Jenkins Authorization

Jenkins has a flexible authorization system that allows you to control access to resources based on user roles, groups, and other criteria. Here are some examples of how to configure authorization in Jenkins:

1. Role-based Authorization:

   Jenkins provides a built-in mechanism for assigning roles to users or groups, and restricting access to resources based on those roles. To configure role-based authorization in Jenkins, go to "Manage Jenkins" -> "Configure Global Security". Under the "Authorization" section, select "Role-Based Strategy" as the authorization method. You can then define roles and assign them to users or groups, and specify permissions for those roles.
2. LDAP/Active Directory Authorization:

   If you are using LDAP or Active Directory for authentication, you can also use LDAP/AD groups to assign permissions and control access to resources in Jenkins. To configure LDAP/AD authorization in Jenkins, go to "Manage Jenkins" -> "Configure Global Security". Under the "Authorization" section, select "LDAP Group Membership" as the authorization method. You'll need to configure the LDAP settings, including the group search base and group search filter, and then specify the Jenkins roles that should be granted to users in those LDAP groups.
3. Matrix-based Authorization:

   Matrix-based authorization allows you to define permissions for individual users and groups on a per-resource basis. To configure matrix-based authorization in Jenkins, go to the configuration page for the resource you want to protect (e.g., a job or a node), and click "Configure". Under the "Authorization" section, select "Matrix-based Security" as the authorization method. You can then specify which users and groups have access to the resource, and what level of permissions they have.
4. Project-based Authorization:

   Jenkins also allows you to configure authorization based on project-level roles. This is useful if you want to assign different permissions to different teams or individuals for specific projects. To configure project-based authorization in Jenkins, go to the configuration page for the project you want to protect, and click "Configure". Under the "Security" section, you can then specify the roles and permissions for that project.

Roles-Based Authorization

Roles-Based Authorization is a popular and flexible method of controlling access to resources in Jenkins. In this method, you define a set of roles, each with a set of permissions, and assign those roles to users or groups. Here are the steps to configure Roles-Based Authorization in Jenkins:

1. Install the Role-Based Authorization Strategy plugin. Go to "Manage Jenkins" -> "Manage Plugins" -> "Available" and search for "Role-Based Authorization Strategy". Install the plugin and restart Jenkins.
2. Go to "Manage Jenkins" -> "Configure Global Security". Under the "Authorization" section, select "Role-Based Strategy" as the authorization method.
3. Click "Add" to create a new role. Give the role a name and a description.
4. Specify the permissions for the role. You can select from a list of predefined permissions or create custom permissions. For example, you might give the role permission to create and run jobs, but not permission to delete jobs.
5. Assign the role to users or groups. You can assign roles to specific users or groups, or use wildcards to assign roles to all users in a particular group or with a specific email domain.
6. Repeat steps 3-5 to create additional roles and assign them to users or groups as needed.
7. Test the configuration by logging in as a user and verifying that the user can access the resources that they are authorized to access.

LDAP/Active Directory Authorization

Here's an example of how to configure LDAP/Active Directory Authorization in Jenkins:

1. Install the LDAP plugin in Jenkins. Go to "Manage Jenkins" -> "Manage Plugins" -> "Available" and search for "LDAP Plugin". Install the plugin and restart Jenkins.
2. Go to "Manage Jenkins" -> "Configure Global Security". Under the "Security Realm" section, select "LDAP" as the authentication method.
3. Configure the LDAP settings. Here's an example of what the settings might look like:
    - Server: the hostname or IP address of your LDAP server
    - Root DN: the root distinguished name for your LDAP directory
    - User search base: the base DN for searching for users in the directory
    - Group search base: the base DN for searching for groups in the directory
    - Manager DN: the distinguished name of the LDAP user that Jenkins will use to bind to the directory (optional)
    - Manager Password: the password for the LDAP user (optional)
4. Test the LDAP connection. Click the "Test LDAP settings" button to verify that Jenkins can connect to the LDAP server and retrieve user information.
5. Configure LDAP authorization. Under the "Authorization" section, select "LDAP Group Membership" as the authorization method. You'll need to configure the group search base and group search filter, and then specify the Jenkins roles that should be granted to users in those LDAP groups.
    - Group search base: the DN under which groups will be searched

- Group search filter: the filter to apply when searching for groups, e.g. `(objectClass=group)`
- Group membership: the name of the LDAP attribute that identifies group membership, e.g. `memberOf`
- Role to add: the Jenkins role that should be granted to users in the LDAP group

6. Save the configuration and test authentication. Once the LDAP and authorization settings are saved, try logging in to Jenkins with an LDAP user account. If everything is configured correctly, Jenkins should authenticate the user and grant the appropriate roles based on LDAP group membership.

Matrix based Authorization

Here's an example of how to configure Matrix-based Authorization in Jenkins:

1. Go to "Manage Jenkins" -> "Configure Global Security". Under the "Authorization" section, select "Matrix-based security" as the authorization method.
2. Click "Add user/group" to add a new user or group to the matrix.
3. Select the permissions that should be granted to the user or group. You can select from a list of predefined permissions or create custom permissions.
4. Repeat steps 2-3 to add additional users or groups to the matrix and specify their permissions.
5. Save the configuration and test authentication. Once the matrix is configured, try logging in to Jenkins with a user account that has been granted one or more permissions. If everything is configured correctly, Jenkins should authenticate the user and grant the appropriate permissions based on the matrix configuration.

Matrix-based Authorization allows for fine-grained control over access to resources in Jenkins. You can grant or restrict access to specific jobs, build steps, plugins, and other resources. This method can be combined with other authorization methods in Jenkins, such as LDAP or Roles-Based Authorization, to provide even more control over access to resources.

Project-based Authorization

Project-based Authorization is a method of controlling access to specific Jenkins projects, such as jobs or pipelines. Here's an example of how to configure Project-based Authorization in Jenkins:

1. Go to the project that you want to control access to, and click "Configure".
2. Under the "Build Authorization" section, select "Project-based Matrix Authorization".
3. Click "Add user/group" to add a new user or group to the matrix.

4. Select the permissions that should be granted to the user or group for this project. You can select from a list of predefined permissions or create custom permissions.
5. Repeat steps 3-4 to add additional users or groups to the matrix and specify their permissions for this project.
6. Save the configuration and test access. Once the matrix is configured, try logging in to Jenkins with a user account that has been granted one or more permissions for this project. If everything is configured correctly, Jenkins should grant the appropriate permissions and allow the user to access the project.

Project-based Authorization can be used to control access to specific resources within a Jenkins instance, without affecting other resources. This method is particularly useful when you need to grant different levels of access to different users or groups for different projects.