Migrating towards Single Sign-On and Federated Identity

Jason Anderson
University of Chicago
Chicago, USA
jasonanderson@uchicago.edu

Kate Keahey
Argonne National Laboratory
Lemont, USA
keahey@mcs.anl.gov

ABSTRACT

This paper describes a two-tier architecture and implementation for single sign-on (SSO) federated identity support in Chameleon and the rationale that shaped it. We also describe how we migrated our users to a new account management system in privacy-preserving ways, a community that numbered in several thousand users and had created hundreds of thousands of digital artifacts.

CCS CONCEPTS

• **Computer systems organization → Cloud computing**.

KEYWORDS

cloud computing, authentication, access control, federation

ACM Reference Format:
Jason Anderson and Kate Keahey. 2022. Migrating towards Single Sign-On and Federated Identity. In *Practice and Experience in Advanced Research Computing (PEARC '22), July 10–14, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3491418.3530770

1 INTRODUCTION

Chameleon [10] is a scientific instrument supporting projects in computer science systems research, education, and emergent applications [24]. Support for research on topics ranging from power management, operating systems, and networking to data science in all its flavors requires deep reconfigurability allowing users to boot from custom kernel, use high level of privilege, and access serial console. Chameleon provides those capabilities by supporting bare metal reconfiguration, layer-2 network reconfigurability and strong support for security on top of the complexity concomitant with cloud infrastructures. Thus, when in 2014 the team was faced with developing a system of this complexity from scratch—and in a very short time to maximize community use of an expensive system in its original 3 year duration—the project adopted the strategy of putting core capabilities of the system in the hands of the users as quickly as possible, at the cost of usability and convenience. Consistent with this strategy, the project adopted the use of the TACC Administration System (TAS) for identity and access management, given direct access to expertise (TACC being one of Chameleon's partners) and support for development and integration of its user and allocation management features.

While this decision allowed us to move fast in terms of system and community development, as the project expanded, integrating new applications/interfaces and sites, the lack of single sign-on (SSO) was proving increasingly onerous for users and operators alike and ultimately posed a barrier to project growth. Further, lack of support for federated identity prevented us from offering users easy integration with testbeds like GENI [7], making experiments over multiple testbeds challenging to orchestrate effectively. While several partial solutions were tried or considered—and allowed us to learn—none of them worked well enough to provide a satisfactory level of service to our users. Given the rapidly growing system and community, we were forced to revisit our approach.

In this paper, we describe a two-tier architecture and implementation for single sign-on federated identity support in Chameleon and the rationale that shaped it. We also describe how this architecture can be used to support continuous migration to a new account management system by a community that by that point in time numbered in several thousand and had created hundreds of thousands of owned digital artifacts (such as images, data objects, notebooks, keypairs, experimental traces, etc.), all of which had to be correctly migrated in privacy-preserving ways. In many ways, the engineering and logistical effort was tantamount to rebuilding the foundation under a skyscraper with thousands of inhabitants. We share implementation insights, our lessons learned, and recommendations for migration strategies for handling authentication and authorization in similar systems as well as a migration architecture for systems in heavy use. In summary, we make the following contributions:

- We describe an architecture and implementation of a two-tier federated identity system based on open source Keycloak [34] and Globus Auth [35] systems and extensible to any federated identity provider.
- We describe the pros and cons of this system as well as lessons learned in its implementation and its use for migration of a large community to federated identity, or any new identity management system.

This paper is organized as follows. We first describe the Chameleon project and relevant parts of its implementation, the problems posed by lack of support for SSO and federated identity, and discuss partial solutions that were tried but failed to solve those problems. We then explain the architecture we designed to solve them, its implementation, and migration strategies put in place to make the transition seamless for our community. Finally, we share our lessons learned from both the implementation and migration.

2 BACKGROUND

2.1 Chameleon

Chameleon is an NSF-funded distributed cloud system supporting computer science research, education, and work on emergent applications [10] [24] [25]. Since announcing its public availability

PEARC '22, July 10–14, 2022, Boston, MA, USA

https://doi.org/10.1145/3491418.3530770

in July 2015, the testbed has supported 6,000+ users working on 800+ projects. To cover the broadest possible range of experiments, Chameleon supports bare-metal reconfiguration giving users full control of the software stack including root privileges, kernel customization, console access, as well as the ability to experiment with software defined networking. A small part of the system is configured as a virtualized KVM cloud, and the system recently added an edge testbed, CHI@Edge [23], giving users access to a variety of edge devices via container-based reconfiguration.

The original deployment of the system was across two sites, the University of Chicago (UC) and Texas Advanced Computing Center (TACC); these were recently joined by smaller associate sites [8] [11] operating resources on a volunteer basis based on a packaged deployment of **CH**ameleon **I**nfrastructure (CHI), called CHI-in-a-Box [22]. In addition to operating resources, Chameleon provides centrally-operated high-level *applications* including resource discovery services, a user portal, a JupyterHub [20] deployment, as well as several internal services; these centralized services integrate with all the participating sites and, along with an OpenStack Horizon GUI [16] at each deployment, provide interface to the testbed.

2.2 Testbed Access and TACC Administration System (TAS)

Testbed access is structured such that any user with valid credentials can log into the system—however, in order to actually use the system resources, users need to belong to a Chameleon *project*, associated with a valid *allocation*. Allocations represent a "budget", expressed in *Services Units (SUs)* (one SU represents one hour of wall clock time on one of our baseline servers) available to the project over a fixed amount of time. To request an allocation, users are first certified for PI eligibility according to infrastructure-specific policies [9] and can then propose a project. Once awarded an allocation, a project PI can add multiple users to their project, which gives them access to the system resources. One Chameleon user can potentially be a member of multiple projects at any given time and switch between them depending on their work. Both projects and users can be disabled, e.g., if the project's allocation expires or the project/user demonstrates irresponsible use or abuse of resources.

Chameleon initially provided access to the system via TAS, which performs Identity Access Management (IAM) as well as manages allocations. It is up to the infrastructure to determine how to charge system tenants for allocation use; typically this is derived from compute hours spent on the system. TAS exposes a LDAP read interface and an administrative GUI [31] and REST API that allow operators and automated systems to manipulate state. Chameleon's access model of users, PIs, projects and allocations is inherited from the design of TAS.

2.3 OpenStack and Keystone

Chameleon's implementation is based on OpenStack, a mainstream open-source cloud implementation. In an OpenStack deployment, IAM is provided by a service aptly named Keystone, authenticating local users and issuing tokens to allow them to interact with cloud services configured within the deployment. User accounts can have memberships in multiple projects, where a project is effectively a single tenant in the cloud; memberships are modeled as role assignments granted to a user; adding a user to a project is equivalent to assigning the user a "member" role on that project.

As more resources and users join the cloud, operators often transition to multiple OpenStack deployments to facilitate scaling and/or to optimize a given cloud's configuration for a target use case. In such cases, the OpenStack architecture's dependency on Keystone introduces the problem of how to continue to provide a single login experience for end-users: because each Keystone deployment has its own database of users, projects, and role assignments, accounting is now decentralized with users effectively managing one account per deployment. To make matters worse, role assignments can differ across sites, leading to confusion about why a user is a member of a project on one site but not another.

Operators typically solve this problem in one of three ways: by clustering the various Keystone backing databases under the hood such that they are consistent; by running automation to sync the state of users, projects, and roles at each site via Keystone's administrative APIs; or by implementing account federation, where Keystone delegates authentication to an external entity [28].

3 PARTIAL SOLUTIONS

In a system composed of multiple sites and potentially multiple applications (such as the user portal) the central challenge is how to provide a single sign on experience for users, such that a user can access all the applications and sites throughout the system with a single set of credentials, and that by logging into the system once, they establish a session valid across all site resources [13].

In the early days of the system, all Chameleon users registered for an account and obtained association with a project via the user portal, which effectively served as a front-end for TAS. Account information from the TAS database was then propagated to Keystone databases on any participating sites, allowing users to log in with their TAS username and password, either via a GUI login or in authentication parameters sent with CLI requests. From the user's perspective, it appeared that they had a single Chameleon account, even though they still needed to log in once per individual application or site. This illusion was however often undermined by the time of propagation between the TAS database and individual sites as well as the occasional brittleness of the system that led to inconsistent state between the user databases at different sites.

To mitigate the unreliability and latency of offline sync across multiple sites, we configured the Keystone databases at both sites as a geographically-distributed MariaDB cluster, with the site at TACC replicating to the UC site. We then only needed to sync TAS state to the TACC Keystone DB, which was more manageable, but resulted in other downsides. First, any network partition between sites would cause data at the replica to become stale, resulting in inconsistent state (i.e., users who created an account or joined a project after the partition could not access the replica site). Second, it introduced strong coupling between the sites, because any schema changes now had to be coordinated to avoid breaking Keystone. Finally, any erroneous writes to the replica DB would not only break the replication, but also necessitate a complex recovery process that involved data copy and close coordination between all sites.

A common approach that avoids offline syncing is to configure Keystone to delegate to some LDAP server for authentication [28]. TAS supported LDAP, but we did not pursue this path because of lack of support in Keystone for mapping LDAP groups to projects: Keystone could delegate identity to LDAP and use it to log in the user, but assignment of a user to projects was assumed to be managed separately on each site via Keystone's API. Keystone is open source and therefore we could have adapted its internal engine to support this, but users would still have to sign in to each site independently (i.e., no SSO.)

As more Chameleon sites came online, the prior solutions did not scale either from an operations or user experience perspective. Hoping to kill two birds with one stone, we implemented a custom authentication mechanism to provide SSO while side-stepping the complex DB setup. With this arrangement, web application logins would redirect to the user portal, whereupon if the user had an active session, the system would sync project assignments to the target application before completing the login action. Ultimately this was insufficient for two reasons. First, it required deep development in each application; JupyterHub for example was never integrated properly for this reason. Secondly, users of CLI or Python clients could technically still authenticate with their username and password, but such events would *not* trigger the live-syncing of assignments specific to the web client. Effectively, after a user was added to a project, they would have to use the web client at least once in order to "update" their account before proceeding to utilize API clients like the CLI; this was understandably quite confusing.

4 GOALS

While the approaches described above did not provide the desired solution, their different trade-offs helped us clarify the list of goals for our system as follows:

Single sign-on user experience, which allows users to access all the sites via all the applications/interfaces throughout the system with a single set of credentials, such that logging into the system once establishes a session valid across the site resources. The solution should scale with the number of sites and applications, provide robustness, and provide low operational cost.

Use of existing/federated credentials to manage any Chameleon session. Many of our users already have accounts managed by their host institutions or general-purpose identity provider (IdP) such as Google or ORCiD [17]. Reusing those credentials not only removes an unnecessary barrier to testbed use but also has the potential to extend the benefits of single sign-on across multiple testbeds as demonstrated in e.g. [29].

Support for multiple identity providers allows us to continue supporting TAS users (essential for implementing continuous migration), but more importantly gives us the flexibility to extend the system to support, e.g., institutions not yet part of InCommon or lacking resources for joining such a federation, international collaborations, or future identity providers.

Flexible, centralized authorization policies, independent of constraints that may be introduced by any one identity provider, and may differ across applications (e.g., without an allocation a user should be able to log into a portal but not JupyterHub). Central policy enforcement additionally provides an important "kill-switch" when malicious users or projects must be expunged.

As an implementation concern, we also wanted to avoid storing—or sharing—sensitive information (such as passwords) about our users.

5 DESIGN, IMPLEMENTATION, AND MIGRATION

Considering these goals in the context of our existing implementation left us with a conundrum. While it would have been possible to provide SSO with TAS providing authentication, we wanted to support federated identity credentials. Supporting other IdPs in addition to TAS would solve this problem, yet violates a core TAS implementation assumption, that only TAS users can be associated with TAS allocations. Hence, the stark realization that entirely migrating off of TAS was ultimately necessary for SSO with federated identity. This added significant complexity, as we not only had to swap out our authentication layer, but now also had to find replacements for the myriad authorization responsibilities TAS had for project management, user roles (e.g., PI status), and allocation tracking and approval.

5.1 Architecture

For the final architecture, we settled on two separate systems: one for accounts (IAM), and one for allocations. The allocation system stores mappings of projects to the budget remaining in their allocation, and also tracks usage of that allocation over time. The account system provides IAM, i.e., integrates authentication providers, maintains assignments between users and projects (including roles) and also provides a central access management point. Having both systems under our purview provides a leverage point to introduce "human in the loop" validation to important workflows such as PI and allocation approval.

From the point of view of existing applications, the account system provides central authentication and session management, allowing us to deliver a true SSO experience; if a user logs in once, they can use any client application in the Chameleon ecosystem. However, the account system does not perform authentication directly. Rather, authentication is delegated to any one of several external identity providers. Not only does this allow us to support multiple authentication methods, but it also opens the possibility for integrating non-authenticating identities into the system, e.g., users can associate their GitHub account to their Chameleon profile to allow invoking GitHub's API on their behalf.

Thus, the functions performed by the account system include mapping external identities (e.g., InCommon, Google, TAS) to Chameleon user accounts, organizing groupings of users into tenants on the system, and providing login and session management for Chameleon's applications, which range from a JupyterHub installation, multiple OpenStack cloud deployments, and the user portal.

We chose to split the allocations system from the account system as they have different access patterns and store different amounts and shapes of data. The allocations system, for example, is involved on every request for resources on the testbed and stores an immutable history of all of a project's charges (e.g., leasing multiple

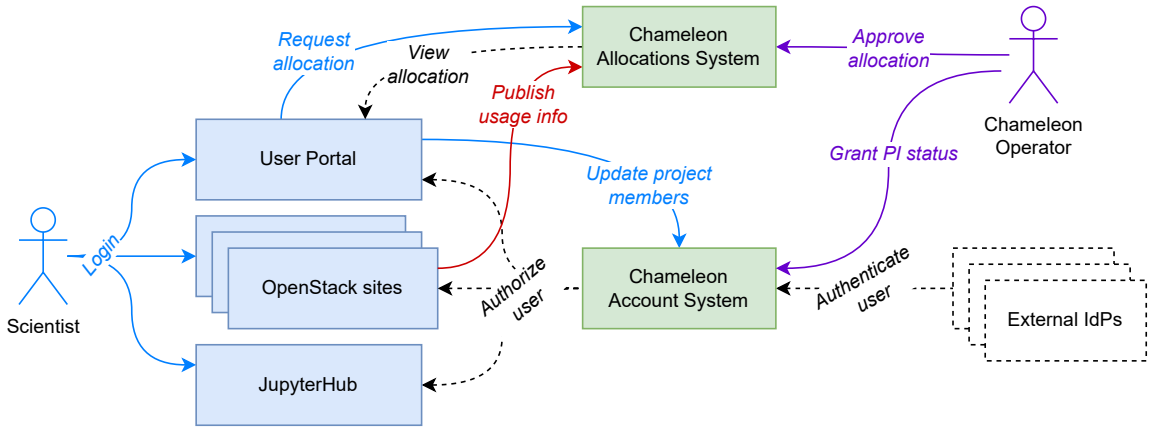

Figure 1: The federated identity architecture; external IdPs authenticate the user, then the account system applies policies which authorize the end-user on a number of applications.

bare metal nodes for a week), while the account system is only involved when a user creates or extends a session, and stores a mutable record of user accounts, which users self-manage. Splitting the systems additionally gives us the ability to vary their implementations with greater ease, e.g., we could change the underlying database for the allocations system without affecting the accounts database.

5.2 Implementation

To implement the account system we selected Keycloak [34], an open source and highly configurable IAM system from Red Hat. It supports integrating with multiple identity provider sources via LDAP/OpenID/SAML, group and user management, customizable per-client enrollment and login workflows, and central session management. Keycloak can act as an OpenID Connect Provider, a common authentication standard compatible with our existing applications. We configured Keycloak to delegate authentication to one of two external identity providers: Globus Auth and TAS. Globus Auth builds on CILogon [5], which allows end-users to authenticate with any host institution in the InCommon federation as well as general-purpose identity providers such as Google Auth and ORCiD [17]. TAS is not part of the InCommon federation and as such could be supported by Globus Auth alone. We selected Globus instead of CILogon for two reasons. First, Globus requires that users self-manage the process of linking their various identities under a canonical account; CILogon requires that operators manually reconcile or configure additional automation. Secondly, we anticipated possible future integration with Globus, so having understanding of Globus identities ahead of time could bear fruit in the future.

While Globus has a Groups feature, which can manage authorization policies and assignments of users, we wanted to keep this logic independent of any identity implementation, both to support identities external to Globus (e.g., TAS), and to have more direct control over policy implementation.

5.2.1 Authenticating Across Applications. Each application was adapted to accept authentication via Keycloak as the account system. The user portal is implemented as a Django web application and as such could utilize open-source plugins like mozilla-django-oidc to provide translation from federated users to Django users. Before, when users made changes to their accounts, e.g., updated their host institution or invited a new user to a project they serve as PI for, a privileged system account would ultimately perform the update against TAS as the backing system; we did not significantly change this model except to use a Keycloak system user and publish changes via the Keycloak Admin REST API instead, though Keycloak's fine-grained authorization services could allow us to remove reliance on such a privileged system account in the future.

For the OpenStack sites, we configured Keystone federation [14], where Keystone delegates authentication to an OpenID or SAML IdP and then maps user "claims" returned by the IdP (which include, e.g., the user's full name, host institution, and in what groups they have membership) to users and projects in Keystone. Unlike the limited LDAP integration we discussed earlier, this federation capability can dynamically create projects and user assignments in the Keystone backend at login time. This transforms the task of identity management in Keystone from one of ongoing maintenance to a one-time configuration: all that must be set up is the mapping from the IdP (in our case, Keycloak) to Keystone entities. While much of this worked off-the-shelf, we nonetheless hit several limitations that required additional development work: Keystone only allowed mapping a user to a single project (our users could be in several simultaneously), did not remove project memberships when a user left a project in the IdP (no offboarding), and the mapping engine, which uses a declarative configuration describing how to perform the translation from IdP to Keystone, had missing or inconsistent behavior with some of the more advanced options that we needed (like regex filtering.) We resolved each issue with patches against Keystone, some of which have already been accepted upstream [2] [3].

JupyterHub is also implemented in Python and supports authentication plugins; we integrated one such, OAuthenticator [21], which supports OpenID Connect. However, this is not the whole story. In the past, users of Chameleon's JupyterHub environment took advantage of its ability to transparently provide authentication

passthrough to the various OpenStack clouds: when users logged in, JupyterHub would re-use the user's submitted password to obtain Keystone access tokens, which were then passed to the user's Jupyter server as environment variables. From there, they could be used by, e.g., Python libraries to access OpenStack cloud APIs, allowing users to orchestrate experiments entirely within Jupyter [4]. In a federated login scenario, this is no longer possible as we are never in possession of the user's credentials directly, yet we were able to configure the Keycloak account system to allow access tokens for the Jupyter application to additionally be used against the Keystone applications (via the "aud" OpenID Audience claim) [3]. Functionally this is equivalent, as we have a way to pivot a user's JupyterHub session into an OpenStack session.

5.2.2 Adapting User Workflows. Chameleon has several policies that we must enforce at user enrollment, such as requiring acceptance of Terms of Service, ensuring that we capture information about the user's declared host institution, country of residence and citizenship, and tracking the user's join date. Prior, this information was collected and stored in TAS. CILogon's COmanage product supports rich configuration of user enrollment workflows; fortunately, Keycloak is similarly customizable via external Java plugins. We implemented several such plugins [33] to add additional actions a user must complete to enable their account. We also extended Keycloak's default theme with Chameleon branding to unify the login experience across our products. Keycloak is customizable enough that we could implement application-specific policies, such as requiring some additional steps when using a given application for the first time, but we ultimately did not require this.

To address the CLI and remote API use case, where users are not using an interactive login flow, we enabled login via CLI password: this is similar to "application passwords" or dedicated credential generation that other systems (e.g., GMail, GitHub) use to accommodate the introduction of MFA or other steps in login that require user interactivity. This was equivalent to allowing OpenID Connect's Resource Owner Password Credentials grant type in Keycloak for the OpenID clients handling authentication for the OpenStack clouds. Users can only use this authentication method when using CLI or e.g., Python clients and they must already have registered to configure this method of access. JupyterHub users do not need to additionally configure such access, as they can transparently authenticate onwards to OpenStack once logged in to Jupyter, as described previously.

5.2.3 Managing Allocations. To implement the allocations system, which needed to replace equivalent functionality in TAS, we built a new API for reporting charges accrued on the testbed (via, e.g., resource reservations) and integrated it into the existing user portal, which already manages much of the allocation lifecycle. We had maintained custom patches in Blazar [15], the OpenStack reservation system, to verify lease requests against a central Redis DB, which held the current allocation balances, to ensure users did not request resources above their budget. We decided to formalize this pattern and wrote a Blazar design specification for a system to gate lease requests and renewals pending according to operator-defined policy rules, including the capability of delegating a policy decision to an external API [1]. This was approved and subsequently developed by the Chameleon team and deployed to the OpenStack sites, which now request approval for all leases and log any changes to the lease to the allocations API. Each lease is modeled as an ongoing charge in a backing database; as a bonus, this allowed us to implement a feature where project members can open a history of all charges from within the user portal to better understand their usage over time; in the past this was challenging to implement due to how allocations were modeled under the old system. The user portal already had workflows that operators used to approve PI status and allocations; these had to be adapted to integrate with the new account system and the new local allocations database, but otherwise remained the same.

Finally, we updated or implemented new forms of reporting to track the total number of users and allocations across all Chameleon systems; this was largely a matter of adapting our existing reports to pull data from Keycloak via the Admin REST API rather than querying TAS, which used to be the source of record.

5.3 Migration

One of the benefits of the architecture described above is that it provides a viable foundation for continuous user migration through its support for multiple IdPs. To leverage it, we first considered the migration needs of our community.

Chameleon's community is naturally volatile: of the thousands of users who accessed the system over the years, most use the system intensively for a relatively short time (on the order of several months) while they work on their research project or participate in a class, after which they might not use the system for a long period of time or may even move on entirely (e.g., graduate, change jobs, etc.). We estimate that only 5-10% of our overall community might log into the system in any given month. Users returning to the system, however, even after a long absence, still expect to have access to the digital artifacts they created which include SSH keypairs, images, notebooks, data, etc. Our migration strategy was thus based on the recognition that (1) the volume of users who will migrate during the rollout period will be relatively high, (2) the volatile community access pattern will lead to a long tail of migration operations as users return to the testbed, and (3) support staff may have to support migration-related tickets for years. Given this dynamic, we based our migration strategy on user self-migration that could be supported during the rollout as well as afterwards. Accordingly, we developed a self-migration tool; this took roughly an additional one FTE week to develop as compared to roughly 10 FTE months of design and development of the architecture described above.

This self-service data migration tool would prompt the user to log in to the target cloud with both the legacy and federated login method; then, it would systematically copy, share, or transfer ownership of the user's stored SSH keys, disk images, and server instances and leases, respectively, from their legacy to their federated account. Keystone's implementation of federation unfortunately necessitates such action. Internally, Keystone stores password users and federated users as separate records. This means that users of Chameleon now have multiple accounts in the OpenStack deployments. Projects function the same way: internally, projects created as part of mapping IdP groups to Keystone projects are stored separately to legacy local projects. Consequently, depending on which method a user used to login to the cloud, they would see that they

have access to the same set of projects (at least, when referenced by name), yet the saved data, e.g., server instances, leases, and disk images, would be entirely different! The migration tool effectively syncs the user's legacy account to their federated account, collapsing these two universes.

We implemented a staged rollout: beginning in October 2020, the new federated login experience was available on an opt-in basis for existing users via the self-migration tool. For some time leading up to launch day, we periodically ran scripts to sync users and project assignments from TAS to Keycloak to ensure the new system remained consistent with user changes. At launch, we also started requiring that new users enroll with federated identity to avoid dealing with more legacy users. One month after initial release, we switched federated identity from opt-in to opt-out; it was at this point that most existing active users created a federated account and performed their data migration. We supported the legacy login for existing users for another month, at which point it was entirely disabled. Over a three-month period, we had 267 users migrate their legacy accounts over, and 60 new users joined the system via federated enrollment; since the migration period, a total of 389 users have performed migration, following the predicted long-tail trend, with the first months of 2021 averaging 15 migrations per month, then leveling off, currently at a rate of 1-3 per month. The migration tool is still deployed to assist users returning after the initial migration window. Developing an on-demand, self-service migration tool reduced support load significantly: while some users required our help to merge their accounts, most were able to complete this alone. The development cost of FTE week was therefore more than offset by the reduction in support volume: staff handled only 25 migration-related tickets during the 2-month window. To increase visibility of the feature, we wrote a dedicated section in our documentation [32] to educate existing users about the change and guide them towards the migration tool and added informational banners that pointed to this documentation, both in the user portal and in the OpenStack GUIs, visible only for legacy users.

6 LESSONS LEARNED

Overall, the approach described here met all our goals (Section 4)—unlike the cheaper but ineffective alternatives we tried before (Section 3). A tiered system allowed us to "have our cake and eat it too" in that we were able to base our SSO implementation on an open source system with widespread mainstream adoption that ultimately provided the flexibility we sought—while the second tier supported the use of existing federated credentials and multiple identity providers needed to implement continuous migration in the short term, and offered the potential for integrating other IdPs in the long term. In particular, using an open source SSO implementation means that we have the freedom of extending it to support needs that cut across our various applications. For example, including human-friendly display names wherever a user's allocation ID is rendered, customizing the login page for greater usability (especially during a potentially confusing migration period), or providing context about whether a user's allocation is for research or education purposes (possibly influencing their default UI.) Further, Keycloak allows us to associate non-authenticating identities to user's accounts; this allows for future functionality leveraging integration with a user's private GitHub or GitLab repositories, or any other useful OAuth 2.0 services. Finally, using a managed IAM (e.g., COmanage's subscription tier) incurs an annual cost that our approach does not, though of course this is offset by the cost of implementing and operating our own IAM service.

These trade-offs may play out differently in different situations. A self-operated IAM incurs normal costs of secure and reliable operation (e.g., backups, upgrades, security patches, network firewalls, etc.), but also is a single point of failure and thus requires careful planning for high availability and disaster recovery; in our case, those costs are amortized given that we already operate a number of centralized services for the system. While working with open source implementation provides the potential for implementing extensions important to us, it of course also implies the need for such extensions and ability to implement them. Lastly, from the perspective of migration, an alternative might have been to operate TAS and federated identity system side by side for a while; this would have been difficult because of the variety of applications we support (Section 2.1), each biased towards a single login mechanism and thus requiring paying implementation costs many times over.

During migration, the most common pain point users encountered was when they changed which identity served as their primary Globus ID. In Globus, primary identities uniquely identify separate Globus accounts; i.e., changing the primary ID effectively creates a new account (users can subsequently re-link prior identities under this primary to allow multiple authentication methods.) In our case, our Keycloak IdP would similarly understand the user to be novel, and may trigger an error due to a collision on the user's primary email address, on which we enforce uniqueness. This was primarily an issue with users who belonged to institutions that supported both Google Auth and InCommon authentication for their domain, though there were other edge cases. In such events, Chameleon support staff manually reconciled accounts through a separate process we designed to prevent the risk of an account takeover attack. Globus provides all of a user's linked identities as part of their identity set at login time; in hindsight, we could have implemented more robust detection of this edge case. In practice, this happened infrequently enough that the support load was manageable, as we encountered roughly 10 such cases in total.

Crafting a login experience that was intuitive and non-invasive (fewer clicks and fewer surprises) took longer than we anticipated, and required several patches to Keystone and also the Horizon GUI; the default configuration for federated login, in particular the Web-SSO flow used by Horizon, introduced several unnecessary steps, such as showing an interstitial form asking the user to confirm their choice to use the federated login method. Supporting multiple login methods (legacy and federated) for the initial months of the rollout also increased development complexity significantly. As we were optimizing for ease of use for a research community numbering in the thousands, these costs felt justified.

One consequence of supporting more identity methods and streamlining user enrollment was that we started to see more spurious accounts in our accounting system. These typically belonged to students or researchers who were interested in learning more about Chameleon but did not proceed to be added to an allocation or request PI status. While the security of the testbed was never threatened because these users by design could not access testbed

resources or interfere with other users, we nonetheless had to reconsider the definition of an "active user" and make changes to our reporting to understand usage, as well as periodically normalize our user corpus to address, e.g., users signing in with Google Auth method and entering their host institution information in a free-text field, often in different forms (this could not be automatically discerned via the user's Google account, as opposed to, e.g., InCommon).

7 RELATED WORK

To our knowledge, this paper presents the first description of continuously migrating a large-scale existing cloud deployment and community from a legacy username/password login to federated identity.

Several scientific high-performance computing (HPC) systems have added support for federated identity in addition to local login [6] [26] [30], including support for multiple IdPs and linking identities, yet these leverage existing in-house authorization services (i.e., not mainstream open-source) and did not require a transition to federated identity or describe a migration. One example of a complete federated identity and SSO solution leveraging the CILogon platform for both authentication and authorization is described in [27], but does not address the CLI use case or discuss strategies for moving to a federated identity system. None of the aforementioned discuss how to integrate applications such as OpenStack and challenges posed.

One frequently-deployed architecture for IAM in cyberinfrastructures is CILogon and COmanage, the former handling authentication and SSO, the latter handling user enrollment and authorization policies [12]. We evaluated COmanage for our two-tiered approach and opted for Keycloak as a cost-effective alternative with a simpler operations footprint; while a self-hosted COmanage deployment requires a number of services ranging from LDAP, a registry server, DB, and potentially other services for group management or identity linking [18], Keycloak consists of a single server and DB, with all configuration and data exportable as simple JSON records. Keycloak overall has a similar feature set, but a larger user and contributor base [19], and thus more experience to draw from with regards to ongoing maintenance and lessons learned, and also more opportunity for leveraging future contributions. At the same time, COmanage is likely to offer better support for research cyberinfrastructure use-cases in its default configuration, and the paid subscription service will offset operational costs.

8 CONCLUSIONS

We presented a two-tier architecture for a single sign-on system with federated identity, implemented with commodity open-source software, which has allowed us to combine the benefits of authentication via federated identity with the need to retain flexibility within the domain of the Chameleon testbed. This flexibility is required to provide fine-grained authorization policies for our users based on such considerations as usage, temporary access (e.g., for artifact evaluation), or host institution requirements. This architecture also allows us to support multiple identity providers, a capability that was key to implementing continuous migration strategy for our community given its volatile usage pattern.

We found this approach to be successful in that it enabled us to achieve our goals of moving the system to federated identity while retaining efficient control over system-specific authorization, and accomplish continuous migration of a large community, with hundreds of thousands of digital artifacts in the care of our system, in a way that was relatively painless for both users and operators. The trade-offs of this approach include primarily identity collisions stemming from one entity using different federated identity accounts; these were both relatively few and easily fixed by a manual reconciliation based on a carefully designed process. Otherwise, our lessons learned were focused primarily on the effects of adoption of federated identity, in particular less reliable information about our users (who may e.g., be using Google accounts without listing their institutional affiliation) and increased proportion of users logging into the system without actually using it in the absence of the additional barrier of creating an account. Both are unavoidable and a result of greater benefits, i.e., more ubiquitous and easier access.

ACKNOWLEDGMENTS

Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation. This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357.

REFERENCES

[1] Jason Anderson. 2020. Blazar Specs: Flexible reservation usage enforcement. https://review.opendev.org/c/openstack/blazar-specs/+/707042
[2] Jason Anderson. 2021. Keystone federation++. https://diurnal.st/2021/07/17/openstack-keystone-federation-part-1.html
[3] Jason Anderson. 2022. *Migrating towards Single Sign-On and Federated Identity (PEARC'22): Notes and Sample Configurations.* https://doi.org/10.5281/zenodo.6582389
[4] Jason Anderson and Kate Keahey. 2019. A Case for Integrating Experimental Containers with Notebooks. In *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom).* IEEE, Sydney, NSW, Australia, 151–158. https://doi.org/10.1109/CloudCom.2019.00032
[5] Jim Basney, Terry Fleury, and Jeff Gaynor. 2014. CILogon: A federated X. 509 certification authority for cyberinfrastructure logon. *Concurrency and Computation: Practice and Experience* 26, 13 (2014), 2225–2239.
[6] Jim Basney, Terry Fleury, and Von Welch. 2010. Federated Login to TeraGrid. In *Proceedings of the 9th Symposium on Identity and Trust on the Internet* (Gaithersburg, Maryland, USA) *(IDTRUST '10).* Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/1750389.1750391
[7] Mark Berman, Jeffrey S. Chase, Lawrence Landweber, Akihiro Nakao, Max Ott, Dipankar Raychaudhuri, Robert Ricci, and Ivan Seskar. 2014. GENI: A federated testbed for innovative network experiments. *Computer Networks* 61 (3 2014), 5–23. https://doi.org/10.1016/J.BJP.2013.12.037
[8] Chameleon. 2021. Setting Up an Associate Site: An Interview With Northwestern University. https://www.chameleoncloud.org/blog/2021/02/19/setting-associate-site-interview-northwestern-university/
[9] Chameleon. 2022. Chameleon Cloud Documentation: PI Eligibility. https://chameleoncloud.readthedocs.io/en/latest/user/pi_eligibility.html
[10] Chameleon. 2022. Chameleon Main Page. https://chameleoncloud.org/
[11] Chameleon. 2022. CHI@NCAR: An Interview with the Newest Associate Site. https://www.chameleoncloud.org/blog/2022/02/28/chincar-an-interview-with-the-newest-associate-site/
[12] Jeffrey S. Chase and Ilya Baldin. 2021. Federated Authorization for Managed Data Sharing: Experiences from the ImPACT Project. In *2021 International Conference on Computer Communications and Networks (ICCCN).* 1–10. https://doi.org/10.1109/ICCCN52240.2021.9522208
[13] Jan De Clercq. 2002. Single sign-on architectures. In *International Conference on Infrastructure Security.* Springer, 40–58.
[14] Marek Denis, Jose Castro Leon, Emmanuel Ormancey, and Paolo Tedesco. 2015. Identity federation in OpenStack-an introduction to hybrid clouds. In *Journal of Physics: Conference Series*, Vol. 664. IOP Publishing, 022015.
[15] OpenInfra Foundation. 2022. OpenStack Blazar. https://docs.openstack.org/blazar/latest/

[16] OpenInfra Foundation. 2022. OpenStack Horizon. https://docs.openstack.org/horizon/latest/

[17] Laurel L Haak, Martin Fenner, Laura Paglione, Ed Pentz, and Howard Ratner. 2012. ORCID: a system to uniquely identify researchers. *Learned publishing* 25, 4 (2012), 259–264.

[18] Internet2. 2022. COmanage Registry Deployment Guide. https://spaces.at.internet2.edu/display/COmanage/COmanage+Registry+Deployment+Guide

[19] Internet2. 2022. COmanage registry GitHub. https://github.com/Internet2/comanage-registry

[20] Project Jupyter. 2022. JupyterHub: A multi-user version of the notebook designed for companies, classrooms and research labs. https://jupyter.org/hub

[21] Project Jupyter. 2022. OAuth + JupyterHub Authenticator = OAuthenticator. https://github.com/jupyterhub/oauthenticator

[22] Kate Keahey, Jason Anderson, and Michael Sherman. 2022. CHI-in-a-Box: Reducing Operational Cost of Research Testbeds. In *Proceedings of the Practice and Experience in Advanced Research Computing* (Boston, MA, USA) *(PEARC '22)*. Association for Computing Machinery.

[23] Kate Keahey, Jason Anderson, Michael Sherman, Zhuo Zhen, Mark Powers, Isabel Brunkan, and Adam Cooper. 2021. Chameleon@Edge Community Workshop Report. https://doi.org/10.5281/zenodo.5777344

[24] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S. Gunawi, Cody Hammock, Joe Mambretti, Alexander Barnes, François Halbah, Alex Rocha, and Joe Stubbs. 2020. Lessons Learned from the Chameleon Testbed. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. USENIX Association, online, 219–233. https://www.usenix.org/conference/atc20/presentation/keahey

[25] Kate Keahey, Pierre Riteau, Dan Stanzione, Tim Cockerill, Joe Mambretti, Paul Rad, and Paul Ruth. 2019. Chameleon: A Scalable Production Testbed for Computer Science Research. In *Contemporary High Performance Computing*. CRC, Boca Raton, 123–148. https://doi.org/10.1201/9781351036863-5

[26] Lee Liming, Ian Foster, and Steven Tuecke. 2015. Building Bridges from the Campus to XSEDE. In *2015 IEEE International Conference on Cluster Computing*. IEEE, 865–868.

[27] Shawn McKee, Benjeman Meekhof, Ezra Kissel, Andrew Keen, Kenneth M Merz, and Micheal Thompson. 2020. OSiRIS: A Distributed Storage and Networking Project Update. In *EPJ Web of Conferences*, Vol. 245. EDP Sciences, 04012.

[28] OpenStack. 2022. Keystone Edge Architectures. https://wiki.openstack.org/wiki/Keystone_edge_architectures

[29] Paul Ruth, Kate Keahey, Mert Cevik, Zhuo Zhen, Cong Wang, and Jason Anderson. 2021. Overcast: Running Controlled Experiments Spanning Research and Commercial Clouds. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE.

[30] Craig A Stewart, Timothy M Cockerill, Ian Foster, David Hancock, Nirav Merchant, Edwin Skidmore, Daniel Stanzione, James Taylor, Steven Tuecke, George Turner, et al. 2015. Jetstream: a self-provisioned, scalable science and engineering cloud environment. In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*. 1–8.

[31] TACC. 2022. TACC User Portal. https://portal.tacc.utexas.edu/

[32] Chameleon Team. 2022. Chameleon Cloud Documentation: Migrating to Federated Identity. https://chameleoncloud.readthedocs.io/en/latest/user/federation/federation_migration.html

[33] Chameleon Team. 2022. Keycloak plugins for Chameleon IdP deployment (GitHub). https://github.com/ChameleonCloud/keycloak-chameleon

[34] Keycloak Team. 2022. Keycloak: Open Source Identity and Access Management. https://www.keycloak.org/

[35] 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 *2016 IEEE 12th International Conference on e-Science (e-Science)*. IEEE, Baltimore, MD, USA, 203–212. https://doi.org/10.1109/eScience.2016.7870901