



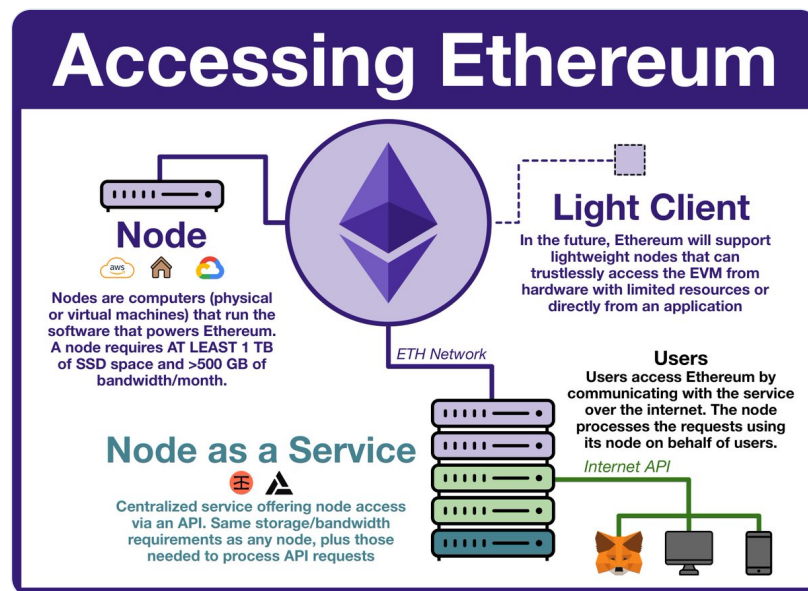
Haym @SalomonCrypto

Nov 9 · 29 tweets · [SalomonCrypto/status/1590137467524435971](https://twitter.com/SalomonCrypto/status/1590137467524435971)

(1/28) [@Ethereum](#) Basics: Accessing the World Computer

In order to access Ethereum, you need a node. But running a node is a big deal, it requires serious hardware, bandwidth and active maintenance.

Today we have centralized solutions, but the future of Ethereum is trustless.



(2/28) [@ethereum](#) is the World Computer, a single, globally shared computing platform that exists in the space between a network of 1,000s of computers (nodes).



Haym

@SalomonCrypto · [Follow](#)



(1/21) [@ethereum](#): The Big Picture

From 1492 to 2022, the context, technology and vision of the World Computer. The complete, top-to-bottom case for [\\$ETH](#).

An (unprecedented) mega-thread.



3:00 PM · Sep 3, 2022



[Read the full conversation on Twitter](#)



993



Reply




Copy link

[Read 47 replies](#)

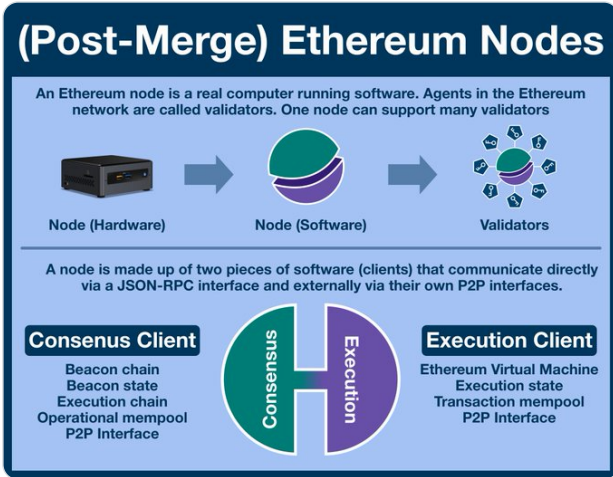
(3/28) You can think of a node in two ways: a real-world computer or the group of software needed to access, validate and otherwise run the World Computer.

Under the hood, a node is actually two independent pieces of software: a consensus client and an execution client.

 **Haym**
@SalomonCrypto · [Follow](#)

(1/16) @ethereum Basics: Nodes

At the end of the day, Ethereum is running atop IRL computers, each running the software that powers the World Computer. But what actually is a node? How does it relate to the different parts of Ethereum? How do all the pieces communicate?



(Post-Merge) Ethereum Nodes

An Ethereum node is a real computer running software. Agents in the Ethereum network are called validators. One node can support many validators


Node (Hardware) → Node (Software) → Validators

A node is made up of two pieces of software (clients) that communicate directly via a JSON-RPC interface and externally via their own P2P interfaces.

Consensus Client
Beacon chain
Beacon state
Execution chain
Operational mempool
P2P Interface

Execution Client
Ethereum Virtual Machine
Execution state
Transaction mempool
P2P Interface

2:30 AM · Nov 8, 2022

 [Read the full conversation on Twitter](#)

363 Reply Copy link

[Read 11 replies](#)

(4/28) Think like this:

Consensus client: responsible for Proof of Stake (PoS), securing @ethereum with the value of \$ETH

Execution client: responsible for operating the computing platform of the World Computer (the EVM).

These clients are HEAVY DUTY pieces of software.

(5/28) < NOTE >

Over the next few tweets I am going to walk you through the resource requirements of an [@ethereum](#) node in November 2022.

All the screenshots you see are real data pulls from the node I operate, so keep that in mind (for better or worse).

< /NOTE >

(6/28) First, let's look at the execution client (pictured here: [@go_ethereum](#)).

This is hard to digest, but the details aren't super important. The summary is ~425 GB to hold the EVM state and ~400 GB for historical data (although we can reduce this).

DATABASE	CATEGORY	SIZE	ITEMS
Key-Value store	Headers	49.25 MiB	90038
Key-Value store	Bodies	6.50 GiB	90038
Key-Value store	Receipt lists	5.33 GiB	90038
Key-Value store	Difficulties	10.11 MiB	119759
Key-Value store	Block number->hash	9.00 MiB	119568
Key-Value store	Block hash->number	622.72 MiB	15926161
Key-Value store	Transaction index	14.09 GiB	420137185
Key-Value store	Bloombit index	2.98 GiB	7966513
Key-Value store	Contract codes	4.54 GiB	729971
Key-Value store	Trie nodes	349.49 GiB	1940176772
Key-Value store	Trie preimages	547.13 KiB	8893
Key-Value store	Account snapshot	8.51 GiB	187028636
Key-Value store	Storage snapshot	59.87 GiB	837893492
Key-Value store	Beacon sync headers	0.00 B	0
Key-Value store	Clique snapshots	0.00 B	0
Key-Value store	Singleton metadata	7.78 MiB	16
Ancient store	Headers	7.11 GiB	15836124
Ancient store	Bodies	254.80 GiB	15836124
Ancient store	Receipt lists	121.45 GiB	15836124
Ancient store	Difficulties	249.75 MiB	15836124
Ancient store	Block number->hash	573.90 MiB	15836124
Light client	CHT trie nodes	0.00 B	0
Light client	Bloom trie nodes	0.00 B	0
TOTAL		836.15 GiB	

(7/28) If you were to start a new node from scratch, your node would not have ~400 GBs of historical data; yours would be (effectively) 0 GBs.

Once online, your node begins communicating with the network and updating the EVM. The old data is moved into a historical database.

(8/28) (Most) historical databases grows by ~15GB/week; this will quickly grow out of control and overwhelm any non-data center node.

Fortunately, (most) execution clients can "prune" this database, discarding all historical records before the most recent snapshot.

(9/28) Now let's take a look at the consensus client (pictured here: [@sigp_io](#)).

Here things are a little simpler; you can see that (my specific instance of Lighthouse at this specific moment) is taking up 172 GB.

```
16K    /ethclient/lighthouse/beacon/network
51G    /ethclient/lighthouse/beacon/freezer_db
1.1G   /ethclient/lighthouse/beacon/logs
120G   /ethclient/lighthouse/beacon/chain_db
172G   /ethclient/lighthouse/beacon
172G   /ethclient/lighthouse
```

(10/28) There is a lot more we can do to bring this number down.

For starters, Lighthouse's db can be stored in as little as 15 GB (at the cost of performance).

Or you can switch to [@ethnimbus](#) which has been designed from first principles to be as lightweight as possible.

(11/28) Bottom line is there are a lot of configurations you can alter to change the storage size of your node, but no matter what it's going to be BIG (>500 GBs).

Here's what the [@Rocket_Pool](#) docs say.

The following are considered *minimum* requirements:

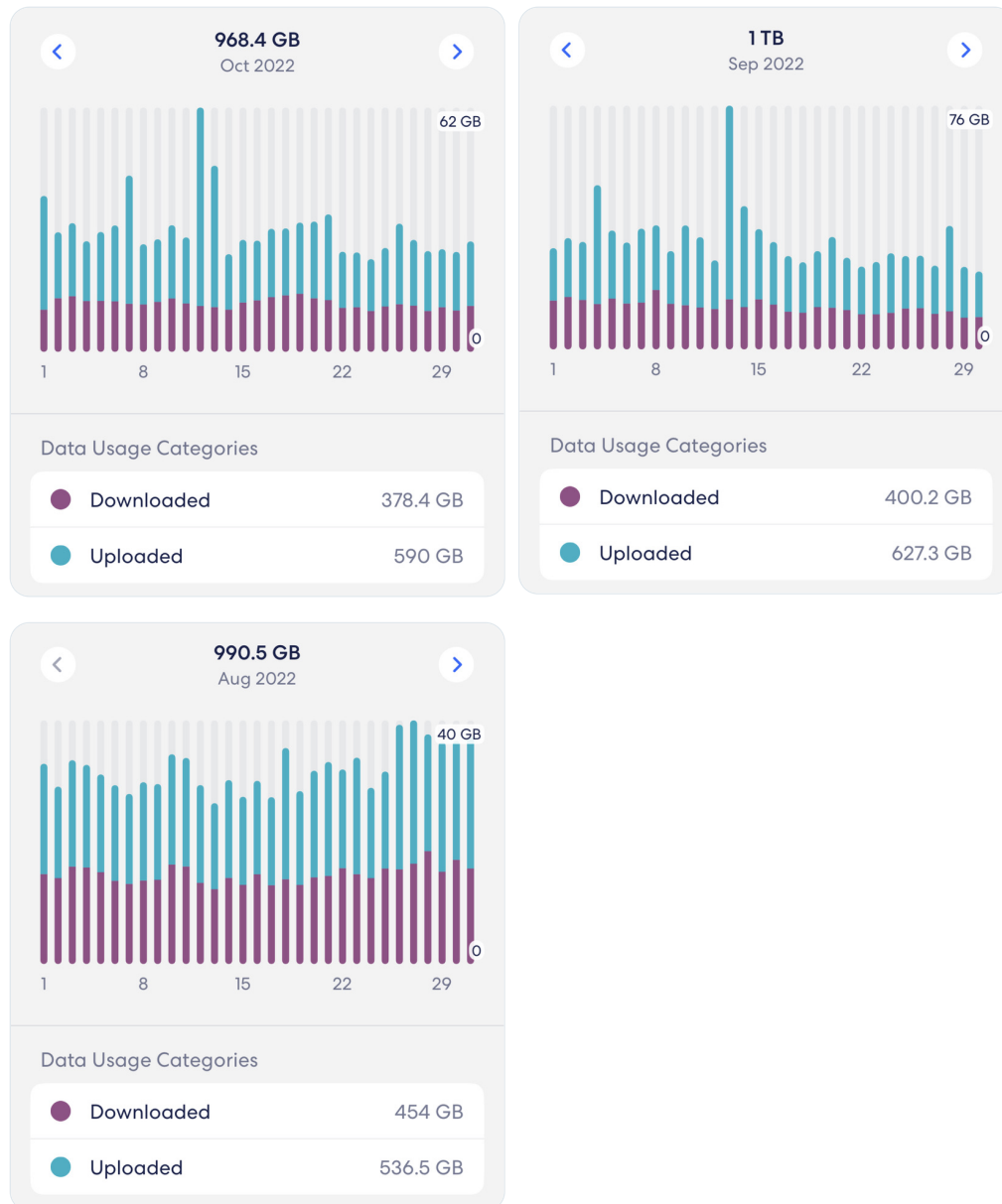
- Linux or macOS Operating System
- Quad core CPU (or dual-core hyperthreaded); both `x64` and `arm64` are supported
- 8 GB of RAM (preferably DDR4)
- 2 TB of free **SSD Disk Space** (*note: this only applies to mainnet; the Prater testnet only requires about 200 GB*)
 - A spinning platter hard drive *is generally not fast enough* to handle the constant random reads and writes that blockchain activity requires; you **MUST** use a solid state drive.

(12/28) You will also notice the CPU requirements. If "Quad core CPU (or dual-core hyperthreaded)" doesn't mean anything to you, you can just think "high end CPU made in the last 2-3 years."

Point is, a node requires a serious CPU.

(13/28) The hardware is not the only IRL resource required to run a node. You also need bandwidth. Lots and lots of bandwidth.

Right now, you're looking at something on the order of ~1 TB worth of bandwidth every month. That's ~13 GB download and ~20 GB upload every single day.



(14/28) We also need to consider the maintenance required to run a node. First and foremost, if a node is supporting one or more validators it MUST stay online 24/7; offline time is penalized by confiscating it's staked \$ETH.

(15/28) But even if the node is not staking \$ETH (merely acting as an access point into [@ethereum](#), not a validator), maintenance is still critical.

If the node falls out of sync, it has to spend a significant amount of time catching back up (and a sync from scratch? 2+ days)

(16/28) Taken all together, running a node is a big deal. Don't get me wrong, it's totally doable and I highly encourage you all to look into it, but it is serious business.

Serious enough that many people interested in accessing [@ethereum](#) have no desire or ability to run one.

(17/28) Just to drive the point home, consider two examples:

Let's say Alice wants to access De-Fi, but doesn't have any experience operating software outside a consumer setting.

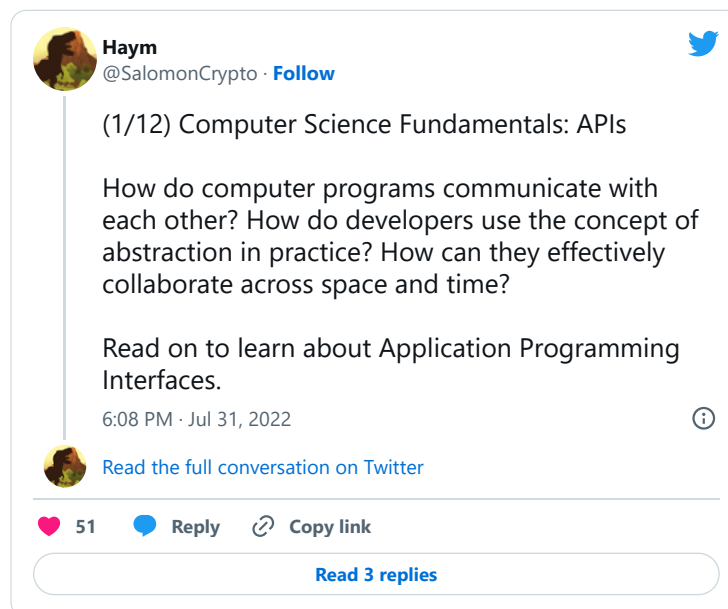
Or let's look at your browser wallet. It definitely doesn't have those hardware specs.

(18/28) Today, the way Alice, your wallet and most users access [@ethereum](#) is through a centralized Node-as-a-Service provider, like [@AlchemyPlatform](#) or [@infura.io](#).

These companies operate full, normal Ethereum nodes, basically the same as if you were running one at home.

(19/28) Users can communicate with these nodes over the internet using an API.

They send their requests to the Node-as-a-Service provider who then interacts with [@ethereum](#) on their behalf (returning any output data through the same API).

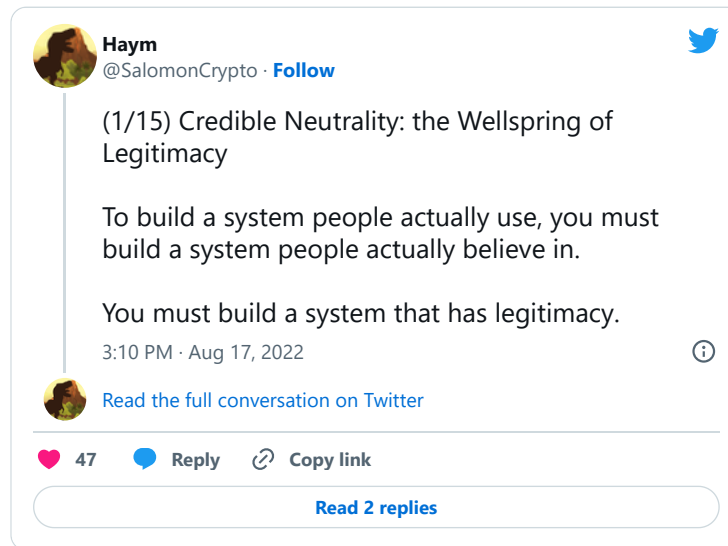


(20/28) This solution is super scalable; these two companies alone power the VAST majority of [@ethereum](#) requests. For example, every time [@MetaMask](#) does basically anything it does it through [@infura.io](#).

The problem? We've basically given up decentralization.

(21/28) Bottom line is that if we give up decentralization, we give up credible neutrality and [@ethereum](#) along with it.

If the only way nearly everyone can access the World Computer is via [@AlchemyPlatform](#)'s servers, we've basically just created a shitty version of AWS.



(22/28) Fortunately, that is NOT the endgame that we are all building toward. The [@ethereum](#) of the future is an Ethereum that supports light clients.

A light client is able to trustlessly connect with [@Ethereum](#) without running a full (heavy) node.

(23/28) A light client world is a completely different world; a MUCH cooler world.

We'll start by placing one directly in your browser smart wallet. Then every user can directly, trustlessly interact with [@ethereum](#).

Then, the only limit is your imagination.

(24/28) Light client low hanging fruit: low-spec'ed computers, computers without steady access to that much bandwidth, smartphones, terminals, vending machines...

Let's just go for it... we can put an [@ethereum](#) light client in a smart contract on another blockchain!

(25/28) Today, we don't have light clients. The software hasn't been built and, besides, [@ethereum](#) can't really support them yet anyway.

But we have gotten started. We've already built some of the more critical plumbing in to Ethereum PoS.

**Haym**
@SalomonCrypto · [Follow](#)



(1/25) [@ethereum](#) Fundamentals: Sync Committees and Light Clients

Just about one year ago, the Altair upgrade gave validators a new duty: sync committee. Learn how this core consensus feature enables the end game:

A truly decentralized World Computer

Ethereum Light Clients

Full Node

The node builds the aggregate public key based on the full validator set and provides it to the validator



Public key verification ensures the block has not been tampered with





Validator Sync Committee

Light Client

The light client builds the aggregate public key by extracting the sync committee from the block



Public key verification ensures the block has not been tampered with



1:30 AM · Oct 10, 2022 

 [Read the full conversation on Twitter](#)

 264  Reply  Copy link

[Read 22 replies](#)

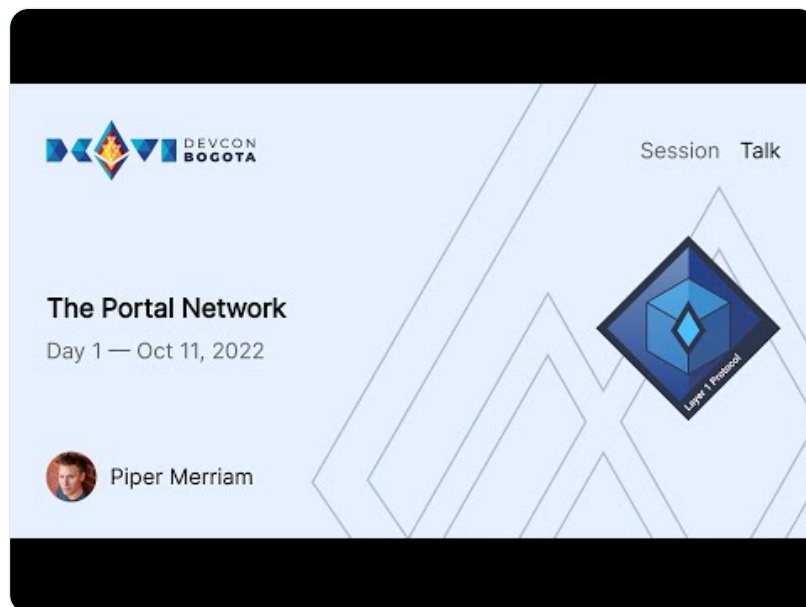
(26/28) Implementing state expiry and statelessness will be critical enabling light clients, but it is not enough.

Just take a look at our storage breakdown; only ~40% can be attributed to the state. We need to enable a MUCH lighter touch to achieve the light client-vision.



(27/28) But, as always with the World Computer, if the problems are understood then someone is probably working on it. This time, [@pipermerriam](#) (Portal Network) has us covered.

Feel free to watch ahead, but don't worry... this is where we are going.



<https://www.youtube.com/embed/0stc9jnQLXA>

(28/28) In conclusion, today there are two ways to access the World Computer: run your own node or ask a centralized actor to do it for you.

But that's just today, [@ethereum](#) is still evolving. Don't fall for the trap of believing this is all we get.

You ain't seen nothing yet.

**Haym**
@SalomonCrypto · [Follow](#)

(1/25) [@Ethereum](#) Roadmap: Middleware

Think back to 2015, does the Ethereum we have today look like what you were imagining back then?

Now think forward to 2030, or even 2122. What will that version of the World Computer look like?

Are you ready for the Middleware Gold Rush?



5:01 PM · Nov 6, 2022

 [Read the full conversation on Twitter](#)

 319

 Reply

 Copy link

[Read 9 replies](#)

More of a long-form reader? Try this:

**Haym**
@SalomonCrypto



Accessing Ethereum

Accessing Ethereum | Haym

(1/28) @Ethereum Basics: Accessing the World Computer In order to access Ethereum, you need a node. But running a node is a big deal, it requires serious hardware, bandwidth and active maintenance. ...

<https://typefully.com/SalomonCrypto/EapkQW6>

Like what you read? Help me spread the word by retweeting the thread (linked below).

Follow me for more explainers and as much alpha as I can possibly serve.



Haym
@SalomonCrypto · [Follow](#)



(1/28) @Ethereum Basics: Accessing the World Computer

In order to access Ethereum, you need a node. But running a node is a big deal, it requires serious hardware, bandwidth and active maintenance.

Today we have centralized solutions, but the future of Ethereum is trustless.

Accessing Ethereum



Node
Nodes are computers (physical or virtual machines) that run the software that powers Ethereum. A node requires AT LEAST 1 TB of SSD space and >500 GB of bandwidth/month.

Light Client
In the future, Ethereum will support lightweight nodes that can trustlessly access the EVM from hardware with limited resources or directly from an application

Node as a Service
Centralized service offering node access via an API. Same storage/bandwidth requirements as any node, plus those needed to process API requests

Users
Users access Ethereum by communicating with the service over the internet. The node processes the requests using its node on behalf of users.

ETH Network

Internet API

12:21 AM · Nov 9, 2022

 [Read the full conversation on Twitter](#)

 295  Reply  Copy link

[Read 14 replies](#)

...