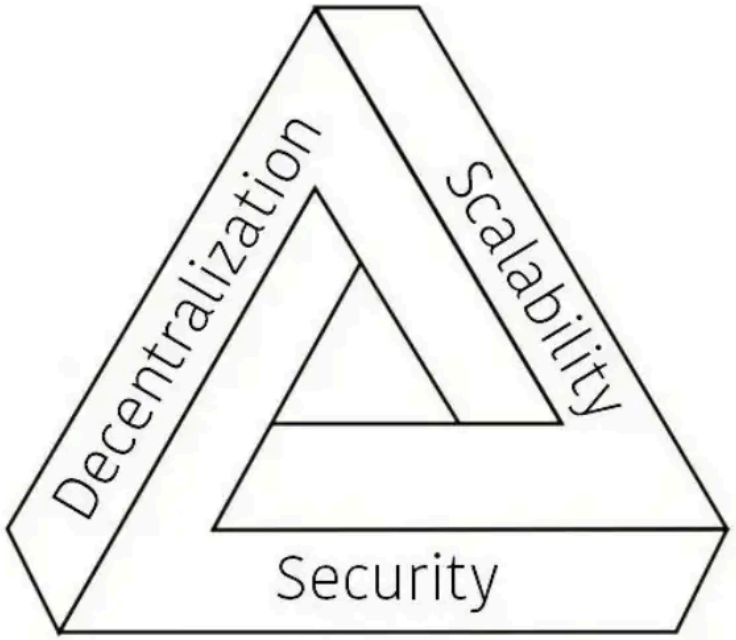
Fantom is an innovative Layer-1 blockchain designed to tackle the “blockchain trilemma,” which refers to the challenge of balancing scalability, security, and decentralization. It is EVM-compatible, enabling developers to build applications using Solidity and Vyper seamlessly.



Fantom has drawn significant attention due to its rapid innovation and growth. Let’s explore its journey, from its inception and technical advancements to strategic pivots, to uncover what makes this blockchain so compelling.

1.**The Early Days and Technical Innovations (2018 - 2019)**

Foundation and Vision: Fantom was founded by computer scientist Ahn Byung and is powered by Lachesis, an advanced asynchronous Byzantine Fault Tolerant (aBFT) consensus mechanism based on Directed Acyclic Graphs (DAGs). This innovation allows nodes to process transactions independently, ensuring fast confirmations without slowing the network. The result? High performance at low costs.

Mainnet Launch: In 2019, Fantom introduced the **Opera Mainnet**, which is compatible with the Ethereum Virtual Machine (EVM). By supporting Solidity and EVM, Fantom made it easy for developers to migrate dApps from Ethereum, earning it the nickname “Ethereum Killer.”

2.**The Rise of DeFi and Ecosystem Growth (2020 - 2021)**

DeFi Explosion: With its high performance and low transaction fees, Fantom emerged as a popular platform for DeFi applications. Key protocols like **SpookySwap** (a DEX) and **Geist Finance** (a lending platform) drove adoption. By 2021, during the bull market, Fantom's Total Value Locked (TVL) reached an impressive $8 billion.**Andre Cronje Joins**: DeFi pioneer Andre Cronje joined the Fantom Foundation, further accelerating its ecosystem growth. Known for launching **Yearn Finance**, Cronje's involvement attracted more developers and users to Fantom.

Market Adjustments and Challenges (2022)

The volatile crypto market and the rise of new Layer-1 blockchains like Solana and Avalanche presented challenges for Fantom. Competition intensified, prompting the network to explore new strategies for growth.

4.**The Sonic Labs Transition and Future Plans (2023 - Present)**

To enhance scalability, security, and user experience, Fantom announced the **Sonic Upgrade Initiative**, a comprehensive plan to optimize the network's performance and solidify its position in the blockchain space.

The launch of **Sonic Labs** marks a significant upgrade in the Fantom ecosystem. This transformation is not just a rebranding but a deep overhaul of the underlying blockchain technology. **Sonic boosts transaction throughput from 2,000 TPS to 10,000 TPS, reduces transaction finality from under 1 second to sub-second levels, and cuts average transaction costs to less than $0.01.** These enhancements lay the groundwork for a more efficient, secure, and scalable blockchain infrastructure, paving the way for the next generation of decentralized applications (dApps) and ecosystem innovation.Sonic Labs introduced **Sonic Gateway**, a groundbreaking cross-chain bridging solution designed to address the security and efficiency challenges of traditional bridges. Compared to other cross-chain solutions like Optimism and Polygon PoS, Sonic Gateway delivers significant advantages:

Faster Bridging Times: **Sonic Gateway** achieves asset bridging times of **10 minutes** (Ethereum to Sonic) and **1 hour** (Sonic to Ethereum), far surpassing Optimism's 1–7 days and Polygon PoS's 30–90 minutes.

2.**No Dispute Period**: Unlike Optimism, which requires a **7-day dispute period**, Sonic operates without any dispute window, ensuring a more efficient cross-chain experience for users.

3.**Decentralization and Security**: Sonic Gateway features fully decentralized ownership and bridge management, controlled by validators. In contrast, Optimism and Polygon PoS rely on centralized architectures, such as multisignature wallets or Sequencer-based control.

4.**Localized Fail-Safe Mechanisms**: Sonic Gateway includes built-in fail-safe mechanisms to enhance reliability, a feature absent in most other solutions.

5.**Heartbeat Monitoring**: Sonic integrates a cross-chain heartbeat mechanism to check status updates every **10 minutes to 1 hour**, reducing inter-chain information delays. Competing solutions lack similar functionality.

These features make Sonic Gateway a new benchmark in speed, security, and decentralization. It delivers a seamless, dependable cross-chain experience while enhancing connectivity between Sonic and other blockchain ecosystems. This advancement facilitates asset transfers and data exchange at an unprecedented level of efficiency and security.The transition from Fantom to Sonic represents not only technological refinement but also an insightful response to evolving blockchain industry demands. With its hybrid architecture, **FVM (Fantom Virtual Machine)**, and Sonic Gateway, Sonic Labs has built a high-performance, low-cost, and secure infrastructure. This robust foundation empowers the next wave of decentralized application innovation, setting new standards for blockchain ecosystems.
The Sonic Mainnet is scheduled to launch in **December 2024**. This guide covers the token mechanics for FTM and S Token and addresses key concerns for both regular users and developers during the migration process.

For the latest updates on the Sonic Mainnet, check the official announcements at: https://www.soniclabs.com

Native Tokens on the NetworkFTM is the native token of Fantom, with a total supply of **3.175 billion**, most of which is already unlocked and in circulation. Its primary uses include payments, staking, and governance.The S Token serves a similar purpose on Sonic. Its circulating supply will match the circulating supply of FTM.Upon the migration of the Fantom to Sonic, FTM will be converted to S Token at a **1:1 ratio** using official tools. For the first six months after the Sonic Mainnet launch, two-way conversion (FTM ↔ S) will be supported. After that, the conversion will become one-way (FTM → S).**Migration Preparation for Regular UsersConverting FTM to S Token**: After the Sonic Mainnet launch, users can

convert FTM to S Token via official tools. For FTM tokens held on centralized exchanges (CEX), users must first withdraw FTM to a Web3 wallet, then swap them for S Tokens at a 1:1 ratio.**Staking and Governance**: Once converted, S Tokens can be staked to earn rewards and used for governance voting.
Sonic is compatible with popular Web3 wallets like MetaMask. Users need to configure Sonic network parameters in their wallets to manage S Tokens effectively.
Migration Preparation for DevelopersSmart Contract Migration: Sonic is fully compatible with the Ethereum Virtual Machine (EVM), allowing applications developed for other EVM-based chains to be deployed on Sonic with minimal or no code changes. After the Mainnet launch, developers will need to redeploy their applications to the Sonic chain.
While Sonic is EVM-compatible, the EVM itself is continuously evolving with updates such as Istanbul, London, Shanghai, and Cancun (as of 2024). Each version introduces new features and optimizations. Sonic, as a new chain, will also undergo iterations, so complete code portability cannot be guaranteed. It is highly recommended to test applications on the Sonic Testnet before migrating.
Incentive Programs: Sonic offers developer support initiatives to encourage ecosystem growth. New projects deployed on Sonic can receive technical assistance and resource incentives.In the rapidly evolving blockchain industry, attracting top-tier developers and incentivizing them to build high-quality applications is a critical challenge. Sonic Labs addresses this with its innovative **Fee Monetization (FeeM)** mechanism—a groundbreaking approach to providing developers with direct, sustainable economic rewards, akin to the “revenue-sharing” model popularized by platforms like YouTube.

How FeeM Works

At its core, FeeM is a gas fee sharing mechanism that redistributes a portion of transaction fees back to developers. This model mirrors YouTube’s ad revenue sharing system, where creators receive a share of the platform’s earnings, fostering ecosystem growth and content innovation.

Implementation Details: Each transaction on Sonic incurs a gas fee. With FeeM, a percentage of this fee is allocated to the developers of the application responsible for the transaction. While the initial share on Fantom was **15%**, Sonic raises this significantly, allowing developers to earn up to ***90%*** of the transaction fees, ensuring higher and more sustainable revenues.**Opting In vs. Opting Out: The Difference**When developers build applications on Sonic, they have the option to join the FeeM program. This decision directly impacts how transaction fees are distributed for their application. The table below highlights the differences between applications that participate in FeeM and those that do not:

For FeeM Participants: Up to 90% of the transaction fees from their application are allocated to the developers, with the remaining 10% going to network validators.**For Non-Participants**: 50% of the transaction fees are burned, 45% are allocated to

validators, and the remaining 5% are deposited into the Ecosystem Vault.FeeM represents a stark contrast to the traditional gas fee models used by most blockchains. In conventional systems, gas fees paid by users typically go entirely to miners or validators as rewards. Sonic's FeeM model, however, redistributes this value, directly benefiting developers while still supporting validators and the broader network.By introducing this revenue-sharing mechanism, Sonic not only provides sustainable incentives for developers but also redefines the value distribution within blockchain ecosystems. This innovation positions Sonic as a developer-friendly platform, fostering long-term growth and aligning the interests of all stakeholders. Sonic's Fee Monetization (FeeM) mechanism introduces groundbreaking innovations in gas fee distribution, providing a stark contrast to Ethereum's traditional gas fee model. Let's look into a detailed comparison across two dimensions: **mechanism design**, **incentive model**.

Mechanism Design: Where User Gas Fees Go

Ethereum (PoS 2.0): In Ethereum, users pay gas fees composed of two parts: **Base Fee** and **Max Priority Fee**, as illustrated below:

Base Fee: Dynamically adjusted based on network congestion, this portion represents the minimum price for a transaction to be included in a block. It is burned to reduce Ethereum's total supply, creating a deflationary effect introduced by EIP-1559.

●**Priority Fee**: This acts as a tip for miners or validators, incentivizing them to prioritize transactions.

While designed to balance network demand and supply and ensure security, this model provides no direct rewards for developers, leaving them reliant on indirect benefits like token appreciation or application revenue.**Sonic's FeeM Model:** Sonic redefines gas fee distribution by allocating a significant share to developers, encouraging ecosystem growth. The distribution varies depending on whether the application participates in the FeeM program:

●**FeeM Applications**: Developers earn **up to 90%** of the gas fees from transactions involving their apps, with the remaining **10%** allocated to validators.

●**Non-FeeM Applications**: Fees are distributed similarly to Ethereum: **50% burned**, **45% to validators**, and **5% to the Ecosystem Vault**.

This redistribution prioritizes developer incentives while maintaining a functional reward mechanism for validators.

Incentive Model: Transparency and Sustainability

Ethereum: it's developers primarily earn through indirect means, such as:

●**Application Fees**: Revenues from DeFi trading fees or NFT marketplace commissions.

●**Token Economics**: Profit derived from the appreciation of native tokens or staking rewards.

These methods are influenced by market volatility and do not provide direct rewards from blockchain infrastructure.**Sonic:** Sonic's FeeM offers developers direct and transparent incentives:
Applications with high user interaction can earn significant shares from gas fees.

●Developers gain predictable revenue streams, reducing dependence on market-driven factors.

For example, a high-frequency payment DApp processing 1,000 transactions per day at a total gas fee of 100 S tokens could yield developers 15~90 S tokens daily under the FeeM program.
Sonic Gateway is a **decentralized cross-chain bridge** developed by Sonic Labs to facilitate secure asset transfers between Ethereum and Sonic. By leveraging Sonic's own validator network—where validators operate nodes on both chains—Sonic Gateway establishes a **secure and decentralized channel** between the platforms, ensuring users maintain complete control over their funds at all times.Moreover, Sonic Gateway is **designed for efficiency**. Transfers from Ethereum to Sonic take a maximum of 10 minutes, while transfers from Sonic to Ethereum are completed within 1 hour.

How Sonic Gateway Works

At its core, Sonic Gateway relies on **light clients** and interactions between blockchains, supported by the validator network. Let's break down the cross-chain process using the example depicted above:**Transferring Assets from Ethereum to Sonic (top path, left to right):**

1.**User Initiates Transfer**: User A sends 1 ETH to the Gateway smart contract on Ethereum. The Ethereum contract generates a **Proof of Receipt**, including the transaction's **Merkle Root Hash** and the corresponding Sonic chain block height as verification data.

2.**Validators Verify and Broadcast**: The validator network verifies the cross-chain transaction and submits the proof to the Gateway contract on Sonic.

3.**Asset Allocation**: Upon receiving the verification proof, the Sonic Gateway contract allocates the equivalent value of 1 ETH to User A's address on the Sonic blockchain.

Transferring Assets from Sonic Back to Ethereum (bottom path, right to left): The reverse process follows the same principles: The Sonic Gateway contract

generates proof for the Ethereum Gateway contract. Validators facilitate the proof submission, enabling User A to retrieve the equivalent asset on Ethereum.

Fail-Safe Mechanism

Sonic Gateway incorporates a robust **fail-safe mechanism** to protect users' assets in the unlikely event of Gateway failure. If Sonic or the Gateway encounters issues, users can recover their cross-chain assets on Ethereum.

Activation After 14 Days: If the Gateway ceases operations for 14 consecutive days, users can unlock their assets on Ethereum through the fail-safe mechanism. This 14-day fail-safe period is immutable, ensuring that once Sonic Gateway is deployed, neither Sonic Labs nor any third party can alter this feature.**Heartbeat Monitoring**: Sonic Gateway exchanges **"heartbeats"** between chains, which include Merkle roots and block heights from each blockchain. If the heartbeat signals stop for 14 days, the system triggers a Gateway failure alert, enabling users to reclaim their funds.

Traditional blockchain technologies often rely on linear chain structures, which limit scalability and performance due to constraints like block production speed and size. Sonic overcomes these limitations by adopting **Asynchronous Byzantine Fault Tolerance (aBFT)** combined with a **Directed Acyclic Graph (DAG)** architecture.

Asynchronous Byzantine Fault Tolerance (aBFT)

In distributed systems, aBFT is a robust fault-tolerance mechanism designed to address the challenges posed by network delays and malicious behavior, ensuring consensus is reached even in adversarial conditions.

The Byzantine Problem The Byzantine problem originates from a classic analogy: imagine a group of generals needing to coordinate an attack or retreat via communication. Some of the generals might be traitors, spreading false information. The group must reach a consensus to ensure success.In distributed networks, malicious nodes (e.g., hackers) or communication failures create similar issues. The goal of aBFT is to ensure consensus even if some nodes act maliciously or fail.**How aBFT Achieves Fault Tolerance**Sonic's fault tolerance leverages a **Proof of Stake (PoS)** mechanism. Validators participate in consensus based on their staked assets, validating and recording transactions. Validators sign transactions and append them to the DAG structure, ensuring every transaction undergoes multiple validations.**Fault Tolerance Level**: Sonic's PoS mechanism tolerates up to **1/3 of validators** acting maliciously or going offline while still achieving consensus. Validators are incentivized to act honestly to avoid penalties that affect their staked assets.**How aBFT Enables Asynchronous ConsensusNo Unified Time Requirement**: aBFT does not rely on a synchronized global clock. Each node processes transactions independently based on its own timeline, constructing the data structure incrementally. This eliminates bottlenecks caused by waiting for network-wide synchronization.**Eventual Message Consistency**: Even in asynchronous systems where message delivery may be delayed, aBFT ensures that the network achieves eventual consistency through mechanisms like voting or

aggregated signatures. As long as honest nodes form the majority, attackers cannot manipulate transaction ordering or disrupt consensus.
Traditional blockchains use a **linear structure** to record transactions, where each block is linked to the previous one, forming a single chain from the genesis block to the latest. However, this sequential design limits scalability. Transactions per second (TPS) are constrained, and during high network activity, unconfirmed transactions queue up, slowing down network response.A linear blockchain is akin to a single-lane road where cars (blocks) must proceed one at a time. Any slowdown affects the entire queue behind it.

What is Directed Acyclic Graph (DAG)?

In Sonic, DAG is the foundational structure for organizing and recording transactions. **Vertices (Nodes):** Represent transactions, containing transaction details and verification signatures. **Edges:** Indicate dependencies between transactions. For example, if transaction A must occur before transaction B, there will be a directed edge from A to B.DAG's structure naturally eliminates forks, a common issue in linear chains, because transactions do not form circular dependencies.**Parallel Execution Enabled by DAG**One of DAG's most significant advantages is **parallel processing**. Tasks are executed concurrently when their dependencies are met, significantly improving efficiency. This can be explained using the "boiling water and making tea" analogy:In the example: Tasks like "washing the kettle" and "Getting the tea leaves" can run independently. Only dependent tasks, such as "boil water" (requiring the kettle to be clean), follow a sequential order. By running independent tasks in parallel, the overall time is reduced to just slightly longer than the most time-consuming task.This same principle applies to DAG-based systems, where transaction processing achieves unparalleled efficiency.**DAG Sorting Mechanism**DAG enables multiple transactions to process concurrently rather than sequentially. However, maintaining network-wide consistency requires a **global transaction order**, achieved through **topological sorting**. Sonic implements this in three steps:

1.**Transaction Addition:** When a transaction is broadcast to the network, nodes add it to their local DAG, positioning it based on its hash and dependency relationships.

2.**Dependency Analysis:** Nodes analyze the DAG's topology to determine an execution order that respects all dependencies.

3.**Concurrent Processing:** Independent transactions (e.g., A and B) are processed simultaneously, while dependent transactions (e.g., C depends on A) wait for their prerequisites to complete.

To determine the execution order of transactions, DAG uses rules such as: **Timestamp Priority**(Older transactions are prioritized), **Reference Weight**(Transactions referenced by more subsequent transactions are given higher priority), **Voting Mechanism**(Consensus nodes vote on the final order of

transactions).For details on Sonic's specific DAG sorting algorithm, readers can explore this paper: On the Topological Ordering of DAG-based Consensus.

Sonic's consensus mechanism integrates **Asynchronous Byzantine Fault Tolerance (aBFT)** and **Directed Acyclic Graph (DAG)** to achieve high-performance and secure transaction validation. Below is a detailed breakdown of how Sonic processes and confirms transactions.

User Submits a Transaction: Users initiate transactions via the Sonic client. These transactions are broadcasted to the network and received by validators, who are responsible for processing them.

2.**Creation of Event Blocks:** Validators bundle received transactions into **Event Blocks** and add them to their local **DAG (Directed Acyclic Graph)**.

●**Event Blocks:** Each Event Block corresponds to a vertex in the DAG. It contains transaction data and references to other Event Blocks it depends on (dependencies).

●**Validation:** Before creating a new Event Block, validators verify the integrity of both the current block and a subset of already received Event Blocks to ensure consistency.

This decentralized approach eliminates the bottleneck of a sequential block creation process.

3.**Propagation of Event Blocks:** Validators use asynchronous communication to share their Event Blocks with others in the network.

●**Asynchronous Propagation:** Unlike traditional blockchains, Sonic does not require network-wide synchronization. Event Blocks can propagate independently, enabling higher concurrency.

●**Validation by Peers:** Upon receiving Event Blocks, validators independently verify and add them to their local DAGs. This ensures that every validator builds a consistent view of the network over time.

4.**Confirmation of Root Event Blocks:** During asynchronous propagation, when **more than two-thirds of validators** receive and validate an Event Block, it is promoted to a **Root Event Block**.

●**Root Event Blocks:** These blocks signify a key milestone in achieving global consensus. They act as anchors for ordering transactions and building the main chain.

●**Consensus Achieved:** Transactions within a Root Event Block are considered confirmed and immutable, ready to be incorporated into the main chain.

5.**Construction of the Main Chain:** The **main chain** is composed of all consensus-approved Root Event Blocks, arranged in chronological order.

●**Topological Sorting:** Validators use DAG's topology to order Root Event Blocks. This ensures all dependencies are satisfied and creates a consistent transaction sequence.

●**Global Finality:** The main chain provides an immutable and universally agreed transaction history, allowing users and applications to confidently query transaction records.

Sonic's consensus mechanism harmonizes DAG's parallelism with aBFT's resilience, offering a robust solution for high-speed, secure, and reliable blockchain operations.

Configuring RPC Information

Before claiming test tokens, you need to configure the Sonic test network. Since Sonic is **EVM-compatible**, it supports multiple wallets such as **Rabby**, **MetaMask**, **Coinbase Wallet**, and **Ledger**.Here, we'll use MetaMask as an example. Open your wallet, and go to **Select a network** → **Add a custom network**. Fill in the following RPC details:**Network Name**: Sonic Testnet**RPC URL**: https://rpc.testnet.soniclabs.com/**Chain ID**: 64165**Currency Symbol**: $S**Block Explorer URL**: https://testnet.soniclabs.comAfter configuring the network, your wallet balance will show **0 $S**. Don't worry, we'll move on to claim the tokens.

Claiming Test Tokens

You can claim Sonic's native test token ($S) and **ERC20 tokens** like Coral, Diamond, and Fluorite via the Sonic testnet explorer. Open Sonic Testnet Explorer. Connect your MetaMask wallet as prompted.

Click the **Request Sonic** button. Your wallet will prompt a signature request—confirm it. Optionally, you can also request ERC20 tokens such as Coral, Diamond, and others. After claiming these tokens, click **Add to MetaMask** to automatically import their contract addresses into your wallet for display.

Swapping ERC20 Tokens

Once you've claimed $S, Coral, and Diamond, you can try out Sonic's **Swap feature** to experience its fast transaction speed. Specify the tokens and amounts you want to swap. Confirm the transaction in your MetaMask wallet.
The transaction will complete swiftly, demonstrating the speed of Sonic's network. With these simple steps, you're all set to explore the Sonic testnet and its features!
In the previous section, we configured our wallet and received Sonic test tokens. Now, let's explore how to deploy a smart contract on the Sonic blockchain.

Sample Smart Contract

Here's a simple counter contract written in Solidity, compiled using version 0.8.19:

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

contract Counter {
    uint256 public number;

    function setNumber(uint256 newNumber) public {
        number = newNumber;
    }

    function increment() public {
        number++;
    }
}
```

Compiler and EVM Version: According to Sonic's official documentation, it is recommended to use Solidity **0.8.19** or earlier, and the EVM version should be set to **London** to match Sonic's current setup.
Note: The above configuration is accurate as of November 2024. Always check the official documentation for the latest recommendations.

Deployment Walkthrough

Compile the Contract: Use the HackQuest contract compiler by visiting this link: HackQuest IDESelect Solidity version **0.8.19** as the compiler. Once compiled, you'll see the output as shown below:
Connect MetaMask Wallet: In the deployment menu, select **Injected Provider - MetaMask** and connect your wallet. Ensure your wallet is set to the Sonic Testnet.
Deploy the Contract: Click the **Deploy** button. Your MetaMask wallet will prompt for confirmation. Once confirmed, the contract will be deployed to the Sonic blockchain.

View Transaction Details: Once deployed, you can obtain the contract address from the IDE. Use this address to view the transaction details on the Sonic Testnet block explorer:https://testnet.soniclabs.com/tx/0x5cf38f156d2a604da43ef41065ccffffc2e80a974af0eac67ce0681f1e910e55

Interact with the Contract: After deployment, you can interact with the contract directly from the IDE. For example, call the increment or setNumber functions to test the contract's functionality.

In the previous section, we successfully deployed a smart contract. Now, let's dive deeper into the deployment details.

Gas Cost Comparison

When deploying a transaction on Sonic (https://testnet.soniclabs.com/tx/0x5cf38f156d2a604da43ef41065ccffffc2e80a974af0eac67ce0681f1e910e55), the **Gas Price** was **3.5 GWei**. If you're unsure what this means, let's compare it by deploying the same contract on the **Sepolia network**.

By switching the network in your wallet to Sepolia and deploying the contract, we observe the following transaction (https://sepolia.etherscan.io/tx/0xe40ce48beb665eadfee59dc30f2990886bb060d26fc74705913833dcf7b59174) with a **Gas Price** of **10.6 GWei**. Since the same contract is deployed, the **Gas Usage** remains similar: **160,291** on Sonic and **160,325** on Sepolia.

From this comparison, we can see that Sonic's **Gas Price** is roughly **1/3** of Sepolia's. However, since Sonic's native token ($S) costs less than **1/3000** of ETH, the actual transaction cost on Sonic is significantly lower. According to the official figures, the cost per transaction is less than **$0.01**, making participation in Web3 activities on Sonic extremely affordable.

Compiler Version Considerations

Sonic's official documentation recommends using Solidity compiler versions **0.8.19 or earlier**, with the **EVM version** set to **London**, rather than the latest **Cancun**. Using a newer compiler version may result in errors, even if there's nothing wrong with your contract:

This indicates that Sonic's blockchain is built on Ethereum's **London upgrade**, requiring the EVM version to match **London**.

Understanding Compiler and EVM Versions

Compiler Version: The compiler converts your Solidity source code into bytecode for deployment on the blockchain. The pragma solidity directive in your contract specifies the compatible compiler versions. For example:

```
1pragma solidity ^0.8.26;
```

EVM Version: The Ethereum Virtual Machine (EVM) version determines the environment where the smart contract's bytecode runs. Each EVM upgrade introduces new features and optimizations. Common versions include Istanbul, London, Shanghai, and Cancun. For instance:

●**London** introduced the EIP-1559 fee mechanism.

●**Cancun** added calldata compression to reduce Layer-2 Rollup costs.

Choosing the correct versions ensures compatibility and avoids potential issues arising from differences in feature sets. For Sonic, it’s best to use the versions verified in their official documentation: **Solidity 0.8.19** and **EVM London**.By adhering to these recommendations, you can ensure smooth deployment and optimal performance of your smart contracts on Sonic. The low gas costs and compatibility features make Sonic a cost-effective and developer-friendly blockchain for Web3 applications.
DeFi redefined. Sonic is the highest-performing EVM blockchain, combining speed, incentives, and world-class infrastructure. Powered by $S.

March '25 Newsletter
Sonic Labs
Sonic Labs
Mar 12, 2025
3 min
The past month has brought our most content-rich newsletter in ages, centered on "the revival of DeFi."

Several major DeFi apps launched, Sonic's TVL reached record heights, new yield opportunities emerged, and our community grew significantly.

Let's dive into the highlights.

Sonic Summit

Announcing Sonic Summit.

📍 Vienna, Austria
📆 May 6–8, 2025

City-wide Sonic takeover of historic venues for three days of events, networking, coffee, cake, and more.

🎟️ Get your early bird ticket.

Latest Sonic News

Fee Monetization
FeeM went live on Sonic, offering developers 90% of the network fees generated by their apps.

Aave
Aave deployed on Sonic, reaching over $100 million TVL as one of the most trusted DeFi protocols.

Sonic × Gasly
Speed meets speed. We partnered with Pierre Gasly for the 2025 F1 World Championship.

Trust Wallet
Trust Wallet integrated Sonic. You can now send, receive, and store Sonic assets, and check your transaction history.

Sonic Notifications
Get personalized notifications from Sonic straight to your Telegram, Discord, or email, powered by Notifi.

Arkham × Sonic
We're partnering with Arkham to bring Sonic to its intelligence platform. More data, more insights, smarter trading.

S on Solana
Users on Solana can now get S and bridge to Sonic when they're ready to experience the leading DeFi ecosystem.

USDT
We deployed USDT on Sonic. Bridge from Ethereum on the Sonic Gateway and explore our DeFi ecosystem.

Euler Finance
Euler Finance went live on Sonic, offering lending and borrowing with both incentives and looping.

scBTC
scBTC launched on Sonic, a yield-bearing, collateralized Bitcoin asset, backed by LBTC and eBTC.

Stargate
Stargate Finance integrated Sonic, allowing users to bridge USDC from various other chains.

1-Click Validator
We introduced 1-click validator deployment, powered by Google Cloud and BCW. Set up in seconds and earn rewards.

Pendle
Pendle went live on Sonic. Earn yield using assets like wstkscUSD, wstkscETH, stS, and wOS initially.

Royco
Royco went live on Sonic, a marketplace where apps can offer you incentives for your liquidity.