Information, Communication & Society

ISSN: 1369-118X (Print) 1468-4462 (Online) Journal homepage: www.tandfonline.com/journals/rics20

Engineering infrastructural power over software production: the case of software-as-a-service

Donald Jay Bertulfo & Seda Gürses

To cite this article: Donald Jay Bertulfo & Seda Gürses (05 Jun 2026): Engineering infrastructural power over software production: the case of software-as-a-service, Information, Communication & Society, DOI: 10.1080/1369118X.2026.2676635

To link to this article: https://doi.org/10.1080/1369118X.2026.2676635

Published online: 05 Jun 2026.

INFORMATION, COMMUNICATION & SOCIETY
https://doi.org/10.1080/1369118X.2026.2676635

Routledge
Taylor & Francis Group

Engineering infrastructural power over software production: the case of software-as-a-service

Donald Jay Bertulfo and Seda Gürses

Faculty of Technology, Policy and Management, Delft University of Technology (TU Delft), Delft, Netherlands

ABSTRACT

Over the past two decades, vendors have moved the development, operation and maintenance of enterprise software into cloud infrastructures managed by a handful of cloud companies. Critical scholars have recognized this capacity to produce and deliver software at scale – now often referred to as software-as-a-service, or SaaS – as part of cloud companies' growing infrastructural power. However, while prior scholarship has examined the political economic ramifications of this entanglement, it often treats scalability and infrastructural power as accomplished facts, rather than contested concepts.

This article complicates these concepts by examining the emergence of SaaS at a time when neither the scalability nor the infrastructural power of cloud companies was yet stabilized. Drawing on diverse sources, it analyzes Salesforce as a case that provides insight into how incremental discursive and material efforts at consolidating its infrastructural power over *software production* shaped the conditions under which the deployment of scalable software services to a vast number of client organizations became possible. By foregrounding software production, this article treats scalability as a sociotechnical achievement forged alongside ongoing attempts by cloud companies to establish and defend their infrastructural power, rather than an inherent attribute of contemporary cloud infrastructures. In doing so, it contributes to critical scholarship on cloud computing by underlining the co-constitutive nature of infrastructural power and scalability while situating them as fragile – rather than firmly established and uncontested – outcomes of historical contingencies.

ARTICLE HISTORY
Received 5 September 2025
Accepted 14 May 2026

KEYWORDS
Cloud computing; software production; infrastructural power; platforms; software-as-a-service; genealogy

Introduction

The cloud has become a key computational infrastructure supporting the operational software needs of corporate and public organizations, providing client organizations[1] with scalable and dynamic access to software and computational resources for developing internal or customer-facing applications (Mell & Grance, 2011; Narayan, 2022). This

CONTACT Donald Jay Bertulfo donaldjay.bertulfo@gmail.com Faculty of Technology, Policy and Management, Department of Multi-Actor Systems, Delft University of Technology (TU Delft), Jaffalaan 5, Delft 2628 BX, Netherlands

emphasis on scalability has come to occupy a central role in critical accounts that situate cloud infrastructures in relation to the expansion of platform companies and, more broadly, to what has been termed 'platform capitalism' – a form of capitalism that comprehends digital platforms as a novel organizational form and regards digital technologies as central to economic and financial practices (Narayan, 2022; Roitman, 2023; Srnicek, 2017).

Within this literature, scalability appears as a core expression of cloud companies' infrastructural power. Within political science scholarship, infrastructural power has often been understood as the state's capacity to 'extract and deploy resources in the territory over which it rules' (Weiss & Thurbon, 2018, p. 780) and 'implement logistically its decisions throughout the realm' (Mann, 1984, p. 189). While the concept of infrastructural power has been commonly deployed in studies concerning nation-states, recent work has applied it to make sense of the ways by which digital platforms and their infrastructures are controlled by private and public stakeholders. Particularly with respect to cloud companies, critical scholars have pointed to concrete manifestations of infrastructural power, such as these companies' ability to centralize control over computational resources, orchestrate complex networked dependencies and disrupt established governance and regulatory arrangements (Luitse, 2024; Rikap, 2022; van der Vlist et al., 2024).

Existing critical scholarship thus offers rich accounts of how infrastructural power operates once established. However, precisely because it often places emphasis on how infrastructural power is exercised in practice, it tends to frame such power as already given. This framing leaves little room for examining the broader conditions through which infrastructural power is forged and – at least in relation to cloud companies – how such power came to be entangled with scalability. By contrast, this article shifts attention from the effects of infrastructural power to its conditions of possibility. Specifically, it opens both infrastructural power and scalability to critical scrutiny by treating these concepts as historically-contingent sociotechnical achievements rather than settled facts.

We recognize that such a project may be approached from multiple analytical entry points. Contemporary literature has developed an expansive view of cloud infrastructure that studies, and extends beyond, material manifestations of data centers or hyperscalers (Ensmenger, 2013; Hogan & Shepherd, 2015; Jansen & ten Oever, 2026; Neilson & Rossiter, 2022). An influential perspective conceptualizes cloud infrastructure as the foundation for cloud computing service models, namely software-as-a-service (SaaS), platform-as-a-service (PaaS) and infrastructure-as-a-service (IaaS) (Liu et al., 2011; Mosco, 2014). This directs attention away from a hardware-focused view toward the service models through which clients and end-users encounter cloud infrastructure.

Building on this perspective and aiming more towards analytical depth rather than generalizability, we chose software-as-a-service as our focal object of analysis. This decision was motivated by the consideration that SaaS predates the modern hyperscaler cloud infrastructures (e.g., Amazon Web Services, Microsoft Azure) that dominate current discussions on cloud scalability and infrastructural power. Examining its emergence thus allows us to study a formative episode in which the relationship between cloud scalability and infrastructural power were still being negotiated rather than already stabilized.

The emergence of SaaS also marks a pivotal conjuncture in the history of software. As will be developed in this article, SaaS can be traced to developments in the late 1990s

precipitating a reconfiguration of enterprise software production: software used in enterprises that had previously been bought from vendors or developed in-house and installed on-premises increasingly came to be centrally designed, developed, hosted, maintained and continuously updated in third-party computational infrastructures and offered to clients as 'services'. While this shift did not fully replace the on-premises model, it introduced a new infrastructural arrangement in which software production became embedded within third-party computational infrastructures that would later be captured under the moniker 'the cloud' (Magee & Rossiter, 2015).

In this article, we use the term 'emergence' as a deliberate reference to the Foucauldian genealogical approach, which informs our method. Genealogy invites a critical engagement with often overlooked phenomena by tracing the historically-contingent processes through which they come into being (Foucault & Rabinow, 1991; Garland, 2014; Tamboukou, 1999). This approach allows us to tease out the social, economic and technological conditions under which today's cloud-based software production – which collapses the distinction between software development and delivery – became possible. Additionally, the approach allows us to situate scalability as one aspect of cloud companies' infrastructural power.

Empirically, we anchor our analysis in a case study of Salesforce, a company prior historical literature has regarded as the 'poster child of on-demand software' (Campbell-Kelly & Garcia-Swartz, 2008) and a leader in specialized cloud services (Mosco, 2014). Though this literature has identified Salesforce as an emblematic actor in the history of SaaS, we do not treat it as a representative case. Instead, we view it as a site where crucial struggles around SaaS were played out, contested and negotiated through time. This treatment aligns well with genealogy's rejection of totalizing histories and its emphasis on situated, local and partial trajectories (Garland, 2014).

Our analysis draws on a diverse set of sources, combining materials related to Salesforce's corporate history and evolution – including early business and technology articles, investor reports, publicly accessible interviews with key Salesforce employees, technical documentation and training videos, among other media coverage – with sources addressing the broader enterprise software and the Application Service Provider (ASP) industry in which Salesforce was embedded. As a reference for identifying pivotal events, we used *Behind the Cloud* (Benioff & Adler, 2009), Salesforce CEO Marc Benioff's autobiographical account of Salesforce's corporate history. However, instead of treating the company's CEO accounts — as well as the viewpoints of other Salesforce's executives and employees in other sources — as authoritative, we brought them in dialogue with other contemporaneous and retrospective sources,[2] and situated them within the broader historical and economic contexts in which they were articulated.

As we shall see, Salesforce was initially seen as part of a group of firms in the late 1990s offering software as a 'hosted service'. Prior work in software history has argued that such firms shifted the infrastructural locus of software from client premises or devices toward external computational infrastructures they managed (Campbell-Kelly & Garcia-Swartz, 2008). Our analysis, however, shows that Salesforce did more than shift the locus of software. It centralized the deployment and ongoing development of its software within computational infrastructures the company built and operated (Huber, 2008). This centralization enabled Salesforce to incrementally expand the functionalities, user base and scope of its service, gradually evolving Salesforce's computational infrastructure into

what we call a software 'production environment'. This shift later reconfigured provider-customer economic relations, allowing some clients to move beyond being mere users of Salesforce's hosted service toward producing software within Salesforce's computational infrastructure. This, in turn, resulted in an arrangement in which Salesforce increasingly determined not only client workflows, but also how and where they carried out their software production activities (Magee & Rossiter, 2015).

These dynamics suggest that Salesforce offers a generative case for examining how what can now be understood as a 'cloud company' gradually consolidated infrastructural power over not only its own software production but also that of its clients. In this sense, the case allows for an empirical analysis of a cloud company's infrastructural power as a contested and evolving capability to structure the conditions under which software is developed, deployed and consumed as digital services.

Our analysis provides an entry point for examining infrastructural power in the specific context of software production and contributes to interdisciplinary scholarship in several respects. First, its focus on SaaS responds to recent calls within cloud computing scholarship to attend more closely to business-to-business dynamics by taking software as a focal analytical object (Narayan, 2022, 2023). In doing so, it extends existing work affirming SaaS as a critical juncture in the business history of software (Campbell-Kelly, 2004; Yost, 2017) and complements prior work in critical media and technology studies that has largely approached SaaS from cultural, organizational and technical perspectives, such as its diffusion across different social and institutional domains (Kaldrack & Leeker, 2015), its reformatting impacts on organizational work (Randerath, 2024) and its implications for practices such as privacy engineering and regulation (Gürses & van Hoboken, 2018).

Additionally, our analysis connects software production with infrastructural power in ways that build on foundational literature within science and technology studies (STS), organization studies and infrastructure studies that emphasize how technologies and infrastructures are incrementally built, maintained, and reworked (Bowker & Star, 1999; Orlikowski & Scott, 2008; Plantin et al., 2018; Pollock & Williams, 2008). It foregrounds how infrastructural power, rather than existing as a fixed or static condition, unfolds temporally through situated practices and devices (Mukerji, 2009; Poon, 2012).

The succeeding sections proceed as follows. We first examine a class of software companies that proposed to take over client organizations' software operations, marking a shift in enterprise software delivery prefiguring today's software-as-a-service. While several of these firms built their business around such an infrastructural shift, not all went as far as to establish control over the production of software. Taking this distinction into account, we examine Salesforce as an empirical site that provides a window into how such a reconfiguration of software delivery gradually began to take shape as infrastructural power over software production. Our analysis ends at Salesforce's platform turn as a continuation and amplification, rather than an endpoint, of its infrastructural power over software production.

Each section uncovers how Salesforce's efforts in producing a technically-scalable web-native, multitenant software, offered via a subscription model, eventually enabled it to orchestrate a more expansive environment for software production. These sections illustrate the incremental actions through which Salesforce came to establish and maintain its infrastructural power while exposing its successes, failures and fragile

contingencies. We end with a discussion of the broader implications of this case study to understandings of cloud infrastructures, infrastructural power and scalability.

Shifting the infrastructural locus of software

The late 1990s saw the rise of application service providers (ASPs), a class of software companies that proposed to reorganize enterprise software delivery around remote access to applications hosted on third-party computational infrastructures. Rather than delivering software as packaged products or custom-built systems, these firms provided access to software over network connections, typically a wide area network or the Internet. They were described as providers of 'clientless' or '(Web)-hosted' applications (Schwartz, 2000), offered as *services* from their own-operated computational infrastructures – an arrangement that would later be associated with software-as-a-service.

ASPs were frequently portrayed in industry discourse as a response to the complexity of operating enterprise software. At the time, such software was typically deployed on-premises within enterprise-controlled computational infrastructures or installed on client devices (Pollock & Williams, 2008). In the on-premises model, client organizations bore responsibility for performance optimization, security and maintenance of software systems. These responsibilities often entailed substantial operational costs and required specialized staffing.

Against this backdrop, industry discourse positioned ASPs as an economical alternative to on-premises deployment. They were framed as relieving client organizations of burdensome operational responsibilities, allowing them to focus instead on core business activities (Cox, 1998). However, such relief was contingent on a material reconfiguration in which software resides within the service provider's computational infrastructure rather than the client's. This reconfiguration was accompanied by a corresponding shift in control over software deployment and operations from client organizations to ASPs, providing the latter with infrastructural power over conditions of service provision, such as which functionalities are made available to clients or when software is updated.

Such control over service provision can be situated within longer trajectories in the computer services industry, specifically time-sharing and facilities management service firms, both of which emerged in the 1960s and developed through the 1970s into the 1980s (Campbell-Kelly, 2004; Yost, 2017). Time-sharing firms provided access to computing resources through centralized computational infrastructures (Campbell-Kelly, 2004; Yost, 2017) while facilities management firms took over their clients' entire data processing operations, typically delivering these services through on-site work within client facilities rather than via remote and network-based access (Yost, 2017). These earlier models reflect forms of centralized provision of computing resources and outsourcing of IT operations that prefigure the ASP business model.

However, rather than comprising a singular model of service provision, ASPs encompassed a range of service architectures underpinned by varied configurations of software ownership and operational responsibilities (Krill, 2002; Maselli, 2000b; McCarthy, 2002). Some ASPs hosted client-purchased packaged software and assumed responsibility for operating it on the client's behalf. Others acquired packaged software from vendors and focused on building and operating the underlying infrastructure required for

multi-client deployments. Still others were packaged software vendors that expanded into hosting their applications in third party or proprietary data centers. Finally, some ASPs developed 'homegrown' or 'web-native' applications especially designed for operation within their own-operated computational infrastructures. This diversity suggests that the ASP model does not necessarily imply a rupture from packaged applications, complicating the idea of a clean shift from software-as-products to software-as-services.

Additionally, this variation suggests that Web-native ASPs, which consolidated software development and operations within computational infrastructures they themselves operated, had greater control over software production across the full development-to-deployment pipeline. This was less the case for ASPs tethered to packaged software vendors which typically released software every one-to-two years. In contrast to Web-native ASPs, these firms operated under conditions where software development and operations were structurally decoupled, thereby limiting their ability to exercise control over how their service offerings could be modified or extended.

These differences became evident as ASPs became more widely adopted in the software industry, particularly in domains where packaged software occupied strong market positions. While ASPs providing remote access to word processing, spreadsheets, presentation graphics and website creation software gained traction, tensions between packaged software vendors and ASPs became especially pronounced in the customer relationship management (CRM) domain (Tehrani, 2003). A notable actor in this contestation was Salesforce, whose co-founder and CEO, Marc Benioff, explicitly positioned the company as a challenger to the then-prevailing model of packaged software deployed on-premises.

Challenging the on-premises, packaged software model

In the 1990s, particularly in the United States, firms increasingly relied on geographically-dispersed sales forces, with many salespeople working outside headquarters. This dispersion generated new challenges in coordinating customer information flows and establishing managerial visibility over sales activity. It was in this context that CRM software emerged as a technology for coordinating access to customer information across the enterprise.

Rather than serving a single user group, CRM software supported varied forms of use across different roles within enterprises. For salespeople, it offered tools for managing their customer relationships, such as tracking leads and opportunities and updating customer information (Fox, 2003; Hall, 2000). For sales managers, it enabled the monitoring, analysis and evaluation of sales force performance (Fox, 2000) while for top management, it presented a means of helping enterprises understand, serve and anticipate customer demand (Apicella, 2002).

Because like other enterprise systems, CRM software typically required monetary investments, enterprises' decision to procure such technology partly depended on its perceived strategic importance and associated implementation risks. However, in contrast to mission-critical, back-office enterprise systems such as enterprise resource planning (ERP), CRM was often regarded in the industry discourse of the late 1990s as a non-core, front-office function (Sweat, 1999). At the same time, enterprises faced uncertainty surrounding returns on investment in CRM systems, underpinned by concerns about CRM adoption among one of CRM's key user groups: salespeople. Industry articles

depicted salespeople as not particularly 'technology-savvy' (Torode, 2001). Some reports indicated that even after having been trained in the use of CRM software, many salespeople remained reluctant to use it, fearing that it could undermine their selling practices, for example by exposing their sales performance to managerial scrutiny or requiring them to enter their contacts into systems accessible to colleagues (Hall, 2000; Sweat, 2000).

CRM's positioning as a peripheral enterprise function together with uncertainties surrounding its adoption weakened justifications in favor of procuring and internally operating CRM systems. These made CRM software appear more amenable to outsourcing than other enterprise systems (Apicella, 2002; Bednarz, 2002). In this context, CRM ASPs emerged as a low-risk means of trialling CRM systems (Mears, 2001) that could be deployed quickly (McKenzie, 2001) and expanded to serve additional end-users and organizational departments as needed (Sweat, 2000). This experimental framing helped normalize remote access to CRM software from third-party computational infrastructures.

It was within this emerging configuration of hosted CRM provision that Benioff established his software business. He founded Salesforce in March 1999 following the acquisition of Left Coast Software, a sales force automation (SFA) start-up owned by three engineers who later joined Benioff as the company's co-founders (Entrepreneurship at Cornell, 2023). Less than a year later, the team launched the company's first CRM offering. The service gave end-users (e.g., salespeople and managers) remote access to tools for managing their sales activities, such as account and contact management, forecasting and reporting (Sweat, 2000). Like many ASPs of the period, Salesforce adopted a subscription pricing model. Its service was free for the first five users, followed by a monthly fee of $50 per end-user thereafter.

Salesforce launched its service offering in an already-competitive CRM software market shaped by the growing importance of managing customer relationships in distributed enterprise environments (Marti et al., 1999). By 1999, the incumbent leader in this market was Siebel Systems. That same year, Deloitte & Touche and Fortune Magazine named Siebel the fastest growing tech firm in America (Saracevic, 1999). The company's packaged CRM application was embraced by large multinational corporations, which normally possessed the financial capital for expensive on-premises deployments.

At the same time, Siebel had begun its foray into hosted software delivery to attract small and midsized firms as well as individual salespeople. This was exemplified in its launch of Sales.com, a web-hosted CRM service providing sales professionals with tools to track their opportunities and account information in exchange for a monthly fee (Gilbert, 1999). Similar moves were visible outside the CRM sector. Oracle, for instance, began leasing access to its ERP applications while continuing to sell packaged software as its primary business (Busse, 1998; Sweat, 1999). These developments suggest blurring distinctions between packaged software vendors and early ASPs as vendors started experimenting with hosted software models to expand their client base.

It was at this juncture of fluid boundaries that Salesforce positioned itself discursively at the forefront of an industry-wide shift. It did so through a marketing strategy centered on redefining – or, as the company put it, 'ending' – software as it was then known. Branded the 'End of Software' campaign,[3] the strategy framed web-based software delivery as a fundamental break from packaged software. A key moment in this campaign was a mock protest at a Siebel User Conference in February 2000, where hired protesters wore red t-shirts bearing the words 'death to software', carried signs with anti-software

messages, and chanted slogans (Holmes, 2001). The spectacle was further amplified by actors posing as crew members of a fictitious TV channel covering the event. The protest attracted the attention of onlookers, reportedly prompting Siebel to block it with parked semi-trucks (Holmes, 2001).

Salesforce's 'End of Software' campaign translated a technical and organizational problem – namely, that software was not only costly to procure but also costly and cumbersome to operate on-premises (Krill, 2002; Mears, 2001) – into a legible critique. Mobilizing this rhetoric, Salesforce's executives argued that its service offering made CRM software accessible to small and midsized firms, unlike Siebel's on-premises software, which catered mainly to large and well-resourced organizations. This discourse of democratized software access contributed to strengthening not only Salesforce's market positioning, but also to framing centralized service provision in third-party computational infrastructures as a competing alternative to the prevailing on-premise model of operating enterprise software.

Producing a web-native, multitenant software

Was the 'End of Software' merely rhetorical? If not, how was this claim realized in the design and operation of Salesforce's service? Accounts indicate that Salesforce realized its claim through technical choices that consolidated the full lifecycle of software production – including the development, deployment and operation – within its own-operated computational infrastructure. Such consolidation was enabled by its early decision to produce 'homegrown software' (Murphy, 2001). This entailed having full ownership of the codebase (Hall, 2000), providing Salesforce with ample latitude to define its own design and architectural scaling strategies.

Salesforce's technical strategy from the very beginning, according to its early engineers, was to design a service that could operate at web scale. An architectural principle called *multitenancy* came to be central to actualizing this vision. Salesforce came to run 'multiple tenants' – essentially its entire customer base – within a shared database architecture and standardized infrastructure, with tenant data separated logically rather than through dedicated hardware and software stacks (Torode, 2001).

Early Salesforce engineers and executives emphasized that the company's decision to adopt a multitenant service architecture was a massive undertaking that involved technical challenges. A key challenge concerned the computational infrastructure underlying the service, which had to be built from the ground up, standardized and purpose-built for multitenancy (Torode, 2001; Constellation Research, 2019). As Salesforce's early chief information officer later recalled, the tools and services that today abstract infrastructure provisioning — enabling startups to deploy software quickly and at-scale — did not yet exist back then (Constellation Research, 2019). To establish the technical conditions under which a multitenant architecture could function, Salesforce's early engineers had to configure foundational components such as file servers and local control systems internally.

These efforts gave rise to a technical architecture that later came to be understood as a crucial enabler of Salesforce's economies of scale in software delivery (Weissman & Bobrowski, 2009), allowing the company to expand its customer base without a proportional increase in cost. Industry articles of the time highlighted two economic

advantages of multitenancy. First, it enabled Salesforce to onboard new clients without needing to rebuild the underlying operational setup for running and maintaining the software. Second, it allowed Salesforce to unilaterally deploy service functionality extensions, security patches and maintenance updates uniformly across its entire client base. These affordances – namely operational cost reduction and scaled implementation of maintenance and updates – are both tied to the infrastructural control afforded by a multitenant architecture. By contrast, ASPs that leased vendor-made packaged software were constrained in their ability to achieve economies of scale as they typically deployed a discrete instance of the software on a dedicated server for each customer (technically referred to as a *single-tenant* architecture).

The infrastructural control that multitenancy afforded Salesforce was, however, neither absolute nor uncontested. Client organizations, mainly large enterprises already invested in on-premises enterprises systems, were initially skeptical and, in some cases, resistant to Salesforce's offering. Their reasons had to do with some material consequences of multitenant service production. Multitenancy initially entailed an arrangement in which centralized software production notably limited its clients' ability to customize or integrate Salesforce's service with existing systems. Shared infrastructure also raised data separation and security concerns, such as clients inadvertently accessing each other's sensitive customer data or security breaches affecting many clients at once.[4] Additionally, unilateral management of updates and maintenance constrained clients' ability to adjust their infrastructure workloads (including decisions over server uptime and performance). Although some of these limitations initially hindered adoption among large enterprises, they would later become sites around which accommodations and adaptations were organized as Salesforce sought to maintain its infrastructural power over its own software production.

Economic pressures during the dot-com downturn made the limitations of multitenancy more acceptable to large enterprises. As the downturn diminished IT budgets across many firms, hosted solutions with lower upfront capital requirements emerged as credible alternatives to capital-intensive on-premises systems (Bednarz, 2002; Ricadela, 2006).

At the same time, the dot-com implosion exposed vulnerabilities in the vendor-dependent, single-tenant ASP model. Unlike multitenant ASPs such as Salesforce, single-tenant ASPs typically operated expensive third-party enterprise software while relying heavily on external financing to sustain operations and subsidize their services often initially offered at low prices to attract clients. As client payments became less reliable and venture capital funding contracted during the downturn, these arrangements became difficult to maintain (Hall, 2001; Maselli, 2000a). In response, some restructured their client base by terminating their contracts with their most financially-precarious clients while keeping those with more stable revenue streams (Maselli, 2000a). Others exited the market through bankruptcy filing or acquisition (Maselli, 2001).

Salesforce itself was not insulated from the destabilizing effects of the downturn. Although it appears to have been less exposed to venture capital withdrawal than many of its contemporaries in the ASP industry due to its early reliance on non-institutional funding sources, particularly capital from family and friends, the downturn nevertheless revealed a deeper tension within Salesforce's business model. Its approach, premised on ongoing software production within a centrally managed, third-party

computational infrastructure, depended on a subscription-based revenue generation strategy that was itself vulnerable under conditions of economic contraction. As elaborated in the next section, Salesforce's response to this tension involved a recalibration of its subscription model.

Subscriptions as attachment devices

Like many other ASPs, Salesforce's business and infrastructural expansion depended, at least in part, on the continuous collection of substantial subscription revenues. Initially, Salesforce collected subscription fees from its clients on a monthly basis. This billing schedule functioned as a low-commitment entry and exit mechanism, assuring clients that they could easily discontinue the service whenever they felt dissatisfied. Additionally, and as earlier noted monthly billing rendered the company's hosted service more affordable relative to the substantial upfront costs associated with perpetual licenses of packaged software, thereby contributing to lowering adoption barriers for small and mid-sized firms.

Though monthly subscriptions proved to be a persuasive financial hook for clients, it turned out to be a fragile driver of revenue generation for Salesforce. This fragility was exposed during the dot-com downturn when the company's low-commitment proposition backfired. While monthly billing enhanced the attractiveness of Salesforce's service to larger organizations seeking to reduce IT spending during the dot-com downturn, it also allowed existing clients to cancel their subscriptions with little friction or consequence (Benioff & Adler, 2009; Storey, 2018). Thus, what had functioned as an incentive mechanism for service adoption also revealed itself as an impediment to cash flows required to sustain Salesforce's infrastructural operations.

Faced with this issue, Salesforce recalibrated its subscription pricing from monthly billing to annual billing upfront. The recalibration was incentivized and implemented through discounted pricing: existing customers could continue at the original monthly rate provided they signed an annual contract and paid cash upfront, while those who preferred monthly billing were offered a higher monthly rate (Benioff & Adler, 2009; Storey, 2018). The shift from monthly billing to annual billing upfront turned subscriptions from instruments of customer-base expansion and low-friction adoption into attachment devices that reduced customer churn risk and bound clients into longer-term economic relationships with the company.

Beyond forging stickier relationships with clients, annual billing upfront materially reconfigured Salesforce's financial architecture. As clients were required to pay upfront for services that would be delivered over time, Salesforce accrued advanced payments known in accounting terms as 'deferred revenue'. This reconfiguration had two distinct but related effects.

First, upfront payments generated immediate liquidity, providing Salesforce with cash resources that could be used in the ongoing maintenance and expansion of its service and computational infrastructure. Early Salesforce employees later highlighted the importance of this liquidity effect, identifying deferred revenue from annual subscriptions as a key factor in Salesforce's transition from a negative to a positive cash flow position within a year of adopting annual subscriptions (Benioff & Adler, 2009). This account is consistent with Salesforce's 2003 annual report, which indicated that the company

funded its operations through cash generated by its operating activities during that reporting year (Salesforce, 2003).

Second, though clients paid upfront, these payments were gradually recognized as 'income' over the contract duration. Fluctuations in new or renewed subscriptions in a given quarter were therefore not immediately reflected in reported revenue but were instead distributed across subsequent quarters (Salesforce, 2003). This time-lagged revenue recognition reduced the visibility of short-term volatilities in subscription activity, creating a more even distribution of revenue across reporting periods.

Salesforce's leadership later noted that this revenue-smoothing effect, combined with the customer retention effect of annual billing upfront, played an important role in the success of Salesforce's initial public offering (IPO) bid (Benioff & Adler, 2009). The IPO provided Salesforce with access to substantial external capital, strengthening its ability to further expand its offerings and underlying computational infrastructure.

The practice of annual billing and deferred revenue recognition continued after the IPO. As of 2025, Salesforce continues to invoice clients in advance, normally in yearly intervals with payments due within 30 days of invoice (Salesforce, 2025). This persistence suggests that the subscription model remained central to Salesforce's long-term infrastructural expansion even with the presence of capital market financing.

Salesforce's platform turn

Though Salesforce's web-native, multitenant and subscription-based service became legible and increasingly attractive not only to small and mid-sized firms but also large enterprises especially at the aftermath of the dot-com implosion, tensions remained around the limitations of its centralized mode of software production. Many enterprises continued to perceive Salesforce's service as inferior to on-premises systems such as Siebel's. These firms demanded customizability, integration with existing systems and stronger performance and security – requirements that seemed in conflict with Salesforce's technical architecture and challenged Salesforce's prospects for further expansion.

Rather than addressing these demands through a single redesign, Salesforce pursued incremental extensions that progressively responded to these concerns while expanding its infrastructural power not only over its own but also its clients' software production capabilities. Salesforce first expanded its service through additions in functionality delivered via its own-operated computational infrastructure. This is exemplified in the company's release of its E-Business Suite and an Offline Edition, released with the Enterprise Edition in 2002. The former added back-office applications such as e-billing, and management of invoices, contracts and orders to Salesforce's CRM service (Krill, 2002), while the latter enabled offline use as Salesforce retained control over synchronization and data storage. In both cases, service functionalities were enhanced without giving clients control or authority over code execution.

Salesforce also redefined customization and integration as capabilities configured within its computational infrastructure rather than as client-side modifications. The Enterprise Edition enabled client organizations to modify their interfaces for different internal groups, such as sales teams or departments (Hagendorf Follett & Darrow, 2002; Krill, 2002; Maselli, 2002) and use XML-based interfaces to integrate the company's CRM service with their existing legacy systems (Maselli, 2002). While these capabilities

allowed users to modify their experience of Salesforce's service, these modifications remained fully governed within Salesforce's computational infrastructure.

Customizability was made possible by through a 'metadata-driven architecture'[5] that reportedly required substantial and costly infrastructural rewiring (Varley, 2017) of the underlying multitenant service architecture in coordination with external business partners (McDougall, 2003). The resulting system interpreted user-defined configurations in real time and stored them as metadata – all of these while promising to maintain performance, stability, and security across thousands of clients. This allowed users to tailor their software experience with simple actions like renaming tabs, adding fields, or creating new objects. As users could not modify the application logic themselves, the system did not admit full customizability, but instead only 'customization through configuration' (Margulius, 2003). Thus, the metadata-driven architecture enabled Salesforce to claim an artificial equivalence between customization and administratively manageable configurations, providing service flexibility while simultaneously consolidating infrastructural control at the level of system architecture.

Meanwhile, infrastructurally-governed integration with existing systems, as suggested by industry accounts, linked to Salesforce's adoption of Web services (Udell, 2002).[6] Before developing into a modern web standard in the decades that followed, Web services were diverse technologies and protocols that gained prominence for allowing clients to connect third party services within existing internal systems (April & Harreld, 2002). In the early 2000s, practitioners and analysts already saw Web services' potential not only to allow clients to access applications or services from different providers but also to combine these distinct applications into new services that could generate additional value for clients (Dickerson, 2002; Margulius, 2002). However, Salesforce's Enterprise Edition only reflected a limited application of Web services. Even though the upgrade extended Salesforce's infrastructural reach by enabling data exchange with other systems operated by client organizations, it remained oriented toward interoperability rather than software production. While external systems could connect to Salesforce's service, client organizations could not yet produce new applications within Salesforce's computational infrastructure.

This changed in 2003 with the launch of *sforce*, a set of enterprise software development tools that exposed Salesforce's data and functionality via Web services to third parties, thereby allowing them to develop new applications within Salesforce's computational infrastructure (Songini, 2003). Sforce therefore marked the expansion of Salesforce's software production beyond its own engineering team. However, this expansion was circumscribed: clients could *extend* Salesforce's CRM, but not yet develop their own services independent of Salesforce's service. Client-led extensions instead operated within Salesforce's infrastructure, interacting with its core services while complying with its technical and contractual constraints. Even so, by opening its computational infrastructure to third party developers, Salesforce began to turn some of its customer base into prospective complementors to its software production (Songini, 2003). Its leadership framed this as a redefinition of 'what it means to be an ASP' (Schwartz, 2003). Importantly, with sforce, Salesforce no longer simply produced a hosted service. Instead, it began to host software development itself.

In this sense, *sforce* can be understood as prefiguring Salesforce's evolution into what technical scholars call 'platform-as-a-service' (PaaS). This transformation materialized more fully with the 2007 roll out of Force.com, an offering that enabled users to

independently develop and deploy business applications as services (Arrington, 2007). With Force.com, Salesforce's computational infrastructure became what we refer to as a more full-fledged 'production environment' for software. The service provided developers with runtime, a programming language (Apex), and an execution environment, abstracting infrastructure management while drawing the production and operation of their applications into Salesforce's computational infrastructure.

Force.com complemented AppExchange, a marketplace for enterprise apps launched the previous year, where Salesforce's clients, developers and partner companies could distribute and acquire business applications (Kawamoto, 2006). The AppExchange thus Salesforce's transition into what business studies scholars describe as a multi-sided market or a 'platform' company. In tandem with Force.com, AppExchange established an arrangement in which client organizations' software production was encouraged and monetized while remaining embedded within Salesforce's computational infrastructure. Through this configuration, Salesforce extended its infrastructural power over software production from consolidating its software production within its own infrastructure to orchestrating an ecosystem of interdependent software producers.

Conclusion

Using Salesforce as a window into the emergence of SaaS allowed us to examine cloud infrastructure as a historically-contingent sociotechnical formation. Our analysis unfolded how Salesforce's cloud infrastructure evolved iteratively and in tandem with incremental efforts that became pertinent to scaling the company's web-hosted, subscription-based, multitenant service. In doing so, it revealed software production as a contested arena in which cloud infrastructure and its scalability were co-constituted.

In Salesforce's early years, what later came to be called 'cloud infrastructure' was still largely organized around remote servers hosting packaged applications. This was part of a broader reconfiguration that began to shift control over software deployment and operations from client organizations to service providers. ASPs exemplified this shift in the infrastructural locus of software operations (i.e., where software resided and was operated) that occurred contemporaneously alongside a shift in software's commodity form from finished *products* to continuously-updated and maintained *services*. ASPs promoted off-premise software delivery that, in some cases, took the form of migrating or replacing clients' on-premises systems into computational infrastructures they operated. In this light, the shift now commonly described as a 'move to the cloud' was already underway more than two decades ago, complicating accounts of cloud infrastructures as recent technological developments.

While all ASPs offered software as services, variations in their underlying technical architecture impacted their capability to produce and deliver scalable services. When they leased vendor-produced software, ASPs were challenged by the consequent decoupling of software development and operations, limiting their ability to scale both their service offerings and their client base. By contrast, ASPs that produced homegrown software, like Salesforce, enjoyed architectural autonomy that allowed them to consolidate infrastructural control over their entire software production pipeline. Salesforce's case demonstrates how this infrastructural power made possible technical and organizational decisions that gradually expanded the scalability of its service across various dimensions

– from expanding its customer base, to mobilizing financial resources, and from evolving its service functionalities to enabling different parties to extend the use cases of its service and infrastructure.

Our analysis illustrated moments in which specific service scaling opportunities – such as establishing economies of scale or scaling service functionality to include customization through configuration – required deliberate engineering of the underlying computational infrastructure. These accounts highlight not only the intertwinedness of infrastructure and service scaling, but also their status as an actively-engineered achievement, rather than one that is preordained or passively instantiated. Re-anchoring this insight back to SaaS as a model for delivering a scalable resource (i.e., software), our analysis allowed us to situate scalability as an emergent outcome of the ongoing and joint reconfiguration of software production and computational infrastructures rather than as a mere corollary of elasticity.

In addition to these points, our analysis of the incremental actions that reconfigured software production in relation to broader social and economic conditions revealed how manifestations of infrastructural power over software production were politically contested. For instance, client organizations with on-premises systems initially resisted ceding control over parts of their IT infrastructures, questioned data security and demanded customization and integration that early SaaS companies could not readily provide. The ASP upheaval during the dot-com implosion represented another broader economic condition that tested the financial viability of many ASPs, as well as the business robustness of their technical architecture. Salesforce navigated these pressures through a combination of discursive and material (technical and economic) strategies, including the company's public challenge to Siebel, its insistence on a fully web-native multitenant service, its refusal to offer on-premises deployment, its restructuring of its subscriptions model to forge sticky relationships with clients, and its deployment of recursive updates to its service to retain existing customers while attracting new ones. Together, these efforts progressively entrenched Salesforce's infrastructural power while exposing it to new forms of political contestation. Importantly, these actions were part of the ongoing work through which Salesforce asserted, defended and stabilized its infrastructural power. In this light, the case complicates accounts that treat the 'infrastructural power' of cloud companies as an already-achieved capability (Luitse, 2024) and aligns more closely with work in STS and anthropology of socio-technical infrastructures that treats infrastructures and their power as continuously maintained and politically-contested arrangements (Bowker & Star, 1999; Poon, 2012).

Finally, the case opens opportunities for further critical reflection on the relationship between cloud, scalability and platforms. While Salesforce was from the outset what might today be called a 'cloud company', it was not initially a 'platform company', neither in the economic sense of being a multi-sided market, nor in the technical sense of being a 'platform' for developers to create new services (Gillespie, 2010). It took incremental steps and more than half a decade for Salesforce to add a marketplace for enterprise applications and a 'platform-as-a-service' (PaaS) to its core offerings. These additions evolved Salesforce's cloud infrastructure from a site of hosted software delivery into a more expansive and evolving environment for software production. By attending to the production processes, including the political and economic forces that condition platform emergence, the production view we introduce here corrects a tendency in some literature to conflate cloud companies with platform companies.

Notes

1. In this paper, we use 'client organizations' interchangeably with 'clients', both denoting organizational customers of cloud companies. Within 'client organizations', there are 'customers/users' (businesses) or 'end-users' (individuals) of cloud services.
2. We have gathered numerous sources (e.g., magazine articles and periodicals). Due to space constraints, we cannot cite them all, so we reference a selection of representative examples in this article.
3. The slogan was technically inaccurate. Software still existed, but SaaS upended the world in which perpetual software licenses dominated.
4. A 2007 incident exemplified this constraint, when a Salesforce employee fell victim to a phishing attack, resulting in a massive leak of confidential user data that was subsequently exploited to target over a million Salesforce users (Fiveash, 2007).
5. The details of this architecture are elaborate, and well-documented in a 16-page whitepaper published in 2008. '*The Force.com Multitenant Architecture: Understanding the Design of Salesforce.com's Internet Application Development Platform*.' 2008, whitepaper, https://www.developerforce.com/media/ForcedotcomBookLibrary/Force.com_Multitenancy_WP_101508.pdf.
6. Contemporaneous accounts suggest that Salesforce was not the only company experimenting with Web services at the time. Another CRM provider, Upshot, reportedly leveraged the same technology, linking it with applications such as Microsoft Outlook and Lotus Notes (Haverstein, 2003). Siebel acquired Upshot in 2003.

Acknowledgments

We are deeply grateful to Martha Poon for her generous support and insightful guidance, especially during the research's ideation stage. We thank the editors of this special issue, Devika Narayan and Jean-Christophe Plantin, the referees who provided insightful comments on a first version of this draft, and our colleagues Michel van Eeten, Thijmen van Gend, Agathe Balayn, Corinne Cath-Speth, Anushka Mittal, Femke Snelting, The Institute for Technology in the Public Interest (TITiPI) for the fruitful conversations that shaped the thinking which made this paper possible. We are also grateful for the valuable feedback and suggestions received from the participants of the 2025 Society for the Advancement of Socio-Economics (SASE) Conference.

Author contributions

CRediT: **Donald Jay Bertulfo:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing; **Seda Gürses:** Conceptualization, Methodology, Supervision, Writing – review & editing.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This paper is based on a broader research conducted within the Programmable Infrastructures Project at TU Delft. It is partially funded by the NWO AlgoSoc Project (NL), Delft Technology Fellowship (NL) as well as a research grant from the Botnar Foundation (CH).

Notes on contributors

Donald Jay Bertulfo is a PhD researcher at the Faculty of Technology, Policy and Management, Delft University of Technology, the Netherlands. He is a member of the Programmable

Infrastructures Project at the Department of Multi-Actor Systems, Organization and Governance Section of the said faculty. He studies the historical antecedents of agile software production in computational infrastructures (e.g., cloud and mobile/end-devices), and explores its role in (global) economic production [email: d.j.bertulfo@tudelft.nl].

Seda Gürses is currently an Associate Professor at the Faculty of Technology, Policy and Management, Delft University of Technology, the Netherlands. There, she leads the Programmable Infrastructures Project which focuses on the rise and impact of current day computational infrastructures for software production. She is trained as a computer scientist and is a member of the NWO Algosoc project, as well as The Institute for Technology in the Public Interest (TITiPI) [email: F.S.Gurses@tudelft.nl].

Data availability statement

The authors confirm that the data supporting the findings of this study are available within the article and its supplementary materials.

References

Apicella, M. (2002). Customer relationship management. *InfoWorld*.
April, C., & Harreld, H. (2002). Bringing it all together. *InfoWorld*.
Arrington, M. (2007). Salesforce enters custom application market with Force.com. *TechCrunch*. https://techcrunch.com/2007/09/13/salesforce-enters-custom-application-market-with-forcecom/
Bednarz, A. (2002). Users find hosted CRM apps hit bull's eye. *Network World*.
Benioff, M., & Adler, C. (2009). *Behind the cloud: The untold story of how Salesforce.com went from idea to billion-dollar company-and revolutionized an industry*. John Wiley & Sons.
Bowker, G. C., & Star, S. L. (1999). *Sorting things out: Classification and its consequences*. The MIT Press.
Busse, T. (1998). ERP outsourcing options grow. *InfoWorld*.
Campbell-Kelly, M. (2004). *From airline reservations to Sonic the Hedgehog: A history of the software industry*. MIT Press.
Campbell-Kelly, M., & Garcia-Swartz, D. D. (2008). The rise, fall, and resurrection of software as a service: Historical perspectives on the computer utility and software for lease on a network. In *The Internet and American business* (pp. 201–230).
Constellation Research. (2019). Fireside Chat with Salesforce's Jim Cavalieri [Video]. https://www.constellationr.com/video/fireside-chat-salesforces-jim-cavalieri
Cox, J. (1998). Why buy when you can rent apps? *Network World*.
Dickerson, C. (2002). Flexibility and control. *InfoWorld*.
Ensmenger, N. (2013). Computation, materiality, and the global environment. *IEEE Annals of the History of Computing*, *35*(3), 80–80.
Entrepreneurship at Cornell. (2023). Eclectic *Convergence* 2023: A fireside chat with Jim Cavalieri '91, SVP, Salesforce [Video recording]. https://www.youtube.com/watch?v=DNHHNYdemlk
Fiveash, K. (2007). Salesforce.com gone phishing. *The Register*.
Foucault, M., & Rabinow, P. (1991). *The Foucault reader*. Penguin Books.
Fox, P. (2000). IT services for rent. *ComputerWorld*.
Fox, P. (2003). Power in the field. *ComputerWorld*.
Garland, D. (2014). What is a "history of the present"? On Foucault's genealogies and their critical preconditions. *Punishment & Society*, *16*(4), 365–384. https://doi.org/10.1177/1462474514541711
Gilbert, A. (1999). Siebel strengthens Sales.com. *InformationWeek*.
Gillespie, T. (2010). The politics of 'platforms'. *New Media & Society*, *12*(3), 347–364. https://doi.org/10.1177/1461444809342738
Gürses, S., & van Hoboken, J. (2018). Privacy after the agile turn. In E. Selinger, J. Polonetsky, & O. Tene (Eds.), *The Cambridge Handbook of Consumer Privacy* (pp. 579–601). Cambridge University Press; Cambridge Core.

Hagendorf Follett, J., & Darrow, B. (2002). Salesforce.com, *Upshot* take CRM to enterprise level. *CRN*.
Hall, M. (2000). These CIOs mean business. *ComputerWorld*.
Hall, M. (2001). Unexpected benefits. *ComputerWorld*.
Haverstein, H. (2003). CRM crisis? ASPs save the day. *InfoWorld*.
Hogan, M., & Shepherd, T. (2015). Information ownership and materiality in an age of big data surveillance. *Journal of Information Policy*, *5*, 6–31.
Holmes, P. (2001). The launch of Salesforce.com and the end of software. *PRovoke Media*. https://www.provokemedia.com/latest/article/the-launch-of-salesforce-com-and-the-end-of-software
Huber, N. (2008). The rise of Salesforce.com. *Computer Weekly*.
Jansen, F., & Oever, N. T. (2026). More compute for a burning planet? A scarcity approach to AI infrastructures. In *AI Infrastructures and Sustainability: Expanding Perspectives on Automation, Communication and Media* (pp. 321–339). Springer Nature Switzerland.
Kaldrack, I., & Leeker, M. (2015). *There is no software, there are just services*. Meson Press.
Kawamoto, D. (2006). Salesforce.com launches AppExchange. *ZDNET*. https://www.zdnet.com/article/salesforce-com-launches-appexchange/
Krill, P. (2002). CRM foes host drum. *InfoWorld*.
Liu, F., Tong, J., Mao, J., Bohn, R. B., Messina, J. V., Badger, M. L., & Leaf, D. M. (2011). NIST cloud computing reference architecture. *NIST*. https://www.nist.gov/publications/nist-cloud-computing-reference-architecture
Luitse, D. (2024). Platform power in AI: The evolution of cloud infrastructures in the political economy of artificial intelligence. *Internet Policy Review*, *13*(2), 1–44. https://doi.org/10.14763/2024.2.1768
Magee, L., & Rossiter, N. (2015). Service orientations: Data, institutions, labor. In *There is no software, there are just services* (pp. 73–89). Meson Press.
Mann, M. (1984). The autonomous power of the state: Its origins, mechanisms and results. *European Journal of Sociology/Archives européennes de sociologie*, *25*(2), 185–213. https://doi.org/10.1017/S0003975600004239
Margulius, D. (2002). Finding the right tools. *InfoWorld*.
Margulius, D. (2003). Fitting your needs. *InfoWorld*.
Marti, E., Saloner, G., & Spence, A. M. (1999). Siebel Systems Inc. Graduate School of Business, Stanford University, Case Number EC-1.
Maselli, J. (2000a). ASPs feel the pinch of dot-com slowdown. *InformationWeek*.
Maselli, J. (2000b). ASPs gain ground. *InformationWeek*.
Maselli, J. (2001). On shaky ground. *InformationWeek*.
Maselli, J. (2002). Salesforce.com thinks bigger. *InformationWeek*.
McCarthy, J. (2002). A fundamental shift. *InfoWorld*.
McDougall, P. (2003). Service bridges teamwork gap. *InformationWeek*.
McKenzie, M. (2001). Keep track of your customers online. *PC World*.
Mears, J. (2001). Package simplifies application integration. *NetworkWorld*.
Mell, P., & Grance, T. (2011). *The NIST definition of Cloud Computing* (NIST Special Publication (SP) 800–145). National Institute of Standards and Technology.
Mosco, V. (2014). *To the cloud: Big data in a turbulent world*. Routledge.
Mukerji, C. (2009). *Impossible engineering: Technology and territoriality on the Canal du Midi*. Princeton University Press.
Murphy, V. (2001). Reinventing software. *Forbes*.
Narayan, D. (2022). Platform capitalism and cloud infrastructure: Theorizing a hyper-scalable computing regime. *Environment and Planning A: Economy and Space*, *54*(5), 911–929. https://doi.org/10.1177/0308518X221094028
Narayan, D. (2023). Monopolization and competition under platform capitalism: Analyzing transformations in the computing industry. *New Media & Society*, *25*(2), 287–306. https://doi.org/10.1177/14614448221149939
Neilson, B., & Rossiter, N. (2022). Automating labour and the spatial politics of data centre technologies. In *Topologies of Digital Work: How Digitalisation and Virtualisation Shape Working Spaces and Places* (pp. 77–101). Springer International Publishing.

Orlikowski, W. J., & Scott, S. V. (2008). Sociomateriality: Challenging the separation of technology, work and organization. *The Academy of Management Annals*, *2*(1), 433–474. https://doi.org/10.1080/19416520802211644

Plantin, J.-C., Lagoze, C., Edwards, P. N., & Sandvig, C. (2018). Infrastructure studies meet platform studies in the age of *Google* and Facebook. *New Media & Society*, *20*(1), 293–310. https://doi.org/10.1177/1461444816661553

Pollock, N., & Williams, R. (2008). *Software and organisations: The biography of the enterprise-wide system or How SAP conquered the world*. Routledge.

Poon, M. (2012). *What lenders see—A history of the Fair Isaac scorecard [PhD Dissertation]*. UC San Diego.

Randerath, S. (2024). Formatting work: Cloud platforms and the infrastructuring of capitalist asymmetries in software work. *Convergence: The International Journal of Research into New Media Technologies*, *30*(6), 2187–2211. https://doi.org/10.1177/13548565241268013

Ricadela, A. (2006). Salesforce at your service. *InformationWeek*.

Rikap, C. (2022). Enclosing the internet – big tech's cloud cover. *Social Europe*. https://socialeurope.eu/enclosing-the-internet-big-techs-cloud-cover

Roitman, J. (2023). Platform economies: Beyond the *North South* divide. *Finance and Society*, *9*(1), 1–13. https://doi.org/10.2218/finsoc.8089

Salesforce. (2003). 2003 *Form S-1*. https://www.sec.gov/Archives/edgar/data/1108524/000119312503096073/ds1.htm

Salesforce. (2025). FY2025 *Annual Report*. https://s205.q4cdn.com/626266368/files/doc_financials/2025/ar/Salesforce-FY25-Annual-Report.pdf

Saracevic, A. (1999). Siebel Systems is nations' fastest-growing tech firm. *SFGate*.

Schwartz, E. (2000). Weightless software takes off. *InfoWorld*.

Schwartz, E. (2003). CRM vendors' paths diverge. *InfoWorld*.

Songini, M. (2003). CRM vendor adds development hosting. *ComputerWorld*.

Srnicek, N. (2017). *Platform capitalism*. Polity Press. https://www.wiley.com/en-us/Platform+Capitalism-p-9781509504862

Storey, F. (2018). *An interview with one of Salesforce's founding board members (video + transcript)*. SaaStr. https://www.saastr.com/how-customers-saves-salesforce/

Sweat, J. (1999). Front-office applications by way of the web. *InformationWeek*.

Sweat, J. (2000). The new force in sales. *InformationWeek*.

Tamboukou, M. (1999). Writing genealogies: An exploration of Foucault's strategies for doing research. *Discourse: Studies in the Cultural Politics of Education*, *20*(2), 201–217. https://doi.org/10.1080/0159630990200202

Tehrani, R. (2003). Hosted versus licensed, or a little bit of both? *Customer Interaction Solutions Magazine*.

Torode, C. (2001). Specialization is gem of an idea for ASP. *CRN*.

Udell, J. (2002). Setting sale for the future. *InfoWorld*.

van der Vlist, F. N., Helmond, A., & Ferrari, F. (2024). Big AI: Cloud infrastructure dependence and the industrialisation of artificial intelligence. *Big Data & Society*, *11*(1), 1–16. https://doi.org/10.1177/20539517241232630

Varley, I. (2017). The database is a magician. *Salesforce Engineering*. https://medium.com/salesforce-engineering/the-architecture-files-ep-4-the-database-is-a-magician-b951945ea5b8

Weiss, L., & Thurbon, E. (2018). Power paradox: How the extension of US infrastructural power abroad diminishes state capacity at home. *Review of International Political Economy*, *25*(6), 779–810. https://doi.org/10.1080/09692290.2018.1486875

Weissman, C. D., & Bobrowski, S. (2009). The design of the *Force.com's* multitenant internet application development platform. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data* (pp. 889–896).

Yost, J. R. (2017). *Making IT work: A history of the computer services industry*.