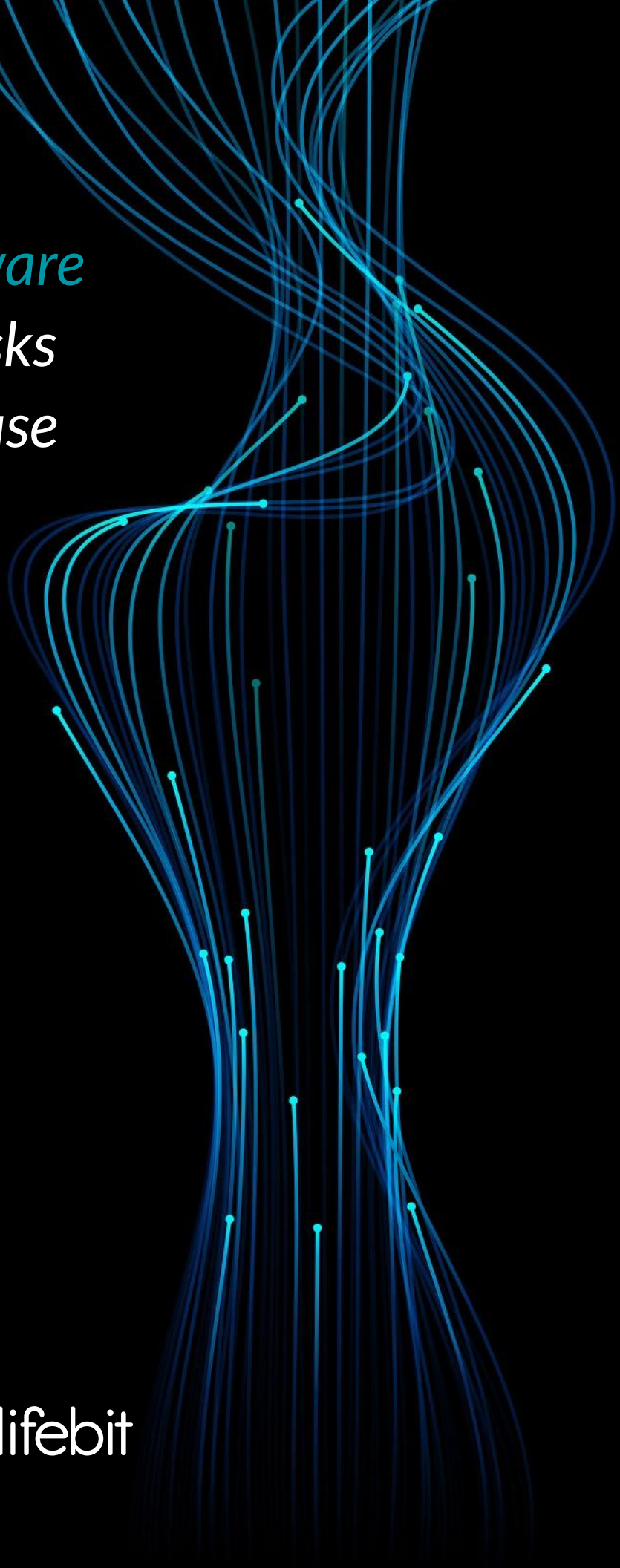


Open source software & its associated risks in organisational use

With a plethora of innovative open source software to choose from in the life sciences arena, it is no wonder that researchers turn to these community projects to access the best-in-class and the latest tools. However, for organisations looking to adopt open source software, there are a variety of risks associated in adopting this method.

In this white paper, we cover risks and challenges that are inherent to working with open source software, and highlight the pros and cons of different approaches organisations adopt to establish research platforms.

WHITE PAPER
R376/V2

lifebit

Open Source & its guiding principles

The term open source refers to releasing software or other products in a transparent manner that allows for open exchange and community development - essentially individuals can view, alter, copy and share because the source code is publicly accessible. The term originated in software development circles to designate a specific approach to designing software. Nowadays, open source initiatives and projects have evolved to embrace and celebrate principles of open exchange, collaborative participation, rapid prototyping, transparency, meritocracy and community-oriented development.

The COVID-19 pandemic & the Open Source upsurge

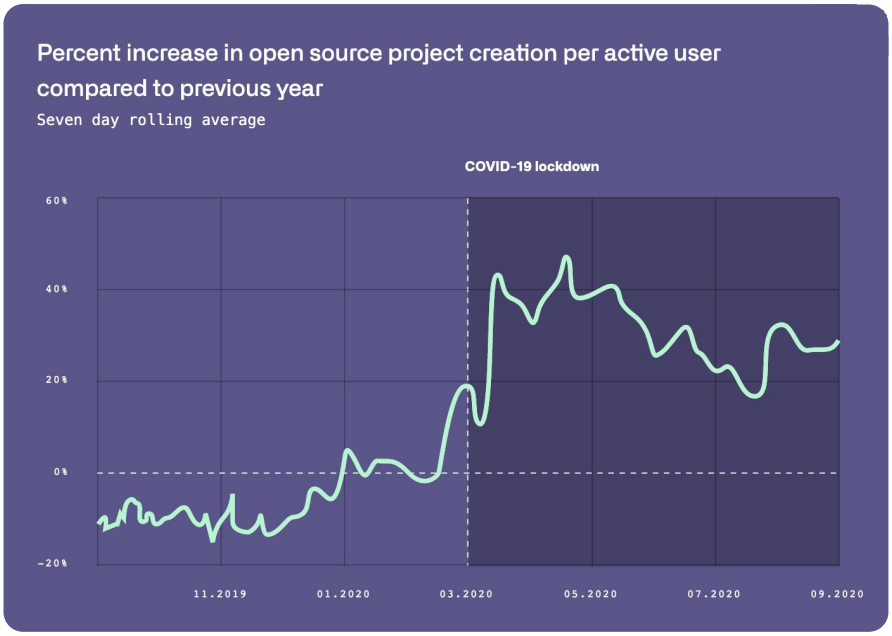


Image credit: GitHub Octoverse

No global event has had a greater impact on the free flow of information than the current COVID-19 pandemic. Although, we are still in the midst of the pandemic, open source solutions have enabled large groups of researchers, healthcare professionals, software developers and innovators across the globe to mobilise quickly and mitigate the impact of the coronavirus.

According to the world's leading code hosting platform for version control and collaboration, GitHub, 35% more repositories were created in 2020, with an estimated 25% more contribution to open source projects, than over the same time period in 2019 - with a sharp peak occurring right around the time that many countries entered national lockdowns.[1]

2020 saw explosive growth and presented an exciting opportunity for new users to engage with open source communities - over 5,646 repositories have been created solely for COVID-19.

- COVID-19 Hospital Impact Model for Epidemics (CHIME) which enables hospitals and public health officials to run models to understand hospital capacity needs as they relate to the pandemic, and
- Nextstrain which harnesses scientific and public health data to track in real-time the genomic evolution of viruses and bacteria

Even top pharmaceutical companies, which have not typically been associated with the open source movement in the past, are jumping on the bandwagon to help combat the coronavirus. Notably, Pfizer announced in its COVID-19 Response Plan that it was committed to making the COVID-19 tools they developed available on open source platforms to share with the broader research community.

Open Source in Life Sciences

Although there has been a strong focus on the open source movement in the wake of the global pandemic, the concept of open source is not new to the life sciences arena. Estimations, for instance, reveal that over 3,000 bioinformatics tools are developed on a yearly basis, with a vast majority of these developed under open source principles by researchers and laboratories.[2] These tools offer freely available source code that address a specific problem or problem domain of biological sciences in a reusable and reproducible way.

Open source bioinformatics software cover a wide range of applications from quality assessment and trimming tools such as FastQC, *de novo* sequence assemblers such as Platanus to reconstruct genomic sequences from massively parallel shotgun sequencing data, to annotation tools such as BRAKER that uses genomic and RNA-Seq data to automatically generate full gene structure annotations in novel genomes. Likewise, all major scientific workflow engines have also been developed under open source principles and are freely available to whomever is interested in using them, including Nextflow, Cromwell, WDL and Snakemake.

Challenges & risks of Open Source software

A recent Synopsys study has revealed that open source plays a significant role in today's software ecosystem - effectively all (99%) code bases audited in the study contain at least one open source component, with open source comprising 70% of the overall code.[4] However, as many researchers and organisations turn to open source as an opportunity to fine-tune and accelerate their workflows at no cost, many are unaware of the challenges and risks associated with adopting open source software.

Support & Longevity

More often than not, open source software lacks enterprise-grade support. Furthermore, not all open source software is maintained equally - some big projects such as Linux, are backed by big vendors, namely Red Hat (acquired by IBM in 2019 [3]), which ensures that disparate components are polished and updated regularly. By and large, open source software relies on a loyal and engaged online user community to deliver support via forums and blogs, which often fails to deliver the high level of response that many professional users expect. Some argue that there is no incentive for the open source community to address a user's problem. For instance, React, a JavaScript library to build user interfaces, depends on thousands of components, without anyone in command of ensuring whether they are secure or even up-to-date. There is an increasingly alarming widespread use of aging or even abandoned open source components - over 91% of codebases contain components that are either outdated by over 4 years or that have seen no development activity in the past 2 years.[4]

Security Risks & Code Vulnerability

The most concerning trends with respect to open source projects is the mounting security risks associated with unmanaged open source. Over 75% of codebases contain open source components with known security vulnerabilities, with more than half of these considered high-risk vulnerabilities.[4]

Although bigger open source projects may have full time developers tracking issues and constantly developing new functionalities, many community-driven projects are side projects which are maintained by developers in their spare time. Consequently, it is not surprising that security issues and vulnerabilities in open source initiatives may take longer to address after the occurrence of Common Vulnerability Exposures (CVEs).

Moreover, open source adopters must also be vigilant of the distribution channels. For instance, when source code is shared as a binary, a majority of IT teams will not undertake further verification than making sure that the provided hashes match the binary. However, both the binary and the hashes often come from the same source and can both be compromised by a hacker, which was what recently occurred with several Linux distribution repositories including Mint and Gentoo.[5]

In traditional commercial solutions, on the other hand, Service Level Agreements (SLAs) are signed by both parties in order to protect the clients against such occurrences. As such, it is critical that organisations making use of such software maintain an accurate inventory of third-party software components, including open source dependencies in order to address these potential risks.

Licence Conflicts

Besides support, maintenance and security issues, open source users must also comply with licences which may put intellectual property (IP) at risk as open source software is no different from any other software in that its use is governed by licences. In order to use open source software, users must accept the terms of a licence. Generally, open source licences grant users permission to use the software for any purpose they see fit. Some open source licences, however, stipulate that anyone who releases a modified open source programme must also release the source code alongside it, which can become an issue for organisations looking to protect their IP.

Governance

Open source communities are guided by high levels of trust amongst the community developers, however, the maintainers' identities are often unclear and oftentimes only have an email address as their only form of identification. Consequently, the possibility of having a developer fall victim to a malicious party impersonating and compromising their account is a very real threat.

Approaches to adopting open source software

Although the technical jargon can be intimidating at times, the differences between closed platforms, DIY solutions and open platforms are relatively straightforward and there are fairly clear pros and cons for each. The best option largely depends on individual organisations and their particular goals. The main objective, however, is the same across the board - to have access to a bioinformatics or digital research environment/platform that is easy for teams to manage on a day-to-day basis, which can look drastically different for organisations.

Approach 1: Closed Platform

Closed platforms, sometimes referred to as blackboxes, are defined as proprietary software distributed under a licensing agreement where authorised users can only essentially view inputs & outputs of their analyses. These solutions may make use of some open source components, but the majority of the source code is proprietary & thereby unavailable for modification or viewing. Consequently, the inner workings of such solutions are completely concealed from end users. As a result of their opacity, these systems:

- require users to typically hand over control of their data and research,
- lack auditability which is fundamental for reproducible research,
- lack flexibility as end users are confined to using the tools and features offered on the platform - integrations with third-party applications are usually non-existent, or at best, limited, and
- often force users to use proprietary languages and frameworks to develop their own custom tools, meaning these lack portability to other platforms

Depending on the complexity of the system, the cost for closed platforms can be quite high, especially when custom integration and services are added to the overall licensing cost. Furthermore, closed proprietary platforms significantly increase the risk of vendor lock-in which make users dependent on a specific vendor and unable to switch to another vendor without substantial switching costs or ability to port their previous resulting data, pipelines and tools.

While the hard cost tends to be higher than other approaches we review in this white paper, clients of such solutions typically experience higher levels of usability and functionality as these environments are designed with the input of experienced UX/UI designers. In addition, end users benefit from best-in-class support and ongoing training as these are typically included as part of the licensing costs. As such, users can rest assured knowing that if anything goes wrong, they will be able to access the relevant support channels to solve their problems, on demand and in real time.

Approach 2: Do It Yourself (DIY)

Instead of going the closed platform route, some organisations choose to build their digital research environment from scratch - essentially boiling down to a Do It Yourself (DIY) approach. DIY can seem appealing to organisations looking to leverage the latest innovative open source software and incorporate it into their R&D pipelines. By designing their own ecosystem, organisations have the freedom and flexibility to tailor their solution to fit their research needs.

A major area of criticism, however, is that by choosing to build their own solution using open source components, organisations become heavily reliant on the developer community for support. As we have previously covered, not all open source software is maintained equally - some projects lack regular updating and the development of new features and functionalities. Furthermore, as DIY environments are typically designed and maintained by one or two key individuals, organisations can face significant challenges around work-continuity and support if these individuals ever leave the organisation.

Furthermore, open source software can lack coding and testing standards. Consequently, it becomes the responsibility of the organisation's IT team to troubleshoot and supervise the implementation and maintenance. If, for instance, the code needs tweaking in order to fit a specific use case, the IT team will be responsible for bringing in the right modifications and debug in order to ensure a tight fit with the rest of the tech stack. As such, the variability of open source software in terms of maintenance and coding and testing standards can have a significant impact on an organisation's IT resources and on the stability, security and scalability of the final DIY solution. Although the open source components that make up the final solution are essentially free, significant costs have to be accounted for in terms of the full time employees (FTEs) setting up, developing and maintaining the environment and ecosystem over time.

Another major challenge to DIY approaches is that the usability falls short in open source software because the technology is not generally guided by UX/UI principles and is designed to cater to developers rather than to the vast majority of researchers with little-to-no programming experience. Although user guides are encouraged in open source communities, they are not enforced and are often filled with technical jargon that is extremely difficult to understand. Such a solution will therefore only be usable by a subset of the users and the organisation, and will not foster collaborative work across different user groups and different mixed teams of technical and non-technical users.

91% of codebases contain components that are outdated by 4+ years or that have seen no development activity in the past 2 years

Approach 3: Open Platform

The open platform approach, or an open ecosystem approach, can be thought of as an intermediate solution between closed platform and DIY approaches as it offers the best of both worlds. These platforms provide the reliability and support of closed platforms as these are usually distributed under a licensing agreement, while also offering end users the endless possibilities to customise their environment by enabling them to integrate their favourite innovative third-party applications, tools and data. This is performed via published external programming interfaces, also known as Application Programming Interfaces (APIs), which enable third-party integrations. Furthermore, open platforms also extend support for a variety of open source integrations and applications, which ensure minimal security risks when it comes to adopting innovative and community-driven applications.

Providing an open platform gives developers the building blocks to better fit their organisation's needs. As every client is different and is interested in using different tools and workflows for their analyses, it is important that they are given the opportunity to choose the software they want to use and be able to make it all work seamlessly together.

Open platforms provide two main benefits to end users:

- they enable users to add features and functionality they would otherwise not be able to in closed systems, and
- they help combat vendor lock-ins and data silos, which are often significant issues when using either a closed platform or DIY solution

Conclusion

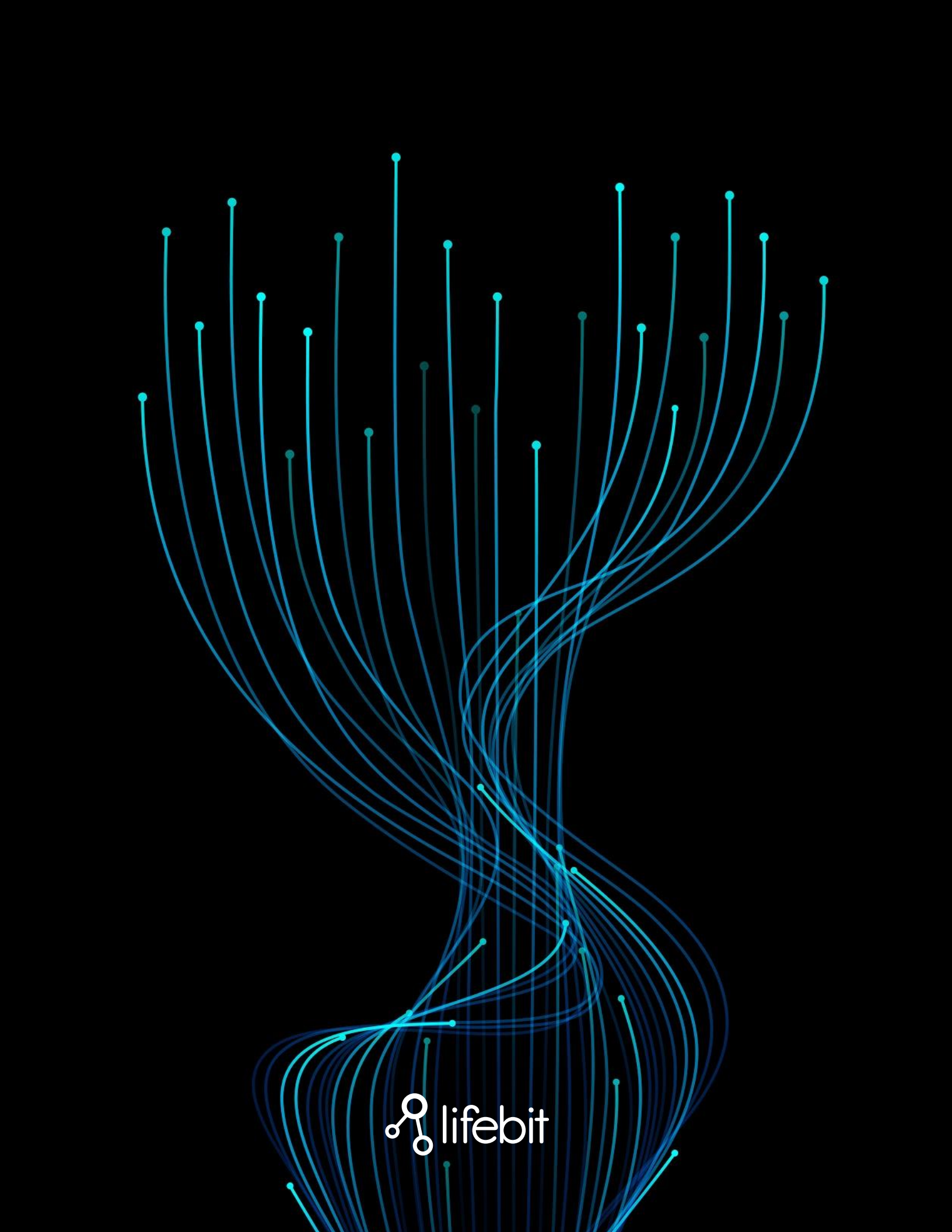
Open source software has the potential to speed up development, thereby freeing organisations to create innovative products that give them the competitive edge. To maintain that edge, however, it is critical for organisations to assess possible risks with incorporating open source components into their environment. By adopting an approach tailored to their key requirements, organisations can drastically mitigate open source risks in the future, ensuring a sustainable and stable ecosystem.

REFERENCES

1. Github (2020). The 2020 State of the Octoverse. https://octoverse.github.com/
2. Clément, L., Emeric, D., Laurent, M., David, L., Eivind, H., & Kristian, V. (2018). A data-supported history of bioinformatics tools. arXiv preprint arXiv:1807.06808.
3. IBM Closes Landmark Acquisition of Red Hat for $34 Billion (2019). https://www.redhat.com/en/about/press-releases/ibm-closes-landmark-acquisition-red-hat-34-billion-defines-open-hybrid-cloud-future
4. Synopsys (2020). Open Source Security and Risk Analysis Report
5. Another Linux community with malware woes (2018). https://nakedsecurity.sophos.com/2018/07/11/another-linux-distro-poisoned-with-malware/

Pros & Cons of different approaches to adopting open source bioinformatics software

	Pros	Cons
Closed Platform	• Best-in-class support • Increased reliability & scalability • High levels of usability & functionality	• Loss of control over data • Lack of audibility • Lack of flexibility • Increased risk of vendor lock-in • Platform infrastructure needs to be licensed • Higher costs
Do It Yourself	• Latest innovative software • Flexibility • Ability to fully customise environment	• Heavy reliance on open source community for support • Customisation, troubleshooting, debugging and maintenance falls back on the organisation's IT team • Significant costs in terms of FTEs for maintenance • Decreased usability for individuals with little programming experience • Decreased stability & scalability • Increased security vulnerabilities and risk of security threats
Open Platform	• Latest innovative software • Flexibility • Ability to fully customise environment via APIs for third-party integrations • Extended support for open source components • Best-in-class support • Increased reliability & scalability • High levels of usability & functionality • Increased collaboration • Control over data & combat data silos • Improved auditability • No risk of vendor lock-in	• Core platform infrastructure needs to be licensed


lifebit