Zero Code and Infrastructure Research Data Portals

Joe Bottigliero
Rachana Anathakrishnan
Kyle Chard
Ryan Chard
Ian Foster
University of Chicago and Argonne National Laboratory
Chicago, Illinois, USA

ABSTRACT

Data portals are web applications that facilitate data discovery, access, and sharing. They are essential to meet the FAIR data principles and for advancing open science, fostering interdisciplinary collaborations, and enhancing the reproducibility of research findings. We present a novel zero code and infrastructure approach to simplify and accelerate the creation and customization of data portals. Our data portals do not require an application server and can be served from static content hosting services, removing the need to administer infrastructure. We present a new generator approach to portal development which allows users to create highly customized and powerful data portals by modifying only a JSON document.

CCS CONCEPTS

• **Human-centered computing** → **Collaborative and social computing systems and tools**; *Accessibility systems and tools*; • **Software and its engineering** → **Software system structures**.

KEYWORDS

Data Portals, Cyberinfrastructure, Globus

ACM Reference Format:
Joe Bottigliero, Rachana Anathakrishnan, Kyle Chard, Ryan Chard, and Ian Foster. 2024. Zero Code and Infrastructure Research Data Portals. In *Practice and Experience in Advanced Research Computing (PEARC '24), July 21–25, 2024, Providence, RI, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3626203.3670595

1 INTRODUCTION

Exponential growth in data volumes presents new challenges for making research data Findable, Accessible, Interoperable, and Reusable (FAIR). New tools are needed to allow humans and machines to easily and efficiently organize and act on ever-larger datasets. Science gateways, data commons, and data portals, herein referred to as *data portals*, are web-based platforms that facilitate the storage, sharing, discovery, and access to data sets, using only a web browser. The primary goal of a data portal is to enhance transparency, collaboration, and effective use of data by providing an organized and user-friendly means of discovering and accessing data resources. Data portals empower researchers to conveniently curate and share their data directly, streamlining the dissemination of scientific knowledge and fostering interdisciplinary collaborations. However, creating, deploying, and maintaining data portals can require substantial, and on-going, development time and computing resources.

In prior work we introduced the Modern Research Data Portal [2] (MRDP) design pattern. The MRDP pattern delivers secure, efficient, and scalable access to research data following two key principles: first, the separation of control mechanisms from data storage, and second, the delegation of essential capabilities such as data transfer to robust and high-performing cloud services. These principles allow business logic and control mechanisms to be performed within enterprise security boundaries, while enabling data to move freely between resources using the Science DMZ model [8]. Collectively, this pattern enhances performance while balancing security requirements.

Our reference MRDP implementation, built with Flask, features a simple web interface that allows researchers to share, find, and access data. This implementation leverages Globus [3] for efficient data transfer, user management, and access control irrespective of geographical location, the type of data, or volume. We also developed a second implementation, using the extensible Django framework [11], to create active data portals that support on-demand computation and sophisticated research workflows.

While both our Flask and Django framework portals have been used to implement many data portals [6, 11], we have observed that the need to extend, deploy, and manage Flask and Django applications is a significant barrier to adoption. Here we present the logical next-generation implementation of the MRDP pattern in which these challenges can be addressed via a zero code and infrastructure approach: the Globus Zero Code and Infrastructure Research Data Portal (ZRDP).

As a single-page application (SPA), implemented using the Jamstack architectural approach, a ZRDP can be served entirely in the browser without requiring an application server. The ZRDP SPA leverages external cloud services for more sophisticated functionality (e.g., authentication, managing access to data). ZRDPs employ a zero code and infrastructure design pattern via modification to a simple JSON document. This document is interpreted by a *generator* to produce a SPA that can be served via cloud hosting services. This allows a portal, such as that shown in Figure 1, to be created using

PEARC '24, July 21–25, 2024, Providence, RI, USA

ACM ISBN 979-8-4007-0419-2/24/07
https://doi.org/10.1145/3626203.3670595

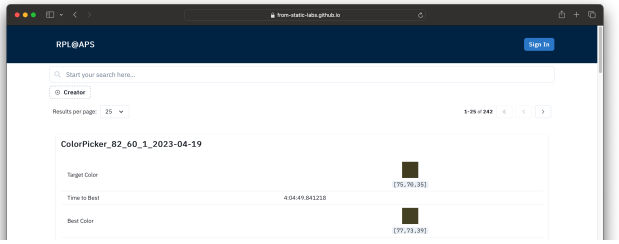

Figure 1: An example ZRDP for Argonne National Laboratory's Rapid Prototyping Lab. The portal is generated from the JSON configuration document in Listing 1, and is served via Github Pages.

```
{
  "_static": {
    "generator": {"name": "@example/search-portal"}
  },
  "data": {
    "version": "1.0.0",
    "attributes": {
      "content": {"headline": "RPL@APS"},
      "globus": {
        "search": {"index": "aefcecc6-e554-4f8c-a25b-147
             f23091944"}}}}}
```

Listing 1: The configuration `static.json` file used to generate the portal in Figure 1. The `generator` statement indicates that the `@example/search-portal` generator is to be used to process this document; that generator will then employ the supplied attributes to determine the `headline` for the portal and the `search` index from which portal contents should be retrieved.

a configuration document, as shown in Listing 1. Our generators use the Globus Javascript SDK to incorporate Globus Auth [13] and integrate other Globus services—enabling ZRDPs to securely share and transfer data, initiate data management flows [5], or dispatch remote analysis tasks via Globus Compute [4].

ZRDPs offer new opportunities for both institutions, individual researchers, and research software engineers by providing the tools needed to easily create and deploy customized data portals. This agility enables institutions to provide enhanced data management capabilities, improve accessibility, and foster openness and collaboration among researchers. Furthermore, the simplicity of ZRDPs empowers individuals to create their own data portals, democratizing access to advanced data management services to securely manage, act on, and share their data.

The remainder of this paper is as follows: section 2 introduces ZRDPs and describes how they are configured, generated, and deployed. section 3 presents our open-source reference implementation available on GitHub. We present related work in section 4 and summarize in section 5.

2 ZERO CODE AND INFRASTRUCTURE DATA PORTALS

The zero code and infrastructure design pattern simplifies SPA implementation by separating the configuration logic from the generation of the application. SPAs do not rely on an application server, instead providing dynamic content through client-side processing, with stateful capabilities, such as authentication, outsourced to specialized services. This approach removes the need to manage server-side state and reduces implementation complexity, and allows the resulting applications to be served via static content hosting solutions such as GitHub Pages and Amazon S3. The pattern encourages reuse and facilitates low-to-no code solutions by minimizing the need for custom development and operational resources.

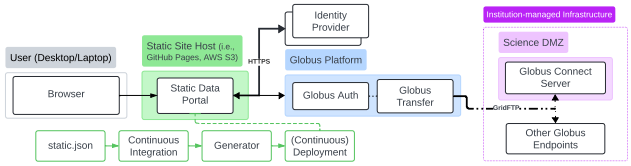

Figure 2: An overview of the ZRDP architecture. A configuration file is processed by a `generator` to produce an SPA that can be served via static content hosting solutions (green). Users interact with the ZRDP using their browser (gray) and, after authenticating using an identity provider, are able to interact with Globus services (blue) to act on data regardless of location (purple).

Our approach, depicted in Figure 2, revolves around a user-defined configuration document that references a *generator* package responsible for processing the configuration file and producing an SPA. Depending on the target generator's usage, the configuration file will be used as a build manifest, configuration file, and content management system. A *toolchain* is used to apply the generator to the configuration and then deploy the resulting data portal. Here we explain how configuration, generators, and the toolchain are used to create, build, and deploy a ZRDP.

2.1 Configuration

Each ZRDP requires a single JSON configuration file, called *static.json*. The configuration document acts as the blueprint that controls behavior of the portal's components, specifying the generator to use, metadata fields, and how the portal should source, process, and display data elements. An example configuration file is shown in Listing 1. This specifies that the *@example/search-portal* generator be used to parse the file and defines a *data* element, which includes a version for generator interoperability, headline title, and Globus Search index that may be used to generate the portal.

2.2 Generators

A generator focuses on pre-generated markup and client-only rendering. The markup and resulting logic used on the client are sourced from a provided configuration file. A generator may be implemented in any language that can parse JSON and produce HTML or an SPA. Generators specify what parts of the configuration file are used and how. The design approach requires some attributes to ensure tooling compatibility and recommends others, but it is ultimately up to the author to define how a configuration file is parsed. In order to take advantage of a common toolchain, a generator must be published in a supported package ecosystem registry and have an accessible git repository.

```
// - out/.gitkeep
// - package.json
//  {"name": "@example/search-portal", "version": "1.0.0",
//   "repository": "github:example/search-portal",
//   "scripts": {"build": "node index.js"}}
// - index.js
const fs = require('fs');
const _STATIC = require('./static.json');
const app = `<html><head>
<script src="https://unpkg.com/@globus/sdk/umd/globus.production.js">
</script>
</head><body><h1>${_STATIC.data.attributes.content.headline}
</h1><pre/></body>
<script type="text/javascript">
  globus.search.query.get("${_STATIC.data.attributes.globus.search}",
  { query: {q: "*"} }).then((r) => r.json().then((p) =>
  document.querySelector('pre').innerText = JSON.stringify(p, null, 2)));
</script></html>`;
fs.writeFileSync('./out/index.html', app);
```

Listing 2: A basic example generator written in Node.js. This generator parses a configuration file to create a single HTML file that uses the Globus JavaScript SDK to run a query against a configured search index.

Listing 2 shows a simple example generator that can act on the configuration defined in Listing 1. This generator produces an *index.html* file for the Rapid Prototyping Lab's ZRDP at Argonne National Laboratory and utilizes the Globus JavaScript SDK to query the Globus Search index specified in the configuration file.

We have created two ZRDP generators using the pattern introduced in this paper: *@globus/static-data-portal* and *@globus/static-search-portal* [12]. These generators provide simple UIs for secure authorization, data transfer, and discovery using the Globus platform and Globus JavaScript SDK. The Globus SDK can be used in isolation and provides convenient way to interact with Globus services in web applications, making it easy to implement features such as third-party file transfers between research institutions, data sharing among collaborators, and invocation of remote analysis tasks and data management flows.

2.3 Build and Deployment

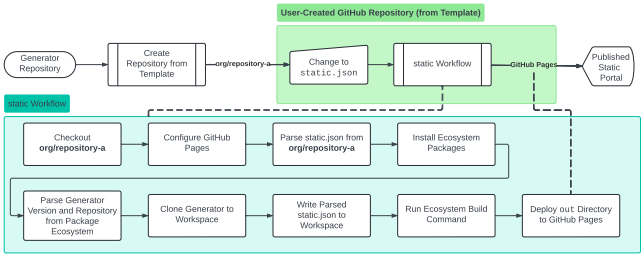

Figure 3: A detailed look at the creation of a ZRDP from a GitHub Template Repository. The workflow is run automatically by GitHub Actions when changes are made to a user's repository.

Zero infrastructure deployment removes the complexity and cost associated with configuring servers, storage, and networking, and thus enables users to focus on developing and deploying applications. The ZRDP toolchain promotes zero infrastructure deployment by automating the build process and deployment of the resulting portal.

The build process, shown in Figure 3, is performed by a GitHub Actions workflow. The workflow configures an environment before cloning the generator into a workspace. The user's configuration file is written to the workspace before the generator's build command is invoked. The resulting portal, produced in the workspace's *out* directory, is then deployed to GitHub Pages.

Publishing generators to a registry, e.g., NPM or PyPI, avoids reinventing the wheel when it comes to many features, including version semantics and pinning. Further, well-known registries allow for the incorporation of existing tooling into the toolchain, thereby solving problems such as security patches and change notifications for downstream consumers.

3 REFERENCE IMPLEMENTATION

We provide template GitHub repositories to simplify distribution of ZRDPs [12]. The template repository includes a simple configuration file specifying a Globus-provided generator and the GitHub Actions workflow used to build and deploy the portal. To use the template one must:

(1) Use GitHub's "*Use this template*" function to create a repository from the template.
(2) Update the repository's settings to allow publishing with GitHub Actions.
(3) Edit the configuration file to fit their needs.

The toolchain will execute upon configuration changes and automatically deploy the portal to GitHub Pages. An example ZRDP using the *@globus/static-search-portal* generator is shown in Figure 1. To accelerate customization of portals our GitHub repository includes a set of example generator packages that showcase various Globus integrations.

4 RELATED WORK

Prior methods for web-based data sharing and publication range from self-service platforms [7, 9] through to customizable portals designed to be deployed and operated by the user [10]. These platforms have been instrumental in facilitating the publication of research artifacts, including datasets and software. However,

self-service platforms rely on the traditional data portal model, where data and catalog are co-located and are limited by the local resources available, while user-operated portals require significant time and resources to deploy and maintain. The ZRDP is designed to balance these tradeoffs by making customizable portals easy to configure and maintain.

Other science gateways and data portals [1] serve as an comprehensive interface for integrating data processing and computational resources while streamlining the interface for users working with extensive datasets. These platforms aim to democratize access to computational resources, reducing the need for researchers to focus on the technical and infrastructure challenges. ZRDPs can also be used to create rich data management interfaces that facilitate analysis.

5 SUMMARY

Data portals are crucial for promoting data discovery, access, and management, supporting the FAIR data principles, and promoting open science. Here we introduced an innovative *zero code and infrastructure* strategy to simplify and accelerate the creation and deployment of custom research data portals. Our approach allows users to quickly create and tailor data portals to their scientific data without the need for implementing, deploying, or managing application servers. We introduce a novel generator-based design pattern for portal development that uses the Globus JavaScript SDK to securely incorporate Globus data management services. ZRDPs enable users to craft specialized and effective data portals by altering only a JSON document. In future work we will develop a catalog of generators to support different use cases.

ACKNOWLEDGMENTS

This work was supported in part by Argonne National Laboratory under U.S. Department of Energy under Contract DE-AC02-06CH11357 and used resources of the Argonne Leadership Computing Facility. We also acknowledge the Globus team for their efforts to support this work.

REFERENCES

[1] Michelle Barker, Silvia Delgado Olabarriaga, Nancy Wilkins-Diehr, Sandra Gesing, Daniel S Katz, Shayan Shahand, Scott Henwood, Tristan Glatard, Keith Jeffery, Brian Corrie, et al. 2019. The global impact of science gateways, virtual research environments and virtual laboratories. *Future Generation Computer Systems* 95 (2019), 240–248.

[2] Kyle Chard, Eli Dart, Ian Foster, David Shifflett, Steven Tuecke, and Jason Williams. 2018. The Modern Research Data Portal: A design pattern for networked, data-intensive science. *PeerJ Computer Science* 4 (2018), e144.

[3] Kyle Chard, Steven Tuecke, and Ian Foster. 2014. Efficient and secure transfer, synchronization, and sharing of big data. *IEEE Cloud Computing* 1, 3 (2014), 46–55.

[4] Ryan Chard, Yadu Babuji, Zhuozhao Li, Tyler Skluzacek, Anna Woodard, Ben Blaiszik, Ian Foster, and Kyle Chard. 2020. FuncX: A federated function serving fabric for science. In *29th International Symposium on High-Performance Parallel and Distributed Computing*. 65–76. https://doi.org/10.1145/3369583.3392683

[5] Ryan Chard, Jim Pruyne, Kurt McKee, Josh Bryan, Brigitte Raumann, Rachana Ananthakrishnan, Kyle Chard, and Ian T Foster. 2023. Globus automation services: Research process automation across the space–time continuum. *Future Generation Computer Systems* 142 (2023), 393–409.

[6] LSST Dark Energy Science Collaboration. 2023. *LSSTDESC Data Portal.* Retrieved March 2, 2023 from https://data.lsstdesc.org/

[7] Mercè Crosas. 2011. The Dataverse network: An open-source application for sharing, discovering and preserving data. *D-lib Magazine* 17, 1/2 (2011).

[8] Eli Dart, Lauren Rotman, Brian Tierney, Mary Hester, and Jason Zurawski. 2013. The Science DMZ: A network design pattern for data-intensive science. In *SC'13*. 1–10.

[9] European Organization For Nuclear Research and OpenAIRE. 2013. Zenodo. https://doi.org/10.25495/7GXK-RD71

[10] Michael McLennan and Rick Kennell. 2010. HUBzero: A platform for dissemination and collaboration in computational science and engineering. *Computing in Science & Engineering* 12, 2 (2010), 48–53.

[11] Nickolaus Saint, Ryan Chard, Rafael Vescovi, Jim Pruyne, Ben Blaiszik, Rachana Ananthakrishnan, Mike Papka, Rick Wagner, Kyle Chard, and Ian Foster. 2023. Active research data management with the Django Globus Portal Framework. In *Practice and Experience in Advanced Research Computing*. 43–51.

[12] The Globus Team. 2024. *Zero Code and Infrascture Research Data Portals.* Retrieved April 26, 2024 from https://github.com/from-static-labs

[13] Steven Tuecke, Rachana Ananthakrishnan, Kyle Chard, Mattias Lidman, Brendan McCollam, Stephen Rosen, and Ian Foster. 2016. Globus Auth: A research identity and access management platform. In *12th International Conference on e-Science*. IEEE, 203–212.