Deploy Your Own Cryptocurrency

Hands-On ERC-20 Lab & Deep Dive

Standalone Technical Lab

"From zero to token in 15 minutes"

Prof. Dr. Joerg Osterrieder

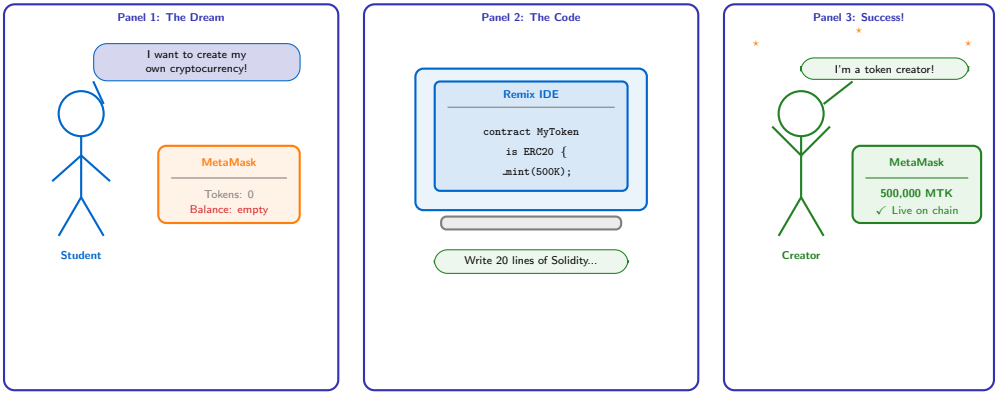
University Lecture Series

March 13, 2026

Part 1: Deploy Your Own Cryptocurrency

A step-by-step hands-on lab using Remix IDE and Sepolia testnet

The Mission



Creating a token is writing a smart contract — anyone can do it in minutes.

Step 1 — The Complete Contract

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract MyToken is ERC20, Ownable {
    uint256 public constant MAX_SUPPLY
        = 1_000_000 * 10**18;

    constructor()
        ERC20("MyToken", "MTK")
        Ownable(msg.sender)
    {
        _mint(msg.sender, 500_000 * 10**18);
    }

    function mint(address to, uint256 amount)
        public onlyOwner
    {
        require(
            totalSupply() + amount <= MAX_SUPPLY,
            "Exceeds cap"
        );
        _mint(to, amount);
    }

    function burn(uint256 amount) public {
        _burn(msg.sender, amount);
    }
}
```

Name & Symbol
"MyToken", "MTK"
Human-readable identity

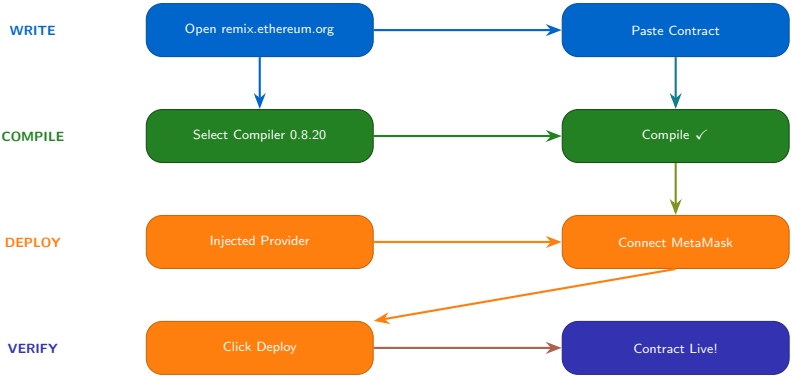
Initial Supply
500K of 1M cap (50%)
Minted to deployer

Mint (Owner Only)
Can mint remaining 500K
Enforces MAX_SUPPLY cap

Burn (Anyone)
Any holder can burn
their own tokens

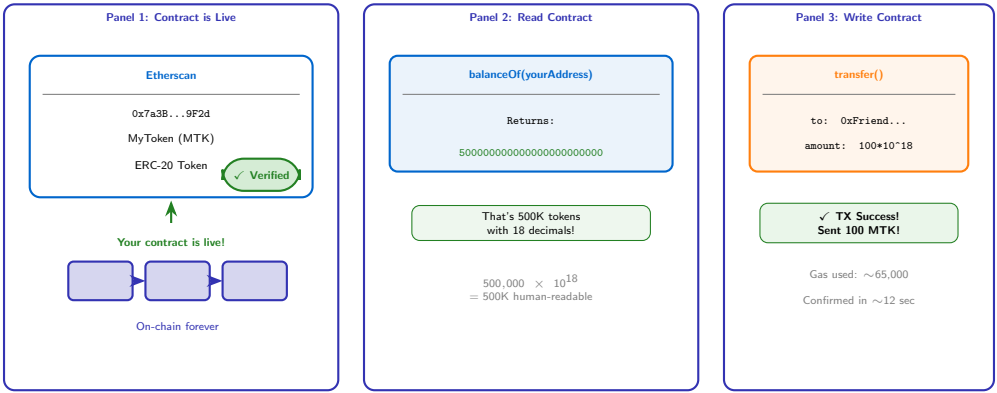
This contract gives you a capped, mintable, burnable ERC-20 token — 500K minted now, 500K mintable later.

Step 2 — Deploy on Remix



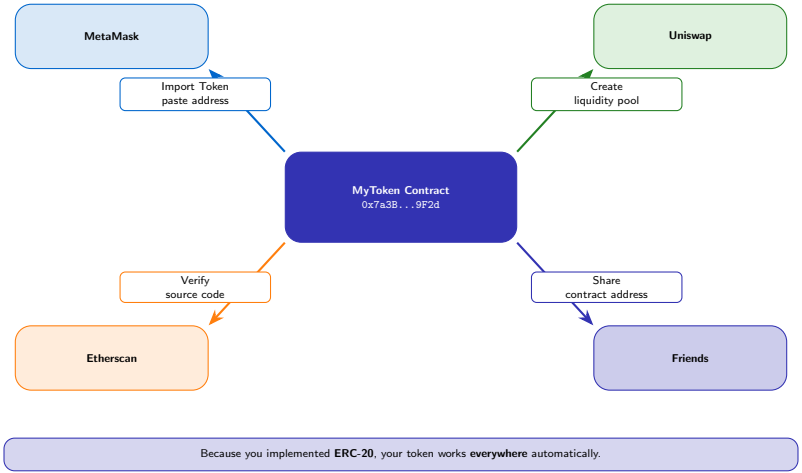
Remix IDE handles compilation and deployment — no local tooling required.

Step 3 — Verify & Interact



Once deployed, anyone can read your contract and interact with your token.

Step 4 — Add to MetaMask & Share



ERC-20 compliance means instant compatibility with the entire Ethereum ecosystem.

Your Token Deployment Recipe

Token Deployment Recipe

Ingredients:

- ✓ MetaMask wallet (with Sepolia test ETH)
- ✓ Web browser (Chrome or Firefox)
- ✓ Remix IDE (remix.ethereum.org)
- ✓ Contract code (from Step 1)

Steps:

1. Open `remix.ethereum.org` in your browser
2. Create a new file → name it `MyToken.sol`
3. Paste the complete contract code (from Frame 4)
4. Select Solidity compiler version `0.8.20`
5. Click "Compile MyToken.sol" → wait for ✓
6. Switch Environment to "Injected Provider – MetaMask"
7. Ensure MetaMask is on **Sepolia** testnet
8. Click "Deploy" → confirm transaction in MetaMask
9. Copy your contract address from the console
10. Go to `sepolia.etherscan.io` → verify it's live!

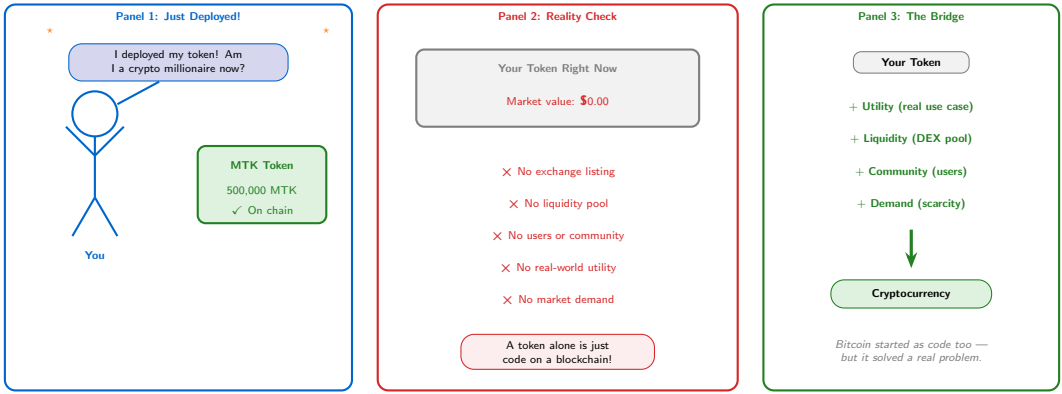
Yield: 1 fully functional ERC-20 token on Sepolia testnet

Prep time: 5 min • Deploy time: 2 min • Verification: 1 min • Total: under 15 minutes

Difficulty: Beginner • Cost: Free (testnet) • Serves: unlimited tokens

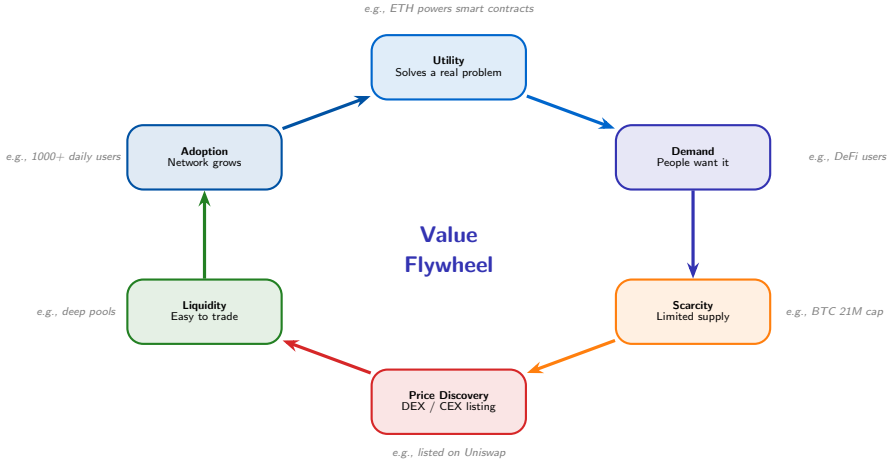
Follow this recipe step by step — you will have your own token deployed in under 15 minutes.

Token ≠ Cryptocurrency — What's Missing?



A token is the technical foundation — utility, liquidity, and community transform it into a cryptocurrency.

How Tokens Get Value — The Value Flywheel



Value is not created by deploying a token — it emerges when the flywheel of utility, demand, and liquidity starts spinning.

Making Money with Tokens — Five Revenue Models

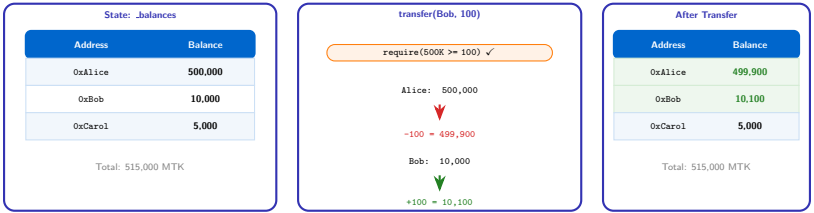
Most successful tokens combine multiple models — e.g., DEX listing + built-in fees + staking rewards. Your MTK token could start with Model 1 (list on Uniswap) and add Models 4–5 in a V2 contract upgrade.

Deploying a token costs nothing on a testnet — generating real value requires a strategy combining these revenue models.

Part 2: Deep Dive — The Technical Details

Formal explanations behind five key concepts from the visual introduction

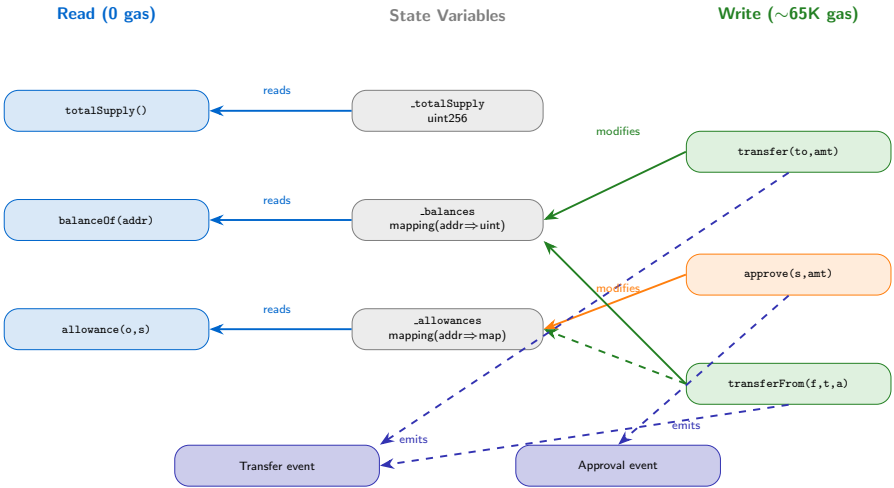
How Tokens Actually Work — The Balance Mapping




```solidity
mapping(address => uint256) private _balances;
function transfer(address to, uint256 amount) public returns (bool) {
    require(_balances[msg.sender] >= amount, "Insufficient");
    _balances[msg.sender] -= amount;
    _balances[to] += amount;
    emit Transfer(msg.sender, to, amount);
    return true;
}
```

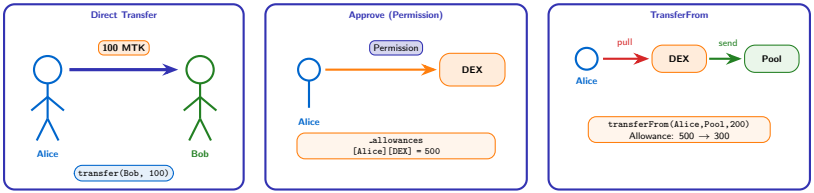
A token is just a mapping from addresses to balances — the smart contract is the ledger.

The ERC-20 Interface — All 6 Functions



Read functions cost zero gas; write functions require a transaction and pay gas.

Transfer vs Approve — The Full Mechanism



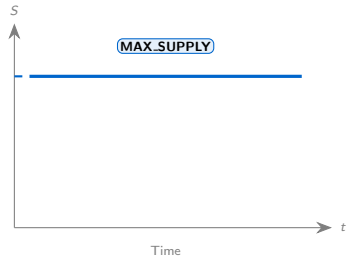
Race condition warning: set allowance to 0 first, then set to new value

```
mapping(address => mapping(address => uint256))
    private _allowances;
function approve(address spender, uint256 amount)
    public returns (bool) {
    _allowances[msg.sender][spender] = amount;
    emit Approval(msg.sender, spender, amount);
    return true;
}
```

approve()+transferFrom() is the delegation pattern that powers all of DeFi.

Token Supply Models — The Mathematics

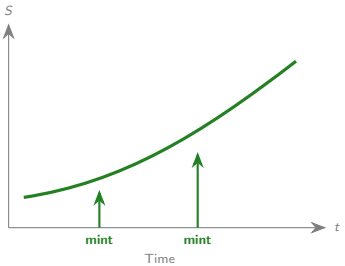
Fixed Supply



$$S(t) = S_0, \quad \forall\, t \geq 0$$

```
uint256 public constant
  MAX_SUPPLY = 1_000_000 * 10**18;
```

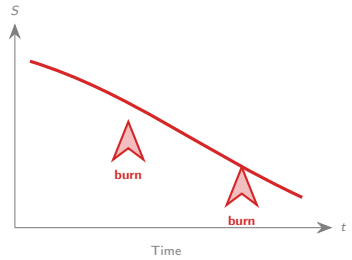
Inflationary



$$S_n = S_0 + \sum_{i=1}^{n} m_i$$

```
function mint(address to, uint256 a)
  onlyOwner { _mint(to, a); }
```

Deflationary

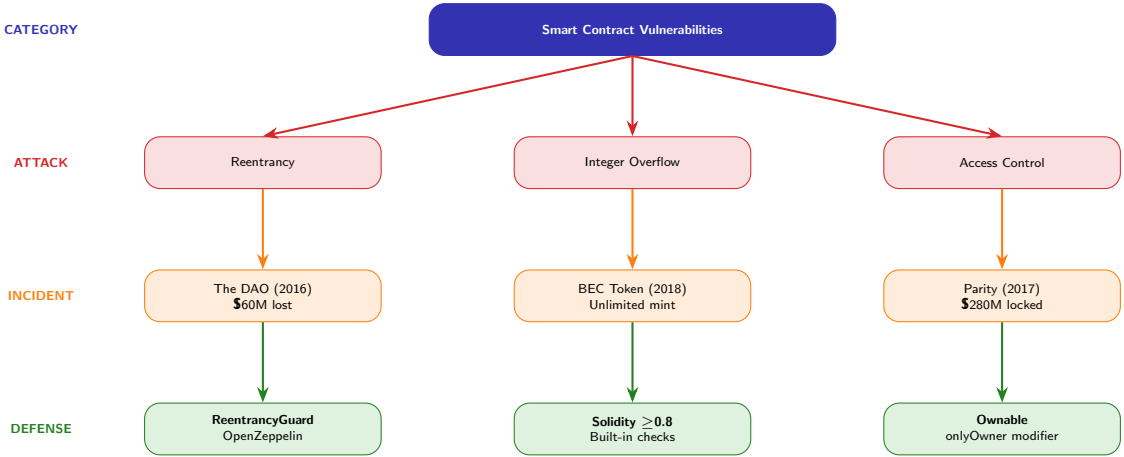


$$S_n = S_0 - \sum_{i=1}^{n} b_i$$

```
function burn(uint256 amt) public {
    _burn(msg.sender, amt); }
```

Tokenomics = mathematical models governing supply evolution via mint and burn events.

Security — Why OpenZeppelin?



Smart contract bugs are permanent and immutable — use audited libraries, always.

Key Terms for Beginners

Term	Definition
Blockchain	A shared, tamper-proof digital ledger that records every transaction across a network of computers.
Ethereum	A blockchain platform that supports smart contracts and is home to most ERC-20 tokens.
Smart Contract	A self-executing program on the blockchain that enforces rules automatically, without intermediaries.
Token	A digital asset created by a smart contract on an existing blockchain (unlike a coin, which has its own chain).
ERC-20	The standard interface (6 functions + 2 events) that all fungible tokens on Ethereum must implement.
Fungible	Interchangeable; one token equals any other of the same type, like dollar bills.
Solidity	The programming language used to write smart contracts on Ethereum.
Remix IDE	A free, browser-based tool for writing, compiling, and deploying Solidity smart contracts.
Gas	A fee (paid in ETH) for executing operations on Ethereum; every transaction costs gas.
Wallet	Software (e.g., MetaMask) that stores your private keys and lets you interact with the blockchain.

Term	Definition
Deploy	Sending your compiled contract to the blockchain, creating it at a unique address forever.
Testnet	A practice blockchain (e.g., Sepolia) where you can deploy and test for free, without real money.
DEX	Decentralized Exchange. A protocol (e.g., Uniswap) that lets users trade tokens directly from their wallets, without intermediaries.
OpenZeppelin	A library of audited, battle-tested smart contract code that prevents common security bugs.
Mapping	A Solidity data structure (like a dictionary) that stores key-value pairs, e.g., address $\rightarrow$ balance.
Mint	Creating new tokens, increasing the total supply. Typically restricted to the contract owner.
Burn	Permanently destroying tokens, decreasing the total supply.
Approve	Granting another address (e.g., a DEX) permission to spend your tokens on your behalf.
Tokenomics	The economic design of a token: supply model, distribution, incentives, and value.
Etherscan	A block explorer website where you can view transactions, contracts, and token balances.

Master these 20 terms and you have the vocabulary to understand any ERC-20 project.

Key Takeaways

Deploy Your Own Cryptocurrency: Summary

1. **Deploy in 15 minutes:** OpenZeppelin + Remix + Sepolia = your own token live on-chain.
2. **The balance mapping is the heart:** `mapping(address => uint256)` is the entire ledger.
3. **6 functions + 2 events = universal compatibility** with every wallet and DEX.
4. **approve()+transferFrom() enables DeFi composability** — the delegation pattern.
5. **Supply model math determines token economics** — choose fixed, inflationary, or deflationary.

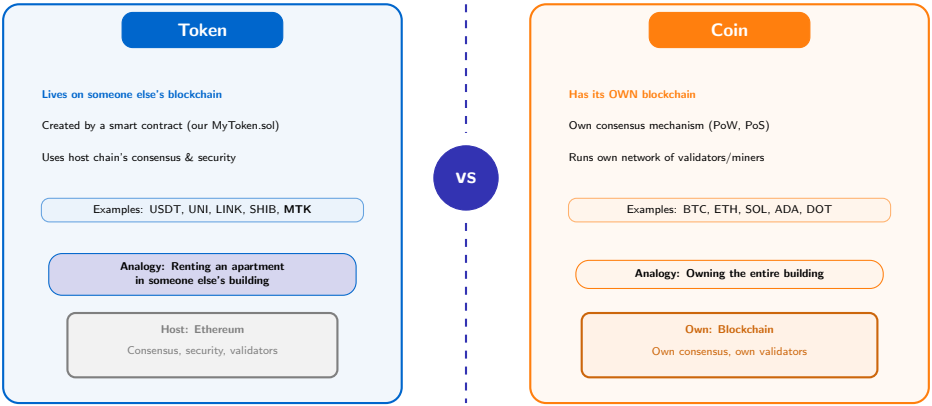
Next: Token economics deep dive and real-world deployment strategies.

You now have the knowledge to create, deploy, and explain your own ERC-20 token.

Part 3: From Token to Cryptocurrency

Understanding the difference and the path to building real digital currencies

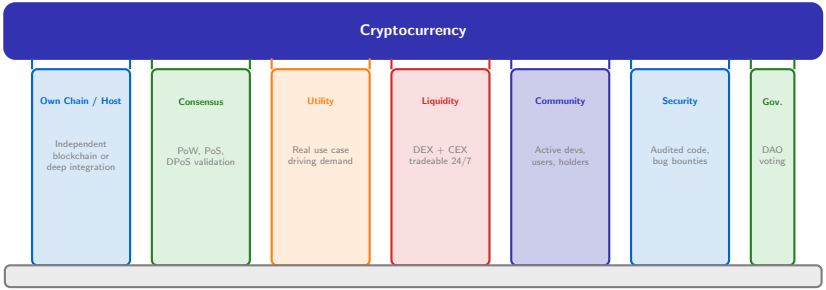
Token vs Coin — The Fundamental Difference



Your MTK is a token (lives on Ethereum). To become a cryptocurrency, it would need its own blockchain — but that's rarely necessary.

Most successful crypto projects start as tokens on Ethereum and only build their own chain if they outgrow it.

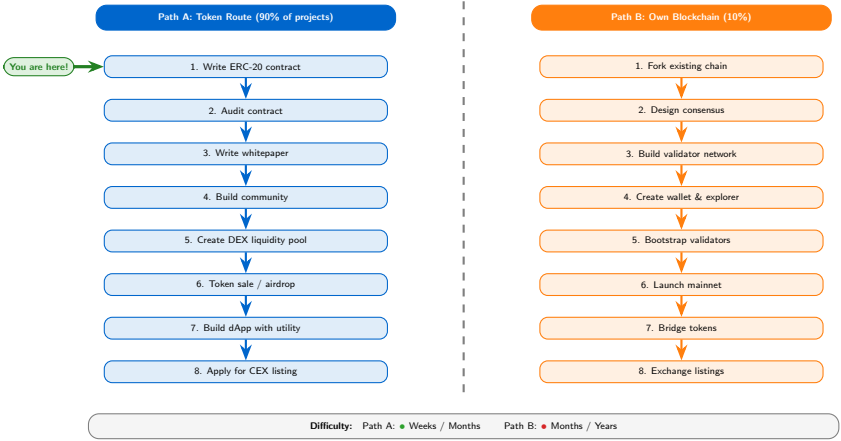
What Makes a Real Cryptocurrency? — The Seven Pillars



	MTK (your token)				ETH (cryptocurrency)		
	1–2 pillars filled				All 7 pillars filled		
MTK	✓	×	×	×	×	×	×
ETH	✓	✓	✓	✓	✓	✓	✓

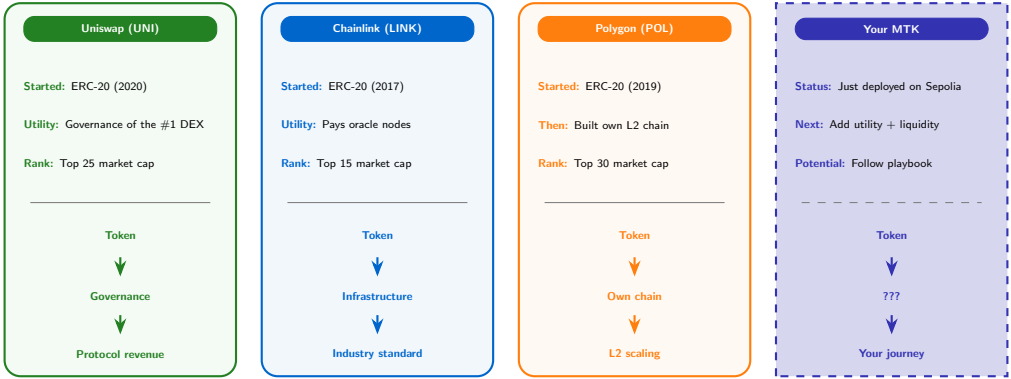
A token becomes a cryptocurrency not by technical means alone — it requires utility, community, and trust built over time.

The Cryptocurrency Creation Roadmap — Two Paths



Most successful projects (UNI, LINK, AAVE) took Path A — starting as tokens on Ethereum and never needing their own chain.

Real Success Stories — Tokens That Made It

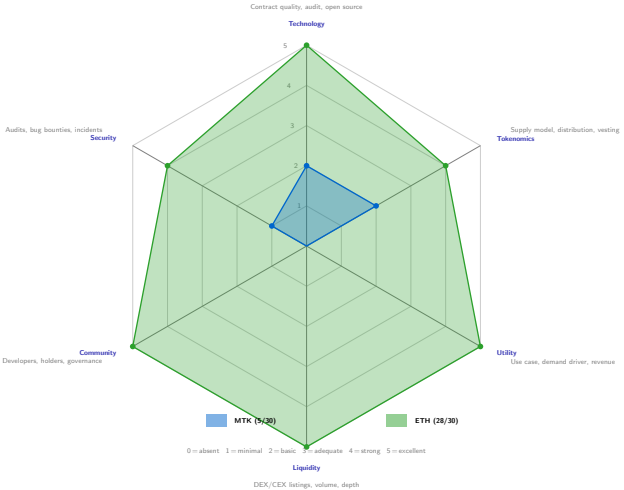


Every major cryptocurrency project started exactly where you are now — with a smart contract and an idea.

Part 4: The Analysis Framework

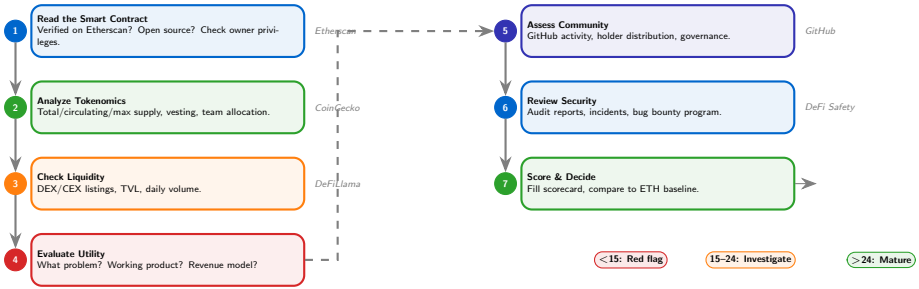
A reproducible methodology for evaluating any token or cryptocurrency project

The Token Analysis Scorecard — Six Dimensions



Score any project on these six dimensions — a total below 15/30 is a red flag, above 24/30 suggests a mature project.

Step-by-Step Analysis Methodology



Follow these seven steps for every project you evaluate — this is your reproducible due-diligence checklist.

Case Study — Scoring MTK vs. UNI

MTK Token (Our Project)

Dimension	Score	Reason
Technology	2/5	OZ base, unaudited
Tokenomics	2/5	Capped, no vesting
Utility	0/5	No use case yet
Liquidity	0/5	Not listed
Community	0/5	Classroom only
Security	1/5	OZ base, no audit
Total	5/30	**Early stage**

Uniswap (UNI)

Dimension	Score	Reason
Technology	5/5	Battle-tested, open
Tokenomics	4/5	Clear distribution
Utility	5/5	Governs #1 DEX
Liquidity	5/5	All major exchanges
Community	5/5	Millions, active DAO
Security	4/5	Multiple audits
Total	28/30	**Mature crypto**

Your MTK can grow from 5/30 to 28/30 — follow the roadmap in Frame 23

This is how you turn a subjective "is this token good?" into an objective, reproducible analysis.

Red Flags and Green Flags

Red Flags — Warning Signs

- ✗ Unverified contract (not open source)
- ✗ No audit report available
- ✗ Anonymous team, no track record
- ✗ Top 10 wallets hold $>80\%$ of supply
- ✗ No working product (vaporware)
- ✗ Promises of guaranteed returns
- ✗ Liquidity pool $<$ \$10K TVL
- ✗ No GitHub repository or activity
- ✗ Hidden `pause()` or `mint()` functions
- ✗ Whitepaper is just marketing

Green Flags — Positive Signs

- ✓ Verified, open-source contract
- ✓ Multiple independent audits
- ✓ Known team with track record
- ✓ Wide holder distribution ($<20\%$ top 10)
- ✓ Working product with real users
- ✓ Realistic roadmap, milestones met
- ✓ Deep liquidity on multiple DEXs
- ✓ Active GitHub, recent commits
- ✓ Time-locked admin (multisig)
- ✓ Clear tokenomics with vesting

Use this checklist as a quick pre-screen before diving into the full six-dimension analysis.

Your Analysis Toolkit — Essential Resources

Etherscan

Read contracts, check holders, verify transactions

Key: "Read Contract" tab

DeFiLlama

TVL, protocol revenue, chain comparison

Key: Dashboard

CoinGecko

Price, market cap, supply, exchange listings

Key: Supply info

DEX Screener

Real-time DEX data, liquidity pools

Key: Pair explorer

GitHub

Code activity, contributors, issue tracking

Key: Commits/month

Snapshot.org

Governance proposals, voting history

Key: Voter turnout %

Bookmark these six tools — they are all you need to perform a thorough due-diligence analysis on any crypto project.