TIER Campus Success Program Case Study: University of Maryland Baltimore County

UMBC and midPoint: Moving from Legacy to a Modern Identity Solution

Executive Summary

University of Maryland Baltimore County (UMBC) first deployed a custom IDMS solution in 2002. Written primarily in Perl, it consolidated information from the campus SIS and HR systems primarily to automate the account lifecycle, and included a web-based administration portal that could be used for user account self-service and onboarding. The functionality has been continuously extended since then, and it now pushes user and course data to more than thirty campus systems.

The age of the system and its reliance on a completely custom codebase is beginning to be a liability. New technologies and desired features are being introduced at an increasing pace, meanwhile security concerns are more critical. It is increasingly difficult for development of the UMBC IDMS to keep pace. UMBC has examined several products over the years that purported to be capable of replacing the custom core of the IDMS system, and thus shifting much of the development burden to a third party. None of the available offerings were fully able to meet our needs until the arrival of midPoint. We have chosen to start using midPoint as a replacement for the current account provisioning system. After that is complete, our goal is to incrementally switchover to fully utilizing midPoint as the core of the entire UMBC IDMS infrastructure.

As a result of our participation in the Campus Success Program, UMBC has taken the first steps toward modernizing and upgrading an existing, deeply integrated, but aging custom IAM solution using midPoint: a community-accepted, best-in-class, open-source system.

Solution summary

UMBC is in the process of deploying midPoint to replace the account provisioning system. We expect to go live within the next quarter. Additional integrations are planned and will be deployed as they are ready.

TIER Feature Supported:

midPoint

Collaborators:

- UMBC

Community resources:

Internet2 TIER Slack channels - especially #tier-midpoint, Subject Matter Experts (SME) office hours, Campus Success Face-to-Face meeting

The Environment

UMBC has been involved in supporting and advancing middleware and identity management since the year 2000, when we were one of ten universities selected to be part of the Internet2 early adopter program. For UMBC, the early adopter program was transformative in that it has demonstrated the importance of identity management. UMBC has invested and used identity management as a key component of our institutional technology strategy.

Fast forward almost 20 years. The heart of our IAM environment is comprised of tens of thousands of lines of custom Perl code and an LDAP repository. The custom code is generally known by and maintained by one staff member. Data is messaged into the IAM system from various systems of record and populates our person, account, course, and enrollment entries in LDAP. The custom code is then responsible for account provisioning/deprovisioning in downstream systems, updating our learning management system (LMS), and numerous other entities across campus.

The Problem:

The current UMBC IAM system is completely custom written and maintained by one staff member. The current account creation system has several weaknesses:

- Somewhat fragile: Some edge cases can result in an incompletely created account.
- Non-transparent: The only methods to see account creation states are command-line only, and aren't comprehensively unified into a single interface.
- Non-extensible: Adding additional account creation endpoints is not as straightforward as one would hope.
- Somewhat inflexible: Inability to create only a subset of accounts rather than all of them.
- Old: The account creation system was originally written almost 20 years ago and is overdue for a complete overhaul.

blocked URL

myUMBC: Campus portal

webAdmin: A set of web services exposed to the campus portal for use in the account creation process.

AccountQD: Custom Perl code used to provision accounts to downstream systems.

The Solution

We expect that migrating to midPoint will solve all of these issues. midPoint has a robust engine that can be flexibly configured to handle various edge cases. It has an accessible GUI to view the status of a given account. The midPoint engine is designed to be easily extensible through the use of resource connector plugins, and flexible on what accounts are created based on configurable conditions. Finally, midPoint is under robust and active development with the potential of being widely adopted.

Our solution will integrate midPoint into our existing account management infrastructure as a core component. midPoint will be used to provision computing resources available to UMBC patrons, these include:

- Active Directory
- Unix Login
- AFS Home Directory
- Kerberos Authentication
- Google Apps

Our solution flows as follows:

- The account creation information, including the desired username and initial plaintext password, will be gathered from the customer by the myUMBC portal account creation front end.
- WebAdmin adds the information to midPoint using the midPoint REST API and a custom perl API wrapper.
- Using its configured connectors, midPoint will create the requested accounts. Errors and warnings are logged and notifications are sent as configured. If any step fails, the steps taken are configurable, and in most cases the failed account creation will be retried until successful.
- After all account creation actions report success, the plaintext password is securely deleted from midPoint and any desired additional steps can be taken, such as notifications.

blocked URL

The Result

Overall, the project has been successful and we are looking forward to continuing to integrate midPoint. UMBC will shortly be using midPoint in production to create user accounts across a variety of diverse systems. We now have an expertise with midPoint and are creating a detailed plan of discrete steps to incorporate midPoint as the core of our infrastructure. While the quantity of custom code has not been significantly reduced, the quality has been significantly improved, allowing for easier future feature expansion (i.e. account renaming). All custom code involved in the midPoint integration now incorporates full automated testing suites.

Lessons Learned

- midPoint is a relatively new product -- growing pains are to be expected. midPoint development is active and healthy; new versions have introduced new features and fixed bugs at a good pace. Documentation is still incomplete and there have been regressions during recent releases. We expect that these issues will improve as midPoint becomes a more mature product.
- We believe midPoint will be suitable as the core of our upgraded IDMS system. It appears to have a solid architecture and meets our needs.
- Ease of collaboration is very helpful.
- The usual: Time is always in short supply. Documentation is key.

Resources

- UMBC Perl midPoint REST API wrapper.
- Detailed account creation architecture diagram

Conclusions

IDMS systems are large and complex, touching and integrating systems and data from many disparate organizations, groups, and departments. As a consequence, making core structural changes can be a daunting task. It has been extremely helpful to us to break the changes into manageable chunks that can be completed individually. As knowledge, tools, and practices are improved, the quality and speed of implementing the remaining changes improves substantially.

UMBC and Grouper: Transitioning to DevOps and Provisioning to Google

Executive summary:

Given limited resources, UMBC was struggling to create and maintain a stable and up-to-date, production-class Grouper environment. One goal was to to integrate group collaboration tools associated with our portal, Box, and Google. By implementing the TIER deployment model via the TIER Grouper Docker containers, UMBC now has a healthy Grouper environment that is much simpler to maintain and ready to take advantage of orchestration and possible cloud migration. By implementing the Google Apps Grouper Provisioner UMBC is now auto-provisioning Google group membership based on campus portal group membership. With this experience, UMBC is comfortable in pursuing other similar integrations, such as Box.

Solution summary:

Use the TIER Grouper container to simplify patching, deployment, and scaling. Use TIER Grouper to auto-provision our campus portal groups into Google Apps groups.

TIER Feature Supported:

Grouper

Collaborators:

UMBC

Community resources:

Valuable resources used during this process were the February Face-to-Face meeting, the Grouper implementation guide, content-focused Slack channels, the Grouper working group, Chris Hubing, and John Gasper.

There were a few Grouper-specific demos that were helpful in getting us started and showing the possibilities.

The Environment

UMBC has been involved in supporting and advancing middleware and identity management since the year 2000, when we were one of ten universities selected to be part of the Internet2 Early Adopter program. For UMBC, the early adopter program was transformative in that it demonstrated the importance of identity management and the benefit of community participation. UMBC has invested and used identity management as a key component of our institutional technology strategy for service provisioning.

Until very recently, UMBC’s group management centered on our aging LDAP environment. This solution was inflexible,with no end-user provisioning tools, limited options for data input and output, and no ability for logic around group membership that wasn’t handled by the systems of record. Due to these shortcomings, our campus portal, myUMBC, maintained its own groups for use by campus clubs and organizations.

Trying to bring more functionality to our group management abilities, UMBC attempted to install, configure, and maintain a (pre-TIER) Grouper environment. Given limited resources the project became difficult to implement and adoption was slow at best. Our few “proof of concept” groups were the only groups we continued to maintain.

The Problem

Our pre-TIER Grouper deployment was small and was the result of many hours of manual installation and configuration. Trial and error was routine. Regular maintenance was neglected due to the effort needed and resources not being available to perform the tasks. As our desire for Grouper adoption grew, these issues had become problematic. For Grouper to be successful, we needed a better method of installation and general care and feeding of the system.

The myUMBC portal includes campus organizations and committee memberships. While the myUMBC group functionality works well, it lacks some of the features that a more robust environment, such as Google Apps, would provide, such as shared file storage and mailing lists. We needed a way t to leverage the features of Google Apps for myUMBC portal groups, but remove any need for manual provisioning and deprovisioning.

The Solution

UMBC has moved away from the manual installation, configuration, and patching of Grouper and has implemented the TIER Docker containers. Both web server and application server containers are running across multiple load-balanced servers.

UMBC is now able to auto-provision Google groups and memberships based on membership in myUMBC groups. Grouper loads myUMBC group membership into Grouper groups and then uses the Google Apps Grouper Provisioner to affect change within the Google groups. In the future, myUMBC will make use of Grouper web services to affect Grouper group membership rather than Grouper querying myUMBC every hour. Enhancements will also need to be made to myUMBC allowing group owners to opt in or out of Google provisioning.

blocked URL

The Result

UMBC now has a production-class Grouper environment deployed. Patches are easily applied when new TIER Docker containers are made available. Deployment errors have been greatly minimized. In the past, our environment was a complete manual configuration which was prone to errors. Implementing changes across numerous servers only multiplied the opportunities for mistakes. With the use of the TIER Docker containers we are assured that each server is a functioning copy of the other. The time from patch release to production is a fraction of what we had previously experienced. By using the TIER-provided Docker containers UMBC has positioned itself to make use of orchestration software such as Kubernetes to automate deployment, scaling, and management of our Grouper environment.

UMBC's auto-provisioning of myUMBC portal groups into Google, via Grouper has the potential for big wins. Being a Google Apps school our students and staff members want to utilize Google Drive and mailing lists with their groups. Seamlessly integrating Google Apps into our campus portal gives additional functionality to our users without having to reinvent the wheel or maintain a new codebase within our portal. This has set the stage for future portal group expansion. At this time we are looking to possibly provision from Grouper to Box.

Lessons Learned

Don't wait to ask the community for help. There are a lot of community members willing to provide assistance. When setting up your Grouper environment, take time to plan out your folder structure. We found it very helpful to ask to see the structure of some mature Grouper environments. A picture is worth a thousand words!

Resources

- Grouper Deployment Guide
- Google Apps Grouper Provisioner
- CSP focused Slack channels.

Conclusions

The Campus Success Program has been extremely valuable in moving UMBC to a more effective IAM life cycle. We will be able to spend more time implementing additional functionality and less time maintaining the application.

DevOps and Shibboleth at UMBC

Executive Summary:

University of Maryland Baltimore County (UMBC) first deployed a Shibboleth Identity Provider (IdP) in 2007, and joined the InCommon Federation in 2008. Over the years, usage of our IdP has increased dramatically as we have added numerous services that make use of web single sign-on (SSO), both in-house and in the cloud. Today, the IdP is one of our busiest and most widely-used services, with 24/7 uptime, handling many thousands of logins per day.

Prior to summer 2018, UMBC ran three instances of the IdP on standalone virtual machines (VMs) behind a load balancer. All configuration changes were handled manually and propagated to each of the server instances by hand. This process was error-prone and often led to inconsistencies between the individual servers, and errors occasionally would result in service outages. The increasing popularity of the IdP exacerbated the situation, as frequent configuration edits often were required when onboarding new cloud services, and more and more critical university services relied on the IdP, making outages increasingly painful. We needed a more consistent and reliable way to handle change management and reduce the likelihood of outages. We found that TIER's packaged Shibboleth IdP and its DevOps-driven philosophy provided an effective means to achieve these goals.

Solution Summary: Migrate UMBC's Shibboleth Identity Provider from a standalone VM environment to a solution based on the TIER Shibboleth Identity Provider platform. TIER provides a customized version of the IdP that runs inside a Docker container, which gives us a path to switch from our legacy VMs to a more modern deployment based on the principles of DevOps. Through this, we aim to achieve our stated goals of increased reliability, consistency, maintainability, and uptime; which in turn will allow us to better serve our campus community.

TIER Feature Supported:

Shibboleth Identity Provider (IdP)

Collaborators:

University of Maryland Baltimore County

Paul Caskey, SME

Community resources:

TIER Packaging Working Group

TIER DevOps Deployment Guide Working Group

The Environment

UMBC's Division of Information Technology (DoIT), Identity Management Group, is responsible for day-to-day operations of the Identity Provider. The IdP provides web single sign-on (SSO) for approximately 100 services, roughly 80% of which are cloud-based, and the rest hosted on-site. Our most heavily used SSO services are our campus portal (in-house), Blackboard LMS (cloud), Google Mail and Calendar (cloud), Microsoft Office 365 (cloud), RT Trouble Ticketing System (in-house), and Box (cloud). A single full-time staff member handles most of the IdP maintenance and troubleshooting, as well as single sign-on integrations with cloud providers.

The Problem

Our IdP currently is deployed across three load-balanced VMs. Each VM contains an entire standalone Shibboleth IdP, as well as an instance of WebAuth, our legacy SSO product. Configurations are maintained in a Subversion repository. There is no change management or automation strategy in place; all configuration changes are made manually on one node, and then propagated to the others by hand. Nodes often end up out-of-sync, particularly if one is unavailable at the time the changes are deployed (in these cases, the affected node will be out-of-sync from the time it is restored to service until an administrator brings its configuration up to date). Service restarts are handled manually as well, and can be cumbersome, as each node must be taken out of the load balancer and restarted in sequence to avoid service outages. Our hope is that migrating to a DevOps based deployment will help to mitigate some or all of these shortcomings.

The Solution

With a well-established legacy infrastructure that has been in place for over 10 years, we felt that a stepwise approach would be the best way to approach our eventual migration to DevOps. We first worked to prepare our existing environment by applying patches, upgrading the IdP to the latest version, etc. The next goal was to bring up a TIER IdP running alongside our non-TIER IdPs, with our external load balancer routing traffic to it. This is where our project currently stands (see Fig. 1). Eventually, we will phase out the non-TIER IdPs and replace the external load balancer with a container orchestration service such as Swarm or Kubernetes.

The diagram below shows our current IdP architecture, including the containerized TIER instance:

blocked URL

The Result

While the project is by no means complete, our TIER IdP has been running successfully in production since late August 2018, handling 90% to 95% of our production single sign-on traffic during the start of our fall semester, historically our busiest time of year in terms of login volume. We have observed no noticeable performance degradation over this period, and performance of the containerized IdP seems to be on par with or better than our legacy environment. Full benefits of the DevOps environment won't be realized until we've phased out our legacy VMs, which we expect to happen sometime later in the fall. Based on initial testing with Docker Swarm, several challenges remain to be resolved prior to launching a container orchestration system, mainly centered around load balancing and IP forwarding. Container orchestration is the last step of our DevOps deployment plan, and likely won't happen until early 2019.

Lessons Learned

Clustering is hard to get right, particularly with an external authentication system (such as CAS) in play. In the early 2000s, we clustered our IdPs out of necessity, which introduced an extra layer of complexity by requiring additional configuration on our load balancer. In more recent years, we've switched to more modern hardware, and found that a single IdP node is capable of handling all of our SSO traffic. We now are routing all of our IdP traffic to a single node, with additional nodes running in a failover capacity only. We have found this to be simpler and more reliable than full clustering.

In the future, we plan to deploy one or more instances of the IdP in AWS. Rather than clustering the off-site and on-site instances together, we likely will "segment" traffic according to its originating IP address, routing on-campus traffic to the on-site node and off-campus to AWS.

Resources

- TIER DevOps Deployment Guide (work in progress)
- TIER Package Delivery
- Shibboleth Consortium
- Shibboleth at UMBC

Conclusions

We feel that TIER and DevOps will transform our IdP and single sign-on environment for the better, making it more robust, consistent, and fault tolerant. However, migrating from an existing, long-lived legacy environment is a considerable task, and care must be taken to form a solid migration strategy. We had greatest success with a stepwise plan that slowly introduced TIER and DevOps components alongside legacy components, coupled with a slow phase-out of the latter.

About UMBC