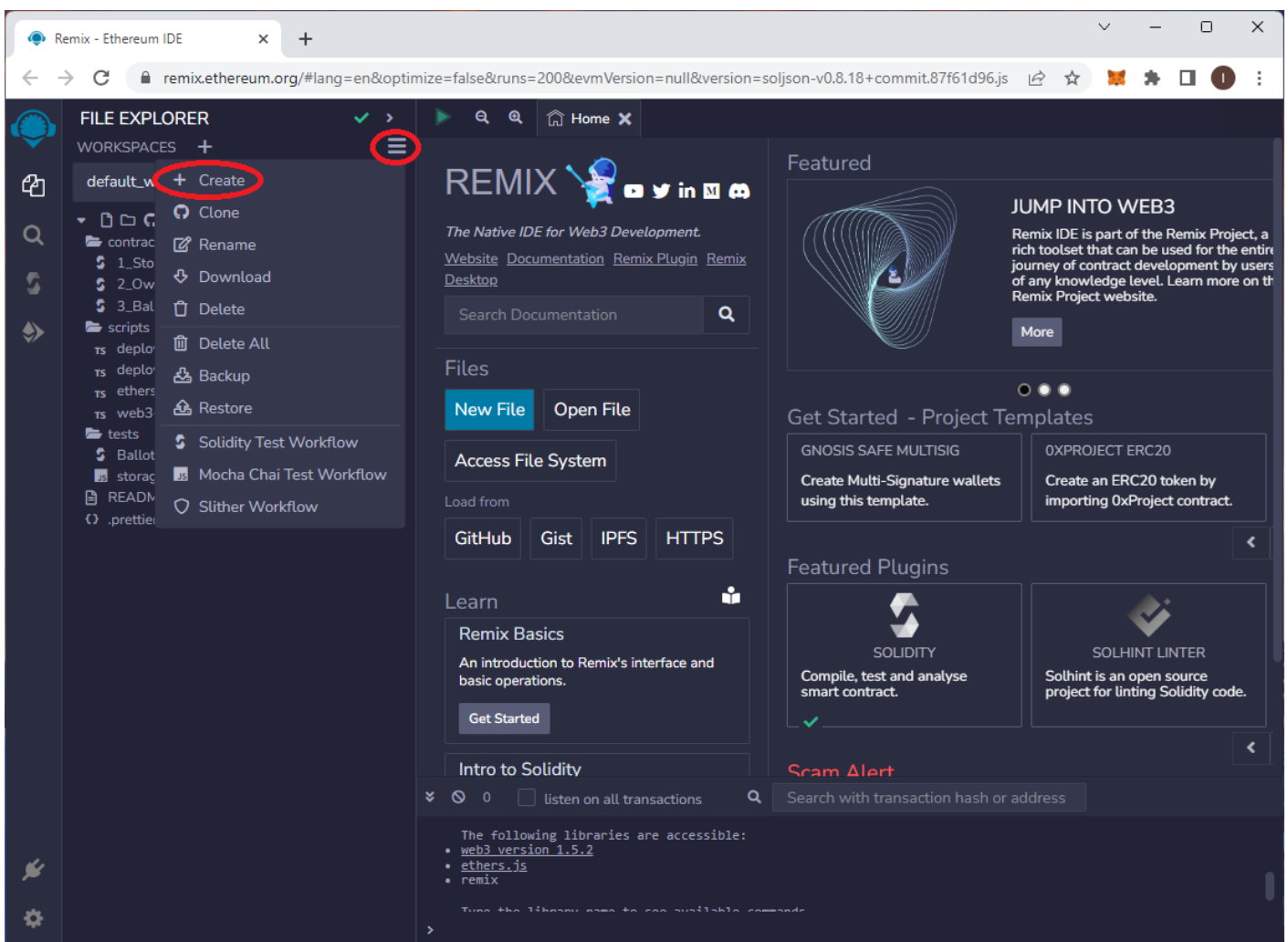
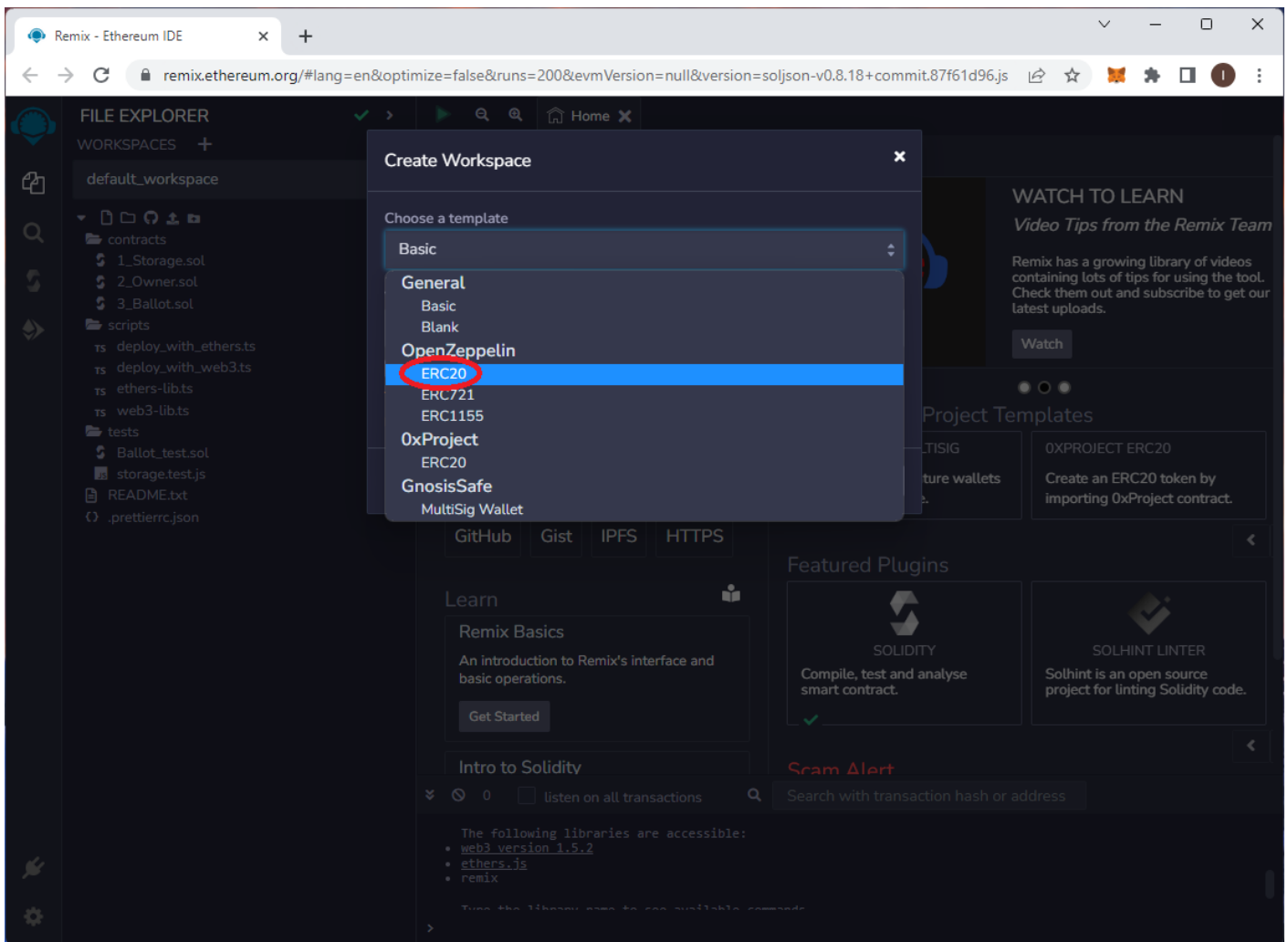


Deploying ERC-20 token on BPX mainnet using Remix IDE

1. Open the [Remix IDE](#). Expand the workspaces menu and select **Create**.



2. Select the **OpenZeppelin / ERC20** template.



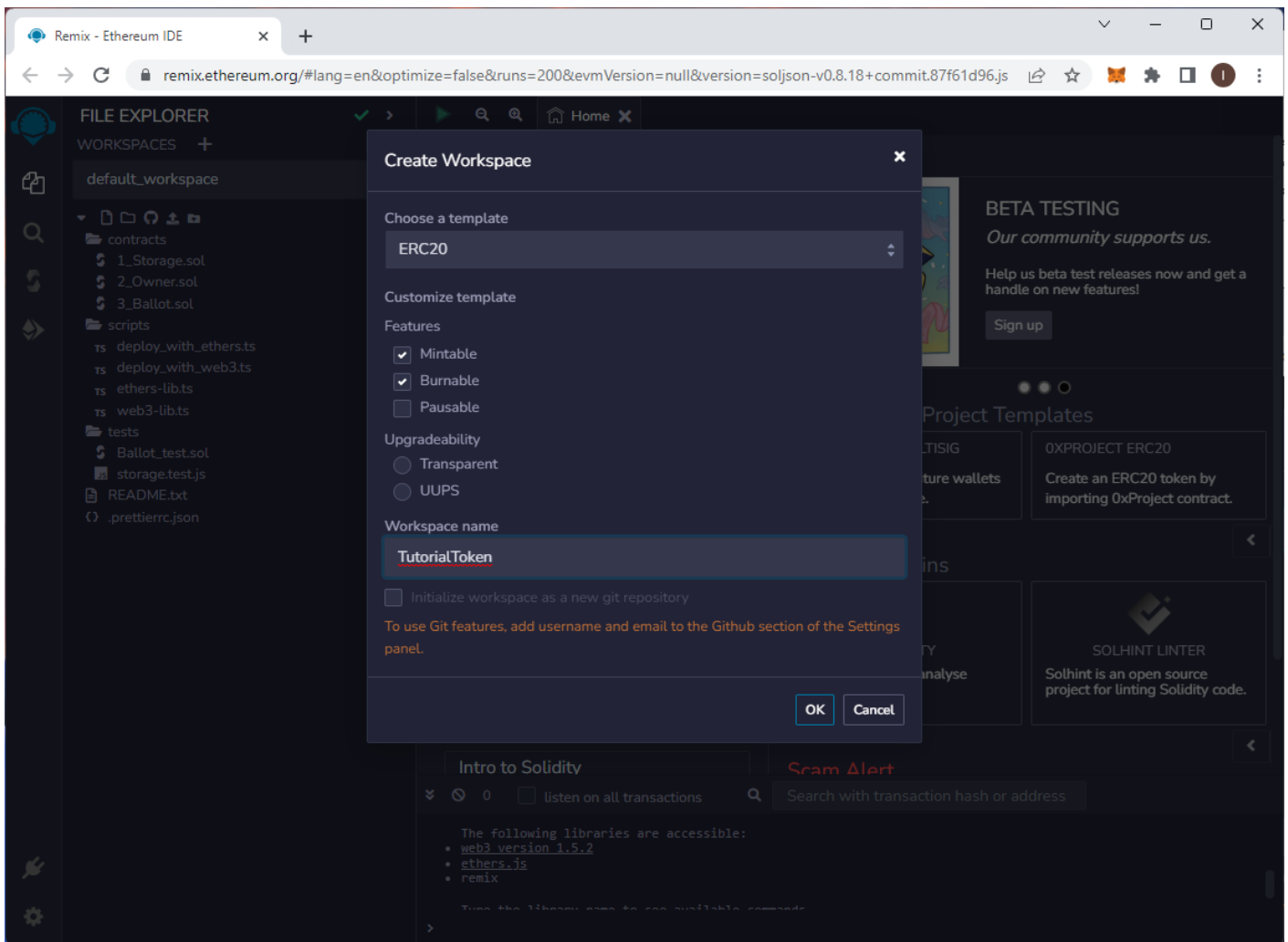
3. Configure your token features.

- Check **Mintable** to be able to mint further units of the token to any address at any time.
- Check **Burnable** to be able to burn token units.
- Check **Pausable** to be able to pause token transfers for all token holders.

In the **Upgradeability** section, you can configure the possibility of updating your token contract in the future. This is an advanced option that we won't cover in the beginner tutorial.

In the **Workspace name** field, enter the name of your project.

Confirm with the **OK** button.



4. The IDE will generate the token contract source code. Change the contract name, token name and token symbol.

Remix - Ethereum IDE

remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js

FILE EXPLORER

WORKSPACES +

TutorialToken

contracts

MyToken.sol

scripts

deploy_with_ethers.ts

deploy_with_web3.ts

ethers-lib.ts

web3-lib.ts

tests

MyToken_test.sol

.prettierrc.json

MyToken.sol

1 // SPDX-License-Identifier: MIT

2 pragma solidity ^0.8.9;

3

4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

5 import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol";

6 import "@openzeppelin/contracts/access/Ownable.sol";

7

8 contract MyToken is ERC20, ERC20Burnable, Ownable {

9 constructor() ERC20("MyToken", "MTK") {}

10

11 function mint(address to, uint256 amount) public onlyOwner {

12 _mint(to, amount);

13 }

14 }

15

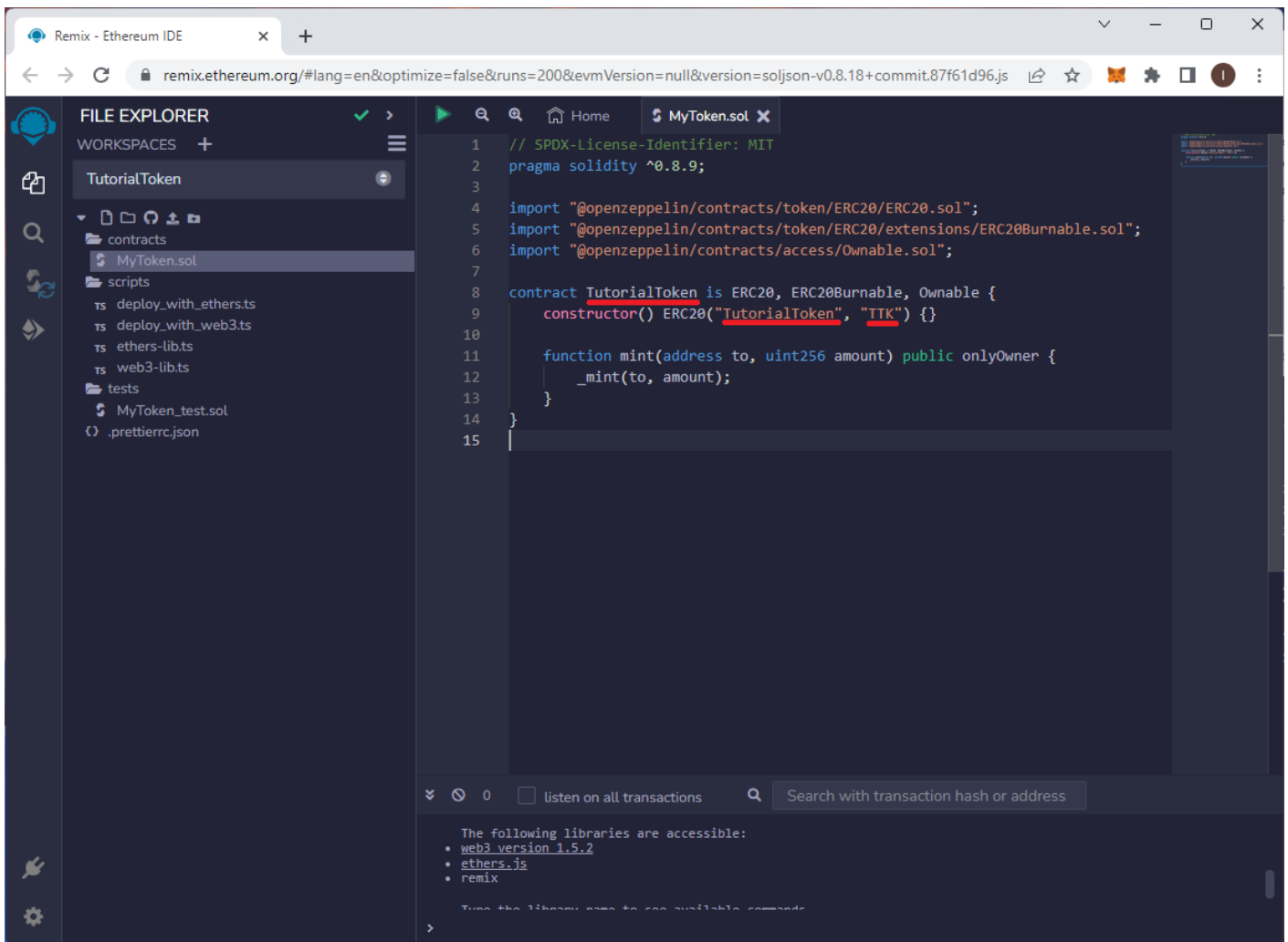
0 ☐ listen on all transactions

Search with transaction hash or address

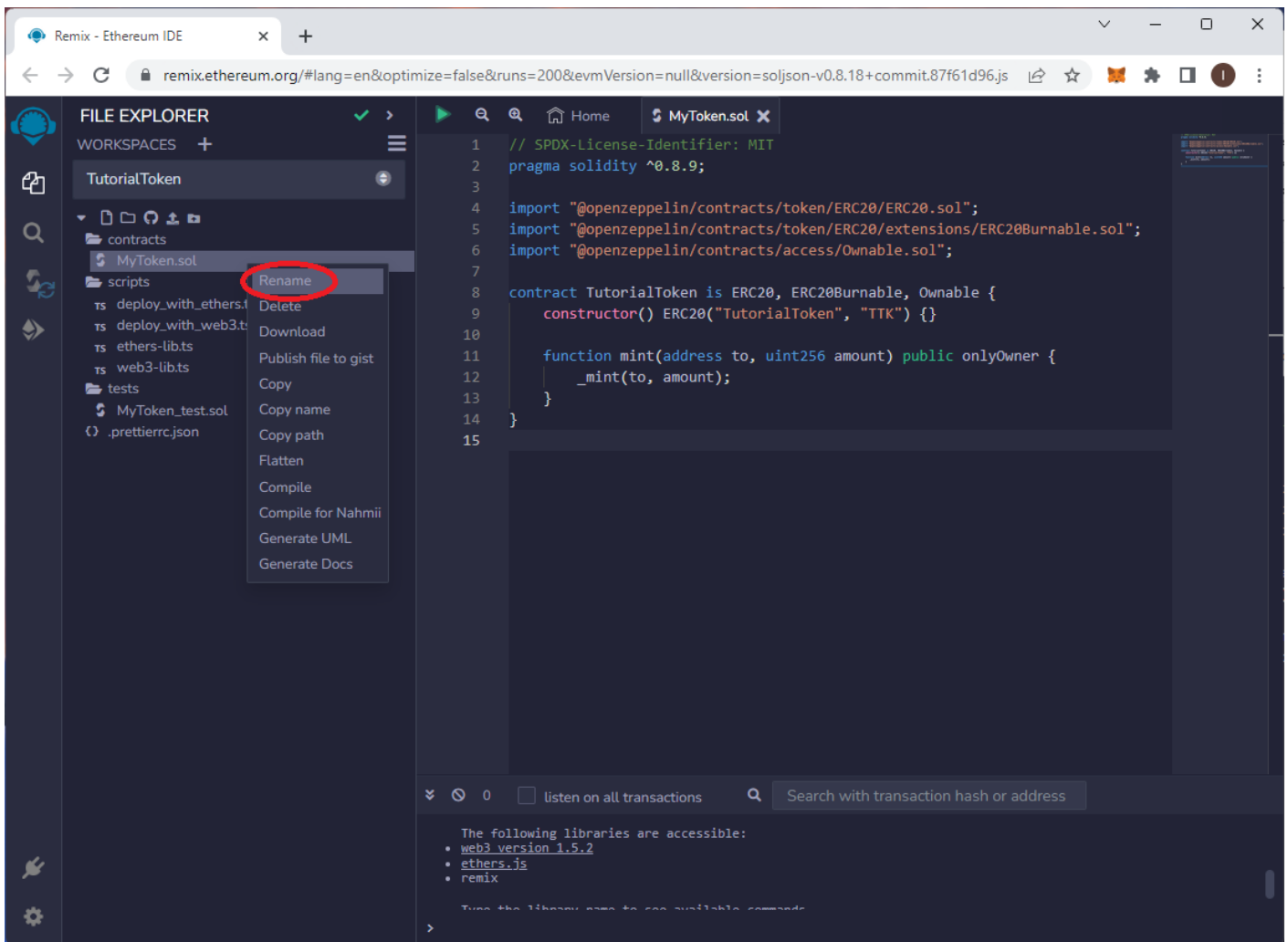
The following libraries are accessible:

- web3 version 1.5.2
- ethers.js
- remix

Try the library name to see available commands



5. Rename the contract file to match the contract name, in our case `TutorialToken.sol`.



6. Compile and deploy the token contract. We explained how to do it in [another tutorial](#).

The screenshot shows the Remix Ethereum IDE interface. The left sidebar contains the 'DEPLOY & RUN TRANSACTIONS' panel, which lists 'Transactions recorded' (2) and 'Deployed Contracts'. Under 'Deployed Contracts', there is a contract named 'TUTORIALTOKEN AT 0X602...AA7' with a balance of 0 ETH. Below this, there is a list of contract methods: approve, burn, burnFrom, decreaseAllow, increaseAllow, mint, renounceOwn, transfer, transferFrom, transferOwn, allowance, balanceOf, and decimals. The main editor displays the Solidity code for 'TutorialToken.sol'. The code is as follows:

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.9;
3
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5 import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol";
6 import "@openzeppelin/contracts/access/Ownable.sol";
7
8 contract TutorialToken is ERC20, ERC20Burnable, Ownable {
9     constructor() ERC20("TutorialToken", "TTK") {}
10
11     function mint(address to, uint256 amount) public onlyOwner {
12         _mint(to, amount);
13     }
14 }
15
```

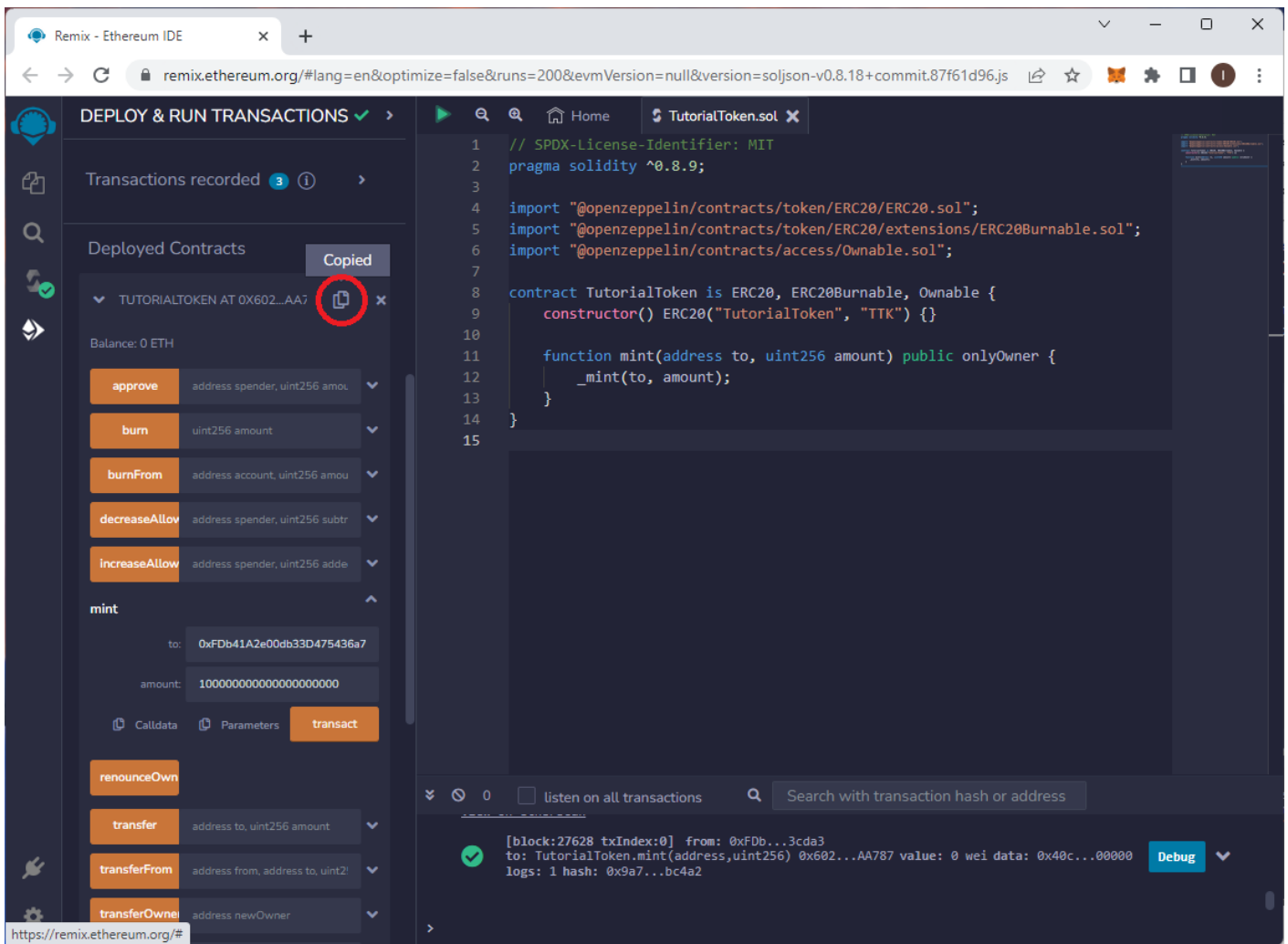
The bottom panel shows a transaction log with a green checkmark, indicating a successful transaction. The log entry is: [block:27624 txIndex:0] from: 0xFDb...3cda3 to: TutorialToken.(constructor) value: 0 wei data: 0x608...20033 logs: 1 hash: 0x05b...5684f. There is a 'Debug' button next to the log entry.

7. The token contract is already on the blockchain. Now we will mint 100 TTK to our wallet. Expand the contract method named `mint`. Enter your wallet address in `to:` field. In the `amount:` field enter the amount of token you want to mint. Keep in mind that the token has 18 decimal places by default, and the amount is entered in the smallest units. So enter 100 followed by 18 zeros to mint 100 TTK.

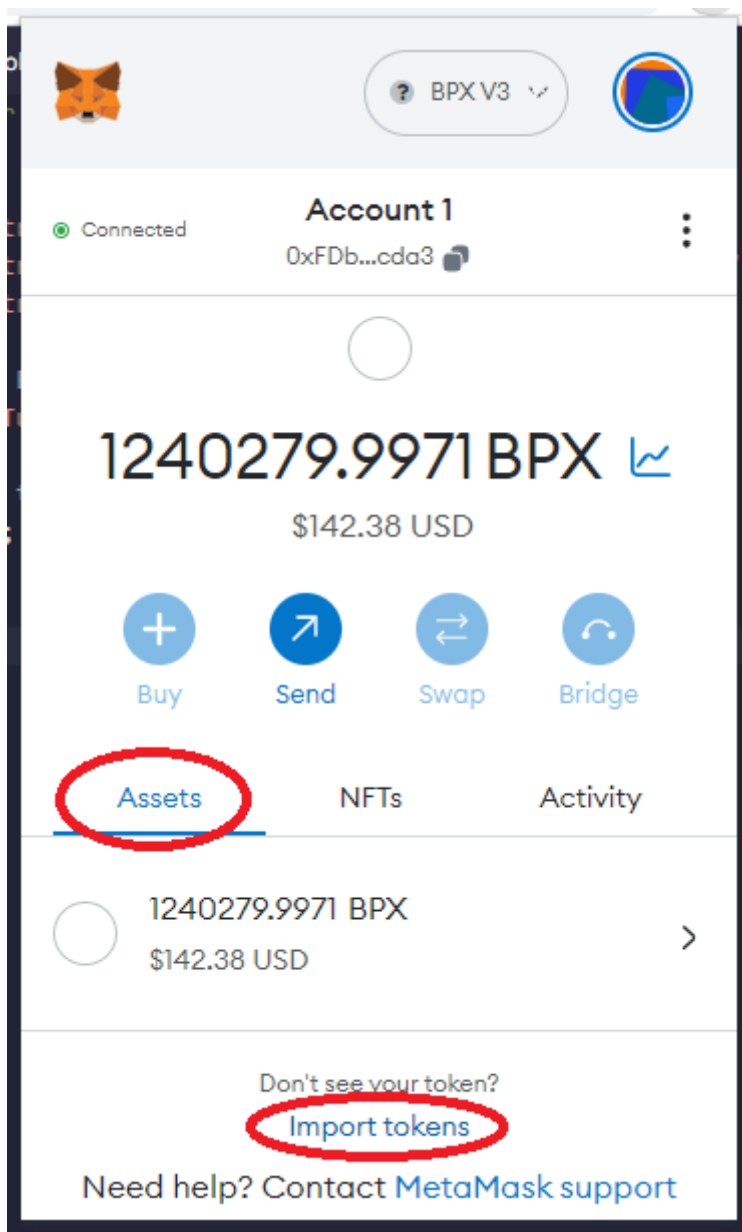
The screenshot displays the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is active, showing a list of deployed contracts under 'TUTORIALTOKEN AT 0x602...AA7'. The 'mint' function is selected, and the 'transact' button is highlighted. The main editor shows the Solidity code for the 'TutorialToken' contract, which inherits from ERC20, ERC20Burnable, and Ownable. The code includes a constructor and a mint function. The bottom panel shows a successful transaction confirmation with a green checkmark and details: '[block:27624 txIndex:0] from: 0xFDb...3cda3 to: TutorialToken.(constructor) value: 0 wei data: 0x608...20033 logs: 1 hash: 0x05b...5684f'.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.9;
3
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5 import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol";
6 import "@openzeppelin/contracts/access/Ownable.sol";
7
8 contract TutorialToken is ERC20, ERC20Burnable, Ownable {
9     constructor() ERC20("TutorialToken", "TTK") {}
10
11     function mint(address to, uint256 amount) public onlyOwner {
12         _mint(to, amount);
13     }
14 }
15
```

8. Click **transact** and confirm the transaction in your wallet. Wait for its confirmation by the blockchain.



10. Now open Metamask, go to **Assets** and select **Import tokens**.



11. Paste the token contract address. The other fields will be automatically read from the blockchain. Click **Add custom token** and **Import tokens**.



BPX V3



Import tokens



Custom token

and make sure you trust the team about scams and security risks.

Token contract address

.5eAe83195486AC85dC40032dAA787

Token symbol

Edit

TTK

Token decimal

18

Add custom token



BPX V3



Import tokens

Would you like to import these tokens?

Token

Balance



TTK

100 TTK

12. Done. The token is visible in your wallet. You can send and receive it to other addresses just like a native BPX token. Keep in mind that your wallet is the smart contract owner and only this wallet can call `mint` function in the future to mint new token units or `burn` function to burn them.

