Exosphere - Bringing The Cloud Closer

Julian Pistorius * §, Chris Martin†, Sanjana Sudarshan‡ and David S. LeBauer*‖
*College of Agriculture and Life Sciences – The University of Arizona – Tucson, Arizona, USA
§julianp@arizona.edu, ORCID:0000-0002-3485-0084
‖dlebauer@arizona.edu, ORCID:0000-0001-7228-053X
†Founding Developer and Maintainer – Exosphere Project – Flagstaff, Arizona, USA, chris@c-mart.in
‡Research Technologies – Indiana University – Bloomington, Indiana, USA, ssudarsh@iu.edu

***Abstract*—Exosphere provides researcher-friendly software for managing computing workloads on OpenStack cloud infrastructure. Exosphere is a user-friendly alternative to Horizon, the default OpenStack graphical interface. Exosphere can be used with most research cloud infrastructure, requiring near-zero custom integration work.**

***Index Terms*—Cloud computing, Extreme Science and Engineering Discovery Environment (XSEDE), Jetstream Cloud, OpenStack, user interface (UI), usability**

I. Background and Motivation

Researchers use cloud services for on-demand and interactive scientific computing workloads. Many institutions are establishing cloud infrastructure and others are increasing the capacity of their existing clouds. OpenStack [1] is the operating system which powers many of these research clouds, including all of the following:

- Recent NSF award recipients: Jetstream 2, Massachusetts Open Cloud, CloudLab, Chameleon Cloud, and Aristotle Cloud Federation
- Campus clouds at many research institutions (commissioning now: SUNY Binghamton and Buffalo, University of Cincinnati, UC Santa Barbara)
- Research clouds at Oak Ridge and Los Alamos National Laboratories
- International research organizations (CERN, New Zealand eScience Infrastructure, Australia's National eResearch Collaboration Tools and Resources project (Nectar))

Research cloud services must provide user-friendly interfaces in order to broaden adoption within scientific communities, particularly by researchers who do not have a strong computational background. Unfortunately, the default user interface for OpenStack (named Horizon) was developed for use by IT system administrators. To complete common tasks (e.g. creating and connecting to a cloud instance), Horizon demands familiarity with firewall security groups, SSH key pairs, and computer networking. Atmosphere (developed by CyVerse) [2] is a user-friendly alternative interface to OpenStack. Atmosphere's source code is freely available, but each Atmosphere deployment requires significant integration work and approximately one full-time developer in ongoing maintenance and specialized user support. Most groups that host research clouds cannot support Atmosphere, so only two organizations (CyVerse and Jetstream [3], [4]) use it in production. With Horizon left as the only widely-deployed graphical interface for research clouds, there is a gap between the potential of these resources and scientists' ability to use them.

II. Proposed Solution

We address this problem with Exosphere: a researcher-friendly software client for OpenStack which lowers the barrier to using powerful and flexible cloud services. Exosphere is designed to be used with any OpenStack cloud without custom integration work or maintenance by that cloud's administrator. Exosphere communicates with OpenStack services: the cloud administrator usually does not need to do anything except provide access to the standard OpenStack APIs. Community members already use Exosphere to access Jetstream Cloud and CyVerse's OpenStack infrastructure. Exosphere is fully open-source (BSD-licensed) software.

Exosphere can support research computing applications in many ways, empowering users to create and manage persistent servers for databases and science gateways, scalable clusters for scientific computing, classrooms and workshops, and disposable sandbox servers for maximum flexibility in testing and development work.

A. Goals and values

- Deliver the most user-friendly way to manage workloads on non-proprietary cloud infrastructure
- Provide a consistent user experience across infrastructures operated by different organizations
- Empower people to use Exosphere along with other tools to manage the same resources, with minimal cost of switching (i.e. make it easy to move away from Exosphere to other tools, and come back at will)
- Support security-focused workloads
- Promote an open, community-driven development approach (e.g. user interviews [5] generate issues tagged with "Community request" [6])

III. Architecture

The Exosphere client application is usable against OpenStack with no other dependencies on back-end services, although proxy servers are used where needed (Fig. 1). These proxies facilitate secure connections to services running on users' instances, and connections to OpenStack APIs when

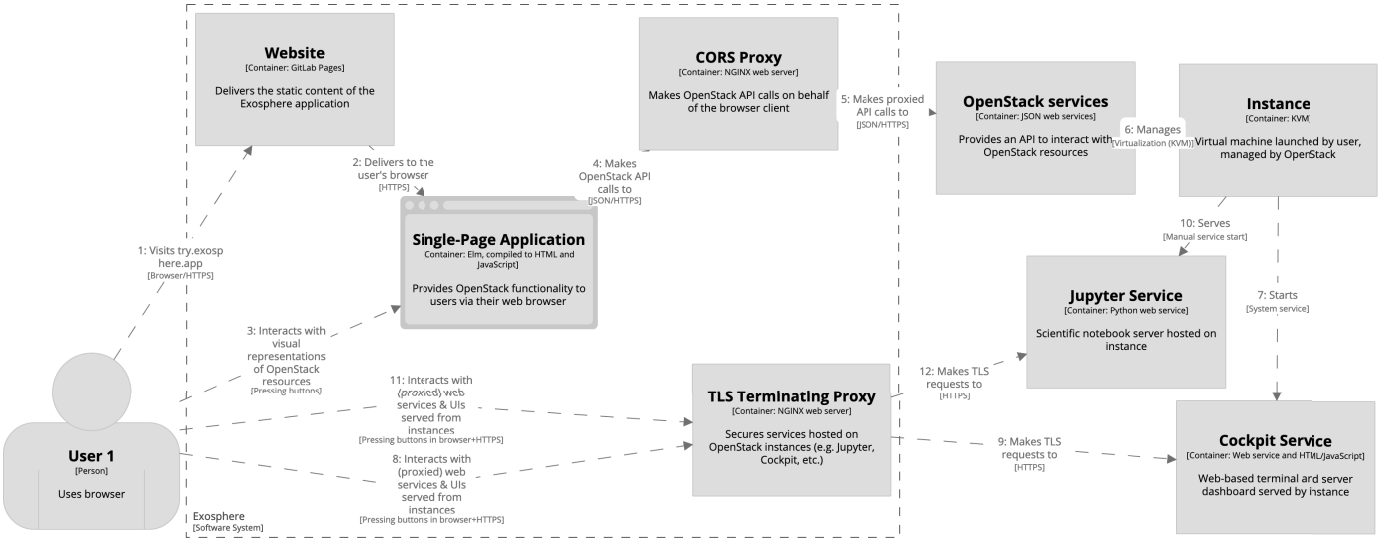

Fig. 1. Exosphere Architecture Diagram.

they would otherwise be unreachable due to restrictions imposed by the user's web browser (same-origin policy) or the user's network (blocking outbound TCP ports used by OpenStack APIs).

Exosphere is written with Elm [7], a functional programming language for building user interfaces. It is compiled to JavaScript, HTML, and CSS. The same codebase is usable in a standard web browser, or as a cross-platform desktop application (currently using the Electron [8] framework).

IV. Features

A. Features available now

- Create and manage cloud instances and volumes (Fig. 2, Fig. 3).
- Delivers on each instance, using Cockpit [9]:
  - One-click terminal, no knowledge of SSH required
  - One-click graphical dashboard
- Usable with nearly any OpenStack cloud[1].
- Completely standalone app, no custom back-end server required for use of core features.
- Secure defaults:
  - No administrative backdoor SSH key needs to be deployed on instances.
  - No need for application to have administrator access to OpenStack; works with standard user privileges.
  - Small, well-defined set of dependencies and "moving parts".
  - Continuous Integration (CI) system scans application dependencies and alerts on known security vulnerabilities. Exosphere uses GitLab CI [10] with two scanners, namely Gemnasium and retire.js. Due to a technical limitation with GitLab, vulnerability reports are currently only available to project maintainers, but we plan to work around this and make the reports public.

[1]Queens release or newer, with service APIs accessible via network from the user's device

B. Non-features

We do not plan to implement the following, as they are incompatible with the project's goals.

- Requiring custom (back-end) services to support core features, to the extent that we can avoid it. (See Challenges section, subsection "Browser-accepted TLS connections to cloud instances".)
- Re-inventing that which is both time-consuming and not uniquely valuable, e.g.
  - User authentication (OpenStack handles this for us, but see "Institutional Single Sign-on" below)
  - User/resource pools (currently using OpenStack projects)

C. Planned features

- Planned features for users:
  - Allow user to deploy a graphical desktop environment to instances launched from popular Linux distributions; provide remote graphical session using Apache Guacamole [11].
  - For clouds that offer GPU-connected instances (such as Jetstream 2), we also plan to support 3D-accelerated streaming desktop environments, using a VirtualGL and TurboVNC back-end for Guacamole. Strudel [12] is a client application which provides GPU-accelerated streaming desktops on certain HPC systems, but we are unaware of an existing solution

for research cloud infrastructure, where the user has the flexibility of full root access to a Linux virtual machine. Exosphere will meet that need.
  - Ease the tasks of data management, with a web-based interface to upload and download files to/from cloud instances. This feature will likely employ the graphical file browser included with Guacamole.
  - Docker/Singularity container integration (see Challenges section below).
  - One-click support for deploying common data analysis workbenches (such as JupyterLab and RStudio) when creating an instance. Exosphere will likely pass cloud-init userdata to the instance which will install the software, and use a proxy server to deliver secure remote access to the workbench in the user's web browser.
  - Smoother installation/upgrade process for the desktop version of Exosphere, with signed binaries and fully automatic updates.
  - Custom workflow/toolchain/stack sharing, which could include, but is not limited to custom disk images.
- Planned features for infrastructure operators:
  - Institutional single sign-on with OpenStack credential management/leases.
  - Allocation service and tools.
  - Reporting tools for operators and principal investigators of projects.
  - Integration with user support and ticketing systems (e.g. Intercom, Zendesk, Jira).
  - On-premises reverse proxy server (when OpenStack services are behind a firewall).

V. Science Use Cases

Exosphere is used by a variety of scientific and engineering contexts to provide both production and development computing environments and support reproducible research. Here we briefly describe a few.

Research groups use Exosphere to deploy and provision virtual machines for development and testing of reproducible analyses [13]. This core use case allows all researchers to share common computing environments as well as scale up analyses using available cloud resources.

The PEcAn Project framework for ecological model-data synthesis [14]. PEcAn includes simulation models, statistical analyses, a database, file management, web interfaces and APIs to support the use and analysis of crop and ecosystem models. Individual components can be orchestrated using docker-compose. This is deployed on a virtual server created by Exosphere on CyVerse's OpenStack.

Exosphere is also used to provision databases that index metadata collected from small Unoccupied Aerial Systems (sUAS, i.e. drones). The ElasticSearch stack (including Kibana) is deployed on virtual servers created by Exosphere. These systems are also used to provide a variety of applications to researchers using sUAS, including plant sciences, agricultural research, and earth science (Geomorphology).

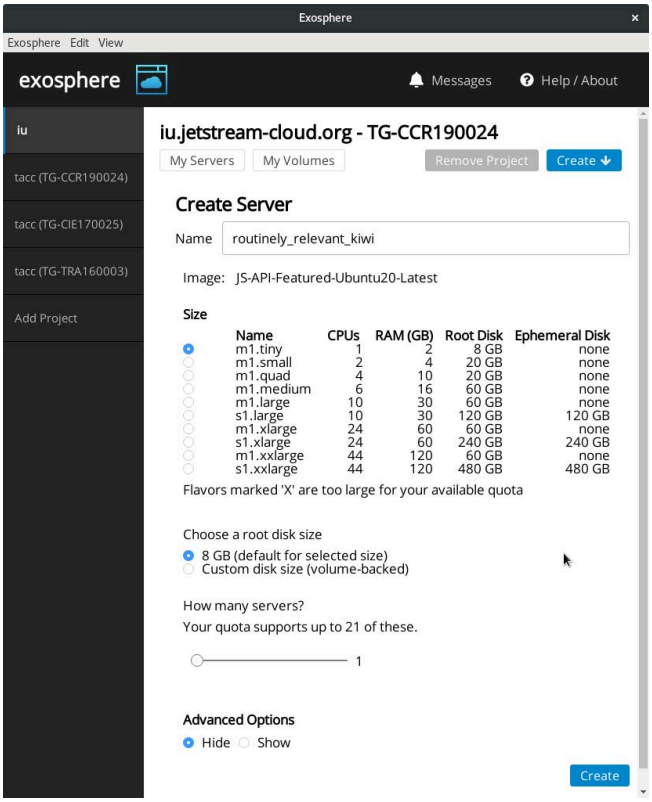

Fig. 2. Launching a virtual server in Exosphere.

Exosphere is also used to enable educational workshops hosted by Jetstream Cloud [15] under the NSF REU program.

VI. Challenges

A. Same-origin policy

Exosphere communicates with OpenStack REST/HTTP APIs on behalf of the user, but modern browsers impose the same-origin policy [16], a security feature which prevents Exosphere from making these API calls unless the OpenStack administrator configures cross-origin resource sharing [17]. This threatens users' ability to access any OpenStack deployment with Exosphere running in a web browser; the administrator of that OpenStack would need to allow the cross-origin requests.

We found two workarounds for this issue, to avoid the need for custom configuration of OpenStack:

- Exosphere can be served alongside a reverse proxy server [18] which makes OpenStack API calls on behalf of the client. This introduces a back-end dependency which is lightweight but still suboptimal, given our goal to distribute and consume Exosphere as self-contained, standalone client.
- When Exosphere is packaged as a desktop client using Electron (rather than served to a web browser), there is no

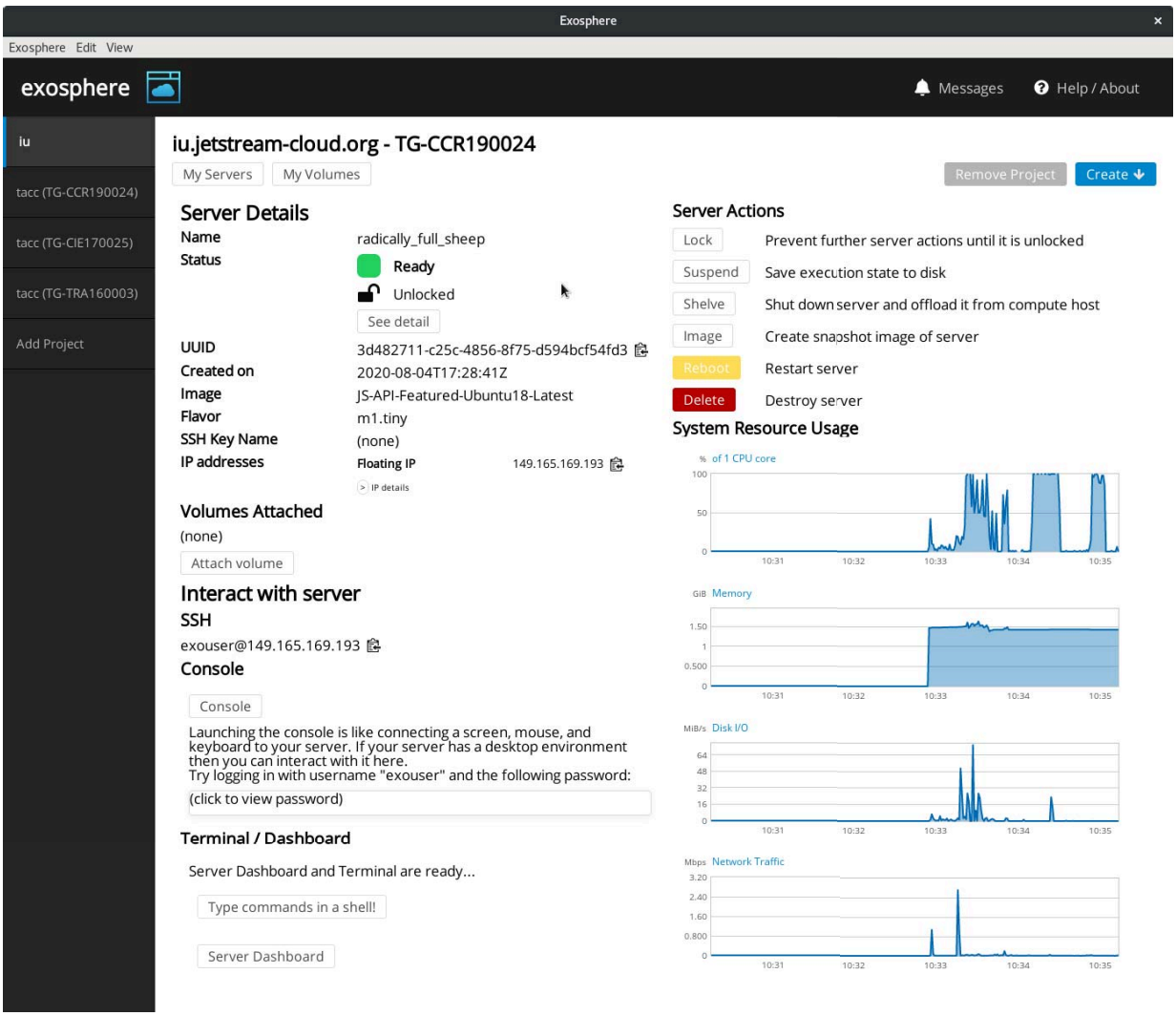

Fig. 3. Exosphere's Server Details view.

same-origin policy and no restriction of where the client makes API calls.

B. *Browser-accepted TLS connections to cloud instances*

In order to deliver easy, one-click interactivity to a user's cloud instances in a secure manner, we must serve content from users' instances using a TLS certificate [19] that the user's web browser will accept. This is needed for a command-line shell and graphical desktop session to each instance; it is also necessary for secure connections to browser-based interactive tools like JupyterLab.

A service like Let's Encrypt [20] could support automatic deployment of certificates to users' instances, but our attempts have not succeeded for two reasons.

First, Let's Encrypt requires the ACME [21] client (i.e. the Exosphere user's cloud instance) to demonstrate control of a public DNS hostname. A DNS record for the instance's public IP address can be pre-created by the OpenStack administrator (perhaps en masse for the entire public IP space of that cloud), but this frustrates our goal of no administrator configuration work required in order for Exosphere to be usable with a given OpenStack cloud. We could require the Exosphere user to register a domain and create a DNS record (pointing it to the cloud instance), but that would not be user-friendly.

Second, Let's Encrypt enforces rate limits [22] which restrict the number of certificates obtained per registered domain (currently 50 per week). If this limit were not imposed, a large OpenStack provider could pre-create public hostnames for its entire public IP space in the form of (e.g.) "instance-128-196-65-75.example-openstack-cloud.org", and a user's cloud instances could automatically obtain TLS certificates for these hostnames at first boot. Under this limit, however, a high-traffic OpenStack provider would quickly exhaust the 50 certificates that Let's Encrypt will issue in a given week for host records

belonging to example-openstack-cloud.org. Unless the cloud administrator was willing to register many domains (perhaps one for each public IP address, which would be costly), this arrangement cannot scale.

A different approach uses a cloud-specific reverse proxy server, for content served by all Exosphere-launched instances in that cloud. The proxy server only needs one TLS certificate, and it can re-terminate TLS for all instances hosted on the same OpenStack cloud (Fig. 4), so long as the cloud administrator can assure another malicious user cannot execute a man-in-the-middle attack [23] on the upstream traffic between the reverse proxy server and the Exosphere user's instance.

```
https://proxy.exo.net/149.165.156.38/12345/cockpit
        \____________/ \_____________/ \___/\______/
              |              |            |      |
  proxy svr hostname   instance IP    port     |
                                                |
  this part of URL path is passed to the instance
```

Fig. 4. URL scheme for cloud-specific TLS proxy server.

This has been implemented [24] but is not yet in production. The team is also exploring other approaches which do not require a TLS-terminating proxy server at each OpenStack cloud.

C. Scope and boundaries of Exosphere's focus

It is important to communicate what Exosphere does and does not do: Exosphere delivers a user-friendly way to create, manage, and access virtual computers (, storage, etc.) on OpenStack clouds, but Exosphere is not currently involved at the level of applications and analyses that users run on these cloud resources, especially not after those resources are created. This delineation of Exosphere's scope will help guide domain scientists (who are typically not IT infrastructure engineers) toward assembling a suite of tools and platforms that achieve their goals.

There are many emerging ways to build, package, and distribute scientific computing workloads: Docker, JupyterLab, and Makeflow, to name a few. As a client for cloud computing, Exosphere can support researchers' use of any of these tools on any cloud infrastructure. It is unresolved which (if any) of these tools Exosphere should integrate more tightly with, and what a useful integration would look like. We want to make Docker/Singularity containers "first-class citizens" [25] in Exosphere, but we have yet to determine the best way to offer this to users.

D. Support for commercial cloud platforms

OpenStack is the dominant open-source, self-hostable cloud operating system used by many research-focused clouds [26], but proprietary cloud platforms (e.g. Amazon Web Services, Google Cloud Platform) now dominate the commercial market for infrastructure-as-a-service. Many researchers use these commercial platforms, and some receive discounted credits. Collaborators at University of Arizona, Indiana University, and others in the community have expressed strong interest for Exosphere to support these platforms. Exosphere could help researchers exploit instance spot pricing and avoid paying for resources when they are no longer needed.

To build this support would be consistent with the goals of Exosphere, though it represents a significant increase in complexity of the application, and it may dilute the developers' focus on providing an excellent user experience for OpenStack. Each commercial cloud platform has its own proprietary API that Exosphere would need to support, though there exist projects like Apache Libcloud [27] and pkgcloud [28] which try to abstract over these various APIs, and that may help the effort. Commercial cloud services like DigitalOcean [29] and Amazon Lightsail [30] already focus on user-friendliness in the way that Exosphere does, but there is no equivalent for the OpenStack ecosystem aside from Exosphere.

E. Support for security-focused workloads

Community members have expressed [31] a desire for security enhancements to support workloads on sensitive and regulated data. The Exosphere developers intend for all features to be secure by default, but some features (namely the Cockpit-powered terminal and server dashboard) currently require users' instances to be reachable via a public IP address, and that connectivity is often blocked for sensitive/secure workloads. Exosphere could be adapted to, e.g., connect to instances through a hardened bastion host [32], or avoid setting a local user password for recovery purposes.

Different communities will have different security requirements that may, at times, conflict with each other; a modular and configurable approach is probably necessary in order to satisfy them.

F. Nomenclature

Researchers from diverse (and often non-computational) backgrounds use different language to describe the same concepts. Exosphere should use language that users are most likely to understand, but it's often unclear which terms are most likely to be understood. Does a user create a "server", an "instance", or a "virtual computer"? Does the OpenStack concept of "project" match a user's idea of a "project"? This issue is tracked [33] on the Exosphere GitLab project. Likely, the team will choose the terms that are most likely to be understood by a wide audience, make them consistent within the app, and provide a good glossary.

VII. Benchmarks

We did a preliminary benchmarking study (Table 1) which compared Exosphere to Atmosphere and Horizon. We tested the following:

- The time taken to deploy an instance.
- The number of steps/clicks required to create an instance.

We looked at the time taken from start to finish for the instance to become accessible. Qualitatively, Exosphere and Horizon took similar times to provision a usable instance, depending how the user wishes to access their instance. The

user could log into their Exosphere-launched instance via SSH (to the floating IP address) at 1 minute or log into their web-based terminal at 5 minutes. The instance that was launched via Horizon was available between those times, at 3 minutes. Atmosphere took 5 times as long (15 minutes) to complete deployment.

When comparing the number of steps/clicks to provision an instance on the three platforms, Exosphere and Atmosphere require fewer clicks compared to Horizon. This is in the best case where the router, network, subnet, and security groups are already set up on Horizon. A new user would have to set up these items before creating an instance, increasing the total number of clicks and deploy time.

Overall, this is in line with Exosphere's goal of combining the ergonomics of Atmosphere with the deployment speed of Horizon.

We will be collecting more samples using browser automation and adding the methods and results to a public repository [34], [35].

TABLE I
PRELIMINARY BENCHMARKING.

	Atmosphere	Exosphere	Horizon
Instance created (minutes)	1.6	0.8	2
Floating IP assigned (minutes)	2.5	1	3
Instance responds to ping (minutes)	2.5	1	3
SSH/shell/desktop available (minutes)	15	5	3
Number of clicks to create instance	3	3	10

VIII. ACKNOWLEDGEMENTS

- Jetstream Cloud, for allocation of compute resources that we're using to develop Exosphere
- CyVerse, for allocation of compute resources, for prior art and inspiration, and for hosting https://try.exosphere.app
- Andrew Lenards and Vasanth Pappu, for valuable code contributions
- Community contributors who provided user interviews [5]
- Jeremy Fischer at Indiana University for helpful review and feedback on this publication

REFERENCES

[1] "OpenStack." [Online]. Available: https://www.openstack.org/

[2] E. Skidmore, S.-j. Kim, S. Kuchimanchi, S. Singaram, N. Merchant, and D. Stanzione, "iPlant atmosphere: a gateway to cloud infrastructure for the plant sciences," in *Proceedings of the 2011 ACM workshop on Gateway computing environments*, 2011, pp. 59–64.

[3] C. A. Stewart, G. Turner, M. Vaughn, N. I. Gaffney, T. M. Cockerill, I. Foster, D. Hancock, N. Merchant, E. Skidmore, D. Stanzione, J. Taylor, and S. Tuecke, "Jetstream: a self-provisioned, scalable science and engineering cloud environment," in *Proceedings of the 2015 XSEDE Conference on Scientific Advancements Enabled by Enhanced Cyberinfrastructure - XSEDE '15*. St. Louis, Missouri: ACM Press, 2015, pp. 1–8. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2792745.2792774

[4] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkins-Diehr, "XSEDE: Accelerating Scientific Discovery," *Comput. Sci. Eng.*, vol. 16, no. 5, pp. 62–74, Sep. 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6866038/

[5] "user testing · Wiki · Exosphere / exosphere." [Online]. Available: https://gitlab.com/exosphere/exosphere/-/wikis/user-testing

[6] "Issues tagged with "Community request" · Exosphere / exosphere." [Online]. Available: https://gitlab.com/exosphere/exosphere/-/issues?label_name[]=Community%20request&scope=all&state=all

[7] E. Czaplicki, "Elm - A delightful language for reliable webapps." [Online]. Available: https://elm-lang.org/

[8] "Electron | Build cross-platform desktop apps with JavaScript, HTML, and CSS." [Online]. Available: https://www.electronjs.org/

[9] "Cockpit Project — Cockpit Project." [Online]. Available: https://cockpit-project.org

[10] "Dependency scanning." [Online]. Available: https://docs.gitlab.com/ee/user/application_security/dependency_scanning/

[11] "Apache Guacamole™." [Online]. Available: https://guacamole.apache.org/

[12] "Strudel (scientific remote desktop launcher)." [Online]. Available: https://trac.version.fz-juelich.de/vis/wiki/vnc3d/strudel

[13] D. S. LeBauer, J. E. Barrios, E. J. Cain, K. M. Huynh, J. Pistorius, K. Riemer, C. Schnaufer, and J. Guo, "Diag team protocols," Oct 2020. [Online]. Available: osf.io/tzmhp

[14] D. S. LeBauer, D. Wang, K. T. Richter, C. C. Davidson, and M. C. Dietze, "Facilitating feedbacks between field measurements and ecosystem models," *Ecological Monographs*, vol. 83, no. 2, pp. 133–154, 2013.

[15] "2020 program overview." [Online]. Available: https://jetstream-cloud.org/research/reu.php

[16] "Same-origin policy." [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

[17] "Cross-Origin Resource Sharing (CORS)." [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

[18] "docs/cors-proxy.md · df3085ab7e945a46f81443469cb3e403466352fe · Exosphere / exosphere." [Online]. Available: https://gitlab.com/exosphere/exosphere/-/blob/df3085ab7e945a46f81443469cb3e403466352fe/docs/cors-proxy.md

[19] "Public key certificate," Aug. 2020, page Version ID: 971028954. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Public_key_certificate&oldid=971028954

[20] "Let's Encrypt - Free SSL/TLS Certificates." [Online]. Available: https://letsencrypt.org/

[21] J. Kasten, R. Barnes, J. Hoffman-Andrews, and D. McCarney, "Automatic Certificate Management Environment (ACME)." [Online]. Available: https://tools.ietf.org/html/rfc8555

[22] "Rate Limits - Let's Encrypt - Free SSL/TLS Certificates." [Online]. Available: https://letsencrypt.org/docs/rate-limits/

[23] "Man-in-the-middle attack," Jul. 2020, page Version ID: 968870047. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Man-in-the-middle_attack&oldid=968870047

[24] "Cloud-specific instance TLS proxy (!308) · Merge Requests · Exosphere / exosphere." [Online]. Available: https://gitlab.com/exosphere/exosphere/-/merge_requests/308

[25] "Minimal support for containers as first-class resources (#82) · Issues · Exosphere / exosphere." [Online]. Available: https://gitlab.com/exosphere/exosphere/-/issues/82

[26] "OpenStack User Survey Report," 2018. [Online]. Available: https://www.openstack.org/user-survey/2018-user-survey-report/

[27] T. A. S. Foundation, "Apache Libcloud is a standard Python library that abstracts away differences among multiple cloud provider APIs." [Online]. Available: https://libcloud.apache.org/

[28] "pkgcloud." [Online]. Available: https://github.com/pkgcloud/pkgcloud#readme

[29] "DigitalOcean – The developer cloud." [Online]. Available: https://www.digitalocean.com/

[30] "Amazon Lightsail." [Online]. Available: https://aws.amazon.com/lightsail/

[31] "Security concerns from NCSA (#288) · Issues · Exosphere / exosphere." [Online]. Available: https://gitlab.com/exosphere/exosphere/-/issues/288

[32] "Bastion host," Feb. 2020, page Version ID: 941752998. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Bastion_host&oldid=941752998

[33] "Cloud Resource Nomenclature (#64) · Issues · Exosphere / exosphere." [Online]. Available: https://gitlab.com/exosphere/exosphere/-/issues/64

[34] S. Sudarshan and J. Pistorius, "Cloud User Interface Benchmarks," Aug. 2020, publisher: OSF. [Online]. Available: https://osf.io/pdgmh/

[35] Sanjana Sudarshan and Julian Pistorius, "cloud-ui-benchmarks." [Online]. Available: https://gitlab.com/exosphere/cloud-ui-benchmarks