Whitepaper

Welcome to the Galaxy of Calamari Ingots (BESKAR)

A Universal Currency for All Space Sagas

Welcome to the new frontier of cosmic finance, where the legacies of interstellar trade and commerce are celebrated and continued. Calamari Ingots, known on exchanges as BESKAR, are not just a cryptocurrency; they are an homage to every galaxy far, far away and the complex economies that drive them—from the bustling trade routes of *Star Wars* to the sprawling cosmic empires of recent anime.

Celebrate the Legacy of Space Sagas

Space sagas, from epic film series to captivating anime, have always stretched the limits of our imagination. They've introduced us to worlds where entire galaxies interact, trade, and sometimes conflict over resources. Central to these narratives is often a universal or galaxy-wide payment system, enabling diverse species and civilizations to engage in commerce and diplomacy. Calamari Ingots (BESKAR) embody this spirit—a currency designed for the vast and interconnected digital cosmos.

Embrace the Future of Universal Transactions

Standing at the brink of our own technological evolution, Calamari Ingots aim to bridge the fictional with the real. Crafted for the age of digital economies, they ensure seamless transactions across borders, planets, and star systems. Whether you're trading on Earth or planning for transactions in the distant Nebula markets, Calamari Ingots provide a stable, secure, and universally accepted medium of exchange.

Join Our Galactic Community

Step into a community of enthusiasts, traders, and dreamers. Calamari Ingots aren’t just about financial transactions; they’re about building a network of users who share a passion for the cosmic tales that have inspired generations. Participate in discussions, trade tips, and celebrate new releases and classic favorites of space saga films and anime.

Ready to Explore?

Your adventure begins here. Navigate through our site to learn more about Calamari Ingots, how to acquire them, and how to use them in your daily and intergalactic transactions. Together, let's turn the page on the future of currency and step into a universe where commerce knows no bounds.

About

Welcome to the Calamari Ingots Meme Coin Page

Discover the dynamic world of Calamari Ingots, the meme coin known as BESKAR on exchanges. Embark on a journey where you can actively participate in shaping the future of digital commerce across galaxies.

Become a Liquidity Provider

Start your adventure by becoming a liquidity provider. By contributing to liquidity pools, you not only facilitate smoother transactions within the ecosystem but also earn rewards. Learn how to add your BESKAR to pools on platforms like PancakeSwap and earn a portion of the trading fees in return.

Stake for Rewards

Take advantage of our staking options to maximize your holdings. Stake your Calamari Ingots (BESKAR) and watch as they grow with competitive annual returns. It's simple, secure, and rewarding—a perfect way to increase your stake in the galaxy's future.

Try Our LP Staking Contract

For liquidity providers looking to further maximize their returns, try our LP staking contract. This feature allows you to stake your LP tokens earned from providing liquidity, enhancing your reward potential through additional staking benefits.

Explore Anonymity with Our Smart Contract

For those who value privacy, dive into our anonymity contract currently available in Remix, with a dedicated user interface set to launch in the coming phases. This feature allows you to conduct transactions with enhanced privacy, ensuring your activity remains confidential.

Coming Soon: Loan Contract

Looking ahead, Phase 2 or 3 will introduce a groundbreaking loan contract. This addition will enable you to leverage your BESKAR holdings to secure loans, empowering you with more flexibility in how you manage and utilize your assets. Stay tuned for this exciting development.

Ready to Join the Calamari Ingots Community?

Embark on your journey today! Dive into the world of Calamari Ingots and take part in a financial revolution that spans across the universe. Whether you’re staking, providing liquidity, or exploring our advanced contract features, Calamari Ingots offers you a gateway to the future of finance.

Explore, engage, and expand your horizons with Calamari Ingots, where your financial adventure awaits!

Roadmap

Phases for Calamari Ingots (BESKAR)

Phase 1: Foundation and Launch

- **Launch Activities**: Kick off with a robust token launch to secure Calamari Ingots' position in the market. This includes provisioning initial liquidity to ensure seamless transactions for early adopters.
- **Community Engagement**: Develop a vibrant community foundation through targeted social media campaigns, strategic airdrops to reward early supporters, and forging partnerships with key fintech and crypto entities. These efforts aim to foster a strong and engaged user base excited about the potential of Beskar.

Phase 2: Ecosystem Development

- **Anonymity Vault Integration**: Introduce an Anonymity Vault to enhance transactional privacy for all users, reinforcing Calamari Ingots' commitment to security and privacy. This feature is crucial for users prioritizing discretion in their digital transactions.
- **Loan System Implementation**: Roll out an innovative loan system that utilizes Beskar as collateral, providing flexible financial solutions to users. This system is designed to support liquidity within the ecosystem while offering additional utility to the token.

Phase 3: Expansion and Sustainability

- **Liquidity Milestones**: Set and achieve critical liquidity milestones that will initiate a staged liquidation process. This strategic move is designed to ensure the long-term sustainability and growth of the Calamari Ingots ecosystem by managing the supply effectively.
- **Ecosystem Expansion**: Continue to build out the Beskar ecosystem with the introduction of new features and utilities. This includes potential integrations with other platforms, expansion into new markets, and ongoing development to keep pace with evolving blockchain technology.

Phase 4: Global Recognition and Operational Excellence

Achieving $20M in Liquidity: Reach and surpass a liquidity threshold of $20 million, which will bolster market stability and investor confidence.

Major Exchange Listings: Aim for listing on major exchanges like Binance and Coinbase, enhancing visibility, accessibility, and trading volume of Beskar.

Regulatory Compliance and Partnerships: Strengthen compliance with global regulatory standards and forge strategic partnerships with major financial institutions to enhance credibility and operational reach.

Phase 5

- **Advanced Feature Rollouts**: Introduce more complex financial instruments and services, such as derivatives and futures, linked to Beskar to diversify its utility and appeal.
- **Sustainability Initiatives**: Implement initiatives aimed at promoting sustainability within the ecosystem, such as green mining practices and contributions to carbon offset programs, aligning with global efforts to combat climate change.

How To Buy

How to Buy Meme Coin

Meme Coin is available for purchase on PancakeSwap, one of the leading decentralized exchanges that allows for swapping of crypto tokens without the need for an intermediary. To buy Meme Coin, simply visit PancakeSwap's swap interface. Ensure you have a compatible wallet such as MetaMask or Trust Wallet connected to the site. For trading, you can use the BNB/BESKAR (Calamari Ingots) pair to swap BNB or any other available token for Meme Coin. Follow the on-screen instructions to execute your trade safely and securely.

https://pancakeswap.finance/swap?outputCurrency=0xd9938C445897e6BDD3cCdA842d55104e7C58d65c

The purchase will be available on the 16th of April at 11pm for North America, CST. which is on the 17th at 9am in Europe, Amsterdam time CET

Tokenomics

Tokenomics of Calamari Ingots (BESKAR)

Initial and Maximum Supply

- **Initial Supply**: 100 trillion (100T) Beskar tokens will be initially minted to establish a strong foundation for market entry and facilitate early distribution and adoption.
- **Maximum Supply**: The total supply of Beskar tokens is capped at 1,000 trillion (1,000T). This cap ensures scalability and utility enhancement within the ecosystem over time, safeguarding against inflation and maintaining token value.

Distribution Strategy

- **Founder's Allocation (30%)**: To support the project's initial phase, ongoing development, and strategic vision, 30% of the initial supply is reserved for the founders. This includes a responsible liquidation plan that allows for up to $150,000 to be liquidated only after achieving a liquidity milestone of $2 million. This plan emphasizes the founders' commitment to sustainable growth and long-term value creation, with an aspirational liquidity goal of $200 million.
- **Liquidity Pool (40%)**: Ensuring a stable and liquid market is vital. Therefore, 40% of the initial supply is allocated to liquidity pools on various decentralized exchanges. This strategic liquidity provision is essential for enabling smooth trading experiences and minimizing price volatility, which attracts and retains users.
- **Future Development (30%)**: Dedicated to ensuring the continuous growth and enhancement of the ecosystem, this portion of tokens is set aside for new projects and initiatives. It funds the development of key features like the Anonymity Vault and Loan System, improvements to the staking protocol, community-engaging airdrops, and robust marketing campaigns to broaden outreach and adoption.

Minting and Burning Mechanism

- **Minting**: New tokens can be minted exclusively through the staking of existing Beskar tokens. This method ties the creation of new tokens to participant activity, aligning token supply with actual ecosystem usage and demand, thus promoting economic stability.
- **Burning**: A burning mechanism is in place to allow for the reduction of the total supply when necessary. This feature is designed to counteract potential inflationary pressures and enhance the intrinsic value of Beskar tokens over time.

Features

Contracts and Unique Features of Calamari Ingots (BESKAR)

Anonymity Vault System

OutOnlyVault: At the heart of our commitment to privacy and security lies the OutOnlyVault, a sophisticated contract system that supports ERC20 transactions with enhanced privacy. This system uses a unique passkey for each transaction, mirroring the privacy of cash transactions while ensuring security and ease of use. Designed for those who prioritize discretion, this system facilitates private transactions without exposing details to the public ledger, maintaining the confidentiality of all parties involved.

Loan System with Auto-Liquidity Contribution

This cutting-edge feature allows users to take secure loans by automatically contributing the loan's collateral to the ecosystem's liquidity pools. Here's how it enhances the ecosystem:

- **Enhanced Liquidity:** By locking the collateral in liquidity pools, this system ensures there is always sufficient liquidity available, thereby stabilizing the market for Beskar tokens.
- **Sustainable Interest Handling:** All interest generated from these loans is directed back into the liquidity pools and is permanently locked, further supporting the ecosystem's financial health.

Staking and Token Minting

Our staking protocol is designed to encourage long-term holding and active participation within the Calmari Ingots ecosystem:

- **Rewarding Engagement:** Token holders can stake their Beskar to earn rewards, promoting a vibrant and active community.
- **Growth Through Participation:** This approach not only secures the network but also ensures that the tokenomics are driven by genuine user engagement and ecosystem utility.

Utilization of Funds

The strategic liquidation of funds from the founder's allocation is executed with precision, focusing on substantial ecosystem enhancements:

- **R&D and Innovation:** Funds are allocated towards research and development, powering continuous improvements and innovations within the platform.
- **Community and Market Expansion:** Significant investments are made in community engagement and marketing strategies to broaden the user base and enhance market presence.
- **Infrastructure Development:** Resources are also channeled into expanding the Anonymity Vault and enhancing the loan system, ensuring the platform remains competitive and relevant.

Commitment to Sustainability and Transparency

Calamari Ingots is dedicated to maintaining a transparent, sustainable growth trajectory:

- **Transparent Liquidation Strategies:** We provide clear, detailed plans for fund usage, ensuring stakeholders are informed of how resources are being allocated.
- **Community-Driven Development:** Our operations are structured around community needs and feedback, fostering a responsive and inclusive ecosystem.

Liquidity

Introduction to Liquidity

Liquidity refers to the ease with which an asset can be converted into cash without affecting its market price. In the context of decentralized finance (DeFi), liquidity is provided by users who contribute assets to liquidity pools. These pools facilitate decentralized trading, lending, and other functions without the need for traditional financial intermediaries.

Liquidity providers (LPs) add their tokens to liquidity pools and in return, they receive liquidity provider tokens or LP tokens. These tokens represent their share of the pool and can be used to reclaim their stake, plus a portion of the trading fees.

How to Add Liquidity on PancakeSwap

1. **Connect Your Wallet**: Start by connecting your digital wallet (e.g., MetaMask, Trust Wallet) to PancakeSwap.
2. **Select a Liquidity Pool**: Navigate to the "Liquidity" section of the PancakeSwap interface. You can choose an existing pool or create a new pair if you have a unique token pairing.
3. **Add Liquidity**: Enter the amount of the tokens you wish to add to the pool. PancakeSwap will automatically calculate the equivalent amount of the paired token based on current market rates.
4. **Confirm the Transaction**: Approve the transaction from your wallet. Ensure you have enough of each token and cover transaction fees (gas fees).
5. **Receive LP Tokens**: Once the transaction is confirmed, you will receive LP tokens that represent your share of the pool.

Locking Your Liquidity

To enhance security and earn rewards, you can lock your LP tokens. Locking your tokens typically means committing them for a set period during which you cannot withdraw them. Here's how to lock LP tokens:

1. **Access the LP Staking Rewards Contract**: Navigate to the staking section on the website where the LP Rewards contract is integrated.
2. **Choose Lock Duration**: Select the duration for which you want to lock your LP tokens. Options typically include 1, 3, or 5 years.
3. **Lock Your LP Tokens**: Enter the amount of LP tokens you want to lock and confirm the transaction. Make sure to review the terms of the lock, especially the lock period and the penalties, if any, for early withdrawal.

Participating in LP Staking Rewards

Participating in LP staking rewards can significantly increase your returns, often offering higher interest rates than traditional banking products.

1. **Stake Your LP Tokens**: After locking your tokens, you can stake them in the LP staking rewards contract. Navigate to the staking interface and select the amount of LP tokens you wish to stake.
2. **Earn Rewards**: Rewards are typically calculated based on the amount staked and the duration of the stake. For example, a 20% annual return rate is applied to your staked LP tokens.
3. **Claim Your Rewards**: Depending on the contract, rewards can be claimed either periodically or at the end of the staking term. Follow the instructions on the platform to claim your rewards.

LP rewards Contract

How to use the lp rewards contract

To engage with the staking contract I provided, users will typically interact with the contract through a user interface that connects to the blockchain using a web3 provider, such as MetaMask. However, if they are engaging directly through a development environment like Remix, they would not deploy the contract (since it's already deployed), but they will interact with it using its deployed address.

Steps to Interact with the Deployed Staking Contract using Remix:

1. Setup Remix:

- Open Remix IDE.
- Go to the "File Explorers" and create a new file, e.g., **InteractWithLPStaking.sol**.
- Paste the interface of the staking contract into this new file. This interface should match the functions that you plan to interact with.

```
// Interface for our LP Staking Contract
interface ILPStaking {
    function stake(uint256 _amount, uint256 _lockDuration) external;
    function unstake() external;
    function calculateReward(address _user) external view returns (uint256);
}

contract InteractWithLPStaking {
    ILPStaking public lpStakingContract;

    constructor(address _stakingContractAddress) {
        lpStakingContract = ILPStaking(_stakingContractAddress);
    }

    function stakeTokens(uint256 amount, uint256 lockDuration) public {
        lpStakingContract.stake(amount, lockDuration);
    }

    function unstakeTokens() public {
        lpStakingContract.unstake();
    }

    function viewMyReward(address userAddress) public view returns (uint256) {
        return lpStakingContract.calculateReward(userAddress);
```

}

}**2. Compile the Interaction Contract:**

- Select the "Solidity Compiler" from the sidebar and compile the **InteractWithLPStaking.sol** file.

3. Deploy the Interaction Contract:

- Go to the "Deploy & Run Transactions" plugin.
- Select "Injected Web3" to connect to your MetaMask.
- Ensure you are connected to the same network where your LP Staking Contract is deployed.
- Deploy the **InteractWithLPStaking** contract by entering the address of the deployed LP Staking Contract as a constructor parameter.

4. Interacting with the Contract:

- Once deployed, you can interact with the **InteractWithLPStaking** contract to stake, unstake, or view rewards.
- Use the functions provided in the contract deployment interface in Remix to call these methods.

Points to Note:

- Ensure that you have the correct address of the deployed LP Staking Contract when you deploy the **InteractWithLPStaking** contract.
- Ensure that you have sufficient tokens and have approved the staking contract to use your tokens if required by the staking mechanism.
- You can also integrate these functionalities into a web-based DApp using web3.js or ethers.js libraries, which would interact with the contract's methods in a more user-friendly manner through a graphical user interface.

This setup provides a basic framework for users to interact with the staking contract programmatically using Solidity in Remix, suitable for testing or direct blockchain interaction without a frontend application.

Or

First, ensure that your project includes the Web3.js library. This can usually be added via a package manager like npm:

bashCopy code

npm install web3

Then, you can create a simple script that allows users to interact with the contract. Here’s an example of how a user might connect to the contract and call its functions:

javascriptCopy code

```
const Web3 = require('web3');

// Initialize web3 instance, assuming MetaMask is used for accessing Binance Smart Chain
const web3 = new Web3(window.ethereum);

// Prompt user to connect their MetaMask wallet
async function connectWallet() {
    try {
        await window.ethereum.enable(); // Deprecated, switch to ethereum.request({ method: 'eth_requestAccounts' }) if issues arise
        console.log('Wallet connected!');
    } catch (error) {
        console.error('User denied account access');
    }
}

// Contract ABI & Address
const contractABI = [/* ABI JSON here */];
const contractAddress = '0x...'; // Replace with your contract's address

// Creating contract instance
const lpStakingContract = new web3.eth.Contract(contractABI, contractAddress);

// Function to stake tokens
async function stakeTokens(amount, lockDuration) {
    const accounts = await web3.eth.getAccounts();
    const account = accounts[0]; // We use the first account provided by MetaMask
```

```
    const amountToSend = web3.utils.toWei(amount.toString(), 'ether'); // Convert amount to Wei (if using Ether denominations)

    try {
        await lpStakingContract.methods.stake(amountToSend, lockDuration).send({ from: account });
        console.log(`Staked ${amount} tokens for ${lockDuration} years`);
    } catch (error) {
        console.error('Error staking tokens:', error);
    }
}

// Function to unstake tokens
async function unstakeTokens() {
    const accounts = await web3.eth.getAccounts();
    const account = accounts[0];

    try {
        await lpStakingContract.methods.unstake().send({ from: account });
        console.log('Tokens unstaked');
    } catch (error) {
        console.error('Error unstaking tokens:', error);
    }
}

// Function to claim rewards
async function claimRewards() {
    const accounts = await web3.eth.getAccounts();
    const account = accounts[0];
```

```
    try {
        await lpStakingContract.methods.claimReward().send({ from: account });
        console.log('Rewards claimed');
    } catch (error) {
        console.error('Error claiming rewards:', error);
    }
```

}**Usage**

This script provides functions to connect to a user's wallet, stake tokens, unstake them, and claim rewards. Users would need to:

1. Connect their wallet using the **connectWallet** function.
2. Call **stakeTokens** with the amount and the desired lock duration.
3. Use **unstakeTokens** to remove their stake after the lock period ends.
4. Call **claimRewards** to retrieve any earned rewards.

Integration

To integrate this into a DApp, you would include these scripts in your application's frontend. You'd also need a user interface that calls these functions, which could be built using HTML/CSS/JS or a framework like React.

This setup assumes that the contract is already deployed and functioning as expected, and that users have MetaMask or similar installed and set up for Binance Smart Chain.

Staking calamari

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

interface IBEP20 {
    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function transfer(address recipient, uint256 amount) external returns (bool);
    function allowance(address owner, address spender) external view returns (uint256);
```

```
    function approve(address spender, uint256 amount) external returns (bool);
    function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}

contract LPStaking {
    IBEP20 public stakingToken = IBEP20(0xd9938C445897e6BDD3cCdA842d55104e7C58d65c); // Calamari Ingots "Beskar" token address

    struct Stake {
        uint256 amount;
        uint256 startTime;
    }

    mapping(address => Stake) public stakes;
    mapping(address => uint256) public rewards;

    address public owner;
    uint256 public annualPercentageRate = 20; // Example APR set at 20%

    constructor() {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Not owner");
        _;
    }
```

```
    function stake(uint256 _amount) public {
        require(stakingToken.transferFrom(msg.sender, address(this), _amount), "Transfer failed");
        stakes[msg.sender].amount += _amount;
        stakes[msg.sender].startTime = block.timestamp; // Update the startTime on new stake
    }

    function unstake(uint256 _amount) public {
        require(stakes[msg.sender].amount >= _amount, "Insufficient staked amount");
        require(block.timestamp > stakes[msg.sender].startTime + 1 weeks, "Stake is locked"); //
Assuming a 1-week lockup period
        stakes[msg.sender].amount -= _amount;
        stakingToken.transfer(msg.sender, _amount);
    }

    function claimRewards() public {
        uint256 reward = calculateReward(msg.sender);
        require(reward > 0, "No rewards available");
        rewards[msg.sender] = 0;
        stakingToken.transfer(msg.sender, reward);
    }

    function calculateReward(address _staker) internal view returns (uint256) {
        if (stakes[_staker].startTime + 1 weeks > block.timestamp) {
            return 0; // No rewards if locked
        }
        uint256 stakedPeriod = block.timestamp - stakes[_staker].startTime;
        uint256 dailyRate = annualPercentageRate / 365; // Calculate daily rate from APR
```

```
        return (stakes[_staker].amount * stakedPeriod / 86400) * dailyRate / 100; // Adjust reward calculation
    }

    // Utility function to adjust rewards manually or change APR
    function setAPR(uint256 _newAPR) public onlyOwner {
        annualPercentageRate = _newAPR;
    }
}
```

How to stake Beskar

Step-by-Step Guide to Staking Tokens via Remix

Step 1: Setup Your Environment

- **Open Remix**: Go to Remix IDE and set up your environment.
- **Load the Contract**: If not already loaded, you can create a new file in the "File Explorers" tab, paste the contract code into this file, and save it.
- **Compile the Contract**: Navigate to the "Solidity Compiler" tab, select the correct compiler version (matching the version in your contract, which is ^0.8.4), and click "Compile".

Step 2: Deploy the Contract (If Not Already Deployed)

- If the contract isn't deployed, you can deploy it directly from Remix:
    - In the "Deploy & Run Transactions" tab, select "Injected Web3" from the "Environment" dropdown to connect to MetaMask.
    - Select the account with the tokens you want to stake.
    - Click "Deploy" to deploy your contract to the network (make sure you are connected to the correct network like Binance Smart Chain).

Step 3: Interact with the Deployed Contract

- **Load the Deployed Contract**:

    - If the contract is already deployed, you can interact with it by using the "At Address" feature in the "Deploy & Run Transactions" tab.
    - Paste the deployed contract address into the "At Address" field and click "At Address" to load the deployed contract.

Step 4: Approve and Stake Tokens

- **Approve the Contract**: Before staking, your tokens must be approved for the contract to use them.
    - Locate the token contract in Remix (you might need the token contract ABI and address).
    - Use the **approve** function to allow the staking contract to spend the tokens. You will need to specify the amount and the staking contract address.
    - Example: If you want to stake 100 tokens, call **approve** with the staking contract's address and **100** as the amount.
- **Stake the Tokens**:
    - Go back to the loaded staking contract in Remix.
    - Enter the amount you want to stake in the **stake** function input.
    - Click "transact" or "write" to execute the staking transaction.
    - Confirm the transaction in MetaMask.

Step 5: Verify the Stake

- You can verify your staked tokens by checking the state in Remix:
    - Use any public view functions available in the contract to check your staked amount and any other relevant details.

Important Notes:

- Ensure that you have sufficient BNB or appropriate blockchain currency for gas fees in your wallet.
- Double-check the network and addresses you are interacting with to avoid any mishaps.
- Consider testing your interactions on a testnet before moving to the mainnet to ensure everything works as expected.

This process will enable you to stake tokens using the LP Staking contract directly from Remix IDE, leveraging the Ethereum or Binance Smart Chain networks and your MetaMask wallet for transactions.