Connecting to Ethereum's Sepolia testnet and interacting with it.

Tools:

1. Sepolia: A test network for the Ethereum blockchain

2. MetaMask: Wallet

3. Remix Ethereum: Online Solidity compiler

Outline

- We show how to connect to Ethereum's Sepolia testnet and interact with it.

- Steps:

  1. Install MetaMask. Create an account (i.e. an address and public-private key) via MetaMask

  2. Request some Ether from a Sepolia faucet

  3. Get familiar with Solidity and the Remix compiler:

     - Write smart contracts, debug and compile them online

  4. Send/deploy the latest version of the contract to the blockchain and interact with the deployed contract
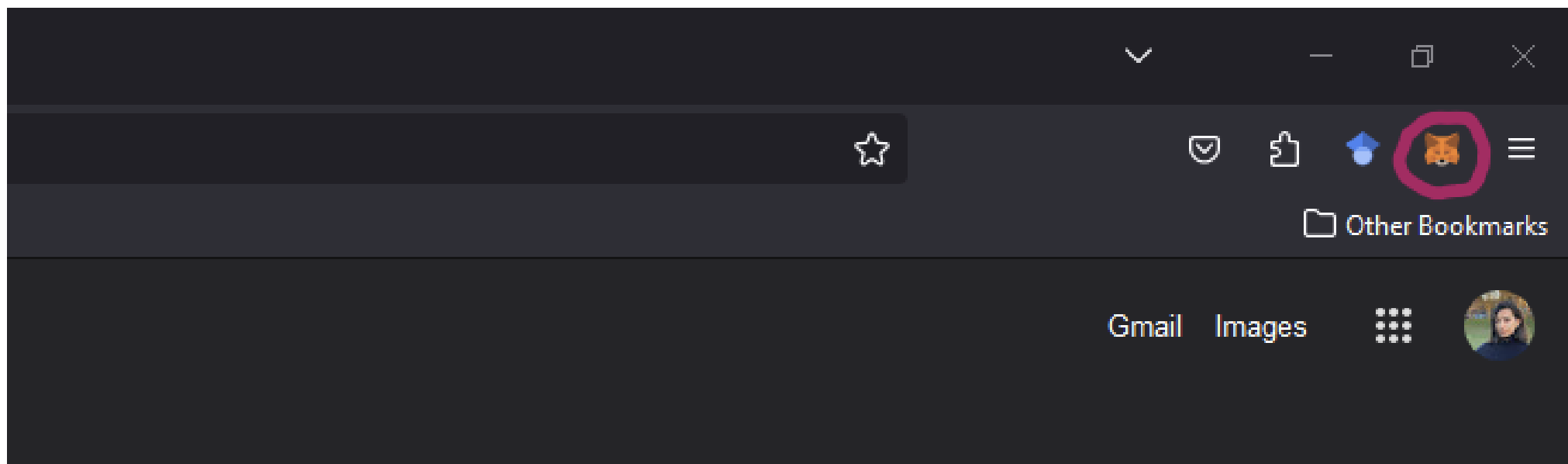
Step 1.1:
Install Metamask

- It is a browser extension for Ethereum wallets (works on Firefox, Google Chrome, Brave and Edge)

- Allows us to create our public/private keys and connect to the blockchain

- We recommend using MetaMask on Firefox

    - Download it from: https://metamask.io/

- Follow the instructions to install it, then refer to the next slides
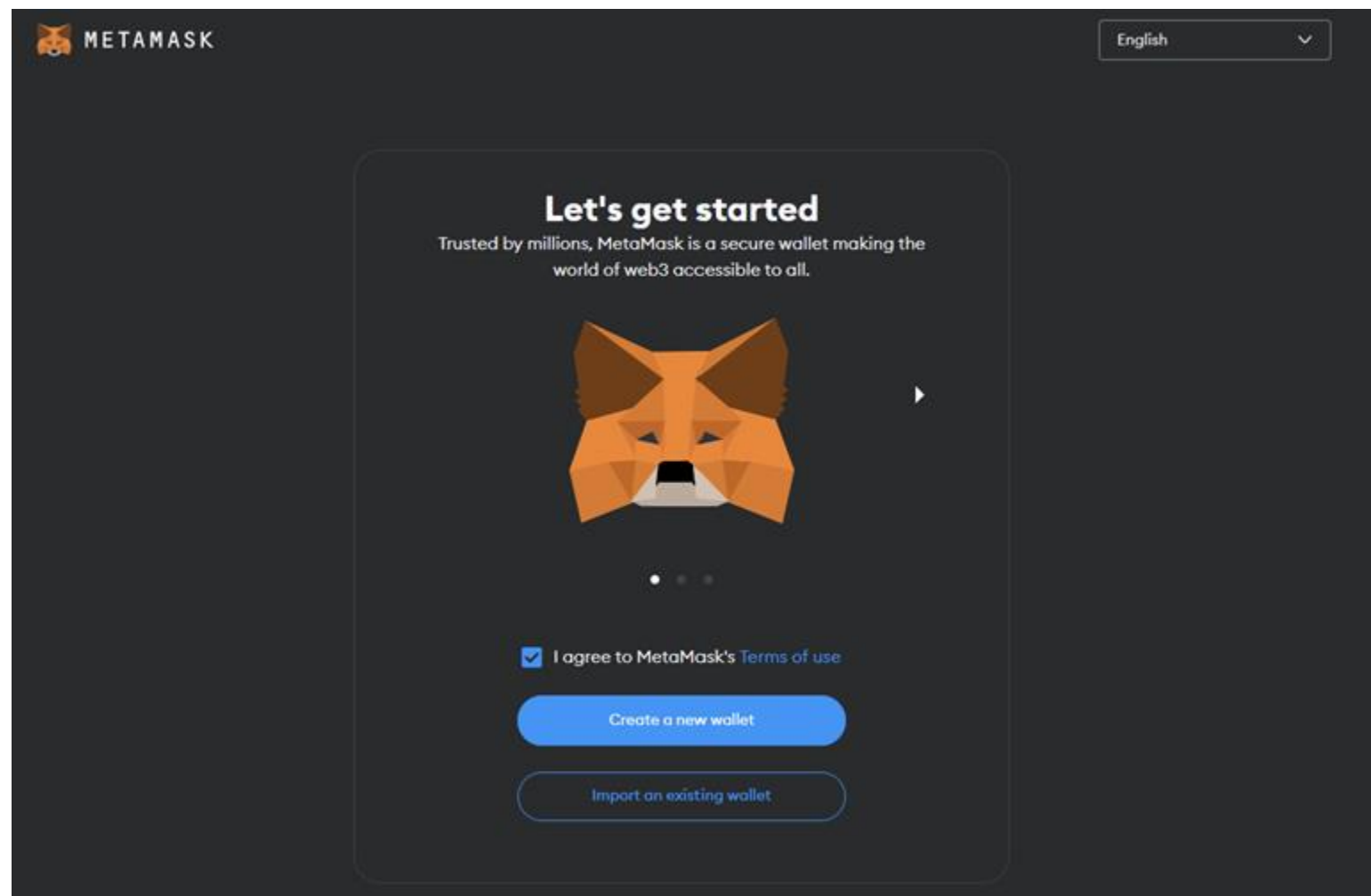
Step1.2:
Create an Account in MetaMask

- Click on the MetaMask icon on the top right side of your browser (or in the extensions folder).

Step1.2:
Create an Account in MetaMask

- Follow the instructions to create an account

Step1.2:
Create an Account in MetaMask

- After you provide a password, an account (i.e. an address, public and secret keys) will be created for you – you can also create more than one accounts per wallet

- Select "Secure my wallet" to generate a seed phrase

- **Store your seed**: you will need it to restore your wallet in case you delete Metamask

Step1.3:

Connect MetaMask to Sepolia Testnet

- Open the MetaMask extension
- Click on the Network dropdown on the top left corner
- Select "Show test networks"
- Select "Sepolia" from the list
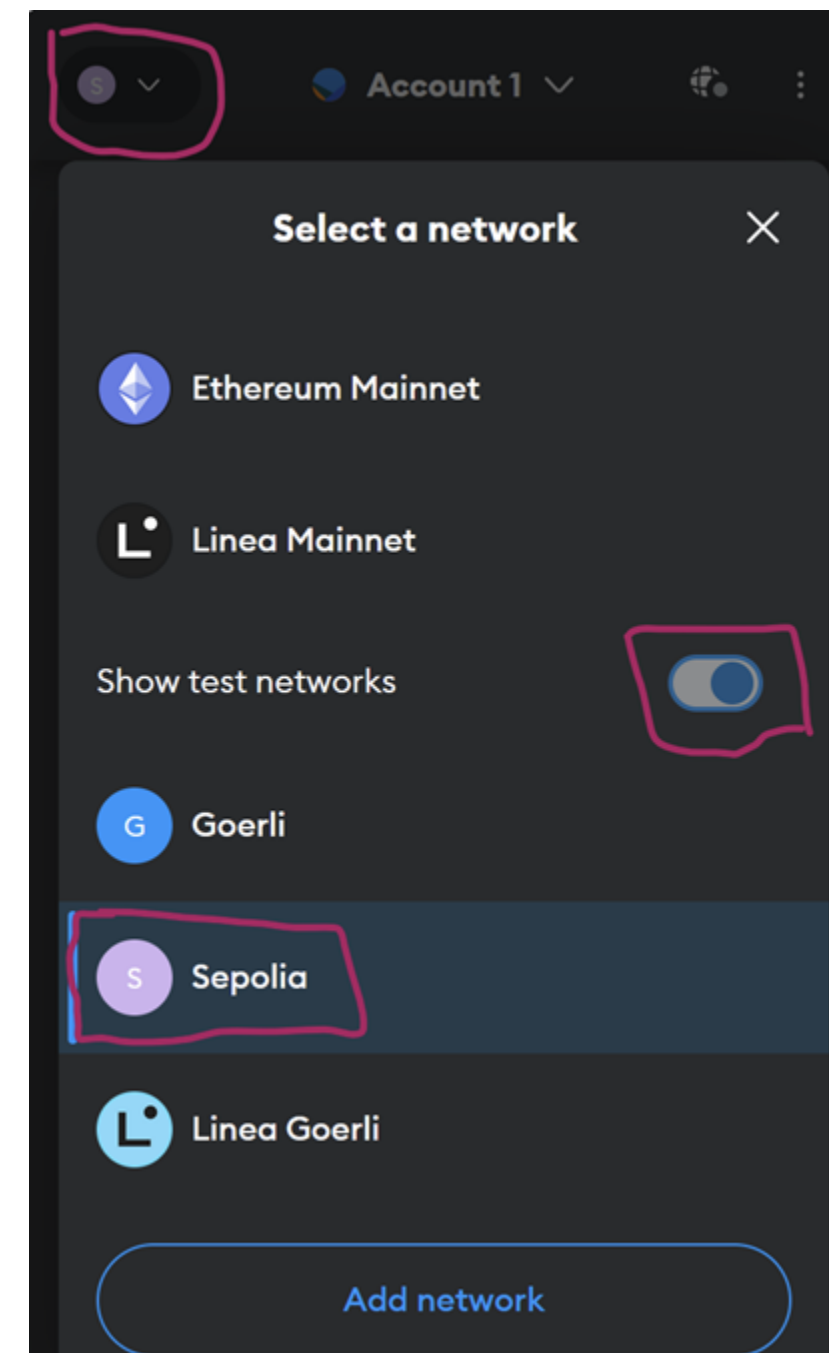
Step1.4:
Your Address

- When you have successfully connected to the chain, this page will appear

- Your address is under the account's name – here it starts with "0x117" and ends with "BC2B"

- Click on it to copy it and then send it to those who want to pay you
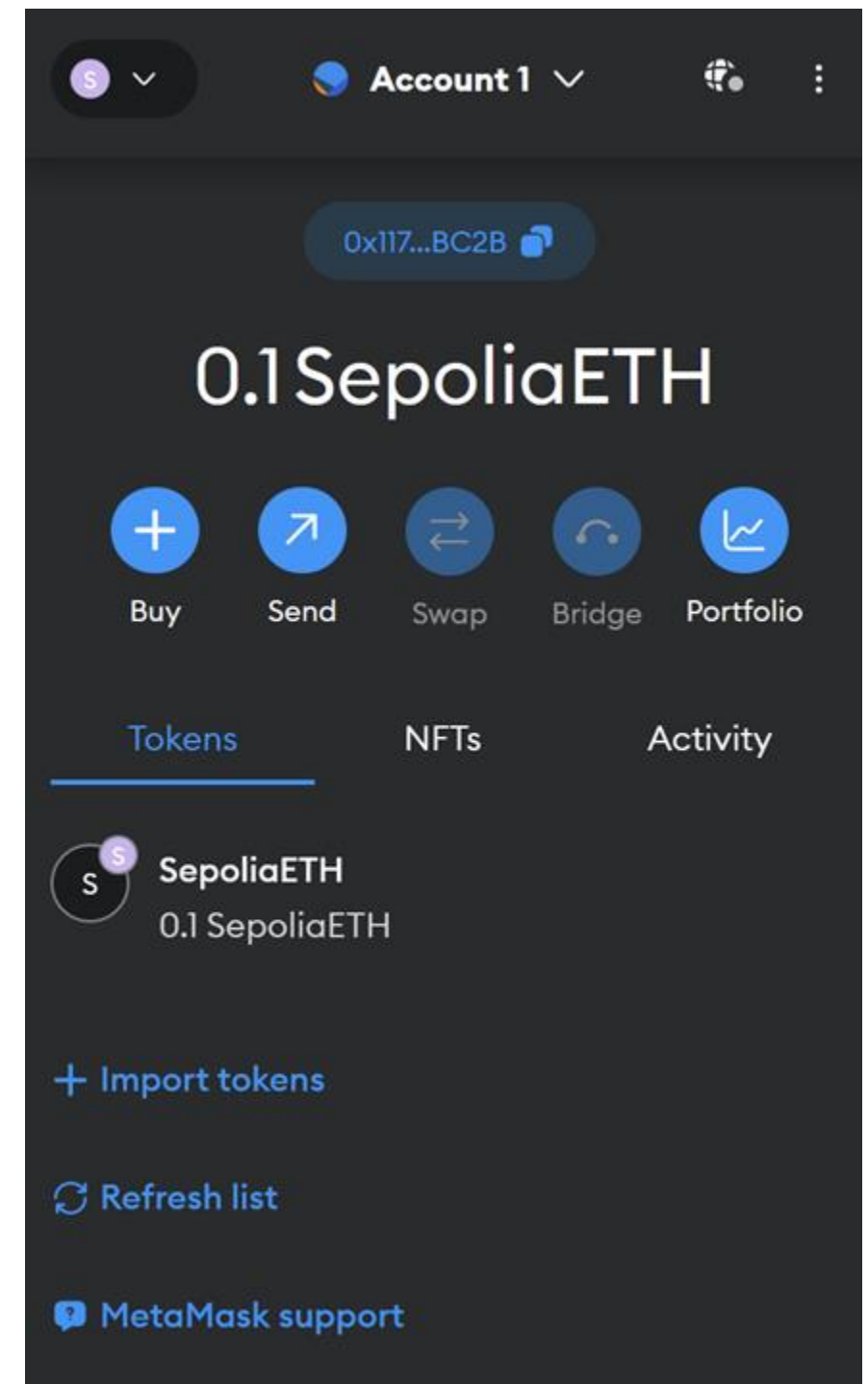
Step 2:
Request your Ether

- You need some (Sepolia) Ether to send a transaction and interact with a smart contract in the testnet

- Request some Ether from a Sepolia Faucet, e.g., from [here](#) by providing the address that you copied (for more faucet options see https://faucetlink.to/sepolia)

- Attention: most faucets give only small amounts of Ether every 24 hours, so you are advised to start accumulating your Ether early, to make sure you have sufficient funds to use during your coursework
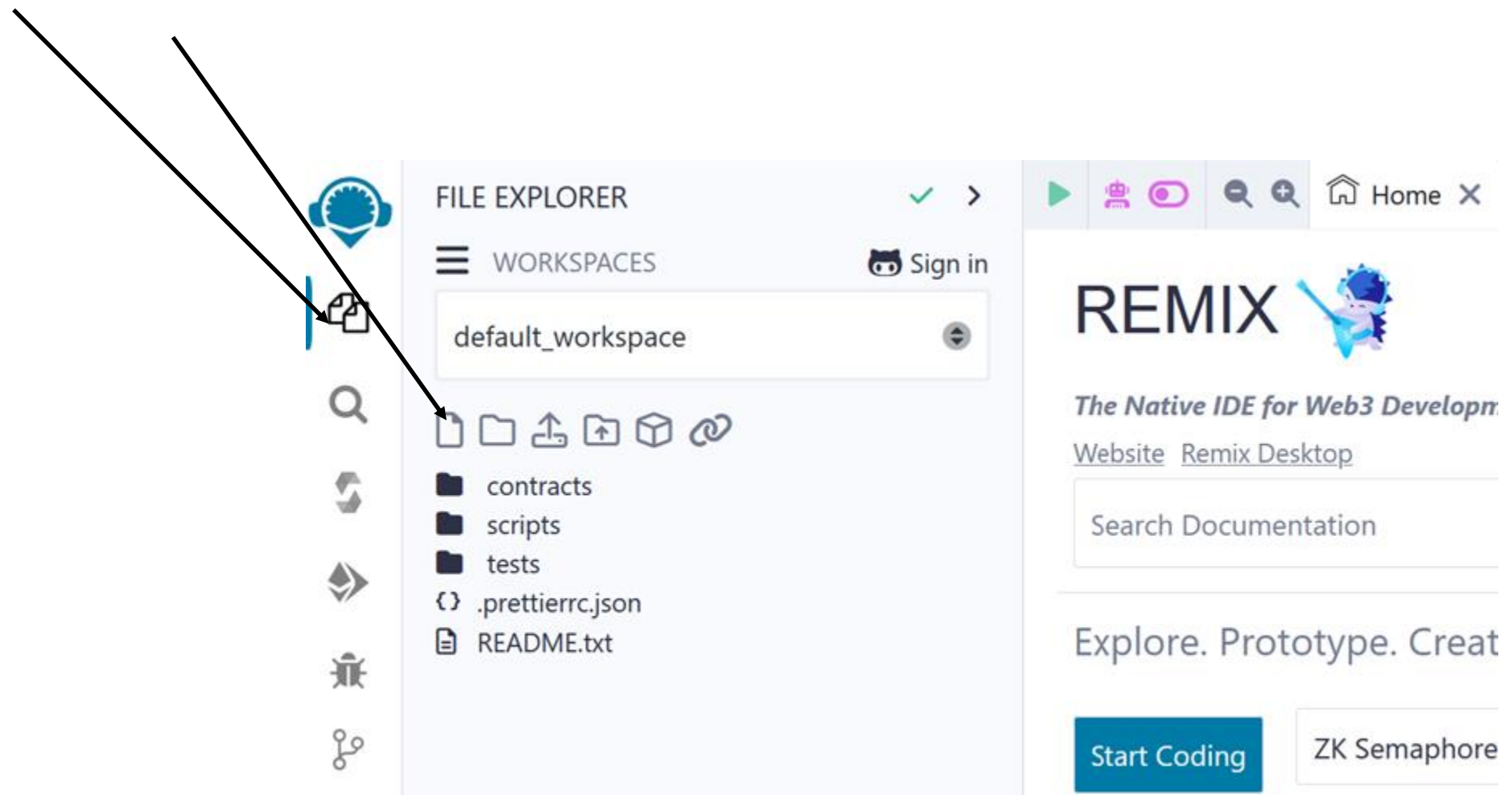
Step 3:
Getting Familiar with Remix Ethereum:
Online Solidity Compiler

- You can write, debug, deploy (i.e. send to a blockchain) your smart contract via remix Ethereum: https://remix.ethereum.org
- After you deploy your contract, you can also interact with it using Remix
- Before you deploy your smart contract to the test chain, run and debug it online

Step 3:
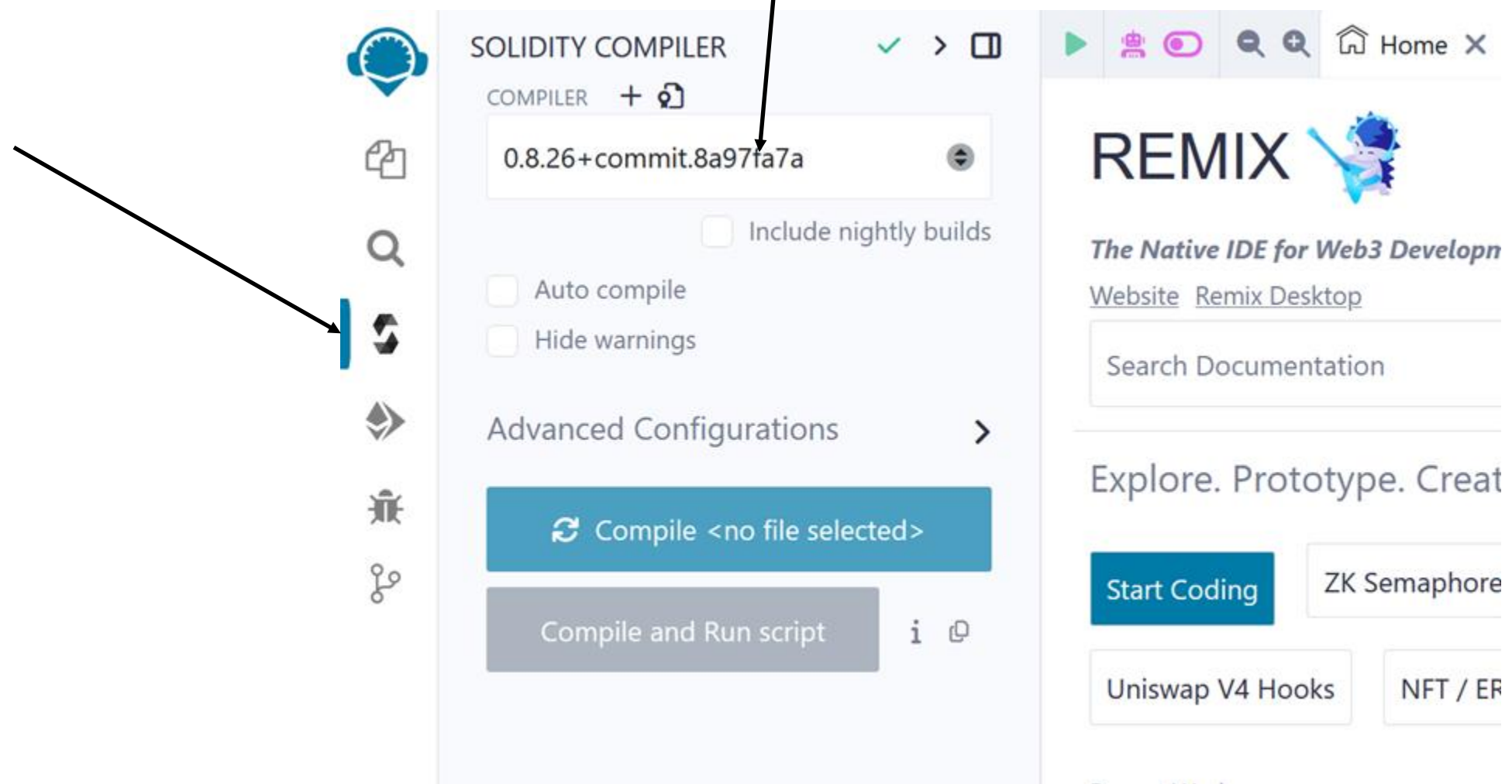# Getting familiar with Remix Ethereum:
Online Solidity Compiler

- To view your files and create a new, click on the file explorer, create a new file and write your Solidity code

Step 3:
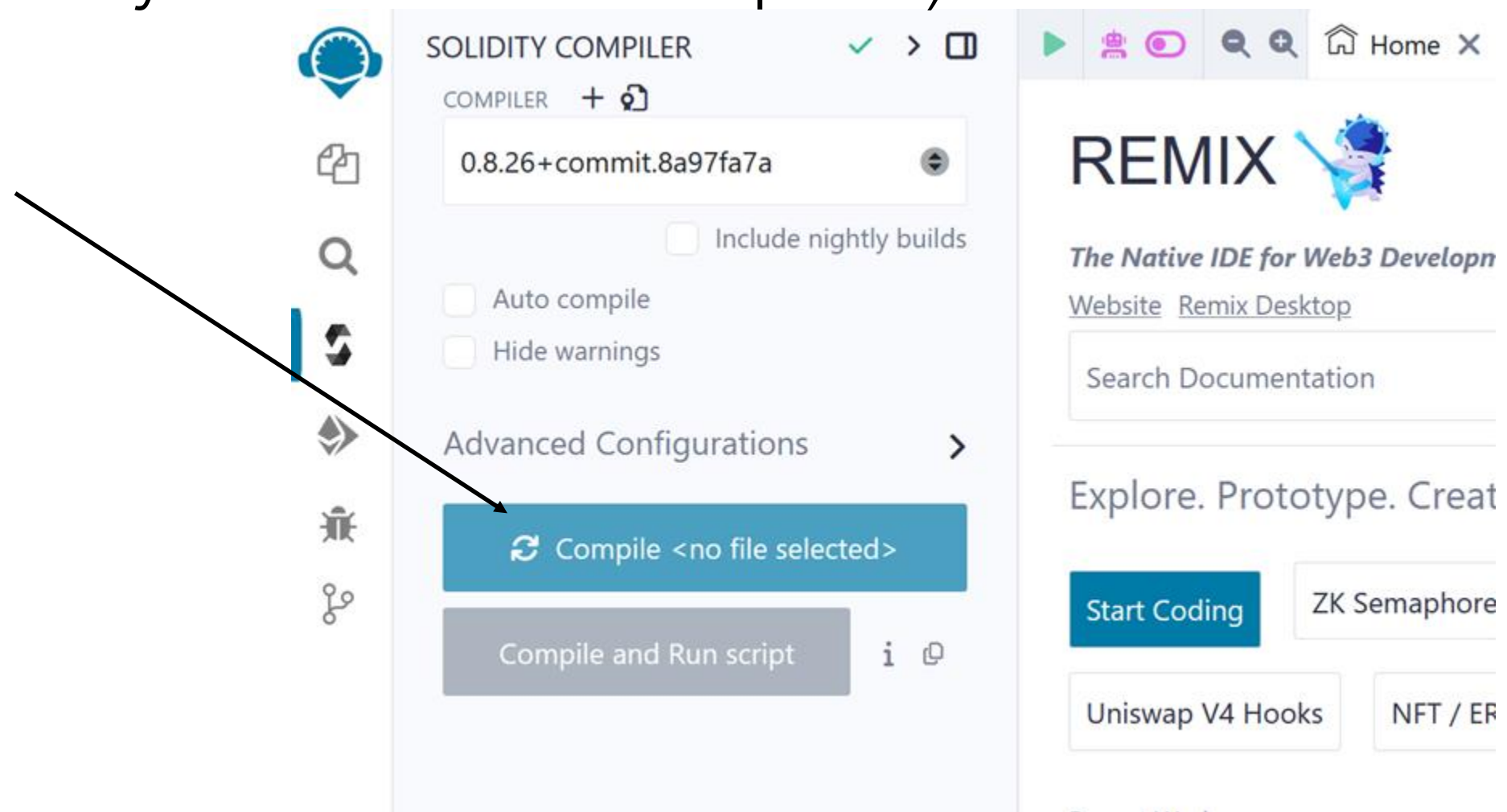# Getting familiar with Remix Ethereum: Online Solidity Compiler

- Next, click the compiler button and then choose the Solidity environment

Step 3:
Getting familiar with Remix Ethereum: Online Solidity Compiler
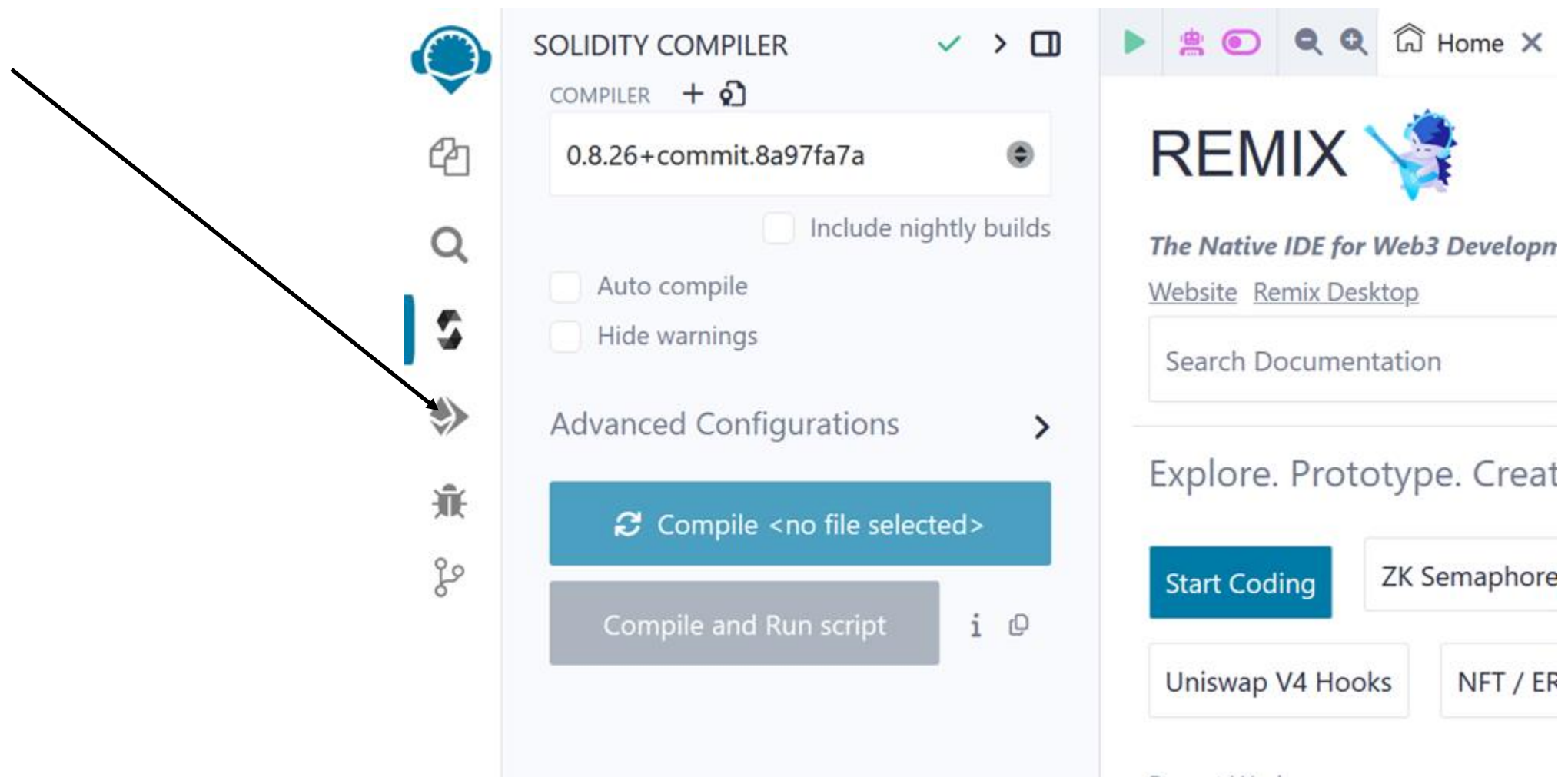
- Next, compile your smart contract (every time you change your contract you need to recompile it)

Step 3:
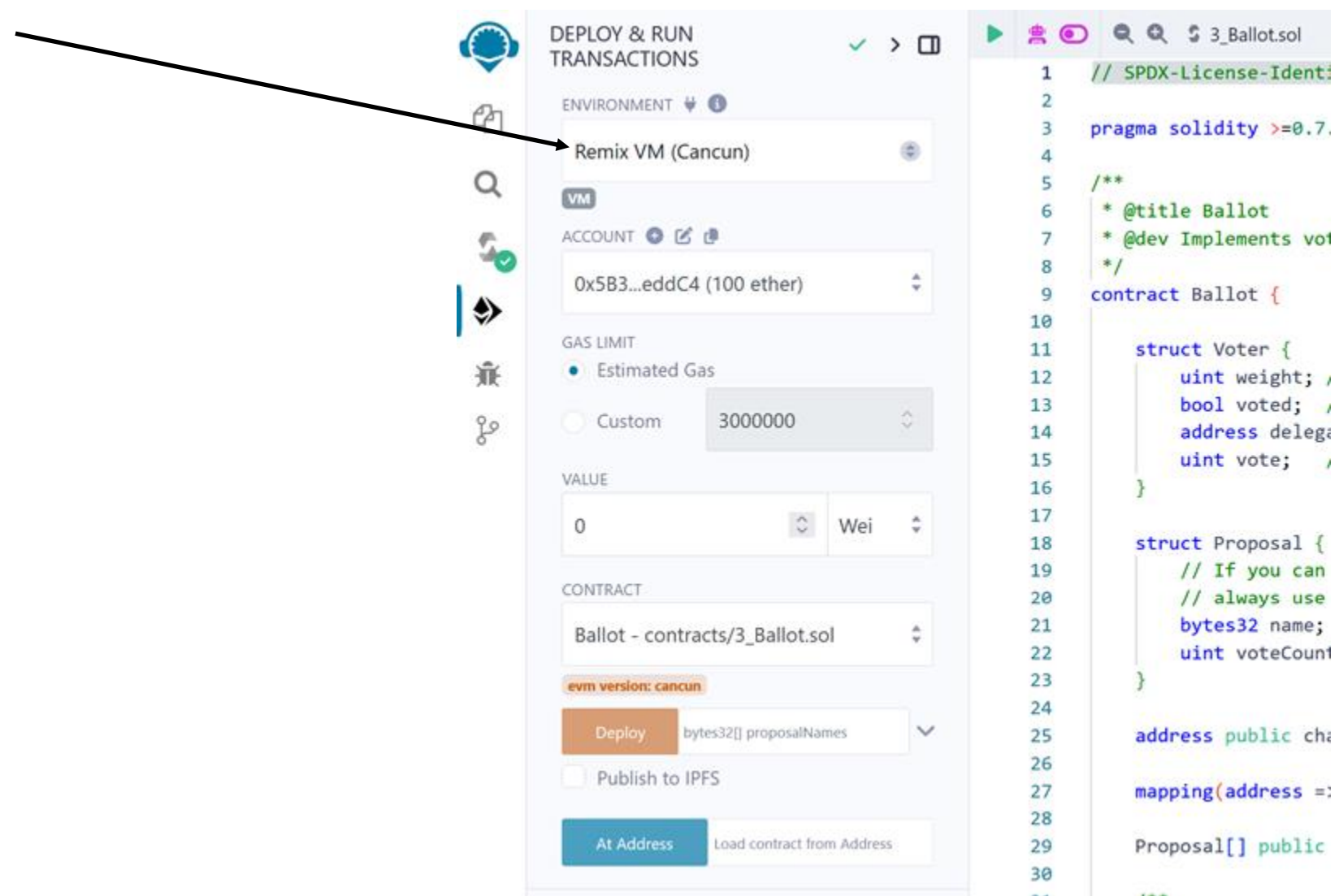# Getting familiar with Remix Ethereum: Online Solidity Compiler

- Next, to deploy and test your contract, choose the deployment explorer

Step 3:
Getting familiar with Remix Ethereum:
Online Solidity Compiler

- You can deploy your contract in a number of environments

Step 3:
Getting familiar with Remix:
Javascript VM environment

- **Remix VM (Cancun)**
  - This is a <u>testing</u> environment
  - It is a local environment that lives on your browser's tab
  - Whatever you do in this environment <u>does not</u> affect your funds, i.e. it does not have access to your wallet
  - When you close the browser it is deleted and when you open it again it is created fresh, so every time you use this environment you need to re-deploy your contracts

Step 3:
Getting familiar with Remix:
Injected Provider environment

- **Injected Provider - Metamask**
  - This environment has access to your Metamask
  - It <u>connects to the network</u> to which Metamask is connected and <u>uses the funds</u> of your wallet
  - Every time you try to use a contract, Metamask will request permission before completing the operation – this is because an actual transaction is posted and the actual funds in your wallet are used

Step 3:
Getting familiar with Remix Ethereum: Online Solidity Compiler

- To test your smart contract, click on Deploy
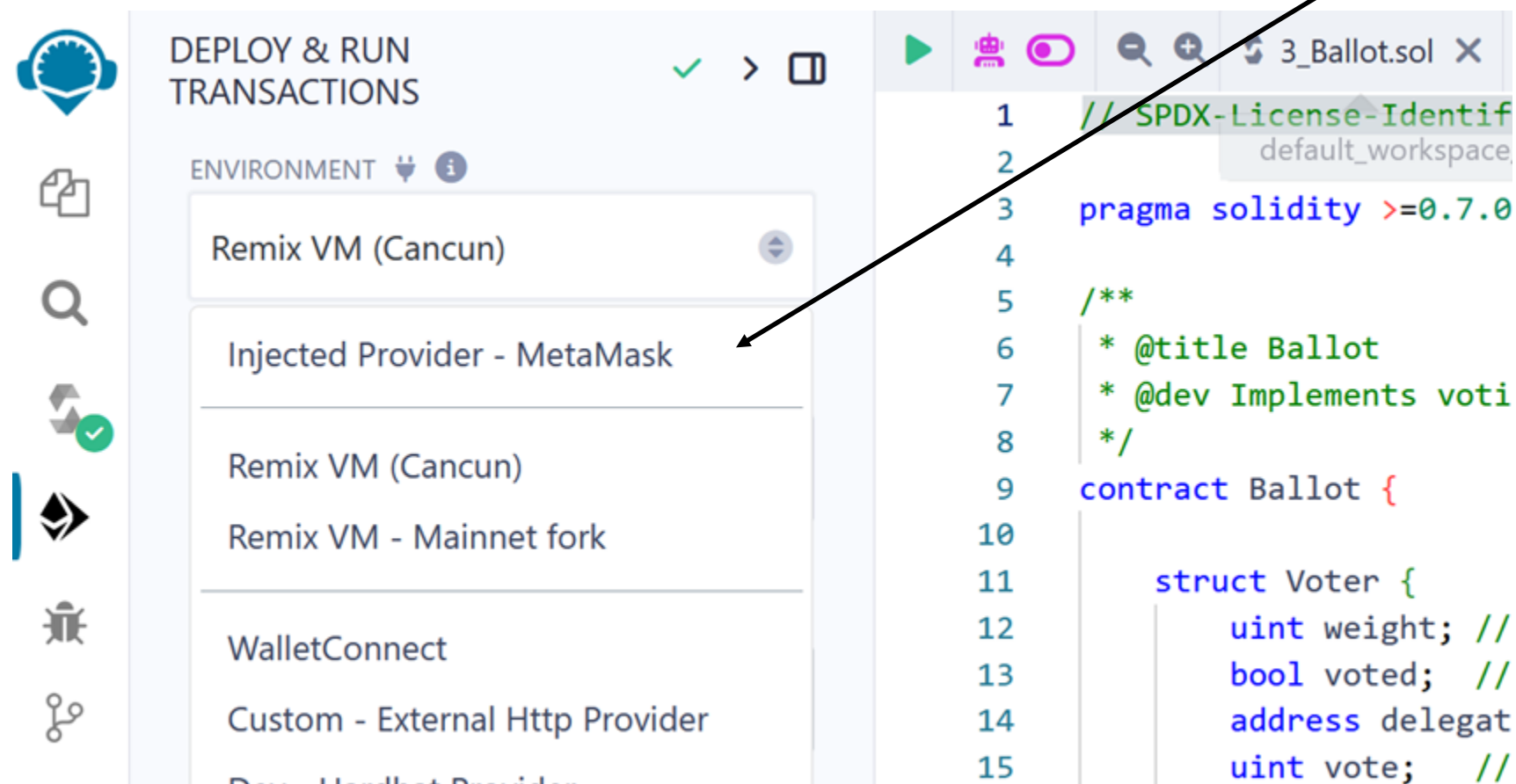- Remix creates a <u>user interface</u> to interact with the contract
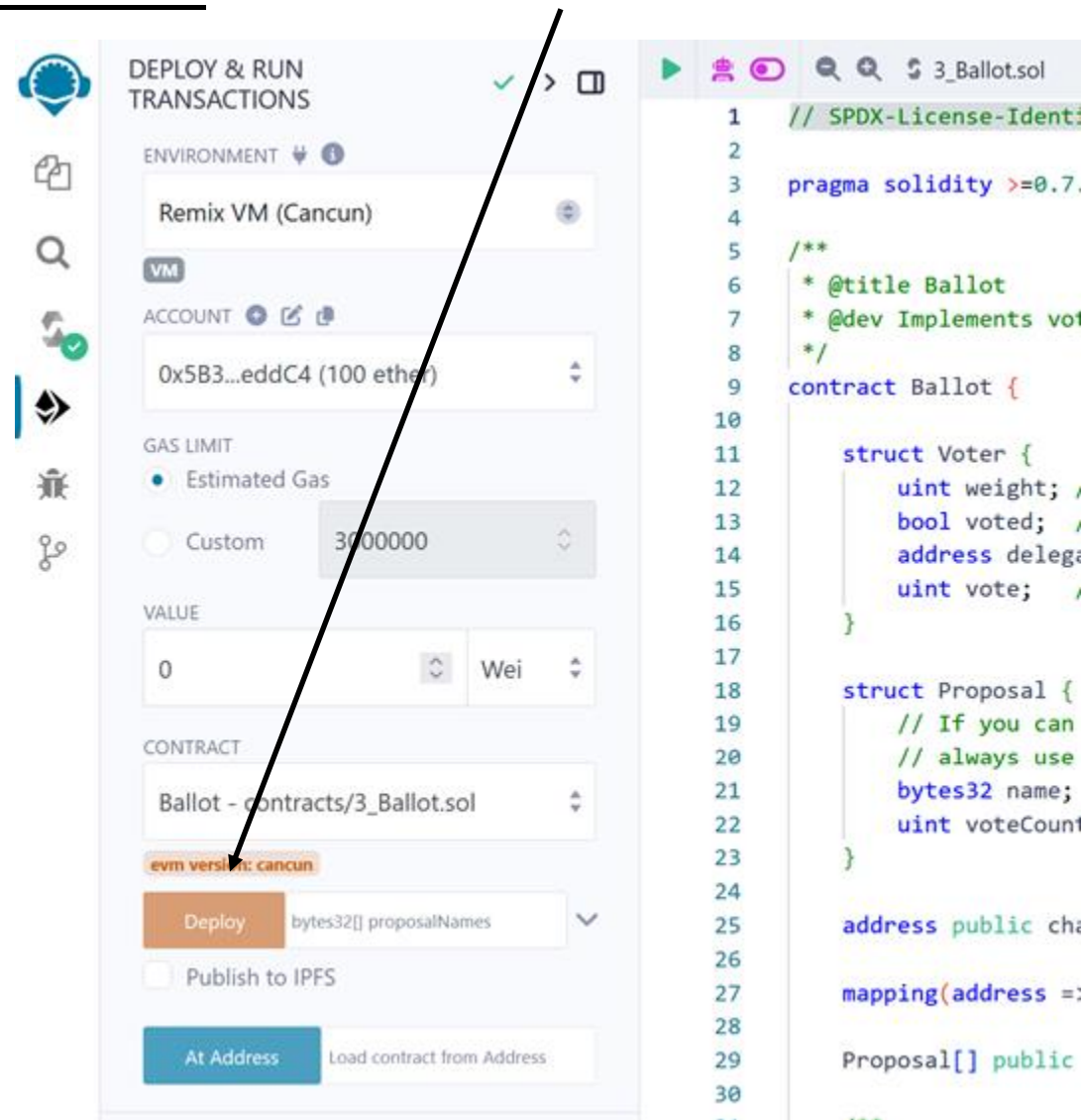
Step 4.1:
Deploying Smart Contract to the Sepolia testnet
Configurations

- First, you need to allow Remix to connect to Metamask
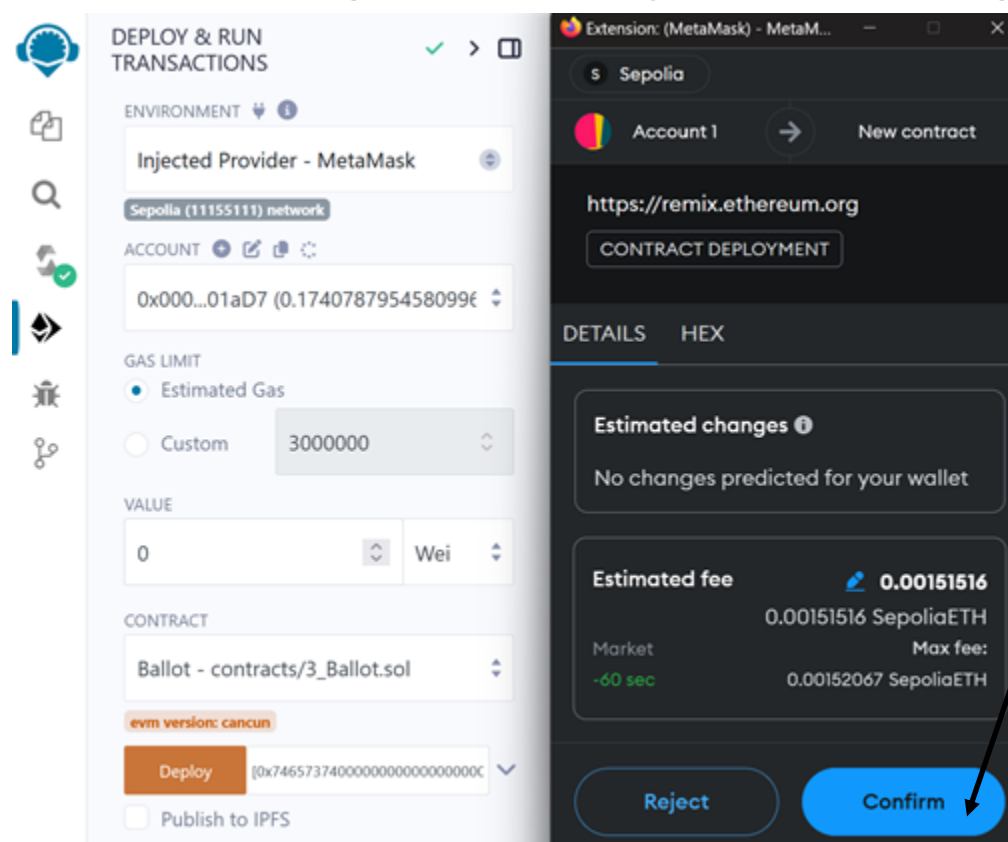- In Remix, set the environment to Injected Provider - Metamask.

Step 4.2:
Deploying Smart Contract to the Sepolia testnet
Deploying a Contract to the Blockchain

- Click on Deploy

- MetaMask will request your permission to send your contract to the Sepolia testnet - by clicking on confirm, you publish your contract (and pay the fee)
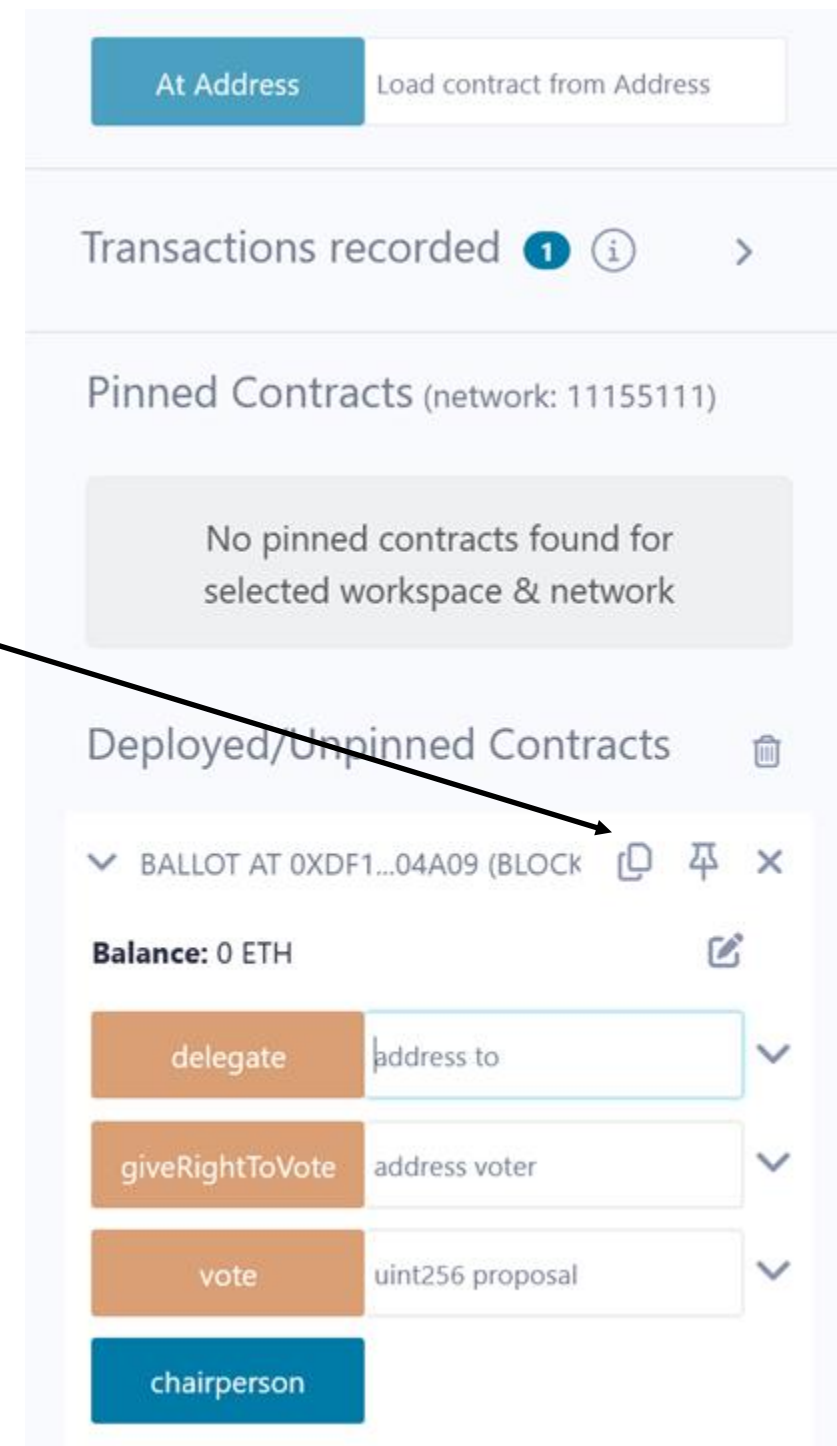
Step 4.3:
Deploying Smart Contract to the Sepolia testnet
Saving the Deployed Contract's Address

- When, your contract is successfully submitted & deployed, Remix provides the contract's address

- You need the contract's code and the address next time you want to interact with your deployed contract.
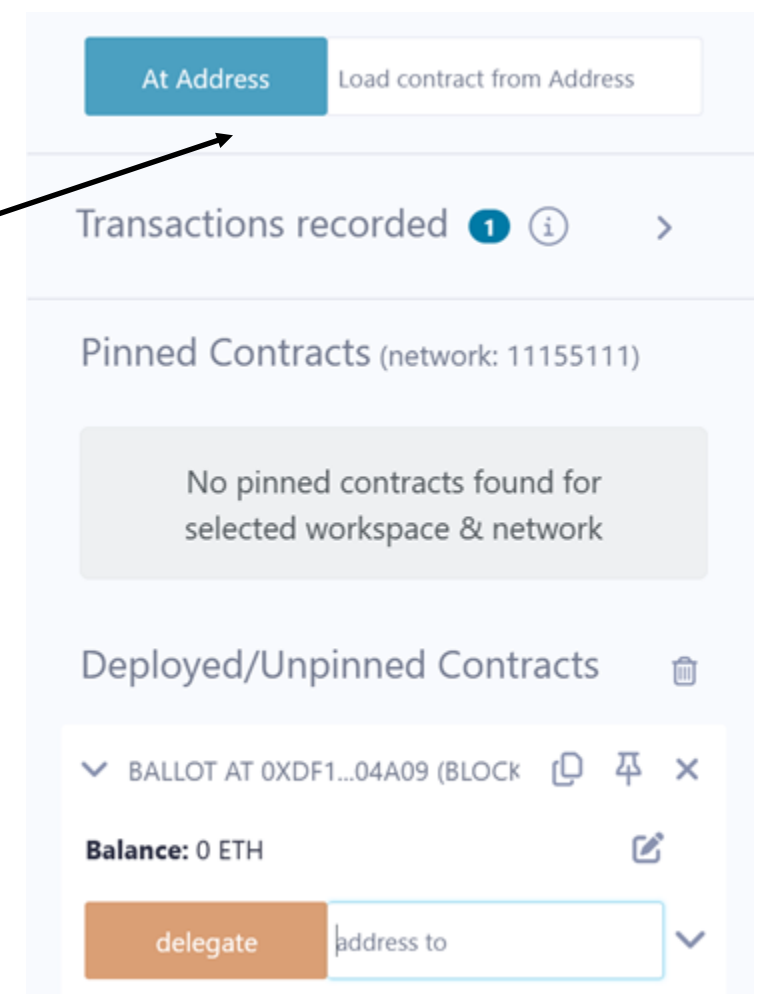
Step 4.3:
Deploying Smart Contract to the Sepolia testnet
Interacting with a Deployed Contract

1. Log in to MetaMask, connect to the Sepolia testnet (as previously explained)
2. In Remix, write and compile your contract, and set the environment to Injected Provider - Metamask.

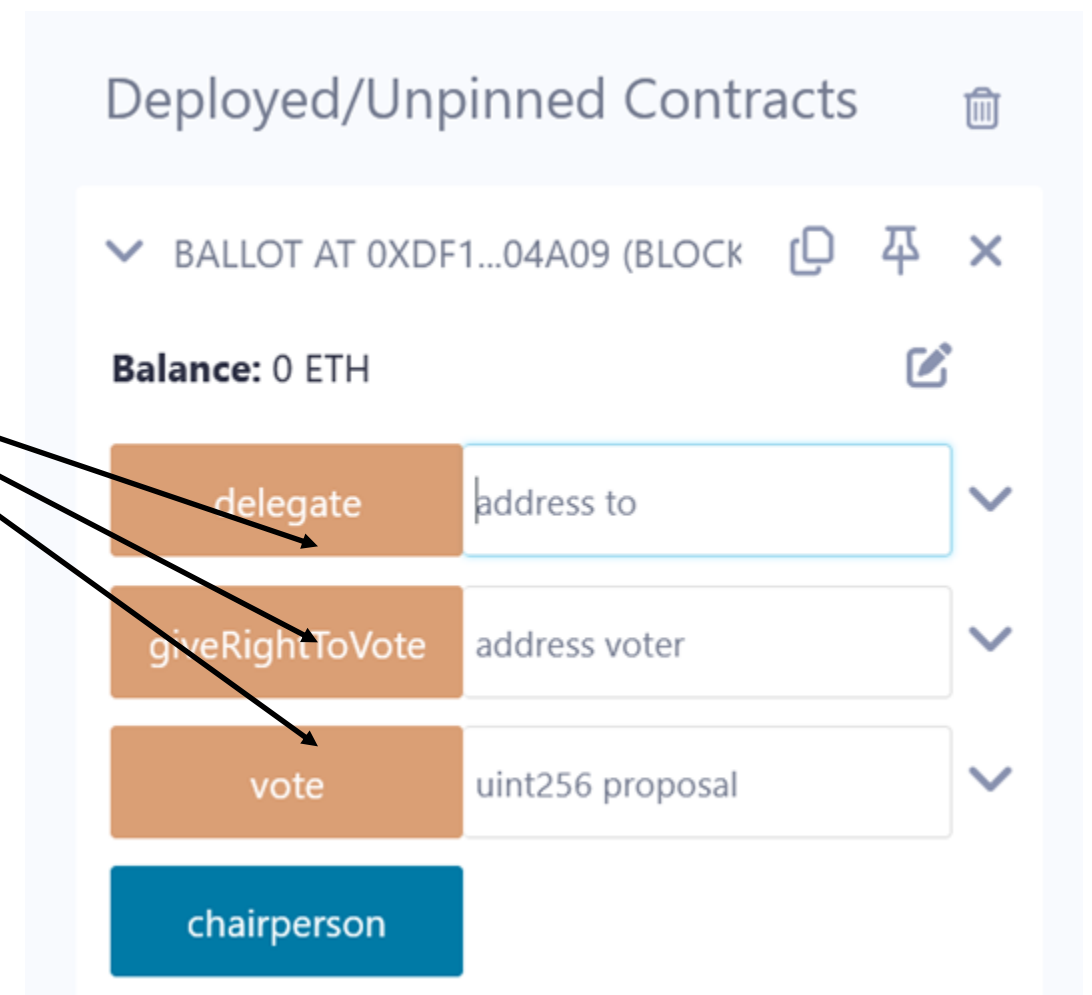3. Insert the deployed contract's address and click on: At Address.

Step 4.3:
Deploying Smart Contract to the Sepolia testnet
Interacting with a Deployed Contract

4- All the public/external functions in the contract are provided and you can pass arguments on them and invoke them

- Orange fields change the contract's state, so you create a transaction and spend funds - at minimum you pay the fees, if your transaction does not send funds to the contract
- Blue fields just show you the contract's state and are free

Deployed/Unpinned Contracts 🗑

⌄ BALLOT AT 0XDF1...04A09 (BLOCK 📋 📌 ✕

Balance: 0 ETH 🖉

delegate | address to | ⌄

giveRightToVote | address voter | ⌄

vote | uint256 proposal | ⌄

chairperson

Troubleshooting

- If you don't see your funds in Metamask
  - Double check that you are connected to the correct network, i.e. the Sepolia testnet and not e.g. the Ethereum mainnet
  - If you are connected to the right testnet, try connecting to a different network (e.g. the Ethereum mainnet) and then reconnecting back to the testnet – sometimes Metamask's connection breaks down, so this will reset the network
  - If you still don't see your funds, try deleting Metamask from your browser (remember to **store your seed** first), and then re-install it, recreate your wallet using your seed (if you have more than one accounts in your wallet, you have to create them all manually again) and connect to the private network – sometimes Metamask's internal transaction generator breaks down (due to temporary network issues), so this will reset your wallet from scratch
  - If you still don't see your funds contact the course's TA

Troubleshooting

- If you want to find past transactions' IDs
  - Metamask keeps a list of all transactions that you have made
  - Click on a transaction and it will redirect you to Etherscan – the public Ethereum network's explorer
  - Etherscan will show you details of your transaction, including itsID (transaction hash)

- If you only see "Injected Provider" in the environment
  - Make sure your metamask extension can read and write data on the remix website
  - If the site access is restricted, click the metamask button and reload the page

- If you interact with a contract but don't see your changes published
  - Make sure that you have set the environment to *Injected Provider - Metamask*
  - Double check that the address of the contract to which you connect is the correct one and the contract's code is <u>exactly the same</u> as the deployed contract