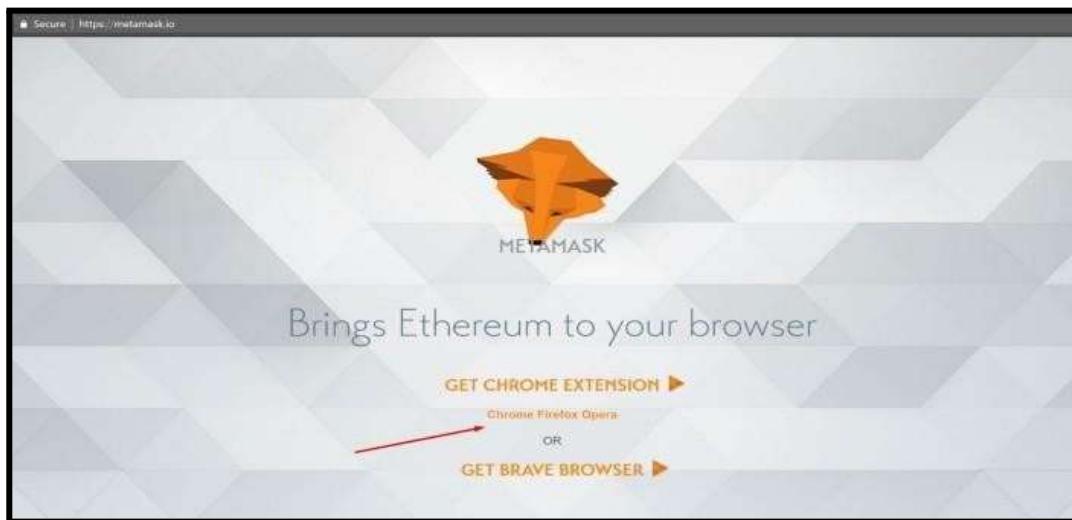
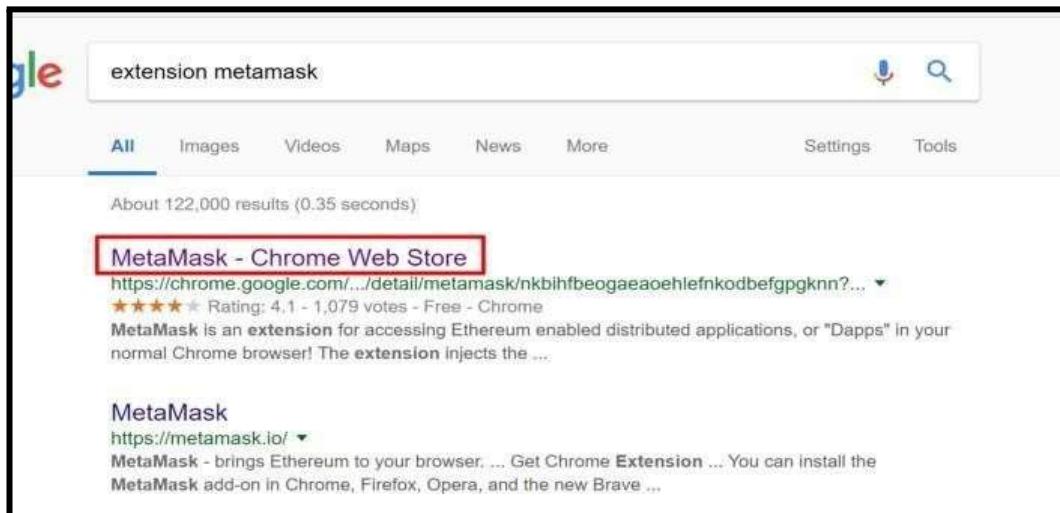
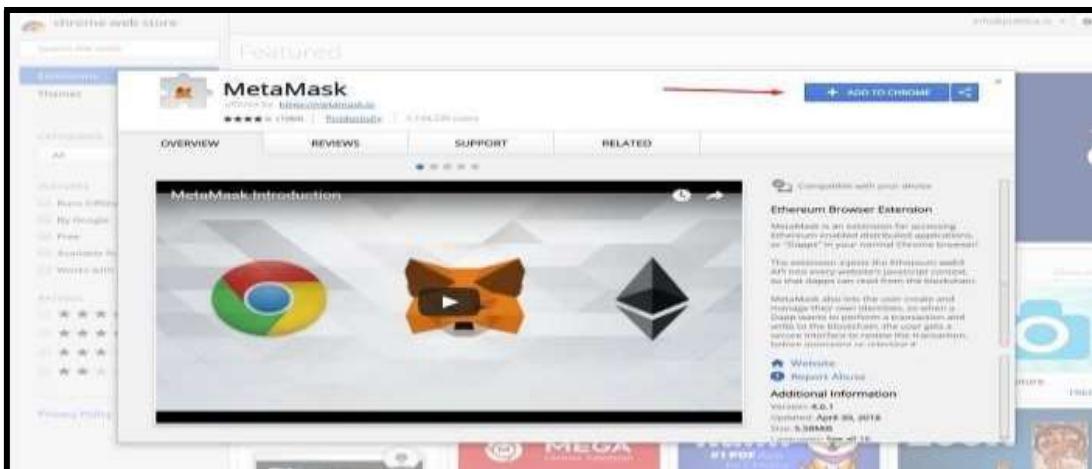
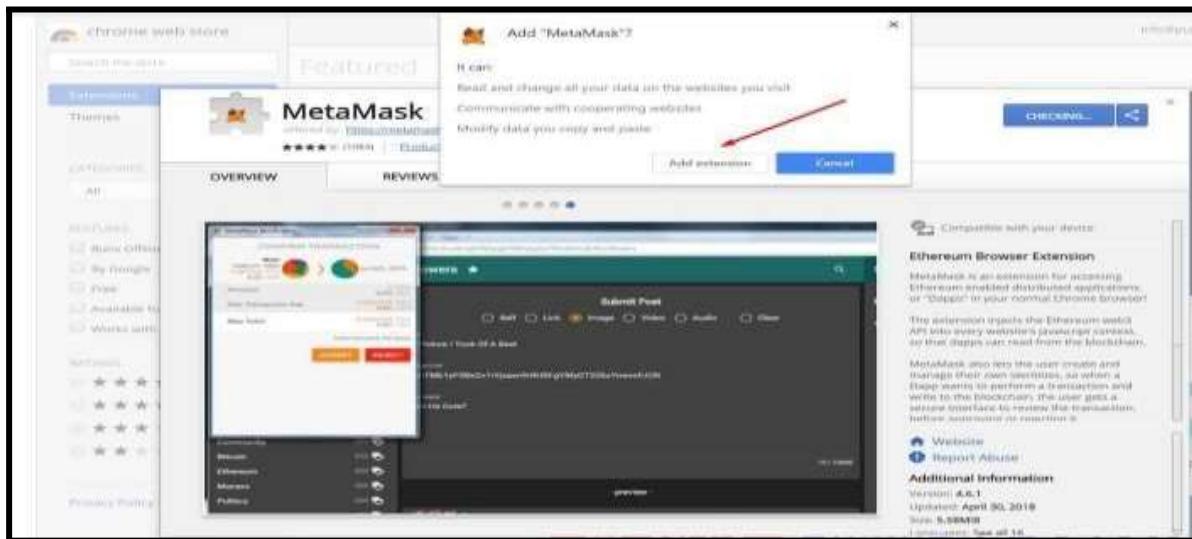


## Install MetaMask on your browser.

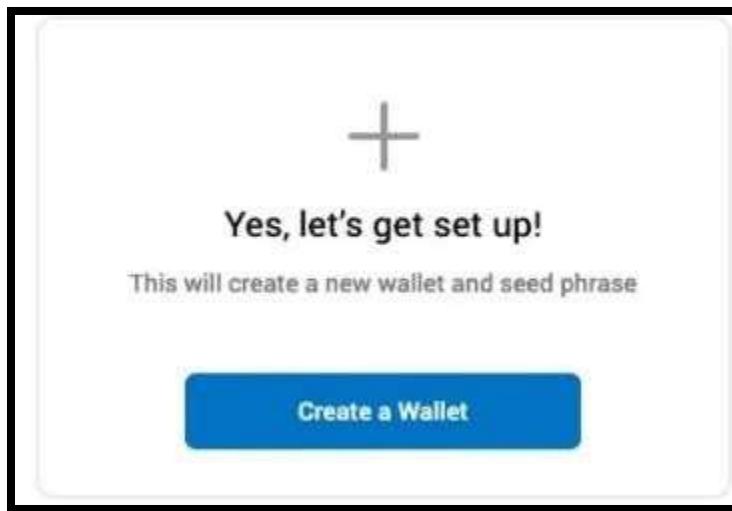


- Click on **Install MetaMask** as a Google Chrome extension.
- Click **Add to Chrome**.
- Click **Add Extension**.





Click the ‘Get Started’ button to begin creating your Ethereum wallet using MetaMask. On the next step, click the ‘Create a Wallet’ button.



then be asked if you want to help improve MetaMask. Click ‘No Thanks’ if this doesn’t interest you, otherwise click ‘I agree’.

## Help Us Improve MetaMask

MetaMask would like to gather usage data to better understand how our users interact with the extension. This data will be used to continually improve the usability and user experience of our product and the Ethereum ecosystem.

MetaMask will..

- ✓ Always allow you to opt-out via Settings
- ✓ Send anonymized click & pageview events
- ✓ Maintain a public aggregate dashboard to educate the community
  
- ✗ Never collect keys, addresses, transactions, balances, hashes, or any personal information
- ✗ Never collect your full IP address
- ✗ Never sell data for profit. Ever!

No Thanks

I agree

This data is aggregated and is therefore anonymous for the purposes of General Data Protection Regulation (EU) 2016/679. For more information in relation to our privacy practices, please see our [Privacy Policy here](#).

Pick a password on the next step. This needs to be at least 8 characters long. We recommend using a completely unique password that hasn't been used anywhere else, and one that contains a mixture of upper and lower case letters, symbols, and numbers.

## Create Password

New password (min 8 chars)

.....

Confirm password

.....

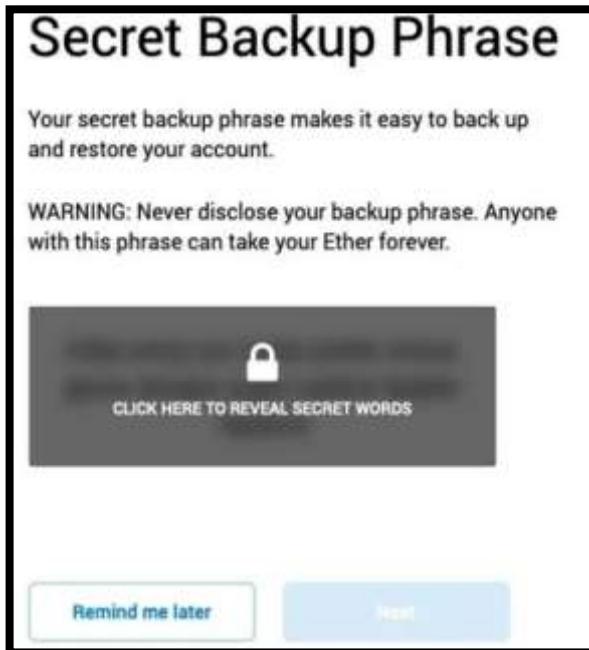


I have read and agree to the [Terms of Use](#)

Create

Read and accept the Terms of Use, and click ‘Create’ once your password has been set. MetaMask will then present you with your 12-word backup phrase. You’ll need to write this phrase down carefully, with the words recorded in the same order displayed on your screen. This phrase will be needed to recover your wallet should you ever lose access to your computer, and should be kept stored somewhere safe. Anybody who has access to your 12-word backup phrase will have access to the funds in your MetaMask wallet, so keep it private.

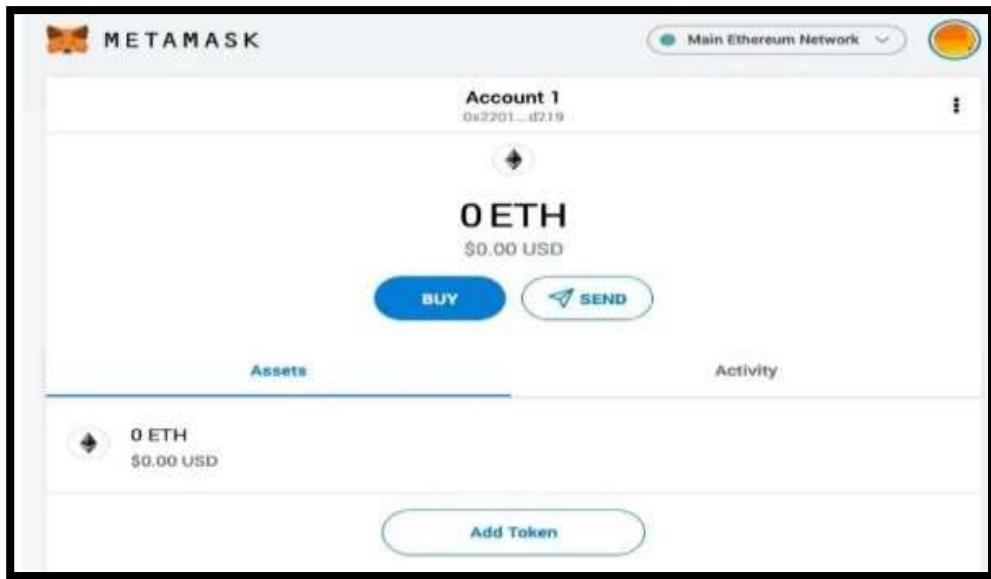
Click ‘Next’ once you’ve written this down.



Confirm your backup phrase on the next screen by entering the words in the same order saved previously. Click ‘Confirm’ once done.



You have now almost completed the MetaMask setup process. Just click ‘All Done’ on the final page, and you will be automatically logged in to MetaMask. If you ever get logged out, you’ll be able to log back in again by clicking the MetaMask icon, which will have been added to your web browser (usually found next to the URL bar).



**Code:**

```
// SPDX-License-Identifier: Unlicensed

pragma solidity ^0.6.0;

contract MyBank

{

    mapping(address=> uint ) private _balances;

    address public owner;

    event LogDepositeMade(address accountHoder, uint amount );

    constructor () public

    {

        owner=msg.sender;

        emit LogDepositeMade(msg.sender, 1000);

    }

    function deposit() public payable returns (uint)

    {

        require ((_balances[msg.sender] + msg.value) > _balances[msg.sender] &&

msg.sender!=address(0));

        _balances[msg.sender] += msg.value;

        emit LogDepositeMade(msg.sender , msg.value);

        return _balances[msg.sender];

    }

}
```

```

    }

function withdraw (uint withdrawAmount) public returns (uint)

{
    require (_balances[msg.sender] >= withdrawAmount);

    require(msg.sender!=address(0));

    require (_balances[msg.sender] > 0);

    _balances[msg.sender]-= withdrawAmount;

    msg.sender.transfer(withdrawAmount);

    emit LogDepositeMade(msg.sender , withdrawAmount);

    return _balances[msg.sender];
}

```

```

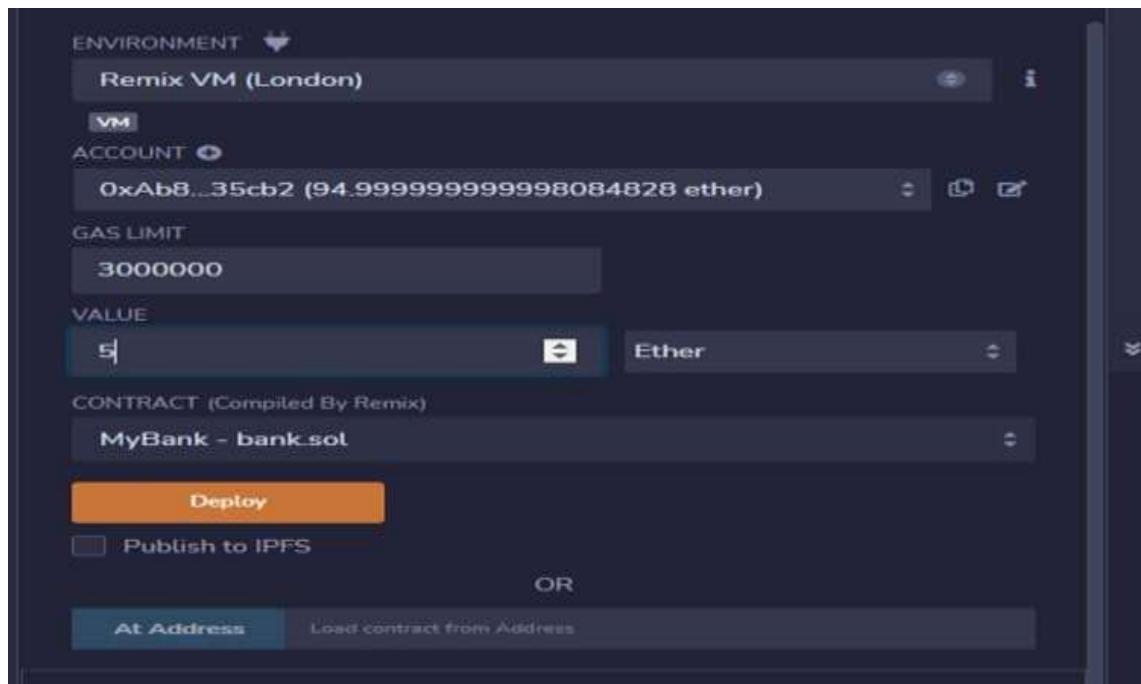
function viewBalance() public view returns (uint)

{
    return _balances[msg.sender];
}

```

#### Output:

The screenshot shows the Truffle UI interface for a deployed Ethereum contract named "MYBANK". The contract's address is 0xD2A...FD005 (MEMORY). The balance is listed as 0 ETH. Below the address, there are four buttons corresponding to the contract's functions: "deposit" (red), "withdraw" (orange with an input field for "uint256 withdrawAmount"), "owner" (blue), and "viewBalance" (blue).



The screenshot shows the Deployed Contracts interface for the MYBANK contract at address **0x843...40406**. The interface displays:

- Balance: 5 ETH**
- deposit** button.
- withdraw** button: uint256 withdrawAmount input field.
- owner** button.
- 0: address: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2**
- viewBalanc...** button.
- 0: uint256: 500000000000000000000000**
- Low level interactions** section.
- CALldata** input field.
- Transact** button.

### Deployed Contracts

MYBANK AT 0X843...40406 (MEMORY)

Balance: 4.9999999999999988 ETH

**deposit**

**withdraw** 120

**owner**

0: address: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2

**viewBalanc...**

0: uint256: 50000000000000000000000000000000

**Low level interactions**

CALldata

This screenshot shows a deployed smart contract named 'MYBANK' at address 0X843...40406. The contract has a balance of 4.9999999999999988 ETH. It features four main buttons: 'deposit' (red), 'withdraw' (orange), 'owner' (blue), and 'viewBalanc...' (teal). Below the withdraw button is a dropdown menu set to 120. The 'viewBalanc...' button is followed by its output: 0: uint256: 50000000000000000000000000000000. A section titled 'Low level interactions' contains a 'CALldata' button.

### Deployed Contracts

MYBANK AT 0X843...40406 (MEMORY)

Balance: 4.9999999999999988 ETH

**deposit**

**withdraw** uint256 withdrawAmount

**owner**

0: address: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2

**viewBalanc...**

0: uint256: 499999999999999880

**Low level interactions**

CALldata

This screenshot shows the same deployed smart contract 'MYBANK' at address 0X843...40406. The balance remains at 4.9999999999999988 ETH. The interface includes the same four buttons: 'deposit', 'withdraw', 'owner', and 'viewBalanc...'. The 'withdraw' button now includes a 'uint256 withdrawAmount' input field. The 'viewBalanc...' button is followed by its output: 0: uint256: 499999999999999880. The 'Low level interactions' section also includes a 'CALldata' button, which is present but inactive in this view.