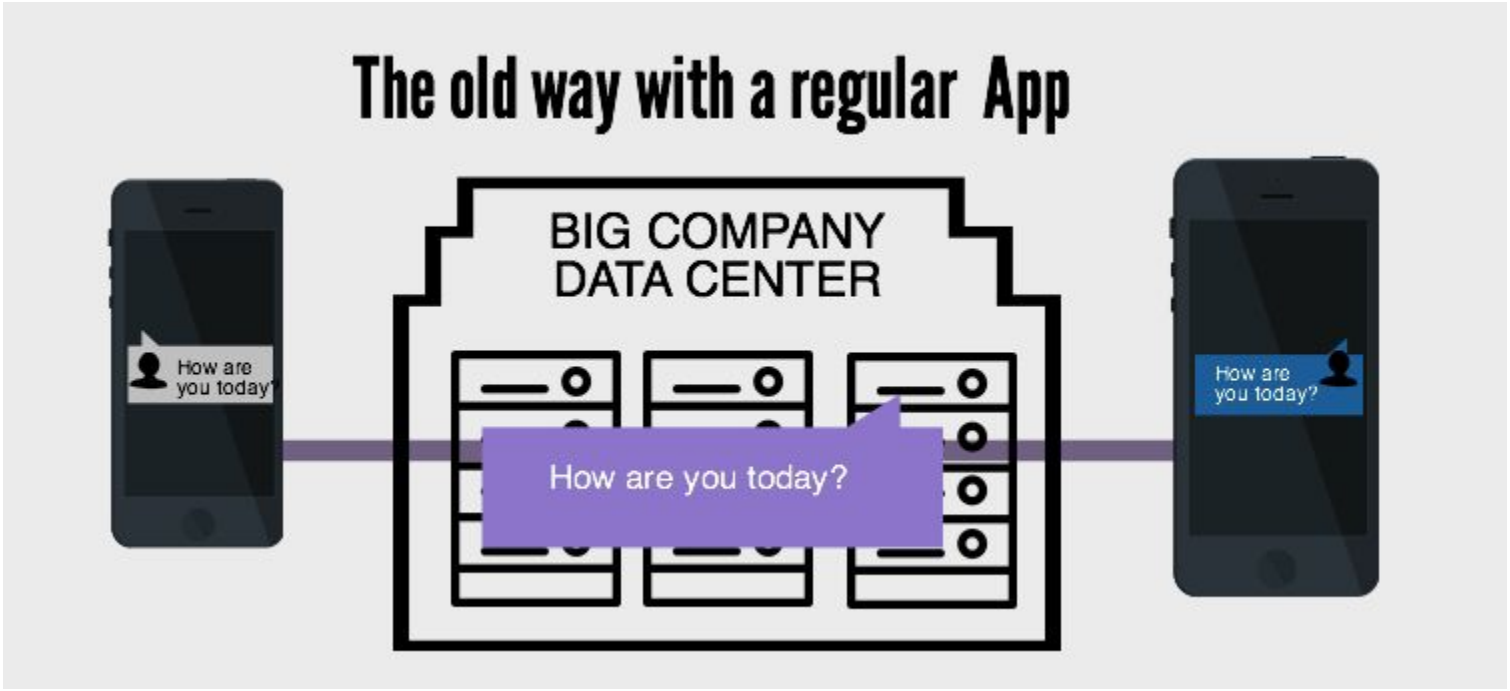
Ethereum - ĐApp

Alessandro Maccagnan

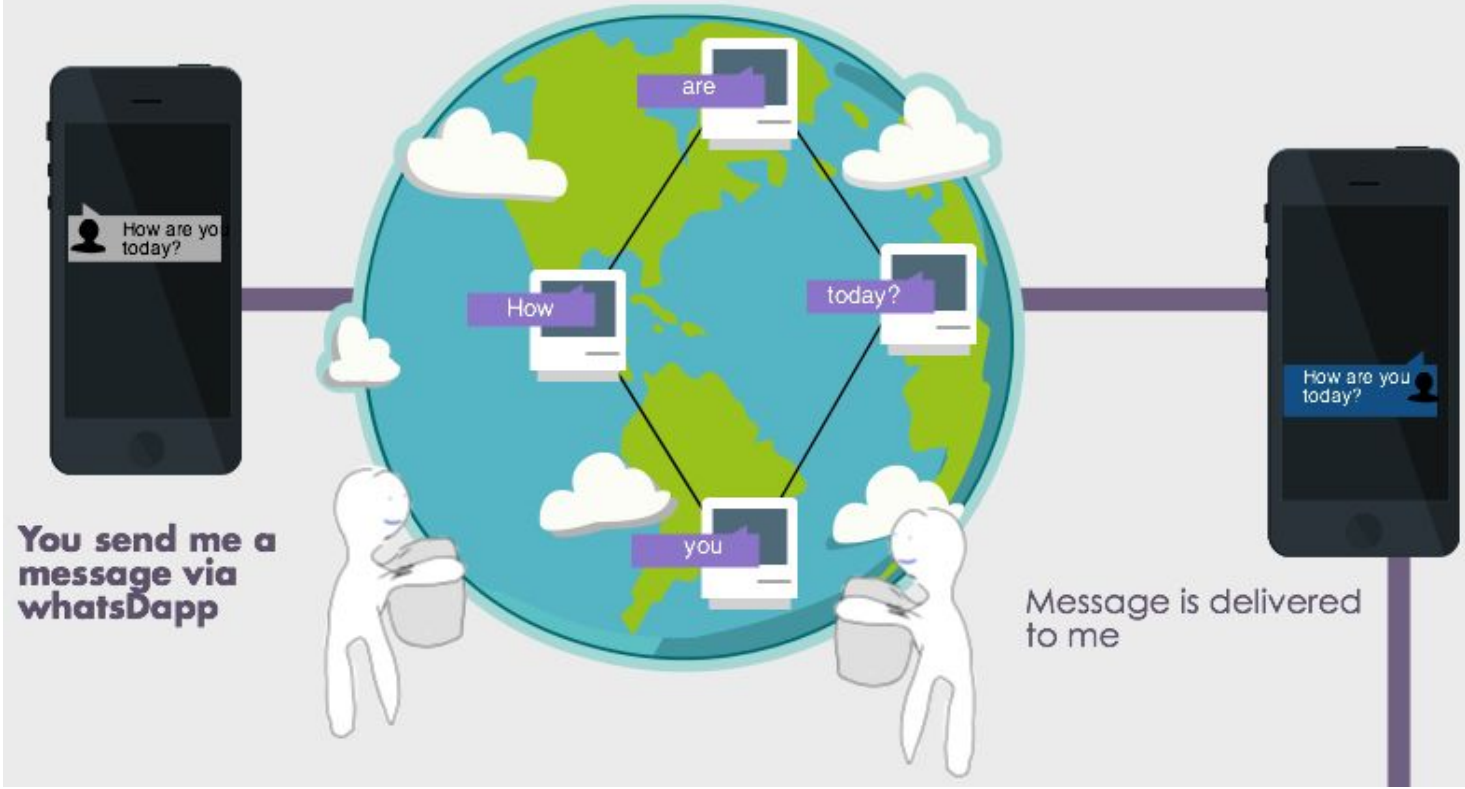
Red Babel

"Ethereum is a network of computers that make a new type of apps possible"

https://medium.com/@angelomilan/ethereum-explained-to-my-mom-infographic-673e32054c1c


The old way with a regular App
BIG COMPANY
DATA CENTER
How are you today?
How are you today?
How are you today?

The ethereum way with a dApp = decentralized App



Every participating computer receives a reward for their work

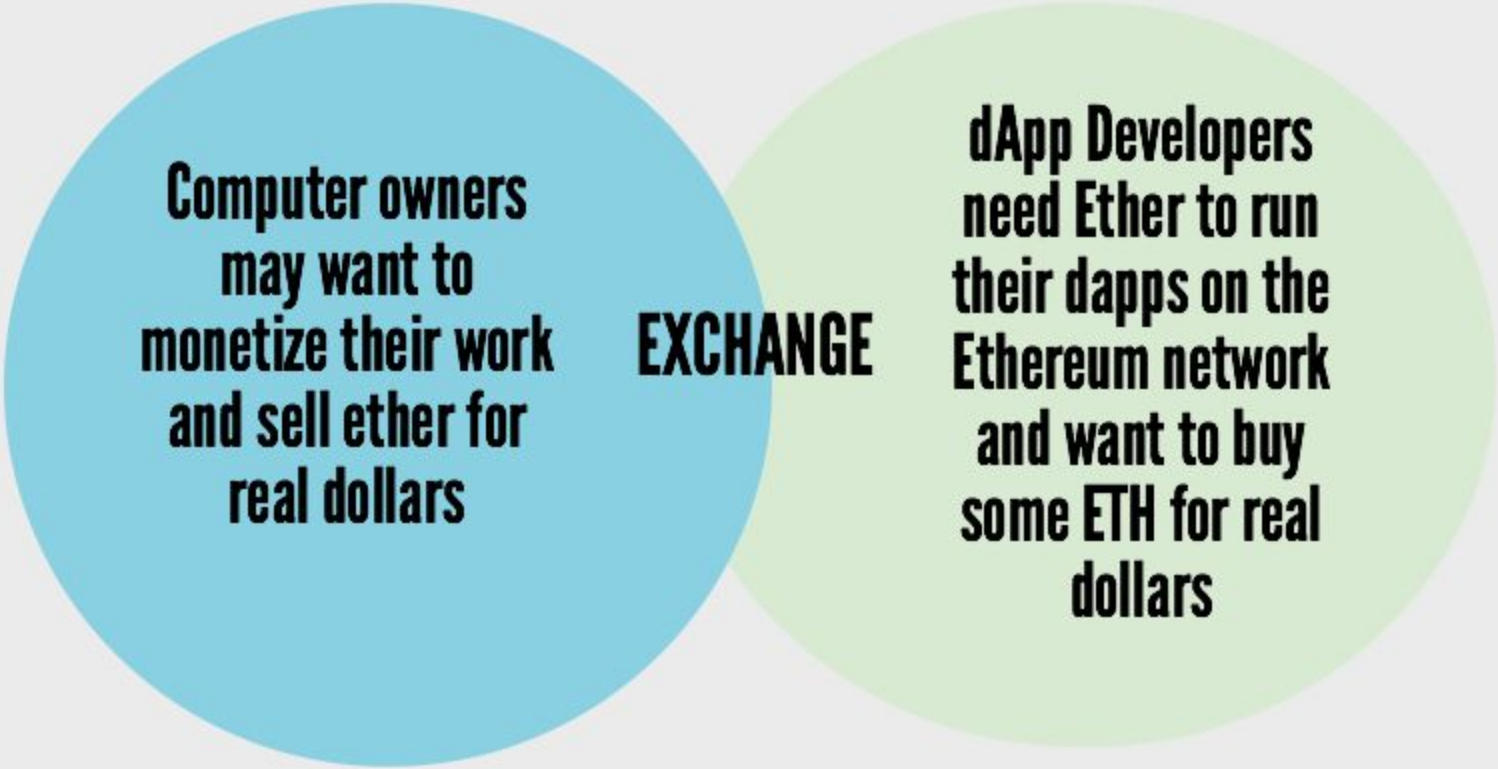
because they paid for hardware, electricity and shared their processing power

they don't get dollars

but a digital asset called ETHER

Ether is the fuel that makes dApps run. Like gas makes your car run.

How can I use this "Ether" ?



Distributed database

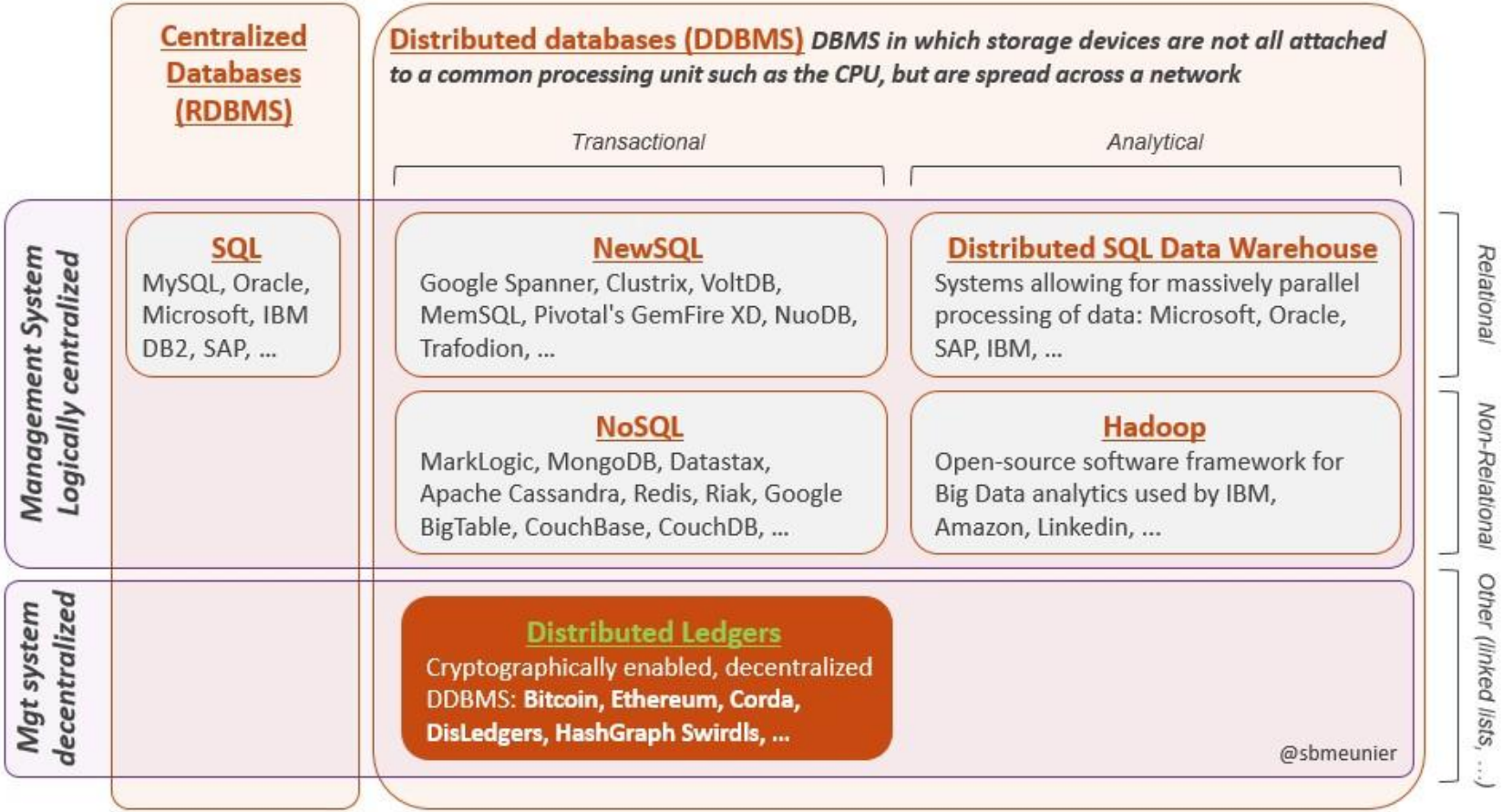
plus

Stored procedures

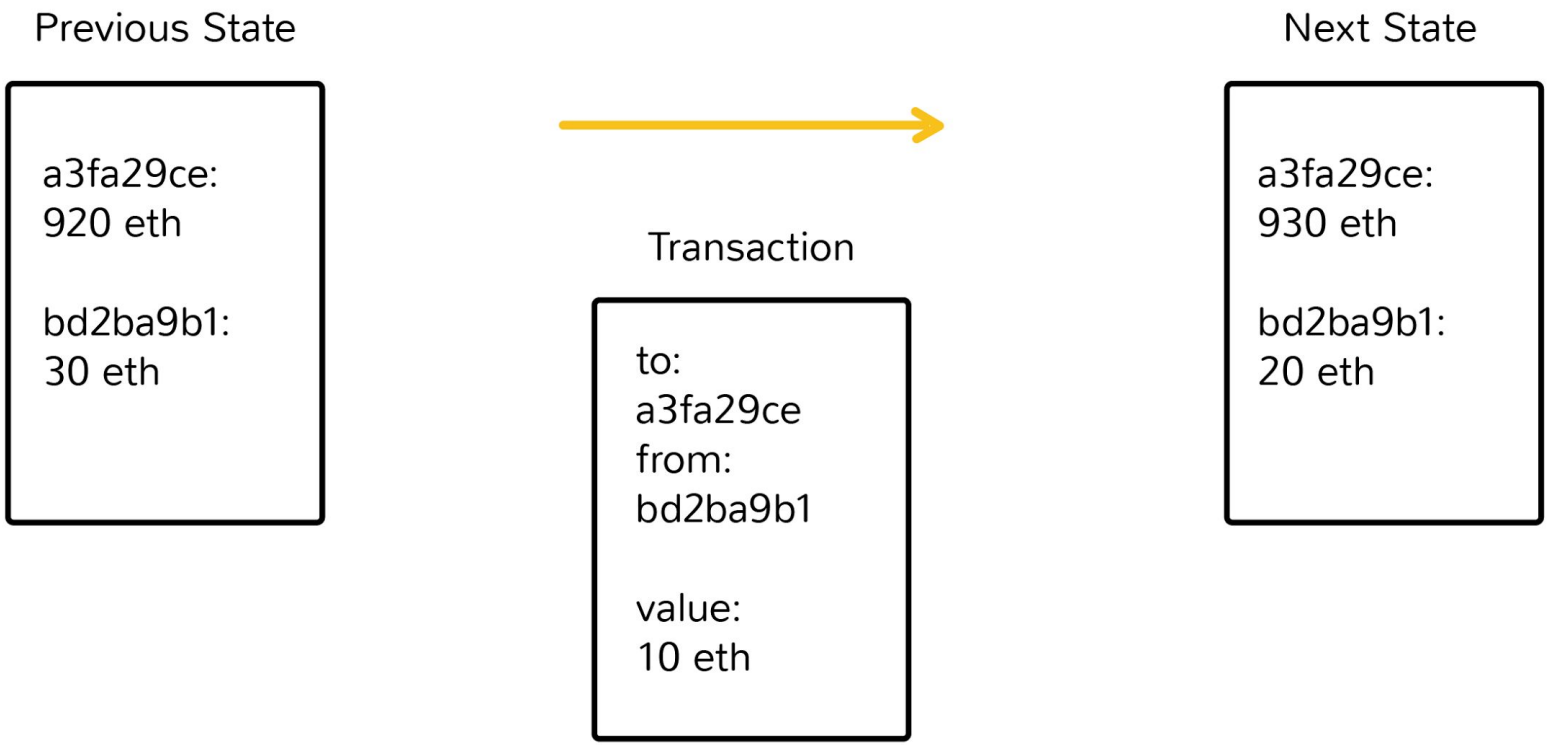
(almost) equal

Ethereum

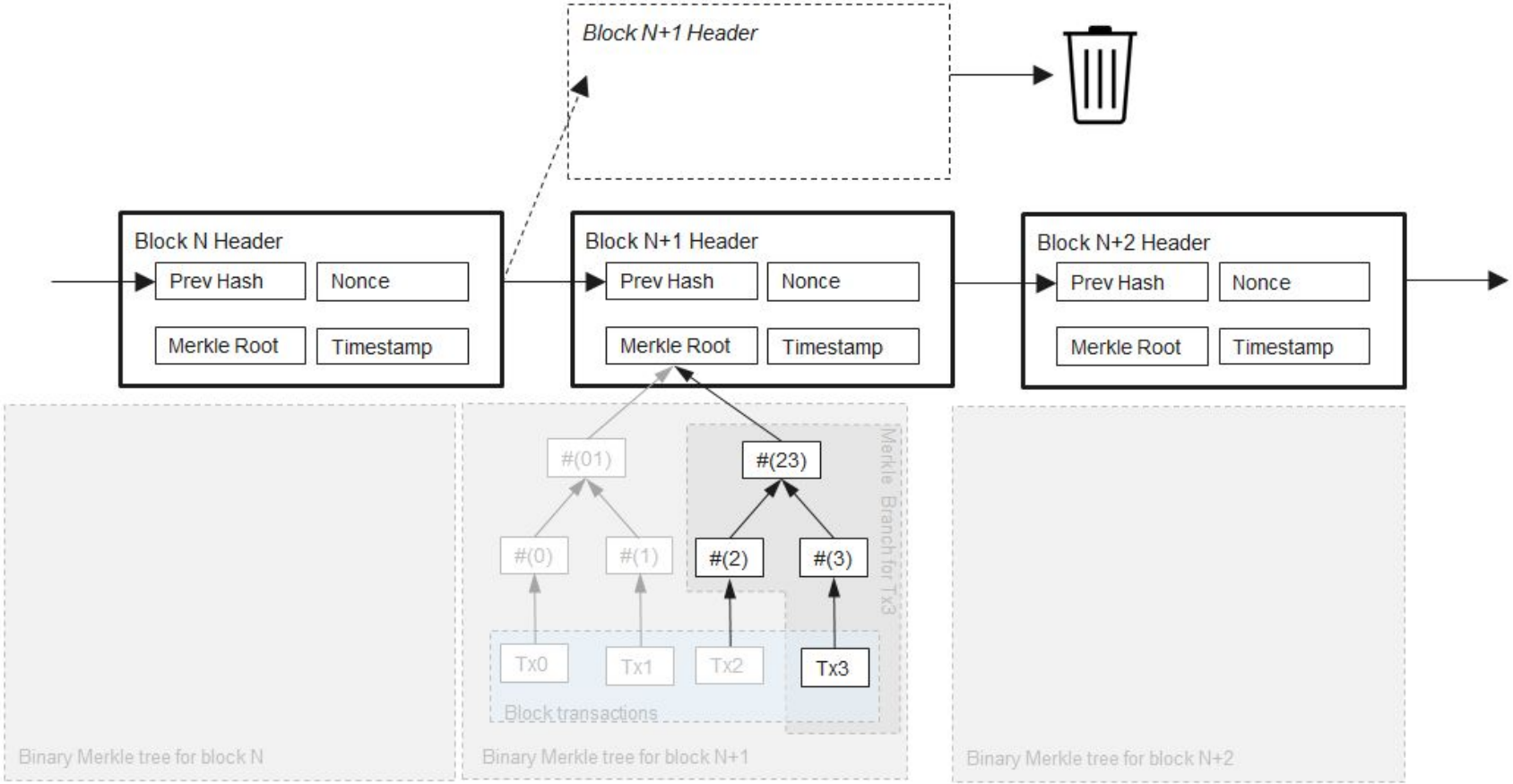
Distributed database



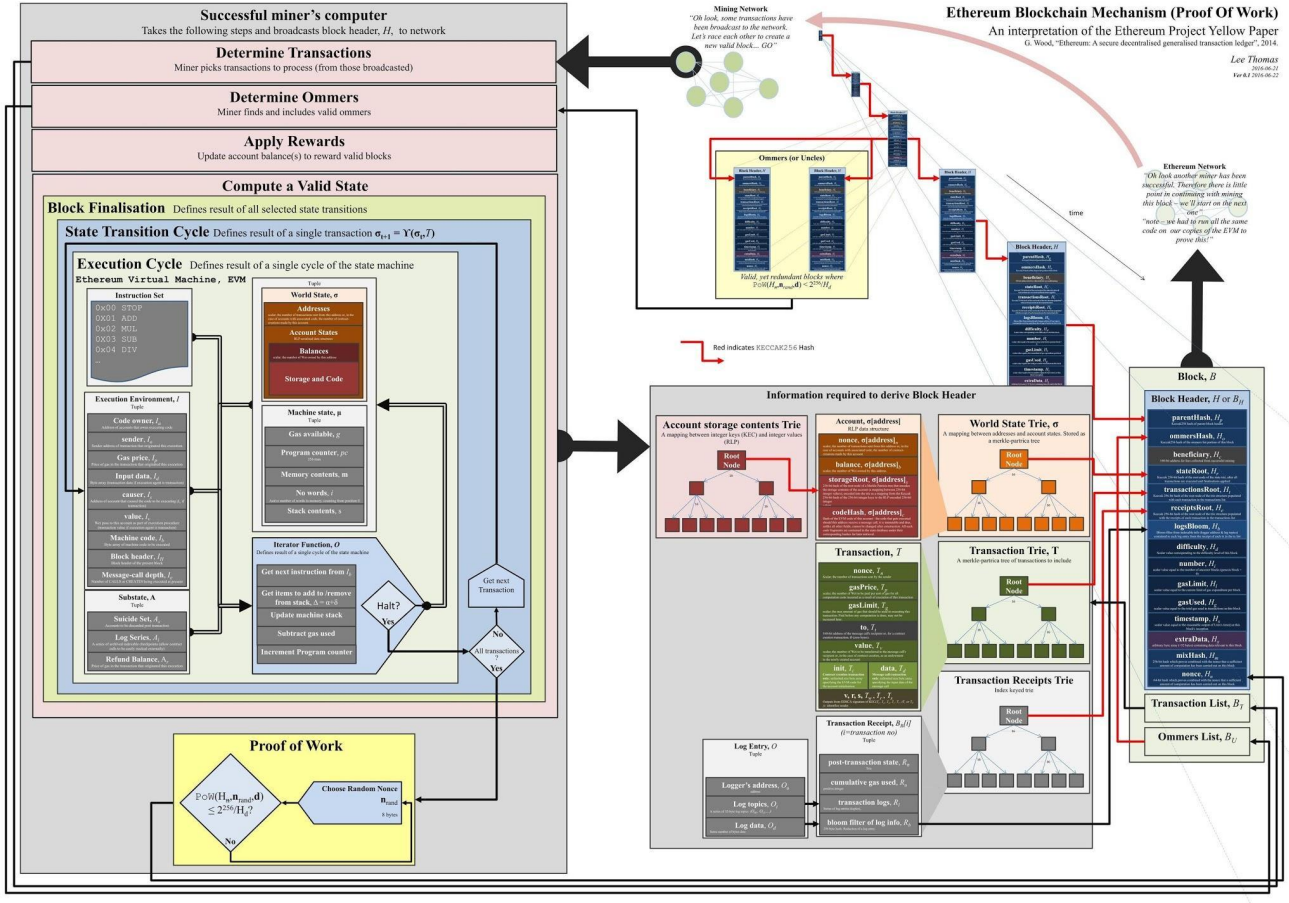
https://medium.com/@sbmeunier/blockchain-technology-a-very-special-kind-of-distributed-database-e63d00781118



https://www.coindesk.com/information/how-ethereum-works/



https://ethereum.stackexchange.com/questions/268/ethereum-block-architecture



https://ethereum.stackexchange.com/questions/268/ethereum-block-architecture

depending on whether the transaction is for contract-creation or message-call. G_{txcreate} is added if the transaction is contract-creating, but not if a result of EVM-code. G is fully defined in Appendix G.

The up-front cost v_0 is calculated as:

$$v_0 \equiv T_g T_p + T_v \tag{63}$$

The validity is determined as:

$$\begin{aligned} S(T) &\neq \varnothing \quad \wedge \\ \boldsymbol{\sigma}[S(T)] &\neq \varnothing \quad \wedge \\ T_n &= \boldsymbol{\sigma}[S(T)]_n \quad \wedge \\ g_0 &\leqslant T_g \quad \wedge \\ v_0 &\leqslant \boldsymbol{\sigma}[S(T)]_b \quad \wedge \\ T_g &\leqslant B_{Hl} - \ell(B_{\mathbf{R}})_u \end{aligned} \tag{64}$$

Note the final condition; the sum of the transaction's gas limit, T_g, and the gas utilised in this block prior, given by $\ell(B_{\mathbf{R}})_u$, must be no greater than the block's **gasLimit**, B_{Hl}.

The execution of a valid transaction begins with an irrevocable change made to the state: the nonce of the account of the sender, $S(T)$, is incremented by one and the balance is reduced by part of the up-front cost, $T_g T_p$. The gas available for the proceeding computation, g, is defined as $T_g - g_0$. The computation, whether contract creation or a message call, results in an eventual state (which may legally be equivalent to the current state), the change to which is deterministic and never invalid: there can be no invalid transactions from this point.

We define the checkpoint state $\boldsymbol{\sigma}_0$:

$$\boldsymbol{\sigma}_0 \equiv \boldsymbol{\sigma} \quad \text{except:} \tag{65}$$

$$\boldsymbol{\sigma}_0[S(T)]_b \equiv \boldsymbol{\sigma}[S(T)]_b - T_g T_p \tag{66}$$

$$\boldsymbol{\sigma}_0[S(T)]_n \equiv \boldsymbol{\sigma}[S(T)]_n + 1 \tag{67}$$

The total refundable amount is the legitimately remaining gas g', added to A_r, with the latter component being capped up to a maximum of half (rounded down) of the total amount used $T_g - g'$.

The Ether for the gas is given to the miner, whose address is specified as the beneficiary of the present block B. So we define the pre-final state $\boldsymbol{\sigma}^*$ in terms of the provisional state $\boldsymbol{\sigma}_P$:

$$\boldsymbol{\sigma}^* \equiv \boldsymbol{\sigma}_P \quad \text{except} \tag{71}$$

$$\boldsymbol{\sigma}^*[S(T)]_b \equiv \boldsymbol{\sigma}_P[S(T)]_b + g^* T_p \tag{72}$$

$$\boldsymbol{\sigma}^*[m]_b \equiv \boldsymbol{\sigma}_P[m]_b + (T_g - g^*) T_p \tag{73}$$

$$m \equiv B_{Hc} \tag{74}$$

The final state, $\boldsymbol{\sigma}'$, is reached after deleting all accounts that appear in the self-destruct set:

$$\boldsymbol{\sigma}' \equiv \boldsymbol{\sigma}^* \quad \text{except} \tag{75}$$

$$\forall i \in A_{\mathbf{s}} : \boldsymbol{\sigma}'[i] \equiv \varnothing \tag{76}$$

And finally, we specify Υ^g, the total gas used in this transaction and Υ^{l}, the logs created by this transaction:

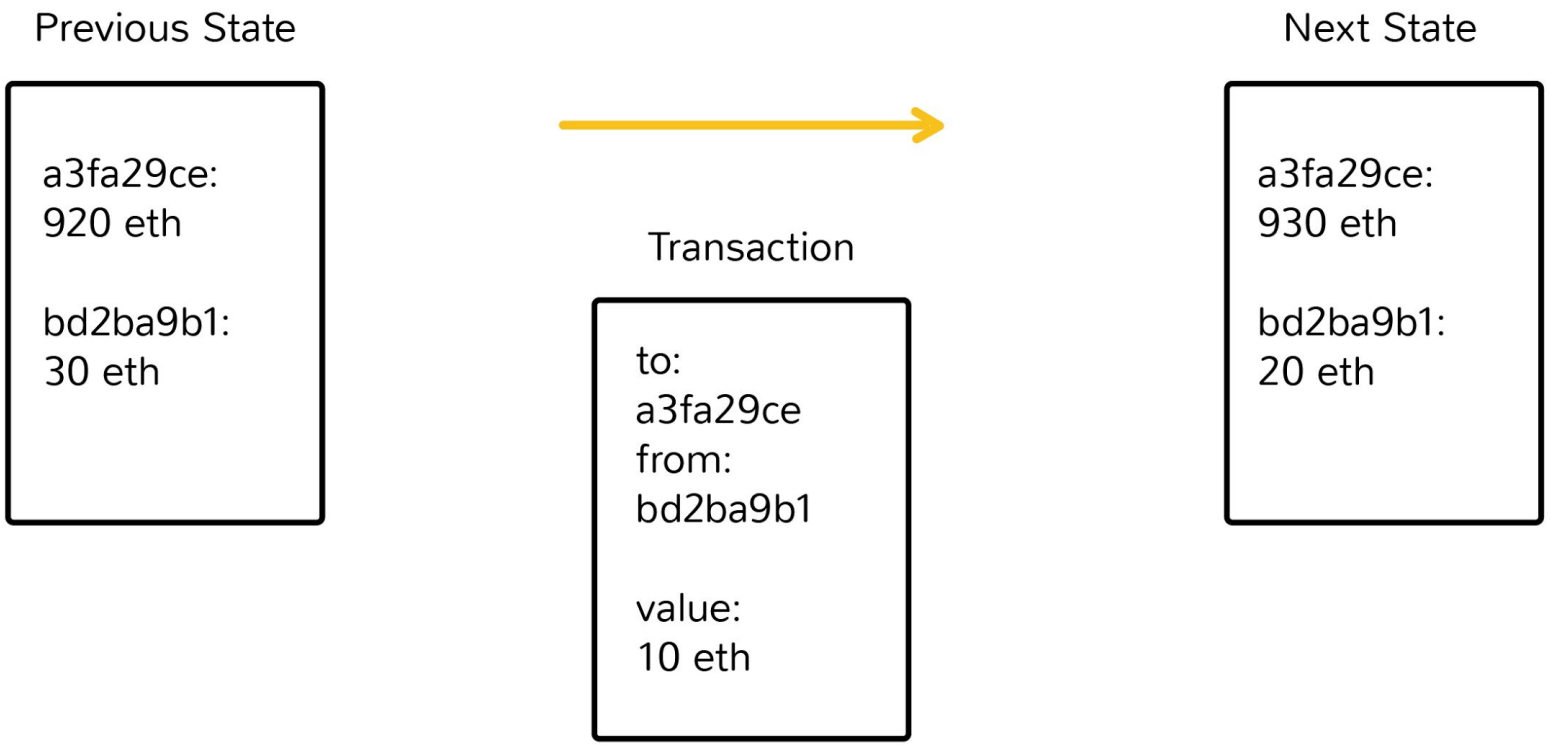
$$\Upsilon^g(\boldsymbol{\sigma}, T) \equiv T_g - g' \tag{77}$$

$$\Upsilon^{\mathrm{l}}(\boldsymbol{\sigma}, T) \equiv A_{\mathrm{l}} \tag{78}$$

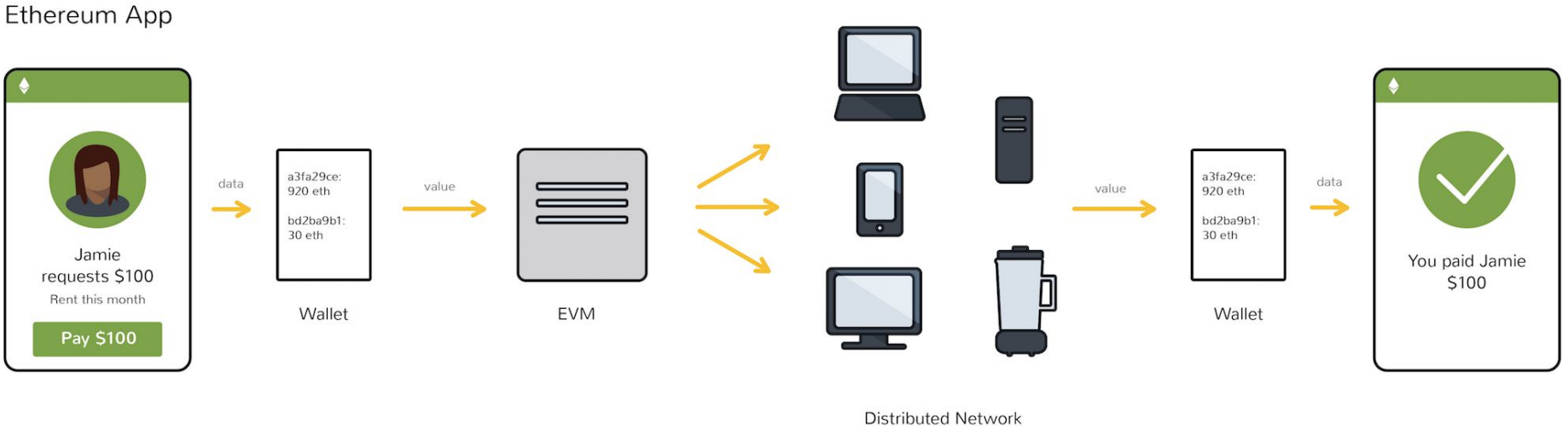
These are used to help define the transaction receipt, discussed later.

7. Contract Creation

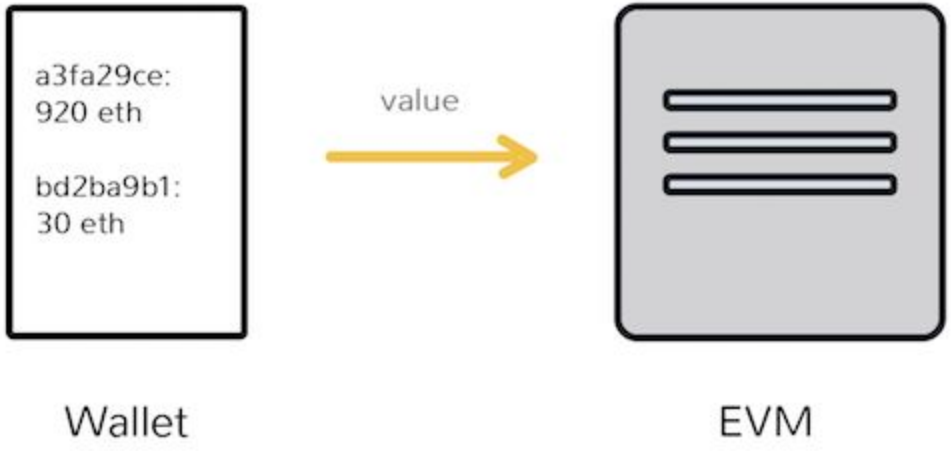
There are a number of intrinsic parameters used when creating an account: sender (s), original transactor (o), available gas (g), gas price (p), endowment (v) together with an arbitrary length byte array, $\mathbf{i}$, the initialisation EVM code and finally the present depth of the message-

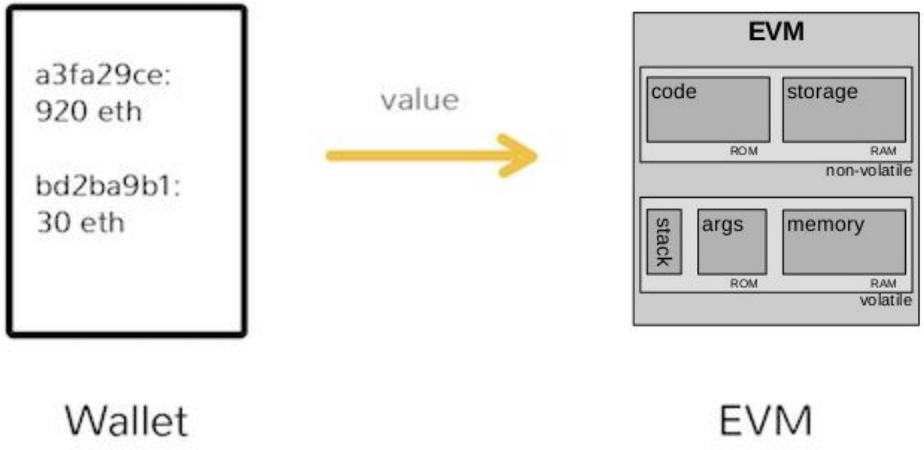


https://www.coindesk.com/information/how-ethereum-works/



https://www.coindesk.com/information/how-ethereum-works/


a3fa29ce:
920 eth
bd2ba9b1:
30 eth
value
Wallet
EVM



https://www.slideshare.net/aeronbuchanan/september-ethereum

Ethereum’s basic unit is the account. The Ethereum blockchain tracks the state of every account, and all state transitions on the Ethereum blockchain are transfers of value and information between accounts. There are two types of accounts:

- Externally Owned Accounts (EOAs), which are controlled by private keys
- Contract Accounts, which are controlled by their contract code and can only be “activated” by an EOA

http://ethdocs.org/en/latest/introduction/what-is-ethereum.html#how-does-ethereum-work

Ethereum’s basic unit is the account. The Ethereum blockchain tracks the state of every account, and all state **transitions** on the Ethereum blockchain are transfers of **value** and **information** between **accounts**. There are two types of accounts:

- Externally Owned Accounts (EOAs), which are controlled by private keys
- Contract Accounts, which are controlled by their contract code and can only be “activated” by an EOA

Ethereum’s basic unit is the account. The Ethereum blockchain tracks the state of every account, and all state transitions on the Ethereum blockchain are transfers of value and information between accounts. There are **two types** of accounts:

- Externally Owned **Accounts** (EOAs), which are controlled by private keys
- **Contract** Accounts, which are controlled by their contract code and can only be “activated” by an EOA

Ethereum’s basic unit is the account. The Ethereum blockchain tracks the state of every account, and all state **transitions** on the Ethereum blockchain are transfers of value and information between accounts. There are two types of accounts:

- Externally Owned **Accounts** (EOAs), which are controlled by private keys
- **Contract** Accounts, which are controlled by their contract code and can only be “activated” by an EOA

Distributed database

plus

Stored procedures

(almost) equal

Ethereum

Stored procedures

What is a smart contract?

A smart contract is code that runs on the EVM. Smart contracts can accept and store ether, data, or a combination of both. Then, using the logic programmed into the contract, it can distribute that ether to other accounts or even other smart contracts.

http://truffleframework.com/tutorials/ethereum-overview


Ethereum Solidity
Solidity is designed to compile to code –
for the Ethereum Virtual Machine.

Distributed database

plus

Stored procedures

(almost) equal

Ethereum

Blockchain

plus

Stored procedures

(almost) equal

Ethereum

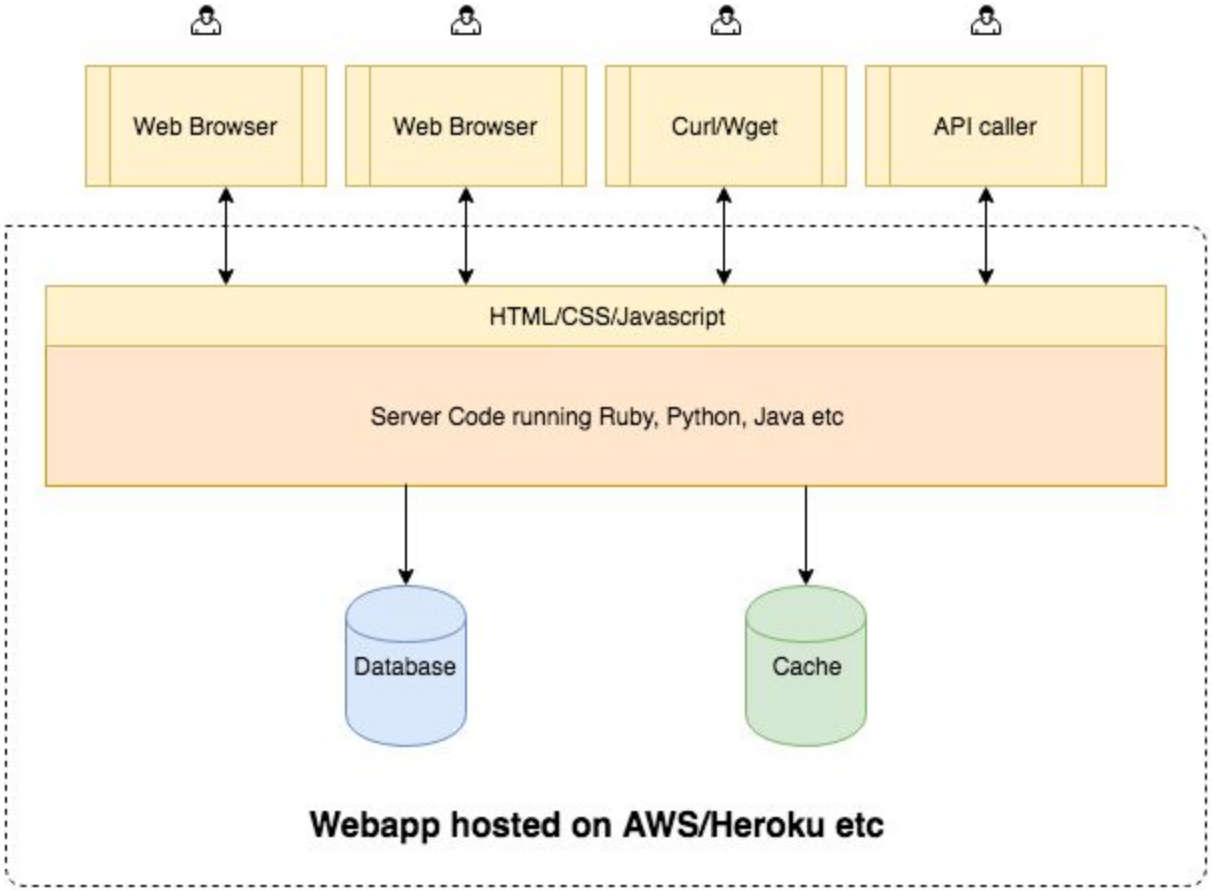
Blockchain

plus

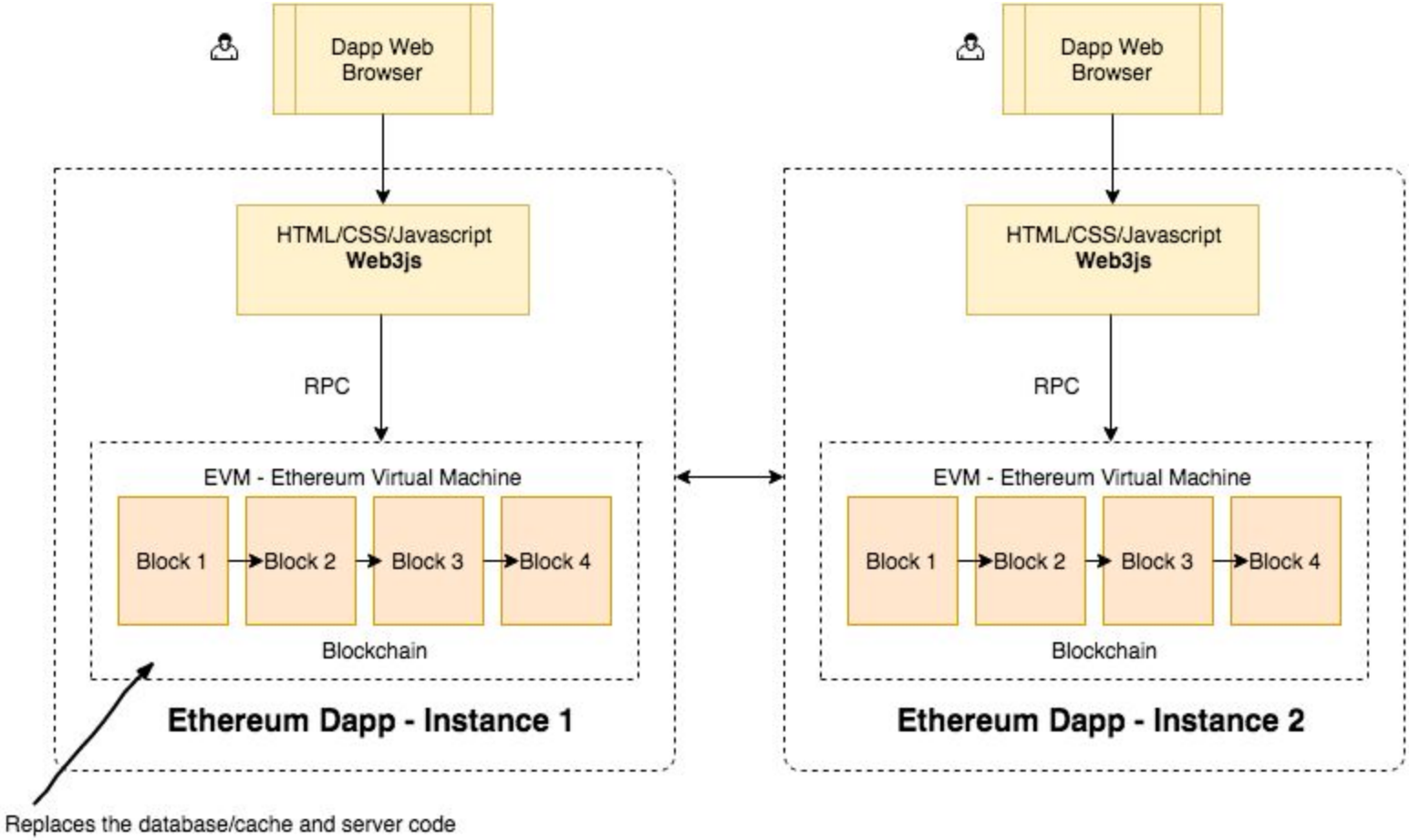
Smart contracts

(almost) equal

Ethereum

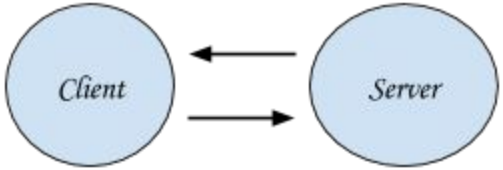


https://medium.com/@mvmurthy/ethereum-for-web-developers-890be23d1d0c

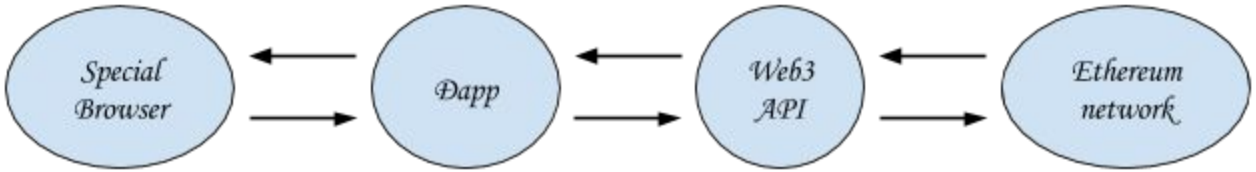


https://medium.com/@mvmurthy/ethereum-for-web-developers-890be23d1d0c

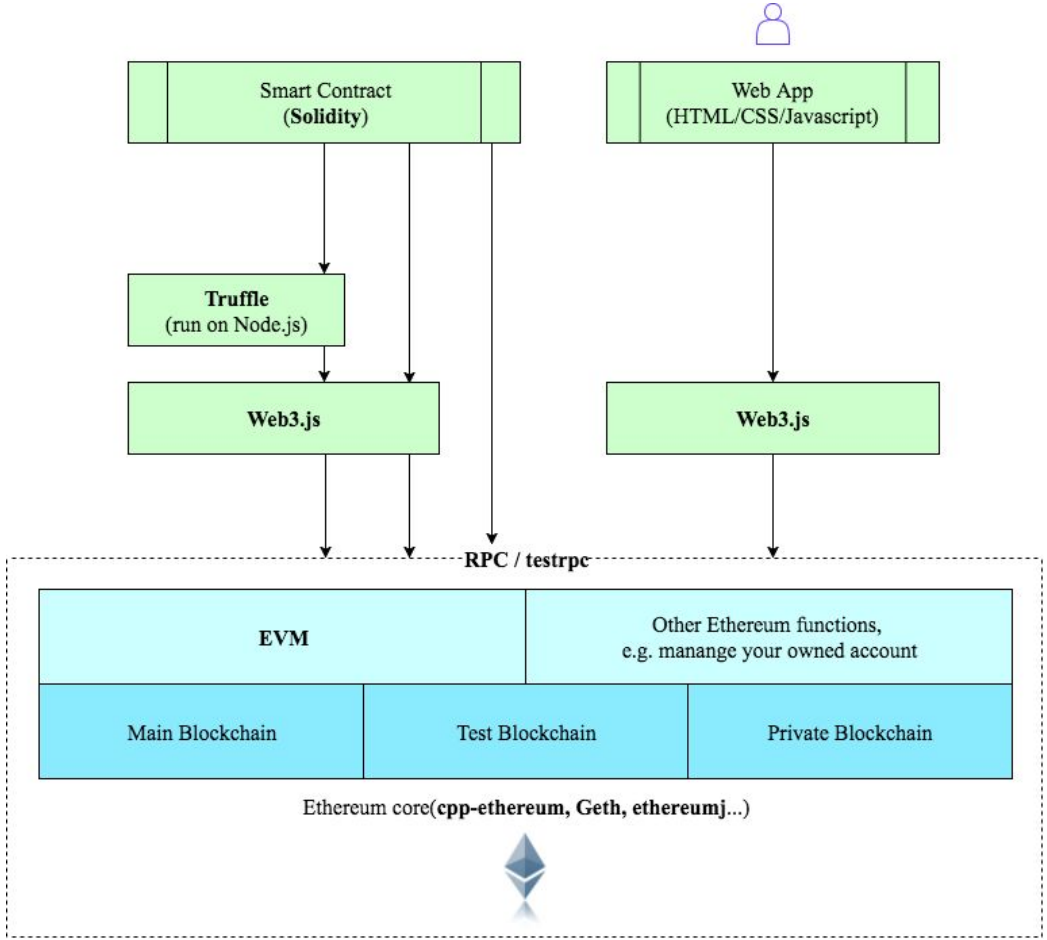
Traditional Web App



Web3 Ðapp



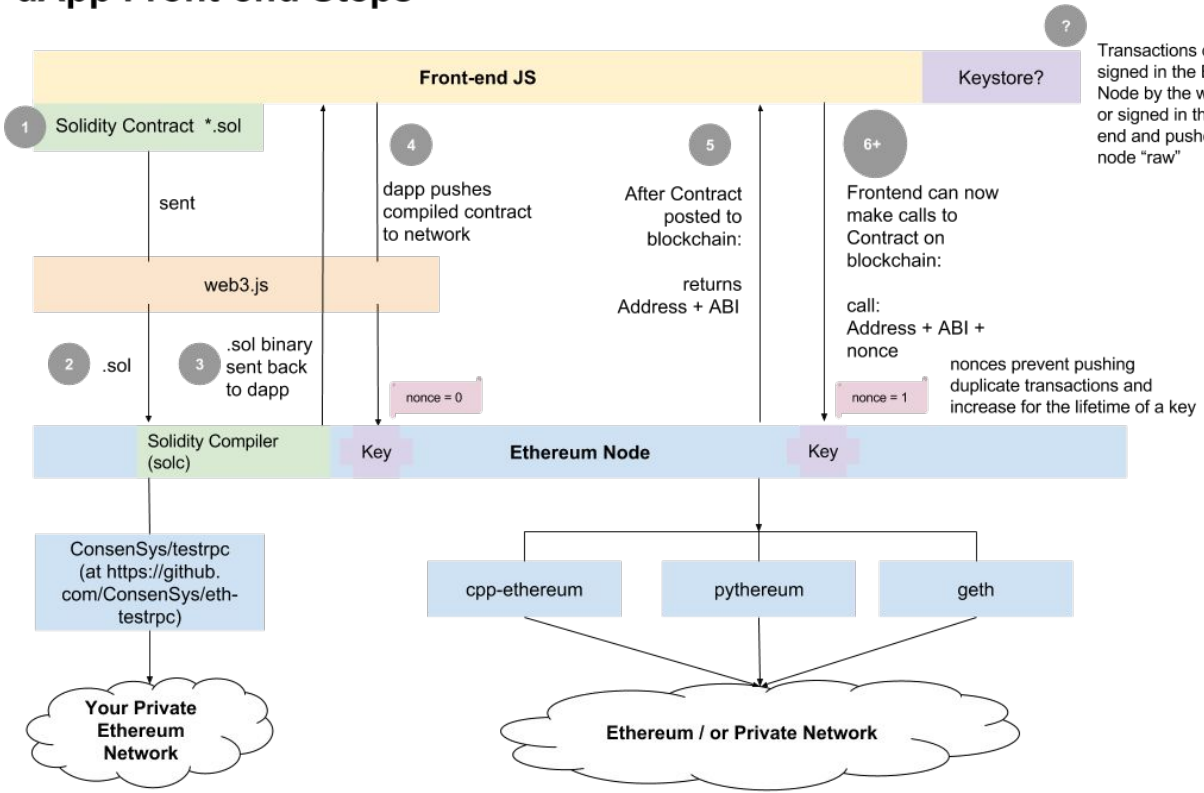
http://tyleryasaka.me/blog/2017/01/14/ethereum-dapps.html



https://forum.ethereum.org/discussion/14411/simple-chart-of-how-smart-contract-and-web-tools-work-together-on-ethereum

Mist/Parity (browsers)
Metamask (Chrome extension)
Infura (public node)

dApp Front-end Steps



A **Contract Creation Transaction** is shown in steps 1-5 at above.

An **Ether Transfer** or **Function Call Transaction** is assumed in step 6.

https://medium.com/@ConsenSys/a-101-noob-intro-to-programming-smart-contracts-on-ethereum-695d15c1dab4

CryptoKitties

https://stampery.com

https://kyc-chain.com

https://www.augur.net

http://www.bspend.com/playeth