

Capital and Enclosure in Software Commons Linux and Ethereum

• Trent Van Epps



Summer of Protocols | 2023

© 2023 Ethereum Foundation. All contributions are the property of their respective authors and are used by Ethereum Foundation under license. All contributions are licensed by their respective authors under CC BY-NC 4.0. After 2026-12-13, all contributions will be licensed by their respective authors under CC BY 4.0. Learn more at: summerofprotocols.com/ccplus-license-2023

Summer of Protocols :: summerofprotocols.com

ISBN-13: 978-1-962872-97-3 print
Printed in the United States of America
Printing history: July 2024

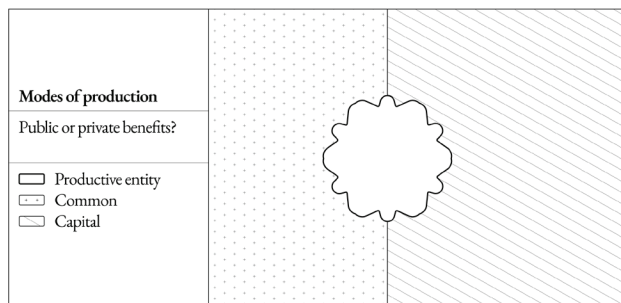
Inquiries :: hello@summerofprotocols.com

Cover illustration by Midjourney :: prompt engineering by Josh Davis
«large circles with other circles encroaching, exploded technical diagram»

Entities that extract profits from software commons like Linux and Ethereum have the greatest incentive and capacity to co-opt them.

1. Common and commons

As Giuliani & Vercellone argue, *common* is a mode of resource production in the same way capital is.¹ These modes have *internal logic[s]* (Wright, 2020) which animate productive entities (e.g., commons, companies).² The allocation of benefits produced by the entity determines which mode it aligns with.



- Common asks: How can the benefits/assurances/surplus value be distributed to the largest public?
- Capital asks: How can the benefits/assurances/surplus value accumulate to private entities?

The standard framework for comparing the resources produced by these systems considers two main characteristics:

- *Excludability*: can anyone be prevented from using the resource?
- *Rivalry*: does someone's use of the resource prevent another from using it? This term is also known as *subtractability*: to what degree someone's use removes a portion from the total available.

1. www.researchgate.net/publication/336605244_From_New_Institutional_Economics_of_the_Commons_to_the_Common_as_a_Mode_of_Production
 2. ianwrightsite.wordpress.com/2020/09/03/marx-on-capital-as-a-real-god-2/

	Anti-Rival	Rival	Non-Rival
Non-Excludable	<i>Symbiotic Goods</i> open source software, information, blockchains	<i>Shared Goods</i> blockspace, city metro, fisheries, forests, groundwater basins	<i>Public Goods</i> weather forecasts, peace and security of a community
Excludable	<i>Network Goods</i> encrypted data, social media, Privacy Pools	<i>Private Goods</i> food, clothing, tokens	<i>Club Goods</i> theaters, private clubs, daycare centers, token-gated chats

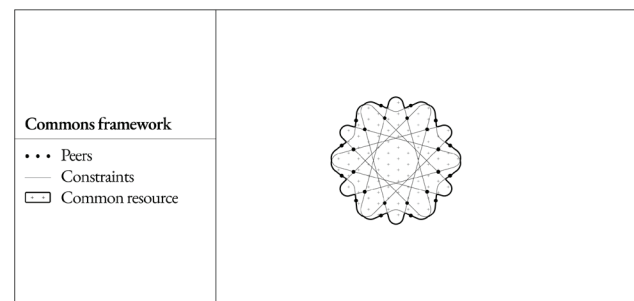
Adapted from Nikander et al. (2020) and the Fractal ID Team (2021)

Software (e.g., Linux, Ethereum) is a Symbiotic Good:

- *Anti-rival*: From Nikander et al. (2020): "For many digital goods, their real (social) value . . . increases the more they are used, in stark contrast to material goods, whose value almost always decreases as they are used and consumed." As a software protocol is more broadly adopted, each user benefits from a wider array of associated services, products, shared local knowledge. As digital artifacts, the cost to produce one additional unit is negligible (i.e., approaches a marginal cost of 0).
- *Non-excludable*: Open Source Software (OSS) licensing enables actors external to the production process to use, modify, and suggest changes to the software. Different licenses on this software can create different levels of excludability, but the overall expectation is that others will use and modify the software, and that the code is publicly accessible (aka "source available").

Commons

Each *commons* is a local manifestation of the greater **common** mode logic. Here's my framing for how they produce and allocate resources.



Peers

A set of individual actors come together to build something of mutual interest. As peers hosted within a commons process, they share the responsibility of shaping the system. They bring their unique perspectives, experiences, and tools which inform their labor.

Constraints

The labor of contributors is structured within constraints:

- Pre-existing: the weather of a particular geography, the licenses attached to existing software
- Self-imposed: social/political/economic relationships, decision-making, norms, frameworks of production, etc.

Constraints define the obligations expected of and rights given to contributors. They also directly affect the inherent characteristics—the shape—that the resource takes on.

Resource

The underlying process, resource output, surplus value/positive externalities generated in the production are all fully available to the entire contributor graph. It's unlikely that commons resource would be stewarded in the same way or have the same characteristics outside this set of peer relationships. Yochai Benkler describes this dynamic as “commons based peer production.”³

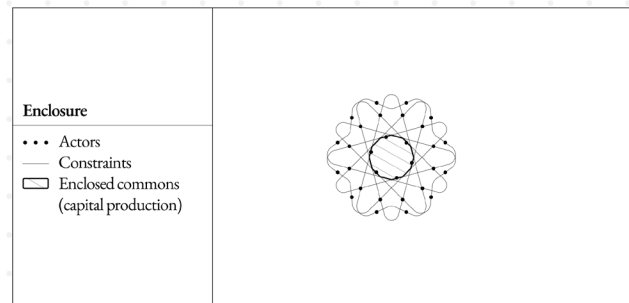
It's interesting to frame the resource and its structure as an entity, or an “egregore.” This is a non-physical being which

exists in virtue of the collective ritual activities of a group yet operates autonomously, according to its own internal logic, to materially influence and control the group's activities. The group creates the egregore, and the egregore creates the group, in a self-reinforcing feedback loop.⁴

Enclosure

In the common mode of production, the benefits/assurances/surplus value are under the domain of involved peers. Because commons are suited for producing unmonetized and yet still valuable artifacts, other modes like capital find their way to the edges. This produces a dynamic where external actors may encroach on the peer production, or co-opt part or all of the resource output towards private benefit. Academia calls it “enclosure,”⁵ Birkinbine calls it “incorporation,” and the Ethereum community calls it “capture” or in another frame, Moloch: the God of coordination failures.⁶

The phenomenon is a suffocating constriction of the possibility space.



The diagram above illustrates a company or a former commons—now fully enclosed by capital. The resource output (i.e., surplus value) is only allocated to a subset of the membership.

Consider the historic enclosure of English pastures.⁷ These shared spaces provided grass to support livestock, which in turn produced manure for fertilizing fields. This production was maintained through social agreements between the local community members.

However, things changed with the royal appointment of a new baron. As he erected walls and combined plots, the previous stewards were prevented from utilizing the land according to their “ancient customary rights.” These have been superseded by a new framework, with the baron and his house at the center. This new dynamic is bootstrapped under a regime of physical violence, and normalized

3. en.wikipedia.org/wiki/Commons-based_peer_production

4. ianwrightsite.wordpress.com/2020/09/03/marx-on-capital-as-a-real-god-2/

5. en.wikipedia.org/wiki/Enclosure

6. slatestarcodex.com/2014/07/30/meditations-on-moloch/

7. en.wikipedia.org/wiki/Enclosure

over time by social, political, or religious justification.

Digital objects are able to sidestep some pressures which physical objects cannot. However, subtle but dangerous encroachment may surface in the political, social, or technical processes of software production. In fact, the surplus value bound up within the anti-rivalry of networked protocols may make them even more attractive.

2. Software Commons

This next section will examine Linux and Ethereum as software commons, including their:

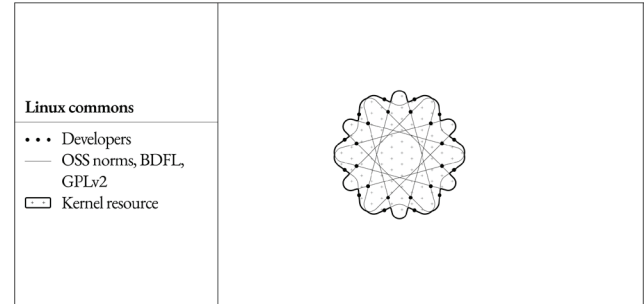
- Resource outputs
 - Who can consume them?
 - How are they used by contributors or others not involved in production?
 - What are the unique characteristics that they have?
- Constituent parts
 - Who can contribute to stewardship?
 - What types of labor are needed?
 - Which frameworks guide the work of contributors?
- Capital entities
 - How do they engage with the output or the production process?
 - What does enclosure look like?

As you read, consider the similarities and differences between the two projects. What can the Ethereum community learn from Linux's 33 years?

2.1 Linux

Linux was started in 1991 by then-student Linus Torvalds. In 2001, Microsoft's Steve Ballmer called it "a cancer."⁸ Today, the project has grown to become what many consider the largest collaborative development project.

8. www.infoworld.com/article/3680048/where-microsofts-open-source-policy-went-wrong.html#:~:text=In%20



Peers—Developers

As of the December 2022 6.1 release, the vast majority of Linux contributors are employed by companies who offer Linux-related products or services. Contributors with no company affiliation were only 4% by changesets/PRs or 2.8% by lines changed. The largest employers were Huawei (9.2% by changesets/PRs) and Oracle (12% by lines changed).

Constraints—OSS norms, BDFL, GPLv2

- OSS etiquette—e.g., detailed pull requests, bug reports.
- Social dynamics: contributors must cooperate to create software of this scale and complexity. Linus has been unofficially designated BDFL (benevolent dictator for life). The term recognizes the significant weight, and even deference, the community gives his opinions on the direction of Linux.
- Legal frameworks: the General Public License v2 (GPLv2)⁹ allows software to interface with the legal system. "You may copy, distribute and modify the software as long as you track changes/dates in source files. Any modifications to or software including (via compiler) GPL-licensed code must also be made available under the GPL along with build & install instructions."
- Labor hierarchies: formal/informal teams work on different parts of the kernel. Various roles include writing code, reviewing, testing, long-term maintenance for older versions, who has merge access, etc.

2001%2C%20then%2DMicrosoft%20CEO,aim%20at%20piracy%20in%20the

9. www.gnu.org/licenses/old-licenses/gpl-2.0.en.html

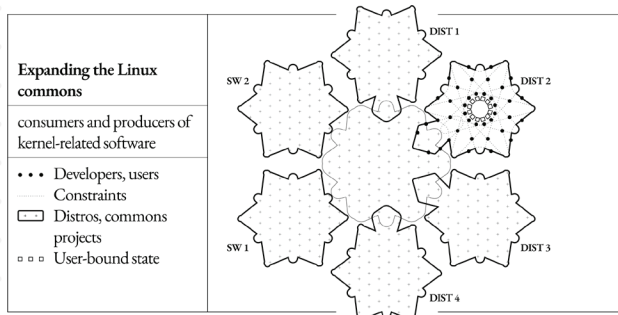
Resource: Kernel

The Linux kernel is the *resource* produced by this commons process.

- The core component of the Linux operating system (OS), made up of 30mm lines of code
- In contrast to most web standards, no spec. According to Linus, “specs are close to useless.”¹⁰
- *Low excludability*: the kernel source code is available for anyone to use.
- *Anti-rival*: one person consuming (reading or running) this digital code does not restrict another’s ability to do the same, and they both benefit from there being more Linux users.
- The kernel cannot be run as user software in isolation.

Expanding the Linux commons

The kernel ecosystem is extended by other communities and projects, here under the umbrella of consumers and producers of kernel related software. Software commons are especially generative in how they allow adjacent ecosystems to emerge.



One of the main community types are called “distributions,” aka “distros” (DIST 1-4). They extend the kernel to create useful software for end users, e.g., Ubuntu, Fedora and Debian. Other non-distro software projects (SW 1, 2) create a wide array of distro related plugins and tools. These projects share some important resources with the base commons:

- *Resource*: All projects must interface with the kernel source code to varying degrees
- *Contributors*: some may be part of both the distro and kernel production processes.
- *Constraints*: core norms and frameworks likely carry over. but it’s also possible they get diluted with distance, or the project hosting context, e.g., within a large commercial entity vs. as a small community project.

User-bound state

Each square is an instance of distro software being run by a user, whether they are individuals, teams, or organizations. The boundaries of each running distro system traces the shape of each user profile, e.g., a corporation running an enterprise Linux OS to manage internal processes, customer records, etc. The software is inward facing.

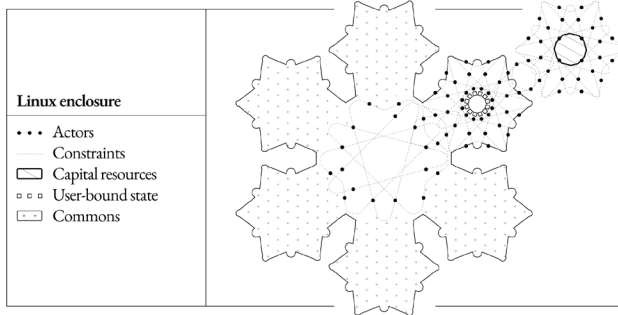
User feedback is an important part of software commons: what’s working, what needs to be improved, and feature requests.

Linux enclosure

I will caveat this section by acknowledging that companies (and their employees) engage in the commons as a condition of their environment, not as abstract adversarial actors. Enterprise software purchasers want certain guarantees when purchasing software: contractually obligated long-term support, or an entity to serve suits to. Though it may be harder for a commons to provide these same guarantees, it certainly doesn’t justify the negative effects brought by conflict between the two modes of production.

Further, the size and complexity of the Linux commons makes it unlikely to ever be completely enclosed (i.e., the medieval pasture example in the first section, a single entity controlling production). There are significant costs to production which would eat into the benefits realized by free-riding on the surplus value of the commons.

10. web.archive.org/web/20060821225841/kerneltrap.org/node/5725



Enclosure happens in degrees, starting at the edges and working its way in. Commercial entities engage in shared production so long as it serves their profit motive.

- Commercial entities hiring/acquiring kernel contributors grants access to their social/political/technical capabilities at the center of the commons. It ostensibly appears as a positive move to share the responsibility of commons stewardship. However, through incentives, pressure, or inertia, employees may end up aligning more with the goals of the employer instead of what the commons needs.
- Commercial entities may also act as a legal host for a distro community, or deeply align their products. In Linux, capital entities are able to fully enclose/host entire distros because that's where the software production process ends.

In both of these approaches, contributors simultaneously participate in the common and capital modes of production. Benkler describes this as “the boundary of the firm becom[ing] more porous.”¹¹

Case study: Red Hat

Founded in 1993, Red Hat (RH) has decades of Linux grassroots experience. They've used that to build what some call the most financially successful open source company, leading to a \$34B acquisition in 2018. However, this path to success has often come at the expense of commons stability. Consider this 10 year progression:

11. https://cyber.harvard.edu/wealth_of_networks/Sentence-sliced_Text_Chapter_4

- In 2014, RH “partnered” with CentOS: a struggling distro project. RH believed the state of the project reflected poorly on their work. In exchange for legal support, job security for contributors, and stability, Red Hat was given a permanent board majority and de facto control of the project.¹²
- In 2019, RH was acquired by IBM. Company posts from the time claimed that “Red Hat’s mission and unwavering commitment to open source will remain unchanged.”¹³
- The honeymoon didn’t last very long. In 2020 RH deprecated CentOS, which had by then been reinvigorated as a popular community distro. Future work and support for past releases was terminated. RH directed previous CentOS users to CentOS Stream, a new project missing most of the “Red Hat Enterprise Linux” (RHEL) compatibility developers were looking for.¹⁴
- In 2023 RH restricted access to RHEL source code—their flagship product. Many in the Linux community believe this violates the spirit of long-standing GPLv2 licensing norms.¹⁵ Even though RH employees “were also conflicted about the new policy,” the external interests of the commercial actor ultimately trump their employees’ well-intentioned posture towards commons norms.¹⁶

A relationship which depended on the health of the commons became more clearly representative of investors and capital.

Seeking resources to sustain their efforts, grassroots communities may unwittingly shape themselves to accommodate entities uninterested in the health of the commons. When priorities shift, capital is happy to disengage from committing resources. This dependency

12. <https://arstechnica.com/gadgets/2020/12/centos-shifts-from-red-hat-unbranded-to-red-hat-beta/>

13. <https://www.redhat.com/en/about/press-releases/ibm-closes-landmark-acquisition-red-hat-34-billion-defines-open-hybrid-cloud-future>

14. <https://arstechnica.com/gadgets/2020/12/centos-shifts-from-red-hat-unbranded-to-red-hat-beta/>

15. <https://www.opensourceforu.com/2023/07/red-hat-new-source-code-policy-sparks-controversy/>

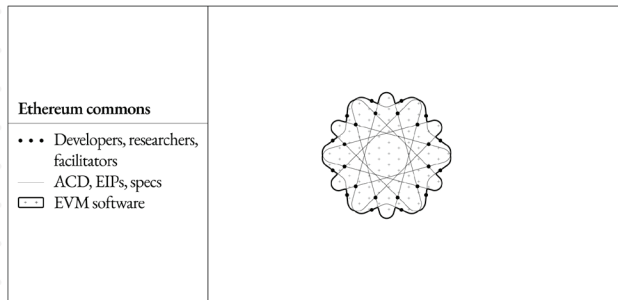
16. www.newsobserver.com/news/business/article279099964.html

dynamic introduces the risk of structural instability for the greater self-reliant commons.

At the end of the day, capital's bottom line is top of mind.

2.2 Ethereum

Ethereum was announced in 2013 by former journalist Vitalik Buterin, and went live in 2015 under the audacious tagline of “world computer.” In 2018, venture capitalist Fred Wilson suggested in strong terms that “Ethereum should be more like a company.”¹⁷ Today, it has become the world's largest blockchain network.



Peers—Developers, Researchers, Facilitators

People with expertise in the areas of distributed systems, networking, game theory, mechanism design, cryptography, virtual machines, etc. These individuals include:

- **Developers:** maintaining and preparing client releases. Sanity checking research ideas for practicality and viability, implementing proof of concepts.
- **Researchers:** dissecting the realities of the network as it exists today, understanding deficiencies that can be improved in the future, crafting proposals to make it more robust & censorship resistant
- **Facilitators:** coordinating network upgrades, ensuring upgrades balance the needs of the ecosystem and the protocol, and devops, security and testing in support of client teams

For-profit and nonprofits provide many contributors with financial support in a variety of

17. youtu.be/K4UNOv6SUcQ?t=2330

forms. There are some unaffiliated individuals who contribute on their own for a variety of reasons, including curiosity, intellectual challenge, respect from peers, etc.

Constraints—ACD, EIPs, Specs

In addition to OSS norms found in the Linux Commons around respecting licenses and collaborative efforts, the Ethereum community holds shared values:

- **censorship resistance:** including both accessibility for users, and decentralization for the network
- **autonomy/self-determination**
- **permissionlessness**
- **privacy advocacy**

Venues like the All Core Devs call,¹⁸ Eth R&D Discord, ethresear.ch,¹⁹ and Ethereum Magician's forum²⁰ host discussions for a globally distributed contributor set. Participants are able to build consensus around which particular changes are of highest priority.

The EIP process structures the way changes to the protocol are specified.

Ethereum Execution Layer Specification (aka EELS),²¹ Consensus Specs,²² Yellow Paper,²³ Ben Edgington's Upgrading Ethereum²⁴ — these specifications and related items are an important set of artifacts that steer how the protocol develops and describes itself. They range from the purely representational to functional testing components and are used at various points in the network upgrade process. This library of references is a key enabler for Ethereum's robust offering of 10+ unique clients which can all interoperate within this distributed system

Social dynamics: there is no single leader. Even though Vitalik's opinions do hold significant weight, he has largely stepped back from regular involvement in core protocol

18. github.com/ethereum/pm

19. ethresear.ch

20. ethereum-magicians.org

21. github.com/ethereum/execution-specs

22. github.com/ethereum/consensus-specs

23. ethereum.github.io/yellowpaper/paper.pdf

24. eth2book.info/capella

stewardship. He doesn't have any override power if social consensus disagrees with his perspective.


Resource—the Ethereum commons

The Ethereum commons produces a body of software artifacts which can come to consensus on the output of EVM computations run in globally distributed environments. This is a collective snapshot of what the Ethereum protocol is today and what it might look like in the near future. It reflects the values of its contributor set, is available for anyone to explore, and is open source.

This includes the client software which can read and write to EVM state. Each Ethereum client implements the spec with opinionated decisions on the language, features, database, research areas, and more:

- Execution layer—Erigon, Besu, Geth, Nethermind, Reth, EthJs
- Consensus layer—Lighthouse, Lodestar, Nimbus, Prysm, Teku

While many of the full-time contributors to each project are paid by a commercial entity which acts as the “host,” the set of contributors is typically broader than just the entity itself. The diversity of client approaches is the product of and reinforces the norm of political polycentrism. In an ideal case, there is a healthy distribution of many client types. This produces a more technically robust, fault-tolerant network. For example, a client in a new language opens a path to Ethereum contributions for any developer in that community, while also providing protection against issues that might appear in other languages.

<p>Embedded capital circuits</p>	
<p>Containment or contradiction?</p> <p>••• Validators, node operators</p> <p><input type="checkbox"/> proof of stake, ETH, blockspace, LST, MEV, restaking</p> <p><input type="checkbox"/> Ethereum software commons</p>	

Embedded capital resources

The fact that the Ethereum commons produces a blockchain and not another form of software has important implications.

Usable right away

Whereas the Linux kernel doesn't produce anything for users until distros add their features, running Ethereum software produces a usable artifact right away. From the genesis block, Ethereum has always had a broad range of individual enthusiasts, communities, organizations, and infrastructure all using the chain at the same time. While this brings a welcome variety of use cases and perspectives, it can sometimes be challenging for the commons stewards to set feature priorities.

Single global artifact

Operators from around the world run the client software releases and come to consensus on a single instantiation of state and history (i.e., “mainnet” “chainID=1” “ETH L1”). In contrast to Linux (the kernel doesn't produce anything on its own, distros needed to produce many isolated/private instances), the Ethereum software commons outputs a single, public software artifact. This is a globally distributed system, *a world computer*. Users running the software as part of consensus activities are an important part of the feedback loop.

Reading state + history is *low excludability*, *anti-rival*: the more people using and referencing EVM infrastructure, the greater the value to the individual user. Anyone wanting to access this data can by running their own node. The chain data can be called a “Knowledge Commons”—the records are available for anyone to verify: a widely available, rich dataset of organizational, financial, and cultural relationships.

Writing to state is *low excludability*, *high rivalry*: the number of concurrent users is capped by the block gas limit to ensure that resource requirements for node operators don't grow too quickly. While it is a digital object, there are constraints to be mindful of: cost of

storage, bandwidth, compute, etc. In the happy path, the network participants only discriminate based on the fee offered by each transaction to write to the state. As Ethereum scales its trustless settlement assurances through Layer 2s, the degree of rivalry will decrease.

Embedded capital engines

Perhaps most significantly: the Ethereum commons has capital circuits directly embedded within itself at the level of production. Monetization/productization/commodification isn't added later by an independent company (as in Linux): the software is birthed with perpetual incentives.

- Proof of Work consensus depended on financial incentives, and Proof of Stake introduced an even more direct relationship between capital and consensus. This area of the protocol also sees growing engagement from extra-protocol actors (e.g., commercial staking services, liquid staking providers, restaking systems, etc), each with their own incentives.
- The native asset ether (ETH) is used to pay for state changes, and to reward validators for participating in consensus which advances the chain. While Ethereum's lack of onchain governance prevents capital having a direct voice (in the form of ETH) in stakeholder decision making, the asset holders may try to steer the protocol down suboptimal paths.
- Blockspace (and any associated MEV²⁵) can be considered a separate infrastructure/resource supply chain

In the perspective of this article, the embeddedness of capital feels contradictory, and potentially counterproductive to commons stewardship. There's a universe of possibilities, with two defining frames:

25. *Maximal extractable value* (MEV) refers to the maximum value that can be extracted from block production in excess of the standard block reward and gas fees by including, excluding, and changing the order of transactions in a block. ethereum.org/en/developers/docs/mev/

- Capital and the protocols/parameters it manifests as are ultimately subject to the influence of the software commons contributors. This allows for a healthy counterbalance to any influence capital is able to exert through the actors engaged in these systems.
- The close proximity causes unforeseen challenges to ongoing protocol stewardship. Capital creates more instability, unpredictability, and faster (and more complete) enclosure in the underlying commons.

Expanding the Ethereum commons

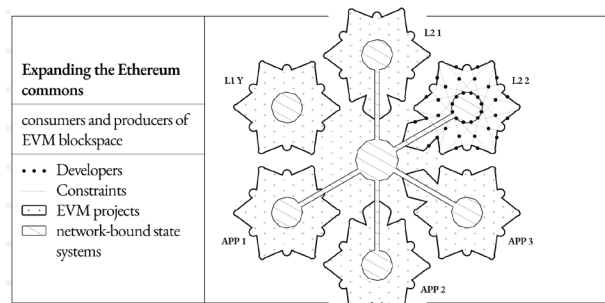
The broader Ethereum commons includes *producers* and *consumers* of EVM blockspace:

- Projects which modify the commons software to produce additional differentiated instances of EVM state/history/blockspace, e.g., other L1s, L2s
- Users transacting
- Applications developed by teams create demand

A broad array of entities are implicated in this production process because of how they share resources and constraints:

They *must* share code, software, information: all actors are building off of the EVM

They *might* share the labor of individuals. Their efforts may shape the core protocol to benefit all other blockspace producing/consuming actors (e.g., EIP-4844) or in the degenerate case, a single L2 trying to influence protocol decisions in their favor. The overlaps are illustrative and do not generalize to every occurrence of the type.



They *might* share norms on how to produce/steward the software.

They *might* share the same blockspace/state/history: network-bound state. This enhances the anti-rival characteristics of the resource.

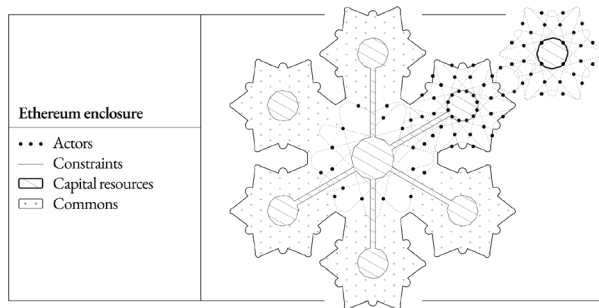
- Independent EVM chains, e.g., L1 Y. there is no blockspace relationship
- EVM chains which have (or are working towards) trustless guarantees for passing state between them (canonical bridges)—L2 A, L2 B. They consume state on the Ethereum L1
- Applications creating demand for blockspace
- Individuals consuming blockspace

To engage with the Ethereum software product, all of these actors are required to step outside of their immediate domain. Practically, this means they can't ever have exclusive control over the underlying software production or the resulting blockspace.

Ethereum enclosure

I'd like to preface this section with some caveats.

- It's possible for companies to share the values of the Ethereum commons. Companies are not default evil, and neither are companies with differentiated revenue streams. However, the fundamental goals of capital and the commons are incompatible.
- Though money/assets/resources will always be part of the commons management process—its effects are most strongly felt when it's seeking a return.



A large multinational corporation has projects adjacent to but outside of the commons. They acquire the team building a Layer 2, which gives them access to the political/social

relationships which comprise the Ethereum software commons.

Superficially, the goals of capital and the commons may seem coincident as “grow the impact of the blockspace product/system.” In the short term, and as expressed by individual actors, this may actually be true! However, on longer timelines, these goals are ultimately conflicting in terms of who the benefits accrue to, and in what form.

The commons asks: How can the benefits/assurances of the system be scaled to the largest public?

- Lowering barriers to verify state validity (running a node)
- Lowering barriers to produce blockspace (validating, censorship resistance)
- Lowering barriers to consume blockspace (permissionless user access, censorship resistance)

Capital asks: where can private benefit be extracted?

- Inserting into/extracting profit from state verification (explorers)
- Inserting into/extracting profit from blockspace production (MEV, hosted validation)
- Inserting into/extracting profit from blockspace consumption (user experiences, products)

There are constraints on what capital can do in pursuit of these goals, including to what degree and by what methods (how easy it is to fit to the capital entities chosen form):

- Isolated entities cannot fully enclose blockspace production systems without compromising crucial characteristics, i.e., availability, credible neutrality, censorship resistance.
- The EVM ecosystem shares the same public/private key infra, making user lock-in to a particular product less feasible—this increases user agency.
- Relative to Linux, the EVM ecosystem is a public system that lends itself to decomposition of the blockspace production stack, aka “protocol surface area.” This makes it harder for any single capital entity/small group to

dominate (absent cartelization) over longer timelines. Ultimately, “big C” Capital benefits even when projects fail to build profitable product and cycle out of the ecosystem.

- In the same way the expanded commons takes on some of the constraints, any systems integrating/layering on top have to be designed in a particular way. Both technically and legally with regard to OSS licenses.
- Relationships between blockspace (e.g., L1 and L2) cannot be broken without fundamentally changing the nature of the blockspace.

Given these limits, capital may turn to commons governance (i.e., any future protocol changes, aka the “process”) to give voice to its interests. This may materialize as:

- Commercial entities (e.g., Layer 2s, investment firms) competing between each other to influence the shape of core protocol features to their or their portfolio companies’ benefit or to prevent others from benefitting.
- Larger companies hosting client teams in order to access/leverage their local knowledge/position in the commons production, either by permanently acquiring a team or short-term consulting.
- Legacy organizations looking to stay ahead of obsolescence (e.g., web2 dipping its toes into web3) or to create regulatory moats.

Downsides to these actions include:

- Capital biases towards speed, not long-term sustainability. Timelines shorten to match the short-term benefit of actors, rather than the long-term benefit of the broader commons and the public welfare it is capable of producing.
- Commons stewardship that leans too heavily on the presence of capital (extrinsic motivations) risks crowding out intrinsic motivations (e.g., vibes, respect from peers, challenging oneself).

3. Conclusion

Software commons are an increasingly important part of our daily lives. The people and frameworks involved are able to provision novel goods outside of the state/market dichotomy.

Not all software commons will experience the same enclosure dynamics. Those that do can still exist and thrive in spite of it—perhaps it only becomes problematic when the scale of extraction becomes too large relative to the rest of the commons. It remains to be seen whether this conflict is an unavoidable early phase of commons bootstrapping or perhaps this will be a site of ever-increasing conflicts between the two modes.

It’s also possible for capital and the people wielding it to exercise restraint, i.e., “don’t kill the golden goose.” The underlying motivation (e.g., build a moat now to profit in the future) will eventually surface.

Linux is an ongoing demonstration of the possibilities and challenges involved in software commons stewardship. The Ethereum community should better develop a critical understanding of capital’s interest.

Today, the protocol community is largely aligned on the vision for near-term Ethereum. Layer 2 teams are a crucial component to making this work, and should be celebrated. In the future, will the different parts of the community agree on the right path forward?

Do native forms of capital embedded in resource production give a better ability to modify negative externalities? Or does it supercharge incentives for external capital to enter and co-opt the resource production system over time? How can we develop better heuristics for thinking about this challenge and developing safeguards?

There are decision-making norms which try to privilege the health of the network commons against apathy and “death by a thousand capital cuts.” As the number of profit-seeking stakeholders inevitably increases, will this always be the case? △

The views shared here are my own and may not necessarily represent those held by Protocol Guild members or the Ethereum Foundation.

I take inspiration from the work of Benjamin Birkinbine, including *Incorporating the Commons*²⁶ and “Commons Praxis: Towards a Critical Political Economy of the Digital Commons.” Thanks to Josh Stark, Joshua Dávila (The Blockchain Socialist), Mike Neuder, Tim Beiko, Anna Thieser, Alex Stokes, Davide Crapis, Scott Moore, Cheeky Gorilla, Kevin Owocki, and reviewers Y + C for feedback.

If you prefer to listen, audio versions can be found here: Protocol Berg, “Linux & Ethereum: Commoning vs Commodifying” (Sept. 2023)²⁷; The Blockchain Socialist, “Safeguarding Ethereum’s Soul: Protocol Guild and governing the digital commons” (Nov. 2023)²⁸; Summer of Protocols, “Capital and Enclosure in Software Commons: Linux & Ethereum” (March 2024)²⁹

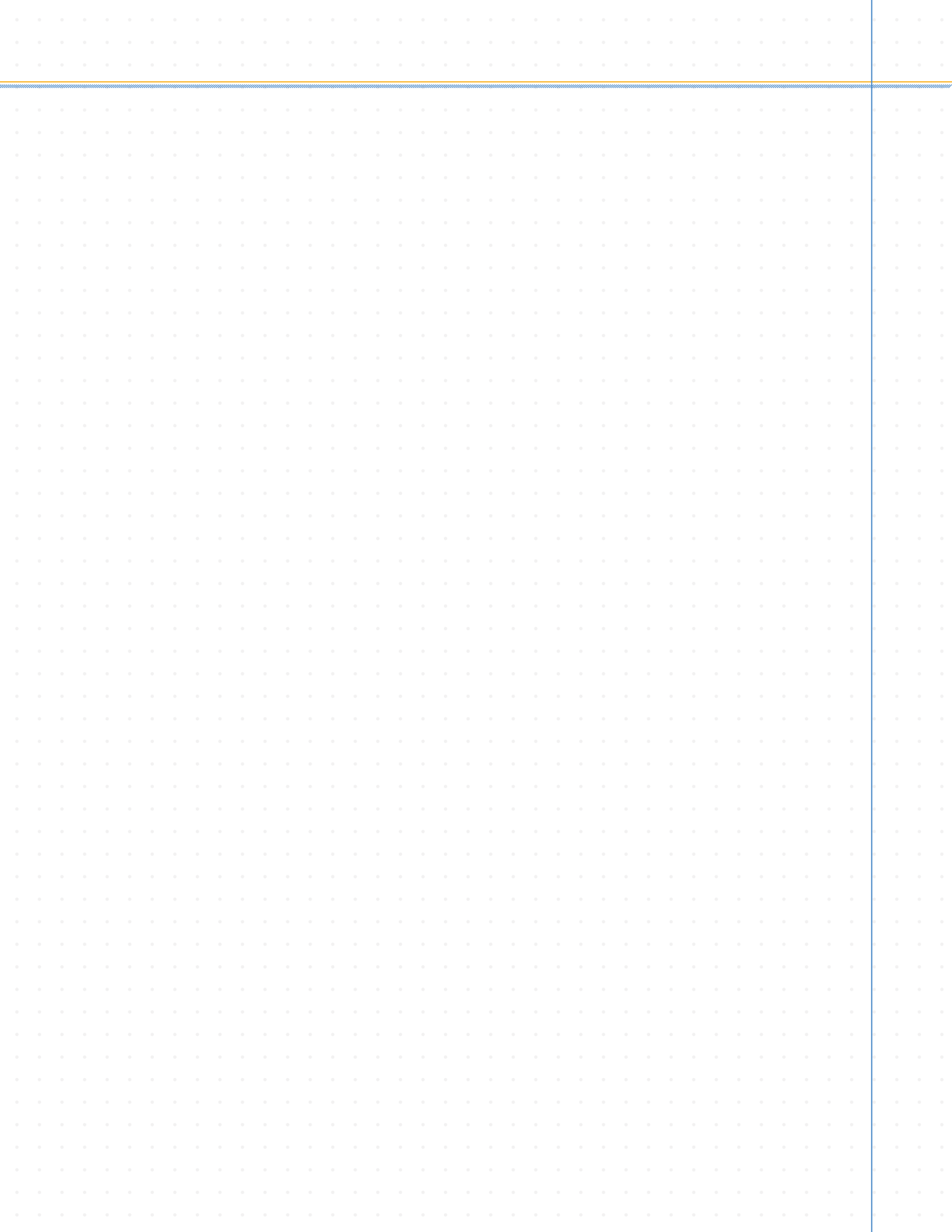
TRENT VAN EPPS is a member of Protocol Guild, Stateful Works, and works for the Ethereum Foundation. trent.mirror.xyz

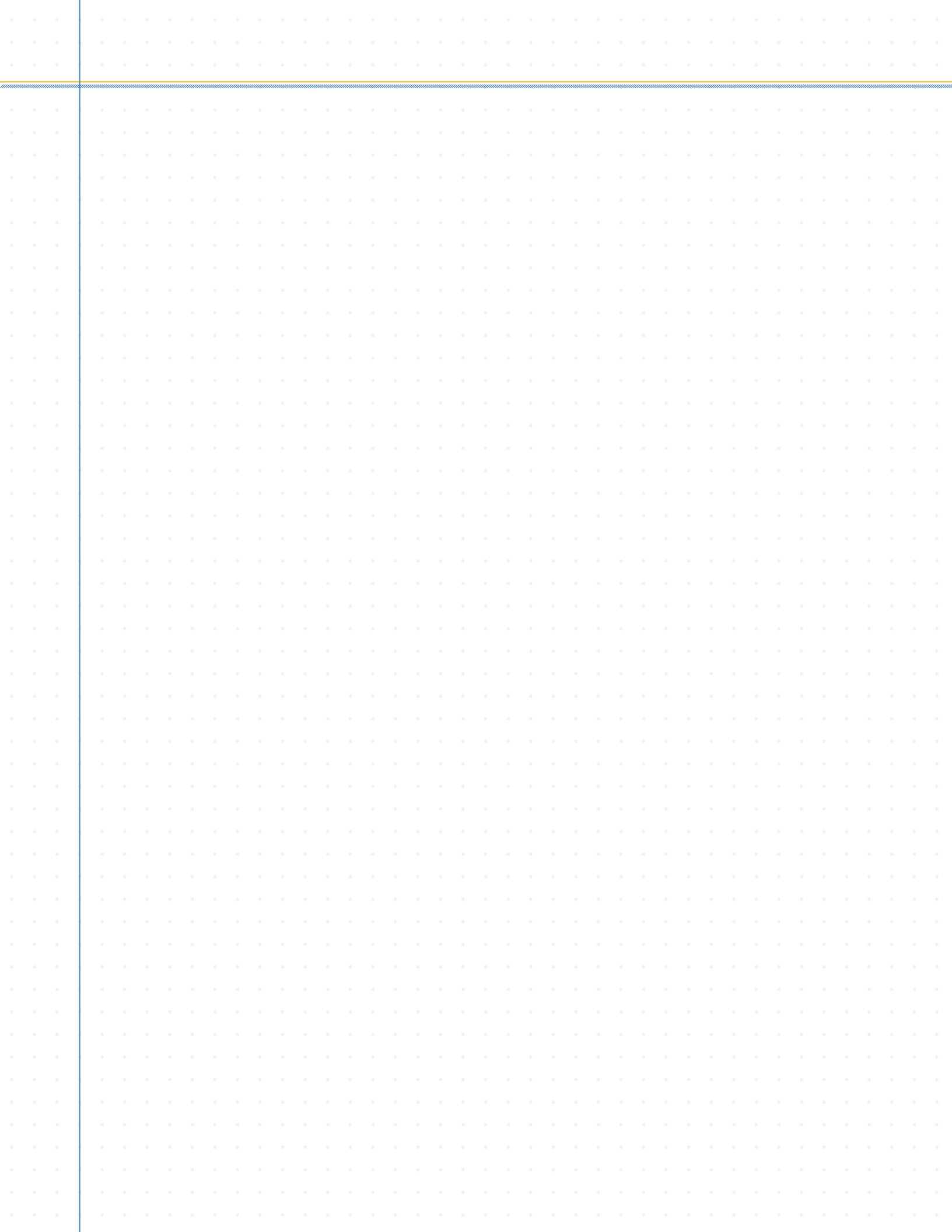
26. doi.org/10.16997/book39

27. www.youtube.com/watch?v=CEwXVUQEwM

28. theblockchainsocialist.com/safeguarding-ethereums-soul-protocol-guild-and-governing-the-digital-commons/

29. www.youtube.com/watch?v=Z9O5_lve10o



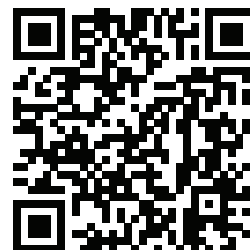


Protocol*Kit*

summerofprotocols.com
hello@summerofprotocols.com



Protocol*Kit*



RETROSPECTUS



NEWSLETTER

