

Deploying (WIP)

Now that we have our completed WrappingERC20 token, the next step is to see if our code actually works!

To do this, we'll be writing tests in typescript using hardhat, and deploying them on our [LocalFhenix](#) environment which we set up earlier

Note At this stage, using hardhat network is not supported, as Fhenix uses extensions to the EVM that enable FHE operations

Compiling the Contract

Set up your Mnemonic

Hardhat uses an .env with mnemonics to deploy and interact with contracts. Copy the .env.example file to .env and set your mnemonics (you can use the default mnemonics if you just want to use LocalFhenix)

```
cp .env.example .env
```

Compiling your contracts

First, let's see that our current contract is even valid. Let's run the following:

`pnpm compile` This will compile our Solidity contract into bytecode, and generate helper components that we'll be able to use for testing and deployment. If everything worked, you should see something like:

```
cross-env TS_NODE_TRANSPILE_ONLY=true hardhat compile
```

Generating typings for: 5 artifacts in dir: types for target: ethers-v6 Successfully generated 28 typings! Compiled 5 Solidity files successfully

Deploying the Contract

Tasks

To help us deploy and perform actions you can make use of tasks. We'll add deployment and usage tasks for our new contract. We'll replace the deployment of the default contract with our WrappedERC20. Notice that the differences are mostly just the naming and constructor arguments that are different. Replace the `deploy/deploy.ts` with the following content:

```
import
{ DeployFunction }
from
"hardhat-deploy/types" ; import
{ HardhatRuntimeEnvironment }
from
"hardhat/types" ;
const func :
DeployFunction
=
async
function
( hre : HardhatRuntimeEnvironment )
{ const
{ deployer }
=
```

```

await hre . getNamedAccounts ( ) ; const
{ deploy }
= hre . deployments ;
const werc20 =
await
deploy ( "WrappingERC20" ,
{ from : deployer , args :
[ "TestShieldedToken" ,
"TST" ] , log :
true , skipIfAlreadyDeployed :
false , } ) ;
console . log ( WrappingERC20 contract: , werc20 . address ) ; } ;
export
default func ; func . id =
"deploy_werc20" ; func . tags =
[ "WrappingERC20" ] ; Now we can use this task to deploy our contract to either LocalFhenix, or the Devnet chain.


- LocalFhenix
- Fhenix Frontier

```

get tokens from localfhenix faucet

pnpm faucet

if this doesn't work, try running it directly

with "node getFromFaucet"

deploy the contract

pnpm deploy:contracts Make sure your deployer address has some tokens, which you can get from []

pnpm deploy:contracts --network devnet Okay, now we know how to create programmatic actions. You can find a few other examples of tasks that interact with the deployed contract in the [tasks](#) folder.

Making Changes? When deploying a contract hardhat creates a static deployment. If you want to make changes and redeploy using this method runpnpm clean Let's move on to writing a few unit tests! [Edit this page](#)

[Previous Adding View Functions](#)