# Sakai 10 and later System Requirements

## Operating system (OS) choices

**Sakai is OS neutral. It is typically run on any of the numerous Linux distributions.** Examples of common Linux distributions include as CentOS, Debian GNU/Linux, Fedora, Gentoo Linux, Red Hat Enterprise Linux (RHEL), SuSe Linux, Ubuntu. Sakai can also run on Mac OS X server, Microsoft Windows and Sun Solaris. Operating systems other than Linux are not nearly as well tested, and all of the community QA servers are running Linux, so this is generally the recommendation.

## Java

**Sakai 10 has been tested most thoroughly with Oracle's Java 7.** It also should still be binary compatible with Java 6, aka Java 1.6 . Requires JDK 6 to build "all code". Sakai 10 is currently not supported in Java 8 and it would advised against running or trying to build it as some tools will not work correctly

~~SAK-28091~~ - New problems (class file not found) compiling on Java 8 (1.8.0_25) RESOLVED

~~SAK-26078~~ - upgrade reflectutils to be JDK8 compatible VERIFIED . Certain files, such as *.jsp and *.jws, require compilation so downloading and attempting to use only the run time environment (e.g. JRE 7.0) will not suffice.

For **Sakai 10** (2014 release)

Targets Java 6+. QA'd with 7. Requires Java 6-7 to build.

Likely for **Sakai 11** (2015 release)

Targets Java 7+. **Likely** QA'd with JRE 8. Likely will require Java 7-8 to build.

What breaks JRE compatibility is when someone wants or needs to use new features of the language or newer libraries. What breaks JDK compatibility are generally internal interface changes where we'd have to do work to support 2 sets of source code.

Oracle's Sun Java J2SE 5.0 (a.k.a Java 1.5) has completed the EOL process and is no longer supported. If are still running Java 1.5 please note that security vulnerabilities exist in JDK/JRE 5.0 updates 1.5.0_17 and earlier.

## Application server choices

**Apache Tomcat 7 is the recommended, and most thoroughly tested servlet container, and is most often used with a web server like Apache Http Server.** Several schools run their Tomcats with Windows IIS and nginx proxies without issue. A few schools such as Hong Kong University of Science and Technology and the Universidad de Guadalajara report have deployed Sakai on JBoss, in the past but no support for this setup is provided. As of Sakai 10's release, Tomcat 8 is beta and has not been tested.

Sakai 2.8.0 and previous releases require Tomcat 5.5 out of the box but can be configured in custom builds to run under Tomcat 7. Tomcat 7 is the requirement for running Sakai 2.9.0+. There are some changes to the Tomcat configuration required to get Sakai to startup in the source or binary form under Tomcat 7. Please see this page for more information!

**Websphere deprecated**

In Sakai 2.7.0 a Websphere module was included in the release in order to facilitate deployment to a Websphere/Db2 production environment; however, support has waned and the Websphere option is currently considered deprecated.

## Database choices

**Sakai production installations typically run MySql 5.5 or later. Oracle is the next most popular choice. There are known installations of Sakai on Oracle 10g, 11g, and 12c.** MariaDB is a binary drop in replacement for Mysql (source) and **MariaDB should work with Sakai but it is not officially supported. There are known installations of Sakai on MariaDB 5.1 or 5.5** (for example : at Rutgers University - source). It should be noted that Sakai is not limited to these database choices and integration with other RDBMS systems is not difficult. In the past at least one installation used Microsoft SQL Server while requests for PostgreSQL integration are occasionally raised on the Sakai developers list. However, to date no one in the Sakai Community has stepped forward to support alternatives to either Oracle or MySQL, a prerequisite for adding additional database options to the release.

**IBM DB2 Deprecated**

Support for IBM Db2 was added for the Sakai 2.7.0 release but for 2.8.0 DB2 conversion scripts were not updated or tested and are currently considered deprecated, including for 2.9.x.

A demo version of Sakai includes HSQLDB; it should never be deployed in production.

## Clustering, file storage and load balancing strategies

A typical Sakai cluster is comprised of one or more application servers running one or more instances of Tomcat 7 operating either in standalone mode or behind the Apache HTTP 2.2 web server. Each Tomcat instance runs a full copy of Sakai. The cluster is backed by a single database providing a transactional store of information.

Storing binary content outside the database is a configurable option in Sakai and highly recommended from a performance standpoint. Most Sakai schools with sizable user populations opt for network-attached storage (NAS) or storage area network (SAN) solutions. Load balancing is provided by Apache (using mod_jk, mod_proxy_balancer or mod_proxy_ajp) or dedicated hardware solutions such as F5 BIG-IP, NetScaler or Zeus.

## External Authentication choices

Sakai can integrate with a variety of external authentication services including CAS, Kerberos, LDAP, Shibboleth and WebAuth.

## Integrating with student information systems

Sakai Community institutions have integrated their Sakai installations with Banner, Datatel and Peoplesoft as well as a variety of home-grown student information systems (SIS).

Sakai has two basic approaches to integrating data from external systems. Most sites use a combination of these approaches. The first approach is to use the internal Sakai "provider" APIs. These APIs are places for Sakai to "consult" while Sakai is running. There are APIs for User Identity, User Directory, Course Listing and User Roles.

**User Identity API:** allows Sakai to call local code to validate users when they log into the system. This commonly uses Kerberos, Active Directory or LDAP to validate the user's credentials.

**User Directory API:** allows user information such as name and e-Mail address to be retrieved from an external system such as LDAP or X.509. The User Directory API has provisions to allow the local site to make decisions when to display student information in order to meet FERPA requirements. Each institution has different interpretation of FERPA so the precise FERPA decisions are delegated to the User Directory API.

**Course Listing API:** consulted when the instructor is creating a course site - this API returns the list of externally stored rosters for which the current user is the instructor. The user can select from one or more of these external rosters to associate with the course they are creating.

**User Role API:** is consulted when users log in to determine which external rosters they user is a member of and what their role is within those rosters. The Sakai internal configuration is updated if there are any changes to an individual's roster status.

The above API's are "pull" APIs--they are consulted when the user logs in or tries to take some action. The Course List API described above does not auto-populate courses.

If there is a desire to "push" information into Sakai, there are two approaches - Quartz and web services.

Sakai utilizes an internal batch system called Quartz that provides a cron-like capability within Sakai. Quartz is used by creating a Java class that does the necessary work and then having Quartz schedule the regular execution of that Java code.

A more common approach to pushing data (e.g. enrollment and course information) into Sakai is through web services. Many of Sakai's APIs can be accessed by web services. Web service access points have been developed for many of the common Sakai APIs used for configuration. These REST and SOAP web services can be called from PHP, Python, Perl, Java, .NET or any other language. The Sakai web service data structures are kept simple to insure the widest possible interoperability with as many languages as possible. Administrators often build scripts to pull data from their SIS system and populate Sakai with that data. These scripts may be automated using cron or manually executed by the administrator at the proper time during a semester.

This combination of pull/push configuration capabilities allows for a very wide range of integration possibilities for Sakai.