

Forum:[Hack Forums](#)

Thread: [Learn how to deploy Flash Loans with USDT](#)

0ppa - August 29, 2025, 9:34 am

? FLASH USDT: The Smart Way to Manipulate Stablecoin Liquidity (DeFi Flash Loan Blueprint)

DeFi isn't a playground. It's a warzone. Learn how to deploy Flash Loans with USDT and make the blockchain work for you without upfront capital.

????????????????

? WHAT IS USDT? ?

????????????????

Tether (USDT) is a stablecoin pegged to the US Dollar — used across exchanges to hedge volatility. It's one of the most traded assets in crypto and a prime candidate for liquidity manipulation, arbitrage, and gas-efficient flash loan operations.

????????????????

? WHAT IS A FLASH LOAN? ?

????????????????

A Flash Loan is an uncollateralized loan executed and repaid within the same block. No collateral. No delay. You borrow millions, execute strategy, repay instantly. Use cases:

[+] Arbitrage between DEXs

[+] Liquidation exploits

[+] Price oracle manipulation (with skill)

????????????????

? PREREQUISITES (NO NOOBS) ?

????????????????

Languages: Solidity or Vyper

Dev Tools: Hardhat or Truffle

Chains: Ethereum, Arbitrum, Polygon, Fantom

Protocols: Aave v2+, dYdX, Uniswap v3

????????????????

? ENVIRONMENT SETUP ?

????????????????

? Step-by-Step: Dev Setup ([Click to View](#))

1. Install Node.js

-> <https://nodejs.org>

2. Install Hardhat:

Code `npm install --global hardhat`

3. Create Your Project:

Code `mkdir FlashLoanProject`

`cd FlashLoanProject`

`npx hardhat`

4. Choose template -> Basic Sample Project

5. Add dependencies:

Code `npm install @aave/protocol-v2 @openzeppelin/contracts`

????????????????????

? FLASH LOAN CONTRACT (USDT) ?

????????????????????

? Solidity Template – Minimal FlashLoan for USDT ([Click to View](#))

Code // SPDX-License-Identifier: MIT

`pragma solidity ^0.8.0;`

`import "@openzeppelin/contracts/token/ERC20/IERC20.sol";`

`import "@aave/protocol-v2/contracts/interfaces/ILendingPool.sol";`

`contract FlashLoan {`

`ILendingPool public lendingPool;`

`address public USDT = 0x...; // Replace with actual USDT address`

`address public owner;`

`constructor(address _lendingPool) {`

`lendingPool = ILendingPool(_lendingPool);`

`owner = msg.sender;`

`}`

`function executeFlashLoan(uint amount) external {`

`lendingPool.flashLoan(address(this), USDT, amount, "");`

`}`

`function executeOperation(`

`address _reserve,`

`uint256 _amount,`

`uint256 _fee,`

`bytes calldata _params`

`) external {`

`// INSERT STRATEGY HERE (arbitrage, swap, etc.)`

`repayLoan(_reserve, _amount, _fee);`

`}`

`function repayLoan(address _reserve, uint256 _amount, uint256 _fee) internal {`

`uint totalRepayment = _amount + _fee;`

`IERC20(_reserve).approve(address(lendingPool), totalRepayment);`

`}`

}

????????????????
? DEPLOY & EXECUTE ?
????????????????

? Deployment Flow ([Click to View](#))

1. Deploy smart contract to testnet (Rinkeby, Goerli, etc.)
2. Use Remix, Hardhat scripts, or Web3 to deploy.
3. Fund wallet with ETH for gas.
4. Interact with contract: run `executeFlashLoan`.

? Always test on testnet first. Flash loans = instant losses if logic fails.

????????????????
? RISK ZONE ?
????????????????

- ? Gas Fees: High volatility = expensive failures
- ? Logic Bugs: One misstep, one wrong swap -> entire loan fails
- ? Price Fluctuation: By the time trade executes, prices may move
- ? Protocol Updates: Aave/Uniswap contract changes = breaks your logic

????????????????
? FINAL WORD (READ THIS) ?
????????????????

This isn't some YouTube-tier tutorial.

Flash USDT operations = real power. With zero upfront capital, you simulate being a whale for one block. You can extract value, liquidate pools, or arbitrage volatility.

Any feedback or tools? Drop below.

SUI-3183372 - August 29, 2025, 10:00 am

Hard to tell if OP is malicious and trying to own unsuspecting users, or if OP is just dumb.

The code is very incomplete and won't work anyways, and anyone looking to use this isn't going to know how to make it operational, so I lean towards dumb.

1. You're missing access controls, ANYONE can call your contract and trigger flashloans in unintended ways.
2. This is seemingly made for Aave but you haven't done anything to ensure compatibility with Aave. This is missing 75% of the parameters that Aave requires for flashloans. This won't compile or execute properly on v2 or v3.
3. You have no repayment guarantees, your code only offers approvals, it doesn't ensure balance.

4. Missing return values for necessary Aave callbacks. Again, not compatible with v2 or v3.

Omniscient - *August 29, 2025, 2:28 pm*

Dangerous for anyone to deploy this. Do so at your own risk.

The Joker - *December 18, 2025, 2:50 pm*

This is what happens when you go straight to chat GPT after reading articles about Crypto scams..