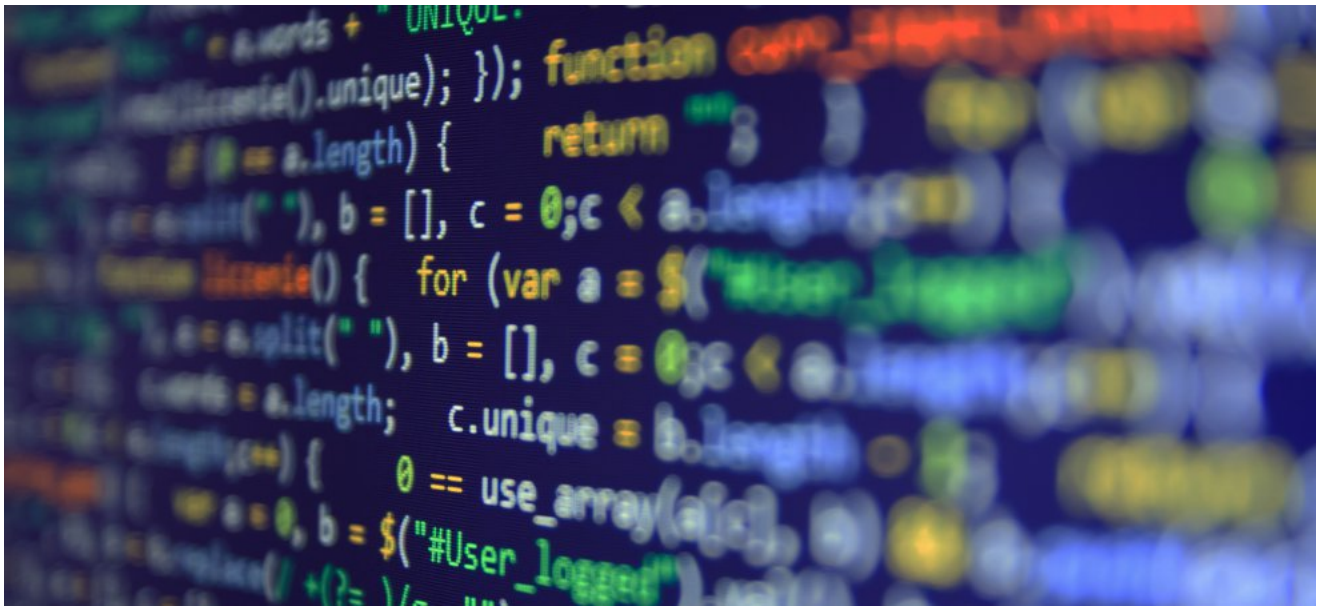
NEW AMERICA



July 2020

Building and Reusing Open Source Tools for Government

Software for Public Benefit Should be Open Source by Default

Mark Lerner, Allison Price, Hana Schank, & Ben Gregori

Last edited on July 10, 2020 at 1:08 p.m. EDT

Acknowledgments

This research and report would not have been possible without the insights and perspectives of a wide range of individuals and partners, all of whom are passionate about open source government solutions. Internally at New America, we benefited from the guidance, insights, and editing of Cecilia Muñoz, Tomicah Tillemann, Dahna Goldstein, and Karen Bannan. We would especially like to thank the Rockefeller Foundation for supporting our work and serving as a thought leader on digital transformation.

Interviewees

Marianne Bellotti, Rebellion, formerly U.S. Digital Service, Auth0

Sebastian Benthall, Research Fellow at the Information Law Institute at New York University

Sean Boots, Technical Advisor for Policy at the Canadian Digital Service

Alex Gaynor, Chief Information Security Officer of Alloy, formerly U.S. Digital Service, emeritus member of the Python Software Foundation Board of Directors

Marc Jones, General Counsel at CivicActions, former Counsel at Software Freedom Law Center

Jeff Maher, Head of Software Development at the Canadian Digital Service, formerly Ash Center Technology and Democracy Fellow at Harvard Kennedy School

Giuseppe Morgana, Digital Director at the New Jersey Office of Innovation, formerly U.S. Digital Service

Kevin O'Neil, Director, Data & Technology, The Rockefeller Foundation

Chukwudi Onike, Senior Associate, Data & Technology, The Rockefeller Foundation

Denis Pitcher, Advisor to the Premier of Bermuda on FinTech

Josh Ruihley, Senior Technical Advisor at the Canadian Digital Service, formerly Director of Product and Custom Partner Solutions at 18F

Adrienne Schmoeker, Director of Civic Engagement and Strategy, and Deputy Chief Analytics Officer for the City of New York

Sherri Trivedi, Director of Design at U.S. Digital Service, formerly GitHub

Cori Zarek, Director of Data + Digital at the Georgetown Beeck Center for Social Impact + Innovation, former U.S. Deputy Chief Technology Officer

Inspiration

Many organizations and people paved the way on this hard work, making it simpler for us to collect the information and produce this resource. While there are far too many to name, we wanted to particularly provide thanks to these people and organizations for laying the foundation.

- The UK's Government Digital Service
- 18F
- The Canadian Digital Service
- The Open Source Institute
- GitHub
- The Consumer Finance Protection Bureau

About the Author(s)

Mark Lerner is a fellow in New America's Digital Impact and Governance Initiative and Public Interest Technology program. Lerner is an engineer, strategist, and design advocate with expertise in digital transformation. He focuses on empowering teams and improving critical services through technology and design.

Allison Price is a senior advisor with the Digital Impact and Governance Initiative (DIGI) and the executive director of Blockchain Trust Accelerator (BTA). The DIGI team at New America is developing the next generation of technology platforms to transform the way governments deliver value for citizens. The BTA is committed to advancing blockchain technology through research and innovative pilot projects designed to address some of the world's most persistent challenges like transparency, identity and corruption.

Hana Schank is the Director of Strategy for Public Interest Technology at New America, where she works to develop the public interest technology field via research, storytelling and fostering connections. She founded and edits The Commons, a publication for people working in and around government innovation efforts.

Ben Gregori is a policy analyst for the Digital Impact and Governance Initiative (DIGI) and the Blockchain Trust Accelerator (BTA) programs at New America.

About New America

We are dedicated to renewing the promise of America by continuing the quest to realize our nation's highest ideals, honestly confronting the challenges caused by rapid technological and social change, and seizing the opportunities those changes create.

About Digital Impact and Governance Initiative

The Digital Governance and Impact Initiative (DIGI) develops technology platforms that transform the way institutions deliver value for citizens. We work with partners in government and the private sector to create modular, interoperable technology solutions built on open source code that address key challenges facing the public sector.

About Public Interest Technology

New America's Public Interest Technology team connects technologists to public interest organizations. We aim to improve services to vulnerable communities and strengthen local organizations that serve them.

Contents

Contents Cont'd

Overview

In this report, we outline a high-level framework for approaching the development of public services using open source software (OSS) principles and practices to maximize the reuse of existing software. Open source principles are an integral part of well-designed public technologies. Software for public benefit, such as election platforms or benefit case management systems, should always be open by default.

This report defines OSS, the benefits and risks associated with it, and provides steps for using open source principles in your organization or agency. While there are many important principles for building good software products, such as user feedback, sustainability, and open data, we will be focusing on the value of OSS in this resource. Open source software is still software, and while we won't be going into all of the components of good software development in this report, it is important to learn from and follow known software development best practices. For example, approaches such as agile development define clear goals, build iteratively, and design with users in mind. We find **GOV.UK** and the **U.S. Digital Services Playbook** to be great starting points for understanding best practices for creating government digital services.

Who Should Read this Report

This resource is intended for public servants, from executives to program managers, who are building the next iteration of digital government solutions. We initially undertook this research with the goal of providing useful information for practitioners, and we understand that there are many different types of practitioners in the government technology ecosystem.

We've outlined below how we anticipate different audiences can best harness this content.

- **Government Leaders:** We aim to inspire government leaders to think differently about technology, and to provide them with a roadmap for how to bring OSS into their country, city, county, state, and agency software and services development lifecycle. This report provides foundational knowledge that executive decision makers can tap when setting organizational strategy.

- **Technology Leaders:** Directed by the strategy and policy set forth by organizational leadership, technology implementers translate desired public outcomes into technical products. They enact the agenda using technology, aiming for on-time, on-budget, and effective delivery of

services. This report outlines how to make open source a core part of the implementation of government services, and provides arguments for why OSS is the best choice for government services.

- **Innovation Officers:** Innovation officers fall into a special category of government leaders, as they tend to be highly technical and hold positions of authority in their organization. We look to innovation officers as a driving force, a catalyst to lead colleagues along government services modernization efforts. This report provides this cohort with the research needed to support this work, as well as actionable recommendations for their efforts.

- **Decision makers including government officials (e.g. legal, procurement, security):** OSS presents a very different operating model than what government is traditionally used to. Software is built or procured at no cost, and ownership is less important than license provisions since open source licenses vary in their permissiveness. This cohort of officials have a duty to reduce the risk of government projects. This report provides information regarding security, licenses, ownership, and costs, arming these approving government officials with the knowledge they need to confidently assess open source at its true value, rather than as an unknown entity.

Software for public benefit, such as election platforms or benefit case management systems, should always be open by default.

The Structure of this Report

The research is broken into four main sections. While we encourage you to read the entire report, we understand that you might want to review the sections that pertain most to your needs.

Section One: An overview of open source

This section is a primer on open source in government, and answers some key high-level questions that many newcomers may have. All readers should read this section as it will help grow their baseline understanding of open source, and

correct some common misconceptions about the feasibility of government open source solutions.

Section Two: Building open source software

In this section, we provide guidance for organizations looking to build their own open source systems and outline considerations to keep in mind.

Section Three: Using open source software

An outline of recommendations on how to best use existing open source software. In this section, we cover topics ranging from finding open source solutions to understanding the licenses of open source software that you might use. Readers looking to find open source tools to bring into their organization should read this section.

Section Four: Managing the details

We discuss some of the details that should be kept in mind when moving your organization into a more holistic open source strategy. These details are common regardless of whether you build new software, or use existing OSS. Topics such as open source implementation policy, in-house technical talent, and copyright fall into this section.

Section One: An Overview of Open Source

What is Open Source Software?

Open source software (OSS), by definition, is "software that is readily available with its source code and license, free of cost to anyone who wants to study, change, modify, or distribute it," as per the **World Bank's Open Source for Global Public Goods report**. For people unfamiliar with the software development world, OSS is essentially chunks of code that are available online for anyone to use. While many non-coders may think software is written from scratch, in fact, much of the modern development process involves piecing together these blocks of code to create new applications. For example, have you ever had to reset your password to access an app? It is a common use case. If a developer needs to add that functionality to a new project, the developer can easily take existing code for resetting passwords and incorporate it into the new project rather than writing that code from scratch.

All kinds of software can be open source, from operating systems and under-the-hood utilities to web browsers and user-facing websites. A **2020 report by Synopsys** notes that the majority of the world's software is open source, and that nearly all software has some open source component to it. Chances are high that your organization is already using OSS, either directly or through other software that depends on it.

At a bare minimum, OSS is simply releasing the software's source code publicly. One version of this is called coding in the open—publishing the source code of a system or application without taking care to ensure its reusability. While still valuable, the benefits of coding in the open fall short of what's possible with OSS. A more robust use of the term includes providing an open license that permits broad use of software, documentation, and guidance so it can be used by others, and encouraging community engagement to facilitate organized collaboration and solve common issues.

Indeed, when fully implemented, OSS can lead to collaboration, innovation, cost savings, and sustainability—all by building software openly and for reuse. Open source aspires to a new way of thinking about ownership and accountability, something built by and responsive to the collective of users rather than purely traditional market mechanisms. Adopting open source practices as an organization means moving an organization's culture away from one of proprietary holdings and closed ownership, and towards collaboration and working in the open.

OSS is released with specific licenses that outline the permitted usage of its source code. The licenses outline terms of use, such as requiring attribution of

the original authors, that must be followed to use the software. When using open source tools, it is important to review the license to ensure you abide by its allowances and restrictions. When releasing your own OSS, it's necessary to provide a license that makes clear what your resources can be used for. Common licenses are reused across many open source projects, such as the **MIT license** or the **GPL license**. The Open Source Initiative maintains a **strongly vetted list of common open source licenses** used by many in the software industry. By using common licenses, projects reduce the burden of understanding unique license terms and provide clear and well-understood terms to the use of their resources.

Why Use Open Source?

The benefits of using open source principles is well understood and documented by the software community around the world. Anna Shipman, former open source lead with the United Kingdom's Government Digital Service, **articulated an excellent set of benefits** for open source work. Sean Boots and Josh Ruihley of the Canadian Digital Service expand on these benefits in **their own analysis of the benefits of open source**. We have provided a brief summary of benefits here for those who are new to open source.

Maximizes Resources and Improves Efficiencies

Maximizing existing resources and pursuing cost-effective solutions is an optimal strategy in government. Many civil servants are asked to do more with less. Equally true is the fact that most problems that civil servants encounter have been solved before, likely by a different office or a different jurisdiction altogether. Building on and reusing other organizations' work can decrease opportunity costs and allows teams to get started making changes more quickly.

When starting a new project, significant effort must be spent on discovery, user needs and wants, development and testing, including security and usability. Using open source code reduces the time and cost of this process and can be a far more efficient route. Reusing existing code and processes can improve the speed of development and deployment, reduce the development costs, and allow for best practices. For example, there may be fewer vulnerabilities and bugs in an application since the source code has been worked on by many people, all of whom have the opportunity to point out and address these issues. These benefits all build upon one another as well—the more your organization uses open source solutions, the more benefits you see from it.

Open source software is free to use. You can copy and use the source code without payment to the original creators, and use it in whatever way you would like (consistent with the software license). Proprietary software often comes with

license fees, contract agreements, and may be packaged as a bundle, which means you may pay for tools and features you don't need.

However, this doesn't mean that there are zero costs associated with using open source software. While the code itself is free and open, there are other types of cost considerations to make, such as server infrastructure or technical talent to modify the source code and paid support options. Still, these costs are often much less than the typical license fees associated with proprietary software. In addition, most of these costs are a constant regardless of whether you are using open source tools, building your own software, or using proprietary solutions.

Improves Trust and Increases Government Transparency

The benefit of transparency goes beyond creating accountability—it's also a core part of working on public services. The public should be able to see the work that their government is doing for their benefit. Transparency makes it clear that progress is being made on important services and issues, and keeps the public informed on future plans. Open sourced work encourages active feedback and participation from the public, who can now view the progress as it happens and hold more confidence and trust in the work. As the **Department of the Interior** puts it, "By using open software and working in the open, you remove barriers to critical evaluation and participation from others. Inviting critique from others can be uncomfortable, but it increases the likelihood that the final product is more effective at meeting the needs of multiple stakeholders inside and outside government."

The public should be able to see the work that their government is doing for their benefit.

When using OSS, either by building it or using other OSS applications, it's easier to explain how the tools work. Nothing is proprietary or hidden. Instead, the code is accessible and available for review by anyone. This makes discovery of efforts within and across government much easier. This is important because it's fairly common for a government organization to have duplicate or competing projects simply because they didn't know of each other's existence. The transparency provided by open source helps governments discover tools that are already vetted by other organizations including their own government organization, and allows them to quickly and cost-effectively adapt those solutions to their immediate challenges.

More Sustainable Tools, Less Vendor-Dependence

Using OSS greatly improves the flexibility of an organization by allowing the use of multiple vendors and giving developers the ability to share software with others in the organization, and use or reuse as much of the software as needed. OSS guarantees that an organization and the public have full access to the source code without needing any permissions from the vendor.

Regardless of whether you open source your own tools or use existing open source tools, you are still ultimately responsible for the service. Using open source software does not put the onus on other contributors—as the service owner, that still falls on your shoulders. For example, you still hold the responsibility of ensuring security and meeting users' needs. However, using open source tools makes each of these easier to achieve, and enables a community of like-minded contributors to support each other in achieving these goals. Open source software also offers flexibility in moving between tools that proprietary services seldom offer.

Grants Full Control of the Software

Much of the software that governments and people rely on is closed source and proprietary. It is owned by its developer. When there are bugs or other issues within the software, the developer is the only one that can fix it. Similarly, if software needs to be upgraded, integrated, or changed, only the company that owns the software can make that change. With OSS, any organization—including software users—can fix or change the software. OSS has roots in a philosophical belief that software should be free and open for all to use and learn from, as described in **this** blog. By using OSS, governments can maintain control over their software, tools, infrastructure, and services.

Encourages Good Development Practices

Open sourcing work encourages good development practices by creating public accountability. **Put simply, if everyone can see something, then it has to be good enough to release to the public right from the start.** The quality of the work, its documentation, code reviews, version control, and other factors are all held to a high standard, encouraging good software development practices. Open source also brings critical considerations, such as security, software design, documentation, processes, and even ownership, immediately to the forefront.

Simplifies Collaboration and Facilitates Innovation

Collaboration of all kinds is simplified when work is done in the open. When software is openly available, there are no required special approvals, agreements, or tools for code to be shared. When the software is well documented and supported by a community, it's simple for new collaborators to start using it and contributing to it. New team members are able to find all project

information since it is available openly. Developers can access and perform work using their tools of choice, rather than proprietary tools. In the current closed-source state, two different teams at the same government agency might require weeks or even months of time to secure the approvals necessary to share work. **With open source software, collaboration can start immediately.** Inviting people to collaborate on open source projects is much simpler, as the work to open it up broadly has already been done. This is especially helpful in times calling for rapid response, such as during a public crisis, when governments can tap into existing software and communities to build solutions that are already open sourced to the public.

Fosters growth of common solutions

Open source collaboration is a learning experience that allows both authors and contributors to exchange ideas about how each developer might solve a specific problem. Even when software has been developed, contributors may apply the software in a different way than the author had originally intended, teaching everyone something new in the process. Collaborators can use open source work as a starting point in solving their own versions of similar problems, saving valuable time and growing the original base of work. For example, the United Kingdom's Government Digital Service **released their Notify platform as open source**, allowing other governments, **such as the government of Canada**, to adopt it for its own purposes by modifying it to support multiple languages. Additionally, open source solutions reduce duplicative work by allowing groups to collaborate on solving common problems with common solutions, rather than creating competing or diverging solutions. By making the work open source, organizations create the opportunity for creative expansion on top of what has already been done.

Inviting critique from others can be uncomfortable, but it increases the likelihood that the final product is more effective at meeting the needs of multiple stakeholders inside and outside government.

Five Paths to Open Source Software in Government

Broadly speaking, there are five paths towards adopting OSS in government. These paths differ largely because of two factors: whether a solution is new or

existing, and whether organizations use a solution they created or one created by an outside entity. Here, we provide brief descriptions of each path, so that you might recognize which one may apply to you. As you read through sections two and three of this report, you'll find that having your path in mind will help you understand how our research applies to you.

1. Working in the Open on a New Solution

On this path, agencies and organizations create new software addressing an unmet need. The software, part of a broader solution, is created specifically for the purposes of a solution. The developers might use a vendor's services to develop the software, having procured their services through a contract or grant, or may utilize in-house technical talent to write the software. Alternatively, they may have a partnership with a third party, such as an academic institution or philanthropic organization. While existing software may be integrated with the project, its main components require software to be written from scratch by developers or software engineers.

2. Migrating an Existing Solution into the Open

Governments use existing software solutions for their services and processes. In some cases, this software is old and unmaintained, though it still functions. Many of these systems are called legacy systems, which simply means software systems that are not being actively developed. You and your organization can move these applications and services into the open, using principles and practices discussed throughout this report. Legacy solutions will require review and some amount of refactoring, to open up the source code. This review and refactoring can have an added benefit of helping to make the software more secure, understandable, and useful—both for the public and your organization. Updating and refactoring legacy systems may not always be possible for a variety of reasons, from fragility of the underlying code and lack of backwards compatibility, to missing relevant expertise in a given programming language. When it is possible, it will take work, but the benefits, including the ability to increase security, collaboration, accountability, and transparency, are worth it.

3. Adapting an Open Source Solution to Meet Your Needs

Many of the standard problems that organizations have, such as inventory management or case management, can be solved by open source solutions, but they may still need customization to match your organization's workflow, language, or needs. For example, open source case management systems can be easily configured to handle the particular types of cases your office processes. When using this kind of software, you need less direct software engineering talent than the first two paths, and instead should look for talent (either in-house or contracted) that can support the configuration and maintenance of the software.

4. Using an Open Source Solution without Customization

In some cases, you can find an open source solution that solves your problem directly, without any need for customization. While it might seem rare, consider that Firefox, one of the most popular web browsers, is open source and requires no customization to use. Open source software like Firefox is high-quality, easy to use, and doesn't require that much technical or development knowledge. In these cases, the fact that the software is open source may not even be apparent to the users even as your organization gains the benefits of the security, feature updates, and larger community that the open source software provides.

5. Using Mixed-Source Software

Lastly, your organization might be using a mix of OSS and proprietary software as a way to ease into open source principles. This path presents challenges, as it can be unclear in vendor contracts what software might be required to be open source and what might be public. Integration can often be difficult, too, since interoperability can vary with custom build solutions. As much as possible, we recommend avoiding this path and using open source software exclusively. When this is unavoidable, clearly articulate the expectation of what software will be proprietary and why, and require that all other software be made open source.

Common Concerns and Questions

The thought of using open source software instead of a commercial product often elicits numerous questions and concerns. A few of the most common questions about OSS include:

Q: Is Open Source Software Secure?

A: There is a common misconception that OSS is inherently less secure than proprietary software, due to the source code being publicly available. The logic for this is that if a malicious attacker has access to the source code, they can find and exploit vulnerabilities. This argument relies on the assumption that a closed-source piece of software is more secure simply because people cannot read the source code.

However, the consensus in the security space is that open source code provides an opportunity for better security practices over closed-source code. Open source code can be reviewed by collaborative and independent security experts for vulnerabilities, and these same reviewers can contribute fixes or work directly with the authors to implement solutions to these vulnerabilities. Independent and open code review makes solutions stronger and more secure. By contrast, relying on obscuring code for the sake of security actually introduces more security risks, as obscured code cannot be assessed by third parties and there may be fewer people using it as a whole. This comes into play often in a

government setting where custom software was developed for exclusive use by the organization. The **Department of Defense** states that studies in software vulnerabilities "make it clear that merely hiding source code does not counter attacks." The **Department of Homeland Security found in their research** that, "there was not an increased risk of low quality and malware in OSS compared to proprietary software." The **United Kingdom government's guidance** states that by "keeping your code closed, you can also introduce other risks and complications including deprioritisation of code review, [and] additional costs and complexity of access controls."

Independent and open code review makes solutions stronger and more secure.

Obscure code also allows vulnerabilities to exist in the codebase in an unknown or undisclosed state. Additionally, attackers do not need access to the source code to take advantage of a vulnerability. Opening the source code allows collaborative security partners to support in protecting against these attackers. In short, when software relies on obscurity to enforce security, it assumes that potential attackers don't have other means to find and access vulnerabilities—which they regularly do, especially when the **most common attack vectors** have nothing to do with source code anyway

The most significant recommendation of the open source movement may be the fact that many of the top security tools on the market are open source as well, such as **HashiCorp**. These projects use more diligent security practices, including rapid security patches, automated vulnerability detection, and separation of sensitive information to stay ahead of security vulnerabilities. **By building these security tools as open source, the developers have created tools that are often more transparent, accountable, and secure than proprietary, closed-source tools.**

Q: What are Open Source Licenses?

A: The OSS license ecosystem is robust, mature, and easily understood. Many resources exist to support navigating licenses, such as the **Open Source Initiative's list of licenses** or **GitHub's choosealicense.com**. Additionally, there are technical legal experts that can assist with the interpretation of common and uncommon licenses. With proper consideration and incorporation of legal guidance, the risk of violating a license is small. Navigating this terrain

generally amounts to determining whether or not attribution must be given, how your current systems and software are licensed, and how your licenses will be affected when integrated with OSS licenses you use. GitHub provides a great guide for understanding some basics of **the legal side of open source**.

Q: Is Open Source Software Kept Up-To-Date?

A: Just as with any software project, OSS varies in degree of maintenance. In some cases, a project may be years old and not updated since it was launched. (However, this doesn't necessarily mean that the software doesn't work). In other cases, a project is actively maintained, often with a vibrant and well-organized community supporting it. When using OSS, look for software that is actively updated with security patches as well as releases of new features. Software that hasn't been updated for months or years likely will not receive attention when it needs an update, such as when a security vulnerability is uncovered. Some well-resourced open source organizations, such as the Apache Foundation or the Linux Foundation, might themselves maintain open source projects, and applications that come out of these groups are typically safer to use. In addition, OSS that garners user groups, discussions, and communities should have longevity and get regular updates.

Q: Will Using Open Source Reveal Policies Prematurely?

A: Code that implements yet-to-be-released policy should remain closed until the policy is public so software implementation details don't potentially complicate the release of the policy. Government policies tend to follow particular paths to being released, and openly publishing the software that implements a new policy could give the public an incomplete or incorrect view of it. Once the policy is public, the code supporting it should be immediately made public as well, to demonstrate how the policy is being implemented. Afterwards, all further code should be developed in the open. Developers working on this type of software should coordinate closely with the release of the policy to ensure proper timing of the software release schedule. GOV.UK has a detailed resource for what should remain closed in these cases, which can be found **here**.

Section Two: Building Open Source Software

Depending on the size, mission, and complexity of your government organization, you likely use custom-built software systems. Maybe your office runs a custom database and processing system for a benefit application, inventory and supply chain system, or public-facing website or mobile app. In this section, we advocate for and provide concrete guidance on writing your software in the open, as an open source system. By opening your source code to the public, providing clear documentation, and allowing others to reuse it for their needs, you can instill best practices, receive collaborative support on your systems, and share your knowledge with other jurisdictions.

Working in the Open

When we talk about "working in the open," we specifically mean publicly publishing your work on software projects, including existing drafts, future progress, and other work products. By definition, this is different from publishing a single finalized version at a point late in the development, or once it's determined to be complete. **Working in the open means performing the actual work—the individual code changes, the code reviews, discussions, project management, and more—in the open, for anyone to see.** This type of process has many benefits, such improving transparency, accountability, and collaboration. With the exception of a small number of cases (e.g. building software for a soon-to-be-released policy), software for the public benefit should always be open source.

Through our research we found many lists of good practices for working in the open, such as **this list produced by GOV.UK**. Here, we present a select overview of good practices to follow when working in the open, as well as key considerations.

Publicly Show Progress

When working in the open, all progress should be public. This includes progressive and iterative changes to the code, as well as discussions about changes, evolutions to documentation, and potential security flaws and bugs.

The source of truth for the software should be a publicly-available code repository, hosted on a service such as **GitHub**. These kinds of services allow simple online access to the code repository, and generally support a wide range of other features, such as access control management, automated security scans, and more. For example, with access control features, you can ensure that only approved developers are able to submit code changes to the codebase, or to approve proposed changes from third parties. Changes to the code should be

made with small, publicly posted modifications (or "commits") to that codebase, which should be reviewed by an owner or maintainer of the codebase during a code review, which should also be public.

By publicly showing progress, the software development team can ensure full transparency, build a historical record of modifications and decisions, and allow others to learn from the progression.

Opening More than Code

As the name implies, OSS's source code must be publicly available. But stopping there would fall short of following open source principles. **Teams should publicly share their project work and documentation.** Project plans and roadmaps should be made available so that contributors and the public can see the long-term plan for the software. Research used to design the service should be made available to increase transparency on the decisions for the service. Even more granular project documentation, such as user stories, testing, and project discussions can be made public to provide contributors the information they need to understand how they can best support the software development and maintenance. For good examples of this, see **CA.gov's**, **New Jersey's Ask a Scientist repository**, and **Canada's VAC Find Benefits and Services documentation**.

Separate the General Solution

For your software to be most useful to others, it needs to be designed to solve a generalized problem. Designing your software in this way means structuring the architecture of the code so it can easily be modified for other uses. Keep the parts of the code that are specific to your context separate from the portions of the software that can address the general problem that other jurisdictions may face. For example, if you have software that sends text reminders to certain city staff, separate out the messages the software sends from its general ability to send messages. Developing your software in this way allows other people and organizations to take the code and reuse it for their purposes easily, without having to make significant modifications.

Managing Sensitive Information

Most software systems contain sensitive information that should not be accessible to the public. Even in closed-source systems, this information should be separated from the codebase as a best practice to ensure the security of the information. Developing in the open more heavily incentivizes this essential security practice.

User data, also frequently referred to as personally identifiable information (PII), should never be included in the source code of any application, especially open source applications. However, sensitive information is more than just user data.

Sensitive information also includes passwords, credentials, keys, and certificates—effectively, any information that would give data access to unapproved parties. This type of information is generally referred to as "secrets," and there are many best practices to follow to manage these secrets. Software should never have passwords written directly into the code. Instead, the best practice is to keep secrets separate from the source code in tools built for this purpose (such as **HashiCorp**), with the running application to be given access to this information only when necessary. In this way, even when the source code is public, the public cannot access sensitive information. By keeping secrets and user data out of your source code, you easily and drastically reduce any risk of data exposure when using open source software.

By keeping secrets and user data out of your source code, you easily and drastically reduce any risk of data exposure when using open source software.

This simple best practice maintains a separation between the public source code and the private secrets. By separating these from each other, you can publicly open your source code without any worry of undue access to sensitive data or systems. When opening source code publicly, take care to ensure that this sensitive information is not included in the project's version history. If it is, there are two ways to resolve this:

1. Change all of the secret information used. Reset these secrets, such as passwords or credentials, so that old versions no longer allow access to key data. Do not include these new secrets in the source code, but instead separate them out. This fully ensures that the risk has been mitigated, as long as the new secret information is not made public. This is the simplest and most surefire way to secure your system after secret information has been added to the codebase.

2. Remove all mentions of the secret information from the codebase, *including from the historical record of the codebase*. Version control systems (which are highly recommended) store the full history of the codebase, and simply removing a password from the code will not prevent people from looking into the history to find it. This can be an arduous but critical task. If even one instance is forgotten, it can lead to vulnerabilities and breaches.

Utilizing Open Source Communities

While working in the open is a critical part of open source software, **creating a truly open source solution also includes building and managing a community.** This community includes contributors, partners, maintainers, and anyone copying the code and repurposing it for their use. Building and managing a community takes work, but it provides an opportunity for expanded support and collaboration. Aside from transparency and distributed lessons, a healthy community may contribute to the source code, improving its quality or creating entirely new features. For example, Mozilla developed a broad community in their **open source form generating tool**, which the **U.S. Digital Service modified and repurposed to create the U.S. Forms System**.

Creating a community around an open source government tool ensures that governments from anywhere can engage with the software and find support to help them adapt it to their needs. When you open source your own projects, you openly invite other people and organizations to use and contribute to your source code. To the extent that people outside of your organization contribute to the project, they do so as coordinated volunteers. You still ultimately own and maintain the system, and while there is not any expectation that public contributors will be compensated for contributions, it does take time and effort to create an environment where a community can thrive.

Building and managing a community takes work, but it provides an opportunity for expanded support and collaboration.

The most critical portion of building a community is understanding what you, as the owner and maintainer of the software, want from the community. For example, you may want the public to see your work for accountability, but may not want code contributions. Alternatively, you may be looking to offer your code to other jurisdictions with similar problems. Whatever the reason may be, making a decision up-front will greatly inform how you actually go about building and managing the community around your open source software, including how you inform others of your work.

As you start building and engaging with a community around your project, you will be publicly attached to the code—which is a good thing! It further creates direct transparency and accountability, and allows the public to recognize you

and your team as the maintainers and experts for this product. It also improves your ability to publicly represent the project by providing a clear connection between the public and the maintainers of the service. Structures should be put in place to manage that interaction between the public and the maintainers, and it should provide the public with a sense of community engagement. By building an effective community, you give the public a place to engage with services they depend on.

Building a Community

Building a community of contributors and enthusiasts incorporates practices that relate to building communities of any kind. People need to be able to find your community and understand what it is and how they can benefit from the code or application. People who join should be able to clearly understand the software or service you are building, and whether it might be useful to them. This clarity will allow the public to self-select whether to join the community, or move on. **GitHub provides a great guide to building a community** that we recommend. Below, we provide select considerations for building an open source community.

Make your work visible and discoverable. This step applies regardless of whether you're looking to add contributors to a project, or just to share the project with other jurisdictions. Some marketing is needed to let your targeted audience know about your open source project. As with any communication and marketing, determine your target audience first and find ways to inform them. This can be via direct contact (e.g. emails, posts in LinkedIn groups or other software or development communities like GitHub and Stack Overflow) or indirectly (e.g. gaining placement in articles in news sources or on social media platforms). One of the most difficult things to navigate in the open source ecosystem is the discoverability of projects or finding a project that fits your needs. As the producer of an open source project, you'll need to put effort into making your project easier to discover by tagging it on open source code hosting services, publishing blog posts about it, and promoting it via your social and personal networks.

If you want your community to contribute and support the codebase, provide them with opportunities to engage once they join. For example, **let the community know how to stay up-to-date with the project,** whether through an email list, a public discussion forum, or following a Twitter account. Inform them about the current state of the project, and any upcoming releases. Be clear about how the project will be maintained and updated, and how they, as contributors, can take part in it. For a great example of building an inclusive code repository, **see Microsoft's GitHub repository for their application** .

Set expectations upfront on how the project will be run. Clearly articulate the various processes and statuses for the project. The public should be able to

see how often the project will be updated, what processes are used to manage contributions, what communication channels you will use, and even expected timelines for the project's lifecycle. This type of communication and clarity will provide contributors and the public everything they need to know to be able to understand and effectively engage with the project.

There should never be an expectation that the community will complete work that you need for your product. Open source communities are not sources of free labor. Instead, **the community can provide insight and updates to the codebase at their own pace**, which will be proportional to how easily they can engage with the project.

Prioritizing Documentation

Clear documentation is key to building an effective open source project. Good documentation lays out the purpose and plans of the project in plain language, and offers guidelines for engagement, contributions, licenses, and other necessary elements of being a member of the community. It also outlines the source code itself, writing in prose the structure and organization of the software. With good documentation, your project will be easily understood by anyone who comes across it.

Documentation must be simple enough that entry-level contributors can understand the project and how to get involved. Detail the expected roadmap, so that contributors and users can see the plans for future modifications and improvements. Other forms of project documentation, such as research or short-term project progress, should be available for contributors and the public to see as well. For good references on projects that have prioritized documentation, see the **Atom repository** and the previously mentioned **VSCode**.

It is critical to have documentation that explains the source code itself. In most cases, this is documentation provided for contributors, maintainers, or the public to refer to when using the software, such as integration general usage instructions. Documents that describe the design of the source code are also helpful for people looking to understand and reuse the code itself. High-level documentation should be included to describe the structure and architecture of the code. More fine-grained documentation should be in-line with the code itself.

Lastly, documentation on how the source code project is run should be available as well. For example, documentation should clearly explain how code reviews are performed on contributions, how third-party contributors can become maintainers, what coding practices are expected of contributors, and how members of the community are expected to conduct themselves. These forms of documentation provide clarity, stability, and bring critical decisions up before they are needed.

Managing Contributions to Open Source Projects

One of the core benefits of building a community is receiving collaborative contributions. These contributions may be bug fixes, suggested features, potential use cases, redesign specifications to allow for more generic application of the software, or other types of modifications. Depending on how you build your community, contributions may even come from people who are not software engineers—for instance, translations of texts into different languages.

It's a best practice for open source projects to include a contributor guide, clearly labeled for people to easily find. Contributor guides let contributors know what is expected of contributions. Contribution guides outline the process to contribute, making it simple for contributors to engage with the open source project in an organized fashion. For example, contributor guides may require that all code adheres to a consistent coding style. GitHub provides **useful advice for building contributor guides**, and the **Atom** and **OpenGovernment** code repositories provide good examples to follow as well.

As mentioned previously, even if code is open source, that doesn't mean anyone can modify the code directly. Instead, people can suggest modifications, which may be pulled into the codebase by the owners after a review. Code repository hosting tools (such as **GitHub**) have simple access controls that let you manage who is allowed to contribute to the codebase, and what processes they must follow to submit their contributions. You should clearly communicate the policies for how contributions are managed in your open source project, so that contributors understand what processes to follow and can submit contributions in a way that is easiest for you to manage. For example, you may require that contributors create their own copy of the codebase (generally called a "fork"), make their modifications on that copy, and then request that the copy be merged into the original codebase pending your review. The **OpenGovernment** repository has a good example of simple contributor guidelines.

Contribution policies should be outlined in a contributor guide included in the codebase and documentation. As the owner and author of the software, you will fill the role and responsibilities of the "maintainer" for the project. Maintainers are the main group of software developers who manage the codebase. Maintainers generally have full access to make changes to the codebase, and also manage access controls for other types of contributors as well. Maintainers are also responsible for reviewing and accepting contributions from the community. Code reviews should be thorough, and allow for back-and-forth conversation between the maintainers and the contributors. Provide feedback to contributors to let them know what needs to change before their proposed change is accepted into the main codebase.

Licensing Considerations

When releasing open source software, you should always release it under an explicit license. Software licenses lay out the terms under which the software can

be used by other people or entities. A license is legally binding, and protects you as the author from unauthorized use of the software you create. It also protects members of the community by clearly outlining permitted uses and granting permission for use broadly so that they don't need to seek individual deals with you. Note that open source licenses don't eliminate copyright, but work around copyright to provide the software as open for all. Ownership and copyright are covered in further detail later on in this report.

When releasing open source software, you should always release it under an explicit license.

Different licenses permit different usage. For example, you might choose a license that requires that the software may be used, repurposed, or modified freely, as long as attribution is given to you. **Opensource.com** has a good breakdown of the different kinds of attributes of software licenses.

The best practice for open source government software is to release under the most permissive licenses available. Many government organizations use the **MIT license**, which effectively gives full permissions to any parties to modify, use, redistribute, or perform other activities with the source code. The MIT license is popular because of its broad permissibility, as well as its brevity. Other organizations use the **Creative Commons Zero license**, which puts the software in the public domain. Public domain commonly refers to creative materials not protected by intellectual property laws such as copyright, trademark, or patent laws. The **Open Source Initiative maintains a list of licenses** that they have reviewed and approved, which can be helpful when determining which license or licenses you will use for your software.

Starting Open Versus Becoming Open

As you move to adopt open source practices across your projects, you will find that there are differences between new projects that use open source from the start, and existing projects and systems that need redevelopment to become open. Starting new projects in an open source discipline ensures best practices up front, enabling collaboration and providing transparency at the outset. Making existing software and services open source allows others to learn from your work and reuse the components you have already built. In addition, allowing

contributions to existing projects can bring in additional features or fixes, and infuse the best practices of documentation and security into the project.

Starting New Projects in the Open

The best time to make a project open source is right from the start. This brings all of the benefits of producing open source to your project immediately, such as ensuring secure designs and simplifying collaboration. It saves time by building for security and public availability as you go along, rather than having to lump it all together at the end.

Starting projects as open source encourages better documentation from the start, which enables the rapid onboarding of new contributors or team members as the project grows. It often results in better designed software, since OSS is typically more modular so that it can be easily understood by contributors. This decreases the complexity of the software and improves the ease with which the system can be built and ultimately maintained.

The best time to make a project open source is right from the start.

Opening Existing Projects

It may be the case that you have existing custom-built, proprietary software supporting your services. Reengineering existing software so it is open source still provides the benefits we have discussed. It also provides an opportunity to revisit the software and improve its security, design, and reusability. There are valuable lessons in projects that have already seen real-world use, and making these projects open source can save time and effort on future projects for you, your organization, and for others around the world.

When making existing projects (sometimes called legacy solutions) open source, there are steps you can take to ensure a smooth transition. We discuss some of these in other parts of the report, but we have consolidated them here as an outline for moving legacy solutions into the open. We also recommend reviewing two documents written by GOV.UK: a **case study about moving one of their projects to the open**, and a **blog post on how to open up closed code**.

Section Three: Using Open Source Software

A modern software system is made up of various smaller software components, and each of these components can be either custom built, or reused from existing open source projects. A **report from Synopsys** shows that the majority of software in use today is open source, and much of that is at this component level —building blocks of open source software stitched together to create systems and applications. From basic open source software (OSS) components like code compilers (such as **gcc**), to entire user-facing systems like **WordPress**, reusable OSS spans a wide range of sizes and use cases.

Most OSS comes in these smaller components, sometimes called "packages" or "libraries." These open source packages are used in virtually all software built, as **reported in the same Synopsys study**. They handle small tasks that systems everywhere need to handle, and do it in a simple, vetted, reusable way. For example, many software systems reuse open source packages to make **data requests across the internet**, or to **encrypt and decrypt data**. These packages allow software engineers to save time by building on the existing work of others.

When using smaller open source packages, software engineers use a "package manager" to search for OSS packages that meet their needs, download those packages, and link them into their own code. Package managers also manage updates and version control. Advanced package managers also work to maintain the security and availability of the packages available on their platform. This is less involved than forking and modifying a project, but more technical than simply installing an application such as Firefox.

In many cases, OSS is a whole system, such as in the case of **WordPress** or **Firefox**. These systems can be wholly reused to fulfill the needs of your organization. For example, open source systems can be used for content management or client relationship management. These complete, open source applications are built to be used by non-technical consumers, and include simple instructions for installation and usage. These systems are proven—they are stable and reliable—and can be customized to fit a variety of needs. For example, the White House has been operating its main **whitehouse.gov** website on WordPress since December 2017, demonstrating the stability, security, and reliability of an open source solution.

Regardless of whether your organization is looking to use smaller packages or complete open source applications or systems, there are a number of considerations to make when using existing open source software. Below, we'll explore these considerations.

Finding the Right Open Source Solutions

Finding open source software that meets a specific government need can be difficult. Unlike proprietary licensed software, open source tools rarely have a sales team marketing their brand. However, there are ways you can search for open source tools to fit your needs and make the discovery process easier including:

- **Search for generic applications based on their main function**. For example, if you are looking for a software system that you can use to track applicants as they move through an approval process for a housing permit, you can look for a more generic **open source permit processing system** that fits your needs. On a smaller scale, if you need a component for your larger system to send emails and text reminders to the public, then you could look for an **open source notification system**.

- **Look for open source solutions used or produced by other similar government entities**. If you want a system to manage searches for business records, chances are high that another agency or municipality has done the same thing using OSS. Reach out to these peers via user forums, developer communities on GitHub and Stack Exchange, LinkedIn groups, and other networking opportunities and tell them what you need. They may make suggestions, right down to giving you modifications that work for your particular situation. Networking and information sharing are particularly useful since you can partner with these users to contribute back to the original open source software—especially if you add helpful features.

- **Look at how other governments are leveraging open source software for their services and operation, and learn about what you can adapt in your own work**. Many other offices and municipalities have already started moving to open source solutions for a wide variety of needs in their government, and can provide insights into what has worked for them. Examine what they use to get a sense of what you can take and reuse for your own needs. You may find some good ideas just by looking at what exists, rather than finding something to fit a predetermined need.

Following the Terms of a License

Licenses provide safety for you as a consumer to use the software, as long as you abide by the license terms. When using externally developed open source software, you must follow the author's license agreement. As a consumer of the open source software, you are legally required to follow the license of software you use. These licenses spell out the terms of use, including

how you are able to use, copy, or modify the code; whether you must attribute your usage of the software; whether your software must use the same license if you use their software as a component; and other types of requirements. The license effectively describes the terms of use for the open source code. Open source projects *do not* charge to use the software itself, and payment should never be a part of an open source license. OSS licenses are not agreements you make and sign with the authors. Instead, they allow the author to proactively give permission to the software for a wide variety of uses. While the authors still carry the copyright (which we cover later in this report), the license proactively allows anyone to use the software.

If the authors of OSS do not provide an open source license, do not use that software. The ambiguity of it introduces too many risks, and it's safer to find an alternative with a license that fulfills the same function. If you find something you want to use that does not have a license, you may ask the authors for a license to their software, but it's up to them how they proceed.

Assessing Non-License Costs

OSS is free. This means that there are no costs to license or use the software code itself. However, there can still be costs associated with using OSS. These costs could include application or web hosting or personnel costs if your staff lacks the skills to customize and support the software. As one of our interviewees told us, "Open source is free as in puppies, not as in beer," meaning that while you don't have to pay for the OSS, there may still be costs for its hosting and maintenance.

When using OSS applications that need to be hosted on a server, such as WordPress, you still need server infrastructure on which the software will run. These servers can range from extremely simple to highly sophisticated, depending on the needs of your organization. In addition, you will need storage for corresponding data and databases. Servers and storage infrastructure come at a cost, albeit often a fairly cheap cost if you tap cloud services.

Even though you're using software built by someone else, you may still need to hire people to customize or update the software to fit your specific need. For example, you may need software developers to copy the software used by one jurisdiction and change it to match the language and function of your own. In some cases, the software will have simple mechanisms for customization or updating such that you won't need technical talent.

Even though you're using software built by someone else, you may still need to hire people to customize or update the software to fit your specific need.

While there are some costs, they are negligible compared to the extensive costs associated with licensing closed-source, proprietary software products. The infrastructure costs can be simplified to merely what is needed for your system, and these infrastructure costs tend to decrease over time. Likewise, the costs of bringing in technical talent ensures that the lessons from performing this work stays within your organization.

Determining Suitability of an Open Source Tool

Software that you bring into your organization should be well-documented. Good documentation accounts for multiple audiences. It should contain simple documentation for non-technical people to understand what it does, even if the software itself is highly technical in nature. The documentation should also explain how to use the software, with guides for integrating the software in your organization and rolling it out to users. More technical documentation, such as technical guides for integrations or explanations of the code itself, are also helpful when using open source components or systems, and especially if you plan on modifying the code itself for your context.

OSS that you use should be actively updated, with developers continually iterating the application, applying security patches and releasing new features. If software hasn't been updated in years or even months, it's likely that it will not receive attention when there are needed updates, such as security fixes. Instead, look for active communities and maintainers that regularly add features, fix security issues and bugs, and generally update the software.

Where possible, look for open source projects that are supported and maintained by established organizations. These organizations tend to have more official practices for maintaining open source projects, and can provide much more consistent support and community management. These maintainers should enforce good community management practices, such as reviewing code being submitted and actively informing the community of developments. Examples of organizations that manage large amounts of open source software include **Apache**, **The Linux Foundation**, **Red Hat**, **Mozilla**, and **18F**.

It's likely that you might find an open source project that mostly matches your needs but is missing one or two features, or has extra features you don't need. This is very normal, and showcases the strength of open source. If you need additional features, you can add them, and submit those changes back to the original project as a contributor so that others can take advantage of your work. If you don't have access to technical talent, you can start a conversation about your needs with that project's community.

Lastly, open source projects should clearly outline the various restrictions, requirements, and processes they follow. The open source license should be simple to find and clearly understood. Contributor guidelines and other policies on contributions should also be simple to find and understand. When these documents are clear and available, it protects you as a user from license risks, and creates a joint understanding of how you can engage with the project.

Incorporating Existing Open Source Tools

Once you've found an open source tool that you want to use—whether it's a component, or a full application—you'll have to work to bring it into your organization. The project's documentation should be clear enough to make this process simple. Depending on the type of software, how much it matches your needs, and the size and complexity of the software, there are a variety of ways to bring these open source tools into your organization.

In some cases, OSS is specifically built to be extremely simple to use. For example, desktop applications like Firefox or **OpenOffice** allow people to simply download and install them, as you might do with any other application. Open source applications are no different from closed-source applications in this case, and are typically easy to install and get up and running. The difference: these open source applications allow for extensive configuration without having to modify any of the source code itself.

For software that needs some amount of modification of source code (as opposed to simpler configuration), you will likely create a copy (or a "fork") of the source code for your own organization to use. You can update and change this fork as you see fit, as long as it is in accordance with the license. For example, the **U.S. Forms System** was forked by the **City of Austin** so that they could make their own modifications to it and maintain it as they needed. The City of Austin started from the code provided by the U.S. Forms System, and modified it to meet their particular needs. If, like the City of Austin did, you make changes you want to contribute, you would need to submit the changes to the maintainer for approval.

Section Four: Managing the Details

As you and your organization start introducing more open source software (OSS) and practices into your infrastructure, you may run into some recurring issues. For this reason—rather than encountering the same challenges every time you embark on a new project, you may want to build an infrastructure that allows for all projects to easily be made open source. Here, we've outlined some of the details that should be kept in mind when moving your organization into a more holistic open source strategy.

Develop Policies to Facilitate and Standardize the Use of Open Source Solutions

Policies should encourage and support the broader usage of open source tools within your organization. This may require getting executive buy-in and creating teams to explore how open source will benefit your organization. Once you have buy-in, clearly articulate that projects and teams are allowed to use open source software, and provide clarity on how open source use may interact with other policies. Give teams that want to use open source the permissions they need to do so, and simplify the processes to let them focus on finding and matching software to the needs of their users.

Create organization-wide policy to reduce the barrier to entry for adopting open source practices. Use policy to standardize processes, clarify risk interpretation, set expectations for open sourcing tools, and provide repeatable procedures for projects just getting started. You can find examples of good open source policies in **GSA's Open Source Policy** and **CFPB's policy**, and supporting details on **Code.gov**.

Open source policy is a great way to encourage and enforce the adoption of open source principles. **U.S. Federal policy M-16-21** set the bar for federal agencies to support and use OSS. Since it was released, nearly 7,000 individual federal source code repositories were made available to the public on **code.gov**. M-16-21 requires that agencies "release at least 20 percent of new custom-developed code as Open Source Software," which has encouraged the creation of open source programs across the government.

Establishing a Licensing Policy

Use policy to clarify the interpretation of open source licenses and provide legal consistency of the risks and benefits of open source. This can minimize the misplaced concerns around legality that many hold around open source software. Consistent interpretation is a huge benefit, as one of the larger hurdles with organization-wide use of open source is the variable interpretations of open

source licenses. As previously mentioned, these licenses range from straightforward to complex, and the interpretations can vary depending on the technical proficiency and risk attitude of the legal counsel involved. Creating a consistent interpretation of open source licenses will smooth its use and access across the board.

For example, creating policy that outlines pre-approved software licenses can alleviate the burden of repeatedly seeking approvals. There are a set of commonly-used licenses available from the **Open Source Initiative**, such as the **MIT license** or the **Mozilla Public License**. Licenses vary in how they handle attribution, license reciprocity, and other factors. Opensource.com discusses some key differences in licenses **in this blog post**. It is highly likely that any open source software that you are looking to use leverages one of the licenses from the Open Source Initiative. Consider creating a policy that pre-approves certain licenses so that any software with those licenses can be used in your organization without requiring a license review.

Ownership

When building OSS, it's important to make correct decisions on the licenses, copyright, and general ownership of the software. There are many factors that play into this, including your jurisdiction's copyright laws. For example, in some jurisdictions the government **cannot hold any copyright**, while in others the government **reserves all ownership**. By collaborating closely with agency or organization legal teams, communicating clearly to community members, and using existing licenses, you can navigate this more challenging terrain and develop a plan that meets your needs. Here, we outline some considerations and common strategies. We also recommend reading **GitHub's simple guide to the legal aspects of open source**. However, the best practice is to involve legal support from within your organization from inception, and task them with figuring out how to make this work positively. Attorney and open source developer Ben Balter provides a **guide for government attorneys on open source licensing** that is worth the read.

Software written for government purposes and paid for by the government, should always belong to the government—not to vendors or contractors. This is true regardless of whether the software is open or closed. With open software in particular, open sourcing the software does not eliminate the government retaining copyright of the software it wrote and financed. When procuring services with vendors, ownership and copyright for custom-written software should be clearly outlined in contract documents with a caveat that it belongs to the government. If your jurisdiction cannot hold ownership, then the software should be placed in the public domain. By owning the copyright, you ensure that you control the ability to open source the software, rather than leaving it in the hands of vendors.

OSS may have contributors who are not contracted vendors. This is, after all, one of the great benefits of open sourcing your software. These contributors might be other offices within the same government, public servants from different municipalities, companies or nonprofit organizations, academics, or constituents. To protect your organization's interests and the interests of the open source project, you should clearly articulate to contributors how the licensing and copyright of their contributions will work. There are many ways to do this, and varying industry opinions on the right path to take. One prevailing option is to allow contributors to retain copyright of their contributions, but stipulate that their contributions fall under your license, meaning that any contributors to your project extend an unlimited license to use their code. Alternatively, if you perform a code review of their software, you may be able to argue that by performing the review, the software is released under the appropriate license, as **the Department of Defense has argued**. Work with your legal team to determine the right type of guidance to provide.

Managing Vendors

The government-vendor ecosystem is a vital one that ensures a constant availability of up-to-date knowledge and talent in a flexible format to the government. Historically, this ecosystem has operated outside the open source world, using proprietary software or restricting ownership and usage of software. The move to an open source paradigm will likely require changes to existing business models, and as such, the vendor community may resist any changes in the status quo. However, vendors can adopt open source practices in a way that will support expanded business. Large tech providers, like IBM, have invested in OSS as a way to break into new markets, offer value-added services and absorb highly-skilled developer communities, proving that companies can still profit from open source development.

While some vendors will adopt open source business models, others will be resistant and hold onto proprietary software as a means of sustaining business. As you start including open source in your procurement plans, take care to build in protections and pay attention to details when commissioning software development. Ensure that any contracts include explicit language stating that all software used for the solution, whether newly written or reused, will be open source. Also make it known that proprietary, closed-source software will not be used for the solution. While this may seem redundant, it protects against vendors who may use loopholes to lock you into their platform. Ensure that the government owns the software that it pays for, and if possible, build into the contract which open source license the software will use. Build on existing precedent, adopting language used from previous open source procurements in your or other jurisdictions. While many of these steps may seem defensive, it will

help protect you and your organization and make the move towards open source faster and more securely.

System and Data Interoperability

Whether you are building a new system in the open or reusing existing open source solutions, you will still be operating a system that likely collects data or works in tandem with other systems or applications. This interoperability is crucial to creating an effective software ecosystem that supports your business needs. There are a few things you can keep in mind that support interoperability:

- **Bring knowledgeable people to the table to discuss interoperability between various systems.** If your operations depend on legacy systems, ensure that someone who manages those systems is present for discussions. They should be able to speak to the integration capabilities and limitations of the legacy system. This simple act of bringing the right people together, when done early on, can help drastically with reducing any integration and interoperability pains. A good rule of thumb is to not assume anything about legacy or new systems unless you have first-hand knowledge.

- **Follow common data standards when recording data.** If you examine data, chances are high that at least one (if not several) data schema standards already exist in your organization. These standards help make the collection and sharing of data more consistent across various organizations. Using one or more popular data standards ensures that you're taking advantage of the hard work that went into creating them, saving you from the trouble of having to come up with the data schemas on your own. Many organizations create and manage data standards, such as **schema.org**.

- **Use systems that provide open interfaces (or, build open interfaces into your systems) to facilitate the transfer of data**. These open interfaces, or APIs, allow other systems and software developers to integrate with your solutions. Open interfaces should be well documented. Wherever possible, use common standard interfaces for standard data or operations, so that other systems know what to expect and can integrate even more seamlessly.

In-House Technical Talent

Using OSS puts your organization much closer to the source code of the technology you're using than you may have ever been before. Many governments

and organizations almost exclusively use customized proprietary software, and remain unfamiliar with application source code. This is in part because many organizations don't have the in-house technical talent or expertise needed to modify or make changes to such code. This is fairly normal, and is why the government-vendor ecosystem is so important. Our recommendation is to bring in talent who will support the usage of open source tools, while helping to build a more modern, digital organization.

The following roles are central to any software product development team, and will enable your organization to move towards an open source development paradigm with confidence:

- **Product managers** are broad experts, serving as the connecting point of business needs, technology, and user experience. Product management, not to be confused with program or project management, is a rare discipline in government. Project managers set the vision and mission of products and services, and drive the organization towards the successful implementation and delivery of those products. Bring in a product manager first—someone who has experience building modern live digital products.

- **Designers and researchers** are critical to ensuring that the solutions being built solve real problems in ways that will result in tangible outcomes. Designers and researchers can take many forms, from interface design to service design. Choose what you need based on your context and recommendations from your product manager.

- **Software engineers** are the backbone of software development. Many organizations and government agencies employ software engineers, but they often work in roles that are removed from decision-making. Consider hiring senior software engineers who can write and review code and who manage technology teams and projects. Avoid various software engineering substitutes, such as enterprise architects, as a senior software engineer should be able to fill these roles.

It may seem like a tall order to bring in the talent required to manage OSS, but ideally this talent can be used elsewhere in modern organizations that build and offer services to the public. You won't need to hire swaths of talent, instead, hire enough in-house talent such that any tech-related departments or projects can make use of it. There is no substitute for in-house expertise.

Additional Resources

There are many considerations to juggle when embarking on an open source strategy or developing open source software (OSS) solutions to improve public services or administration. We have created the following resources you may want to refer to in tandem with this report.

Checklist: How Governments Can Leverage Open Source Solutions: In order to facilitate the steps we have covered in this report, we've provided a consolidated list of recommendations for you to refer to as you think through or manage an open source government project.

Open Source Project Hubs for COVID-19: In the wake of the 2020 outbreak of the COVID-19 pandemic, multiple organizations produced lists of open source software solutions that can be used in an effective response and recovery plan. These project hubs shoulder the burden of open source project discovery, making it easier for governments to find projects that fit their needs.

Further Reading on Open Source and Government: We recognize that for some readers, our report may be a lot to digest. For other readers, this report may have prompted a desire to dig deeper into the topic or brush up on proven open source development practices. Alternatively, there may be more foundational questions on the broader topic of software development that need to be addressed before our open source content can be maximally useful to you.

Checklist for How Governments Can Leverage Open Source Solutions

Software for public benefit should be open source by default

1. **Create organization-wide open source policies:** Create organization-wide policies that establish that all teams and offices pursue open source solutions before acquiring, designing, or building their own systems. Policies can also outline processes and decisions to make it easier for teams to adopt open source, such as pre-vetting open source licenses that can be used.

2. **Make usage of open source a priority:** Technology modernization is a top priority for many governments in response to aging legacy systems and heightened constituent and user expectations of transparency and service quality. Open source software should be a part of these priorities and efforts. Open source software can reduce costs, improve transparency, and lead to more efficient services by building on solutions that have been developed to address the same challenge, rather than reinventing wheels.

3. **Clarify misconceptions on the security of open source solutions:** Many believe that open source software is less secure, due to the nature of the source code being available to the public. However, the process of making application source code open can make it more secure in a number of ways: open source code can be reviewed by independent security analysts, making discovery of any issues faster and more actionable. The **Department of Defense**, **Department of Homeland Security**, and **UK Government** have all vetted open source as a secure method of developing software.

4. **Work in the open:** When adopting open source principles, projects should not just be opened upon completion. Instead, all development work should be done in the open. This includes everything from coding to documentation, as well as research and decision-making artifacts. In some cases, such as when policy is not yet released, this may not be possible. But as soon as it is, all historical work and all work going forward —not simply the finished product—should be done in the open.

5. **Publicly document your work:** Open source projects should include comprehensive public documentation that is maintained as up-to-date as possible. While most organizations that have used closed systems are

accustomed to receiving end-user documentation like user manuals, good open source software is itself well-documented, from project roadmap documentation that outlines the plan for the project, to contribution guidelines that guide outside developers on how they can best contribute, and software documentation that explains how different portions of the software work. For good references on projects that have achieved these goals, see the **Atom repository** and the **VSCode**.

6. **Use permissible open source software licenses:** The open source software license ecosystem is robust, mature, and well-understood. Many resources exist to support navigating licenses, such as the **Open Source Initiative's list of licenses** or **GitHub's choosealicense.com**. We recommend permissive licenses, such as the **MIT license** and the **Apache license**, as they give anyone the ability to use, modify, and redistribute the software as they see fit, which will help not only your organization, but others who are trying to solve a similar problem for their constituents.

7. **Migrate existing software and code to open source:** While the best practice is to make projects open from the start, an existing closed system doesn't necessarily have to remain closed. Opening existing projects is worth the effort, even though it can be difficult. The process provides value in the various arenas described throughout the report, including security, transparency, and accountability. Migrating existing projects tends to be more difficult because the entire project needs to be reviewed prior to being made open. Reviews should check for and resolve security vulnerabilities, including bugs as well as sensitive information (e.g. passwords) that were written into code. It also requires all documentation to be updated.

8. **Hire in-house technical talent to manage common open source solutions:** As with any technology project, certain types of in-house expertise are critical in using and managing open source solutions. Hire technical talent, such as product managers, designers, and senior software engineers, to support building, modifying, and using open source software.

9. **Facilitate interoperability:** Whether you are building a new system in the open or reusing existing open source solutions, you will still be operating a system that likely collects data or works in tandem with other systems or applications. Incorporating team members who are knowledgeable about existing systems and prevalent data standards will drastically reduce integration pains down the road. Use open interfaces, or APIs, that are well-documented and automate the transfer of data to ease future system integrations.

10. **Follow product development best practices:** Whether software development is open source or not, it is essential to learn from and follow known software development best practices. Follow established approaches such as **agile development** to define clear goals, build iteratively, and design with users in mind. We find **GOV.UK** and the **U.S. Digital Services Playbook** to be great starting points for understanding best practices for creating government digital services.

Open Source Project Hubs for COVID-19

In the wake of the outbreak of the COVID-19 pandemic, multiple organizations have produced lists of open source software that can be used in an effective response and recovery plan. These project hubs shoulder the burden of open source project discovery, making it easier for governments to find projects that fit their needs. We recommend that any governments looking to expand their use of open source look to these resources. Using open source tools to tackle problems and systemic changes related to the pandemic provides all of the benefits that we've listed throughout this report, including increasing accountability and transparency, reducing risk and cost, and enabling faster deployment of tried and tested solutions.

- New America produced the **Pandemic Response Repository**, a collection of open source digital resources to help governments respond to the Coronavirus.

- The Government of Canada produced **Open Call**, currently in alpha, which includes free digital tools to address common COVID-19 challenges and free technical support for government teams.

- The United Nations Development Programme's Global Centre in Singapore produced the **Open Source Digital Tools to tackle COVID-19**, which is a collection of open source tools to accelerate the digital response to COVID-19.

Further Reading

Primers on Open Source

Open Source Guides

A series of guides recommended by the United States General Services Administration that introduce readers to the open source development model.

Open Source Licensing: What Every Technologist Should Know

A short introduction to the various types of open source licenses, including the history of OSS licenses and the differences between permissive and copyleft licenses.

Making Source Code More Open and Reusable

A brief article about improving the security, effectiveness, and reusability of code.

Be Open and Use Open Source

Resource that delves into the difference between OSS and open standards. This tool also discusses benefits such as readily-available software, saving time and resources to solve common challenges, lower implementation and running costs, and the ability to integrate with closed-source software.

Roads and Bridges: The Unseen Labor Behind our Digital Infrastructure

Explains OSS economics, describes a historical description of the rise of OSS and its contrast with traditional proprietary software.

Benefits of Open Source

Why Open Source Matters

Blog post from the Canadian Digital Service (CDS) that describes the benefits of open source, the approach taken by the CDS, and an FAQ about security, personally identifiable information, and what to not open.

The Benefits of Coding in the Open

Nine reasons why the United Kingdom's Government Digital Service recommends coding in the open.

Government Open Source Resources and Projects

Open Source Software in Government: Challenges and Opportunities

Identifies common challenges with collaborative software development and its use in government, including misconceptions and questions about open source and policies that obstruct OSS development.

Department of Defense Open Source Software (OSS) FAQ

Long-form FAQ from the DoD about OSS, licensing, security, and its development, use, and release by the U.S. government.

Digital Public Benefits Alliance

The Alliance hosts a platform that aids organizations in discovering openly-licensed technologies, data models, and other resources that can be deployed to support the sustainable development goals (SDGs).

Open Source Best Practices

Best Practices for Maintainers

A set of resources to help open source developers who are responsible for maintaining a software project, including clearly communicating project expectations, mastering documentation, and empowering contributors to find solutions to shared challenges.

When Code Should Be Open or Closed?

Summarizes the grounds for keeping code closed and why other code should be open.

How to Open Up Closed Code

A set of simple suggestions if government offices want to open previously proprietary code.

Security and Open Source

Security Considerations When Coding in the Open

Guidance from the United Kingdom's Government Digital Services about security and open source software.

Keeping Programs Secure with the Appropriate Level of Security

Advice on how to assess security risks of a government program, including performing access audits, building risk mitigation plans, and fostering a culture of constant monitoring and improvement.

Vulnerabilities in the Core

A report by the Linux Foundation presenting the initial findings of a census assessing the scope of open source software used in programs and infrastructure within the private and public sectors.

Don't Be Afraid to Code in the Open: Here's How to Do It Securely

The United Kingdom Government Digital Service provides guidance on the different situations in which it is most secure to keep source code closed to the public versus when it is safe to offer it openly.

Legal and Open Source

The Legal Side of Open Source

GitHub's high-level primer for legal teams to understand the legal implications of open source.