# Lab X. Access Control

Goal

In this lab, you will learn how to configure the RBAC plugin for a typical security model.

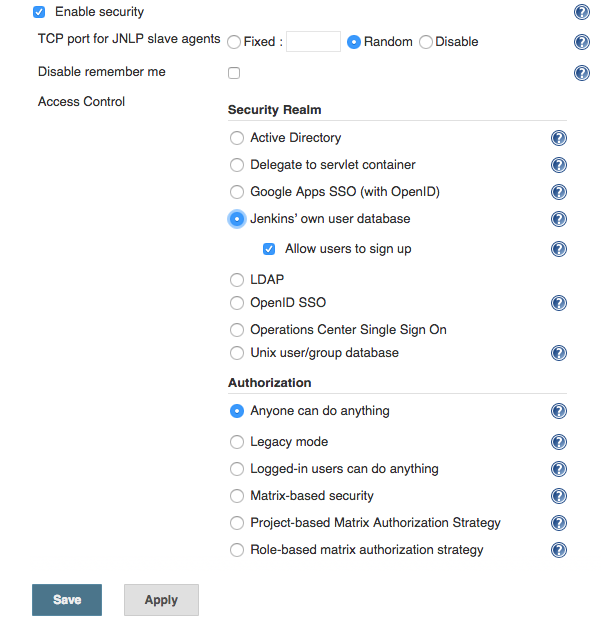
Step 1. Configure Jenkins Security Realm

Before we can enable the RBAC plugin, we need to configure the security realm that will provide the user authentication. For the purposes of this lab we will use Jenkins’ own user database. There are some limitations with this security realm: for instance, it does not provide group information. Also we **need to have at least one user created before we switch over to the RBAC plugin** otherwise we will be locked out of Jenkins.

With a real deployment of Jenkins we would not be using Jenkins’ own user database, so some of the hoops we jump through for this lab exercise disappear.

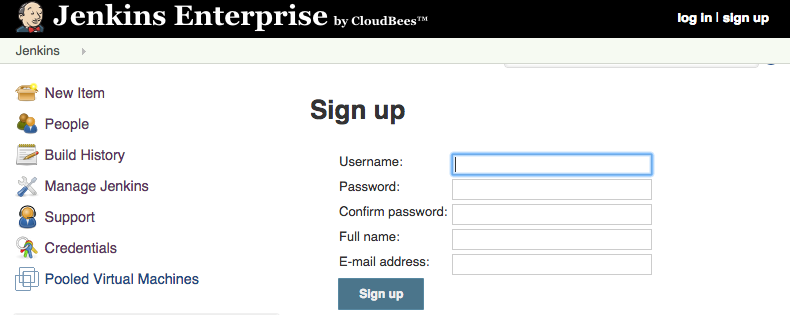
Go to the Jenkins security configuration screen (“Manage Jenkins” -> “Configure Global Security”).

Make sure the “Enable security” option is checked, then scroll to the “Security Realm” section. Now select the “Jenkins’s own user database” radio button. Finally, scroll down to the Authorization section and select “Anyone can do anything”:



Now save your changes.

Now we are going to add three users to Jenkins’ user database. Log out of your account, then hit the “Sign up” button in the top right-hand corner of the screen:



Now create three users with the following details:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User 1** | **User 2** | **User 3** |
| **Username** | harryh | barryb | sallys |
| **Password** | harryh | barryb | sallys |
| **Full name** | Harry McHarry | Barry O’Barry | Sally Somebody |
| **E-mail address** | [harry@a.a](mailto:harry@a.a) | [barry@a.a](mailto:barry@a.a) | [sally@a.a](mailto:sally@a.a) |

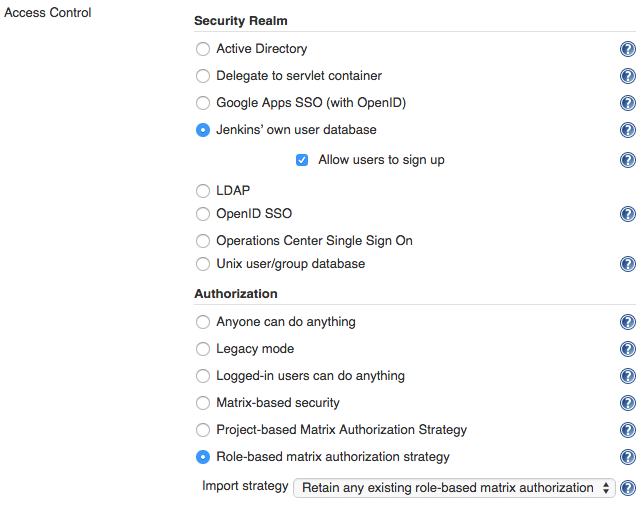
**Note:** You will have to log out after creating each user, before you can create the next.

Once all three users have been created, we have configured the security realm with enough information that we can safely turn on the RBAC authorization strategy without locking ourselves out.

Step 2. Configure the roles

Log in as “harryh”, the go back to the “Configure Global Security” page (“Manage Jenkins” -> “Configure Global Security”).

In the “Authorization” section, select “Role-based matrix authorization strategy” for authorization and save the changes:



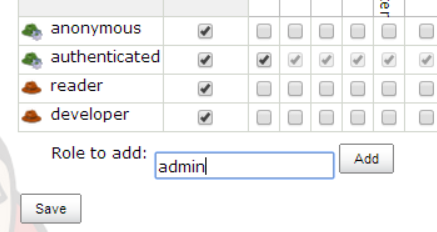
Now we’re going to assign roles to our users. First, go to the “Manage Roles” screen (“Manage Jenkins”-> “Manage Roles”). Note that if you do not see this option, you may have forgotten to save the change of Authorization strategy to “Role-based matrix authorization strategy”!

Our goal here is to create a security set up that has the following characteristics:

* Anonymous users have no access whatsoever, and they need to login first before even seeing the Jenkins top page.
* Once logged in, users have generic read access.
* Jobs can put some users to the “reader” role that has read-only access to jobs, their test results, build results, etc.
* Jobs can put some users to the “developer” role, who can start a new build, configure jobs, and so on.
* A few people will be in the “admin” role, which gets irrevocable full access to the whole of Jenkins.

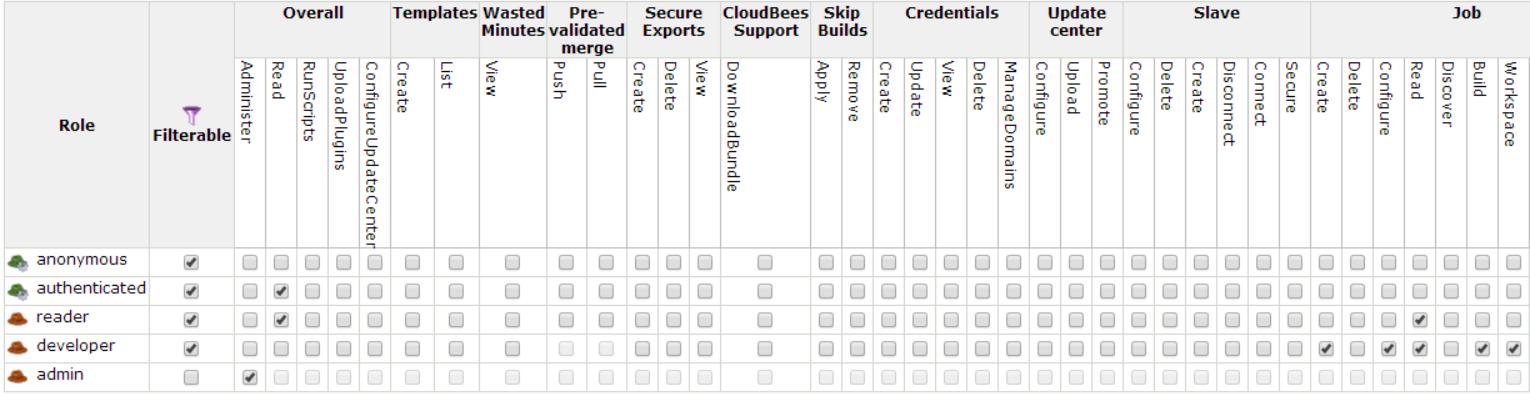
This maps **to two pre-defined system roles** (“anonymous” and “authenticated”) and **three additional roles we create** (reader, developer and admin).

To create the additional roles, type in the their titles to the “Role to add” field on the “Manage Roles” page and then hit the add button:



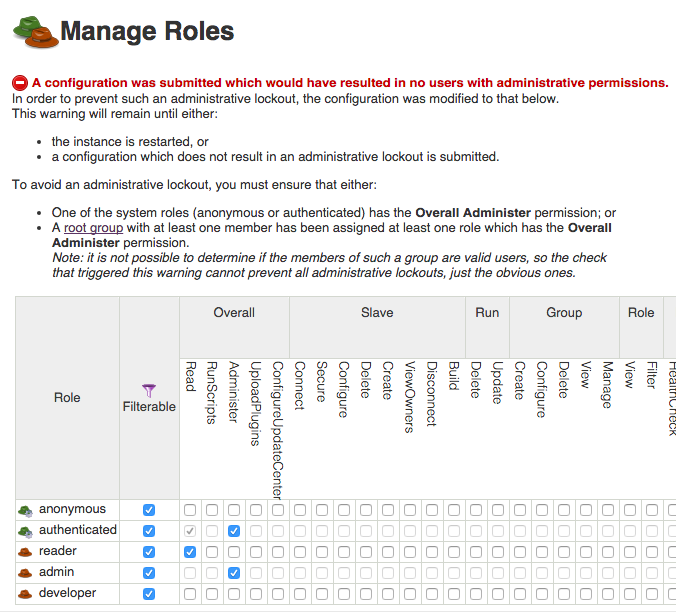
By default, “filterable” will be set to true for them all. As such, you’ll now need to edit each role’s permissions\*\*. Note that the first permissions column shows the name of the ***section*** the permission is for, while the second column shows the name of the ***checkbox*** in that section:

|  |  |  |  |
| --- | --- | --- | --- |
| **Role** | **Permissions (section, checkbox)** | | **Filterable** |
| Anonymous | None | None | Yes |
| Authenticated | Overall | Read | Yes |
| Reader | Overall  Job | Read  Read | Yes |
| Developer | Job  Job  Job  Job  Job | Read  Create  Configure  Build  Workspace | Yes |
| Admin | Overall | Administer | No |

Once you are ready to edit the “authenticated role”, note that you’ll be able to uncheck all boxes at once by scrolling to the right and clicking on the JEBC Screenshots:unchck.PNGbutton.

Now try to save your configuration.

Note that the “Anti-lockout” guard will kick in and change our configuration:



This is because the “admin” role is not associated with a group that contains at least one member. As such, saving the configuration as-is would have “locked” you out.

Instead, the “Anti-lockout” guard will now give the “authenticated” role overall admin permissions.

For now, let’s save these altered roles.

Step 3. Configure the root groups

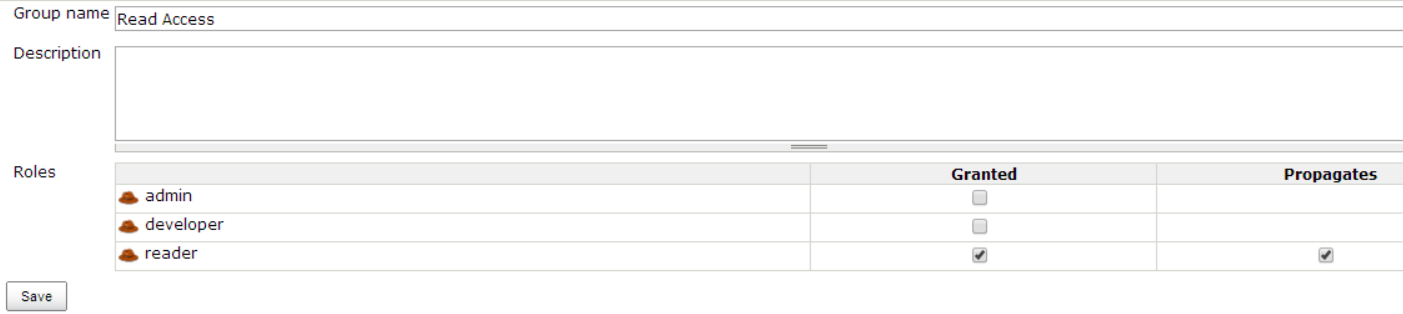
We need the next step is to create groups that connect users to the roles, and populate these groups.

From the Jenkins top page, click on the “Groups” link in the left-hand menu. This will take you to the “Groups” list, which will be unpopulated:

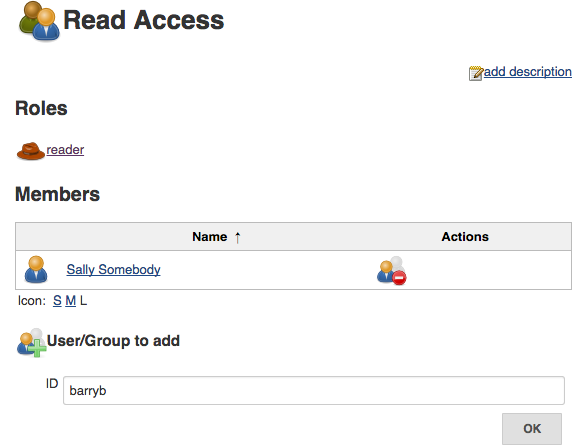


Click on the “add some?” link and start creating these three groups by populating the following fields with the following values:

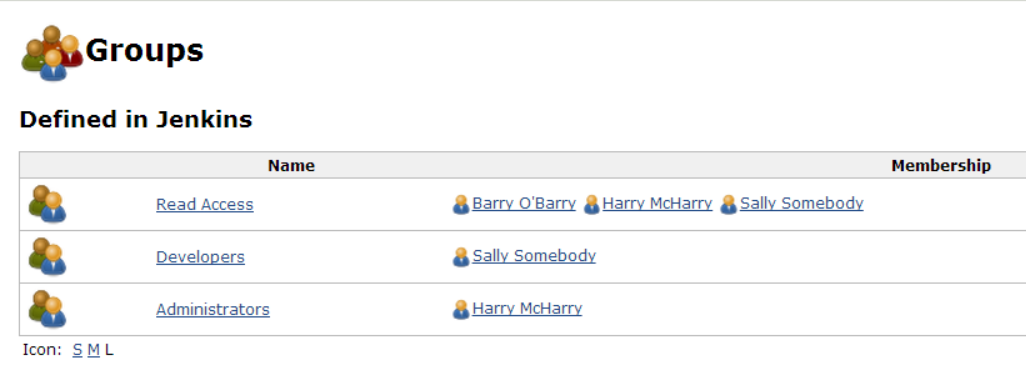
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Group Name** | **Roles** | **Granted at** | **Propagated** | **Members** |
| Read Access | reader | Current level | Yes | barryb, harryh, sallys |
| Developers | developer | Current level | Yes | sallys |
| Administrators | admin | Current level | Yes | harryh |

For example, “Read Access” is configured like this:

Once the Group Name and Roles Granted/Propagated fields are set, hit “Save” and add the corresponding users to the group:



Repeat this process until all 3 groups have been created by hitting the “Back to groups” link in the left-hand menu and then hitting the “New Group” link in the same menu while on the “Groups” page.



Step 4. Completing configuration of the roles

Now that we have a group with the Overall/Administer permission, we can complete configuration of the roles.

Go back to the “Manage Roles” screen (“Manage Jenkins” -> “Manage Roles”).

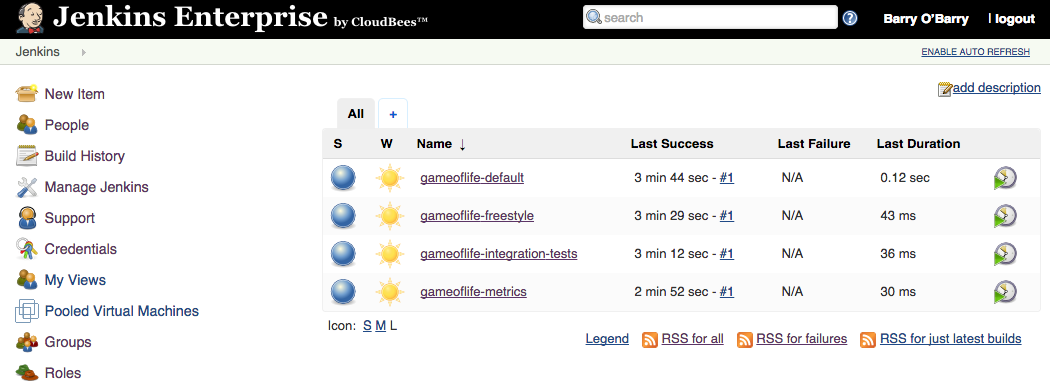
De-select the Overall/Admin permission for the “authenticated” role and ensure that the Overall/Read permission is still on selected.

Now save the configuration. Since the “admin” role is now associated with a populated group, you should not receive an “Anti-lockout” warning.

Step 5. Explore the permissions

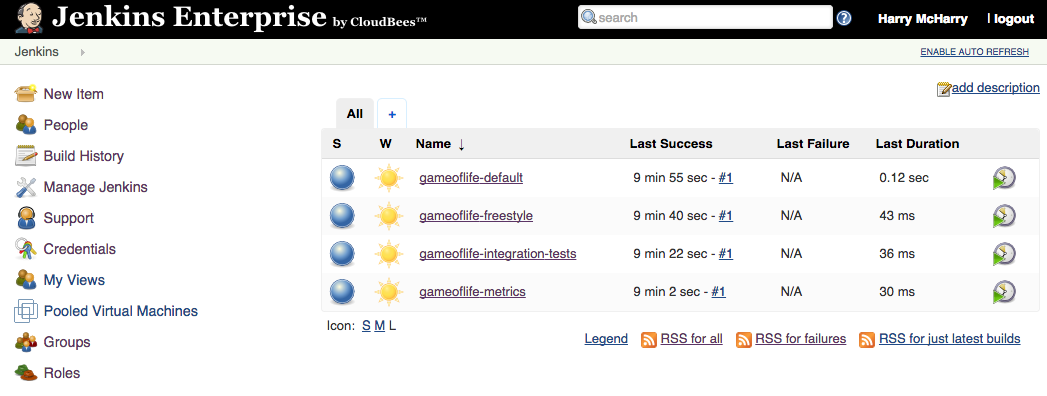
Logout of Jenkins and verify that you cannot see any of the jobs in Jenkins.

Login as barryb and verify that you can see the jobs created in previous labs but you cannot trigger any builds nor modify the job configurations:



Logout and login as sallys. Verify that you can create a new freestyle job and trigger a build of the job, but that you do not have access to the “Manage Jenkins” screen:

Logout and login as harryh. Verify that you can configure the job you created as sally, and that you can trigger it to build. Verify that you can access the *“*Manage Jenkins” screen:

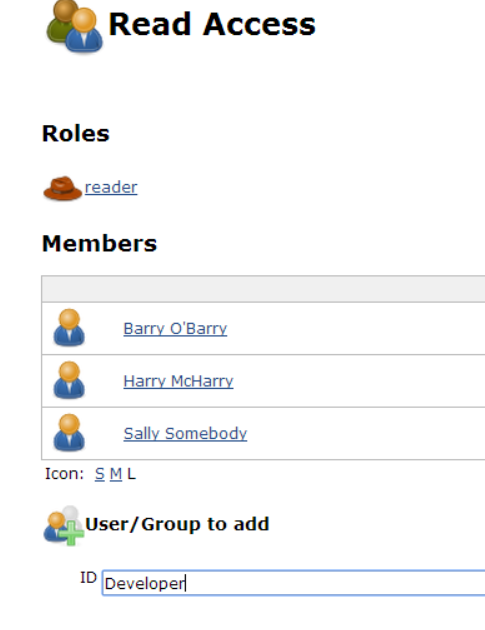


Step 6. Refine the groups

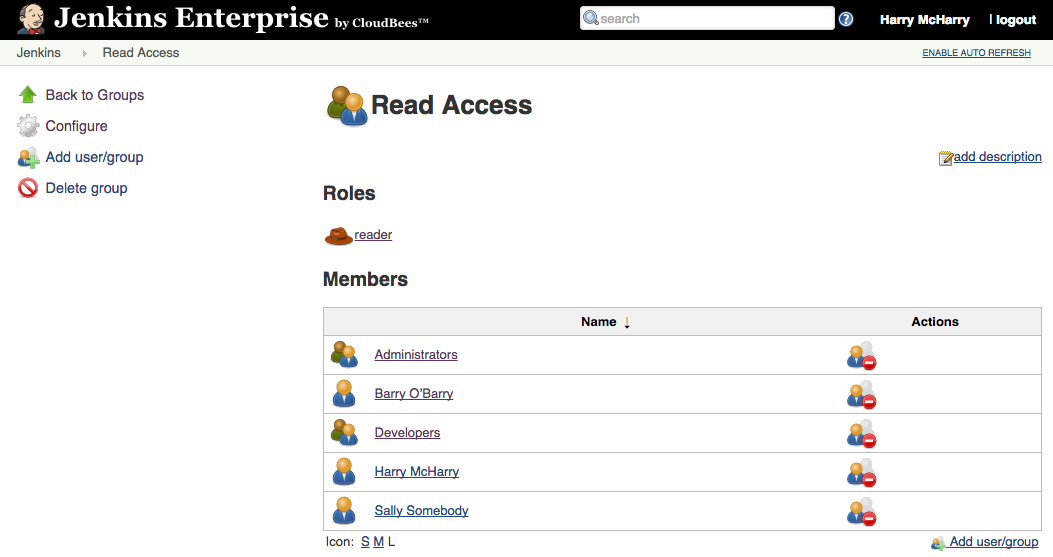
It is a little bit messy that we have added all the developers and administrators to the Read Access group as well as their respective group. It would be much better if all the members of the Developers and Administrators groups were automatically members of the Read Access group[[1]](#footnote-1).

To change this, let’s go back to the Groups management screen and click on the “Read Access” group.

Add the following groups as members: “Developers” and “Administrators”:



If you have spelled the names correctly they should show up with the Group icon and not the single user icon. Your screen should look something like this:



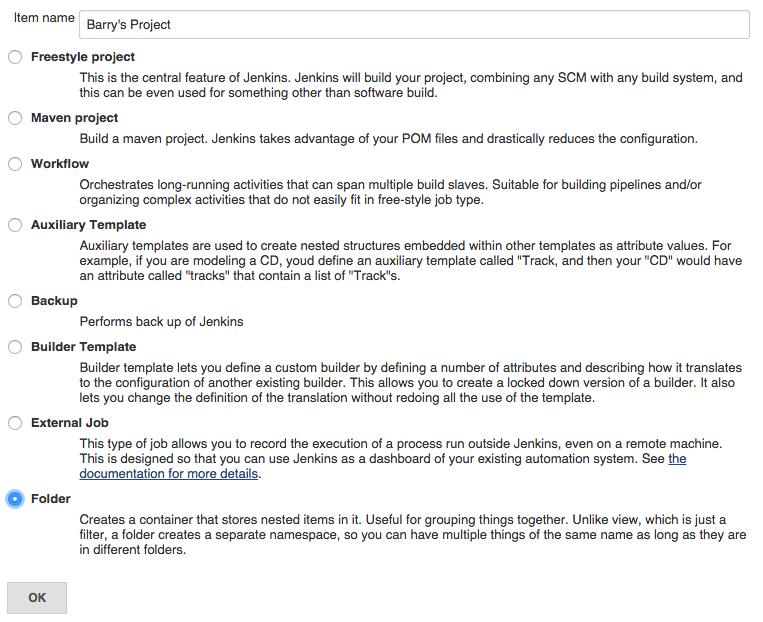
Now we can remove the Harry and Sally users from the Read Access group as they are already included by virtue of their group membership. To do this, hit the JEBC Screenshots:remove.PNGicon next to Harry and Sally’s names and hit the JEBC Screenshots:confirmremove.PNGto confirm their removal. You should now have only “Administrators”, “Developer”, and “Barry O’Barry” listed as members of the “Read Access” group.

**Note:** if we were using a non-toy Security realm, such as Active Directory, Unix user/group database, LDAP, etc. Then the external group name could alternatively be added as a member of the group to achieve similar effects.

Step 7. Create a secret project

We will now create a secret project of which Barry will be the sole developer.

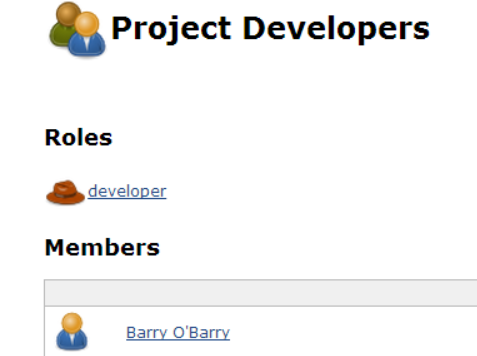
First, make sure you are logged in as harryh. Now create a new folder in the root of Jenkins called “Barry’s project” by clicking on the “New Job” link in the left-hand menu and selecting the “Folder” type for the job:



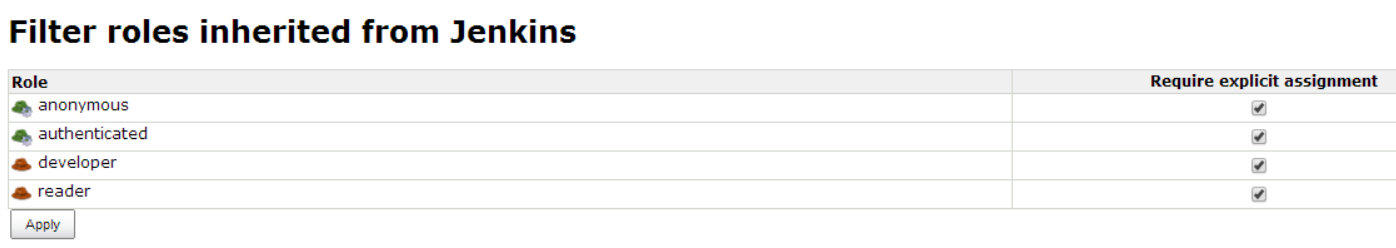
Once the folder has been created, open it up and click on the “Groups” link in the left-hand menu.

We’re going to create a new group for this folder, so hit the “New Groups” link in the left-hand menu and name it “ProjectDevelopers”.

Grant the group the “Developer” role, and on the next page add *barryb* as a member of the group:



Now go back to the “Barry’s Project” folder and click on the “Roles” link in the left-hand menu, then click the “Filter” link and mark all roles as requiring explicit assignment:



What are we doing here? Normally, permissions granted closer to the root gets inherited through to folders and jobs in them. So Sally, who has the developer role to the root, will also get the developer role in the “Barry’s project” folder.

When we check “require explicit assignment”, we are essentially saying that someone explicitly needs to be assigned this role in this folder to be able to get the role. In this way, Sally no longer has the developer role in this particular folder.

Logout and login as barryb. Verify that you can create and build jobs, but only in the Barry’s project folder.

Logout and login as sallys. Verify that you cannot see the project Barry’s project. Try to access the project directly using the URL, e.g. <http://localhost:8080/job/Barry%27s%20project/> and confirm that you get a 404 error page.

Step 8. Extra credit

Create a new ‘manager’ role with the Group/Manage; Group/View and Role/View permissions.

Create a group within the “Barry’s project” folder called “ProjectManagers”

assigned this role but pinned, and with barryb as a member.

Login as barryb and then add sallys to the ProjectDeveloper’s group. Barry will only be able to manage the existing groups directly in the Barry’s project folder. He will not be able to manage or view the groups in sub-folders or child jobs.

Verify that sallys can now see Barry’s project.

As different users visit the <http://localhost:8080/roles/whoAmI> and <http://localhost:8080/job/Barry%27s%20project/roles/whoAmI> screens to see the information reported.

As the harryh user, visit the <http://localhost:8080/job/Barry%27s%20project/roles> and <http://localhost:8080/roles> screens to see how the effective permissions are reported.

Refine the Read Access group some more by replacing the user ‘barryb’ with the system identity ‘authenticated’.

Open the Read Access out to everyone by adding the other system identity ‘anonymous’ to the Read Access group. Note that Barry’s project is still as secret as Barry configured it to be.

1. In this lab, the admin role is a superset of the reader role, so every member in the Administrators group gets every permission enjoyed by the Read Access group. Therefore, technically speaking, adding the Administrators group as a member of the Read Access group is redundant. [↑](#footnote-ref-1)