**Chapter 5. Access and Security**

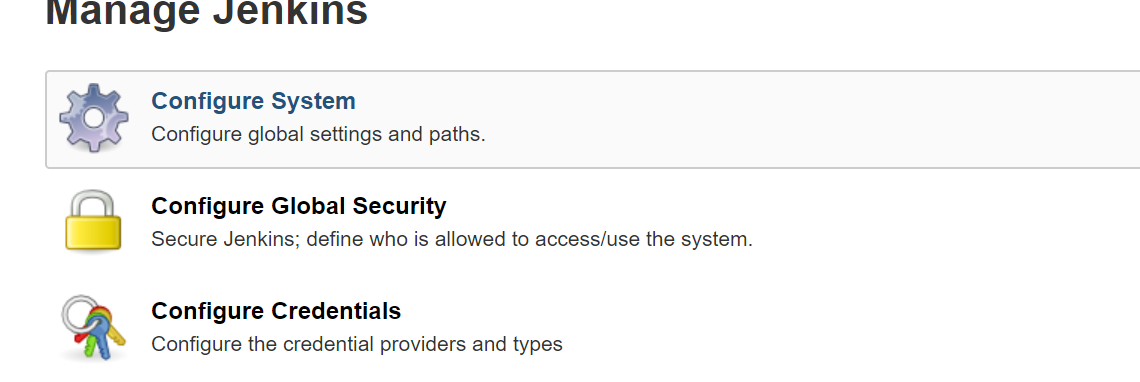
Being able to create pipelines-as-code offers enormous potential and flexibility. In Scripted Pipelines, calls to any Groovy construct or Jenkins functionality or external method can be keyed into the pipeline script. However, that also significantly increases the ability to accidentally or intentionally do something within the code for a pipeline that shouldn’t be done. So, security has to be a first-class concern—and a first-class feature—for both pipelines and the Jenkins environment they are created and run in.

we’ll survey the different ways that Jenkins has for controlling access and security. We’ll first look at the overall security options, then we’ll survey the traditional credentials mechanisms that Jenkins offers and how to use those in pipelines.

# Securing Jenkins

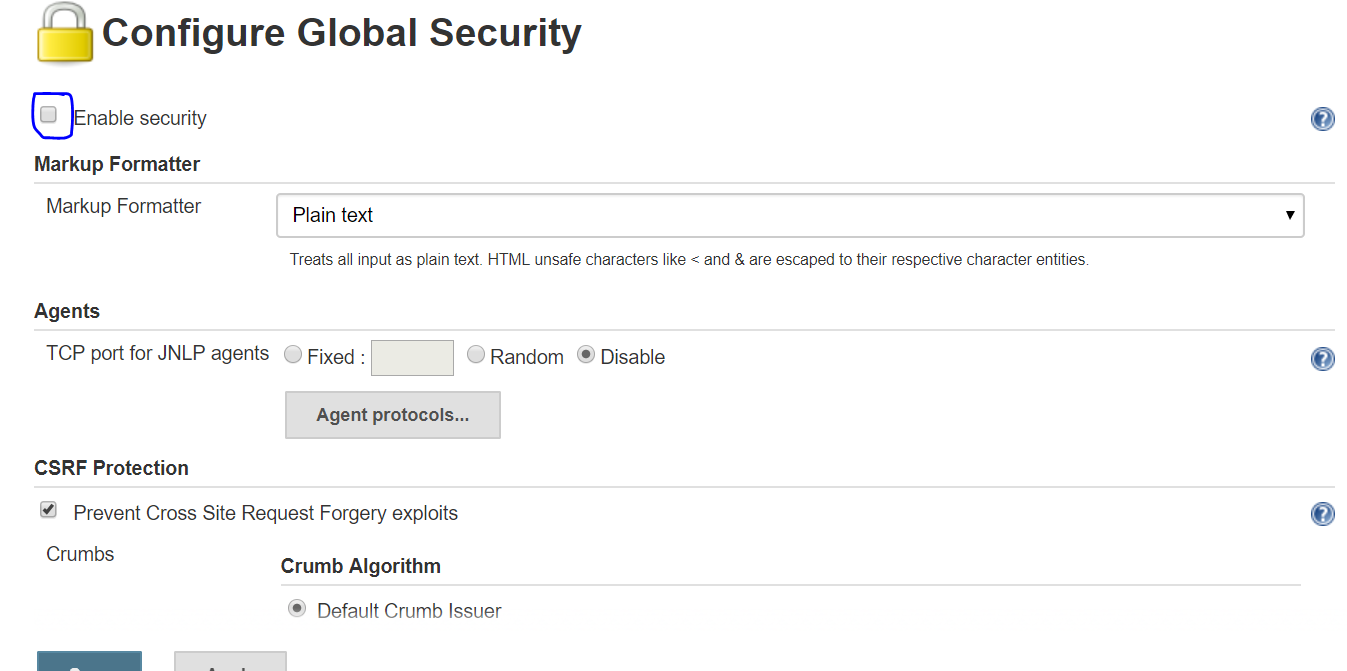
Prior to Jenkins 2.0, the default configuration for Jenkins was to have security disabled—not doing any security checking. This meant that Jenkins was wide open by default. Since Jenkins 2.0, the default has changed to have security enabled. Initially this means that when you use Jenkins, you need to supply a user ID and password. In fact, when you install Jenkins 2.0, you must enter a generated initial password—provided in an obscure file—as part of the installation. You also need to create an initial user with a user ID and password.

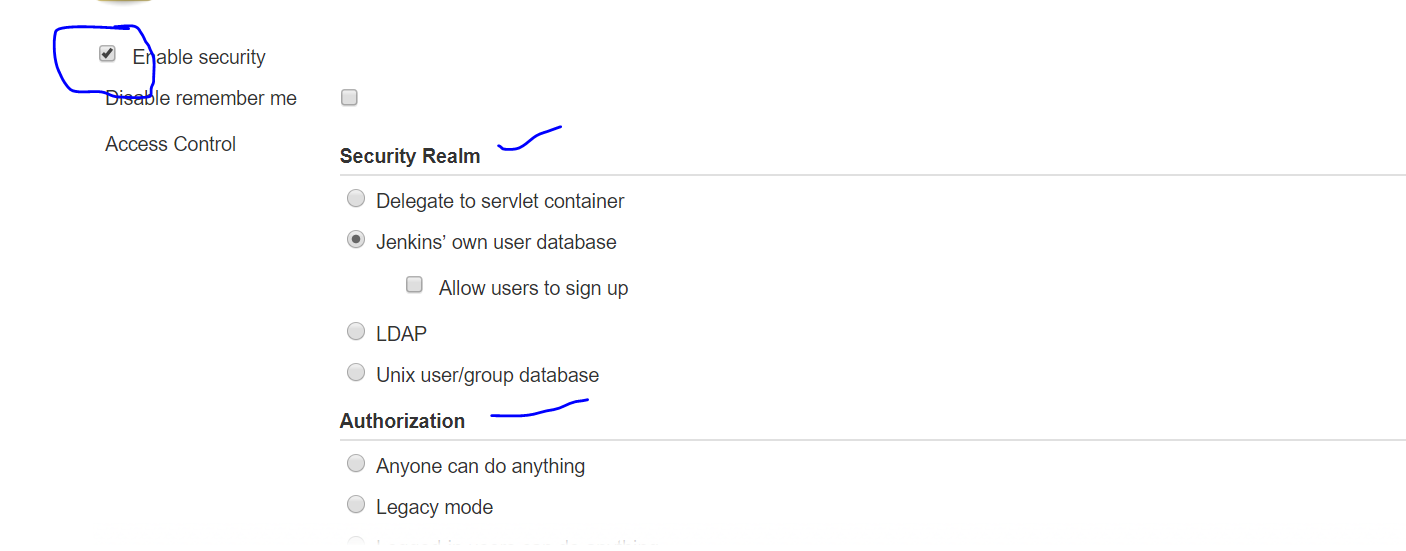
Beyond the basic logins, a number of security mechanisms are available through the Configure Global Security link on the Manage Jenkins page. This should be your starting point for a secure instance.



## **Enabling Security**

The top option on the global security configuration page is also the most high-level one—meaning that it encompasses the most related functionality. Without the “Enable security” option checked, security-checking operations are not enabled. With this option turned on, security can be configured along two dimensions—authentication and authorization.

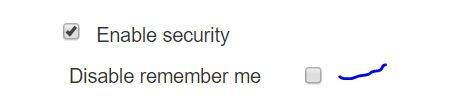




Authentication here refers to how users can identify themselves to the system, such as by user ID and password. This is now called “Security Realm” in Jenkins. Authorization refers to what permissions authorized users have. These two orthogonal dimensions can, together, implement nearly any desired security policy.

The username/password login info is required for any operation unless anonymous users are specifically allowed to do the operation. In Jenkins 2, by default, logged-in users have full control and anonymous users have no access.

Under the checkbox to enable security, we have the “Disable remember me” checkbox. Checking this option to enable it removes the “Remember me on this computer” option from the login screen.



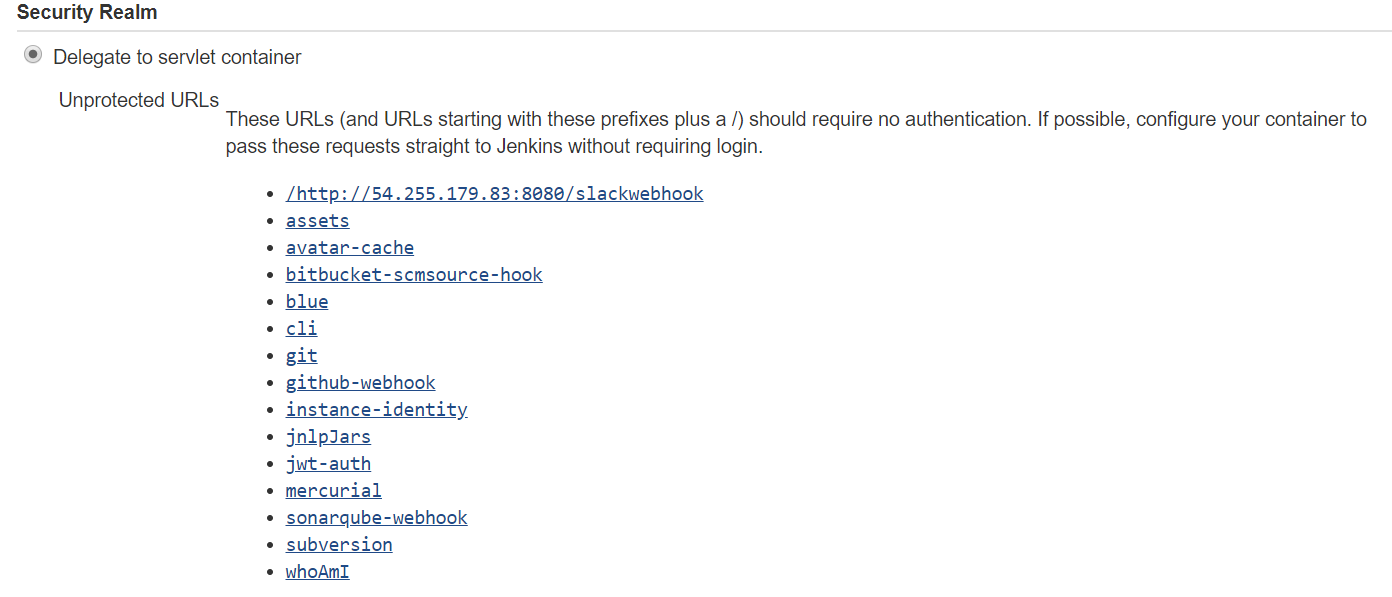
### **ACCESS CONTROL—SECURITY REALM**

This section allows us to specify which entity will be responsible for authenticating users to Jenkins. There are several choices.

#### **Delegate to servlet container**

The servlet container being referenced here is the one running the Jenkins instance. These days, this is usually Jetty, but it might also be Tomcat or some other servlet if the installation has been customized. With this option, you are allowing authentication via whatever mechanism the servlet container uses.

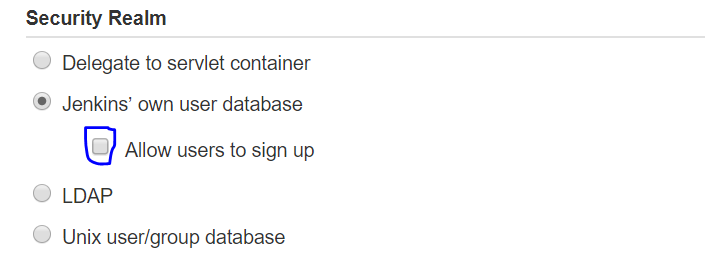
The specifics of how to set this up depend on how authentication is configured for the particular servlet container being used. The best approach is to consult the documentation for the servlet container. Up until v1.163 this was the default security realm. It is not as likely to be used these days given the other options, but can still be worthwhile for backward compatibility, or if you have invested significant setup for authentication in the servlet container’s configuration.



#### **Jenkins’ own user database**

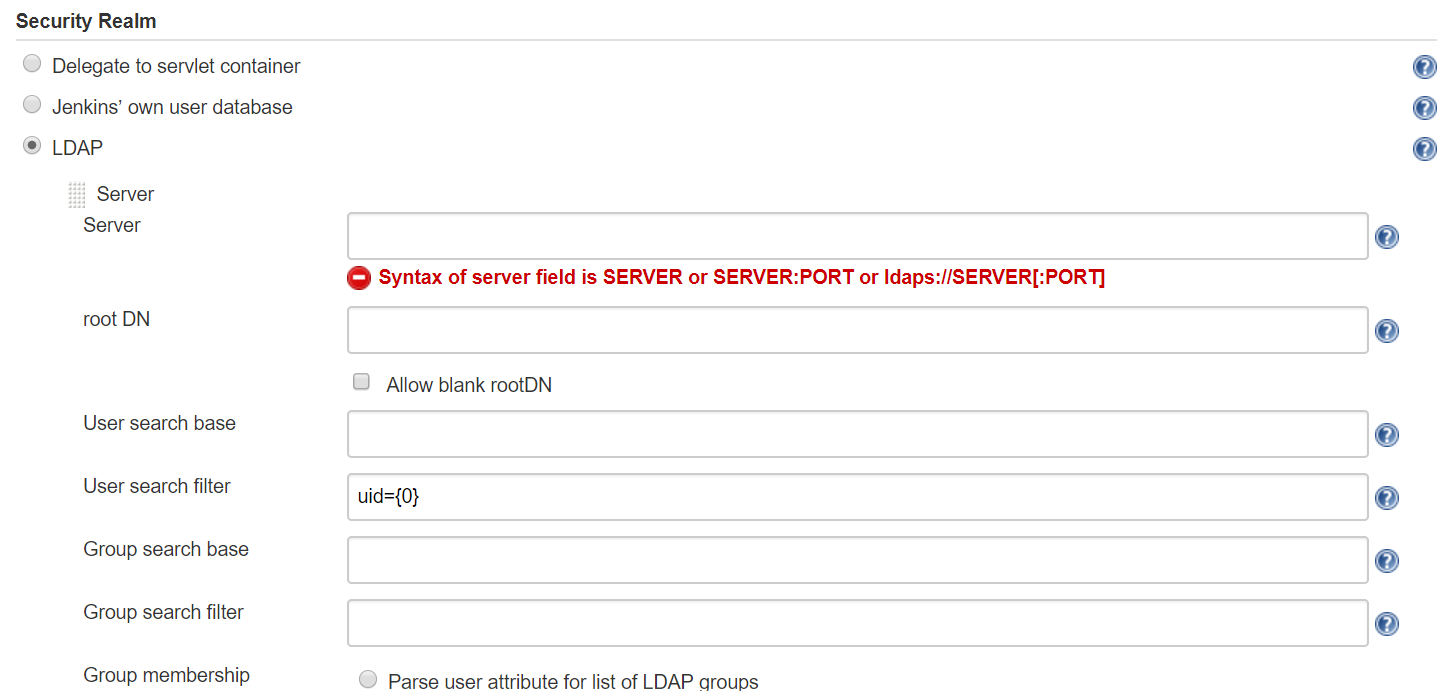
This option delegates authentication to the list of people maintained by/known to Jenkins. This is not a typical use case, but can be suitable for smaller, basic setups. Note that this includes not only all the users that Jenkins specifically knows about, but also users mentioned in commit messages.

A suboption allows enabling users to “sign up”—meaning they can create their own accounts at the time they first need to log in to Jenkins. This suboption is disabled by default to more tightly control access.



#### **LDAP**

The Lightweight Directory Access Protocol (LDAP) is a software protocol for locating people, organizations, devices, and other resources on a network. If your company uses LDAP, this is where you can configure it for Jenkins. You can add more than one LDAP server (each having a different configuration if needed).



#### **Unix user/group database**

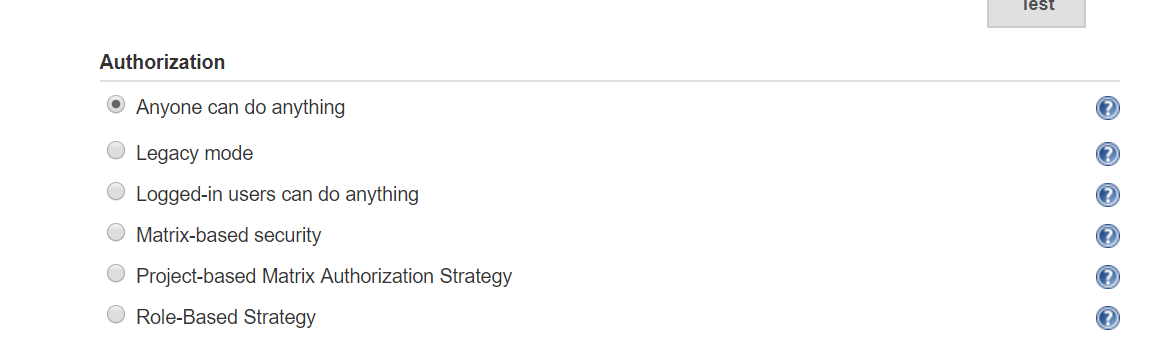
This option delegates authentication to the host Unix system’s user database. If this is used, users can log into Jenkins using their Unix username and password. Unix groups can also be used for authentication. If a user and a group have the same name, prepending an “@” onto the name differentiates it as a group. Note that there may be extra configuration required to make this all work, such as making Jenkins a member of the shadow group for access on operating systems that use that.

### **ACCESS CONTROL—AUTHORIZATION**

Once authenticated, Jenkins needs to know what kind of operations users should be allowed to do. Like in the Security Realm section, there are several options here.

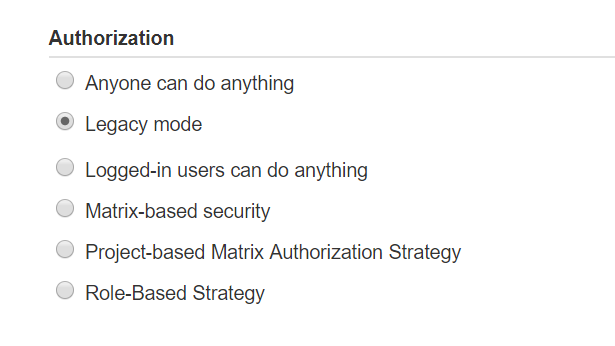
#### **Anyone can do anything**

No real authentication is done with this option. Basically, everyone is considered as “trusted”—including anonymous users (even if they haven’t logged in yet). This is not recommended, but can be suitable in rare cases for completely trusted environments to allow unrestricted access for simplicity and efficiency.



#### **Legacy mode**

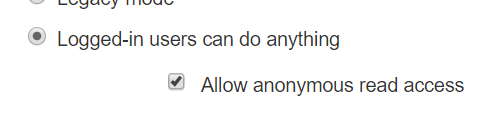
This mode emulates Jenkins behavior prior to v1.164: anyone who has the “admin” role has full control, and everyone else has read-only access.



#### **Logged-in users can do anything**

As the name implies, users must first log in, but then have full access. This is useful if you don’t mind allowing everyone full access, but want to keep track of who is doing what (via them being logged in).

A suboption here enables anonymous users to have read-only access.



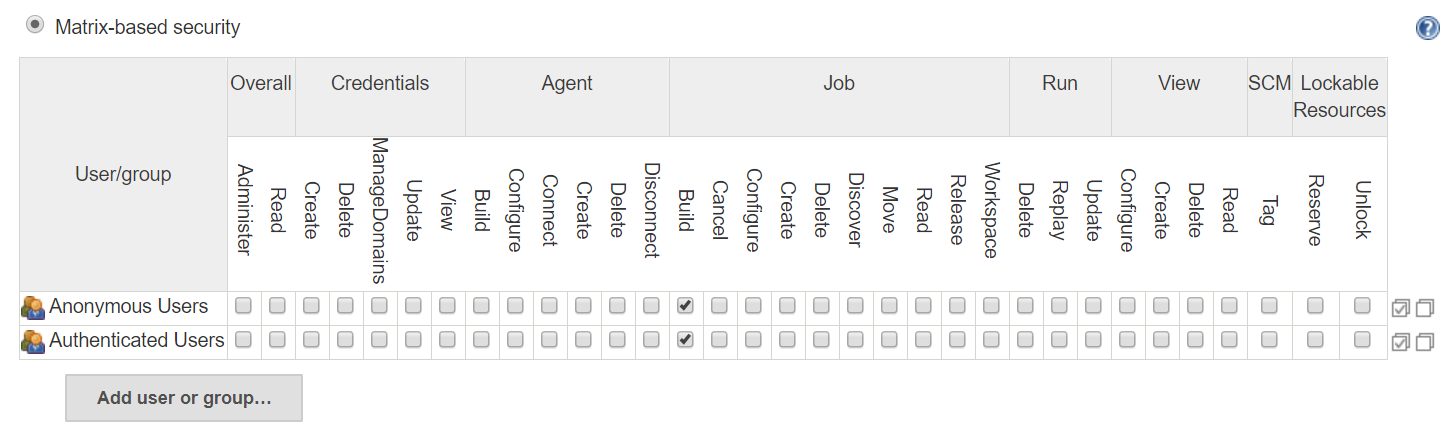
#### **Matrix-based security**

This option allows you to specify very specific permissions for individual users or groups via checkboxes in a matrix arrangement. The columns in the matrix are divided into categories (groupings) such as “Overall,” “Job,” “Run,” etc. Then underneath each of those items are further specific permissions related to that category.

The rows of the matrix each represent a user or group. There are two default groups that are automatically added: “Anonymous Users” (users who have not logged in) and “Authenticated Users” (users who have logged in). A text box under the matrix allows you to add new users.

Granting a particular permission to a user or group is just a matter of clicking in the box that corresponds to the appropriate row for the user/group and the column for the specific permission. Removing a permission involves just clicking again to clear the checkbox.

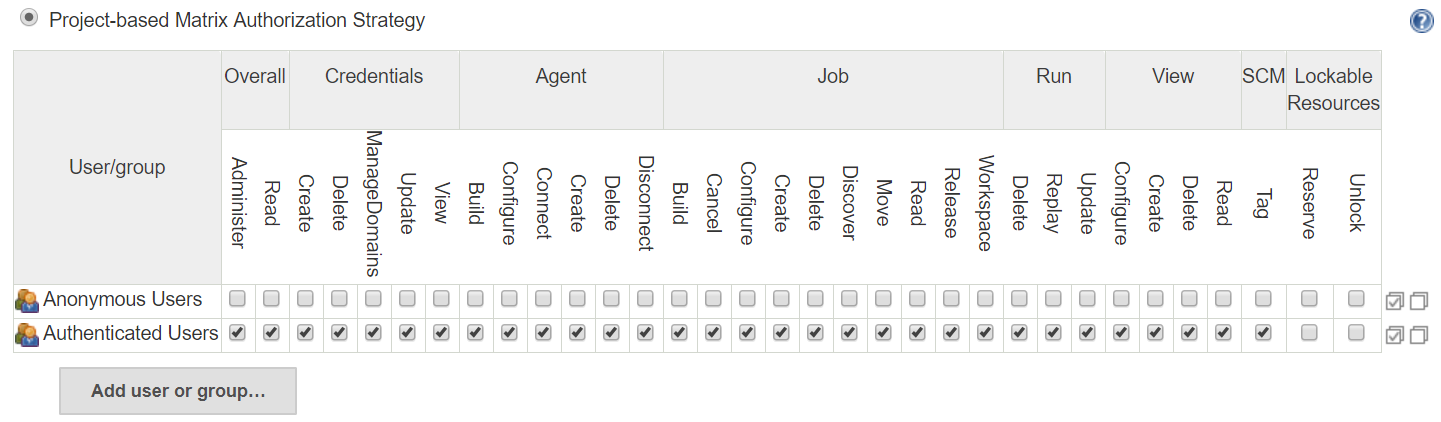
At the end of each row are boxes you can click on to grant all permissions or remove all permissions for that user/group.

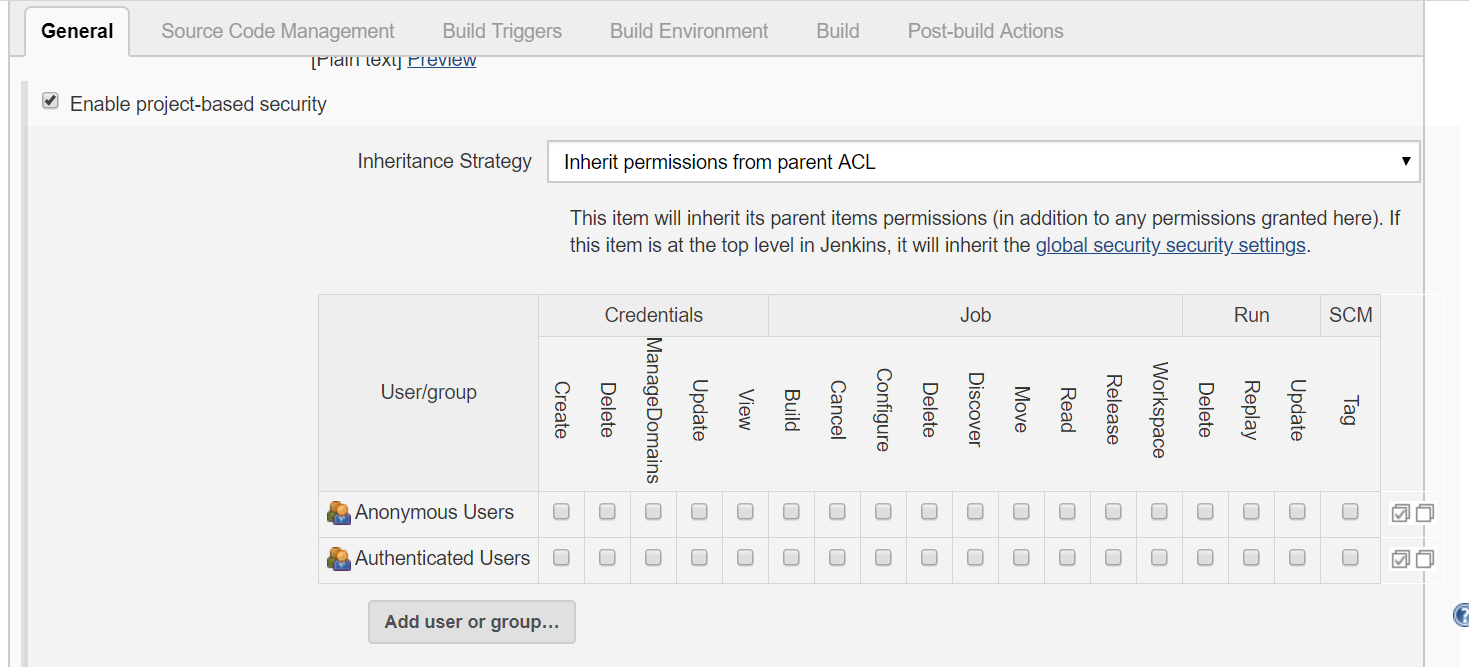


#### **Project-based matrix authorization strategy**

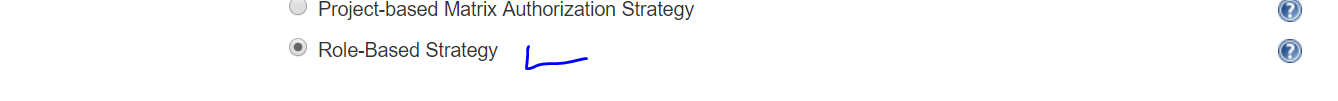
This option is an extension to the “Matrix-based security” model described in the preceding section. When selected, this adds a similar matrix to each project’s configuration page. This allows for configuration by user/group per project, so you can restrict access to some projects while allowing it for others.

More specifically, when this option is set in the global security page, each project’s configuration page will havean “Enable project-based security” option in the General configuration section. Selecting this option will thenpresent an authorization matrix for that project that can be configured like the global matrix to provide project-specific access. An additional option allows you to select whether to inherit permissions from a parent access control list, the globally defined permissions, or not at all.

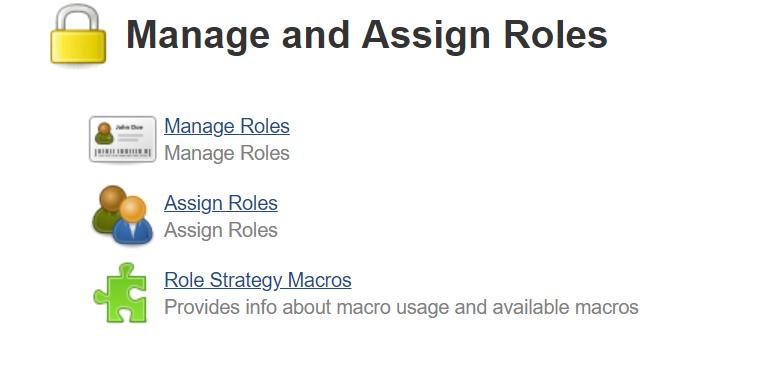
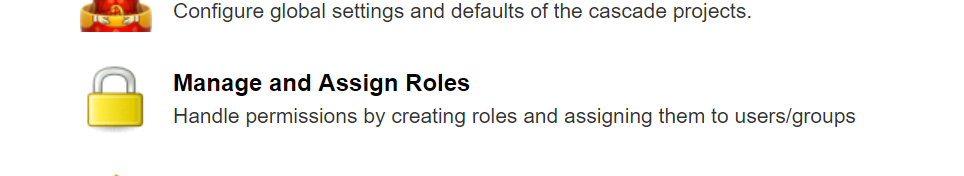




Role Based Strategy



When enabled the above option we will get a option in the manage Jenkins page as Manage and Assign Roles as below

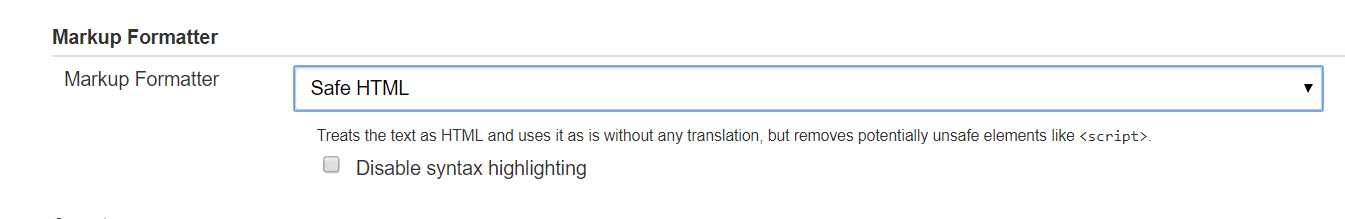


## Other Global Security Settings

Beyond the authentication and authorization settings, there are a number of other options on the global security configuration page that can be set. This is a miscellaneous collection centered around keeping Jenkins implicitly safe (locking down security holes) rather than explicitly defining access.

### MARKUP FORMATTER

Jenkins allows users to put in free-form text in various fields, such as job descriptions, build descriptions, etc. You can choose to format those as plain text or HTML. If you want to use HTML, set this option to Safe HTML. “Safe” here refers to allowing only HTML constructs that don’t pose a security risk of being hacked (i.e., modified in a way that would execute operations that would put the system at risk). Examples of safe HTML constructs include the basic ones such as bold, italics, hyperlinks, etc.

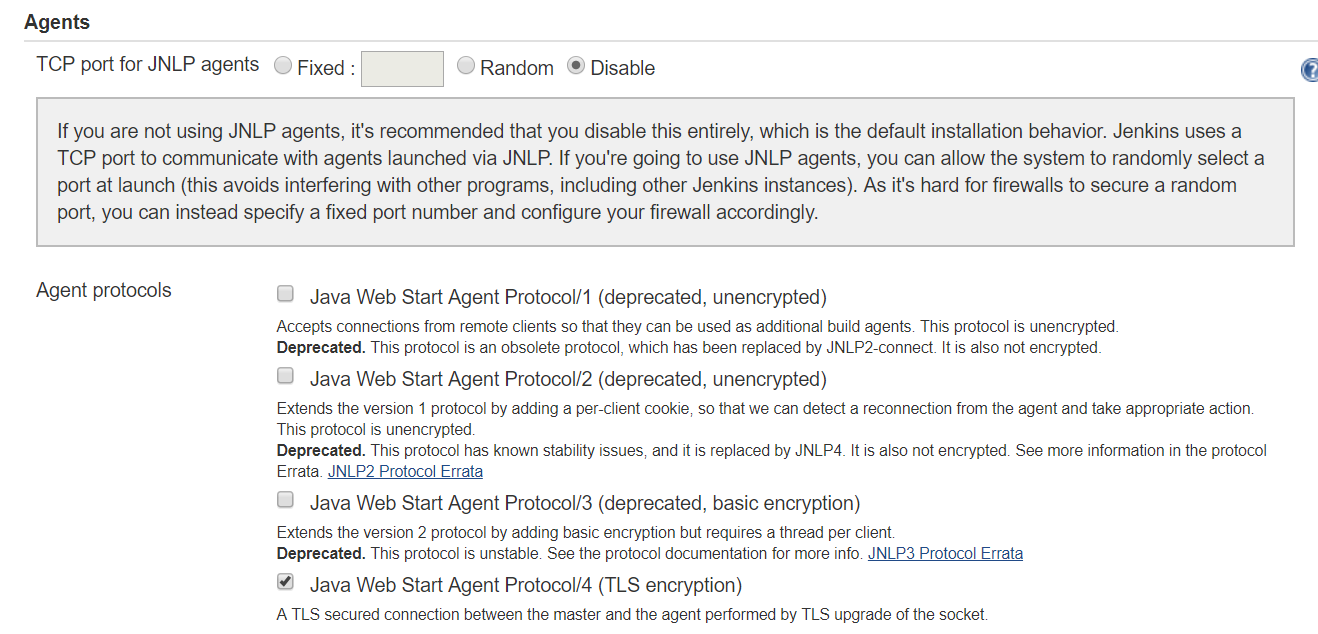


### AGENTS

Despite the generic name, this section is about configuring the TCP port for agents launched through the JNLP process. (JNLP refers to the Java Network Launch Protocol—a way that an application can be launched on a client’s desktop using resources hosted on a remote server.)

Normally, a random port is used for this. However, you can specify a fixed port instead to make it more secure (only having to open the firewall for the fixed port). If you are not making use of the JNLP functionality, you can use the Disable option here to make your system even more secure.

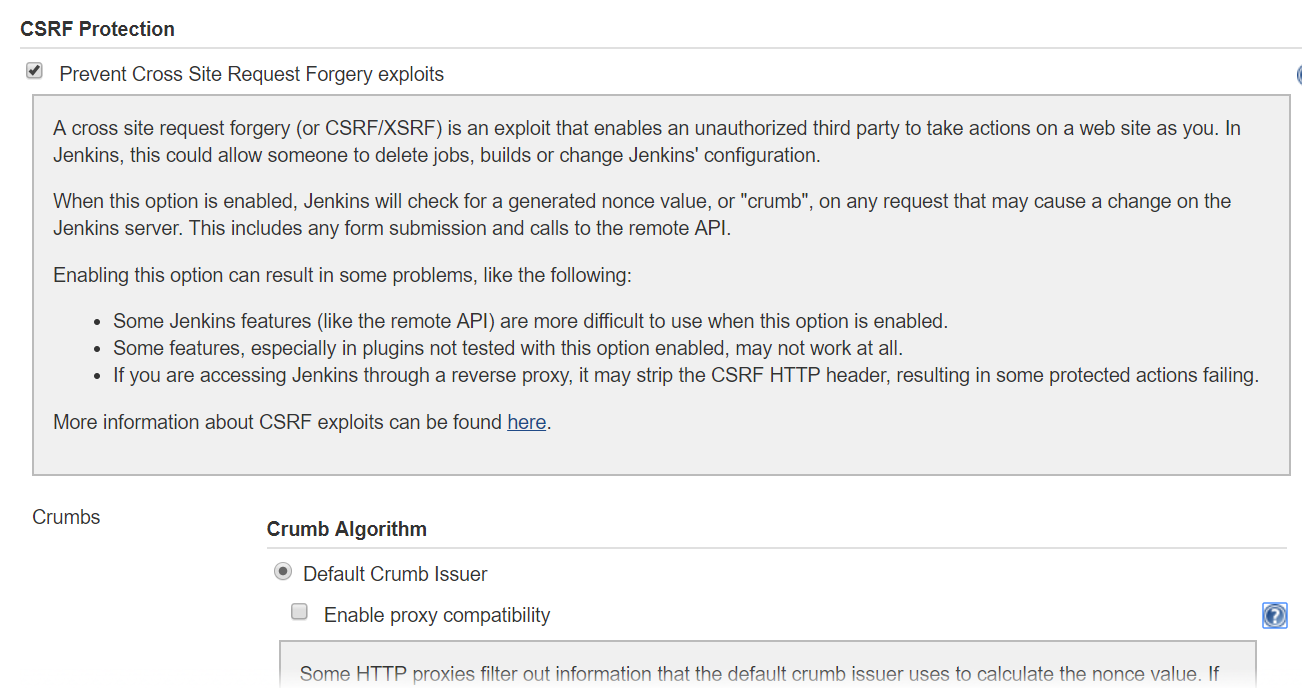
A suboption allows you to choose a particular version of the JNLP protocol if needed

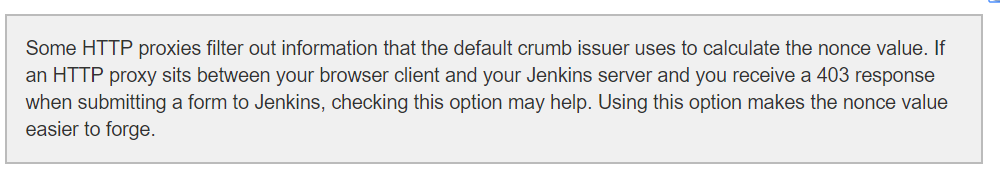


### PREVENT CROSS-SITE REQUEST FORGERY EXPLOITS

Cross-Site Request Forgery (CSRF) is a type of attack that can force a user to execute unwanted actions on a web application that they are authenticated to. Part of the prevention of this has to do with verifying that a crumb trail (navigation history) exists for the user in Jenkins.

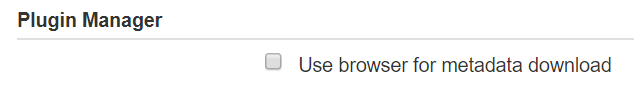
A suboption allows specifying proxy compatibility to help prevent the proxy from filtering out information about the crumb trail.





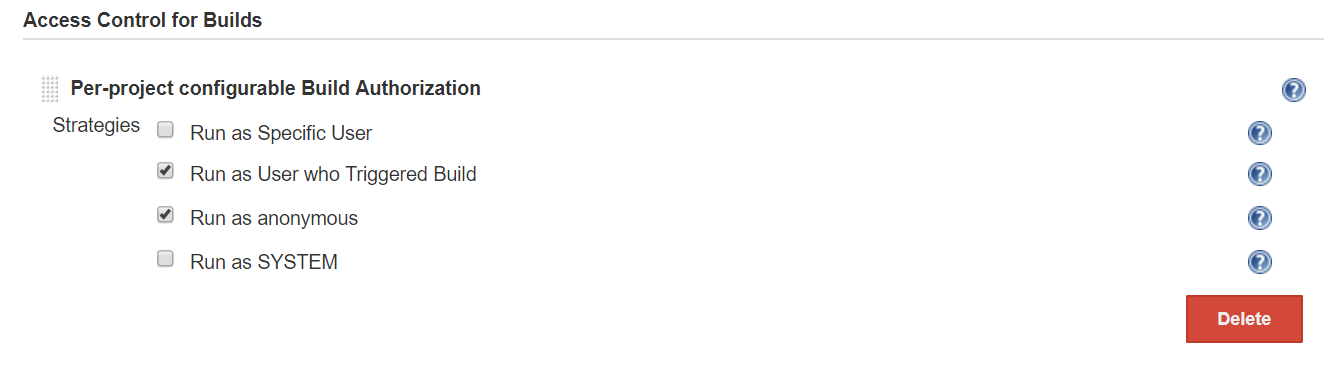
### **PLUGIN MANAGER**

The option here is “Use browser for metadata download,” and it is normally unchecked (off). Turning this option on tells Jenkins to let the browser download metadata around plugins instead of Jenkins doing it itself. Unless you have a specific reason to activate this, it is best to just leave it turned off and allow Jenkins to do the downloads.

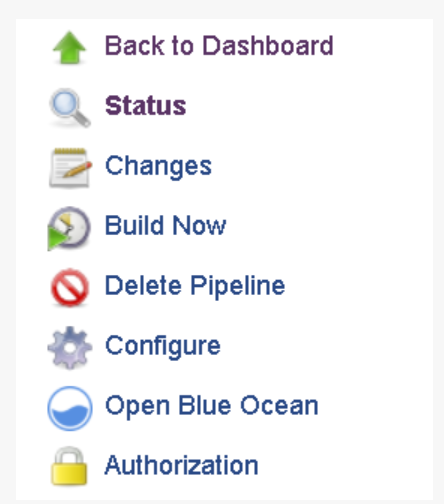


If you choose to install the [Authorize Project plugin](https://plugins.jenkins.io/authorize-project), you may have additional entries here. This plugin allows additional per-project options for running builds with specific authorization.

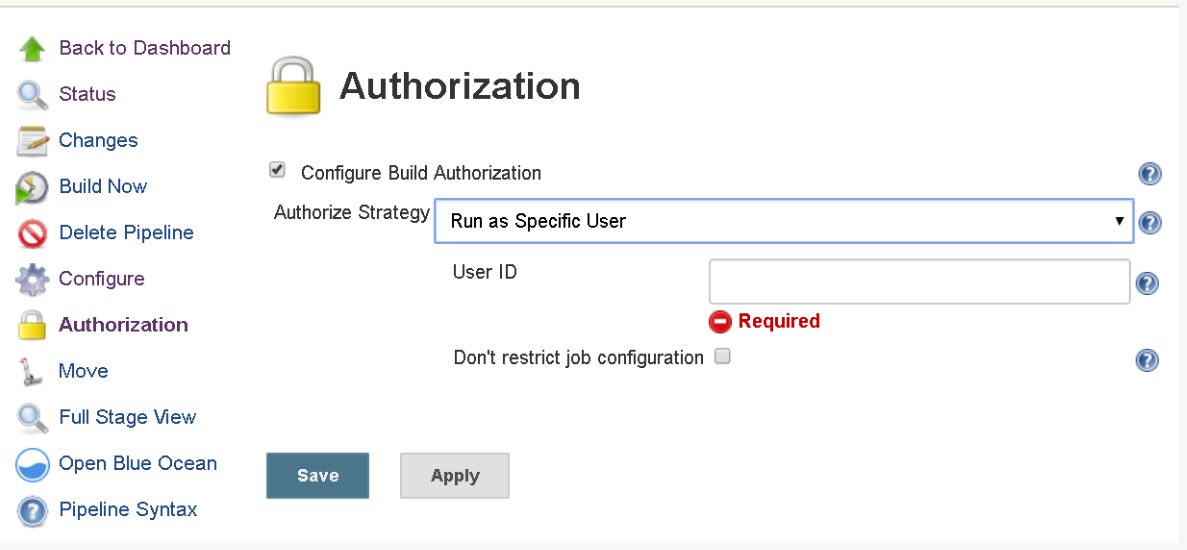
The global configuration part that would appear here allows you to select which types of authorized usersappear as choices in projects. The list is shown in [Figure 5-4](https://www.safaribooksonline.com/library/view/jenkins-2-up/9781491979587/ch05.html#fig_ls_choice_prjauth_plug) and is fairly self-explanatory.



The plugin adds a new Authorization item on the page for each job



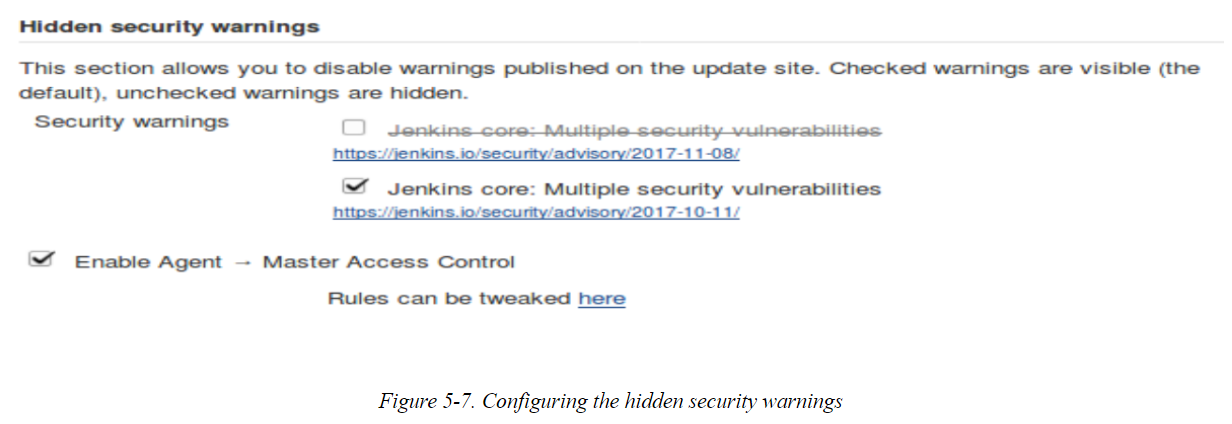
Clicking on that link brings up a simple configuration screen that allows you to select from the choices configured globally for the plugin, to control who can run the job



### HIDDEN SECURITY WARNINGS

The options here have to do with surfacing security warnings from update sites for installed components. (In older versions of Jenkins, these weren’t shared directly in Jenkins but rather in emails, blogs, etc. Starting with v2.40, they can now be shown directly in Jenkins.) If you have a list of warnings present, then checked warnings are shown, and unchecked ones aren’t.

Below shows an example of configuring these warnings if they exist. Note that there are two warnings—one is unchecked here and one is checked.



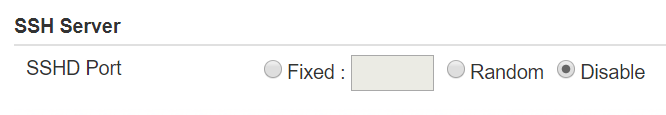
Below shows what the warnings look like with this configuration. Notice that only the one that is checked is shown.



Another option here is to enable “agent to master” access control. This has to do with what commands agents can send to the master to make those interactions safer. If you need to tweak those rules to work with a specific instance or plugin, there is a link here to do that as well.

### SSH SERVER

For executing a subset of command-line commands over SSH, Jenkins can function as an SSH server. Some plugins may also use this functionality. If this is needed, a fixed port can be set up here to simplify security. A random port can also be chosen each time to avoid conflicts. If this functionality is not needed, it is best to use the Disable option to disable having an open port exposed.



# Credentials in Jenkins