so, buckle up because this blog post will cover exactly that! We'll walk through the process of updating our Environment Variable (.env) file for an actual testnet.

Clearly, we need an actual testnet for a real network. But our trusty MetaMask has built-in Infura connections that are incompatible. Why? Because they're tailored specifically for MetaMask. Hence, we need our own Remote Procedure Call (RPC) URL.

**Creating our Own RPC URL for a Testnet**

*To create one, we could run our own blockchain node, but let's be honest — many folks prefer avoiding that route. Instead, we utilize Node as a Service (NaaS) applications to expedite the process.*

One promising option is using Alchemy - a free NaaS platform that we can send the transactions to. This procedure resides within the *Deploying to Testnet or Mainnnet* section in the full course repo of the Foundry.

To access the Alchemy platform, we simply click on the aforementioned function. On the platform, we sign up (I used Google sign-in for this demo).

Our next step is creating a new app in the Alchemy user interface. I named mine *Sepolia Testing* and kept the description the same, given that our chain will be an Ethereum one based on Ethiopia.

We can bypass advanced features for now and finalize our app. Now we have the app details needed for our node, including frequency of calls and other details. We also have a new https endpoint by clicking view key, which functions exactly the same way as our ganache or MetaMask endpoint.

**Altering our Private Key**

Next, let's do something about our private keys. Our ganache private key will no longer cut it — it has neither real money nor any testnet ETH in it.

Our solution is to use one of our MetaMask private keys. To do this, we switch back to Sepolia in our MetaMask, choose an account with money in it, click on account details, and export the private key. *Remember, never share your real private key!*

Upon confirmation with your password, copy the private key and omit the line in the env file — hashtag or pound sign denoting comments.

**Executing the Transaction**

With our Sepolia RPC URL and private key from MetaMask, executing a transaction now becomes tremendously easier.

source .env

forge script script/deploySimpleStorage.s.sol --rpc-url=$Sepolia\_RPC\_URL --private-key=$PRIVATE\_KEY –broadcast

This command deploys our contract to the testnet, and we can monitor the transaction on our Alchemy dashboard.

We soon find that our contract, Simple Storage, has been deployed on the Sepolia chain. We can grab our transaction hash and input it into Sepolia etherscan IO to confirm the successful transaction.

After we refresh our Alchemy dashboard, we'll verify the requests sent and track the ETH send raw transaction that transmitted our transaction to the blockchain.

So, this is how we deploy our contract on a real testnet leveraging Foundry and Alchemy!

Our next step will explore adding real-world components to the mix. Stay tuned!