Solidity Scripting:

Introduction
Solidity scripting enables declarative contract deployment using Solidity, offering a more user-friendly alternative to forge create. Unlike JavaScript-based scripts used with tools like Hardhat, Solidity scripts run on Foundry's EVM backend, providing dry-run capabilities.

High-Level Overview
`forge script` operates asynchronously and can be divided into four phases:

1. **Local Simulation:** The contract script runs in a local EVM. If an RPC/fork URL is provided, it executes in that context. External calls from `vm.broadcast` and `vm.startBroadcast` are appended to a list.
2. **Onchain Simulation:** Optional. With an RPC/fork URL, collected transactions are sequentially executed.
3. **Broadcasting:** Optional. With the `--broadcast` flag, collected and simulated transactions are broadcast.
4. **Verification:** Optional. With the `--verify` flag and an API key, the contract is verified (e.g., on Etherscan).

Set Up

Creating a New Project
1. Initialize a new Foundry project:
   ```
   forge init solidity-scripting
   ```
2. Install Solmate and OpenZeppelin contracts:
   ```
   cd solidity-scripting
   forge install transmissions11/solmate Openzeppelin/openzeppelin-contracts@v5.0.1
   ```
3. Delete `Counter.sol` and create `NFT.sol`:
   ```
   rm src/Counter.sol test/Counter.t.sol && touch src/NFT.sol && ls src
   ```

NFT Contract Code
Add the following code to `NFT.sol`:

```solidity
// SPDX-License-Identifier: UNLICENSED
pragma solidity >=0.8.10;

import "solmate/tokens/ERC721.sol";
import "openzeppelin-contracts/contracts/utils/Strings.sol";
import "openzeppelin-contracts/contracts/access/Ownable.sol";

error MintPriceNotPaid();
error MaxSupply();
error NonExistentTokenURI();
error WithdrawTransfer();

contract NFT is ERC721, Ownable {

    using Strings for uint256;
    string public baseURI;
    uint256 public currentTokenId;
    uint256 public constant TOTAL_SUPPLY = 10_000;
    uint256 public constant MINT_PRICE = 0.08 ether;
```

```
    constructor(
        string memory _name,
        string memory _symbol,
        string memory _baseURI
    ) ERC721(_name, _symbol) Ownable(msg.sender) {
        baseURI = _baseURI;
    }

    function mintTo(address recipient) public payable returns (uint256) {
        if (msg.value != MINT_PRICE) {
            revert MintPriceNotPaid();
        }
        uint256 newTokenId = ++currentTokenId;
        if (newTokenId > TOTAL_SUPPLY) {
            revert MaxSupply();
        }
        _safeMint(recipient, newTokenId);
        return newTokenId;
    }

    function tokenURI(uint256 tokenId)
        public
        view
        virtual
        override
        returns (string memory)
    {
        if (ownerOf(tokenId) == address(0)) {
            revert NonExistentTokenURI();
        }
        return
            bytes(baseURI).length > 0
                ? string(abi.encodePacked(baseURI, tokenId.toString()))
                : "";
    }

    function withdrawPayments(address payable payee) external onlyOwner {
        uint256 balance = address(this).balance;
        (bool transferTx, ) = payee.call{value: balance}("");
        if (!transferTx) {
            revert WithdrawTransfer();
        }
    }
}
```

Compile Contract
Compile the contract to ensure everything is correct:
```
forge build
```

Deploying the Contract

Environment Configuration
Create a `.env` file and add the following variables:
```
SEPOLIA_RPC_URL=
PRIVATE_KEY=
```

ETHERSCAN_API_KEY=
```

#### Update `foundry.toml`
Add the following lines to `foundry.toml`:
```toml
[rpc_endpoints]
sepolia = "${SEPOLIA_RPC_URL}"

[etherscan]
sepolia = { key = "${ETHERSCAN_API_KEY}" }
```

Writing the Deployment Script
Create `script/NFT.s.sol` with the following content:
```solidity
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

import "forge-std/Script.sol";
import "../src/NFT.sol";

contract MyScript is Script {
    function run() external {
        uint256 deployerPrivateKey = vm.envUint("PRIVATE_KEY");
        vm.startBroadcast(deployerPrivateKey);

        NFT nft = new NFT("NFT_tutorial", "TUT", "baseUri");

        vm.stopBroadcast();
    }
}
```

Running the Deployment Script
Make sure the `.env` variables are loaded:
```
source .env
```

Deploy and verify the contract:
```
forge script --chain sepolia script/NFT.s.sol:MyScript --rpc-url $SEPOLIA_RPC_URL --broadcast --
verify -vvvv
```

Deploying Locally

Using Anvil’s Default Accounts
Start Anvil:
```
anvil
```
Update your `.env` file with an Anvil-provided private key and run:
```
forge script script/NFT.s.sol:MyScript --fork-url http://localhost:8545 --broadcast
```

Using a Custom Mnemonic
Add your mnemonic to `.env`:
```

```
MNEMONIC=
```
Start Anvil with the mnemonic:
```
source .env
anvil -m $MNEMONIC
```
Run the script:
```
forge script script/NFT.s.sol:MyScript --fork-url http://localhost:8545 --broadcast
```

These notes guide you through Solidity scripting, from setup to deployment on testnets and local environments.