What are the differences among calldata, memory and storage in Solidity?

一張含有 文字, 螢幕擷取畫面, 字型, 筆跡 的圖片

AI 產生的內容可能不正確。

What are the differences between POW and POS in solidity?

一張含有 文字, 螢幕擷取畫面 的圖片

AI 產生的內容可能不正確。

calculate hash in blockchain

一張含有 文字, 字型, 螢幕擷取畫面, 文件 的圖片

AI 產生的內容可能不正確。

Merkel

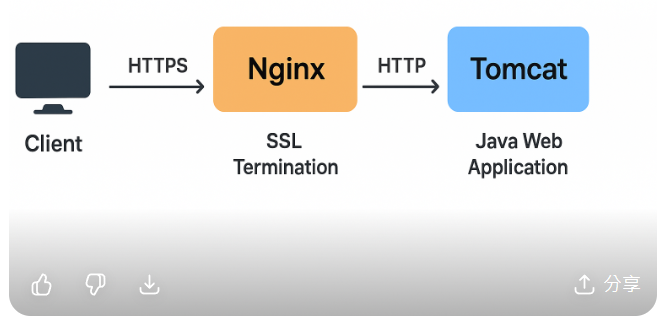
一張含有 文字, 圖表, 方案, 圖解 的圖片

AI 產生的內容可能不正確。

一張含有 文字, 字型, 白色, 收據 的圖片

AI 產生的內容可能不正確。

應用程式 ---> Docker (打包成容器) ---> Kubernetes (管理容器群)



**以太坊 Solidity 的存儲佈局 (storage layout)** 裡，EVM 的 storage 是以 **slot** 為單位的，每個 slot 固定 **32 bytes (256 bits)**

 如果它們的大小總和 **不超過 32 bytes**，編譯器會嘗試把它們 **打包 (pack)** 到同一個 slot 裡。

 例如：

contract C {

uint128 a; // 16 bytes

uint128 b; // 16 bytes

// a 和 b 會放在同一個 32-byte slot 裡

一張含有 文字, 圖表, 行, 字型 的圖片

AI 產生的內容可能不正確。

一張含有 文字, 圖表, 方案, 螢幕擷取畫面 的圖片

AI 產生的內容可能不正確。

Go channel VS (RabbitMQ or Kafka)

**Summary:**

* **Go Channels** are great for **concurrent** programming within a single application. They provide simple **message passing** between goroutines and do not persist messages.
* **Message Queues** like **Kafka** and **RabbitMQ** are designed for **distributed systems** and provide **reliable delivery**, **persistence**, **scalability**, and **fault tolerance** across multiple machines.
* Go channels and MQs have similar goals (communication between producers and consumers), but **Go channels are more lightweight** and fit well within **single-process** concurrency, while **MQs** are used for **distributed message passing**.

Do not communicate by sharing memory. Instead, share memory by communicating.

一張含有 文字, 螢幕擷取畫面, 字型, 文件 的圖片

AI 產生的內容可能不正確。

**Assembly Block:**  
The assembly block is a low-level way to interact directly with the Ethereum Virtual Machine (EVM) and is used here for efficiency and to avoid the overhead of Solidity's higher-level constructs.

Go 语言作为一个强类型语言，不同类型之间必须要显式的转换（而且必须有相同的基础类型）。但是，Go 语言中 interface 是一个例外：非接口类型到接口类型，或者是接口类型之间的转换都是隐式的。这是为了支持鸭子类型，当然会牺牲一定的安全性。

In Go, "鸭子类型" (duck typing) refers to the ability of a type to implicitly satisfy an interface if it has the required methods, without the need for explicit declarations. This allows Go to have a flexible and powerful type system while maintaining static typing.

func panic(interface{})

func recover() interface{}

**Conclusion:**

* **Transparent Proxy** is good for those who prefer explicit control over the proxy and implementation separation.
* **UUPS Proxy** is more gas-efficient and self-contained, with a focus on reducing complexity and improving upgrade efficiency, making it a better option for modern upgradable smart contracts.

If you're working with OpenZeppelin's contracts, both proxy types are available, but **UUPS** is gaining popularity due to its gas efficiency.

lsof -i :1234

**Summary: What is Protobuf Used For?**

* **Data Serialization**: Efficiently serializing structured data to a binary format.
* **APIs**: Defining and exchanging structured data in APIs, especially in high-performance environments.
* **gRPC Communication**: Protobuf is the core data format used in gRPC to define services, methods, and message types.
* **Cross-Language Communication**: Ensuring that different programming languages can communicate using the same schema and message format.
* **Efficient Communication**: Ideal for environments where performance and bandwidth are important, such as microservices, mobile applications, and IoT devices.

SHIB 风格代币的合约设计

通常一个 SHIB 风格代币合约会基于 **ERC-20** 标准，并增加一些附加功能（如税费、流动性池、交易限制等）。这意味着，它不仅具有基本的代币转账功能，还增加了更多的定制化功能。

ERC-20 是以太坊區塊鏈上的一個標準，規範了智能合約中代幣的基本行為。這些標準確保了不同的代幣能夠互操作並被支持，以便它們能夠在各種應用中使用，如去中心化交易所（DEX）、錢包等。

**ERC-20 的主要功能**

ERC-20 定義了七個必須實現的功能和事件，這些功能使代幣能夠在以太坊網絡上進行轉移和查詢。

1. **totalSupply()**
   * 返回總供應量，即目前發行的代幣數量。
2. **balanceOf(address account)**
   * 查詢某個地址的代幣餘額。
3. **transfer(address recipient, uint256 amount)**
   * 將代幣從發送者賬戶轉移到接收者賬戶。
4. **allowance(address owner, address spender)**
   * 查詢某個地址（owner）授予另一個地址（spender）可花費的代幣數量。
5. **approve(address spender, uint256 amount)**
   * 允許 spender 代替 owner 花費一定數量的代幣。
6. **transferFrom(address sender, address recipient, uint256 amount)**
   * 允許 spender 代替 sender 轉移代幣至 recipient。
7. **increaseAllowance(address spender, uint256 addedValue) 和 decreaseAllowance(address spender, uint256 subtractedValue)**
   * 這些是可選的擴展功能，允許增減授權。

**必須的事件**

ERC-20 還要求代幣必須發出兩個事件：

* **Transfer(address indexed from, address indexed to, uint256 value)**
  + 當代幣發生轉移時觸發。
* **Approval(address indexed owner, address indexed spender, uint256 value)**
  + 當用戶設定 approve 授權時觸發。

**ERC-20 的優點**

* **互操作性**：ERC-20 代幣遵循標準，因此可以在各種平台之間無縫運行。
* **簡單**：它為代幣操作提供了清晰、簡單的規範，便於開發者實現。
* **廣泛支持**：由於是廣泛使用的標準，幾乎所有主流的以太坊錢包和去中心化交易所都支持 ERC-20 代幣。

**ERC-20 的限制**

* **性能瓶頸**：由於每個轉帳都需要支付以太坊的 gas 費用，因此在大量交易時，可能會遇到高昂的交易成本或交易延遲。
* **功能單一**：ERC-20 僅處理基本的轉移、批准等操作，無法處理更複雜的用例，如可變性、治理等需求。

https://etherscan.io/

一張含有 文字, 字型, 螢幕擷取畫面, 代數 的圖片

AI 產生的內容可能不正確。