Ceramic Ecosystem Mapping Report

A Look Inside Web3's Decentralized Data Solution

BlockScience

November 2023

Introduction

Ceramic is a stream-based data processing system for peer-2-peer (P2P) network architecture, a part of the broader *Web3* ecosystem of protocols. In Web3, Ceramic *provides on-chain processes with off-chain data storage and processing efficiencies* while remaining within the decentralized, P2P networking paradigm of Web3.

With stream-based data processing, data can be processed in real-time as a sequence of events[1]. One component of stream-based data processing is an event queue. Event queues store related events for subsequent processing. Without such storage, events are ephemeral. Events are written, or published, to a queue in sequence, and the queue maintains that sequence. Subscribers to a queue read events from the queue. The event store provides guarantees to subscribers that deal with event delivery and ordering. With an event queue, events can be recalled after they occur, and events will be recalled in the order in which they were observed to occur. For the purpose of this report, these event-queue operations will be collectively referred to as **streaming**.

Streaming, by itself, **is sufficient** for the purpose of, for instance, publishing a digitally encoded song or movie, and enabling subscribers to listen to the song or watch the movie. In this record/playback use case, the stream is a fragmented and ordered (*i.e.*, event-based) representation of a song or movie. The fragment size can vary (*e.g.*, note-by-note or bar-by-bar for a song, and frame-by-frame or scene-by-scene for a movie) depending on circumstances such as event processing overhead or network throughput. The stream as a whole can be considered *a representation of* the song or the movie, respectively, and the song or movie represented by the stream can be said to be an object. In the record/playback use case, the represented objects are immutable, and the queues of write and read operations, respectively, *are* ***isomorphic***.

Streaming, by itself, **is insufficient** for use cases in which a represented object is mutable, for example, when a composer is in the process of creating a song. Composition can be considered a stream mutations to a song. Parts are created and modified, ordered

and reordered, arranged and rearranged. All of these operations are performed in an order as a stream of *write events*. Each write event is an immutable list of changes, but the song mutates with each write event. Playing back the write events in sequence recreates the song, but it does not playback the song. That still requires note-by-note sequencing. In addition to the represented object being mutable, the queues of write and read operations, respectively, *are* ***non-isomorphic***.

To alleviate the inefficiency of recomposing a song (*i.e.*, playing back a write stream from its origin) just to reach a state from which to perform playback of the song, object storage is coupled to event queuing by the computation necessary to instantiate a storable representation of an object, and to perform event-driven mutation of an object's representation.

For the purpose of this report, these state object store and mutation operations will be collectively referred to as **stream processing**. Stream processing extends beyond simple object mutation. It also performs object enrichment, joining and aggregation. Enrichment here includes combining multiple event streams into one state object. Joining includes combining multiple state objects. Aggregation includes combining multiple states of an object. For the purpose of this report, the combination of streaming and stream processing are referred to as *stream-based data processing*.

Stream-based data processing has been a staple client-server (Web2) data processing architecture for a decade. Centralized, client-server-based protocols such as Kafka and Flink have been providing critical infrastructure for the past decade or so. Kafka, is primarily a *streaming event queue*. Flink is primarily a *stream processor*.

Stream-based data processing is well suited to, for example, time-sensitive data and to enabling real-time analytics. Such abilities provide strong advantages in areas like algorithmic asset trading and operational monitoring, respectively. Client-server (Web2) solutions like Kafka exist for this space. Kafka is similar to Ceramic in that they are permissioned writer, multi-reader streaming services. They differ in that Ceramic is geared for distributed data streams in a P2P environment, rather than being focused on centralized servers.

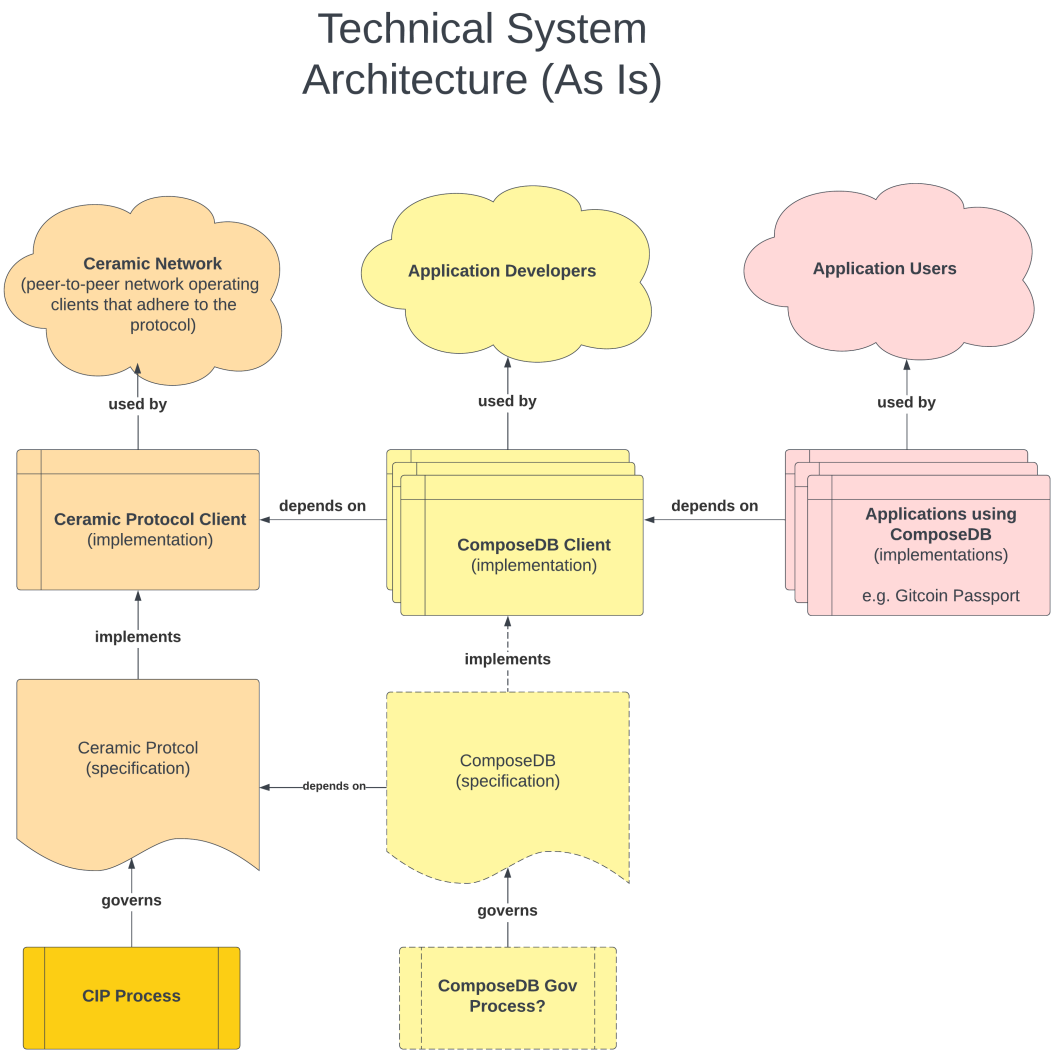
Why we're excited about Ceramic

The current Web3 space has a distinct gap in its ability to provision distributed off-chain data to on-chain apps and this is where Ceramic offers a unique value-add over existing tools. Just as high-throughput, client-server streaming implementations such as Kafka provide efficient event-driven data ingestion functionality, the Ceramic streaming protocol provides an efficient peer-to-peer implementation. With the addition of Flink-like stream processing functionality through ComposeDB, Ceramic brings full-stack, stream-based data processing to peer-to-peer networks. Ceramic eschews the computationally expensive overhead of *maintaining globally consistent state*, as is done by decentralized, distributed ledgers. Instead, Ceramic favors responsiveness and economy by maintain-

ing eventual, *locally consistent state* (*i.e.*, per stream). The consensus mechanisms for individual streams are similarly biased in favor of liveness.

Ceramic Technical Infrastructure

This section aims to lay out the architecture of the Ceramic ecosystem at a high level from a technological perspective.



Base layer – Ceramic Protocol & Streaming API

Ceramic is a protocol for creating and maintaining updatable documents in real time without relying on centralized servers. Ceramic stores immutable records of each update event as a stream of committed changes. Streams are stored in event logs across multiple peers linked through gossipsub mechanism. Gossipsub is libp2p's decentral-

ized implementation of the pub/sub messaging pattern, although Ceramic is currently in the process of migrating to a more robust and scalable replication protocol called Recon, which can be explored further in CIP-124. Each stream is identified by a stream ID, which is a unique address on the Ceramic Network. Each stream has a single owner identified by an owner Ceramic Network account. Ceramic accounts are denoted by Decentralized Identifiers (DIDs). Only the stream owner and those permissioned by the owner can write to a stream.

Event logs are append-only, immutable sequences of messages published to a stream. Each message contains committed changes to a mutable state object represented by the stream. Each commit is cryptographically signed, and the sequence of commits to a Ceramic stream is secured using hash links to form a DAG rooted in the stream's genesis commit. This hash-linked DAG structure enables client nodes subscribing to a stream to locally verify the commit sequence. A subscriber to this stream would be able to recreate the state of this document at any point in its history by replaying the stream's event log up to the desired point. Ceramic guarantees that any two nodes replaying the same event sequence for a stream will arrive at the same document state.

Additionally, each stream is periodically anchored to a blockchain (*e.g.*, Ethereum) for additional security through a proof-of-publication record effectively locking a canonical view of the stream at a time point.

Ceramic is a decentralized protocol. As such, there is no central stream repository with which to define a canonical event sequence. Instead, streams must define a consensus mechanism to resolve conflicting views. Ceramic differentiates itself from distributed ledger protocols such as blockchain by focusing consensus on individual streams rather than global network state.

Ceramic streams are classified by streamtype. Streams of a given streamtype share processing logic that performs state transitions using the data in stream events along with consensus mechanism processing logic. By specifying where a consensus mechanism is rather that what it is, Ceramic allows a stream owner to use a consensus mechanism that is appropriate for the stream.

In general, any consensus mechanism will be a trade-off between liveness, consistency and safety. Each stream will have its own sweet-spot in that trade-off space. Ceramic's architecture provides the flexibility that naturally supports this. It is telling that Ceramic's future roadmap doubles down on this idea of decentralizing consensus by recognizing the applicability of Conflict-Free Replicated Data Types (CRDTs) to future streamtypes, or other alternatives as explored in Ceramic Improvement Proposal (CIP) 145. In this way consensus primitives such as sequencing can be pushed out to the data types in a given event log.

Aggregation and Indexing Layer - ComposeDB

ComposeDB is layered on top of Ceramic's streaming protocol enabling stream processing. ComposeDB views data in Ceramic streams as a queriable[2] property graph. Streams are manipulated using a subset of the GraphQL language. For full details of how to use ComposeDB, see the Ceramic docsite for ComposeDB. In database terminology, ComposeDB's object store is a *materialized view* of underlying Ceramic streams.

ComposeDB changes the Ceramic application developer experience from working with an event stream to working with a database of queriable and mutable documents. ComposeDB also extends the Ceramic stream model by enabling stored objects to be composed on multiple data stream, and it allows the definition of relationships between stored objects defined by models written in GraphQL Schema Definition Language.

The Ceramic streaming protocol is implemented as APIs. Applications interact with the Ceramic Protocol procedurally by calling appropriate API functions. ComposeDB is implemented as GraphQL. Applications interact with ComposeDB declaratively through queries. ComposeDB databases are specified by one or more models compiled into a Composite. *A Composite is a database view, with a particular schema made up of one or more ComposeDB models.* Databases that share models also share the underlying data objects of the shared models. So, even database instantiation and data sharing follow the declarative programming model.

Application Layer

Most developers building on Ceramic are using the declarative GraphQL interface exposed by ComposeDB, and Ceramic is in the process of deprecating direct application developer interaction with its streaming protocol API. GraphQL is the most common database language in web3, but in the future we expect to see additional database emulators built for the Ceramic network. Which database languages get supported in the future will be driven by application developer demand. On top of this infrastructure, many applications for different use cases are possible, some of which are explored below.

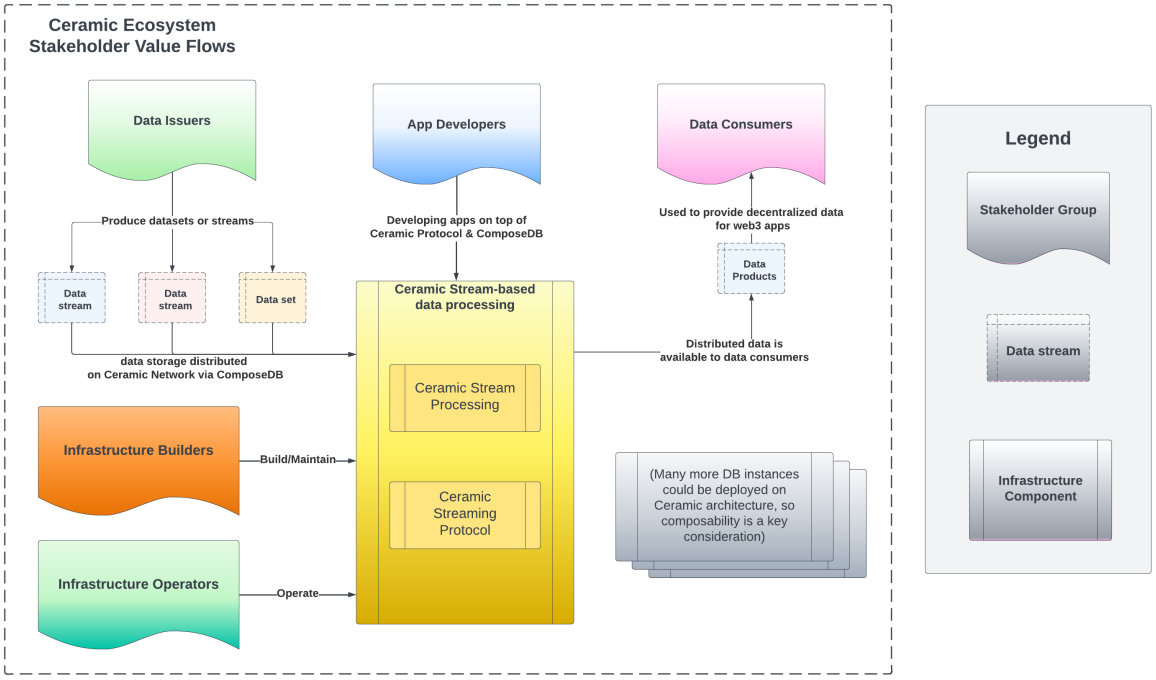
Possible uses of Ceramic Network:

- **Distributed reputation systems**
  - C2B – e.g. (cross)platform reputation
  - C2C – e.g. social network protocols
- **Verifiable data platforms & services**
  - Content – Censorship resistant blog and chat apps
  - VCs – Resilient verifiable credential storage
  - NFTs – Identity as a stateful, mutable object
- **P2P data pools for storage & access**
  - DeSci – Simple sharing of verifiable data sets
    - Data Governance & Access Control are key considerations for data pools (to prevent them from becoming data swamps)

    - Incentivization of data pool provisioning
- **Governance, Finance and Accounting Applications**
  - Enrichment and Annotation of Onchain transaction data
    - Standard Web3 tools such as Safe (formerly Gnosis Safe) lack mechanisms for adding transaction metadata such as contract documents, or simple annotations required to adhere to finance and accounting best practices.
  - Measurement, Reporting and Verification data
    - Standard Web3 tools such as DAO platforms have no instrumentation for storing data about the outcome associated with proposals passed, or comparing those outcomes to intents expressed in proposal documents.

Ceramic Ecosystem Functional Mapping

This section aims examine the Ceramic ecosystem from a functional perspective. Below is a diagram of the various stakeholders in the Ceramic ecosystem, and how they relate to each other and the different functions of the network.



General Stakeholder Roles and Functions Stakeholder mapping is a key part of ecosystem design. Our aim in this phase is to understand the Ceramic system *"as is"*, so that appropriate *"to be"* system designs can be put forward that are aware of the constraints and respectful of the entanglements between different groups with diverse preferences.

Infrastructure Builders: Builders — those who produce and maintain the technical infrastructure — form one of the initial primary stakeholder roles. It is important to distinguish between those who build the infrastructure and those who operate it. Builders are therefore mostly users who develop and maintain systems required by one organizational layer using the the functionality provided by the layer beneath.

As with any system, building it does not suffice on its own; a system needs to be operated continuously to be functional. Additionally, it is unlikely that infrastructure will be built without any demand for it.

Infrastructure Operators: We class operators into two subgroups:

- Those who operate p2p network aspects by providing core services to the network by running nodes, and
- those who operate aspects of the ecosystem on traditional client-server systems, such as explorers for the Ceramic Network.

Both of these subgroups exhibit a common characteristic; namely, they are likely to operate those services only if there is demand from other users, or if they are in need of those services for their own operations. While operators need users to instantiate demand, they also require builders to provide the infrastructure to run. The existence of these operators, and the nodes they run, is clearly required to satisfy any demand for network services by data issuers and consumers. This pattern of *vertical interoperability* and *horizontal integration* repeats throughout Ceramic's organizational stack.

Data Issuers and Data Consumers: Data is valuable to the extent that it is used. Data flows are assets; data at rest is a liability. Data Issuers, by publishing data to streams, and Data Consumers, by subscribing to streams to use the data they contain, define Ceramic's data flows, and therefore they define Ceramic Network's value flows. Data Issuers provide supply by collecting raw data and making it available via Ceramic streams. Data Consumers provide demand by utilizing aggregated data streams in their desired context.

Additional Stakeholder Groups: Aside from direct use of Ceramic infrastructure, there are some additional stakeholder groups who are involved in the creation, funding and steering of the Ceramic ecosystem. These groups will be discussed briefly below, and then situated in a 'layered' view of the Ceramic stack.

Founders: One prominent stakeholder role not previously discussed is that of the founders. 3Box Labs has initially provided much of the development effort, and with it governance over the ecosystem within their group. 3Box Labs strategically and operationally directs much of the effort on both base layer development as well as expansion through ComposeDB, and is working on engagement of new builders towards the distribution of development and progressive decentralization in the Ceramic ecosystem.

Governance Participants: Governance within the ecosystem emanates through the layers above them, inducing the need for particularly stable governance at the lowest layers. The governance we identified currently falls mostly within three areas. - 3Box Labs and other core developers exert influence through allocations of initial funding and strategic as well as operational decision making on core developments, such as a focus on the implementation of ComposeDB. - Other layers of the stack are likely mostly self-governed by entities and their own communities, with little direct influence on the other parts of the stack. - Most prominently, CIPs regulate development of core Ceramic services and are open and accessible to all Ceramic Ecosystem participants.

In the future, governance might be more accurately characterized through meta-layers, where each layer has their own (or even several) localized governance surfaces. As such, the overall surface of governance within the ecosystem might become larger, but more clearly constrained. We see the core functionality of governance within the ecosystem in decisions on requirements against which the code infrastructure and operations are maintained. In the future, governance might also include pre-decision making processes and ongoing maintenance including further ecosystem funding allocations as well as signalling processes for informational alignment.

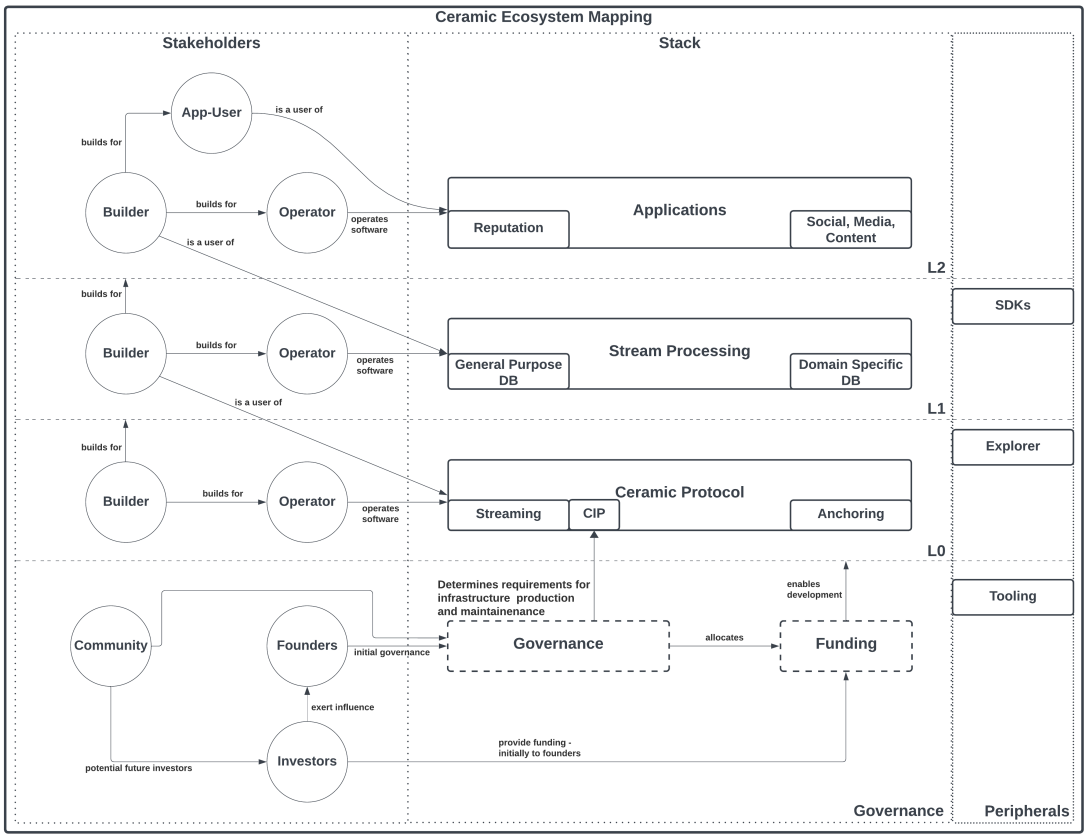
Users: As emphasized before, we see the initial primary users in the Ceramic Ecosystem to be builders. Aside from application end users, who are logically separated from the ecosytem and might not have any knowledge about the Ceramic infrastructure being utilized at all, earlier users need infrastructural layers to be built and operated according to their own needs. They can participate in governance additionally, or become builders for their own needs. One of the core user groups might be found on layer 2 as an example: Those who build new database emulators are using both the tools provided one layer below as well as the infrastructure provided two layers below. They can be considered Ceramic users, while also being database builders.

Disclaimer At this point it is important to denote the difference between a stakeholder role and a real user. It is very likely that many of the real users will fall into multiple stakeholder roles at the same point in time. As an example, a real user might develop a domain-specific database emulator at layer 2, as they need this type of infrastructure for an application they are developing on layer 3. As they have direct interest in the development of layers below, they may also participate in governance. This vertical integration of roles allows users to make their own lives easier and to develop solutions for real needs. They are a Ceramic User and fulfill many functions within the infrastructure simultaneously.

Descriptions of Functional Layers of the Ceramic Ecosystem

From the architectural considerations described above, we can derive a functional layering of the Ceramic Ecosystem. These layers help identify functions that must be fulfilled more granularly and define interactions between stakeholders.

A layered view of the Ceramic Ecosystem consists primarily of **builders** and **operators**. While it might be easy to envision Data issuers and Data consumers as the primary end user group, it might be better to describe early Ceramic Users as the infrastructure builders and operators. Only final end users are expected to provide and consume data at scale, but they do so indirectly, and ideally without needing much of an understanding of the infrastructural layers below.



As can be seen from the diagram above, Ceramic stakeholders build up and operate layers to serve those stakeholders in layers above them. For example, one can imagine builders of the Stream Processing layer to be users of the Ceramic protocol base layer. Without stakeholders building and operating the base layer, developers of aggregation and indexing solutions have no feasible infrastructure to build upon. However, they themselves build infrastructure, which is again used by stakeholders one layer above.

It is important to stress that those layers are only being built if there is demand for usage from above. Additionally, each layer requires operators - from node operators to service providers (as in the case of explorers for example). The layering therefore provides two essential insights:

- Builders of one layer are users of the layer below. They build infrastructure and

services for those on the same layer, as well as those in the layers above.
- Operators are needed to maintain the infrastructure under demand. And demand from layers above is needed for anyone to desire operating or building the infrastructure.

Layered architectures allow for separation of concerns across layers. The layers, then, need to also be separate in order for concerns to remain separated. To highlight this requirement, we say that the infrastructural layers of Ceramic's Ecosystem are vertically interoperable, or 'infra-operable'. By this we mean that each layer makes use of another infrastructural layer by communicating with it, as opposed to subsuming it or replicating it in a centralized fashion.

Aside from builders and operators - which exist on every layer - there are also participants to the governance and funding layer. While these are currently a small group, future developments might increase the surface area of this group extensively.

Governance and Funding Layer

What happens in this layer The Governance and Funding layer combine two connected functions. On one hand, governance within the Ceramic Ecosystem is currently assumed to happen mostly through CIPs. Similar to other software development, code changes are governed through proposals made and voted on by the community. Such changes have effects on all layers above them, even though they mostly concern the Ceramic Base layer. Anyone in the ecosystem can participate in the CIP process and can therefore fulfill the function of this layer, to steer the direction of upgrades.

Funding, on the other hand, serves to enable the development of governance decisions. Funding decisions are made according to requirements of the ecosystem, but do not necessarily correlate directly to decisions made through the CIP process. Especially on higher layers, funding likely comes directly for specific solutions, resulting either from an identified demand for services or from a need for infrastructure to achieve higher vertical interoperability.

Which concerns made them a separate layer We separate out the governance and funding layer, as they are likely to emerge as meta-layers later. While governance currently resides partially within 3Box Labs and partially within the CIP process, future iterations might expand on that notion of governance more broadly.

One future version of the stack might have localized governance consisting of a clear and concise scope, attaining higher legibility for participants. Localization enables also a clearer view on the expanding governance surface of a growing system. A future Ceramic Ecosystem might consider governance localized per layer, through processes similar to those existing in current CIPs. Additionally, future governance might expand the view by including further informational provisioning, governance of non-code

aspects, administration of key functionality, funding disbursement and development coordination and more.

On the funding side, we consider most of the current funds disbursement decisions to be fairly contained within the 3Box Labs team. An initial CapEx oriented funding layer might in the future adapt to more OpEx oriented funding, while direct funds might come from ecosystem members - especially those with a desire to achieve higher vertical interoperability. Such developments can steer the system towards higher internal sustainability and long-term development.

Connections to layers above and below The governance and funding layer form a particular instance, as they could be considered as a meta-layer that inform and steer all layers above - implicitly or explicitly. Particular care must be taken when designing governance mechanisms that exhibit effects on a protocol stack with such clear dependencies emanating downwards. Additionally, stakeholders from layers above are expected to take part in this layer too, becoming a user of the layer below. It is currently unclear how much interaction and constraints are imposed by funding on governance, or vice versa.

L1 - Ceramic Protocol Base Layer

What happens in this layer The base layer of Ceramic handles core services necessary for layers further up. The core data streaming functionality, anchoring of stream states, handling of DiDs, networking capabilities and more are conducted in this layer. Users of any layer above are depending upon the base layer. Additionally, at this layer the CIP mechanism is handled. CIPs define core changes to the base layer of the Ceramic protocol. Any change to this layer has direct implications for all builders and operators in layers above. Builders at this layer also provide core functionality to both Operators on the same level - node operators running core Ceramic services, including IPFS and sometimes anchoring functionality - and builders and operators on layers above.

Which concerns made them a separate layer Separating the base layer from some functionality above - say the ComposeDB stack, which often incorporates core base layer functionality into one implementation - allows a clear definition of boundaries between layers and the effects emanating from one to another. Any changes made to this layer are influencing all others, likely requiring a higher degree of stability of the protocol design. Additionally, the majority of current community governance flows into this layer.

Connections to layers above and below In describing users of Ceramic, we have mentioned before that our view lies first on developer users, then on application users.

While applications might use the Ceramic Protocol stack as infrastructure, the application user would likely not have much interest in specifics. On the other hand, our consideration for Ceramic users is mostly on those building and operating the layers above. A Ceramic Protocol builder provides infrastructure for various layers, including operators on the same layer. Node operators are similarly likely to affect a large part of the user base, as their services are required for all other layers to function properly.

L2 - Stream Processing

What happens in this layer The Stream Processing layer allows for users to change from working directly with event streams to working with a database fit for their purpose. Initially, the main implementation consists of ComposeDB, which also bundles together services from layers below. This layer likely consists of various aggregation and indexing services, which enable application builders above to integrate the Ceramic stack into a functional and practical experience.

While ComposeDB allows users above to emulate a GraphQL database, it is likely that not only different general purpose DBs are built in the future, but also domain-specific DBs for specific applications (such as a MusicDB, as an example).

Which concerns made them a separate layer While a user might receive a bundling of layers through current implementations, it seems functionally more appropriate to separate out the Ceramic base layer, which allows for event streaming, network services and more, from the aggregation layer, where events become objects and users can interact with a database similar to more standard development experiences. It is clear that builders on this level require some development tooling provided by the builders below, as well as demand for their solutions from the layer above.

Connections to layers above and below At layer 2, builders provide core functionality for the users above - application developers. Additionally, they require both development work and infrastructure operations from layers below. Additionally, builders on this layer are highly likely to respond strongly to demand from above. While vertical interoperability of builders is possible, especially for domain specific database implementations, there is also a wide design space for incorporating database types known from web2 that can be implemented if demand enables it. Applications using certain implementations can drive high demand and provide core incentives to builders on this layer.

Operators on this layer are likely to be bundled with certain core services from layer 1. As an example, when running the current ComposeDB implementation, a node operator may additionally run core Ceramic node services. This might change with future implementations of other database emulators and would need to be addressed in any governance and economic design work, since it might change the dynamics from

an opt-out of running core network services to an opt-in, likely to reduce the share of layer 0 operators.

The builders and operators on this layer are once again users of the layer below. They require tooling, in particular likely SDKs, as well as a functioning and stable core protocol. While governance is currently focussed through CIPs more widely, it is likely that this layer - or well possible only parts of this layer - develop their own localized and purpose-built governance to address any changes in a more direct and local governance process, allowing for higher legibility and operational capacities.

Applications Layer

What happens in the layer The application layer builds those aspects that are visible to the end-user. They integrate aggregating and indexing implementations from layer 2 to provide functional applications that consume and produce data. This layer is likely the main driver for demand for any layer below and can drive various governance and funding decisions rippling through the layers. Currently, there is a strong focus on Reputation applications, with various other application types having integrated already, yet still leaving a wide design space open.

Which concerns made them a separate layer It seems clear that the application layer deserves its own place in the stack. Serving as the only interface to end-users, this layer influences all others through demand propagating downwards. With this particular factor, it is necessary for this layer that those below exhibit some stability. As the end-user should likely not know much about which services are used for their experience, any effects rippling upwards from governance or operational events are likely to feed back into demand considerations, which effect the layers below, and so on. The application layer also clearly shows dependencies on the layers below, especially in layer 2 where application development is enabled and directly experienced. A well functioning stream processing layer, combined with a well functioning tooling layer are likely to enable any conversions from potential demand to used implementation.

Connections to layers above and below For the application layer, builders are engaged in developing applications on top of the layers below, to achieve a functional end-user experience. Similar to web2 applications, it is likely that end-users have considerably limited interest in the same expansive details as database enthusiasts. This layer therefore also functions as a separator between infrastructure and end-users, enabling them to enjoy the benefits without having to constantly worry about their safety. Any disruption in service in any of the layers below can have direct effects on this layer. Builders are potentially also implementing stream processing capabilities on a layer below, but would generally fall into the camp of users of those services. Similarly, operators are using services provided from below - while similar to the peripheral layer it is likely that they are contained within the same entities as the builders.

Peripheral Layer - Ceramic Tooling

What happens in this layer The peripheral layer was added after feedback in discussions with the 3Box Labs team. Initially unidentified by BlockScience, they serve all layers of the stack by developing user tooling. We now consider this a meta-layer, that spans the entire stack. The tools built here can range widely, from explorers to SDKs and beyond. As before, the tooling layer depends upon development from lower layers. Similarly, operators are required to maintain a functional ecosystem. Unlike before, where operators were mainly considered to be node operators, we similarly class client-server infrastructure operators here.

Which concerns made them a separate layer The peripherals layer forms an integral part of the future development of the Ceramic Ecosystem. Tooling is likely to be partially built due to demand from specific layers, but also creating new possibilities in itself. This layer requires a functional base layer, but also enables a wider design spectrum for any development and operation above.

Connections to layers above and below We once again consider builders and operators of the peripherals layer to be users of the layer below the one they build for. Builders here are providing core tooling for the rest of the stack. It is likely that operators of a tool are the same entity as the builders of a tool, but this is not necessarily a given. Together, they enable users in layers above to enjoy a better development experience. Additionally, this layer will likely be building out additional governance capabilities in the future event of widening governance considerations. Informational capabilities and the tools to effectively conduct wider governance operations fall squarely within community tooling, but might not always be considered as such. For any governance tooling purposes, it is important to highlight that the final voting is only one aspect, but the process of enabling participants to find and verify the necessary information requires often purpose-built and localized tooling.

Environmental Considerations

Beyond stakeholders directly represented and acting within the Ceramic Ecosystem, there are those who effect it without regular direct activity.

Regulatory Space We consider regulatory constraints as environmental factors to activities within all layers of the ecosystem. These define various constraints in which the ecosystem can safely evolve and can exert outside influence on all functions, from restrictions on governance and funding, to design path limitations due to, for example, regulatory constraints on data protection and data security. One specific regulatory concern is whether applications built on Ceramic streams can be compliant with the "right to be forgotten" that is explicit in some privacy related regulations such as the European Union's General Data Protection Regulation (GDPR) Article 17. Ceramic

currently lacks a mechanism to delete data from a stream. Ceramic's lack of a data deletion mechanism leaves compliance with Article 17 entirely up to application developers who must work around this. Community conversations on the topic can be found on the Ceramic Forum.

Web3 Space As web3 data infrastructure, the Ceramic Network is partially nested within the wider web3 space, where most demand and supply of data is expected to come from. Additionally, services within the ecosystem (such as anchoring, but also potential future services such as monetary compensation and further governance) are directly reliant on other web3 infrastructure and are expected to work in tandem. Specifically, subsets of the web3 space - such as reputation systems - are expected to strongly drive the conversion from environmental factors to ecosystem members.

Within the web3 space one can also find the most likely competitors, as traditional solutions, such as Kafka, will likely serve different purposes.

- Potential Competition / Collaborators
  - Source.network
  - Streamr.network

Web2 Space Web2 Solutions drive the frame of reference for this solution space in the context of traditional markets. These tools are likely to be more focused on high-throughput, performance focused tooling, e.g. Kafka or Flink.

Endnotes

[1]In the context of streams of events, an event consists of a the following: * a reflexive identifier — a means to refer to the event such a name or an ID * frequently, a sequence identifier — a means to determine the event's ordinal position in the event stream such as a time stamp. * optionally, a payload of additional data

[2]For the purpose of this report the term "query" refers to both Data Definition Language (DDL) statements and Data Manipulation Language (DML) statements. In Ceramic documentation, to query more commonly refers to DML statements, and more specifically to read as opposed to write statements (Data Query Language or DQL).

Disclosure

BlockScience is an early supporter and investor in 3Box Labs, creators of the Ceramic protocol.