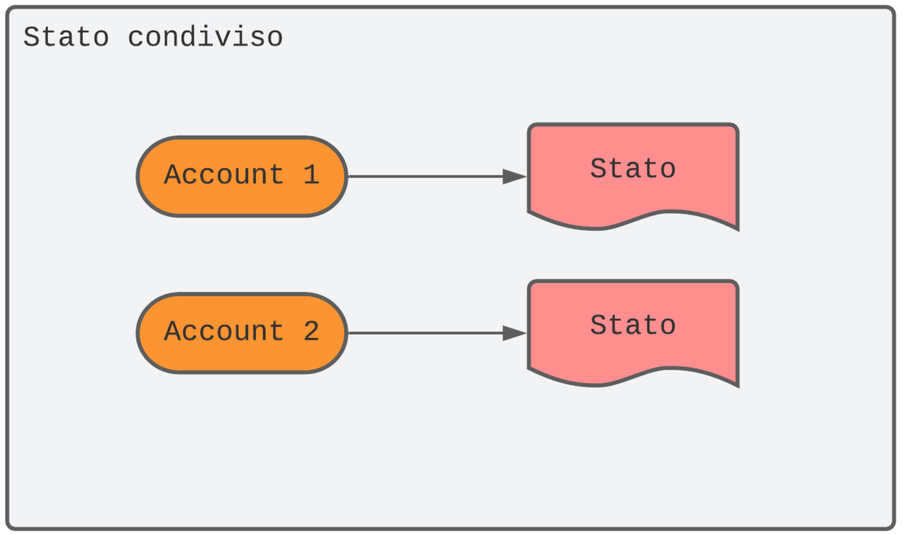
📖 Ethereum and EVM: deep dive

- Externally owned accounts (EOA) e Contract accounts
- Blocchi
- Transazioni
- Gas
- Esecuzione di uno smart contract
- Q&A

Cos’è un account?

Un account è rappresenta un mapping nello *stato condiviso* tra un indirizzo e lo *stato* di tale indirizzo.



Externally owned accounts (EOA) e Contract accounts

In Ethereum sono presenti due tipi di account:

- **externally owned** accounts (*EOA*)
- **contract** accounts (*smart contracts*)

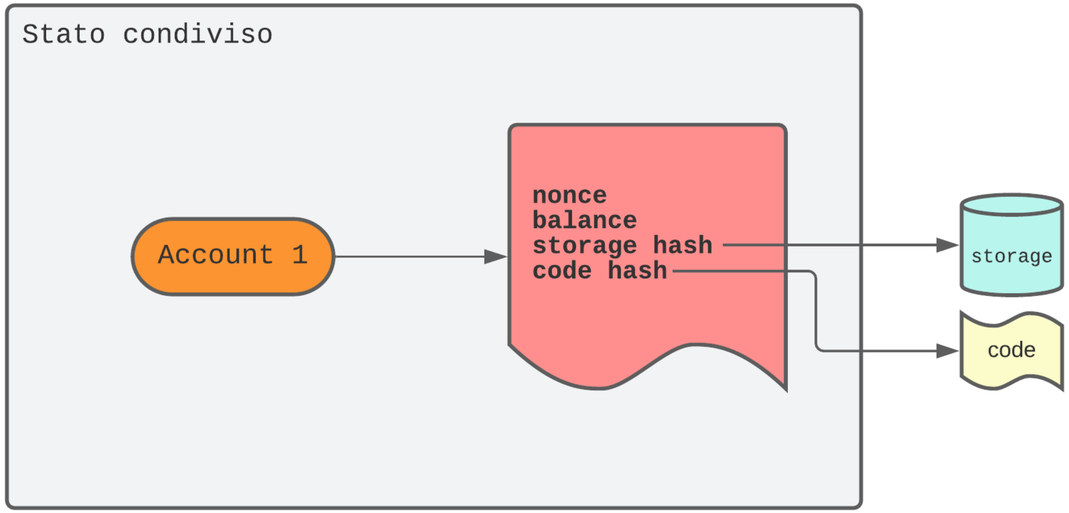
Entrambi i tipi di account possono:

- ricevere e inviare ETH
- creare altri smart contract
- interagire con gli smart contract esistenti

Lo *stato* di un account

Lo stato di un account contiene informazioni riguardo:

- ***nonce***: un counter per il numero di transazioni eseguite dall'account (nel caso di uno smart contract il numero di contratti deployati dal creatore)
- ***balance***: il saldo di Ether dell'account (espresso in wei)
- ***storage hash***: un hash che identifica lo *storage* dell'account (vuoto per gli EOA)
- ***code hash***: un hash che identifica il codice che può essere eseguito dall'account (hash di una stringa vuota per gli EOA)



Derivazione degli indirizzi

Un account è identificato da una stringa formata da 40 caratteri esadecimali (*address*).

Un address che identifica un EOA viene derivato prendendo in considerazione gli ultimi 20 byte dell'hash (*keccak256*) della chiave pubblica associata ad un'account.

L'address di uno smart contract viene invece derivato prendendo in considerazione gli ultimi 20 byte dell'hash (*keccak256*) della stringa formata concatenando l'address e la nonce dell'account che lo crea.

Differenze tra smart contract ed EOA

Uno smart contract si differenzia da un EOA in quanto:

- la sua creazione ha un costo
- non può inviare transazioni autonomamente
- può eseguire del codice e mantenere uno *stato* su cui basare transazioni condizionali

Blocchi

Un blocco è un aggregato di transazioni che vengono eseguite sequenzialmente e cambiano lo stato della EVM.

I blocchi vengono validati tramite un meccanismo chiamato Proof-of-Work.

Ogni blocco contiene:

- **timestamp**: il momento in cui viene minato il blocco.
- **blockNumber**: un numero progressivo che identifica l'altezza del blocco
- **baseFeePerGas**: il *gasPrice* minimo richiesto per una transazione
- **difficulty**: un parametro dell'algoritmo Proof-of-Work
- **mixHash**: un identificativo unico per il blocco
- **parentHash**: l'hash del blocco precedente
- **transactions**: una lista di transazioni incluse nel blocco
- **stateRoot**: l'hash dell'intero *stato condiviso*
- **nonce**: un hash che combinato con il *mixHash* prova il "lavoro" svolto dal miner

Transazioni

Una transazione è un’azione eseguita da un EOA (eg. Bob invia ad Alice 1 ETH).

Ogni transazione (che cambia lo stato della EVM) viene propagata all’interno della rete di Ethereum e verrà successivamente validata da un miner per essere inclusa in un **blocco**.

Ogni transazione richiede una **fee** espressa in termini di **gas**.

Transazioni

Una transazione contiene i seguenti campi:

- **recipient**: l’address destinatario della transazione
- **signature** (r, s, v): una firma che identifica il mittente
- **value**: il valore in ETH della transazione (espresso in wei)
- **data**: un campo opzionale che contiene dei dati (utile per eseguire smart contract)
- **gasLimit**: il numero massimo di unità di gas che possono essere consumate dalla transazione
- **maxPriorityFeePerGas**: il numero massimo di wei per unità di gas che viene incluso come una **tip** al miner
- **maxFeePerGas**: il costo massimo (espresso in wei) di una unità di gas

Tipi di transazione

In Ethereum esistono vari tipi di transazione:

- **regular transactions**: transazioni da un account all'altro
- **contract deployment**: transazioni che creano uno smart contract (identificate dal fatto di avere il campo *to* vuoto)

In particolare le **regular transactions** possono essere sfruttate per eseguire smart contract indicando nel campo *to* l'indirizzo dello smart contract e nel campo *data* un payload per l'esecuzione di quest'ultimo.

Messaggi

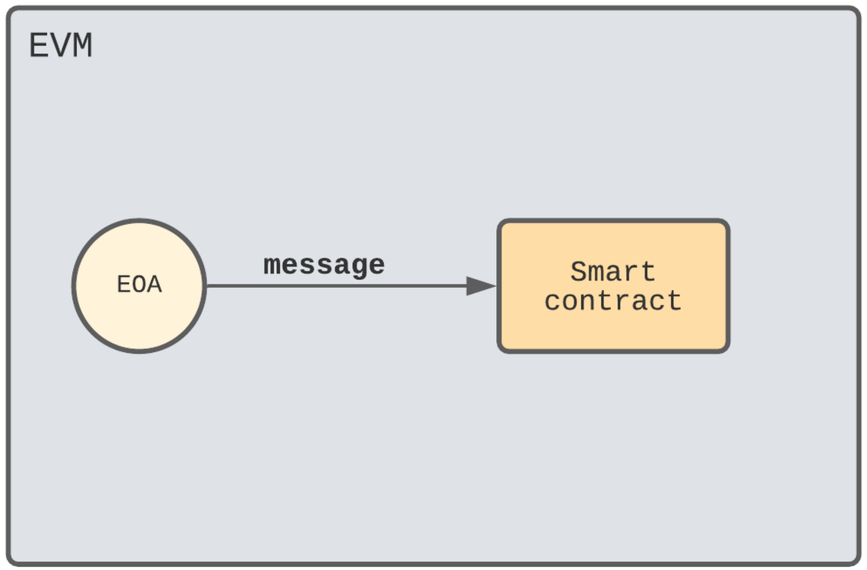
I **messaggi** vengono inviati da un account all’altro.

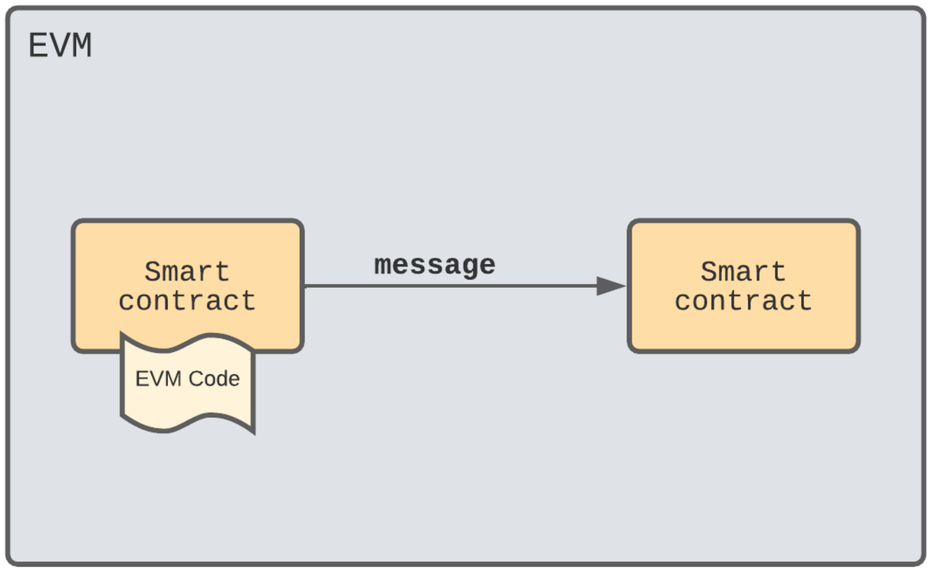
In solidity è possibile accedere al messaggio tramite la parola riservata *msg*.

I messaggi includono dei dati e un valore (specificato in wei).

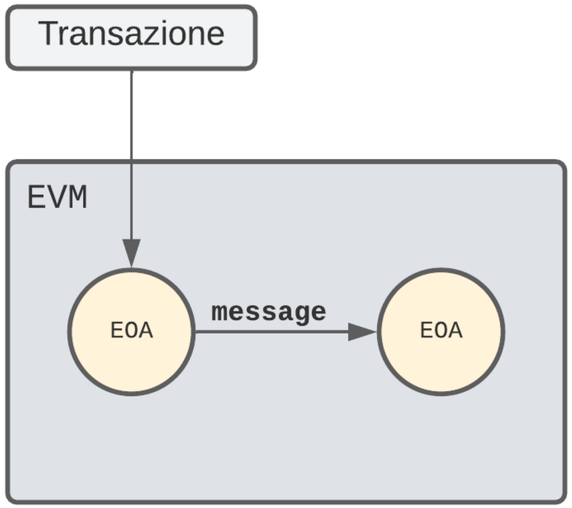
Messaggi

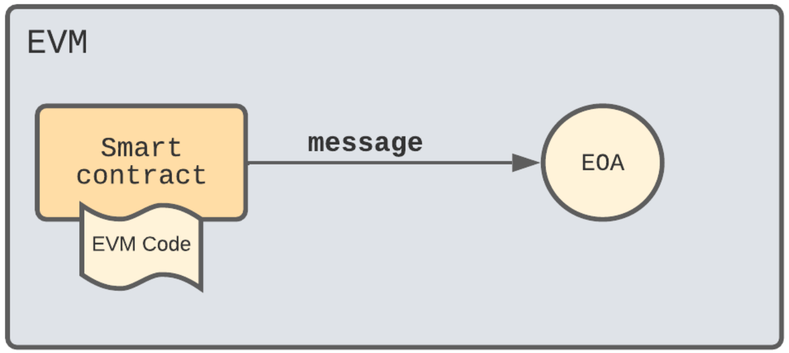
I messaggi possono essere inizializzati dalle transazioni inviate dagli EOA o dagli smart contract stessi.

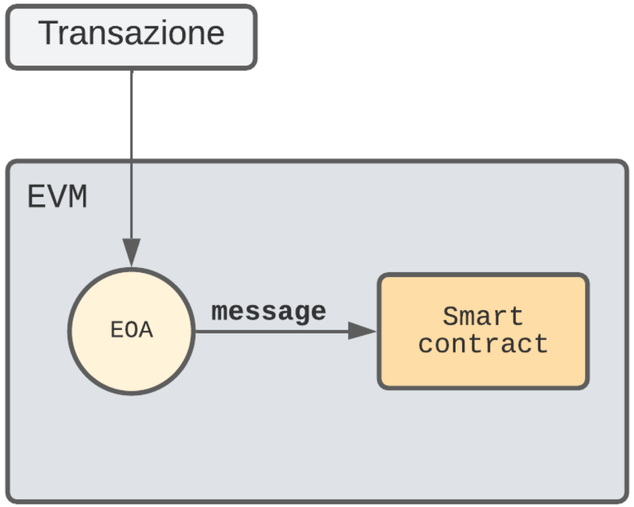


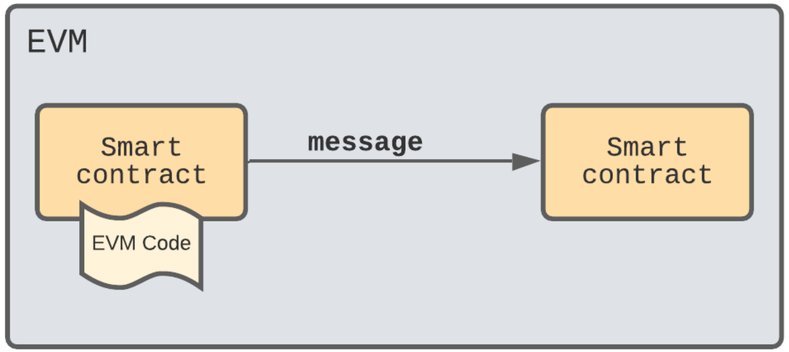


Messaggi


Transazione
EVM
EOA
message
EOA


EVM
Smart contract
message
EOA
EVM Code


Transazione
EVM
EOA
message
Smart contract


EVM
Smart contract
message
Smart contract
EVM Code

Gas

Per evitare che lo spam di transazioni Ethereum richiede una **fee** associata ad ogni transazione (che richiede uno sforzo da parte dei miner). Ogni operazione richiesta alla EVM richiede quindi un quantitativo di Ether.

Ogni operazione (OPCODE) viene prezzata e i prezzi sono hardcoded nei client che partecipano alla rete Ethereum.

Un semplice invio di ETH da un account all’altro ha un valore fisso di 21000 per il *gasLimit*.

Gas

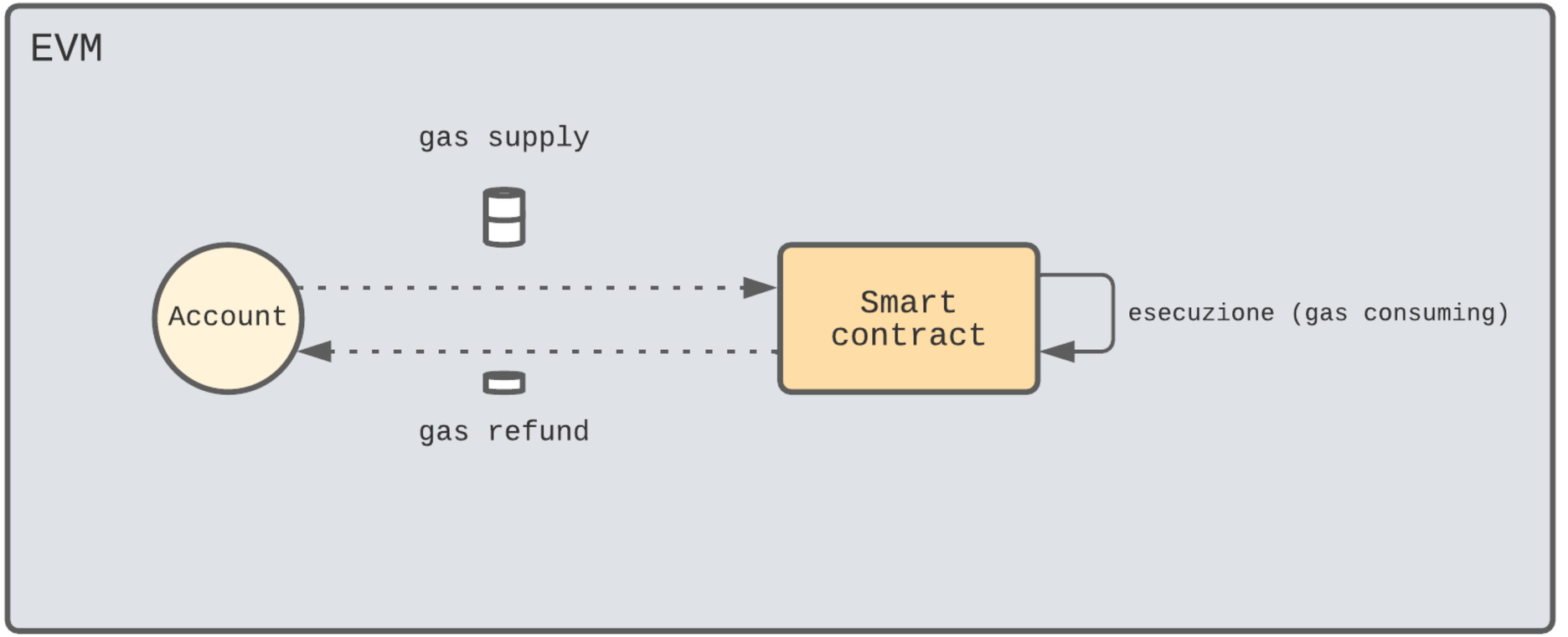
Il gas consumato da una transazione viene calcolato come la somma di tutte le operazioni eseguite moltiplicata per la somma tra *maxPriorityFeePerGas* e *maxFeePerGas*:

$$gas = (maxPriorityFeePerGas + maxFeePerGas) * gasLimit$$

Un account che invia una transazione ha quindi bisogno di abbastanza ETH nel suo *balance* per eseguire la transazione.

Il gas che non verrà consumato durante l'esecuzione della transazione verrà rimborsato all'utente.

Gas


EVM
gas supply
Account
Smart contract
esecuzione (gas consuming)
gas refund

Gas - esempio di invio di ETH

Supponiamo che Bob voglia inviare 1 ETH ad Alice. L'invio di ETH richiede 21000 unità di gas.

La *baseFeePerGas* al momento della transazione è di 10 gwei (0.00000001 Ether) e Bob vuole inserire una **tip** per il miner di 1 gwei (0.000000001 Ether).

La transazione costerà a Bob 1 Ether + (0.00000001 Ether + 0.000000001 Ether) * 21000 = 1 Ether + 0.000231 Ether = 1.000231 Ether.

Un totale di 1.000231 Ether verrà detratto a Bob.

Un totale di 1 Ether verrà accreditato a Alice.

Un totale di 0.000021 Ether vanno al miner.

Un totale di 0.00021 Ether verrà bruciato (post EIP-1559).

SayWhat

```
/// @notice A simple contract.
contract SayWhat {
    /// @notice what
    string private what;

    /// @dev Constructor, called when the contract is deployed.
    /// @param what_ The initial what.
    constructor(string memory what_) {
        what = what_;
    }

    /// @notice The main function. This contract can say
    function say() external view returns(string memory) {
        return what;
    }

    /// @notice Anyone can make this contract say anything.
    /// @param what_ The new what.
    function setWhat(string memory what_) external {
        what = what_;
    }
}
```

Göerli address: 0x06Ad3DC1c8e07243EA7E2D59195DF8f4544661fD