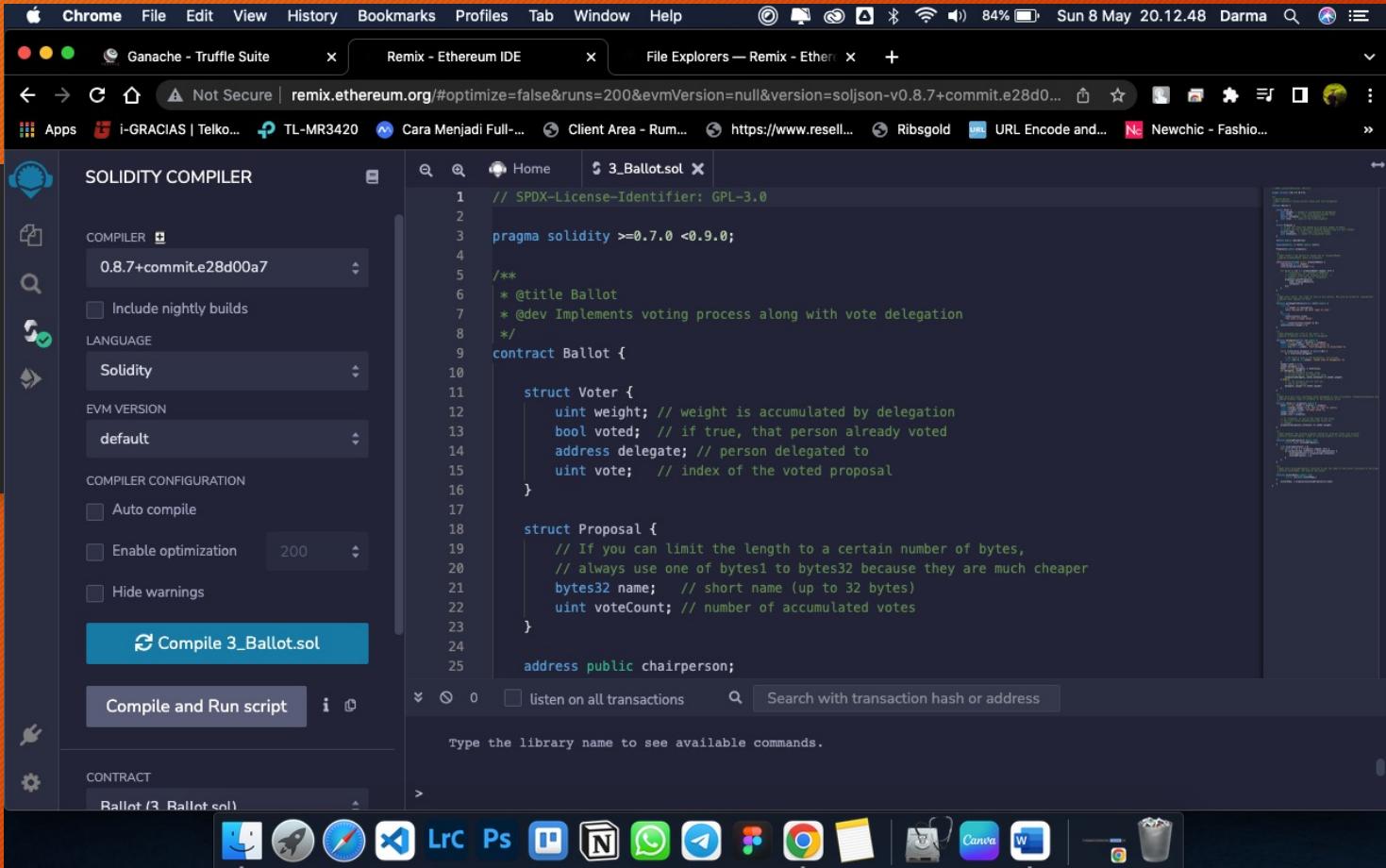


In the center window, you will see some default code, which is a sample Solidity code. You will type your contract code in this code editor. Your code may be auto-compiled.



The screenshot shows a web-based Ethereum IDE running in a Chrome browser. The title bar indicates the window is titled "Remix - Ethereum IDE". The main content area displays a Solidity code editor with the file "3_Ballot.sol" open. The code is a template for a Ballot contract, defining a Voter struct with fields for weight, voted status, delegate address, and proposal index, and a Proposal struct with a name (bytes32), vote count (uint), and delegate address. It also includes a public chairperson address. The left sidebar contains a "SOLIDITY COMPILER" panel with settings for the compiler version (0.8.7+commit.e28d00a7), language (Solidity), EVM version (default), and compiler configuration options like auto-compile and optimization level (200). A prominent blue button labeled "Compile 3_Ballot.sol" is visible. Below the code editor, there's a "CONTRACT" section showing "Ballot (3_Ballot.sol)" and a "Deploy" button. At the bottom, a toolbar features icons for various Ethereum tools and services, including Ganache, Truffle Suite, and several wallet and developer applications. The browser's address bar shows the URL "remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d0...".

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;

/**
 * @title Ballot
 * @dev Implements voting process along with vote delegation
 */
contract Ballot {

    struct Voter {
        uint weight; // weight is accumulated by delegation
        bool voted; // if true, that person already voted
        address delegate; // person delegated to
        uint vote; // index of the voted proposal
    }

    struct Proposal {
        // If you can limit the length to a certain number of bytes,
        // always use one of bytes1 to bytes32 because they are much cheaper
        bytes32 name; // short name (up to 32 bytes)
        uint voteCount; // number of accumulated votes
    }

    address public chairperson;
}
```

Solidity for Contract Writing

– Compiling the Contract

The screenshot shows the Remix Ethereum IDE interface. On the left, the Solidity Compiler sidebar is open, displaying compiler version 0.8.7+commit.e28d00a7, language Solidity, EVM version default, and configuration options like Auto compile and Enable optimization set to 200. A blue button labeled "Compile mycontract.sol" is visible. The main workspace contains the Solidity source code for a contract named MyContract. The code defines a constructor that initializes amount to 0 and value to 1000. It includes three public view functions: getBalance, getAmount, and send, which modify the amount variable. At the bottom of the code editor, there are buttons for "Compile and Run script" and "Deploy to Ganache". The status bar at the bottom shows the current block number (0), a checkbox for listening on all transactions, a search bar, and a message to type a library name to see available commands.

```
contract MyContract {
    uint amount;
    uint value;

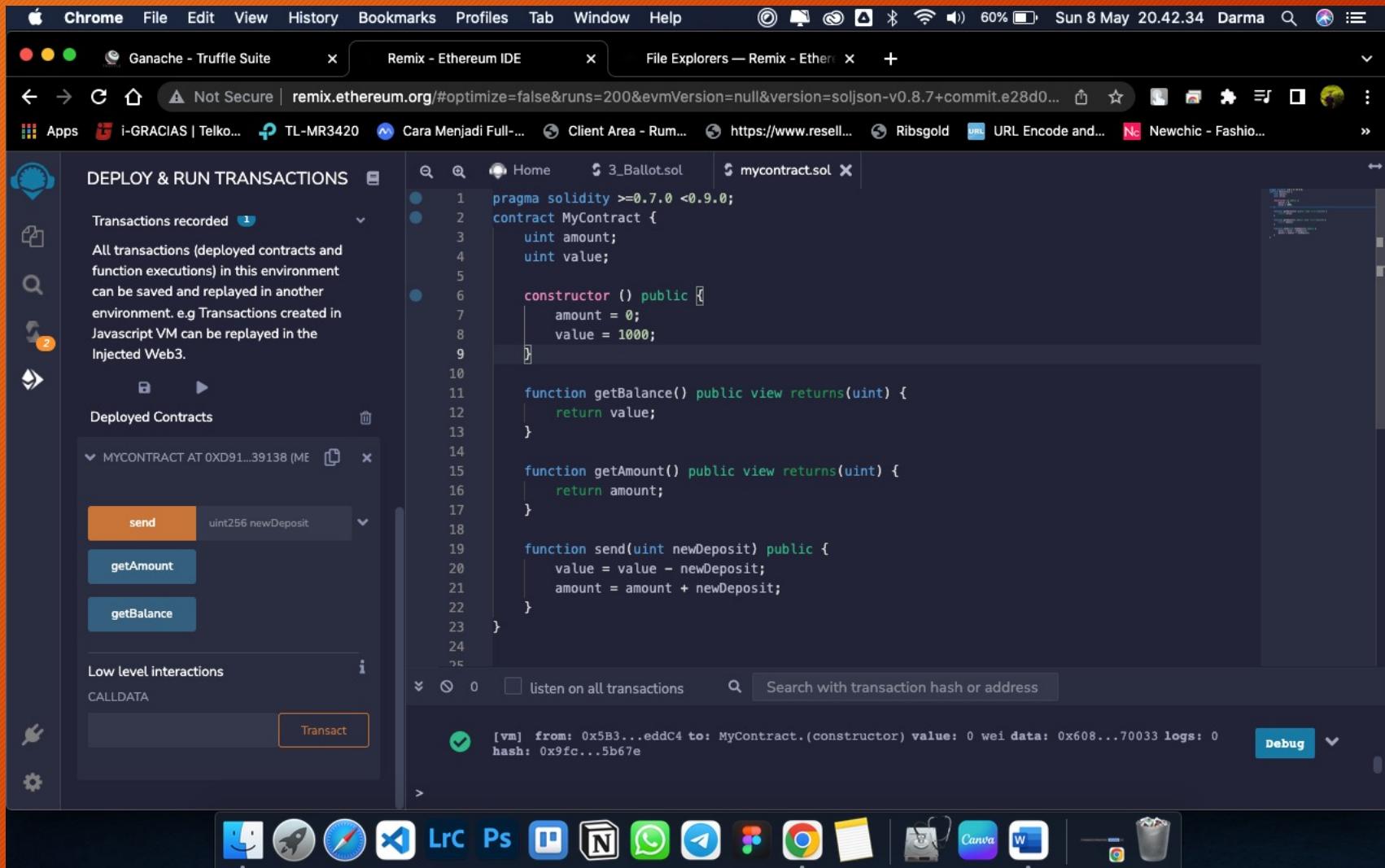
    constructor (uint initialAmount, uint initialValue) public {
        amount = 0;
        value = 1000;
    }

    function getBalance() public view returns(uint) {
        return value;
    }

    function getAmount() public view returns(uint) {
        return amount;
    }

    function send(uint newDeposit) public {
        value = value - newDeposit;
        amount = amount + newDeposit;
    }
}
```

6. Ethereum – Deploying the Contract



Sending Money

The screenshot shows the Ethereum IDE interface running in a Chrome browser window. The main area displays a Solidity smart contract named `MyContract.sol`. The code defines a contract with variables `amount` and `value`, a constructor that initializes `amount = 0` and `value = 1000`, and three functions: `getBalance`, `getAmount`, and `send`. The `send` function takes a parameter `newDeposit` and updates the contract's state. On the left, the sidebar shows a list of deployed contracts, including `MYCONTRACT AT 0xD91...39138 (ME)`. Below the sidebar, there are buttons for `send` (set to 100), `getAmount`, and `getBalance`. The `getBalance` button is highlighted. The bottom section shows the transaction history with one entry: `[vm] from: 0x5B3...eddC4 to: MyContract.(constructor) value: 0 wei data: 0x608...70033 logs: 0 hash: 0x7e4...4ce37`. The interface includes various icons for file operations, search, and deployment.

```
pragma solidity >=0.7.0 <0.9.0;
contract MyContract {
    uint amount;
    uint value;

    constructor () public {
        amount = 0;
        value = 1000;
    }

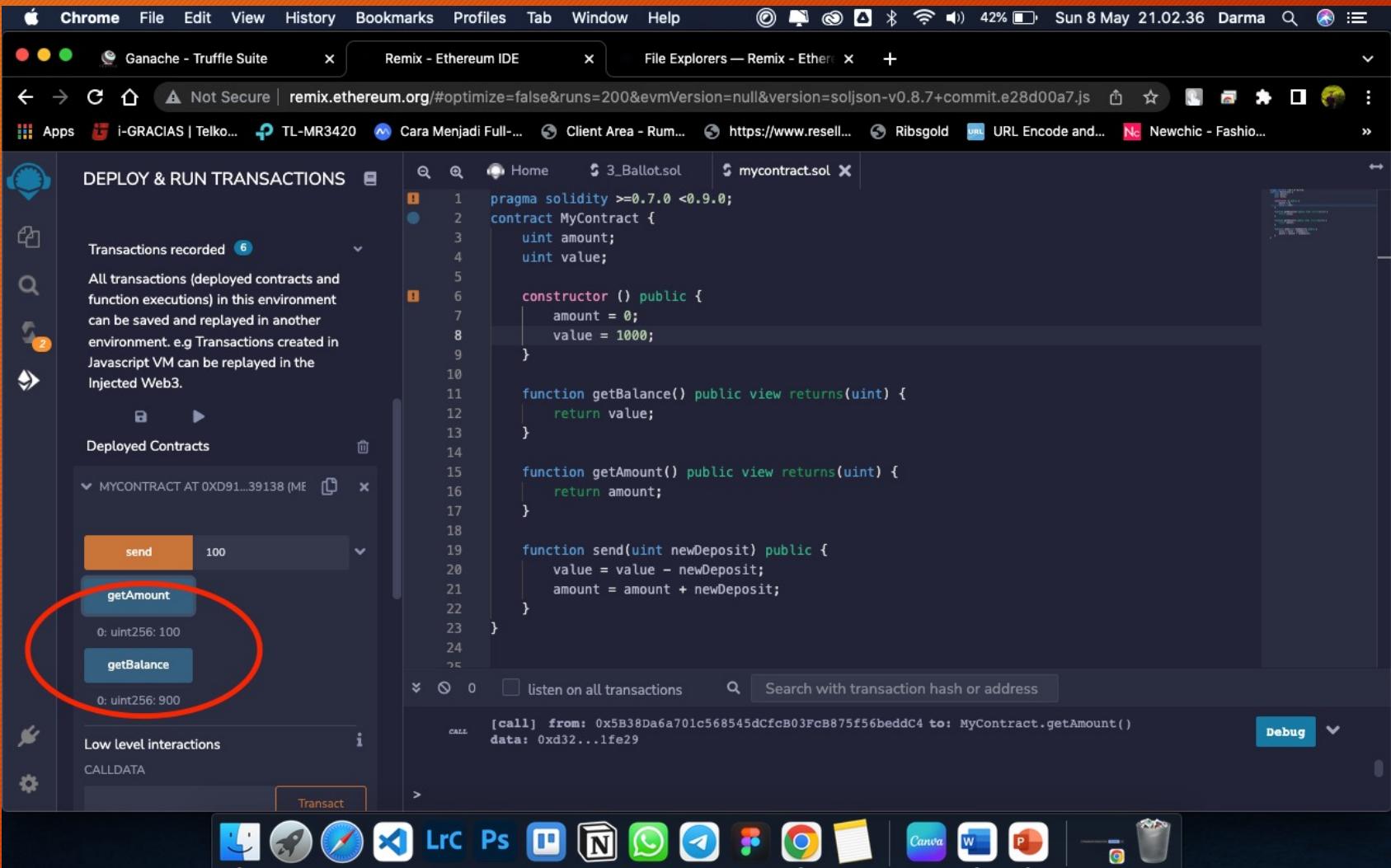
    function getBalance() public view returns(uint) {
        return value;
    }

    function getAmount() public view returns(uint) {
        return amount;
    }

    function send(uint newDeposit) public {
        value = value - newDeposit;
        amount = amount + newDeposit;
    }
}
```

Examining Contract Value

Examining Collected Amount



The screenshot shows the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar displays a transaction history with 6 recorded transactions and a list of deployed contracts. A specific contract, 'MYCONTRACT AT 0xD91...', is selected, showing its address and a transaction history. One transaction is highlighted with a red circle, showing a 'send' operation of 100 and two function calls: 'getAmount' and 'getBalance'. The 'getAmount' call returns 100, and the 'getBalance' call returns 900. On the right, the code editor shows the Solidity source code for 'mycontract.sol'. The code defines a contract named 'MyContract' with variables 'amount' and 'value', a constructor setting initial values, and functions for getting the balance and amount, and sending new deposits.

```
pragma solidity >=0.7.0 <0.9.0;
contract MyContract {
    uint amount;
    uint value;

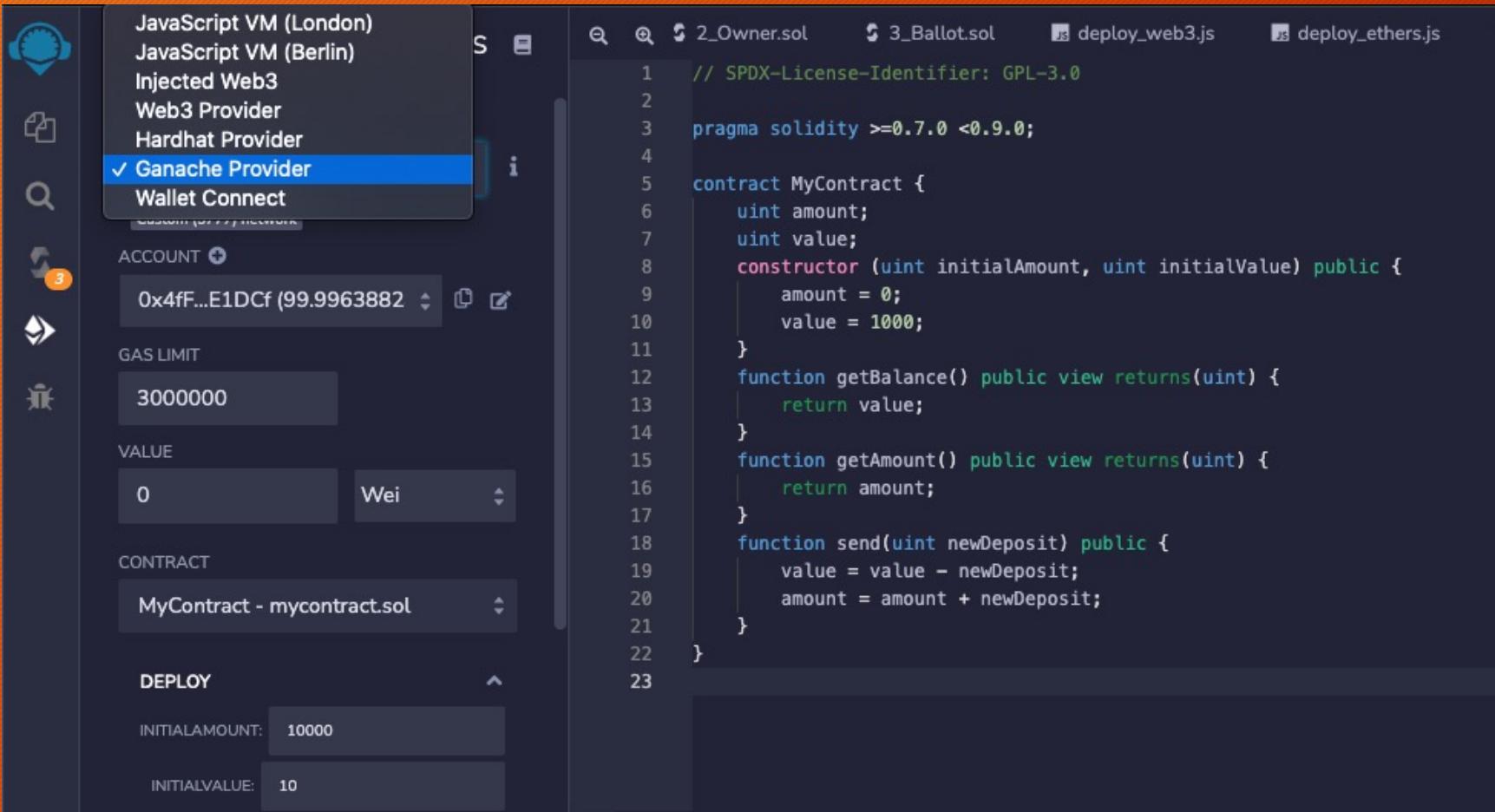
    constructor () public {
        amount = 0;
        value = 1000;
    }

    function getBalance() public view returns(uint) {
        return value;
    }

    function getAmount() public view returns(uint) {
        return amount;
    }

    function send(uint newDeposit) public {
        value = value - newDeposit;
        amount = amount + newDeposit;
    }
}
```

Koneksikan pada Provider Ganache

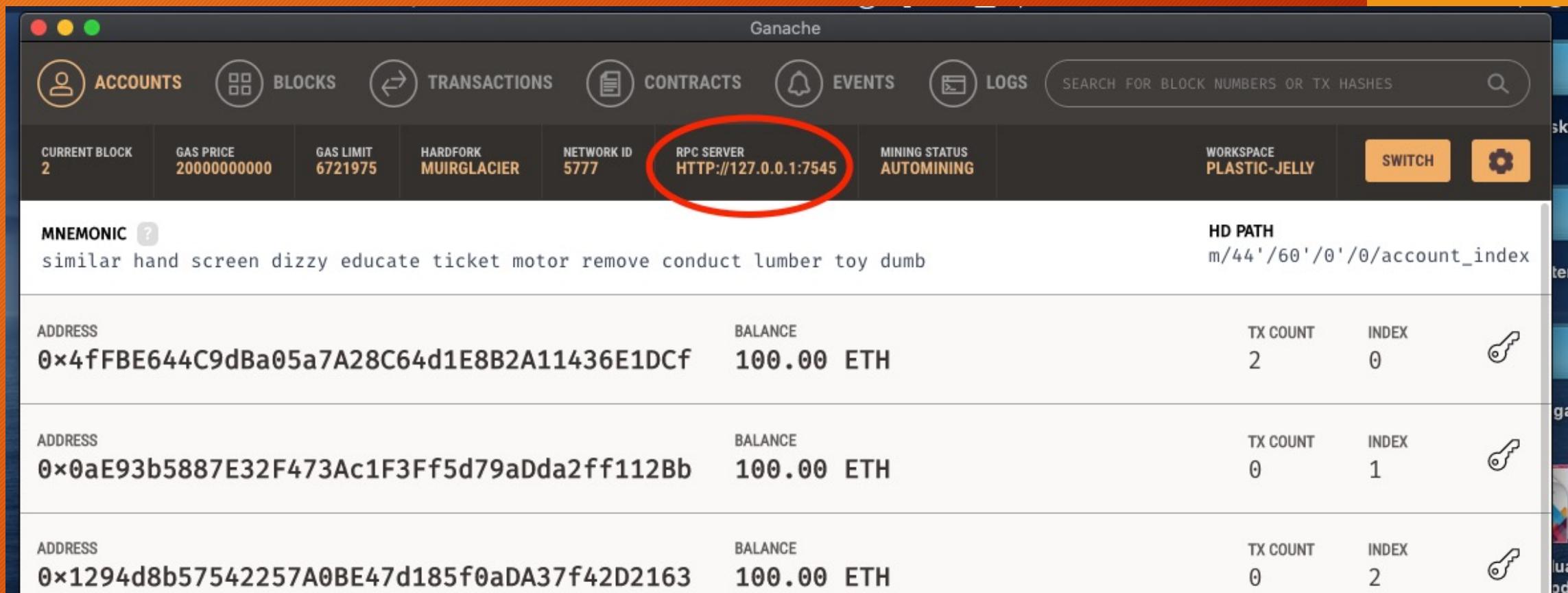


The screenshot shows the Truffle UI interface. On the left, there's a sidebar with various icons and a dropdown menu for selecting a provider. The 'Ganache Provider' option is highlighted with a blue selection bar. The main area contains tabs for different Solidity files (2_Owner.sol, 3_Ballot.sol) and deployment scripts (deploy_web3.js, deploy_ether.js). Below the tabs, a code editor displays the MyContract Solidity code. The interface also includes sections for ACCOUNT, GAS LIMIT, VALUE, CONTRACT, and DEPLOY.

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;

contract MyContract {
    uint amount;
    uint value;
    constructor (uint initialAmount, uint initialValue) public {
        amount = 0;
        value = 1000;
    }
    function getBalance() public view returns(uint) {
        return value;
    }
    function getAmount() public view returns(uint) {
        return amount;
    }
    function send(uint newDeposit) public {
        value = value - newDeposit;
        amount = amount + newDeposit;
    }
}
```

Copy Server RPC yang ada Ganache



The screenshot shows the Ganache interface. At the top, there is a navigation bar with tabs: ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, LOGS, and a search bar. Below the navigation bar, there are several status indicators: CURRENT BLOCK (2), GAS PRICE (20000000000), GAS LIMIT (6721975), HARDFORK (MUIRGLEACIER), NETWORK ID (5777), and an RPC SERVER entry (HTTP://127.0.0.1:7545). The RPC SERVER entry is circled in red. To the right of the RPC SERVER, the MINING STATUS is listed as AUTOMINING. On the far right of the top bar, there are buttons for WORKSPACE (PLASTIC-JELLY), SWITCH, and settings. Below the top bar, there is a mnemonic phrase: similar hand screen dizzy educate ticket motor remove conduct lumber toy dumb. To the right of the mnemonic is an HD PATH: m/44'/60'/'0'/'0/account_index. The main content area displays three accounts with their addresses, balances, transaction counts, and indices. Each account has a key icon next to it.

ADDRESS	BALANCE	TX COUNT	INDEX	
0x4fbe644c9db05a7a28c64d1e8b2a11436e1dcf	100.00 ETH	2	0	🔑
0xaE93b5887E32F473Ac1F3Ff5d79aDda2ff112Bb	100.00 ETH	0	1	🔑
0x1294d8b57542257A0BE47d185f0aDA37f42D2163	100.00 ETH	0	2	🔑

Lakukan Deploy

The screenshot shows a deployment interface for an Ethereum smart contract. On the left, there are input fields for GAS LIMIT (set to 3000000), VALUE (set to 0 Wei), and a CONTRACT dropdown menu showing "MyContract - mycontract.sol". A prominent orange button labeled "Deploy" is highlighted with a red circle. Below it is a checkbox for "Pubush to IPFS". At the bottom, there are two buttons: "At Address" and "Load contract from Address". On the right, the actual Solidity code for the contract is displayed:

```
10     value = 1000;
11 }
12 function getBalance() public
13     return value;
14 }
15 function getAmount() public v
16     return amount;
17 }
18 function send(uint newDeposit
19     value = value - newDeposit
20     amount = amount + newDepo
21 }
22 }
23 }
```

Below the code, there are some transaction-related controls: a dropdown arrow, a magnifying glass icon, a "0" balance indicator, a checkbox for "listen on all transactions", and a search bar.

Coba melakukan transaksi etherium

The screenshot shows a blockchain development environment with the following components:

- Debugger:** A sidebar titled "Transactions recorded" with a count of 2.
- Deployed Contracts:** A list showing "MYCONTRACT AT 0X74E...14E5A (BL)".
 - A button labeled "send" with the value "100" is highlighted with a red circle.
 - Other buttons include "getAmount" and "getBalance".
- Smart Contract Code:** A code editor on the right showing the following Solidity code:

```
12     function getBalance() public
13         return value;
14     }
15     function getAmount() public view
16         return amount;
17     }
18     function send(uint newDeposit) public
19         value = value - newDeposit;
20         amount = amount + newDeposit;
21     }
22 }
23 }
```
- Low level interactions:** A section at the bottom left.
- Transact:** A button at the bottom center.
- Logs:** A log pane at the bottom right showing a transaction record:
 - Block: 2
 - txIndex: 0
 - from: 0x4ff...
 - value: 0 wei
 - data: 0xa52...00064

Sudah berhasil melakukan transaksi, terlihat pada history sudah muncul di software ganache

The screenshot shows the Ganache application window on a Mac OS X desktop. The window title is "Ganache". The top menu bar includes "File", "Edit", "View", "Window", and "Help". The top right corner displays system status: battery at 97%, signal strength, and the date/timestamp "Tue 28 Jun 22.54.16". The main interface has a dark theme with orange highlights. At the top, there are navigation icons for "ACCOUNTS", "BLOCKS", "TRANSACTIONS" (which is selected), "CONTRACTS", "EVENTS", and "LOGS". A search bar says "SEARCH FOR BLOCK NUMBERS OR TX HASHES" with a magnifying glass icon. Below the header, a row of status indicators includes: "CURRENT BLOCK 2", "GAS PRICE 20000000000", "GAS LIMIT 6721975", "HARDFORK MUIRGLEACIER", "NETWORK ID 5777", "RPC SERVER HTTP://127.0.0.1:7545", "MINING STATUS AUTOMINING", "WORKSPACE PLASTIC-JELLY", a "SWITCH" button, and a gear icon.

TX HASH
0x8c7db06665ed13967ff68bbfd9a73ba582bdbbd56c3b93c280921a84c14cbb43 CONTRACT CALL

FROM ADDRESS	TO CONTRACT ADDRESS	GAS USED	VALUE
0x4fFBE644C9dBa05a7A28C64d1E8B2A11436E1DCF	0x74E915acCe75903B206aF6eB44877Ae7A1914E5a	48072	0

TX HASH
0xd4266343f2cafcb8c321e7881be4f5b943a9ea867a971888912114d1c869de79 CONTRACT CREATION

FROM ADDRESS	CREATED CONTRACT ADDRESS	GAS USED	VALUE
0x4fFBE644C9dBa05a7A28C64d1E8B2A11436E1DCF	0x74E915acCe75903B206aF6eB44877Ae7A1914E5a	132514	0