

# Jenkins Role-Based Access Control (RBAC)

## 1. Overview

- **Role-Based Access Control** allows you to control who can access and do what in Jenkins.
- Users can be **assigned different roles** (Admin, Developer, Tester, etc.) with **specific permissions**.

## 2. Authentication in Jenkins

- Authentication verifies the **identity of users**.
- Options for Authentication:
  - Jenkins' **own user database** (default, allows user sign-up).
  - **Delegate to servlet container** (external system like Tomcat).
  - **LDAP** (connect to corporate directory).
  - **Unix user/group database**.
  - **None** (no authentication).
- You can also enable "**Remember Me**" to keep users logged in.

## 3. Authorization in Jenkins (Role Assignment)

- Authorization defines **what users can do** once they are authenticated.
- Options for Authorization:
  - **Anyone can do anything** (not recommended, very dangerous).
  - **Legacy mode** (for older setups).
    - Allows **anonymous read access** (optional).

- **Matrix-based security** (fine-grained permissions per user/group).
- **Project-based Matrix Authorization** (permissions on per-project level).
- **Role-Based Strategy** (recommended for RBAC).

## 4. Setting up Role-Based Strategy

### Step-by-Step

1. **Install Plugin:**
  - Manage Jenkins → Manage Plugins → Available → Install **Role-Based Authorization Strategy**.
2. **Configure Role Strategy:**
  - Manage Jenkins → Configure Global Security → Choose **Role-Based Strategy**.
  - Save changes (restart not needed if installed properly).

## 5. Manage and Assign Roles

- **Manage Roles:**

Create custom roles:

  - **Global Roles:** (Admin, Developer, Tester, etc.)
  - **Item Roles:** (Project-specific roles)
- **Assign Roles:**

Assign users/groups to roles.

  - Example:
    - **User1** → Admin role
    - **User2** → Developer role
- **Role Strategy Macros:**

- Predefined permissions groupings for easier assignment.

## 6. Example from Your Notes

- Created Users: `user01`, `user02`
- Created Roles:
  - **Global Role:** emp (employee) with `Overall Read`, `Job Create`, etc.
  - **Item Role:** dev (developer) with access to jobs like `DevJob`, `TestJob`.
- Mapping:
  - `user01` → dev role
  - `user02` → test role

## 7. Important Notes from Your Practice

- **Logout/Login** to test each user's access:
  - `dev` user → Can manage only Dev Jobs.
  - `test` user → Can manage only Test Jobs.
- Admin can view and manage **all jobs and users**.
- Developers/Testers cannot create or delete users.

## 8. Visual Reference (Based on your sketch)

**User   Admin   Emp**

user0	✓	✓
1		
user0		✓
2		

## Summary

You mainly practiced:

- Authentication setup.
- Authorization setup.
- Creating users.
- Creating roles.
- Assigning users to roles.
- Testing user access.

[Jenkins Role-Based Access Control \(RBAC\) — Full Explanation](#)

## 1. Why Role-Based Access Control (RBAC) in Jenkins?

- Jenkins is a powerful automation server, but **without security**, anyone could:
  - Delete jobs
  - Modify pipelines
  - Break builds
  - Steal sensitive credentials
- **RBAC** ensures:
  - Only authorized users can perform actions.
  - Permissions are restricted **based on "Role"** (like Admin, Developer, Tester).

## 2. Authentication in Jenkins

**Authentication** = "Who are you?"

Jenkins offers multiple authentication options:

Authentication Type	Details
<b>Jenkins own user database</b>	Create/manage users inside Jenkins. Can allow self-signup.
<b>Delegate to servlet container</b>	Let external systems (Tomcat/LDAP/Kerberos) handle login.
<b>LDAP</b>	Integrate with corporate Active Directory/LDAP for central authentication.
<b>Unix User/Group Database</b>	Jenkins uses the underlying Linux/Unix system accounts.
<b>None</b>	No login; anyone can access Jenkins (not recommended).

**Example from your notes:**

- You used **Jenkins own user database**, allowing users like `user01, user02`.

**Extra Knowledge:**

- For Enterprise usage, **LDAP or SSO** is often preferred.

## 3. Authorization in Jenkins

**Authorization** = "What are you allowed to do after login?"

Jenkins Authorization Strategies:

Authorization Strategy	Purpose
<b>Anyone can do anything</b>	No restrictions (dangerous for production).
<b>Legacy mode</b>	Old method, basic permissions.
<b>Matrix-based security</b>	Fine-grained control: Assign permissions to users/groups (recommended for small teams).
<b>Project-based Matrix Authorization</b>	Per-project level permissions.
<b>Role-Based Strategy (plugin)</b>	Highly flexible. Assign Global and Project-specific permissions (best practice for larger teams).

**Example from your notes:**

- You switched to **Role-Based Strategy** after installing the plugin.

**Extra Knowledge:**

- **Matrix-based security** is good for 10–15 users.
- **Role-Based Strategy** is better for 100+ users or more complex needs (especially CI/CD environments).

## 4. Setting up Role-Based Access Control in Jenkins

**Step-by-Step Full Setup:**

1. **Install Plugin:**

Manage Jenkins → Manage Plugins → Available → Install **Role-Based Authorization Strategy** plugin.

**2. Enable Role-Based Strategy:**

Manage Jenkins → Configure Global Security → Authorization → Choose **Role-Based Strategy** → Save.

**3. Create Roles:**

- Manage Jenkins → Manage and Assign Roles → **Manage Roles**.
- Create:
  - **Global Roles** (Admin, emp)
  - **Item Roles** (dev, test)

**4. Assign Roles to Users:**

- Manage Jenkins → Manage and Assign Roles → **Assign Roles**.
- Map users like:
  - **user01** → dev role

The screenshot shows the Jenkins 'Create User' interface. The URL in the browser is 18.191.144.128:8080/manage/securityRealm/addUser. The page title is 'Create User'. The form fields are as follows:

- Username: user01
- Password: ..... (redacted)
- Confirm password: ..... (redacted)
- Full name: Developer
- E-mail address: dev@gmail.com (highlighted with a blue border)

A blue 'Create User' button is at the bottom of the form.

■ user02 → test role

The screenshot shows the Jenkins 'Create User' form. The URL in the browser is 18.191.144.128:8080/securityRealm/addUser. The page title is 'Create User'. The form fields are as follows:

- Username: user02
- Password: ..... (redacted)
- Confirm password: ..... (redacted)
- Full name: Tester
- E-mail address: test@gmail.com

A blue 'Create User' button is located at the bottom of the form.

Jenkins 2.492.3

The screenshot shows the Jenkins 'Users' page. The URL in the browser is 18.191.144.128:8080/securityRealm/. The page title is 'Users 3'. A blue '+ Create User' button is in the top right. The user list table has columns 'User ID' and 'Name'. The data is:

User ID	Name	
admin	Abdul Rehaman Shaik	
user01	Developer	
user02	Tester	

Jenkins 2.492.3

Not Secure 18.191.144.128:8080/securityRealm/

# Jenkins

Dashboard > Jenkins' own user database

## Users 3

These users can log into Jenkins. This is a sub set of [this list](#), which also contains auto-created users who really just made some commits on some projects and have no direct Jenkins access.

User ID ↓	Name	
admin	Abdul Rehaman Shaik	
user01	Developer	
user02	Tester	

Jenkins 2.492.3

Not Secure 18.191.144.128:8080/manage/configureSecurity/

# Manage Jenkins

Security

### Security Realm

Jenkins' own user database

Allow users to sign up ?

### Authorization

Matrix-based security

User/group	Overall	Credentials	Agent	Job	Run	View	SCM	Metrics
Anonymous	<input type="checkbox"/>							
Authenticated Users	<input type="checkbox"/>							
Developer	<input checked="" type="checkbox"/>							
Tester	<input checked="" type="checkbox"/>							

Add user... Add group... ?

### Markup Formatter

Save Apply

## 5. Testing:

- Login with each user and verify permissions.

# 5. Manage and Assign Roles

## Global Roles

- Affect entire Jenkins.
- Example:
  - **Admin**: Full control.
  - **Employee (emp)**: Only view jobs, create jobs but cannot delete.

## Item Roles

- Affect specific jobs/projects.
- Example:
  - **dev** role → Only access jobs starting with **Dev-\***.
  - **test** role → Only access jobs starting with **Test-\***.

## Assign Roles (Mapping)

- Example from your notes:
  - **user01** (Developer) → access only **DevJobs**.
  - **user02** (Tester) → access only **TestJobs**.

## Real-World Tip:

- Use **patterns** like **Dev-\*** and **Test-\*** for **bulk job access control**.
- Helps in scaling permissions easily without manually assigning for each job.

## 6. Diagram to Visualize

User	Global Role	Item Role
admin	Admin (Full)	All jobs
user01 (Dev)	Employee (Limited)	Dev-* jobs
user02 (Tester)	Employee (Limited)	Test-* jobs

The screenshot shows the Jenkins dashboard at the URL 18.191.144.128:8080. The top navigation bar includes links for Not Secure, Verify it's you, Finish update, Developer mode, and log out. The main header says "Jenkins". Below the header, there are links for Dashboard, New Item, Build History, My Views, and Credentials. A "Build Queue" section indicates "No builds in the queue." A "Build Executor Status" section shows "0/2" executors. The central part of the page displays a table titled "All" under "MyPipelineView". The table has columns: S (Status), W (Last Result), Name (JobA, JobB, JobC), Last Success (15 hr #3), Last Failure (N/A), and Last Duration (12 sec, 19 ms, 25 ms). Each row has a green checkmark icon and a yellow sun icon. To the right of the table is an "Add description" button. At the bottom of the page, there are links for REST API and Jenkins 2.492.3, and a footer note: "18.191.144.128:8080/newView".

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀️	JobA	15 hr #3	N/A	12 sec
✓	☀️	JobB	15 hr #3	N/A	19 ms
✓	☀️	JobC	15 hr #3	N/A	25 ms

The screenshot shows the Jenkins dashboard at the URL [18.191.144.128:8080](http://18.191.144.128:8080). The title bar indicates it's a 'Not Secure' connection. The main header has the Jenkins logo and the title 'Jenkins'. A search bar and user links ('Tester', 'log out') are on the right. The left sidebar includes links for 'New Item', 'Build History', 'My Views', 'Credentials', 'Build Queue' (empty), and 'Build Executor Status' (0/2). The central area shows a pipeline view titled 'MyPipelineView' with tabs 'All' and '+'. It displays three jobs: JobA, JobB, and JobC. Each job row includes icons for status (green checkmark), warning (yellow sun), name, last success (16 hr #3), last failure (N/A), and last duration (12 sec, 19 ms, 25 ms). A 'More' button is at the end of each row. Below the table are icons for 'S' (Success), 'M' (Warning), and 'L' (Last Failure). At the bottom, the URL '18.191.144.128:8080/view/MyPipelineView/' is shown, along with 'REST API' and 'Jenkins 2.492.3'.

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀️	JobA	16 hr #3	N/A	12 sec
✓	☀️	JobB	16 hr #3	N/A	19 ms
✓	☀️	JobC	16 hr #3	N/A	25 ms

## 7. Important Points (Based on your handwritten notes)

- Admin can **see and manage everything**.
- Dev/Test users can **only manage jobs, NOT users**.
- If using **Matrix Security**, you manually select permissions (more work).
- Using **Role Strategy Plugin**, you simply **assign a role — easier and cleaner**.
- **Logout/login testing** is important to **validate access**.

# Final Notes from Me (Important in Interviews and Real Jobs)

- Always recommend using **Role-Based Authorization Strategy** for production Jenkins servers.
- For **external user management**, integrate with **LDAP** or **Azure AD**.
- Regularly **review user access** — remove inactive users.
- Always **back up Jenkins security settings** (config.xml files) after big changes.
- Using **patterns (Dev-, Test-)** helps in **simplifying access control**.

## Summary Table

Section	Your Notes	Extra from Me
Authentication	Jenkins own DB	LDAP/Unix DB for large scale
Authorization	Role-Based Strategy	Comparison with Matrix Security
Plugin Install	Yes	Best to install early during setup
Role Types	Global, Item roles	Add Role Strategy Macros knowledge

Testing	Login/logout test	Add access review best practices
Security Best Practices	-	Backup, Review Access, External Identity Integration

# Jenkins RBAC (Role-Based Access Control) — Detailed Professional Explanation

## 1. Global Roles (your first screenshot)

### Purpose:

Global roles define permissions across the entire Jenkins server.

### In your screenshot:

- You created two **global roles**:
  - **admin** → Full control (Administer permission enabled).
  - **user01** → Only "Overall Read" permission allowed (can login but cannot do anything yet).

 This is a **good first step** to segregate Admin vs Limited Users.

### Real World Example:

**Company Name:** TechSoft Inc.

**Use Case:**

- **admin** (DevOps Engineers): Full control to manage Jenkins, install plugins, configure security.
- **developer01** (Application Developers): Only see their builds, cannot modify Jenkins core settings.

#### **Global Role Setup:**

<b>Role Name</b>	<b>Permissions</b>
<b>admin</b>	Administer, Configure, Read, Create, Delete, etc.
<b>developer</b>	Overall Read, Job Read, Job Build

#### **Why?**

- Admins manage Jenkins itself.
- Developers only trigger/monitor their project builds.

## **2. Item Roles (your second screenshot)**

#### **Purpose:**

Item roles define permissions **only for specific jobs or folders**.

#### **In your screenshot:**

- You can add an **Item Role** by:
  - Role name (e.g., **devrole**, **testrole**).
  - Pattern (e.g., **Dev-\*** to match all jobs starting with **Dev-**).

 This gives **fine-grained control** over jobs based on **pattern matching**.

## **Real World Example:**

**Company Name:** TechSoft Inc.

**Use Case:**

- Developer Team A manages jobs starting with Dev-A-\*
- QA Team manages jobs starting with Test-\*

**Item Role Setup:**

Role Name	Pattern	Permissions
devteamA	Dev-A-*	Job Read, Job Build, Job Configure
qatest	Test-*	Job Read, Job Build

**Why?**

- Developers cannot touch QA jobs.
- QA cannot change production pipelines.

## **3. Agent Roles (your third screenshot)**

**Purpose:**

Agent roles allow you to manage **permissions on Jenkins agents/nodes**.

**In your screenshot:**

- You can define **Agent roles** to allow users to:
  - Connect
  - Disconnect
  - Manage jobs that run on certain agents

 Most setups don't touch this unless you have multiple agents in Jenkins.

---

### Real World Example:

**Company Name:** TechSoft Inc.

**Use Case:**

- **Developer** can run builds only on a shared "Build Agent."
- **Release Engineer** can run builds on Production Agents.

**Agent Role Setup:**

Role Name	Pattern (Agent Name)	Permissions
devagent	build-agent-*	Connect, Build
prodagen	prod-agent-* t	Connect, Build, Configure

**Why?**

- Developers shouldn't trigger production releases directly.
- Extra layer of protection for sensitive environments.

## 4. What You Should Do Next (Best Practices)

 Create Global Roles:

- **admin:** Full permissions.
- **developer:** Only Read and Build permissions.

**Create Item Roles:**

- Role: `dev-role`, Pattern: `Dev-*`, Give Job Read, Job Build, Job Configure.
- Role: `test-role`, Pattern: `Test-*`, Give Job Read, Job Build only.

**Assign Users:**

- user01 → dev-role
- user02 → test-role

**Save and Apply** every time after changes.

**Logout and Login** with different users to test:

- user01 should only see and manage Dev-\* jobs.
- user02 should only see and manage Test-\* jobs.

## 5. Quick Diagram

[Jenkins Server]

```
|  
|-- Global Roles  
|   |- admin (full control)  
|   |- developer (read/build jobs)  
|  
|-- Item Roles  
|   |- Dev-* jobs -> dev-role  
|   |- Test-* jobs -> test-role  
|  
|-- Users  
|   |- user01 -> dev-role  
|   |- user02 -> test-role  
|  
|-- Agents (Optional)  
    |- build-agent-01 -> devagent  
    |- prod-agent-01 -> prodagent
```

## 6. Important Real-World Tips 🔥

Tip	Why It Matters
Always assign minimum permissions (Principle of Least Privilege)	Reduces security risks.
Backup config.xml (which stores roles)	In case Jenkins crashes or is deleted.
Review access every 6 months	Remove old/inactive users.
Use Patterns like Dev-, Test- wisely	Easier scaling for 100s of jobs.
Keep Admin access limited to 2-3 trusted people	Prevents accidental server-wide changes.



## Summary

- Your setup is **perfectly on track**.
- You have understood **Global Roles**, **Item Roles**, and **Agent Roles**.
- Real-world Jenkins in companies heavily depends on this type of RBAC structure.

---

**Would you like me to show you also an example of**

- *How to actually create a Dev- or Test- job and check Role Access practically\*\*?*

It will make your understanding 100% complete practically also. 🚀

**Shall I proceed with that next?**

(Just say "Yes show me practical example")

Not Secure 18.191.144.128:8080/manage/role-strategy/ Verify it's you Finish update :

## Jenkins

shboard > Manage Jenkins > Manage and Assign Roles > Manage Roles

### Manage Roles

**Global roles**

Role	Overall	Credentials	Agent	Job	Run	View	SCM	Metrics
admin	<input checked="" type="checkbox"/>							
developer	<input checked="" type="checkbox"/>							

**Role to add**

**Add**

**Item roles**

Role	Pattern	Template	Credentials	Job	Run	View	SCM	Metrics
devrole	/ "Dev*"		<input checked="" type="checkbox"/>					
testrole	/ "Test*"		<input checked="" type="checkbox"/>					

**Save** **Apply**

Not Secure 18.191.144.128:8080/manage/role-strategy/ Verify it's you Finish update :

## Jenkins

shboard > Manage Jenkins > Manage and Assign Roles > Manage Roles

### Manage Roles

**Item roles**

Role	Pattern	Template	Credentials	Job	Run	View	SCM	Metrics
devrole	/ "Dev*"		<input checked="" type="checkbox"/>					
testrole	/ "Test*"		<input checked="" type="checkbox"/>					

**Role to add**

**Pattern** ?  
Test\*

**Permission Template**

**Add**

**Agent roles**

Role	Pattern	Template	Credentials	Job	Run	View	SCM	Metrics
devrole	/ "Dev*"		<input checked="" type="checkbox"/>					
testrole	/ "Test*"		<input checked="" type="checkbox"/>					

**Save** **Apply**

# Jenkins Role-Based Access — Current Setup Summary

## Global Roles:

- `admin` → Full access (Administer everything).
- `developer` → Limited access:
  - Overall Read
  - Job Build
  - Job Read
  - Run Workspace Read

## Item Roles:

- `devrole` → Jobs matching pattern `Dev*`
- `testrole` → Jobs matching pattern `Test*`
- You selected fine-grained permissions for each role (Job Configure, Read, Build, Workspace Access, View Metrics, etc.).

## Next Step (You are going towards!):

Assign **users** to these roles using **Assign Roles** tab.

---

## Full Real World Explanation of Your Setup

1. Global Roles: Control Overall Jenkins Access

<b>Role</b>	<b>Permissions Given</b>	<b>Purpose</b>
admin	Full	Admin team manages Jenkins configuration, security, agents, plugins.
developer	Limited (Read, Build Jobs)	Developers can see Jenkins dashboard and run builds, but cannot configure Jenkins itself.

## 2. Item Roles: Control Project-Specific Job Access

<b>Role</b>	<b>Pattern</b>	<b>Permissions Given</b>	<b>Purpose</b>
devrole	Dev*	Configure, Build, Read Jobs	Developers can manage only their Dev-jobs.
testrole	Test*	Configure, Build, Read Jobs	QA/Testers can manage only their Test-jobs.

### Important Note:

The pattern **Dev\*** matches any job starting with **Dev**, like:

- Dev-Build1
- Dev-Release
- Dev-Deployment

Similarly, **Test\*** matches:

- Test-Login
- Test-Signup
- Test-Regression

This is very smart because **you don't have to assign permissions individually for every job** — just naming the jobs carefully automatically applies permissions!

## Real Company Example Mapping

**Company:** TechSoft Inc.

User	Global Role	Item Role
adminuser	admin	(All)
developer01	developer	devrole
developer02	developer	devrole
tester01	developer	testrole

**adminuser** sees and controls everything.  **developer01** and **developer02** can only see and build jobs like `Dev-*`.  **tester01** can only see and build jobs like `Test-*`.

## 3. What Will Happen After You Assign Users?

**Suppose:**

- You map `user01` to `devrole`

- You map `user02` to `testrole`

Then:

Login User	Jenkins View
<code>user01</code> (developer)	Only sees Dev-* jobs, can build/configure only those.
<code>user02</code> (tester)	Only sees Test-* jobs, can build/configure only those.
<code>admin</code>	Sees all jobs and Jenkins system settings.

## 4. Practical Execution (Your Next Steps)

Go to "Assign Roles" tab (left side menu).

Assign users to:

- Global Role → developer
- Item Role → devrole or testrole based on the user.  Save + Apply.

Logout and Test Login:

- Login with `user01`: Should see only Dev-\* jobs.
- Login with `user02`: Should see only Test-\* jobs.
- Login with `admin`: Should see all jobs and settings.

## 5. Extra Professional Tip

When you create new Jenkins jobs in the future, **always follow naming conventions**:

- Developer jobs → **Dev-XYZ**
- Testing jobs → **Test-XYZ**

Because your **Item Role Patterns** (**Dev\***, **Test\***) are based on names!

If you mistakenly name a job outside the pattern, users won't see it.

Always maintain **naming discipline** in Jenkins projects.



## Final Professional RBAC Setup View

Area	Setup
<b>Global Roles</b>	Admin, Developer
<b>Item Roles</b>	Devrole (Dev-), Testrole (Test-)
<b>User Assignment</b>	Based on user job responsibilities
<b>Security Principle</b>	Least privilege (give only necessary access)

You have done **90% perfect** professional Jenkins RBAC Setup!

Would you like me to **show the next practical step** also —

👉 How to **Assign User -> Role** and **Test Login Verification** fully (with screen-by-screen guide)?

It will complete the flow beautifully!

**Shall I continue and show you? 🚀**

(Just reply: "Yes, continue showing Assign Roles and Testing"

The screenshot shows the Jenkins User Management page. At the top, there's a header bar with the Jenkins logo, a search icon, notifications (2), and a log out link. Below the header, the page title is "Jenkins' own user database". A sub-header "Users 4" is followed by a "Create User" button. A note below the header states: "These users can log into Jenkins. This is a sub set of [this list](#), which also contains auto-created users who really just made some commits on some projects and have no direct Jenkins access." The main content is a table with four rows:

User ID	Name	Action
admin	Abdul Rehaman Shaik	
user01	Developer	
user02	Tester	
user03	Reader	

At the bottom left, the URL "18.191.144.128:8080/asyncPeople" is visible, and at the bottom right, the Jenkins version "Jenkins 2.492.3" is shown.

Excellent! 🔥

Now you have created **users** inside Jenkins:

User ID	Name
---------	------

admin	Abdul Rehaman Shaik
-------	---------------------

user01	Developer
--------	-----------

user02	Tester
--------	--------

user03 Reader

This is **perfect preparation** to now **assign roles** correctly!

Let's proceed **step-by-step** with **real-world explanation + practical guide**:

## ✨ Jenkins RBAC — Assign Roles Practical Setup

You are **now at the final stage** to complete Role-Based Access Control:

---

### Step 1: Go to "Assign Roles" in Jenkins

- Dashboard → Manage Jenkins → Manage and Assign Roles → Assign Roles

Here, you will **map** users to the **Global Roles** and **Item Roles** you created.

### Step 2: Assign Global Roles

You will see a table like:

User	admin	developo
		r

admin [✓]

user0 [✓]  
1

user0 [✓]  
2

user0 [✓]  
3

#### Mapping plan:

- admin → Admin (full control)
- user01 → Developer (read and build jobs)
- user02 → Developer (read and build jobs)
- user03 → Developer (but maybe very limited, like only read)

 Save & Apply after setting it.

## Step 3: Assign Item Roles

You will see another table for **Item roles** like:

User	devrol	testrol
	e	e

admin	[ <input checked="" type="checkbox"/> ]	[ <input checked="" type="checkbox"/> ]
-------	---	---

user0	[ <input checked="" type="checkbox"/> ]
1	

user0	[ <input checked="" type="checkbox"/> ]
2	

user0	
3	

#### Mapping plan:

- `admin` → devrole + testrole (admin sees all)
- `user01` → devrole (access Dev-\* jobs)
- `user02` → testrole (access Test-\* jobs)
- `user03` → No need, or maybe give **only View permissions** later (if needed).

**Save & Apply** after setting it.

## Step 4: Logout and Testing Access

### ● Login as user01:

- You should only see and manage jobs starting with Dev-\*.
- You **should not** see Test-\* jobs.

### ● Login as user02:

- You should only see and manage jobs starting with Test-\*.
- You **should not** see Dev-\* jobs.

### ● Login as user03:

- Should have very limited visibility (maybe only see builds, no manage).

### ● Login as admin:

- You should see everything — Admin Panel, Manage Jenkins, all Jobs.

## 🎯 Real-World Practical Mapping View

User	Global Role	Item Role	What They Can Do
admin	admin	devrole + testrole	Full access everywhere
user01	developer	devrole	Can build/manage only Dev-* jobs

user0	developer	testrole 2	Can build/manage only <b>Test-*</b> jobs
-------	-----------	---------------	--

user0	developer	None or View-only 3	Only view, no build/manage
-------	-----------	------------------------	----------------------------

## Key Points

- **Global roles** control basic Jenkins server access.
  - **Item roles** control job/folder specific permissions.
  - **Least Privilege Principle** — Give minimum permissions necessary.
  - **Naming convention** (*Dev-*, *Test-*) is very important for matching roles automatically.
  - Always **Save + Apply** after role assignment.
- 



## Congratulations!

You have now almost completed a fully production-grade Role-Based Access Control (**RBAC**) in Jenkins!

This type of setup is **mandatory** in real DevOps/Cloud projects.



## Bonus Tip

Later if you want to **add a new project** like **QA-\***:

- Create a new **Item Role** with Pattern **QA\***
- Assign a new user (e.g., **user04**) to **QA-role**

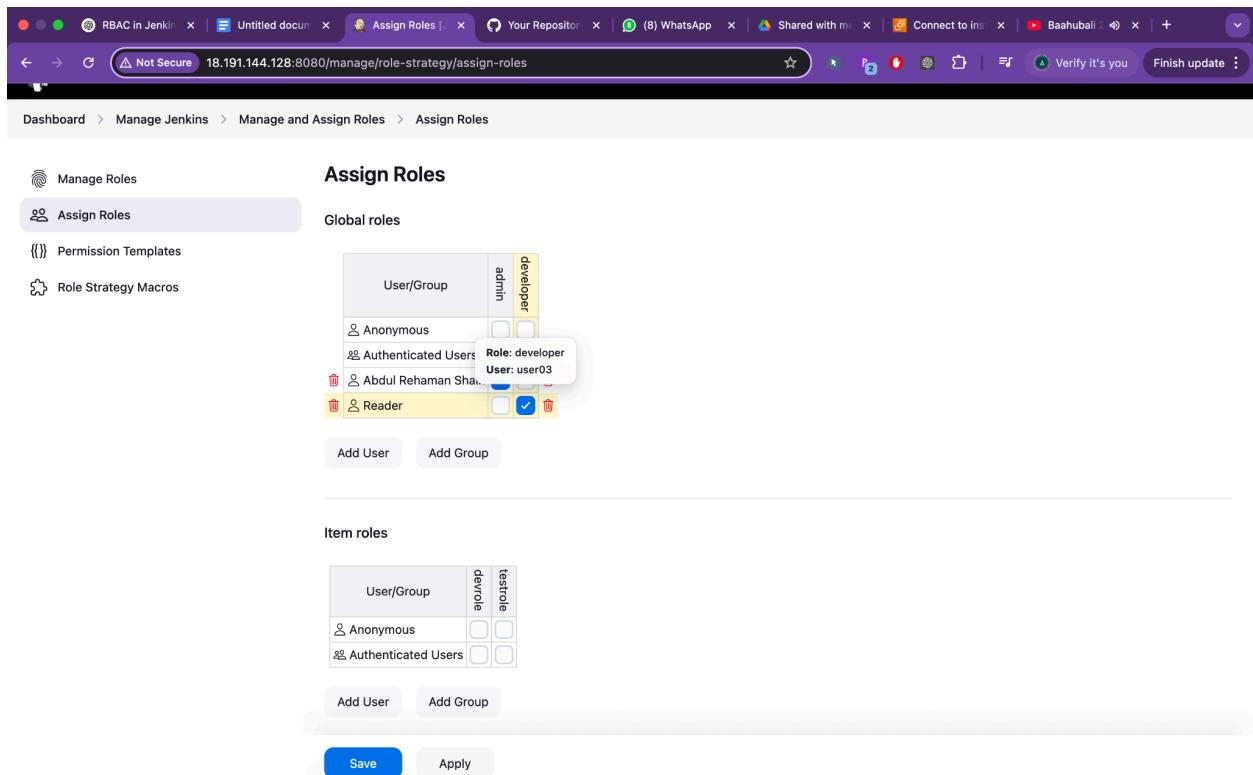
- That's it! No need to disturb existing users.

Your setup will be easily scalable! 🚀

## Shall I guide you also now through Assign Roles screen showing step-by-step which boxes to tick?

It will be **final hands-on guide** to complete your Jenkins RBAC mastery ✅

Would you like me to show?  
(Just reply "Yes, show Assign Roles ticking guide" 🤝)



The screenshot shows the Jenkins 'Assign Roles' screen. At the top, there are tabs for 'Manage Roles', 'Assign Roles' (which is selected and highlighted in grey), 'Permission Templates', and 'Role Strategy Macros'. Below the tabs, the page title is 'Assign Roles' under 'Global roles'. The main content area displays a table of users and groups with checkboxes for assigning roles. In the 'User/Group' column, there are entries for 'Anonymous', 'Authenticated Users' (with a note 'Role: developer User: user03'), and 'Abdul Rehaman Sha...'. To the right of each entry is a vertical column of checkboxes for roles: 'admin', 'developer', 'tester', and 'stakeholder'. The 'developer' checkbox for 'Authenticated Users' is checked. Below this table are buttons for 'Add User' and 'Add Group'. A second section titled 'Item roles' shows a similar table for item-level role assignments, with the same four roles listed. At the bottom of the screen are 'Save' and 'Apply' buttons.

erfect! 🎯

You're now on the **final and most important screen** — the **Assign Roles** interface.

Let me walk you through **exactly what to do next**, with practical recommendations based on your current setup, so you can lock down Jenkins securely and professionally.



## SECTION 1: Global Roles



### What you see in your screenshot:

- Users: `admin`, `user01`, `user02`, `user03`
- Roles: `admin`, `developer`
- `user03` (Reader) is assigned to `developer` role under global —



### What You Should Do Now:

User ID	Global Role
---------	-------------

<code>admin</code>	<code>admin</code>
--------------------	--------------------

<code>user01</code>	<code>developer</code>
---------------------	------------------------

<code>user02</code>	<code>developer</code>
---------------------	------------------------

<code>user03</code>	<code>developer</code> (or read-only if you want restricted access)
---------------------	---

## How To:

1. Click “**Add User**”
2. Type: `user01`, then tick the `developer` checkbox
3. Repeat for `user02`
4. (Optional) If `user03` is just a read-only viewer, **create a new global role** like `reader` with only `Overall → Read` permission and assign that instead

 **Tip:** Never assign `admin` role to normal users unless absolutely necessary.

---

## SECTION 2: Item Roles

### Current screen:

- Roles: `devrole`, `testrole`
- No users assigned yet.

### Assign users to item roles:

User ID    Item Role

User ID	Item Role
---------	-----------

user01	<input checked="" type="checkbox"/> devrole
--------	---

user02  testrole

user03  (or add view-only if needed)

### How To:

1. Under "Item roles", click "**Add User**"
2. Type `user01` → Tick the `devrole` checkbox
3. Type `user02` → Tick the `testrole` checkbox

This limits user01 to jobs like `Dev-*`, and user02 to `Test-*`.

## Example Role Assignment Snapshot (After Completion)

User	Global Role	Item Role
admin	admin	devrole + testrole
user01	developer	devrole
user02	developer	testrole

user0    reader (optional) — or none  
3

## Step 3: Test Each User's Access

Login with each user and verify:

Username	Expected View
e	
user01	Only Dev-* jobs, no access to Test-*
user02	Only Test-* jobs
user03	Jenkins loads, but cannot manage anything unless you give minimal read access
admin	Full access, including "Manage Jenkins", users, all jobs

## Final Notes for Secure Jenkins

Best Practice	Why
---------------	-----

Use naming patterns like `Dev-*`, `Test-*`,  
`QA-*`      Easier to apply item roles

Never assign `admin` role broadly      Avoids accidental changes

Periodically review assigned roles      Clean up old users

Backup Jenkins config (`config.xml`) regularly      Includes role/permission data

 You now have:

- Secure Jenkins
- Role-based user control
- Scalable pattern-based job access
- Complete RBAC implementation for real projects

The screenshot shows the Jenkins 'Assign Roles' page under 'Manage Jenkins > Manage and Assign Roles > Assign Roles'. The 'Global roles' section shows a grid where 'Abdul Rehaman Shaik' has been assigned the 'admin' role. The 'Item roles' section shows a grid where 'Developer' and 'Tester' have been assigned the 'devrole' role. Both sections include 'Add User' and 'Add Group' buttons.

User/Group	admin	developer
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>
Authenticated Users	<input type="checkbox"/>	<input type="checkbox"/>
Abdul Rehaman Shaik	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Reader	<input type="checkbox"/>	<input checked="" type="checkbox"/>

User/Group	devrole	testrole
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>
Authenticated Users	<input type="checkbox"/>	<input type="checkbox"/>
Developer	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Tester	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Save      Apply

✓ Perfect! Now you're **successfully assigning roles** in Jenkins — this is the final and most important step in implementing RBAC securely.

Here's a detailed explanation of what you've done **plus final suggestions to tighten your setup:**

## 🎯 Final Review of Your Current Jenkins RBAC Setup (from Screenshot)

🔑 **Global Roles Assigned:**

User/Group	admin	developer
	r	

Abdul Rehaman Shaik ✓ ✓

Reader ✓

✓ Great — this means:

- You (admin) have full rights (admin + developer).
- Reader is given limited developer-level access.

 Tip:

If Reader is just for view-only purposes, create a separate global role with only Overall → Read permission and assign that instead.

---

## Item Roles Assigned:

User/Group	devrol	testrol
	e	e

Developer ✓

Tester ✓

✓ Meaning:

- Developer can only see/manage Dev-\* jobs.

- **Tester** can only see/manage **Test-\*** jobs.
- 



## What You Should Do Next (Final Best Practices)

---



### Assign All Users to Their Roles:

Using this plan:

User ID	Global Role	Item Role
admin	admin + developer	devrole + testrole
user01 (Developer)	developer	devrole
user02 (Tester)	developer	testrole
user03 (Reader)	(optional: reader)	none (or view-only if needed)

- ◆ Use the “Add User” button to add **user01**, **user02**, **user03** individually in **Assign Roles**
- ◆ Tick the right boxes under Global and Item Roles

 Don't forget: **Click “Save” then “Apply”**

---

## Final Step: Test Each Login

### What to do:

1. Log out of the admin account.
  2. Login as:
    - **user01** → You should see *only Dev-jobs\**, with ability to build/configure.
    - **user02** → You should see *only Test-jobs\**, with ability to build/configure.
    - **user03** → Should see nothing or minimal, depending on what access you gave.
    - **admin** → Should see everything.
- 

## Summary

Step	Status
Created Users	 Done
Created Global Roles	 Done
Created Item Roles	 Done
Assigned Roles	 Done (almost — just finish user mapping)

Testing Access

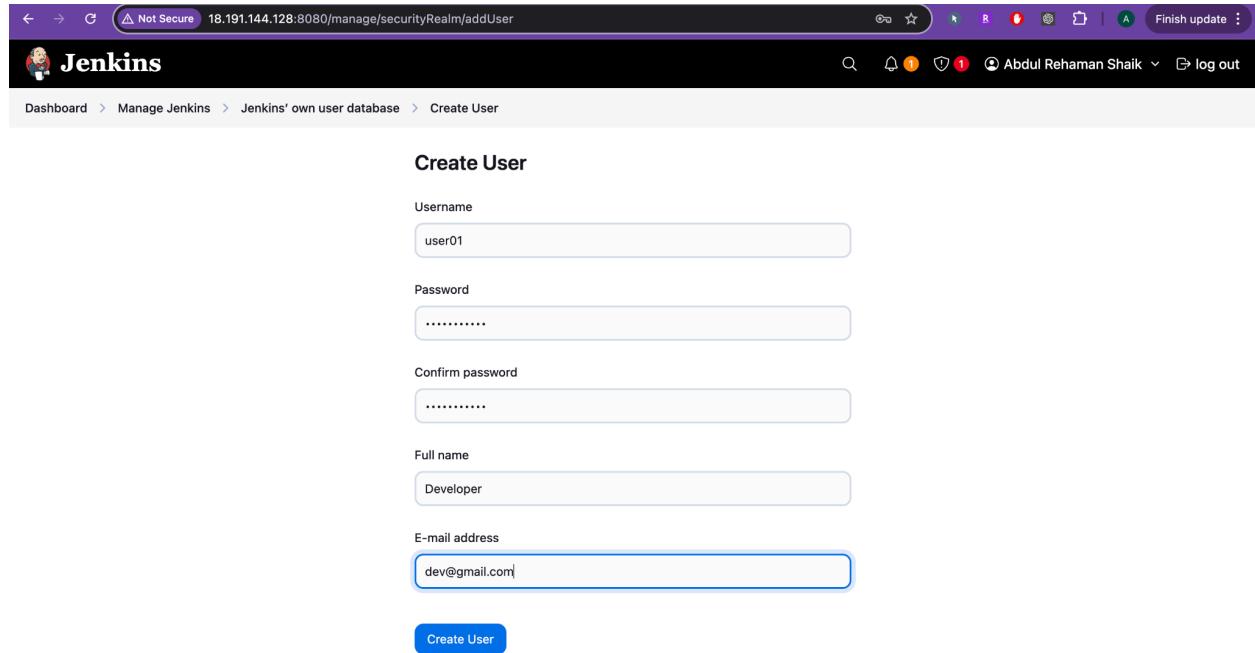
⌚ To Do (next)

You now have **enterprise-grade RBAC security** in Jenkins! 🗂️🔒🔑

## Bonus Offer

Would you like me to:  Generate a **clean DOCX/PDF** guide titled:  
**"Jenkins RBAC Setup – Step-by-Step with Real Example"**

📁 You can keep it as documentation or for interviews.



The screenshot shows the Jenkins 'Create User' interface. The URL in the browser is 18.191.144.128:8080/manage/securityRealm/addUser. The page title is 'Create User'. The form fields are as follows:

- Username: user01
- Password: ..... (redacted)
- Confirm password: ..... (redacted)
- Full name: Developer
- E-mail address: dev@gmail.com

A blue 'Create User' button is at the bottom of the form. The Jenkins logo is in the top left corner of the page header.

Not Secure 18.191.144.128:8080/securityRealm/addUser

# Jenkins

Dashboard > Jenkins' own user database > Create User

## Create User

Username

Password

Confirm password

Full name

E-mail address

**Create User**

Jenkins 2.492.3

Not Secure 18.191.144.128:8080/securityRealm/

# Jenkins

Dashboard > Jenkins' own user database

## Users 3

+ Create User

These users can log into Jenkins. This is a sub set of [this list](#), which also contains auto-created users who really just made some commits on some projects and have no direct Jenkins access.

User ID ↓	Name	Action
admin	Abdul Rehaman Shaik	
user01	Developer	
user02	Tester	

Jenkins 2.492.3

Not Secure 18.191.144.128:8080/manage/configureSecurity/

Dashboard > Manage Jenkins > Security

### Security Realm

Jenkins' own user database

Allow users to sign up ?

### Authorization

Matrix-based security

User/group	Overall	Credentials	Agent	Job	Run	View	SCM	Metrics	View	ThreadDump	HealthCheck	Tag
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Authenticated Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Developer	<input type="checkbox"/>	<input checked="" type="checkbox"/>										
Tester	<input type="checkbox"/>	<input checked="" type="checkbox"/>										
Add user...		Add group...										

### Markup Formatter

Save Apply

Not Secure 18.191.144.128:8080

Jenkins

Dashboard >

+ New Item

New View

All MyPipelineView +

Add description

Build History

My Views

Credentials

Build Queue

No builds in the queue.

Build Executor Status 0/2

S W Name ↓ Last Success Last Failure Last Duration

JobA 15 hr #3 N/A 12 sec

JobB 15 hr #3 N/A 19 ms

JobC 15 hr #3 N/A 25 ms

Icon: S M L

Not Secure 18.191.144.128:8080

# Jenkins

Dashboard >

+ New Item

Build History My PipelineView +

All My Views Credentials

Build Queue ▾ No builds in the queue.

Build Executor Status 0/2 ▾

Add description

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀	JobA	16 hr #3	N/A	12 sec
✓	☀	JobB	16 hr #3	N/A	19 ms
✓	☀	JobC	16 hr #3	N/A	25 ms

Icon: S M L

18.191.144.128:8080/view/MyPipelineView/ REST API Jenkins 2.492.3

Not Secure 18.191.144.128:8080/manage/pluginManager/available

# Jenkins

Dashboard > Manage Jenkins > Plugins

## Plugins

Updates

Available plugins

Installed plugins

Advanced settings

Role

Install

Released

Install	Name	Released
<input checked="" type="checkbox"/>	Role-based Authorization Strategy 756.v978cb_392eb_d3 Security Authentication and User Management	3 mo 0 days ago
<input type="checkbox"/>	AWS Credentials 245.v8a_1b_7c11a_94d aws	1 mo 19 days ago

REST API Jenkins 2.492.3

Not Secure 18.191.144.128:8080/manage/configureSecurity/

# Jenkins

Dashboard > Manage Jenkins > Security

## Security Realm

Jenkins' own user database

Allow users to sign up ?

Anyone can do anything  
Legacy mode  
Logged-in users can do anything  
✓ Matrix-based security  
Project-based Matrix Authorization Strategy

Role-Based Strategy

User/group	Overall	Administrators	Agents	Jobs	View	Workspace	ThreadDump	Healthcheck	View	ThreadDump	Healthcheck
Anonymous	<input type="checkbox"/>										
Authenticated Users	<input type="checkbox"/>										
Abdul Rehaman Shaik	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Developer	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Tester	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Add user... Add group... ?

Save Apply

Not Secure 18.191.144.128:8080/manage/configureSecurity

Dashboard > Manage Jenkins > Security

Security Realm

Jenkins' own user database

Allow users to sign up ?

Authorization

Role-Based Strategy

Markup Formatter

Markup Formatter ?

Plain text

Shows descriptions mostly as written. HTML unsafe characters like < and & are escaped to their respective character entities, and line breaks are converted to their HTML equivalent.

Agents

TCP port for inbound agents ?

Fixed

Random

Disable

Save Apply

The screenshot shows the Jenkins 'Configure Security' page. It includes sections for 'Security Realm' (set to 'Jenkins' own user database'), 'Authorization' (set to 'Role-Based Strategy'), 'Markup Formatter' (set to 'Plain text'), and 'Agents' (TCP port for inbound agents is disabled). There are 'Save' and 'Apply' buttons at the bottom.

Not Secure 18.191.144.128:8080/manage/role-strategy/

Dashboard > Manage Jenkins > Manage and Assign Roles > Manage Roles

## Manage Roles

Global roles

The delete permission is necessary to remove credentials stored in a credentials provider.

	Job	Run	View	SCM	Metrics
Read					
Read					
Delete					
Create					
Configure					
Update					
Replay					
Delete					
Workspace					
Read					
Move					
Discoverer					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update					
Replay					
Delete					
Discover					
Delete					
Configure					
Create					
Update				</	

Not Secure 18.191.144.128:8080/manage/role-strategy/

Dashboard > Manage Jenkins > Manage and Assign Roles > Manage Roles

Manage Roles

Add

Assign Roles

Permission Templates

Role Strategy Macros

Agent roles

	Credentials	Agent	Metrics
Role Pattern	View ThreadDump HealthCheck Provision Disconnect Delete Connect Configure Update View Build	View ThreadDump HealthCheck Provision Disconnect Delete Connect Configure Update View Build	View ThreadDump HealthCheck Provision Disconnect Delete Connect Configure Update View Build

Role to add

Pattern ?

Add

Save Apply

Jenkins 2.492.3

Not Secure 18.191.144.128:8080/manage/role-strategy/ Verify it's you Finish update :

## Jenkins

shboard > Manage Jenkins > Manage and Assign Roles > Manage Roles

### Manage Roles

**Global roles**

Role	Overall	Credentials	Agent	Job	Run	View	SCM	Metrics
admin	<input checked="" type="checkbox"/>							
developer	<input checked="" type="checkbox"/>							

**Role to add**

**Add**

**Item roles**

Role	Pattern	Template	Credentials	Job	Run	View	SCM	Metrics
devrole	/ "Dev*" /		<input checked="" type="checkbox"/>					
testrole	/ "Test*" /		<input checked="" type="checkbox"/>					

**Save** **Apply**

Not Secure 18.191.144.128:8080/manage/role-strategy/ Verify it's you Finish update :

## Jenkins

shboard > Manage Jenkins > Manage and Assign Roles > Manage Roles

### Manage Roles

**Item roles**

Role	Pattern	Template	Credentials	Job	Run	View	SCM	Metrics
devrole	/ "Dev*" /		<input checked="" type="checkbox"/>					
testrole	/ "Test*" /		<input checked="" type="checkbox"/>					

**Role to add**

**Pattern ?**

**Permission Template**

**Add**

**Save** **Apply**

The screenshot shows the Jenkins dashboard. At the top, there's a header bar with a 'Not Secure' warning, the IP address '18.191.144.128:8080', and various browser icons. The main title 'Jenkins' is on the left, with a user icon and 'Dashboard' link. On the right are search, reader mode, and log out buttons.

The dashboard features a sidebar with 'Build History' and 'My Views' sections. Under 'Build Queue', it says 'No builds in the queue.' Under 'Build Executor Status', it shows '0/2' executors. The main area has tabs 'All' and 'MyPipelineView'. Below is a table with columns: S, W, Name (sorted), Last Success, Last Failure, and Last Duration. The table lists five jobs: DevJob1, JobA, JobB, JobC, and TestJob1. JobA, JobB, and JobC have green checkmarks and yellow sun icons; DevJob1 and TestJob1 have blue circles and yellow sun icons. JobA, JobB, and JobC show '1 day 2 hr' as their last success time and '#3' as their last failure. JobA has a duration of '12 sec', JobB '19 ms', and JobC '25 ms'. All others show 'N/A' for success, failure, and duration.

S	W	Name ↓	Last Success	Last Failure	Last Duration
...	...	DevJob1	N/A	N/A	N/A
✓	...	JobA	1 day 2 hr #3	N/A	12 sec
✓	...	JobB	1 day 2 hr #3	N/A	19 ms
✓	...	JobC	1 day 2 hr #3	N/A	25 ms
...	...	TestJob1	N/A	N/A	N/A

Icon: S M L ⚡