**Chapter 7. Securing Jenkins**

**Introduction**

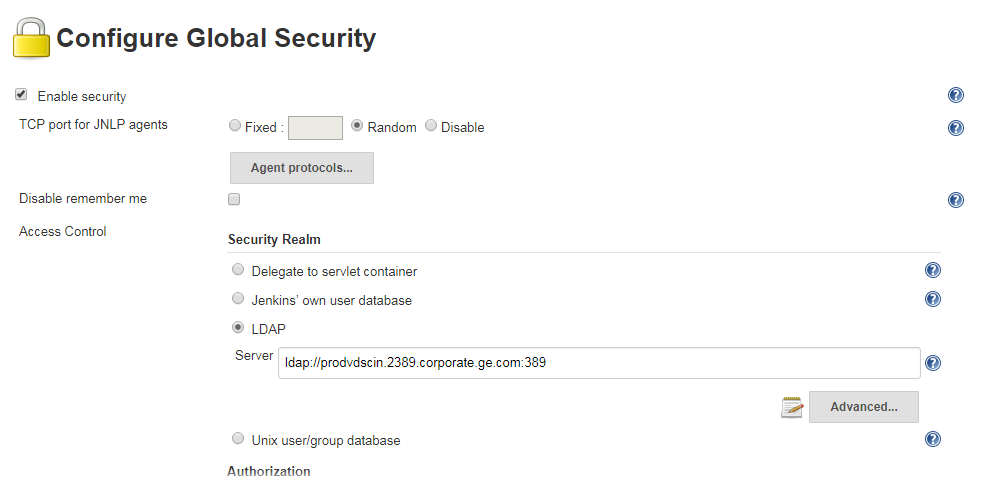
Jenkins supports several security models, and can integrate with several user repositories. In smaller organizations, where developers work in close proximity, security on your Jenkins machine may not be a large concern—you may simply want to prevent unidentified users tampering with your build job configurations. For larger organizations, with multiple teams, a stricter approach might be required, where only team members and system administrators are allowed to modify their build job configurations. And in situations where the Jenkins server may be exposed to a broader audience, such as on an internal corporate website, or even on the Internet, certain build jobs may be visible to all users whereas others will need to be hidden to unauthorized users.

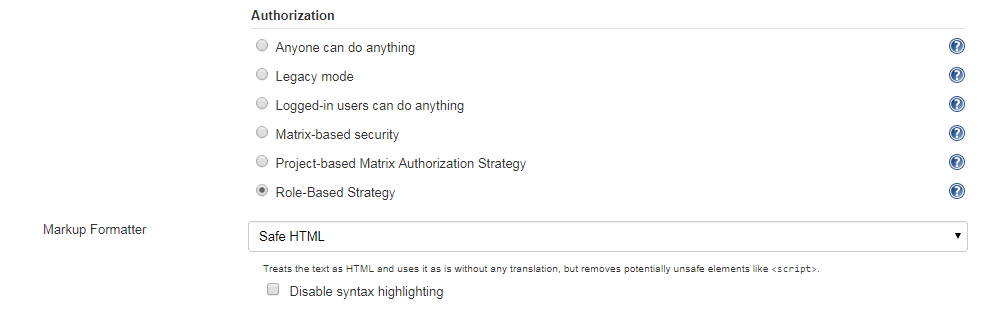
In this chapter, we will look at how to configure different security configurations in Jenkins, for different environments and circumstances.

# Activating Security in Jenkins

Setting up basic security in Jenkins is easy enough. Go to the main configuration page and check the Enable security checkbox (see [Figure 7-1](https://www.safaribooksonline.com/library/view/jenkins-the-definitive/9781449311155/ch07s02.html#fig-hudson-enabling-security)). This will display a number of options, that we will investigate in detail in this chapter. The first section, Security Realms, determines where Jenkins will look for users during authentication, and includes options such as using users stored in an LDAP server, using the underlying Unix user accounts (assuming, of course, that Jenkins is running on a Unix machine), or using a simple built-in user database managed by Jenkins.

The second section, Authorization, determines what users can do once they are logged in. This ranges from simple options like “Anyone can do anything” or “Logged-in users can do anything,” to more sophisticated role and project-based authorization policies.





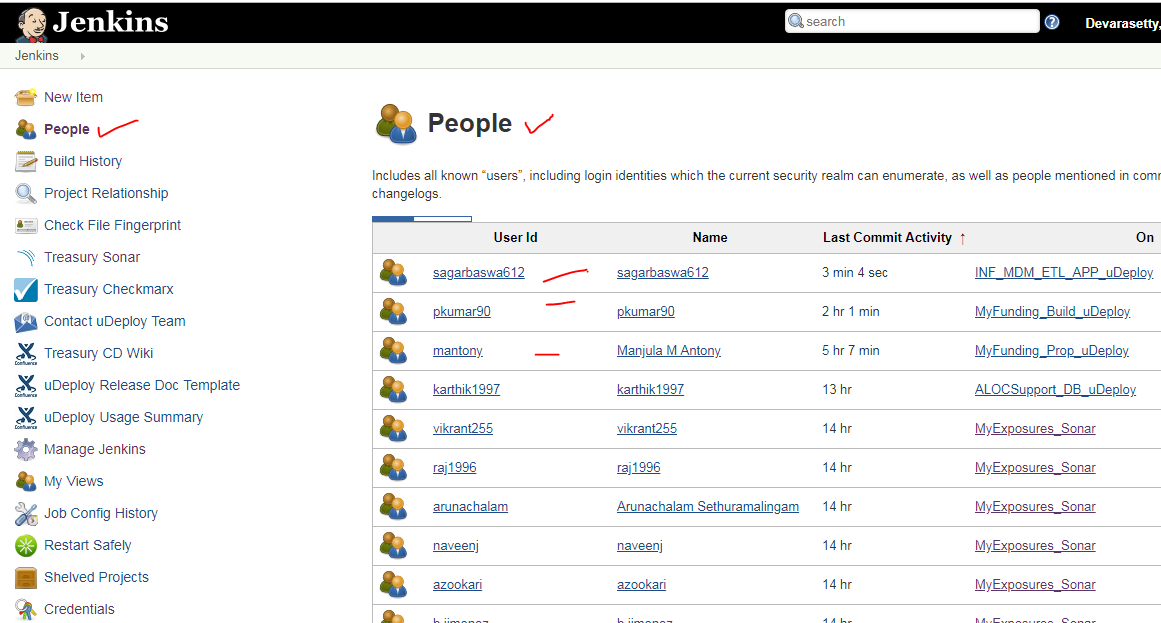
# Security Realms—Identifying Jenkins Users

Jenkins lets you identify and manage users in a number of ways, ranging from a simple, built-in user database suitable for small teams to integration with enterprise directories, with many other options in between.

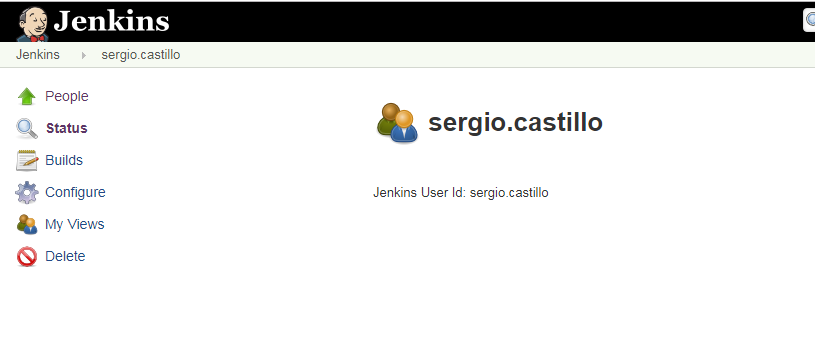
## Using Jenkins’s Built-in User Database

The easiest way to manage user accounts in Jenkins is to use Jenkins’s internal user database. This is a good option if you want to keep things simple, as very little setup or configuration is required. Users who need to log on to the Jenkins server can sign up and create an account for themselves, and, depending on the security model chosen, an administrator can then decide what these users are allowed to do.

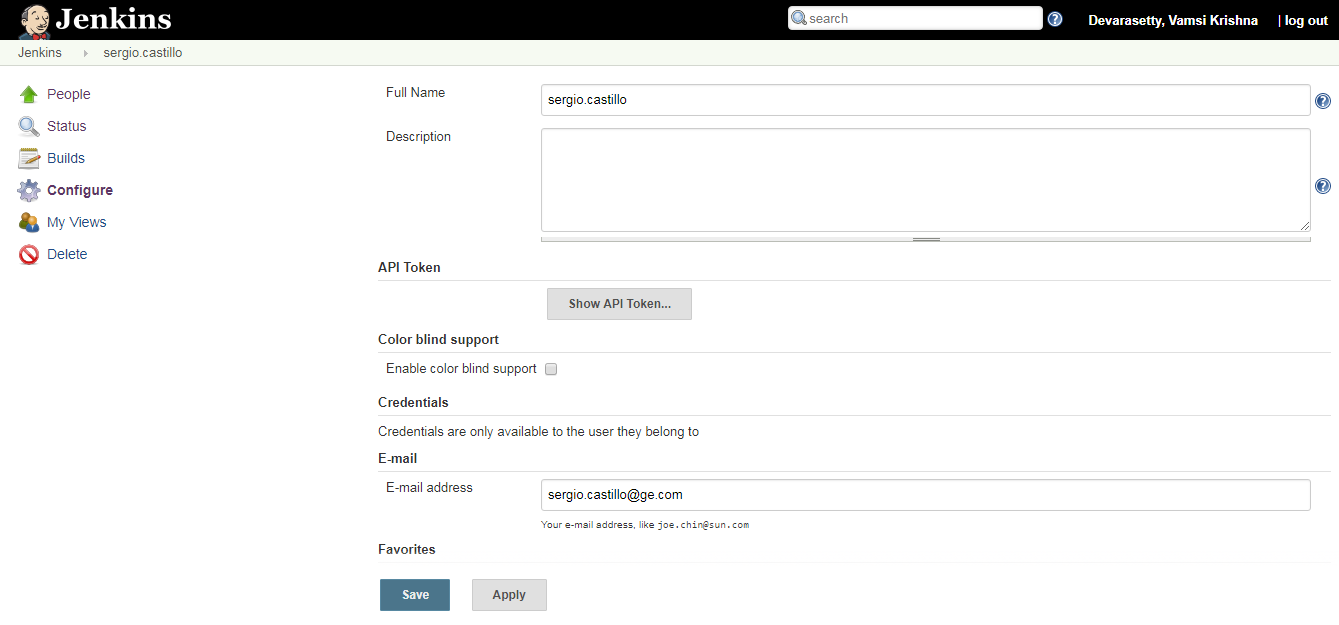
Jenkins automatically adds all SCM users to this database whenever a change is committed to source code monitored by Jenkins. These user names are used mainly to record who is responsible for each build job. You can view the list of currently known users by clicking on the People menu entry (see [Figure 7-3](https://www.safaribooksonline.com/library/view/jenkins-the-definitive/9781449311155/ch07s04.html#fig-hudson-user-list)). Here, you can visualize the users that Jenkins currently knows about, and also see the last project they committed changes to. Note that this list contains all of the users who have ever committed changes to the projects that Jenkins monitors—they may not be (and usually aren’t) all active Jenkins users who are able to log on to the Jenkins server.



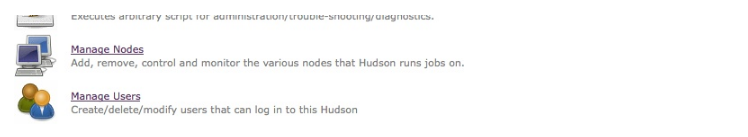
If you click on a user in this list, Jenkins takes you to a page displaying various details about this user, including the user’s full name and the build jobs they have contributed to (see [Figure 7-4](https://www.safaribooksonline.com/library/view/jenkins-the-definitive/9781449311155/ch07s04.html#fig-hudson-user-builds)). From here, you can also modify or complete the details about this user, such as their password or email address.



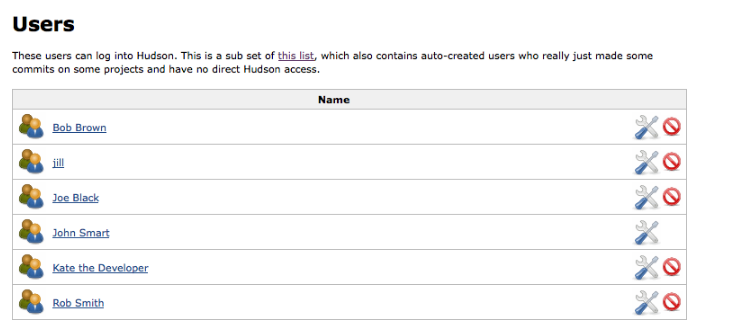
A user appearing in this list cannot necessarily log on to Jenkins. To be able to log on to Jenkins, the user account needs to be set up with a password. There are essentially two ways to do this. If you have activated the “Allow users to sign up” option, users can simply sign up with their SCM user name and provide their email address and a password (see [Simple Security in Jenkins](https://www.safaribooksonline.com/library/view/jenkins-the-definitive/9781449311155/ch07s03.html)). Alternatively, you can activate a user by clicking on the Configure menu option in the user details screen, and provide an email address and password yourself



Another way to manage the current active users (those who can actually log on to Jenkins) is by clicking on the Manage Users link in the main Jenkins configuration page



From here, you can view and edit the users who can log in to Jenkins (see [Figure 7-8](https://www.safaribooksonline.com/library/view/jenkins-the-definitive/9781449311155/ch07s04.html#fig-hudson-user-database)). This includes both users that have signed up manually (if this option has been activated) and SCM users that you have activated by configuring them with a password. You can also edit a user’s details (for example modifying their email address or resetting their password), or even remove them from the list of active users. Doing this will not remove them from the overall user list (their name will still appear in the build history, for example), but they will no longer be able to log on to the Jenkins server.

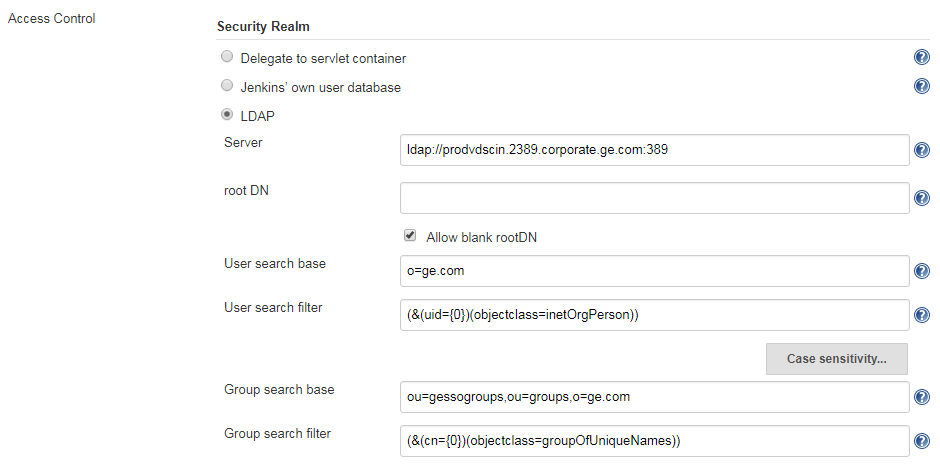


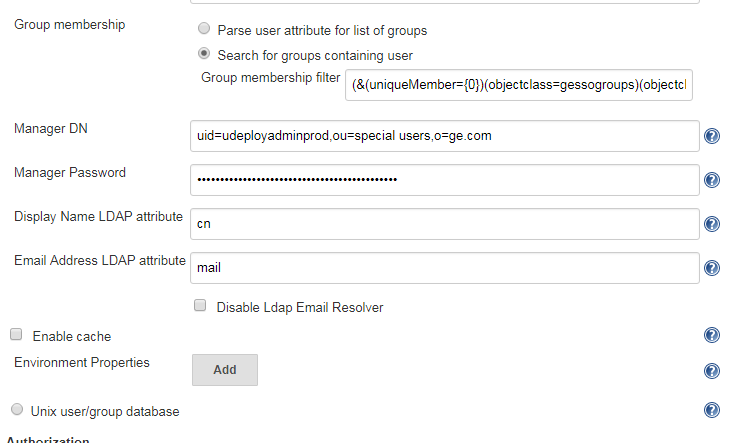
The internal Jenkins database is sufficient for many teams and organizations. However, for larger organizations, it may become tedious and repetitive to manage large numbers of user accounts by hand, especially if this information already exists elsewhere. In the following sections, we will look at how to hook Jenkins up to other user management systems, such as LDAP repositories and Unix users and groups

## Using an LDAP Repository

Many organizations use LDAP directories to store user accounts and passwords across applications. Jenkins integrates well with LDAP, with no special plugins required. It can authenticate users using the LDAP repository, check group membership, and retrieve the email address of authenticated users.

To integrate Jenkins with your LDAP repository, Just select “LDAP” in the Security Realm section, and fill in the appropriate details about your LDAP server. The most important field is the repository server. If you are using a non-standard port, you will need to provide this as well (for example, *ldap.acme.org:1389*). Or, if you are using LDAPS, you will need to specify this as well (for example, *ldaps://ldap.acme.org*)

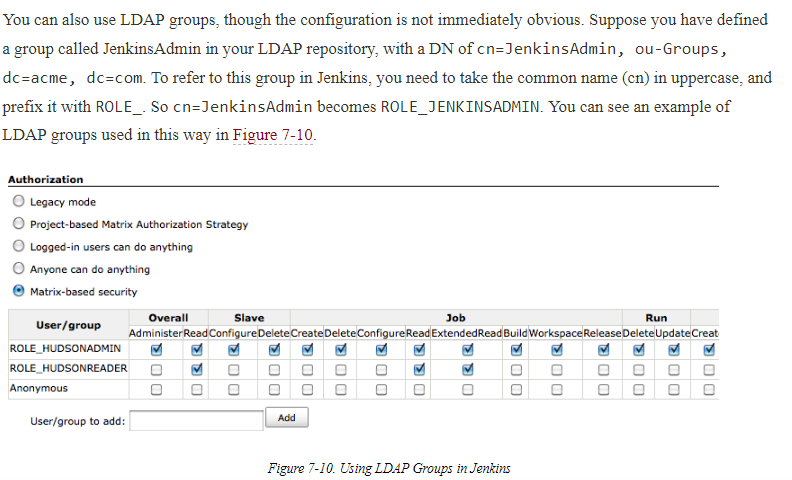




If your server supports anonymous binding, this will probably be enough to get you started. If not, you can use the Advanced options to fine-tune your configuration.

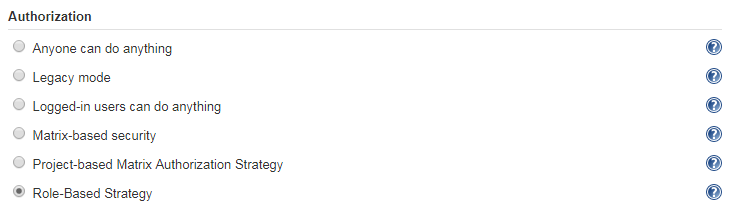
Most of the Advanced fields can safely be left blank unless you have a good reason to change them. If your repository is extremely large, you may want to specify a root DN value (e.g., dc=acme, dc=com) and/or a User and Group search base (e.g., ou=people) to narrow down the scope of user queries. This is not usually required unless you notice performance issues. Or, if your server does not support anonymous binding, you will need to provide a Manager DN and a Manager DN password, so that Jenkins can connect to the server to perform its queries.

Once you have set up LDAP as your Security Realm, you can configure your favorite security model as described previously. When users log on to Jenkins, they will be authenticated against the LDAP repository.



# Authorization—Who Can Do What

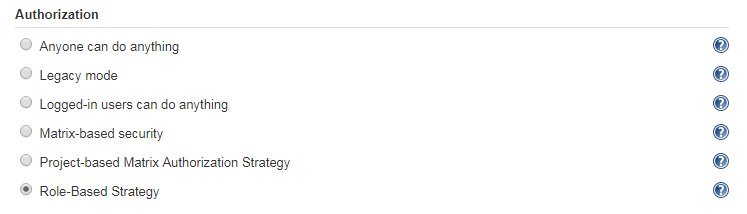
Once you have defined how to identify your users, you need to decide what they are allowed to do. Jenkins supports a variety of strategies in this area, ranging from a simple approach where a logged-in user can do anything to more involved roles and project-based authentication strategies.



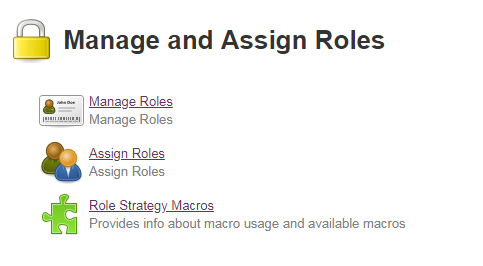
## Role-based Security

Sometimes managing user permissions individually can be cumbersome, and you may not want to integrate with an LDAP server to set up groups that way. A more recent alternative option is to use the Role Strategy plugin, which allows you to define global and project-level roles, and assign these roles to users.

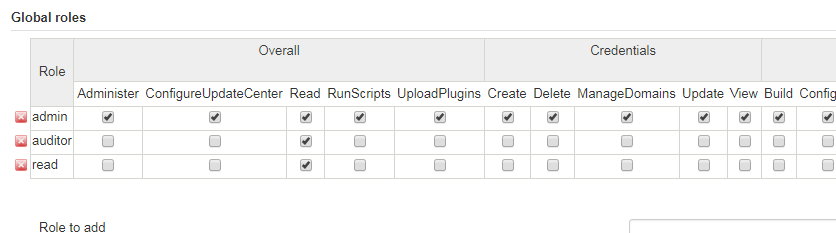
You install the plugin in the usual way, via the Plugin Manager. Once installed, you can activate this authorization strategy in the main configuration page

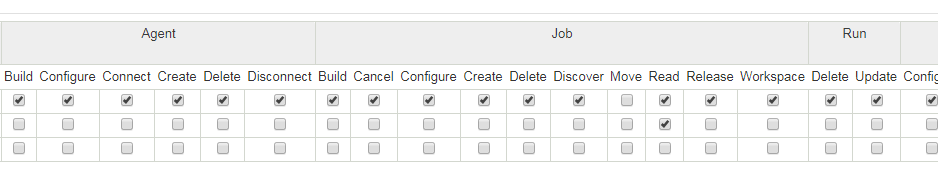


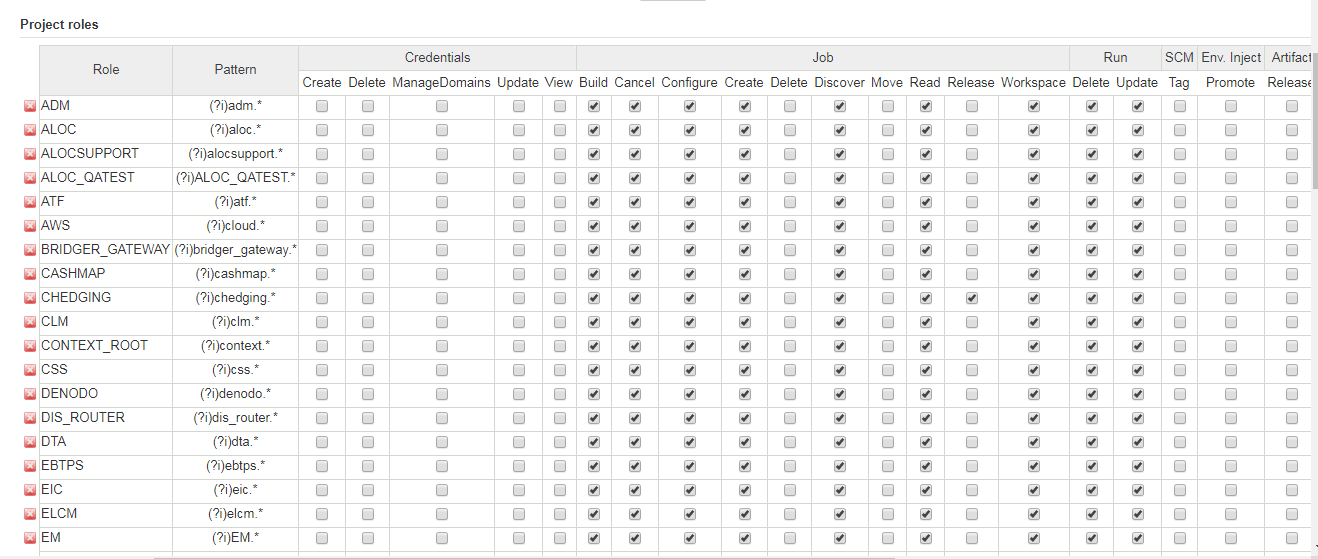
Once you have set this up, you can define roles that regroup sets of related permissions. You set up and configure your roles, and assign these roles to your users, in the Manage Roles screen, which you can access in the Manage Jenkins screen



In the Manage Roles screen, you can set up global and project-level permissions. Global permissions apply across all projects, and are typically system-wide administration or general access permissions

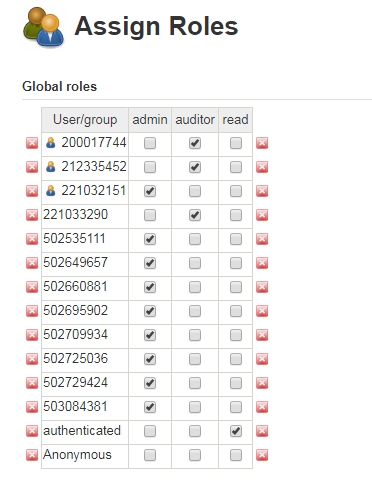


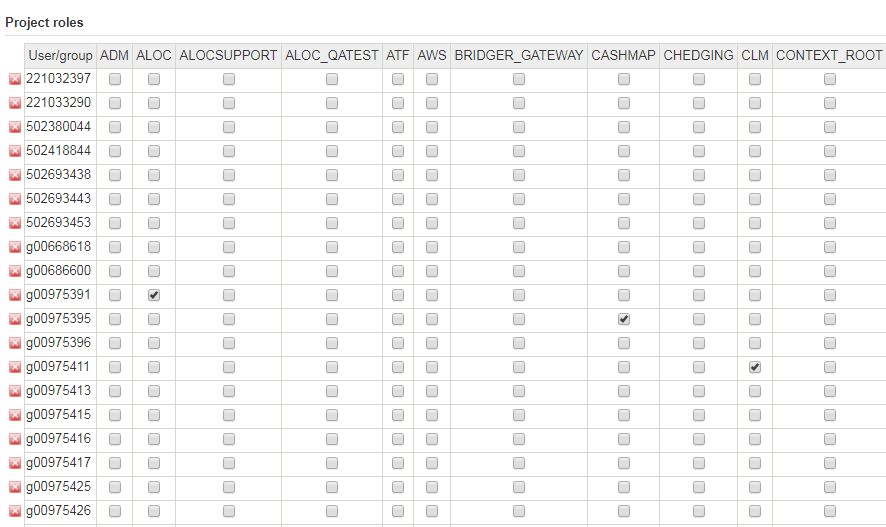




Project roles are slightly more complicated. A project role regroups a set of permissions that are applicable to one or more (presumably related) projects. You define the relevant projects using a regular expression, so it helps to have a clear and consistent set of naming conventions in place for your project names (see [Figure 7-26](https://www.safaribooksonline.com/library/view/jenkins-the-definitive/9781449311155/ch07s05.html#fig-hudson-security-project-roles)). For example, you may wish to create roles distinguishing developers with full configuration rights on their own project from users who can simply trigger a build and view the build results, or create roles where developers can configure certain automated deployment build jobs, but only production teams are allowed to execute these jobs.

Once you have defined these roles, you can go to the Assign Roles screen to set up individual users or groups with these roles





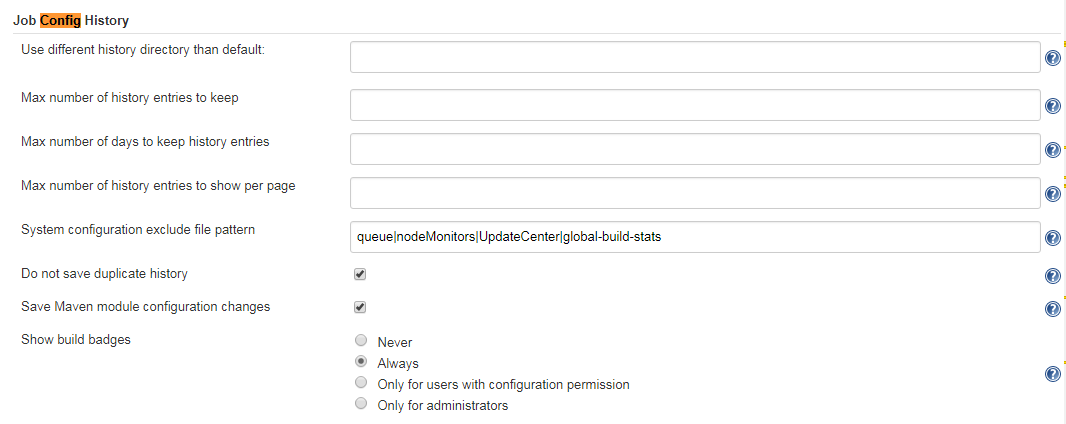
Role-based strategy is relatively new in Jenkins, but it is an excellent way to simplify the task of managing permissions in large, multiteam and multiproject organizations.

# Auditing—Keeping Track of User Actions

In addition to configuring user accounts and access rights, it can also be useful to keep track of the individual user actions: in other words, who did what to your server configuration. This sort of audit trail facility is even required in many organizations.

There are two Jenkins plugins that can help you do this. The Audit Trail plugin keeps a record of user changes in a special log file. And the JobConfigHistory(using in our Jenkins) plugin lets you keep a copy of previous versions of the various system and job configuration files that Jenkins uses.

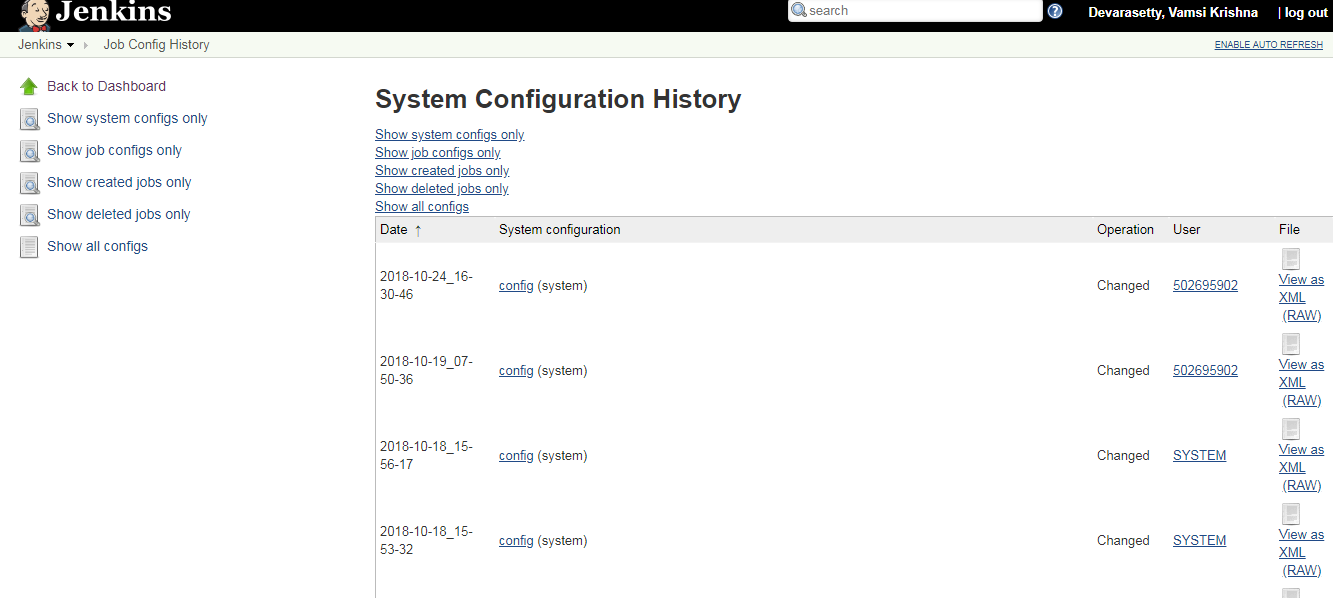
The JobConfigHistory plugin is a powerful tool that lets you keep a full history of changes made to both job and system configuration files. You install it from the Plugin Manager in the usual way. Once installed, you can fine-tune the job history configuration in the Manage Jenkins screen



Here, you can configure a number of useful nonstandard options. In particular, you should specify a directory where Jenkins can store configuration history, in the “Root history folder” field. This is the directory where Jenkins will store a record of both system-related and job-related configuration changes. It can be either an absolute directory (such as /var/hudson/history), or a relative directory, calculated from the Jenkins home directory (see [The Jenkins Home Directory](https://www.safaribooksonline.com/library/view/jenkins-the-definitive/9781449311155/ch03s04.html)). If you don’t do this, job configuration history will be stored with the jobs, and will be lost if you delete a job.

There are a few other useful options in the Advanced section. The “Save system configuration changes” checkbox lets you keep track of system-wide configuration updates, and not just job-specific ones. And the “Do not save duplicate history” checkbox allows you to avoid recording configuration updates if no actual changes have been made. If not, a new version of the configuration will be recorded, even if you have only pressed the Save button without making any changes. Jenkins can also cause this to happen internally—for example, system configuration settings are all saved whenever the main configuration page is saved, even if no changes have been made.

Once you have set up this plugin, you can access the configuration history both for the whole server, including system configuration updates, as well as the changes made to the configuration of each project. In both cases, you can view these changes by clicking on the Job Config History icon to the right of the screen. Clicking on this icon from the Jenkins dashboard will display a view of all of your configuration history, including job changes and system-wide changes



If you click on a system-wide change (indicated by the “(system)” suffix in the list), Jenkins takes you to a screen that lists all of the versions of that file, and allows you to view the differences between the different versions (see [Figure 7-31](https://www.safaribooksonline.com/library/view/jenkins-the-definitive/9781449311155/ch07s06.html#fig-hudson-jobconfig-config-history)). The differences are displayed as diff files, which is not particularly readable in itself. However, for small changes, the readable XML format of most of the Jenkins configuration files makes this sufficient to understand what changes were made.

The JobConfigHistory plugin is a powerful tool. However, at the time of writing, it does have its limits. As mentioned, the plugin only displays the differences in raw diff format, and you can’t restore a previous version of a configuration file (those doing this out of context could be dangerous in some circumstances, particularly for system-wide configuration files). Nevertheless, it gives a very clear picture of the changes that have been made, both to your build jobs and to your system configuration.