5CFE9D;}

.css-kun0x7{fill:transparent;opacity:0.5;margin:0 0.2rem;}.css-kun0x7:hover{fill:#FAF40A;}

.css-1ix0nx7{fill:transparent;opacity:0.5;}.css-1ix0nx7:hover{fill:#F14544;} On this page

Deployment

Deploying Uniswap V4 Hooks involves several steps:

1. Deploying the PoolManager Contract
2. : This contract is typically pre-deployed on many test environments. However,
3. you have the option to deploy it locally on your machine if required.
4. Deploying the Hook Contract
5. : The hook contract needs to be deployed at a predetermined address. You can useCREATE2
6. for deterministic deployment. A Deterministic Deployment Proxy, usually found
7. at0x4e59b44847b379578588920cA78FbF26c0B4956C
8. , is employed for this purpose and is already available in most
9. environments.
10. Deploying Test Tokens
11. : These tokens are essential for creating the pool. They need to be deployed before
12. initializing the pool.
13. Initializing the Pool with the Hook Contract Address
14. : This is achieved by invoking
15. theinitialize(PoolKey memory key, uint160 sqrtPriceX96, bytes calldata hookData)
16. function on the PoolManager contract.
17. Adding Liquidity or Modifying Position
18. : If you wish to add liquidity to the pool or alter its position, a
19. utility contract that implements theILockCallback
20. interface is necessary. You may consider deploying a utility
21. contract likePoolModifyPositionTest
22. for these operations.

Deployment Scripts

The template includes a few scripts that help with deploying hooks. These scripts are located in thescripts folder.

Lets look at these scripts one by one:

1. Deploying Your Own Tokens

The template includes Mock UNI and Mock USDC contracts for testing. Deploy them using:

forge create script/mocks/mUNI.sol:MockUNI \ --rpc-url [your_rpc_url_here]

\ --private-key [your_private_key_on_goerli_here]

\

forge create script/mocks/mUSDC.sol:MockUSDC \ --rpc-url [your_rpc_url_here]

\ --private-key [your_private_key_on_goerli_here]

\ Copy

2. script/01_CreatePool.s.sol

This script contains the steps for initializing the pool with an existing hook. It uses the pre-deployed PoolManager contract and token contracts

contract

CreatePoolScript

is Script { using

CurrencyLibrary

for Currency ;

//addresses with contracts deployed address

constant GOERLI_POOLMANAGER =

address (0x3A9D48AB9751398BbFa63ad67599Bb04e4BdF98b) ;

//pool manager deployed to GOERLI address

constant MUNI_ADDRESS =

address (0xbD97BF168FA913607b996fab823F88610DCF7737) ;

//mUNI deployed to GOERLI -- insert your own contract address here address

constant MUSDC_ADDRESS =

address (0xa468864e673a807572598AB6208E49323484c6bF) ;

//mUSDC deployed to GOERLI -- insert your own contract address here address

constant HOOK_ADDRESS =

address (0x3CA2cD9f71104a6e1b67822454c725FcaeE35fF6) ;

//address of the hook contract deployed to goerli -- you can use this hook address or deploy your own!

IPoolManager manager

IPoolManager (GOERLI_POOLMANAGER) ;

function

run ( )

external

{ // sort the tokens! address token0 =

uint160 (MUSDC_ADDRESS)

<

uint160 (MUNI_ADDRESS)

? MUSDC_ADDRESS : MUNI_ADDRESS ; address token1 =

uint160 (MUSDC_ADDRESS)

<

uint160 (MUNI_ADDRESS)

? MUNI_ADDRESS : MUSDC_ADDRESS ; uint24 swapFee =

4000 ; int24 tickSpacing =

10 ;

// floor(sqrt(1) * 2^96) uint160 startingPrice =

79228162514264337593543950336 ;

bytes

memory hookData = abi . encode (block . timestamp) ;

PoolKey memory pool =

PoolKey ({ currency0 : Currency . wrap (token0) , currency1 : Currency . wrap (token1) , fee : swapFee , tickSpacing : tickSpacing , hooks :

IHooks (HOOK_ADDRESS) }) ;

// Turn the Pool into an ID so you can use it for modifying positions, swapping, etc. PoolId id = PoolIdLibrary . toId (pool) ; bytes32 idBytes = PoolId . unwrap (id) ;

console . log ("Pool ID Below") ; console . logBytes32 (bytes32 (idBytes)) ;

vm . broadcast ( ) ; manager . initialize (pool , startingPrice , hookData) ; } } Copy

3. script/00_Counter.s.sol

This script deploys the Counter hook using Deterministic Deployment Proxy. It uses the pre-deployed PoolManager contract and proxy

contract

CounterScript

is Script { address

constant CREATE2_DEPLOYER =

address (0x4e59b44847b379578588920cA78FbF26c0B4956C) ; address

constant GOERLI_POOLMANAGER =

address (0x3A9D48AB9751398BbFa63ad67599Bb04e4BdF98b) ;

function

setUp ( )

public

{ }

function

run ( )

public

{ // hook contracts must have specific flags encoded in the address uint160 flags =

uint160 (Hooks . BEFORE_SWAP_FLAG | Hooks . AFTER_SWAP_FLAG | Hooks . BEFORE_ADD_LIQUIDITY_FLAG | Hooks . BEFORE_REMOVE_LIQUIDITY_FLAG) ;

// Mine a salt that will produce a hook address with the correct flags (address hookAddress ,

bytes32 salt)

= HookMiner . find (CREATE2_DEPLOYER , flags ,

type (Counter) . creationCode , abi . encode (address (GOERLI_POOLMANAGER))) ;

// Deploy the hook using CREATE2 vm . broadcast ( ) ; Counter counter =

new

Counter { salt : salt } (IPoolManager (address (GOERLI_POOLMANAGER))) ; require (address (counter)

== hookAddress ,

"CounterScript: hook address mismatch") ; } } CopyEdit this page .css-1tclyyl{margin-top:1.5rem;} .css-1c3fvx8{display:-webkit-box;display:-webkit-flex;display:-ms-flexbox;display:flex;-webkit-flex-direction:row;-ms-flex-direction:row;flex-direction:row;-webkit-align-items:center;-webkit-box-align:center;-ms-flex-align:center;align-items:center;-webkit-box-pack:center;-ms-flex-pack:center;-webkit-justify-content:center;justify-content:center;} .css-1wsnqg4{font-size:1rem;padding-right:0.5rem;} Helpful? .css-y2jwfw{fill:transparent;opacity:0.5;}.css-y2jwfw:hover{fill:#5CFE9D;}

.css-kun0x7{fill:transparent;opacity:0.5;margin:0 0.2rem;}.css-kun0x7:hover{fill:#FAF40A;}

.css-1ix0nx7{fill:transparent;opacity:0.5;}.css-1ix0nx7:hover{fill:#F14544;} Previous Testing Hooks Next Overview * Deployment Scripts * * 1. Deploying Your Own Tokens * * 2. script/01_CreatePool.s.sol * * 3. script/00_Counter.s.sol