Sakai Admin Guide - Operations Best Practices

Operations Best Practices

Introduction

This section is intended to bring together a handful of best practices not covered elsewhere in the admin guide. These are not canonical, but represent the range of techniques production sites use to manage the uptime and performance of their sakai installations.

Maintenance Schedules

Although continuous availability is the ultimate goal, many sites find it useful to announce routine daily and/or weekly maintenance times to ensure that routine maintenance can be performed in an orderly fashion. A basic approach is to schedule a short daily window, and a longer window that occurs once a week. The daily maintenance window is scheduled for 15-30 minutes each day, which can be used for minor updates, restarts, log rotation, et cetera. The weekly maintenance window typically occurs on the slowest day of the week, and is long enough to allow for patches, cold database backups, version updates, hardware migrations, etc. Both the daily and weekly window should either be based on clearly observed usage patterns among your users, or on the scheduled down time of systems on which your installation depends (database, distributed storage), et cetera.

> ⚠ **Your downtime schedule should be clearly advertised on both the gateway site (which users see before logging in) and in each user's "My Workspace" site.**
>
> **This is critical to establishing the expectation among end users that the system will be unavailable during scheduled maintenance times.**

Another useful feature is to set up a separate apache installation that is the failover node for your load balanced cluster. This apache instance should present a page that reminds users of the published maintenance schedule, and also serves as a notification mechanism in explaining unexpected outages.

Hot Deployment of Changes

If you have a load balanced installation of Sakai, application changes can be hot deployed, although there are caveats. If you wish to take an application server out of the load balanced pool and then take it down, you must keep in mind the sticky session timeout for your sakai installation, and must carefully monitor the sticky sessions assigned to a node before shutting it down. Here's an example methodology for hot deployment of changes:

1. remove the node from the load balanced pool while continuing to allow current active connections
2. monitor sticky sessions until all users are off of the node
3. take down the node
4. patch or otherwise update the node
5. bring the node up and test individually
6. return the node to the load balanced pool
7. continue the process with the next node

If your session timeout is longer than an hour, you can imagine that it may take some time to ensure that all traffic is off of a node before restarting. If you choose to shorten the process by shutting down the node once the load falls below a certain threshhold, keep in mind that any users who remain connected to a node when it shuts down may lose their work in progress, including assessments. This is another benefit of clearly advertising your downtime schedule, it gives you a safer time in which to introduce changes.

Automated Deployment

If you are deploying Sakai with local customizations and configuration changes (as most sites do), you may find a benefit in creating an automated deployment process. This process would do something like:

- download the base version of Sakai from subversion
- overlay any local tools and code changes
- test the build and notify an administrator if there are failures
- stop Sakai
- deploy the updated build
- restart Sakai

The downloading and overlaying of code can be managed to a large extent by establishing your own subversion repository and using the svn:externals functionality to create a combined source directory. For more information on svn:externals, see:

The Subversion Documentation

An example of an automated build process was detailed in David Haines' talk at the 5th Sakai Conference in Atlanta:

http://bugs.sakaiproject.org/confluence/display/CONF06/The+University+of+Michigan+Build+Process

Security

Security updates are not publicly announced until there has been enough time for existing installations to address the problem. All security notifications are coordinated through Anthony Whyte. For more information, see the "Sakai Admin Guide - Joining the Community" section of this document or the "Documenting your Sakai instance as part of the Sakai community" section of the SakaiPedia.

Patches

You may find that your version of Sakai suffers from a known bug for which there is a patch. Applying a patch to a sakai installation requires a source build and the standard patch utility. The steps are typically as follows:

- Obtain the patch or updated source
- Shut down your sakai installation
- Remove any old versions of the affected project from $CATALINA_HOME/shared/lib, $CATALINA_HOME/components and $CATALINA_HOME/webapps.
~~undeploy the affected component using the Sakai plugin for Maven ("maven sakai:undeploy" from the component's source directory)~~.
- build and deploy the affected components using the Sakai plugin for Maven ("maven sakai" from the component's source directory).
- start your sakai installation

Monitoring

Many institutions choose to use programs like Nagios and Big Brother to monitor the health of their Sakai installation and notify key staff. These programs typically verify that all processes are running, and that Sakai is listening on the desired ports. There are also scripts to monitor the contents returned by a web request. One approach is to have a simple tool installed within Sakai that's accessible without a password return a summary of the status of the internal health of a Sakai install (database connections, memory, etc.).

If you are running in a load balanced installation, your load balancer may also offer tools to monitor the health of a node and either reduce the amount of load directed to the node over time, or flag the node as unavailable and redirect all traffic from that node. Where possible, you should tie your load balancer's monitoring to something that reflects the internal health of Sakai rather than whether it is simply up and perhaps not responsive. One suggestion would be to use a internal health check script as mentioned above and tie the load balancer to that. Another would be to test the response time of a node and flag it as down if it responds too slowly.

Notification

A key practice of any monitoring system is to quickly notify staff who can attend to the problem. Some institutions use a rotating on-call system where one staff member responds to all alerts during a given time period. Some institutions use a system in which the first person to respond to the problem indicates to everyone else that they are working on the problem. Some institutions with dedicated operators use a call tree that begins with first responders and continues through the organization.

Proactive review

It's also good practice to review the errors recorded in catalina.out regularly and compare to known bugs to help isolate configuration changes or previously unseen bugs. This is important as an ongoing activity, and as a part of quality assurance and change management.