Arctic Archives: The slough politics of GitHub's Code Vault

Abstract

In 2020, GitHub printed 21 terabytes of data scraped from users' repositories onto a series of film reels to store in a decommissioned mine shaft in the Arctic. The Code Vault calls itself an archiving project that aims to "protect the world's software from apocalypse." At the center of this project is their Pace Layer Strategy, a model inspired by Stewart Brand's 1994 book *How Buildings Learn.* By revisiting these texts, I trace how architectural notions of 'timelessness' came to form the foundation of a software preservation project that is deeply situated within the broader history of Silicon Valley's historical techno-configuration of time. Drawing on science and technology studies and platform studies, I argue that platform uncertainty is increasingly met with an emphasis on durability and permanence through innovations in storage. I introduce the *slough politics of platform archives* to describe how Github transformed ongoing relational forms of software labour into inert archival objects by naturalizing processes of loss. In doing so, the Arctic Code Vault recasts platform power as a question of material survival, obscuring the ongoing social labor and maintenance practices that sustain software ecosystems.

Introduction

Located up the hill from the World Seed Bank - a biopolitical experiment in Svalbard, Norway that keeps seeds on a "cusp without end" (Nadim, 2018) - is the GitHub Arctic Code Vault. From the outside, the building looks like a shipping container positioned on the edge of a steep hill, one of several overlooking Isfjorden in the Norwegian archipelago of Svalbard. Inside the decommission coal mine, down a darkened railway filled with loose rocks and debris (remnants of an industry that has only recently ceased in the most northern town in the world), is a steel container with reels of film storing 21 terabytes of data scraped from GitHub users' repositories.

This project was launched in 2019 by then-GitHub CEO Nat Friedman who wanted to "protect the world's software from apocalypse," and preserve a "modern way of life" (Vance, 2019). While these two goals seem to reflect concerns about environmental, political and economic collapse in the midst of global poly-crisis, the project itself comes at an interesting time. GitHub had just been acquired by Microsoft, and Nat Friedman was about to step down as CEO. In light of these contextualizing events, the "apocalypse" and "way of life" desperately needing preservation may have reflected the ruptures in leadership and ownership the platform was about to face, rather than any direct threats to software's very existence.

But this project presents an interesting case: why is software being taken offline, printed onto film reels, and placed in the northernmost town in the world? GitHub joins several other Silicon Valley tech companies who are experimenting with data storage technologies, signaling a turn in platform temporalities towards long-term data preservation strategies that move beyond relying on data centers and the cloud.[1] Platforms have previously been

[1] For example, Microsoft Cambridge developed Project Silica, a glass storage technology for archived data (see more: https://www.nature.com/articles/s41586-025-10042-w) and Florida-based company Lonestar Data

described as “infrastructures on fire” (Edwards, 2022) an unstable socio-material technology that has become increasingly ubiquitous across all sectors of public and private life over the past three decades. Platforms also notoriously die, even when they hold and share software and code repositories and even while they are brought into software preservation processes. Their futures have always been uncertain.

This uncertainty is met with reconfiguration of platform power. The guiding framework for this preservation project is called the Pace Layer Strategy, a model inspired by Stewart Brand’s 1994 book *How Buildings Learn.* Stewart Brand, an influential Silicon-Valley figure and futurist, became deeply inspired by the concept of ‘shearing layers’ from architecture studies following the work of Christopher Alexander’s *A Timeless Way of Building (1979)* and Frank Duffy’s *Measuring Building Performance* (1990). By revisiting these texts, I describe the genealogies of thought and processes of conceptual translation that begin with theories of change rates in buildings, later adapted in software, and now deployed in the Code Vault project.

In this transference, the architectural theories of materiality and timelessness are inverted, and the role of maintenance disappears from the technological imaginary and is replaced with notions of innate durability and persistence. By focusing on GitHub’s Pace Layers, I argue that rather than merely mapping the preservation strategies in place, this model creates a hierarchy of obsolescence where the platform anticipates and naturalizes processes of loss over time: shedding the users, the platform, the code, and archive, until all that’s left is an inscription.

This model demonstrates their approach to platform preservation, which engages less with archival practices, theories and methods and is more embedded in Silicon Valley notions of temporality and longtermism. By tracing the epistemological roots of the Code Vault and situating it within architectural notions of ‘timelessness’, I demonstrate how GitHub recasts platform legacy as a matter of natural hierarchy and materiality rather than the social labour and maintenance practices that sustain it, ultimately becoming a *slough archive*: an inscription of software legacy that persists through losses.

What if GitHub died?

GitHub has emerged as a shorthand for code storage, with their free personal accounts and collaboration possibilities, many developers rely on GitHub as part of their broader workflow. As a platform, it has been theorized as both a “hybrid and hybridizing” space, where F/OSS (free and open source) and proprietary software boundaries collapse (Otto et al, 2023), and as a “software intermediary,” (Bounegrou, 2023) where the platform operates simultaneously as a developer platform, community hub, and storage container. Mackenzie (2018) has argued that GitHub is distinct from other code sharing platforms in that it configures coding as a ‘social networked practice’, where code repositories are adorned with social media-style apparatus of following, watching, liking, and tagging. GitHub has publicly attributed its

---

Holdings sent data to the moon on March 6, 2025 (see more: https://www.technologyreview.com/2025/03/03/1112758/should-we-be-moving-data-centers-to-space/).

success as a platform to these affordances, often praising its active, dedicated user base of developers who continuously monitor, update, and maintain the code.

In the history of software developer platforms, GitHub became synonymous with software development despite the many competitors in the market and the popularity of other platforms over the past 20 years. In 2018, GitHub was purchased by Microsoft for $7.5 billion USD. This led to concerns from users who looked to GitHub's competitors to move their code repositories, such as Bitbucket (owned by Atlassian), GitLab and SourceForge (owned by Slashdot), who reported spikes in new users intending to migrate projects from GitHub to their respective services.

Microsoft's acquisition revealed an ongoing and historical tension between the commercial or technological promise of material durability and persistence versus the consumer and cultural experience of dysfunction and obsolescence that all technologies embody. Previous work on "platform death" has demonstrated how the entanglement of social, political, economic, and technical factors drive platforms to points of failure, producing not a single moment of death but one of compounding endings (Corry, 2022; McCammon and Lingel, 2022; MacKinnon, 2022). The uncertainty and untimeliness of platform death produce anxieties for platform owners and users around enduring media (inter-)operability, durability and permanence.

The Arctic Archive, otherwise known as the Code Vault, contains a selection of GitHub repositories that were taken offline from the platform in 2020 and printed on microfilm reel. These reels were purpose-designed as a cold-storage archival medium that works through customized polyester boxes and located inside a decommissioned mineshaft. The code was selected through an algorithm scraping the platform's "greatest hits," which resulted in just over 17,000 GitHub repositories. GitHub hosts over 420 million repositories, uploaded by individual users and companies alike, with reportedly over 100 million developers and 28 million public repositories. The selection process for the Code Vault remains opaque, as the algorithm used to select repositories is not disclosed, and the selection criteria remains vague. Upon inspection into the published list of repositories included, it appears to be selected through popularity scores, although not exclusively.

This article first positions the GitHub Code Vault within the wider context of software preservation practices, before describing the *slough politics of platform archives* that the project is deeply inspired by and embedded in. The empirical material for this article comes from GitHub's own documentation, collected from the Archive Program Repo [https://github.com/github/archive-program], and Arctic Vault website [https://archiveprogram.github.com/arctic-vault/], including historical versions of these pages accessed through the Internet Archive Wayback Machine. In these documents GitHub has published details on their process and approach, including their Guide, Pace Layer Strategy, Tech Tree, promotional videos, partnerships, and a list of all 17,000 repositories included [https://archiveprogram.github.com/assets/img/archive-repos.txt]. Building an analysis from these empirical materials, I argue that GitHub's Code Vault demonstrates a strategy based in Silicon Valley technopower, where platform companies, wary of their own mortality, are

experimenting with materiality and intentional loss to develop processes of platform permanence.

Literature Review

Software Forever

Software has been instrumental in scientific and technological domains since the 1960s, for practices of observation, data collection, measurement, validation and modelling. As Acker (2021) has described, software's centrality is due to "its development, standardization and use, maintenance, uptake and transmission, as well as its influence as an epistemic tool for knowing the world." While its instrumentality and significance are often praised, less obvious is the work of maintaining software conducted by "invisible technicians" who are only elucidated when examining who cares for and manages knowledge and information systems, and data archives. As Cohn (2019) describes, "code is lively and changing," and known for having a "promiscuous evolvability" that it mutable and dynamic. In STS studies, there has been significant research on the "highly imbricated social relationality of software-in-the-making" (Cohn, 2019; Mackenzie, 2006; Gillespie, 2014; Bucher, 2012). For decades, software has been understood as deeply entangled with "forms of duration, the entangled lifetimes of careers, professional identities, and programming languages and paradigms, all of which come and go" (Cohn, 2019).

Digital preservation of software and software-dependent objects have been underway across many organizations for several decades. There are, of course, many approaches to preserving software and dynamic technical artifacts that are in a near-constant state of dying. The labour of keeping software from obsolescence is often undertaken by institutions like libraries, archives, and museums, but also by corporate entities and grassroots preservationists who organize online, through open-access repositories like the Internet Archive's Software Library. Within these various organizations, there are shared standards and practices that inform approaches to working with difficult, often proprietary, material that depends on a broader ecosystem to function.

Until recently, preservation focused on migrating data into newer formats to ensure continued access and functionality. Format migration remains the most common method of digital preservation of digital-born media. However, this type of preservation can fundamentally transform the structure, experience and evidentiary nature of media. As Acker has described, "preservationists have turned to emulation approaches to address the challenges of accessing information whether mirroring an older computer, software, software dependent objects, in the service of preserving the experience of using software." Even within these archival organizations and institutions, there are disputes, disagreements and losses that occur through the ongoing assessment and preservation of software.

The Code Vault takes a different approach to software preservation, where software is rendered inoperable as an analog medium that must be re-digitized in the future. This logic contradicts traditional software preservation, where operability and access are two of the key foundations of their archival activities. It calls into question whether GitHub's Code Vault can

be considered an archive at all, or whether archival practices run through their project's design. Hogan and Roberts (2023), in examining other archival projects from big tech companies, like Alphabet's "The Selfish Ledger," Big Tech's "Genomics in the Cloud," and Arch Mission's launch of a "Solar, Earth, Lunar, and Mars Library," have called their approaches a "distortion of archival values." We can see a similar distortion taking place with this project if we consider how much it diverges from long histories of software preservation strategies.

Deep Code

Storage has become a crucial matter of concern in digital societies. Across administrative, institutional and industrial environments, questions of when and how to store data become increasingly complicated through, on the one hand, emerging standards and expectations around data protection, security and responsibility, and on the other hand, increasingly dynamic complexities of data accumulation, use, access and production. Across geography, political ecology and environmental studies, the politics of storage have been well traversed. From storing and distributing capitalist value of data to storage's worldmaking capacities, to the degenerative and obfuscatory material realities, to its embeddedness in cryogenic architectures: it is clear the many ways in which storage is a political, temporal and spatial technology (Kowal and Radin, 2017). Digital data raises its own distinct challenges in material terms, where storage is often conflated with memory, and the labour and energy required to prevent loss is intensive. Storage studies demonstrate a long history of uncertainty, where storage both contains and compromises, conceals and reveals its contents (Horst, et al 2021; White-Nockelby, 2022; Klik, 2026)

The Code Vault project experiments with storing data in *deep time* where the temporal horizons exceed 10,000 years into the future. Storage in deep time is otherwise thought of as within the scope of earthly timescales: fossils, stone, mountains, genetic strands, or nuclear waste disposal. But archives and data centers imagine long-term storage usually through hot and cold terminologies to refer to frequency of access; deep time data storage, in this instance, refers to data that is sloughed and rendered analog. In other scientific contexts, frozen temperatures are necessary for blood banks, seed banks, and endangered species samples to enable longer temporalities. This type of storage is energy intensive, requiring specific temperatures and freezing infrastructures to be sustained. These activities are part of what has been called a cryogenic culture, where samples must be sliced, reformatted and made to fit into the narrow box for long-term preservation (Braun, 2024; Hopman, 2024). The Code Vault, instead, uses a mix of digital and analog storage technologies to produce a new type of storage for software infrastructure. Through printing code onto long-term proprietary archival film format, GitHub's code vault renders over 17,000 archived repositories inaccessible and unusable, unless processed through specific machines to read and re-assemble the code.

To create the conditions in which the data might be preserved for up to 10,000 years, the open-source software code from GitHub underwent a series of transformations. The repositories were algorithmically selected by internal popularity scores, then automatically

crawled and scraped before sent to Piql. Piql Film is a long-term preservation format created by a Norwegian company, whose CEO is also the head of the Arctic World Archive (which houses the Github Code Vault, among other cultural heritage data). 21TB of data was copied onto 186 reels of 35mm polyester film coated with a gelatin emulsion containing microscopically small light-sensitive silver halide crystals. This is commonly known as silver halide film, a popular photography and microfilm medium, but is here referred to as PiqlFilm.

Piql uses this material to transfer binary data using photons in frames along the film using their piqlWriter. A photochemical processor, known as the piqlProcessor, is a machine where the information written on the film is chemically developed and fixed to ensure image presence and permanence. Data is thereafter read through a piqlReader, which reads frames from the film and converts them back into sampled images which are then decoded back into digital data. GitHub-hosted repositories are materially transformed for the Code Vault challenge through a process that builds on archival practices yet ultimately renders datasets fixed in space and time, while making processes that are relient on Piql products - inscribing these machines and formats into the future.

While other Piql projects provide an interface through which their registered partners can make modifications or pull requests to their data deposits, the Code Vault provides no access at all to their users, meaning data cannot be seen, assessed, modified, or destroyed once included. GitHub has publicly stated that they will revisit and evaluate the project every 5 years. This is common with cold-storage archives (Radin and Kowal, 2017), where “cryo-objects are thus available and unavailable at the same time” (Braun, 2024). However, the incremental visits to the code are not to make any modifications to the repositories that were selected in 2020, or to provide any significant maintenance to the archive, but rather to ensure that the vault endures through continued replication and distribution to colder layers. In 2025, five years after the initial algorithmic scrape, a copy of the vault will be proliferated across Microsoft’s Silica archival technology that inscribes data into glass platters.

Therefore, this case presents a slight a shift from the many ways storage has been theorized as a “buffer” (Braun, 2025) engaged in “chronopolitics of stalling,” (Zee, 2017), facilitating the perpetual production of capitalist value (Banoub & Martin, 2020; Breithoff and Harrison 2020; Thylstrup, Archer, Steiner, 2024). Instead, this paper explores storage through the politics of sloughing, where platforms live forever through processes of intentional loss, not only through the material requirements of deep time storage, but also through the pace layer strategy that lies at the heart of the Code Vault.

Silicon Valley’s Deep Time

As a platform project, GitHub’s Code Vault can also be situated within a longer history of Silicon Valley’s investments in the techno-configuration of social time, alongside calendars (Wajcman, 2018), data centers (Velkova and Plantin, 2023), or clocks (Kemper, 2024). Over the past several decades, Silicon Valley has exerted ‘temporal supremacy’ and ‘paternalism’ (Cheney-Lippold, 2025) through technological development, ‘in which one cohort of people live in a future that everyone else is about to enter’ (Chakrabarty, 2007; Au, 2024). Following Cheney-Lippold (2025), who takes seriously the temporal activities and rhetoric of Silicon

Valley engineers and executives, we can see how this archiving project is another instantiation of ideological power expressed through notions of 21st-century futurism. Cheney-Lippold argues that Silicon Valley industries exist in an already-actualized futurity, 'a time prior to, and thus outside of, the present'. Through this temporal move, 'uncertainty is evacuated, bypassing the possibility of critique' and creating conditions where technological development stands unfettered.

In the Code Vault's techno-configuration of social time, the temporal horizon of software extends beyond the end of human existence. Alongside the code is a 'Tech Tree,' inspired by the Long Now Foundation's Manual for Civilization – a list of 3,500 books of 'essential knowledge' to 'restart humanity'. The Tech Tree 'gives an overview of the archive, describes the structure of the archive, how to use it and extract projects, directories, files, and data formats, and how to use dependencies. It is essentially a quick start into software development and computing bundled with a user guide for the archive.' It includes texts across thirteen sections, including 'Fundamentals of computing and the Internet', 'Algorithms and data structures,' and 'Fiction, culture, and history.' The short list of texts that 'convey, on a human level, the history and culture of *our time'* (emphasis added) include *Crime and Punishment* by Fyodor Dostoevsky, *Cyberiad* by Stanislaw Lem and *Anathem* by Neal Stephenson.[2] The user guide has been translated into four languages: Arabic, Chinese, Hindi, and Spanish, and the GitHub community has further translated it into languages like Farsi, French, Greek, German, Indonesian, Italian, Malayalam, Polish, Portuguese, Punjabi, Russian, Tagalog, Tamil, Turkish, and Urdu. It is also designed to be "machine-readable". Across their documentation, it becomes clear that the Code Vault project presents GitHub's investment in creating their own ontologies for the future of knowing software, even extending beyond their domain and towards writing histories – and futures – of 'our time.'

The Code Vault also enables what Taylor (2021) calls, "deep time data storage," where data stored in deep time, through subterranean bunkers, outer-space or the arctic, permit a shift from business continuity toward *cultural continuity*, where platforms can position themselves as "deep-time media infrastructures" (Mattern 2015; Zielinski 2006) enabling the long-term storage and preservation of digital cultural heritage. A move towards cultural heritage, however, might have less to do with an awakening to the cultural value of code, and more to do with the benefits of giving cultural artefacts long-term designations.

This type of deep time thinking, also known as a longtermist perspective, is what led to the *Clock of the Long Now,* a 10,000-year clock first conceptualized by computer scientist Danny Hillis in the late 1980s. The purpose was to create a "charismatic object" that helps people to think long-term, to reverse the effects of "instant-gratification" and instead embody deep time in a way that everyone can grasp. The clock "ticks once a year, bongs once a century, and the cuckoo comes out every millennium," displaying a temporal order that anticipates many millennia to come. While the project was conceptualized with members of the *Long Now Foundation,* including Stewart Brand and Brian Eno, the first full prototype of the clock was

[2] To see the complete Tech Tree, visit: https://github.com/github/archive-program/blob/master/TheTechTree.md.

funded by Jeff Bezos for $42 million USD and was built on his property in Sierra Diablo mountains in Texas, USA.

In a description of the project, Stewart Brand wrote, “instead of discounting time, we can embrace and *exploit* time’s depth” [emphasis added]. Time’s depth is presented as a resource for extraction. This echoes Achille Mbembe’s (2024) critique of imperial time, where he writes, “all power indeed dreams, if not to make itself time, at least to annex it and colonize its intrinsic properties.” To annex and colonize the intrinsic properties of time is to have influence over pace, structuring the experience of its passing, and the endurance of infrastructure and materials that sustain certain timescapes (Adam, 1998; Sharma, 2014; Velkova and Plantin, 2023). To extend the timescape of a platform like GitHub is an act of platform power, generated not through network effects or concentration, but through the creation of a monument designed to persist through time.

In the case of the Long Now project, The Clock itself is hard to access. After reaching the top of the mountain, there is an entrance that leads you to a long underground tunnel, where you’ll pass through a series of airlock gates before getting to a several hundred foot spiral staircase. The stone of each step was carved by a robot created for this purpose. Over the course of two years, the robot grinded one step at a time. The clock is suspended down the spiral staircase, and designed to keep time without being wound. Instead, this clock captures energy from the sun, allowing the pendulum to tick “without human intervention to 500 years.” Between the robot stair grinder and the self-sustaining clock, it becomes clear that this project and others like it are invested in developing technologies that replace the human work of engineering.

Stewart Brand compared the clock to other “technological artifacts, such as fragments of pots and baskets,” that have survived 10,000, suggesting that there is precedent for human artifacts surviving a very long time, however, “very few human artifacts have been continuously tended for more than a few centuries.” The issue at the center of these projects is not knowing if these technical artifacts will “receive continued care and maintenance for such a long time.” In other words, the core obstacle to technological endurance is the reliance on human care and maintenance, something that can be effectively avoided through crafting architectures with intentionally long-term materials.

Naturalizing Loss: Pace layers and abscission in software archives

The major ontological framing for the Code Vault comes in the form of a “Pace Layer Strategy:” a model published on their website that describes a “flexible, durable strategy for archiving code” by organizing each archival layer by its temporal horizon – or how long it might be expected to live.

Pace Layers

At the top of the temporal stack is GitHub, the platform. As the hottest layer, GitHub the platform is updated “near real-time,” where Git data is replicated across multiple datacenters around the world with every push. This is the site of data generation, and the central node through which users interact with their repositories. Here code is uploaded, forked, pulled,

starred, watched, tested and branched. Repositories are ranked, reviewed, modified, updated and fixed. And significantly, users can delete comments, versions, repositories, and their full accounts.

The second hottest layer is the GHTorrent, created in 2012 by Greek computer scientists Georgios Gousios and Diomidis Spinellis. The torrent, created via firehose to GitHub's REST API, monitors the GitHub public event timeline and creates an offline mirror. The datasets this creates has facilitated a number of large-scale studies and meta-analysis of GitHub performance metrics, as it is the only scalable, queriable, offline mirror of near-up-to-date data. By 2026, the domain, https://ghtorrent.org/, is inaccessible.

It is similar to the third layer, the GHArchive, is a project hosted by Ilya Grigorik (@igrigorik), an engineer at Shopify, that has collected over 15 event types, like new commits and fork events, and makes them accessible through GitHub's API as JSON files. The GHTorrent differs from the GHArchive in that while the archive collects and stores the event stream, the Torrent also "applies dependency based retrieval on all entities (e.g. commits, pull requests, etc.) that are linked from the events and stores the results in two databases: a raw data one (MongoDB) that stores the unprocessed responses from GitHub API and a relational one (MySQL) that stores links between the entities (e.g. commits are linked to projects)" (GHTorrent FAQ). Both storage projects create replicas of GitHub, creating conditions for increased data privacy breaches. As Gousios described in an interview, several years into the project they found that "even personal data, such as e-mail addresses or full names, were included." While this was public information that was collected by the archiving project, the creation of a mirrored databased for large-scale studies created the conditions for increased privacy breaches. These discoveries were cause for increased access restrictions for users.

At the fourth and fifth pace layer is the Internet Archive and the Software Heritage Foundation, who both crawl public repositories on a regular basis and provide access through their own APIs. In 2024, Hugging Face published The Stack v2, a training dataset from the BigCode project that contains 3.28B unique files belonging to 104.2M GitHub repositories that were collected by the Software Heritage Foundation. While the BigCode project strived to create a dataset of "open scientific data for the responsible development of Large Language Models for Code," many GitHub coders expressed their surprise and dismay. Matt Nish-Lapidus (@enemel) wrote in a post on Mastodon: "I found two of my old Github repos in there. Both were deleted last year and both were private. This is a serious breach of trust by Github and @huggingface."

Hugging Face created a tool, called "Am I in the Stack?", for coders to search for their repositories by username in the dataset. Here they can also request data to be removed from future versions of The Stack, as it will be marked with "-rc" to remove it from release. Another coder was concerned with licenses of code included in the BigCode project, where preliminary tests or checks for license files were not being performed by Software Heritage

Foundation, BigCode or Hugging Face. [3] The models released from this training dataset would obtain their own licenses, forgoing any previous conditions or agreements in the GitHub repositories used as training data. Because this project is community-facing, where coders are presented with their own repositories as a new type of resource for developing LLMs, Hugging Face's ethical response was to create a tool where anyone could remove their repositories and modify the stack. This layer of the Pace Layer is another instance where modifications to the 'archive' come from ethical and legal concerns expressed by the users.

At the sixth layer is the Arctic World Archive, the layer where the Piql film reels are located underground in a decommissioned mine shaft, within a steel vault. The code vault, despite its location in the Arctic surrounded by freezing temperatures, is still placed at the sixth layer of the temporal stack. It is revisited by members of the archive project team every five years for maintenance checks, but will otherwise remain unchanged since the inscription 21TB algorithmically selected GitHub repositories in 2020. This layer prevents any modifications or changes from users by preventing access to the contents of the archive.

After the Arctic Archive, at a temporal pace conceptualized as longer than steel and ice, are the copies that have been sent to the Bodleian, Alexandria and Stanford Libraries. Institutional libraries are known for their capacity to store and protect culturally significant artifacts, even in moments of global crises, although not without significant effort. The GitHub archives were sent to these libraries on Piql film stored in a wooden box, which have in some cases subsequently been placed in off-site storage facilities. This layer is not only difficult to access, it is also difficult to find. Even when visiting these locations, like the Bodleian Libraries, the Code Vault's whereabouts are not commonly known by archivists.

This is followed by an even more durable archival medium: glass. At the deepest layer of the stack is Microsoft's Project Silica, a cold storage project that has developed novel forms of data inscription and storage for "high-value, long-lived data that continues to grow exponentially" (Lunt, et al, 2025). By inscribing data into glass with a femtosecond laser, data is secured in place unless the glass is melted down. These glass plates are shelved across a "library" and designed to work alongside data center servers to serve as the stable, archival layer that is rarely accessed.

[3] https://news.ycombinator.com/item?id=39772212

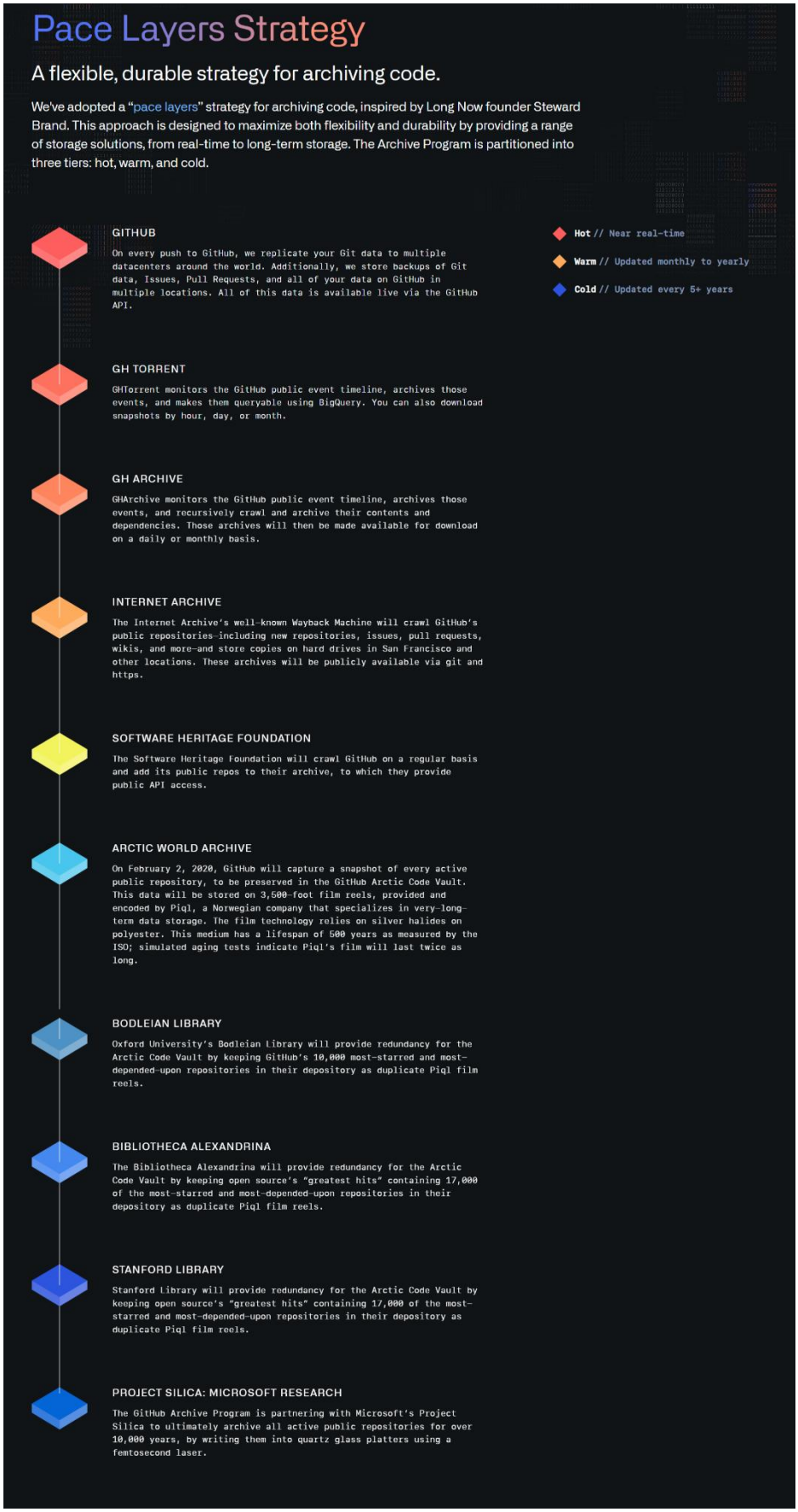

Figure 1: Arctic Vault Pace Layer Strategy, screenshot taken from https://archiveprogram.github.com/approach/ on February 10, 2026.

Abscission: Shearing Layers

The pace layer strategy created by the GitHub Code Vault project is directly modeled on Stewart Brand's (2018) "pace layer" model, originally developed in his 1994 book *How Buildings Learn: What Happens After They're Built.* In this model, each layer contains its own speed that either innovates or stabilizes society, where pace layers, also known as "shearing layers," are defined as "as a set of integral components that adapt or change in different timescales." As Brand writes, "… [because of] the different rates of change of its components, a building is always tearing itself apart" (p. 13). Shearing can therefore also be read as the act of destruction, cutting off or peeling away. The relationship made between time (pace) and loss (shearing) in this model is presented as a natural effect of evolution.

Brand's development of the shearing layer concept was also highly influenced by the work of two architectural theorists Christopher Alexander and Frank Duffy, both of whom he thanks in the acknowledgement section of his book. Alexander's first volume was an influential philosophy of architecture series called *A Timeless Way of Building* (1979), which was followed by *A Pattern Language* and *The Oregon Experiment.* These texts have bridged architectural studies to find purchase in information systems theory, software design and computation, influencing projects like the first Wiki and the creation of SimCity 2000. In *Timeless,* Alexander proposes novel conceptualization of buildings through the recognition of natural design patterns by introducing a "quality without a name." This quality is grounded in a "timelessness," where buildings can be imagined as "forests and meadows of the human heart" (p. 15), once identified through pattern languages. It is an evocative and influential text that places the human builder at the center, advocating for an openness to architectural design not steeped in architectural tradition, but in patterns of human co-existence.

Brand also modelled his pace layers on the work of architect Frank Duffy's "time-layered perspective" (p. 17), where he identified four temporal layers of a building that each have their own specific expiration ranges. Duffy advocates here for the role of the facilities managers as the people who introduce important dimensions of time and use into buildings, rather than architects. He states that rather than describing buildings by their material terms, we should describe them in terms of time (ie: shells that last up to 50 years, services that last 15 years, scenery that has a duration of five or less years, and sets that change daily), oriented around use, wear and breakdown informed by the records created by building managers.

Duffy draws attention to the often-overlooked role of building management and their intimate knowledges of buildings as spaces where humans participate in their use, wear and breakdown. He identifies that one of the flaws in architectural design is the bias towards hardware of the building, rather than the software of user requirements – to which the facilities manager is most connected. Duffy even states that these managers would be seen as "custodians of the future, rather than troubleshooters of the past" (p. 19). This emphasis on managers, or those who care for buildings and witness their various uses across time, is a call for a human-centered approach where building design is guided by use. In the Code Vault project, these building managers are akin to the software developers and archive maintainers, those who operate at the warmer layers of the stack. Rather than building long-term

preservation strategies that encompass and expand the efforts of already-existing archival projects, the Code Vault anticipates shedding these earlier, messier, more labour-intensive, and costly projects behind.

This mimics Brand's interpretations of Duffy's "no-maintenance building," where he writes "masonry buildings can endure neglect that would destroy other structures" (Brand, 122). To Brand, maintenance can be minimized through the careful selection of more durable materials that operate with a quality of forgiveness. To explore these qualities, he compares materials such as bricks which "manage time beautifully… last[ing] nearly forever" and concrete, "a miracle material." Concrete, he writes, is "impossible to change… so solid and so laced with steel reinforcing that cutting a new door or window is virtually unthinkable." It makes concrete an ideal building material, not unlike Arctic Archive's silver halide crystals microfilm, a WORM (write once read many) medium, that prevents loss through rewriting, or Project Silica's glass inscriptions. Once the data is taken offline and printed on the film, like concrete that cannot be cut into for doors and windows, no changes can be made to the code.

In all preservation and archival work, materiality often plays a central role in determining rates of decay or deterioration. While concrete is subject to "blistering, chipping, coving, cracking, crazing crumbling, etc…", brick buildings are different, in that, "once they become decrepit, ugly or irrelevant, they are either demolished with vast noise and expense or left to become particularly unattractive ruins." Concrete, on the other hand, "is treated like nuclear power: we try not to think about decommissioning." These declarative statements about materials and their inherent durability are echoed in the pace layer strategy, where the ordering of each layer indicates a process of assessment, anticipation and evaluation of its physical longevity, producing a hierarchy of existence.

The hierarchical logic embedded in each layer is not accidental, as the development of Brand's "shearing layer" concept also references ecological and systems theorists who write on "hierarchies of ecosystems". In their book, *A Hierarchical Concept of Ecosystems* (1986), O'Neill et al., describe a theory of ecology where hierarchies are based on process rates rather than taxonomical sorting. They write: "the dynamics of the system will be dominated by the slow components, with the rapid components simply following along," placing emphasis on the cold layer, where "the slow provides continuity and constraint." In their examples, they compare fast processes like photosynthesis with slow processes like soil development. This implies a natural, evolutionary style of thinking that places loss on a continuum – as part of the cycle of everything and necessary for evolutionary growth. When adapted for the Code Vault's, the pace layers at first glance suggest a software updating pattern, but in reality propose an order of decay, where the coldest layers, the deep time data archives that promise to prolong life the most, are presented as a potential answer to platform anxiety around obsolescence and endings.

Following along with ecological thinking present in this model, I draw from natural sciences as well to describe the processes of *abscission* to argue that the steps between the Code Vault's pace layers indicate where loss can occur. Abscission is a form of sloughing, which refers specifically to "the natural separation or detachment of a part of a plant, typically a

dead leaf or ripe fruit" (OED). The programmed detachment of a dead leaf occurs at specific points called "abscission zones," that demarcate an area on the branch structure where breaks are anticipated. These specialized cellular structures produce enzymes that break down the cell walls and create conditions of acceptable loss. While it is a natural and reoccurring event that follows chemical processes of plant life and a response to aging and environmental conditions, biotechnologists in the food industry also explore ways to control abscission to maintain fresh harvests and reduce loss during periods of food storage (Lers, 2012).

Similarly, Schneider's 'slough media' (2020) describes the process of obsolescence as a shedding over time – producing a transformation, akin to skin or layers peeling off — but applied to media and performance histories that don't simply disappear. Instead, Schneider argues, they linger, ossify, re-emerge, and remain in bodies and contexts across time. *Sloughing,* therefore, denies a binary presence-absence and instead invokes the notion that remains linger and persist differently through material transformations. Storage in deep time evokes a sense of permanency, that objects that are preserved in the "abyss of time" (Heringman, 2023) are inert and stable through their material sedimentation: biological remains of humans are recovered from natural preservation systems like bogs or permafrost, with materials like clothing still intact; radioactive materials are placed in deep geological "repository sites," that anticipate a future billions of years in the future, where the waste will eventually have to be dealt with.

In the Code Vault, ideas of permanency are also present, yet dispersed through their pace-layered approach: they imagine that only the deepest, coldest layer will exist in 10,000 years, and everything else will eventually fade away. This form of intentional loss, when analyzed through the lens of sloughing, is perhaps better understood as an abscission point: a natural breaking point where obsolescence can take place. As a slough archive, the Code Vault does not necessarily create permanent loss, as remnants of the code and the platform persist elsewhere – in machines, systems, memories and bodies. But the control over obsolescence that this project asserts through engagement with material permanence and anticipated losses indicates a shift for platforms, where the scale of their power and influence extend far beyond the utility of the current moment, and are instead imagined at the end of the world, persisting.

Conclusion: Code without Coders

Archival distortions are prevalent in big tech regimes (Hogan and Roberts, 2023), and in GitHub's Code Vault we can see how the LOCKSS principle (Lots of Copies Keep Stuff Safe) creates the conditions where data storage infrastructure is proliferated around the world. In a decommissioned mineshaft in northern Norway, in wooden boxes across libraries in the United Kingdom, Egypt, the United States, and in data centers across Europe and North America, we find material traces of over 17,000 algorithmically selected GitHub repositories. The contents of which cannot easily be accessed or read: some are temporarily inaccessible, others have been deleted, made private, or removed, while others can only be read with specific reader technology. Deborah Lupton (2018) has theorized data as producing a "new type of human remains," and in deep time data storage the transformation of data to persist across centuries produces a material object tethered to capitalist enterprise for eternity.

The Code Vault project's development of the pace layer strategy moves beyond archival distortions and borrows from architectural theory to suggest a natural hierarchy of code storage and a preference for algorithmically selected futures. In their temporal stack, it becomes clear that if platforms want to live forever, they have to become a *slough archive*: first shed the users, then the platform, then the code, and the archive, until all that's left is an inscription. While it may be easy to dismiss this as a vanity project with no real consequences for software availability, functionality or interoperability, if we do take seriously the economic investment in scientific experimentation with materiality, data storage and deep time, we can see at once the transformation of ongoing relational forms of software labour into inert archival objects by naturalizing processes of loss. In doing so, the Arctic Code Vault recasts platform power as a question of material survival, obscuring the ongoing social labor and maintenance practices that sustain software ecosystems.

There are, however, other preservation projects like the Code Vault that situate themselves as actors working against growing concerns about climate change, energy scarcity, security, and economic uncertainty. GitHub's archive project was introduced as something that would protect the "world's infrastructure" from security threats or crisis. Friedman even described the archive project as a "product that we hope never gets used," hinting towards an assumed catastrophic future that would necessitate the deployment of this powerful repository. Preservation projects like the seedbank and the code vault experiment not only with geopolitical placement (the advantages of Svalbard, both as a place and as an imaginary, is that it is located far away from sites of global crisis – like wars and climate change) but also experiment with materiality and longevity as part of a wider cryogenic culture and fascination with deep time.

Coders and software developers who have received a platform-badge from GitHub celebrating their contribution to the Code Vault often convey confusion or mirth: why *that* repository? How did *mine* make the cut? Should it really be preserved for '*eternity*'? Digital preservation projects take place all the time without consultation or consent: the web is constantly crawled by a number of actors, first from search indexes like Google and archives like the Internet Archive, and now from data holders like Common Crawl and AI developers, who each create massive data collections primed for actions like large-scale analysis. What sets the Code Vault apart is their conceptual contradictions: creating a hierarchy of preservation projects that will eventually all become obsolete, saving code forever by rendering it inoperable, and imagining the software age emerging at the end of the world. Software is notoriously difficult to preserve, and for GitHub to forego all previously held archival principles and practices in place of overt attempts at eternal legacy building demonstrates that despite previously held beliefs that platforms do not care about preservation, they are increasingly investing in material experiments for platform permanence.

References

Acker A (2021) Emulation practices for software preservation in libraries, archives, and museums. *Journal of the Association for Information Science and Technology* 72(9): 1148–1160. https://doi.org/10.1002/asi.24482

Acker A (2024) Accessing software: Emulation in information institutions. *Information & Culture* 59(1): 1–19. https://doi.org/10.7560/IC59101

Adam B (1998) *Timescapes of Modernity: The Environment and Invisible Hazards*. London and New York: Routledge. https://doi.org/10.4324/9780203981382

Alexander, C. (1979) *The Timeless Way of Building.* New York: Oxford Univ. Press.

Alt C (2011) Objects of Our Affection: How Object Orientation Made Computers a Medium. In: Huhtamo, E. and Parikka, J. ed. *Media Archaeology: Approaches, Applications, and Implications*. Berkeley: University of California Press, pp. 278-301. https://doi.org/10.1525/9780520948518-015

Au Y (2024) Data centres on the moon and other tales: A volumetric and elemental analysis of the coloniality of digital infrastructures. *Territory, Politics, Governance* 12(1): 12–30. https://doi.org/10.1080/21622671.2022.2153160

Banoub D and Martin SJ (2020) Storing value: The infrastructural ecologies of commodity storage. *Environment and Planning D: Society and Space* 38(6): 1101–1119. https://doi.org/10.1177/0263775820911942

Burgess J and Baym NK (2020) Twitter. New York: New York University Press. Available at: https://doi.org/10.18574/9781479841806.

Berger PL and Luckmann T (1991) *The Social Construction of Reality: A Treatise in the Sociology of Knowledge*. Penguin Books.

Bounegru L (2024) The platformisation of software development: Connective coding and platform vernaculars on GitHub. *Convergence: The International Journal of Research into New Media Technologies* 30(6): 2212–2232. https://doi.org/10.1177/13548565231205867.

Brand, S. (2018) 'Pace Layering: How Complex Systems Learn and Keep Learning', *Journal of Design and Science* [Preprint]. doi:10.21428/7f2e5f08.

Brand, S. (1995) *How buildings learn : what happens after they're built*. [New ed.]. New York: Penguin Books.

Braun V (2024) The stuff of memories: Planning hindsight in animal cryobanks. *Social Studies of Science* 54(5): 728–748. https://doi.org/10.1177/03063127241252081

Breithoff E and Harrison R (2020) From ark to bank: Extinction, proxies and biocapitals in ex-situ biodiversity conservation practices. *International Journal of Heritage Studies* 26(1): 37–55. https://doi.org/10.1080/13527258.2018.1512146

Bucher T (2012) Programmed sociality: A software studies perspective on social networking sites. Doctoral dissertation, Faculty of Humanities, UiO.

Burgess J and Baym NK (2020) *Twitter*. New York: New York University Press.

Chakrabarty D (2007) *Provincializing Europe: Postcolonial Thought and Historical Difference*. Princeton, NJ: Princeton University Press.

Cheney-Lippold J (2025) The silicon future. *New Media & Society* 27(7): 4164–4180. https://doi.org/10.1177/14614448241234864

Cohn ML (2019) Keeping software present: Software as a timely object for STS studies of the digital. In: Vertesi J and Ribes D (eds) *Digital STS: A Field Guide for Science & Technology Studies.* Princeton, NJ: Princeton University Press.

Corry, 202;

Corry F (2023) Never forget? Memory maintenance on an aging platform. *Convergence: The International Journal of Research into New Media Technologies* 29(3): 602–619. https://doi.org/10.1177/13548565221129224

Dieter M (2024) Interface critique at large. *Convergence: The International Journal of Research into New Media Technologies* 30(1): 49–65. https://doi.org/10.1177/13548565221135833

Duffy F (1990), "Measuring building performance". *Facilities*, Vol. 8 No. 5 pp. 17–20, doi: https://doi.org/10.1108/EUM0000000002112

Edwards PN (2021) Platforms are infrastructures on fire. In: Mullaney TS, Peters B, Hicks M and Philip K (eds) *Your Computer Is on Fire*. Cambridge, MA: The MIT Press, 313–336. https://doi.org/10.7551/mitpress/10993.003.0021

Gillespie T (2014) The relevance of algorithms. In: Gillespie T, Boczkowski PJ and Foot KA (eds) *Media Technologies: Essays on Communication, Materiality, and Society*. Cambridge, MA: The MIT Press. https://doi.org/10.7551/mitpress/9780262525374.003.0009

Halpern O and Günel G (2017) Demoing unto death: Smart cities, environment, and preemptive hope. *The Fibreculture Journal* (29). https://doi.org/10.15307/fcj.29.215.2017

Hansen KB and Thylstrup N (2024) Stack bricolage and infrastructural impermanence in financial machine-learning modelling. *Journal of Cultural Economy* 17(1): 20–38. https://doi.org/10.1080/17530350.2023.2229347

Heringman N (2023) *Deep Time: A Literary History.* Princeton University Press.

Hogan M and Roberts ST (2023) Archiving for extinction. *Media-N* 19(1): 7–26. https://doi.org/10.21900/j.median.v19i1.936

Hopman R (2024) Snails, time, data: On the politics of mass-digitization and the possibility of data drift. *Big Data & Society* 11(3): 20539517241267760. https://doi.org/10.1177/20539517241267760

Horst H, Sinanan J and Hjorth L (2021) Storing and sharing: Everyday relationships with digital material. *New Media & Society* 23(4): 657–671. https://doi.org/10.1177/1461444820953517

Kemper J (2024) Deep time and microtime: Anthropocene temporalities and Silicon Valley's longtermist scope. *Theory, Culture & Society* 41(6): 21–36. https://doi.org/10.1177/02632764241240662

Klik E (2026) *Negative Media: Erasure and the Limits of Retention*. Stanford, CA: Stanford University Press.

Lers A (2012) *Plant Biotechnology and Agriculture: Prospects for the 21st Century.* Academic Press.

Lingel J (2020) *An Internet for the People: The Politics and Promise of Craigslist*. Princeton, NJ: Princeton University Press.

Lunt BM, Riviera F, Santos J, Carreon A, Isaacson J and Linford MR (2025) Writing for the future: What are the options? *Journal of Imaging Science and Technology* 69(2): 1–7. https://doi.org/10.2352/J.ImagingSci.Technol.2025.69.2.020402

MacKinnon K (2022) The death of GeoCities: Seeking destruction and platform eulogies in web archives. *Internet Histories* 6(1–2): 237–252. https://doi.org/10.1080/24701475.2022.2051331

Mackenzie A (2006) *Cutting Code: Software and Sociality*. New York: Peter Lang.

Mackenzie A (2016) Infrastructures in name only?: Identifying effects of depth and scale. In: *Infrastructures and Social Complexity: A Companion*, 379–390. London: Taylor and Francis Ltd.

Mackenzie A (2018) 48 million configurations and counting: Platform numbers and their capitalization. *Journal of Cultural Economy* 11(1): 36–53. https://doi.org/10.1080/17530350.2017.1393443

Mattern S (2015) Deep time of media infrastructure. In: Parks L and Starosielski N (eds) *Signal Traffic: Critical Studies of Media Infrastructures*. Urbana: University of Illinois Press, 94–114.

Mbembe A (2024) *Brutalism*. Durham, NC: Duke University Press.

McCammon M and Lingel J (2022) Situating dead-and-dying platforms: Technological failure, infrastructural precarity, and digital decline. *Internet Histories* 6(1–2): 1–13. https://doi.org/10.1080/24701475.2022.2071395

Nadim T (2018) Haunting seedy connections. In: *Routledge Handbook of Interdisciplinary Research Methods*. London: Routledge.

Otto EI, Salka JH and Blok A (2023) How app companies use GitHub: On modes of valuation in the digital attention economy. *Journal of Cultural Economy* 16(2): 242–259. https://doi.org/10.1080/17530350.2023.2186916

Radin J and Kowal E (2017) *Cryopolitics: Frozen Life in a Melting World*. Cambridge, MA: The MIT Press. https://doi.org/10.7551/mitpress/10456.001.0001

Schneider R (2020) Slough Media. In Ewa Bal & Mateusz Chaberski (eds) *Situated Knowing: Epistemic Perspectives on Performance*. United Kingdom: Routledge, pp.15-43.

Sharma S (2014) *In the Meantime: Temporality and Cultural Politics*. Durham, NC: Duke University Press.

Starosielski N (2021) *Media Hot and Cold*. Durham, NC: Duke University Press.

Taylor ARE (2021) Future-proof: Bunkered data centres and the selling of ultra-secure cloud storage. *Journal of the Royal Anthropological Institute* 27(S1): 76–94. https://doi.org/10.1111/1467-9655.13481

Thylstrup NB, Archer M and Steiner H (2024) Desiloization and its discontents: The politics of data storage in the age of platformization. *Information, Communication & Society* 27(13): 2419–2437. https://doi.org/10.1080/1369118X.2024.2371803

Vance A (2019) Open Source Software Will Survive the Apocalypse in an Arctic Cave, *Bloomberg News*. https://www.bloomberg.com/news/features/2019-11-13/microsoft-apocalypse-proofs-open-source-code-in-an-arctic-cave?embedded-checkout=true

Velkova J (2019) Data centres as impermanent infrastructures. *Culture Machine* 18. http://culturemachine.net/vol-18-the-nature-of-data-centers/data-centers-as-impermanent/

Velkova J and Plantin J-C (2023) Data centers and the infrastructural temporalities of digital media: An introduction. *New Media & Society* 25(2): 273–286. https://doi.org/10.1177/14614448221149945

Wajcman J (2019) How Silicon Valley sets time. *New Media & Society* 21(6): 1272–1289. https://doi.org/10.1177/1461444818820073

White-Nockleby C (2022) Grid-scale batteries and the politics of storage. *Social Studies of Science* 52(5): 689–709. https://doi.org/10.1177/03063127221109605

Zee JC (2017) Holding patterns: Sand and political time at China's desert shores. *Cultural Anthropology* 32(2): 215–241.

Zielinski S (2006) *Deep Time of the Media: Toward an Archaeology of Hearing and Seeing by Technical Means*. Cambridge, MA: MIT Press.