University of Oregon’s innovative Federation model allows different organizations to share one Puppet environment

Industry: Education

Background
The University of Oregon has historically been a decentralized IT campus, where each college would handle their own technology needs while Information Services would provide centralized services like Active Directory and OpenLDAP, email, and Banner ERP.

Challenge
The success IS had as a result of using Puppet led to many departments wanting to leverage this tool for similar purposes, however there would be a need to provide a consistent platform at scale while accommodating the disparate needs of other departments and maintain stability across the environment while respecting other departments’ desire for autonomy. All of this and avoid being the blocker of rapid innovation and not be inundated with support requests.

Customer Environment
The University uses a “delegated administration” (or federation) model where 12+ teams of admins and developers are responsible for their own tech stacks, while the University’s Systems Automation Services’ (SAS) within Information Services provides a consistent platform (Windows or Red Hat Linux with base profiles for central authentication, logging, DNS settings, etc.) that is maintained at scale

Solution
• Puppet Enterprise (PE) for its web UI and RBAC controls
• Puppet Bolt and tasks within Puppet Enterprise
• In addition to PE, Continuous Delivery for Puppet Enterprise (CD4PE) to leverage its features to enhance federation model

Results
• Possibility of human error reduced to nearly zero
• Quicker infrastructure and environment build times.
• Time to deploy VMs reduced from days to 8 minutes
• Increased ability to meet customer demands and compliance needs
Background

In an age where collaboration empowers higher education institutions to leverage experiential learning from one another, the University of Oregon is leading the way in successfully using a “delegated administration” model (also known as a federation model) that equips individual departments to manage their own tech stacks while assuring centrally-managed consistent configuration.

The University of Oregon has historically been a decentralized IT campus, where each college would handle their own technology needs while Information Services would provide centralized services like Active Directory and OpenLDAP, email, and Banner ERP. A decade ago, the campus started centralizing services like web hosting, security scanning, virtual machine hosting (VMWare), then moved toward further centralization to reduce waste and improve efficiency.

This led to a consolidation of Tier 1 support across campus and spawned initiatives to improve their service catalog and increase availability to campus customers. A core tenet of this centralization is to standardize as much as possible while still allowing needed customization at the application layer.

The University uses a “delegated administration” (or federation) model where 12+ teams of admins and developers are responsible for their own tech stacks, while the University’s Systems Automation Services’ (SAS) within Information Services provides a consistent platform (Windows or Red Hat Linux with base profiles for central authentication, logging, DNS settings, etc.) that is maintained at scale.

Challenge

Information Services was an early adopter of Open Source Puppet since they needed to make changes at scale that were self-documenting, easy to modify, and commitable. The success they had as a result of using Puppet led to many departments wanting to leverage this tool for similar purposes. Questions soon arose:

• How does our small team continue to provide a consistent platform at scale while accommodating the disparate needs of our customers?
• How do we maintain stability across our environment while respecting the customers desire for autonomy?
• How can we avoid being the blocker of rapid innovation and not be inundated with support requests?

SAS attempted to use pull-requests with approvals to maintain consistency of the code base. This led to a higher support burden and slower iterative progress. It also still meant that if errant code was introduced into the environment, other groups would still be adversely impacted.
Solution
The SAS team ended up breaking out each department’s code into their own Git repositories. Each repo would then contain that group’s roles, profiles, hiera, and tasks. Permissions could then be set at the repository level giving only certain people or services access to a given department’s infrastructure code.

A custom Facter fact, “pp_role”, would be set on each host that serves two uses. First, in the service of catalog compilation to implement a given configuration on that machine. Secondly, “pp_role” is used to derive another fact, “pp_group”, which is how we determine ownership of each node for the purposes of RBAC and reporting.

“Using Code Manager and Puppet Enterprise, we’re able to partition each team’s code into their own repositories and pull them in dynamically during a deployment action. All their Puppet-related files live in that repo. This allows us to permission the repositories correctly, so that each group has write access to their own module and nobody else’s,” said Lucas Crownover, a lead Systems Administrator and Puppet architect for Information Services at University of Oregon. “It really was about solving a multi-tenancy problem.”

To make multi-tenancy as user friendly as possible, SAS uses Puppet Enterprise (PE) for its web UI and RBAC controls (using the “pp_group” Facter fact) so that delegated admins would have access to create reports, view their infrastructure’s status, and run tasks. To that end, they also associate “pp_group” to a particular Active Directory (AD) group so that a given administrator can only run tasks against nodes within their department. Specifically, a user can run a task against any node that includes a role that is within the repository of that group.

Puppet Bolt and tasks within Puppet Enterprise turned out to be a surprising hit with their customers: “Tasks and the ability to use the Orchestration API along with the PCP protocol have been a massive win in terms of providing tools to our developers without having to worry about firewalls, permissions, etc. Our PE server is effectively our bastion host (for scripts) and being able to write tasks that can be used by other users of PE all from one interface is great,” said Crownover.

The migration from Open Source Puppet to Puppet Enterprise took around six months to complete and required refactoring old code to adhere to Puppet 4 / Puppet 5 standards. In addition to PE, SAS also opted to use Continuous Delivery for Puppet Enterprise (CD4PE) to leverage its features to enhance their federation model.

CD4PE improves SAS’s overall quality of service by reducing compilation errors via syntax validation and allowing for testing of new code in isolation using feature branches. “The primary use of CD4PE in our environment is the use of the “feature_+” branch workflow. We’ve gotten a huge amount of praise for the implementation of this. It allows our developers to dynamically create their own sandboxes to test Puppet code without risk of causing group-wide compilation errors,” said Crownover.

To expedite the CD4PE rollout, SAS worked directly with a Puppet engineer on the implementation. “Our experience with the onsite Puppet engineer during our CD4PE rollout was an overall very positive experience. Our environment was set up in a very unique way for our needs and it was incompatible with how CD4PE operated. We completely restructured how we use our Puppet repositories during the two weeks the engineer was with us, and he was very patient during that time. We were able to get CD4PE running pretty quickly after finishing the work on our codebase and it’s still working today,” said Crownover.

What is CD4PE?
Continuous Delivery for Puppet Enterprise gives IT teams a holistic approach to managing infrastructure at scale. By eliminating the silos and helping teams accelerate application deployment timeframes, it provides a simple yet prescriptive way to build, test, promote and deploy infrastructure code in order to deliver changes faster and more reliably.
Results

“While I believe Open Source Puppet is a fantastic product that already serves the majority of needs, there are some huge benefits we’ve gained by moving to Puppet Enterprise,” said Crownover. Some of the more notable improvements the University has seen in their federation approach include:

- **Reporting.** Having a central interface for inspecting agent reports is very useful when troubleshooting an issue.
- **Orchestrator.** The Orchestration component is quickly gaining traction as their primary location for users to run automated tasks. They’ve also been making use of the Orchestrator API for our automated provisioning system (VRealize Automation) to trigger tasks.
- **Support.** While the University reports that they haven’t needed too much support, the times where they’ve submitted a ticket “have all been pleasant. Techs are quick to respond with helpful information and a pleasant attitude.”

“As of today, we’re completely rolled out with the latest version of Puppet Enterprise, the latest version of CD4PE, and all agents at the latest version,” said Crownover. The University reports that their long-term goals are to improve on their provided Puppet services (both desired state config and tasks) and reduce the learning curve of getting developers up to speed with Puppet.

“Puppet Enterprise was absolutely essential to meet our needs. It offers a nice user interface for various groups to log in and create reports, see the state of their machines, or run tasks related to their infrastructure” said Matthew Shepard, Associate Director, Systems Automation Services for Information Services, University of Oregon.

“While I believe Open Source Puppet is a fantastic product that already serves the majority of needs, there are some huge benefits we’ve gained by moving to Puppet Enterprise.”

Lucas Crownover,
Lead Systems Administrator and Puppet architect for Information Services,
University of Oregon