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
user01	✓	✓
user02		✓

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
Jenkins own user database	Create/manage users inside Jenkins. Can allow self-signup.
Delegate to servlet container	Let external systems (Tomcat/LDAP/Kerberos) handle login.
LDAP	Integrate with corporate Active Directory/LDAP for central authentication.
Unix User/Group Database	Jenkins uses the underlying Linux/Unix system accounts.
None	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
Anyone can do anything	No restrictions (dangerous for production).
Legacy mode	Old method, basic permissions.
Matrix-based security	Fine-grained control: Assign permissions to users/groups (recommended for small teams).
Project-based Matrix Authorization	Per-project level permissions.
Role-Based Strategy (plugin)	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

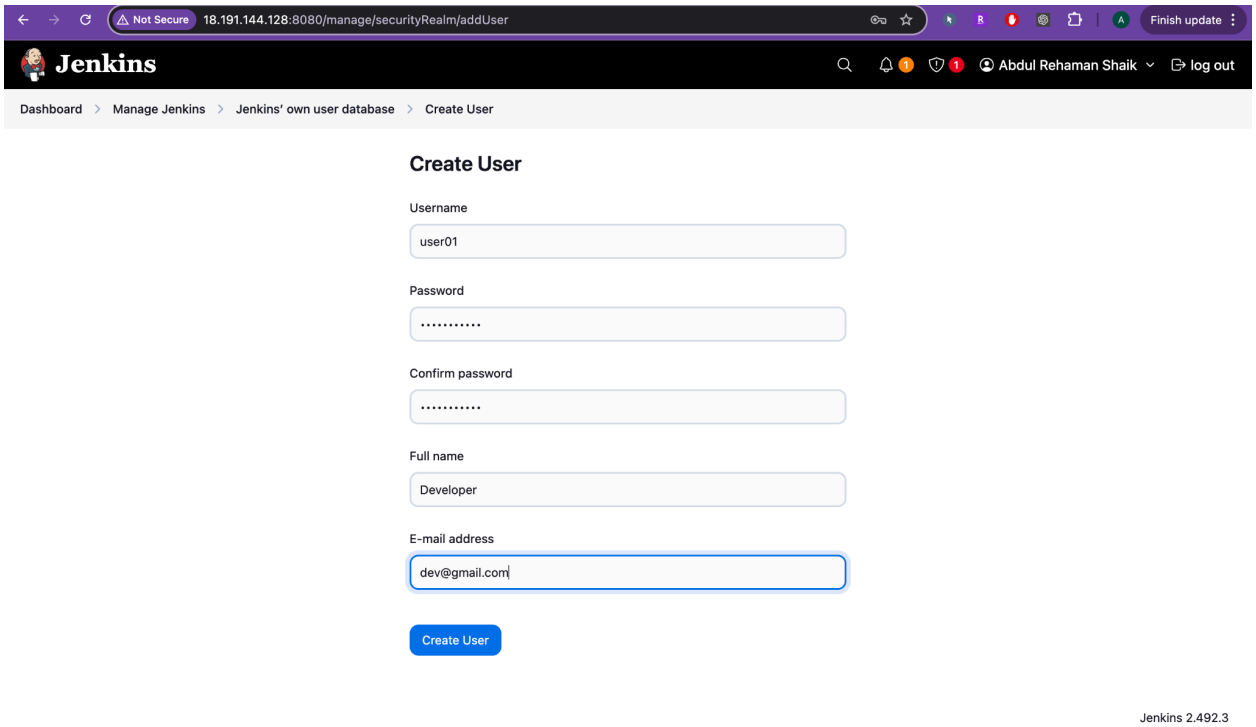
Step-by-Step Full Setup:

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.

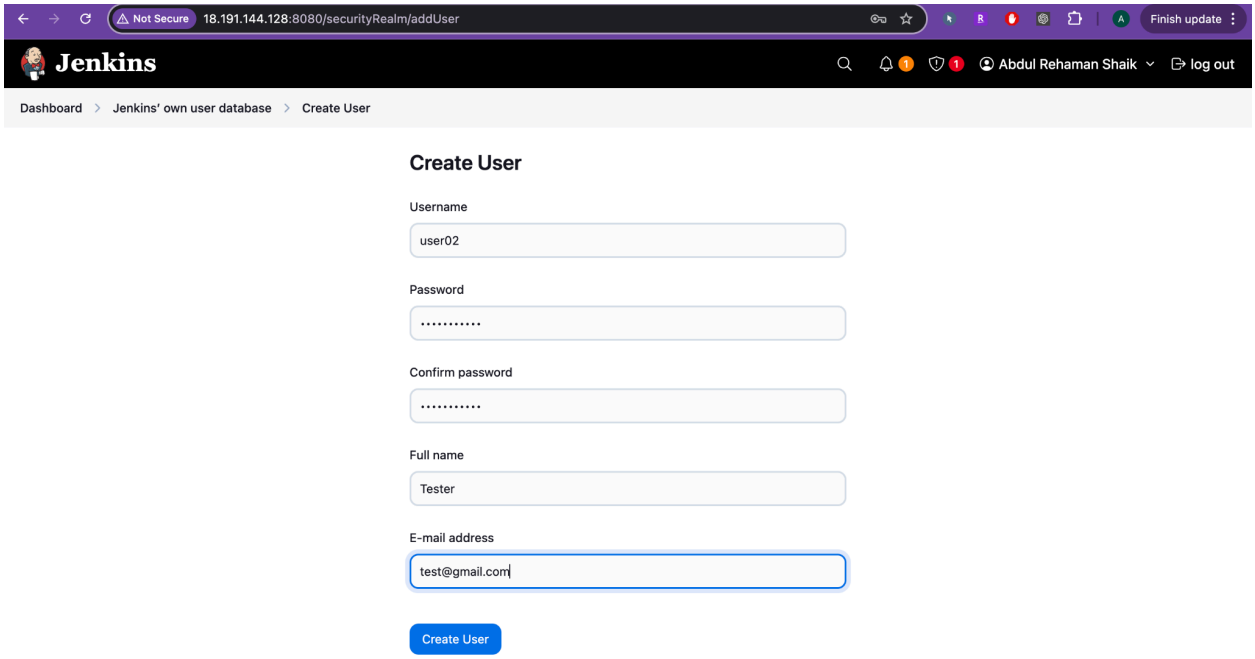
3. **Create Roles**:
   - Manage Jenkins → Manage and Assign Roles → **Manage Roles**.
   - Create:
     - **Global Roles** (Admin, emp)
     - **Item Roles** (dev, test)

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

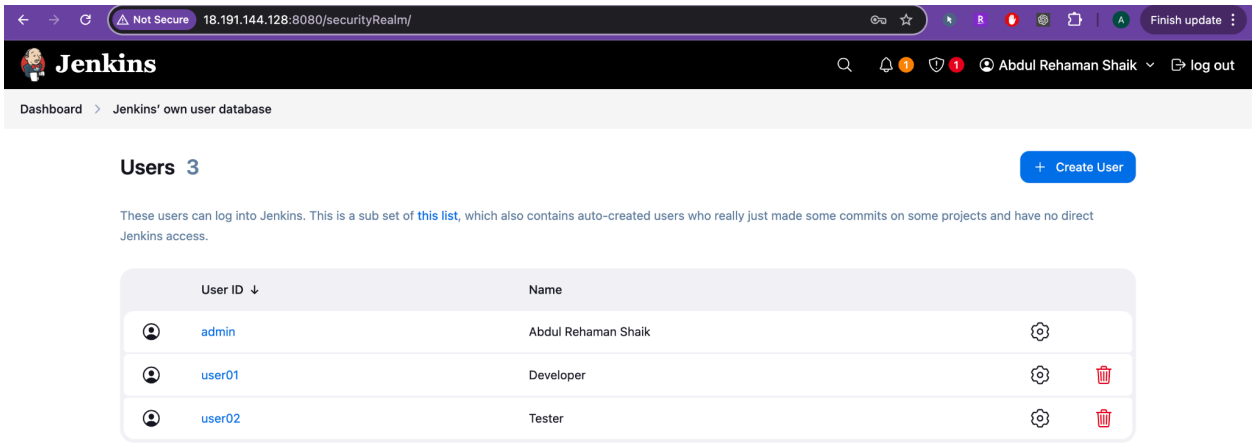


-

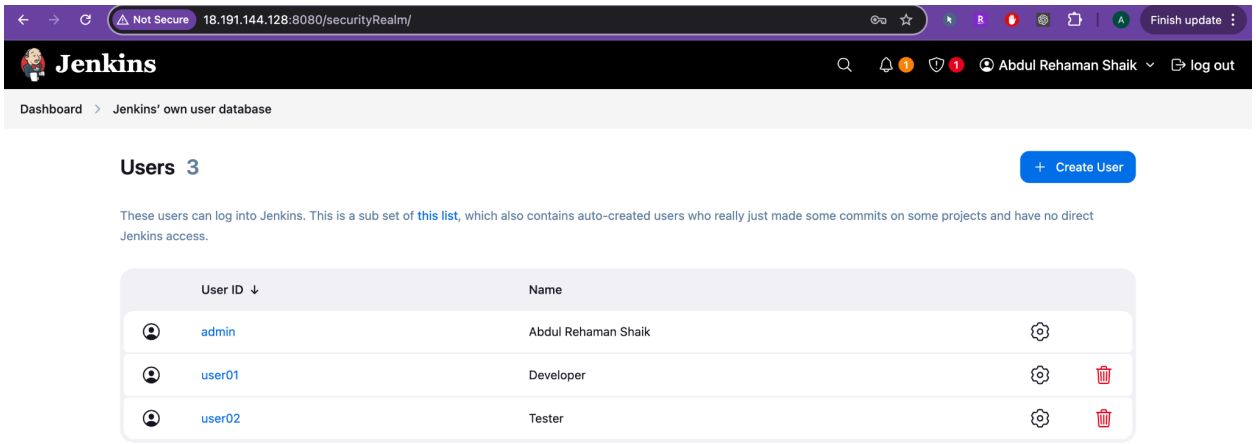
- `user02` → test role



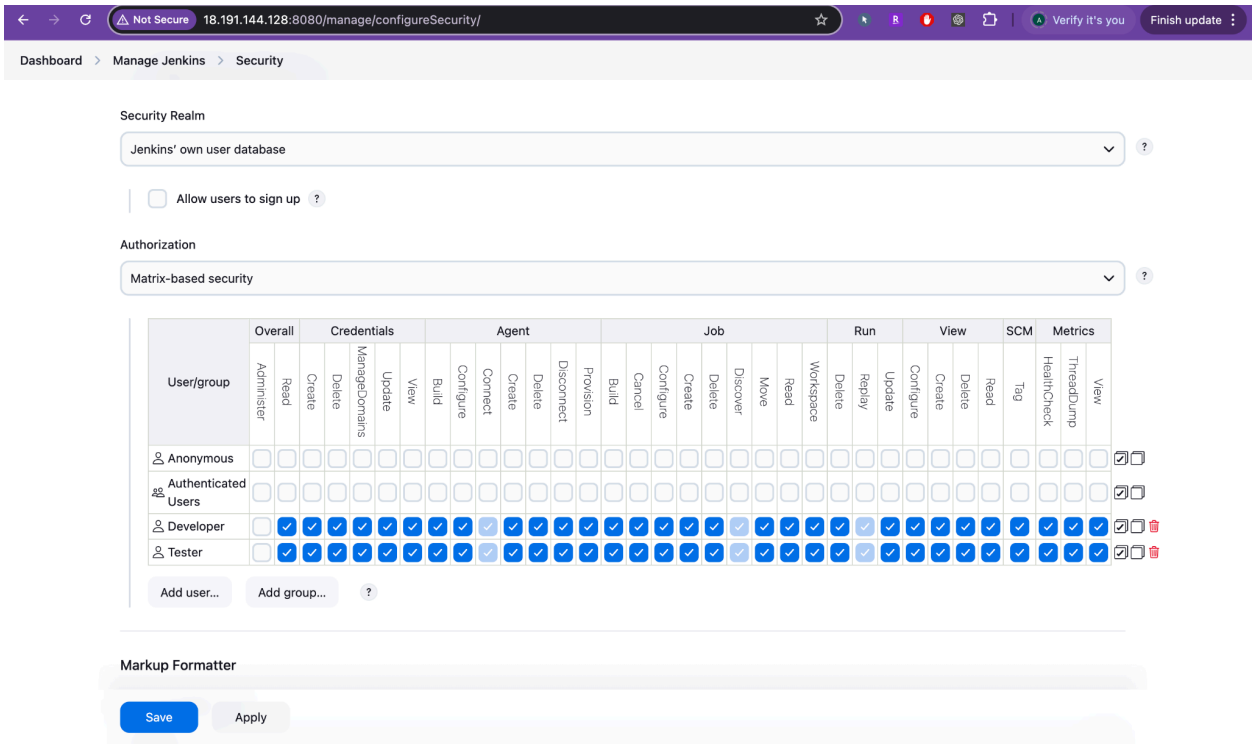
Jenkins 2.492.3



Jenkins 2.492.3



Jenkins 2.492.3



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)

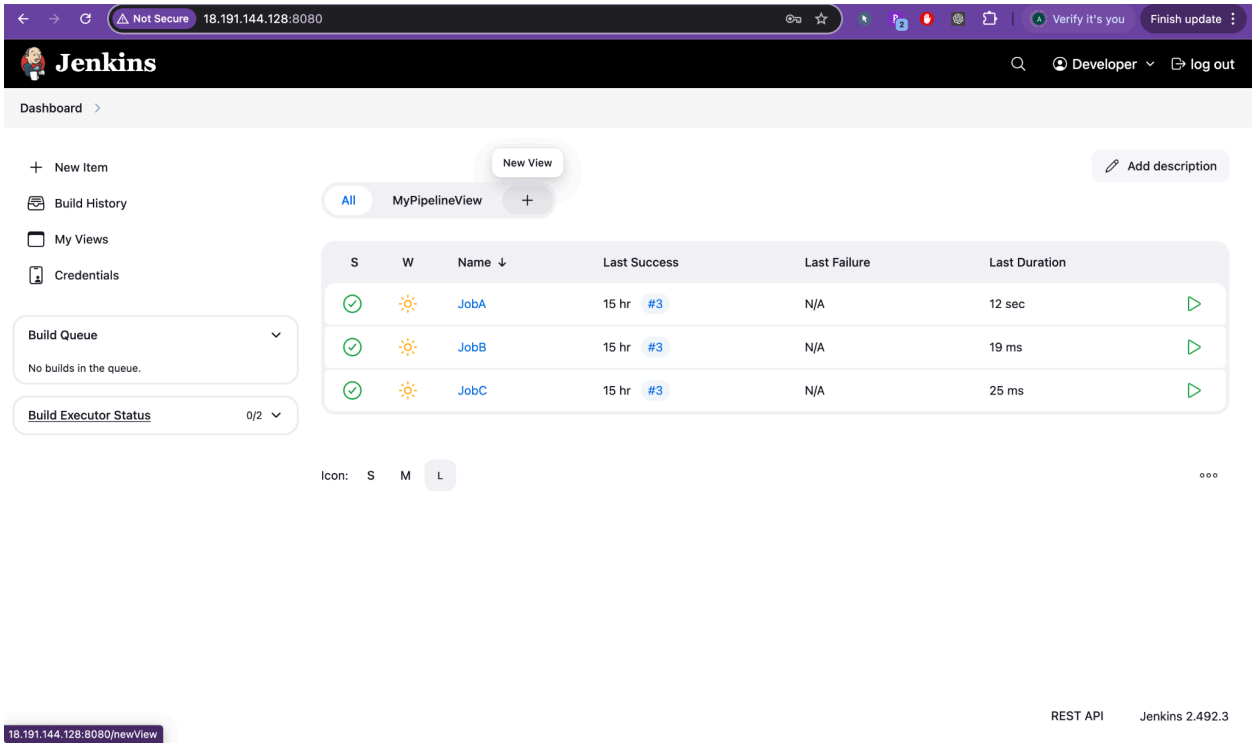
- Example from your notes:
    - `user01` (Developer) → access only `DevJobs`.
    - `user02` (Tester) → access only `TestJobs`.

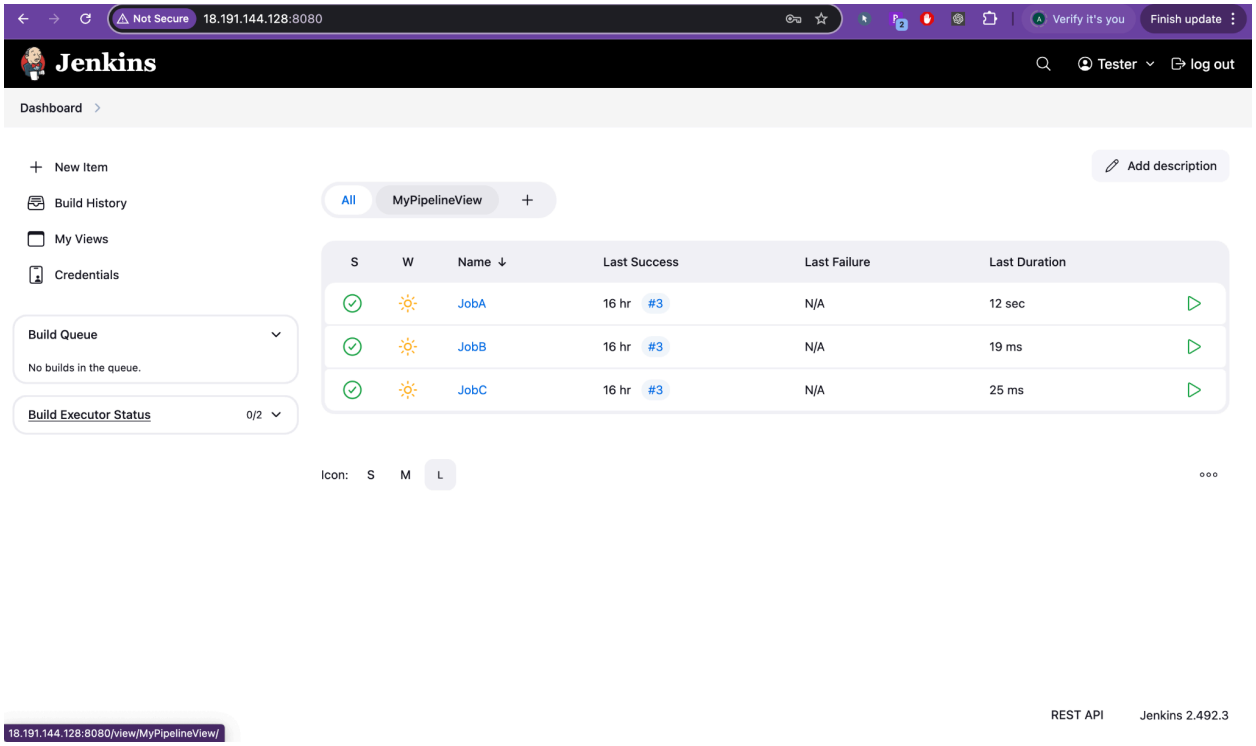
Real-World Tip:

- 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   |
+-----------------+-----------------+---------------+
```





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:

Role Name	Permissions
`admin`	Administer, Configure, Read, Create, Delete, etc.
`developer`	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
`prodagent`	`prod-agent-*`	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

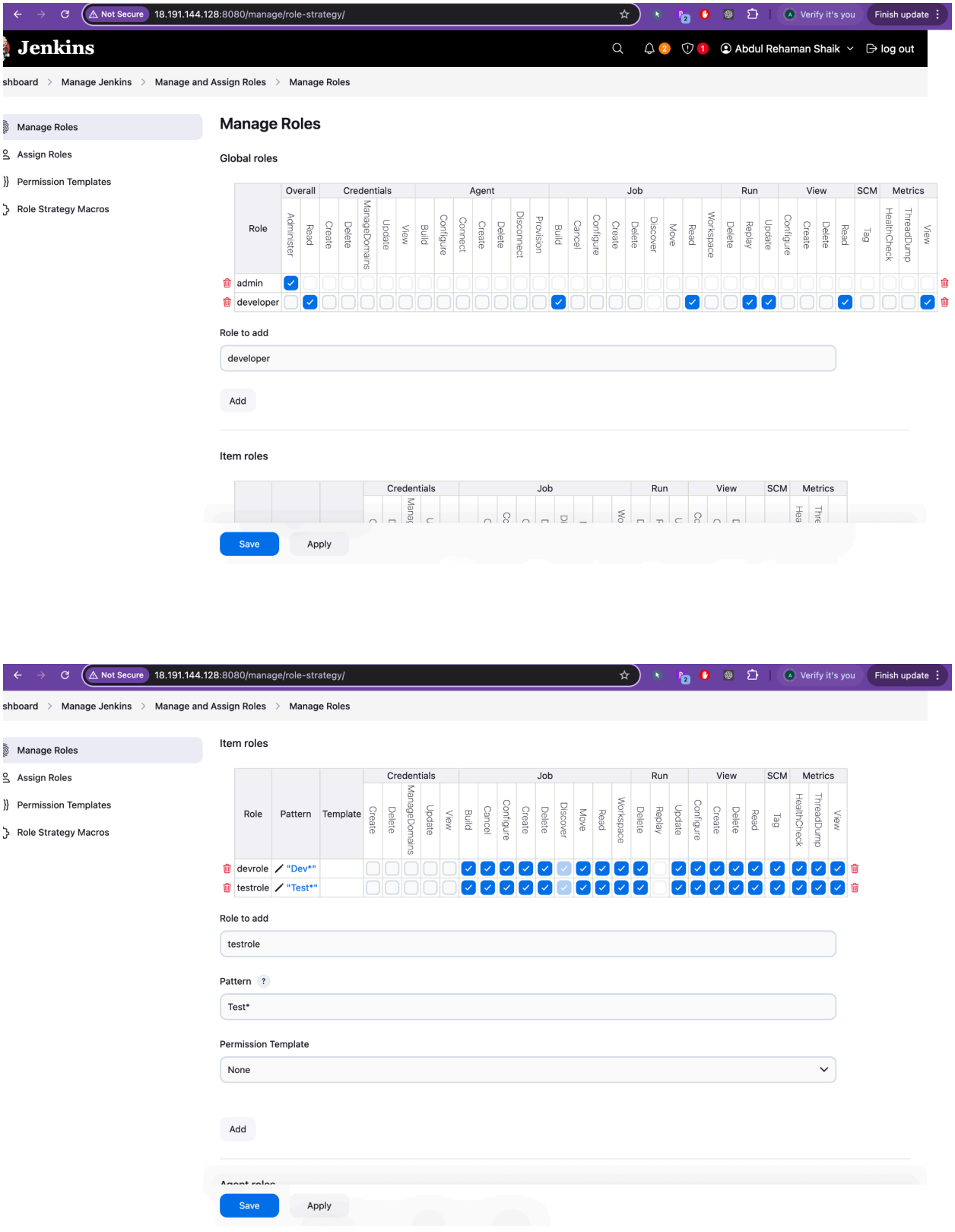
- Your setup is **perfectly on track**.
- You have understood **Global Roles**, **Item Roles**, and **Agent Roles**.
- Real-world Jenkins in companies heavily depends on this type of RBAC structure.

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
Abdul Rehaman Shaik
log out
shboard > Manage Jenkins > Manage and Assign Roles > Manage Roles
Manage Roles
Assign Roles
Permission Templates
Role Strategy Macros
Manage Roles
Global roles
Role
Overall
Administer
Read
Credentials
Create
Delete
ManageDomains
Update
View
Agent
Build
Configure
Connect
Create
Delete
Disconnect
Provision
Job
Build
Cancel
Configure
Create
Delete
Discover
Move
Read
Workspace
Run
Delete
Replay
Update
View
Configure
Create
Delete
Read
SCM
Tag
Metrics
HealthCheck
ThreadDump
View
admin
developer
Role to add
developer
Add
Item roles
Credentials
Job
Run
View
SCM
Metrics
Save
Apply
Not Secure 18.191.144.128:8080/manage/role-strategy/
Verify it's you
Finish update
shboard > Manage Jenkins > Manage and Assign Roles > Manage Roles
Manage Roles
Assign Roles
Permission Templates
Role Strategy Macros
Item roles
Role
Pattern
Template
Credentials
Create
Delete
ManageDomains
Update
View
Job
Build
Cancel
Configure
Create
Delete
Discover
Move
Read
Workspace
Run
Delete
Replay
Update
View
Configure
Create
Delete
Read
SCM
Tag
Metrics
HealthCheck
ThreadDump
View
devrole "Dev*"
testrole "Test*"
Role to add
testrole
Pattern
Test*
Permission Template
None
Add
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

Role	Permissions Given	Purpose
`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

Role	Pattern	Permissions Given	Purpose
`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
`user01` (developer)	Only sees Dev-* jobs, can build/configure only those.
`user02` (tester)	Only sees Test-* jobs, can build/configure only those.
`admin`	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.

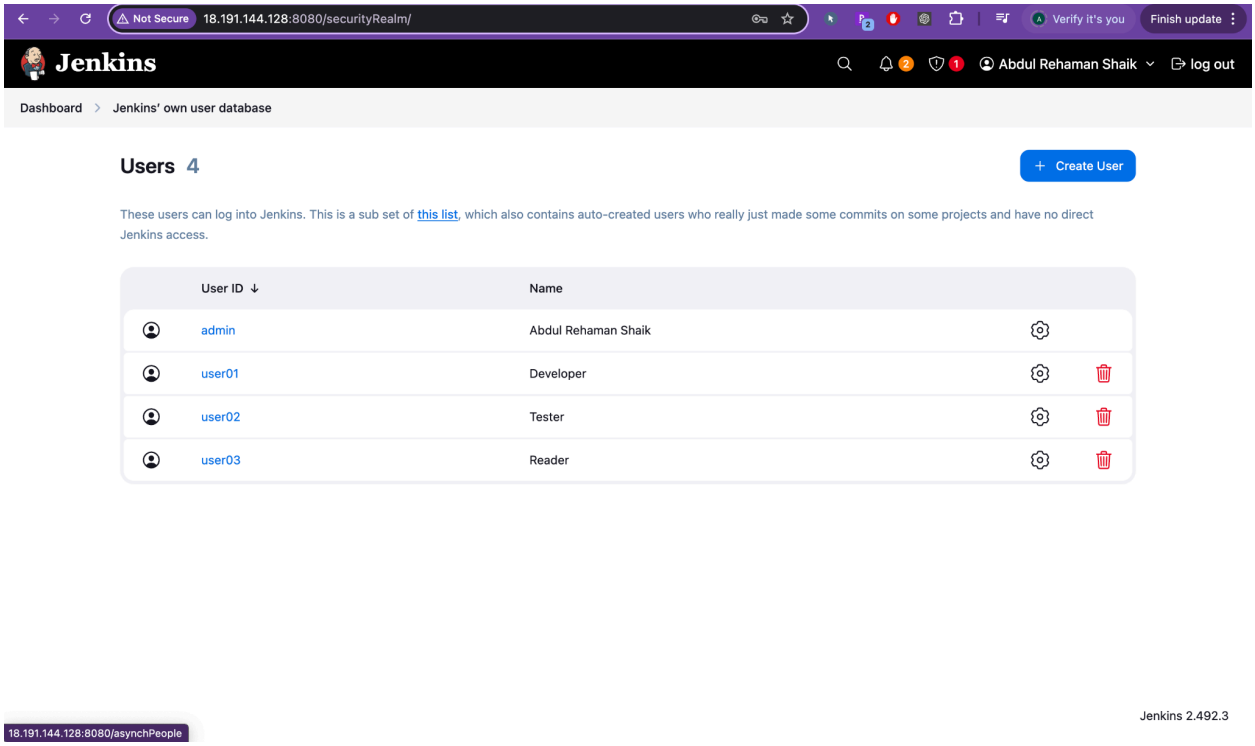
📊 Final Professional RBAC Setup View

Area	Setup
Global Roles	Admin, Developer
Item Roles	Devrole (Dev-*), Testrole (Test-*)
User Assignment	Based on user job responsibilities
Security Principle	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**" ✅)



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**	**developer**

admin [✔]

user01 [✔]

user02 [✔]

user03 [✔]

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	devrole	testrole
admin	[✔]	[✔]
user01	[✔]	
user02		[✔]
user03		

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

user02	developer	testrole	Can build/manage only `Test-*` jobs
user03	developer	None or View-only	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`

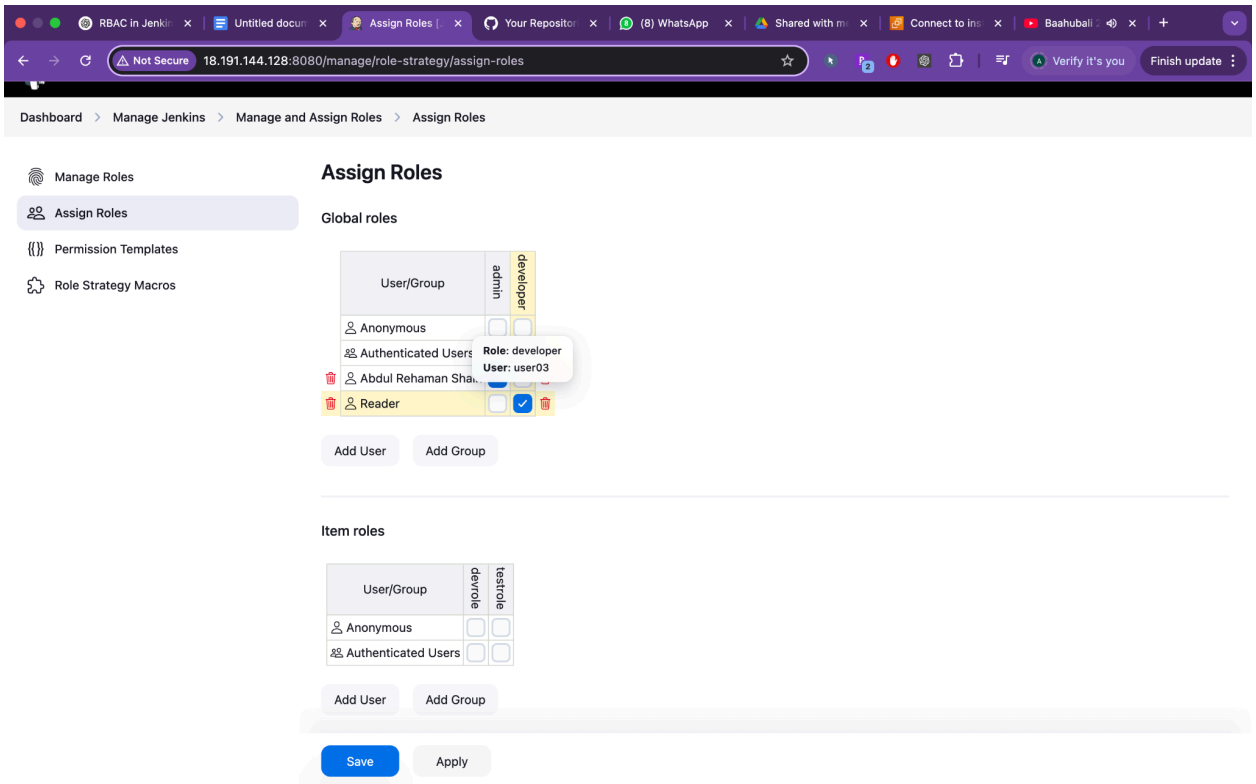
- That's it! No need to disturb existing users.

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**" ✍️)



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
`admin`	✅ admin
`user01`	✅ developer
`user02`	✅ developer
`user03`	✅ developer (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`	✅ 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

user03	reader (optional)	— or none

🧪 Step 3: Test Each User's Access

Login with each user and verify:

Username	Expected View
`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

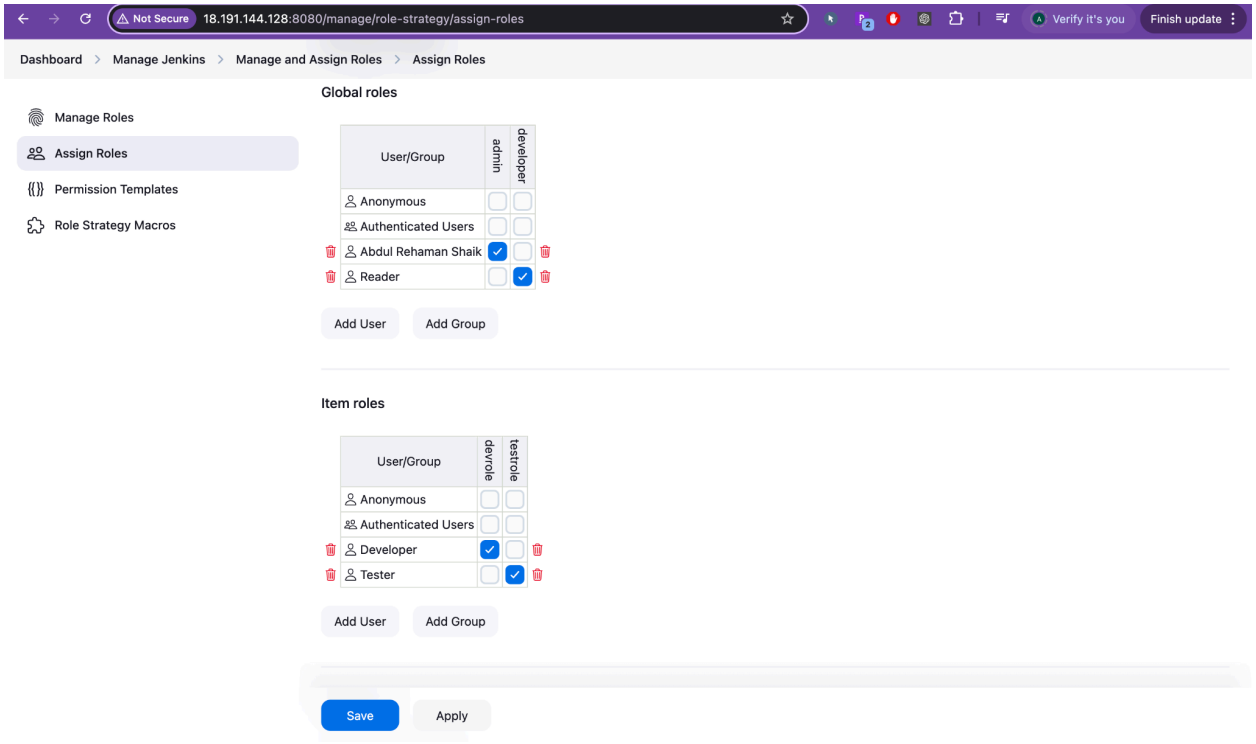
🛡️ Final Notes for Secure Jenkins

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



✅ 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
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	devrole	testrole
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

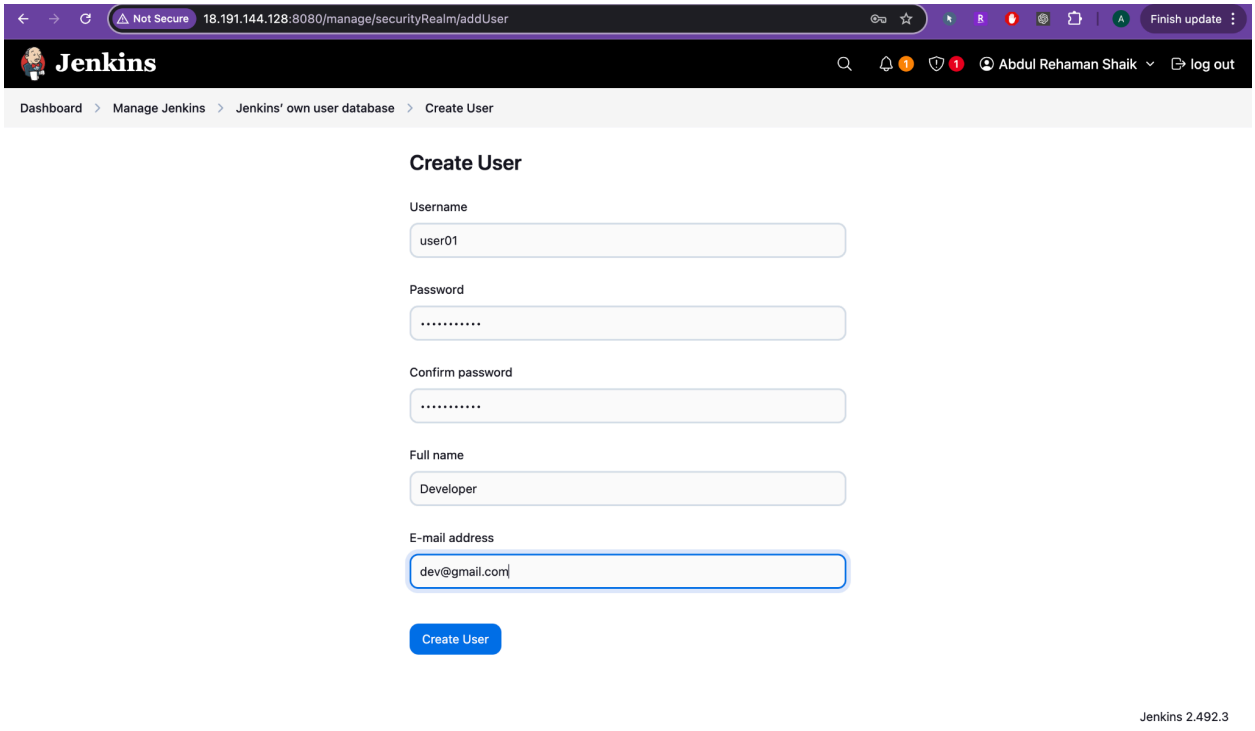
Step	Status
Created Users	✅ Done
Created Global Roles	✅ Done
Created Item Roles	✅ Done
Assigned Roles	✅ Done (almost — just finish user mapping)

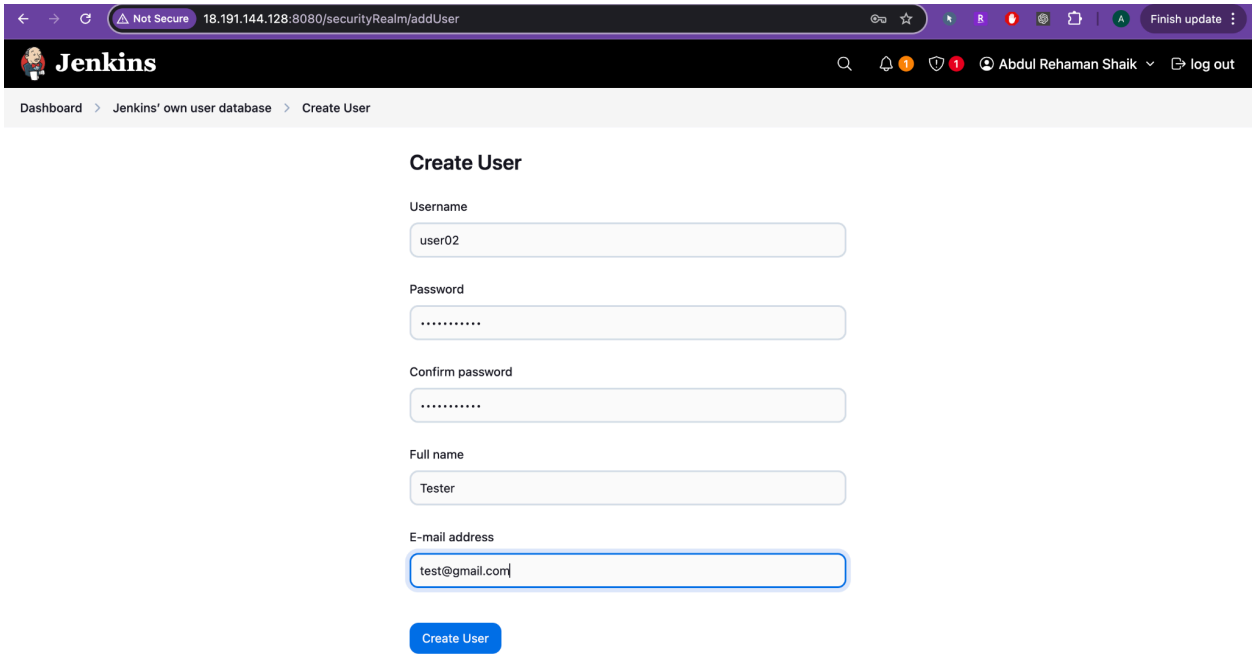
Testing Access ⏳ To Do (next)

You now have **enterprise-grade RBAC security** in Jenkins! 💼🔐

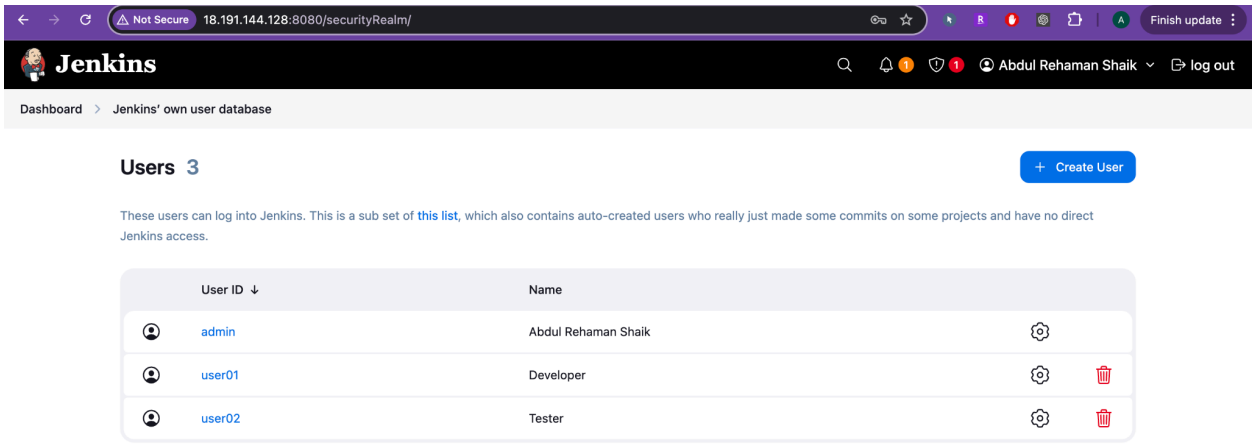
📥 Bonus Offer

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.

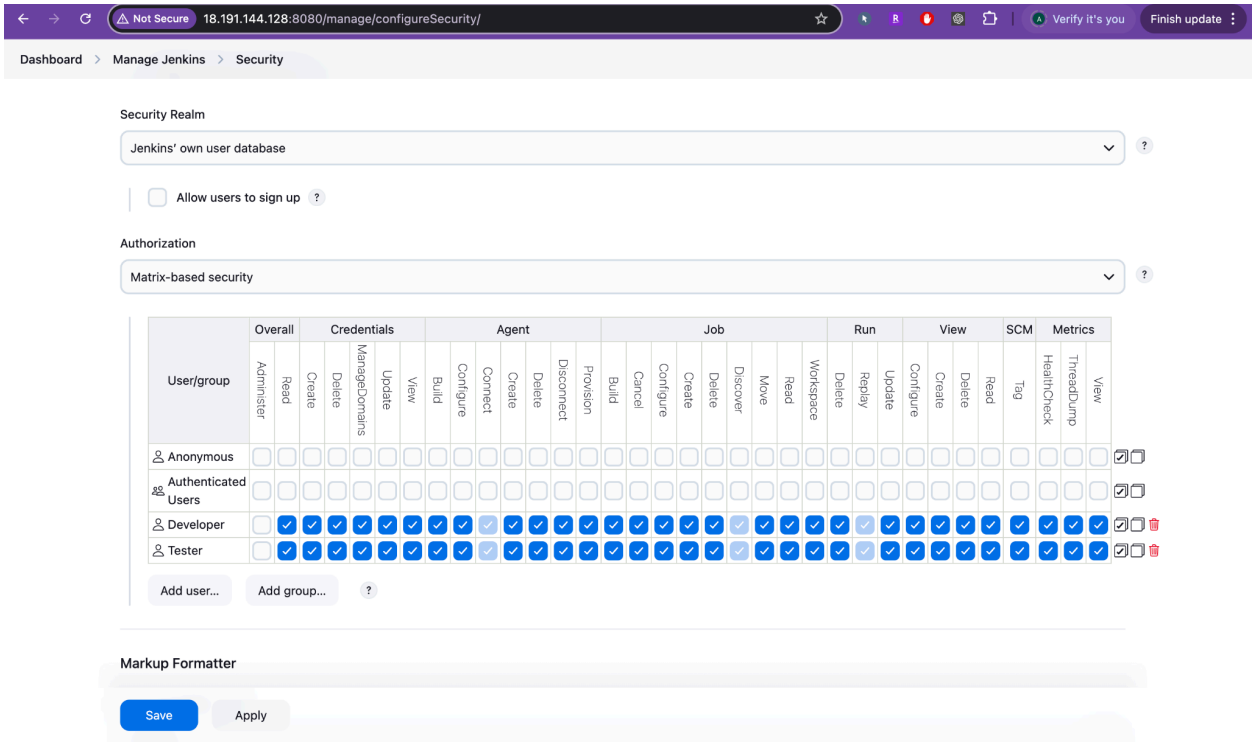


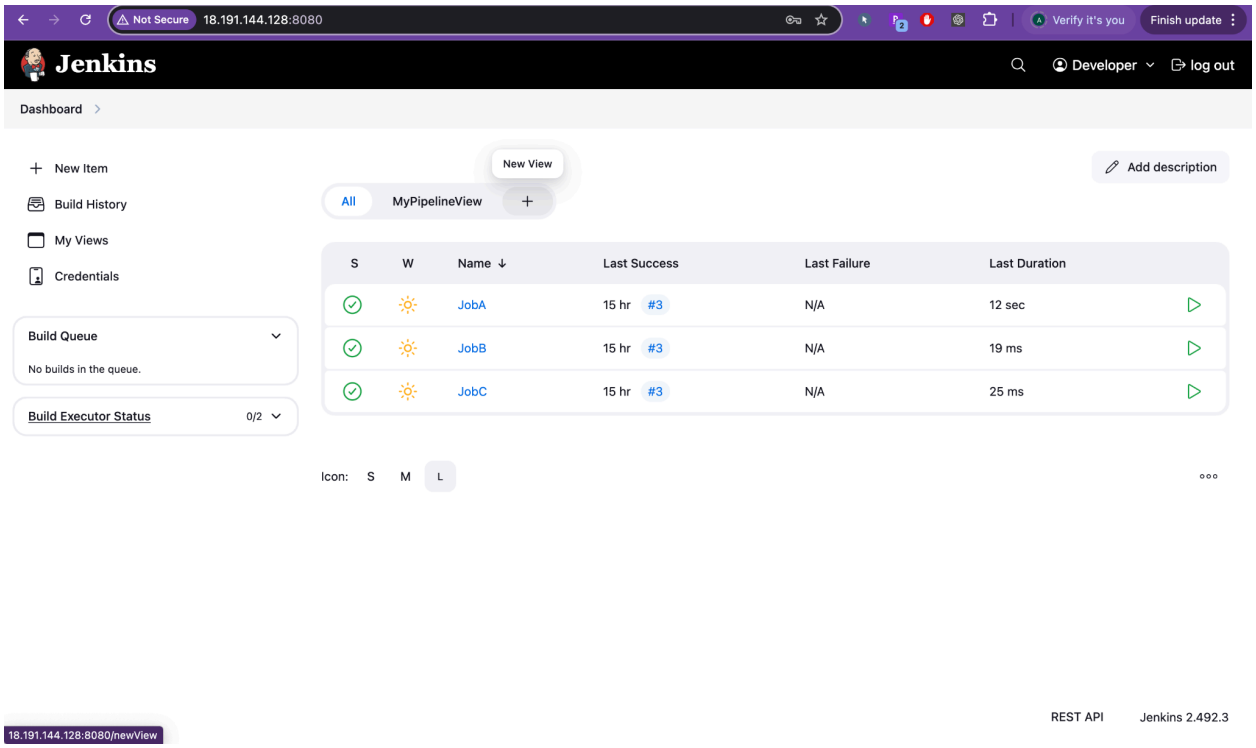


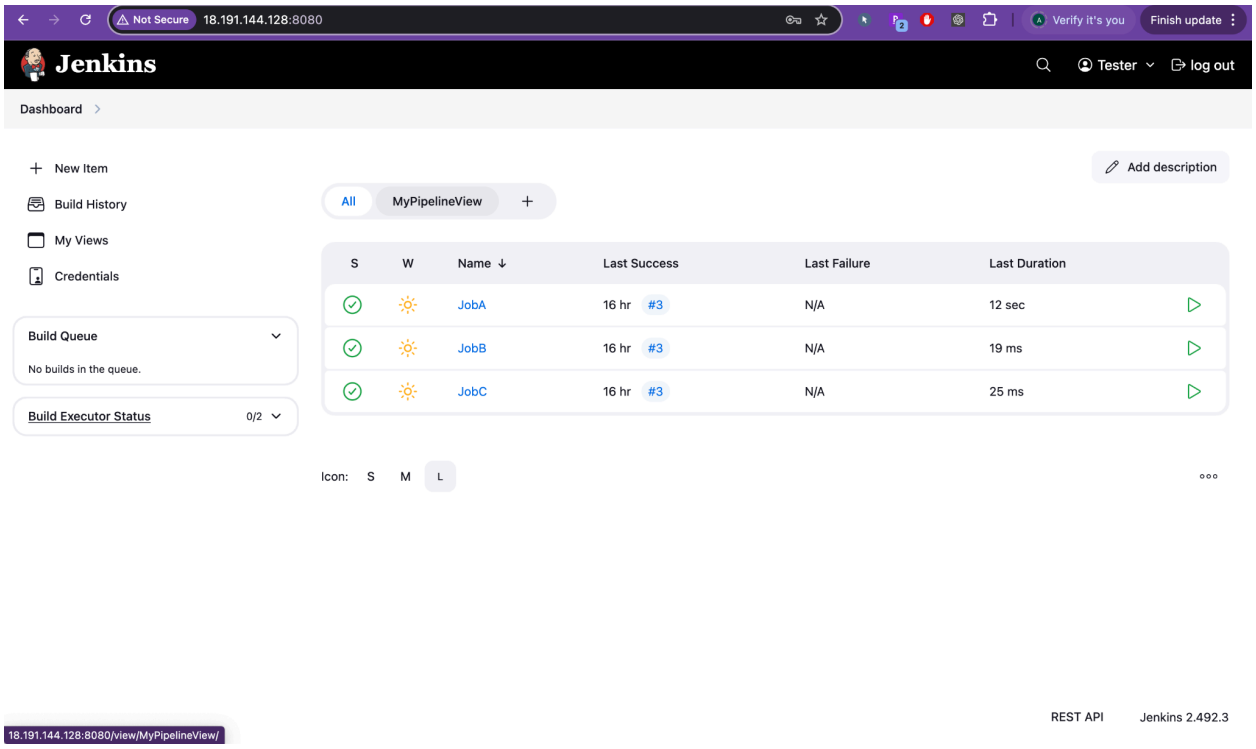
Jenkins 2.492.3

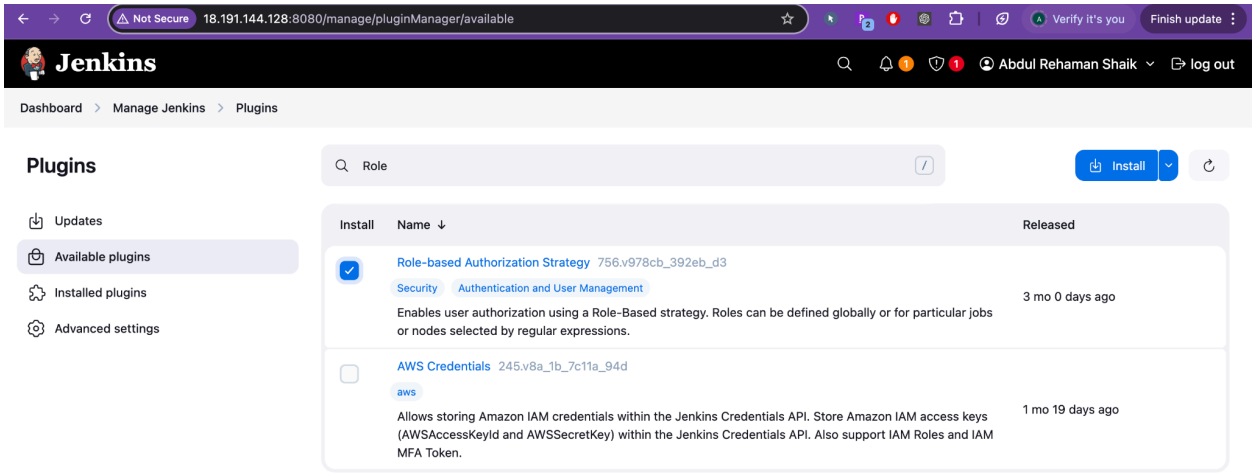


Jenkins 2.492.3

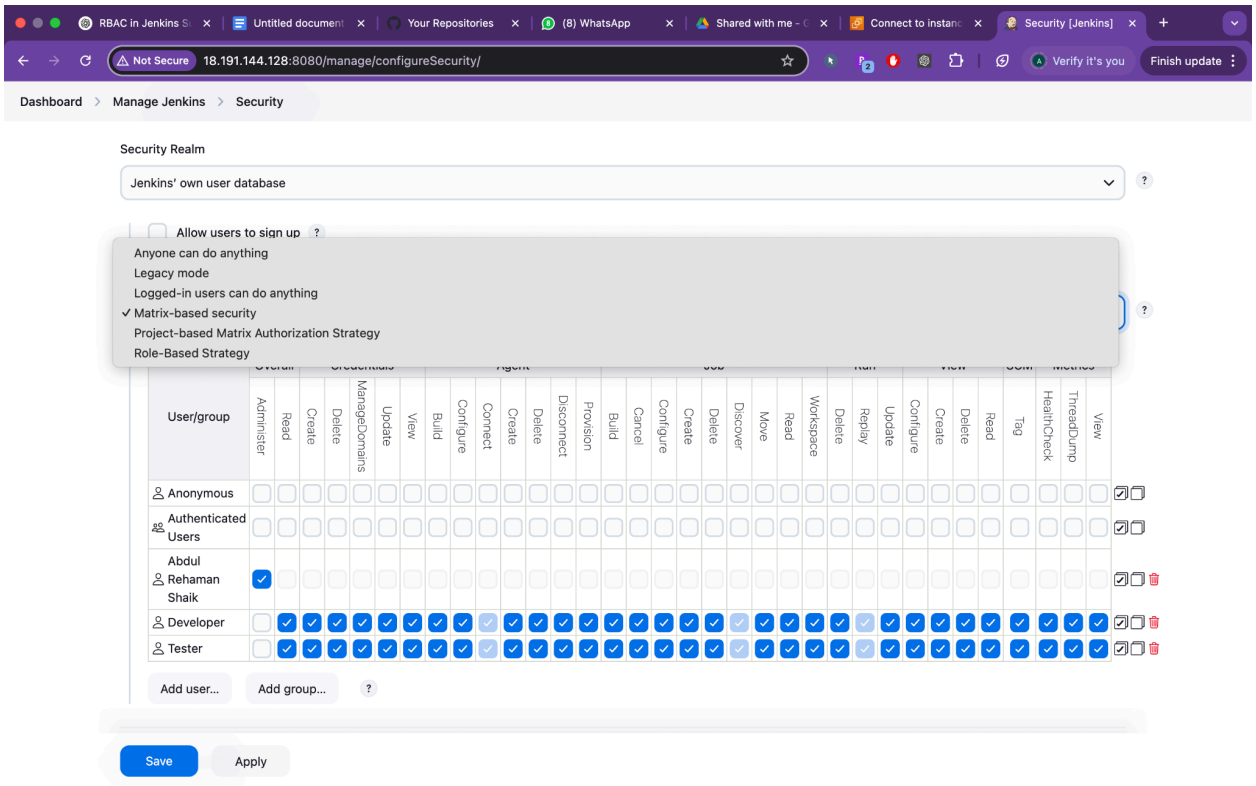

Not Secure 18.191.144.128:8080/manage/configureSecurity/
Verify it's you
Finish update
Dashboard > Manage Jenkins > Security
Security Realm
Jenkins' own user database
Allow users to sign up
Authorization
Matrix-based security
User/group
Overall
Credentials
Agent
Job
Run
View
SCM
Metrics
Administer
Read
Create
Delete
ManageDomains
Update
View
Build
Configure
Connect
Create
Delete
Disconnect
Provision
Build
Cancel
Configure
Create
Delete
Discover
Move
Read
Workspace
Delete
Replay
Update
Configure
Create
Delete
Read
Tag
HealthCheck
ThreadDump
View
Anonymous
Authenticated Users
Developer
Tester
Add user...
Add group...
Markup Formatter
Save
Apply

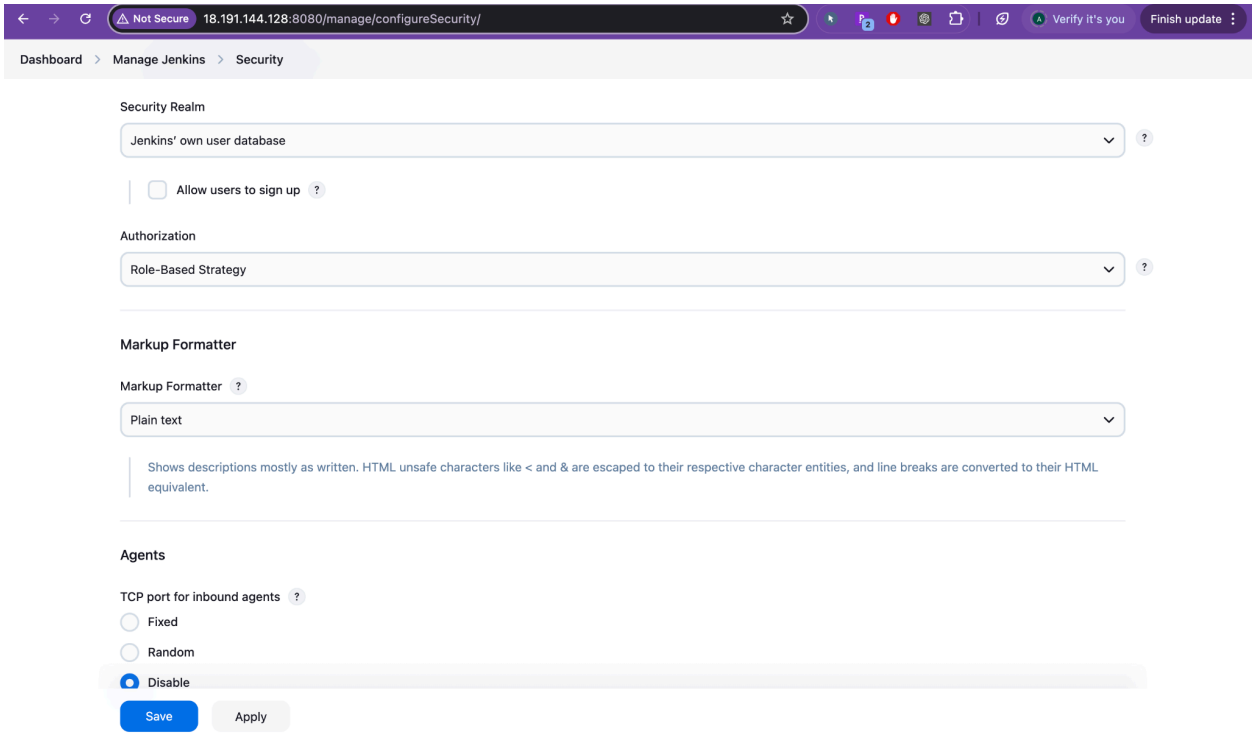

Not Secure 18.191.144.128:8080
Verify it's you
Finish update
Jenkins
Developer
log out
Dashboard
New Item
Build History
My Views
Credentials
Build Queue
No builds in the queue.
Build Executor Status 0/2
New View
All
MyPipelineView
Add description
S
W
Name
Last Success
Last Failure
Last Duration
JobA 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
REST API
Jenkins 2.492.3
18.191.144.128:8080/newView

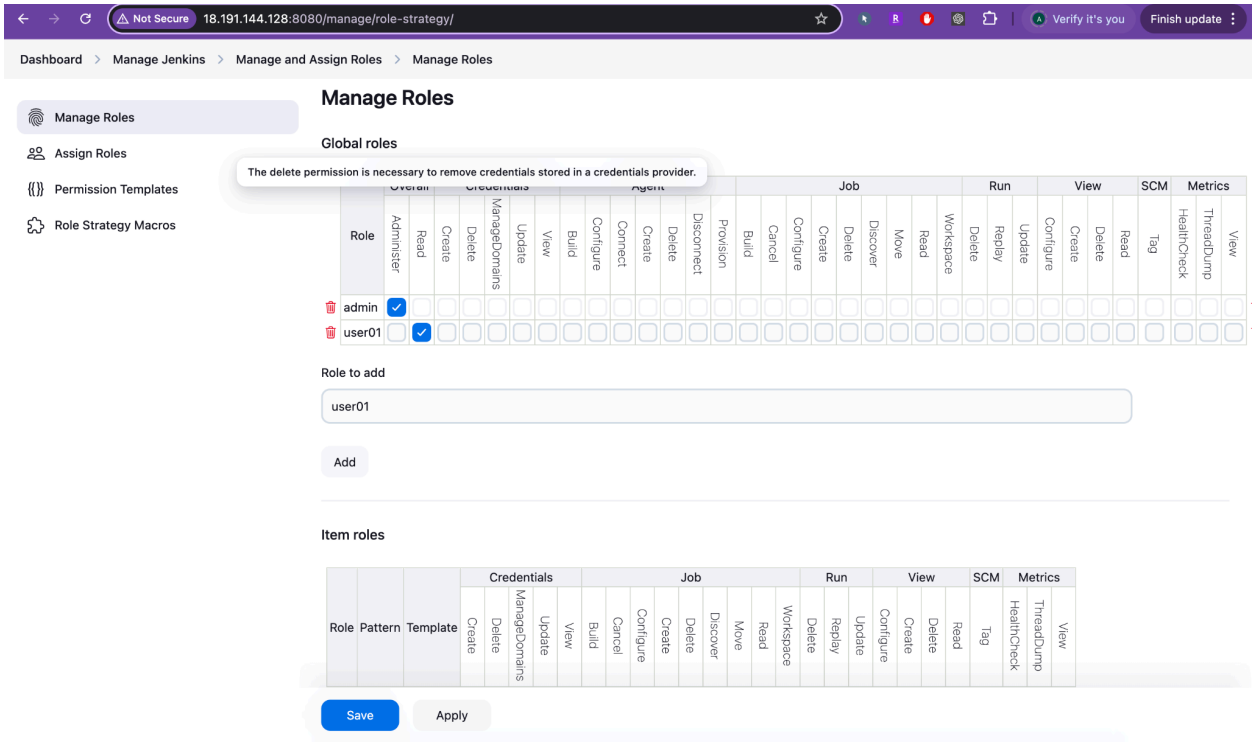

Not Secure
18.191.144.128:8080
Verify it's you
Finish update
Jenkins
Tester
log out
Dashboard
New Item
Build History
My Views
Credentials
Build Queue
No builds in the queue.
Build Executor Status
0/2
Add description
All
MyPipelineView
S
W
Name
Last Success
Last Failure
Last Duration
JobA
16 hr #3
N/A
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

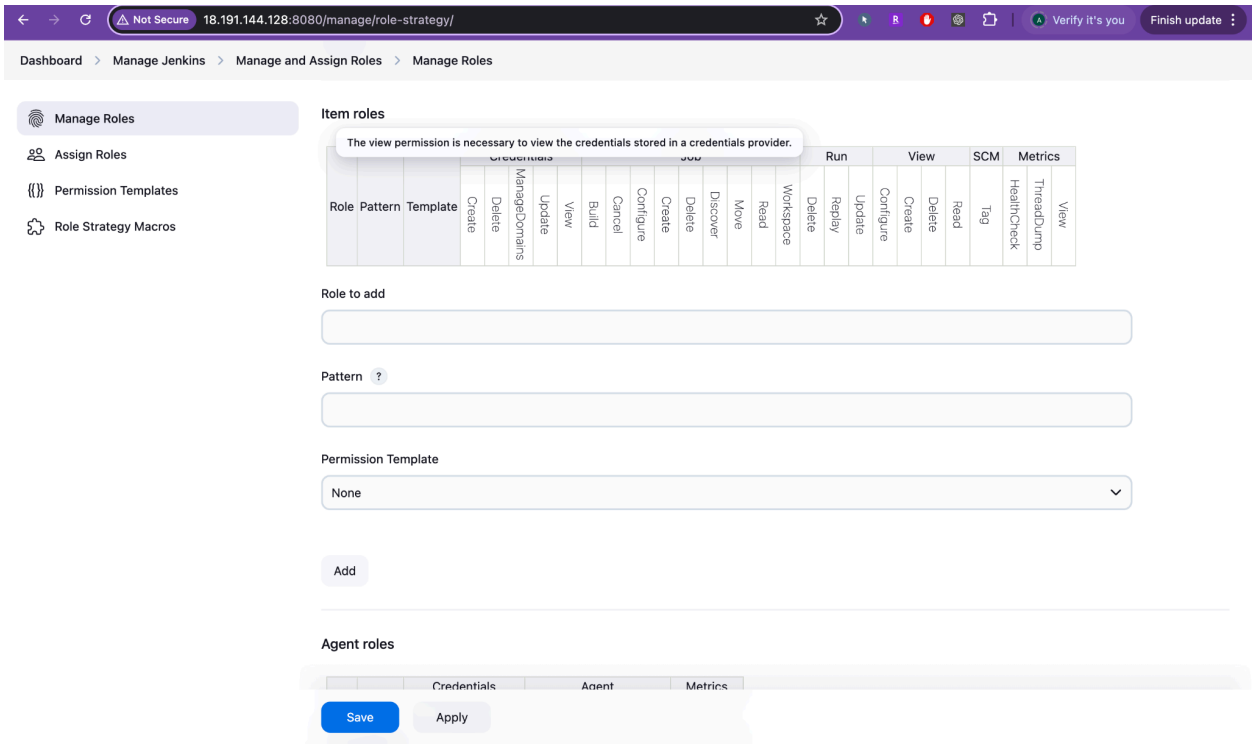


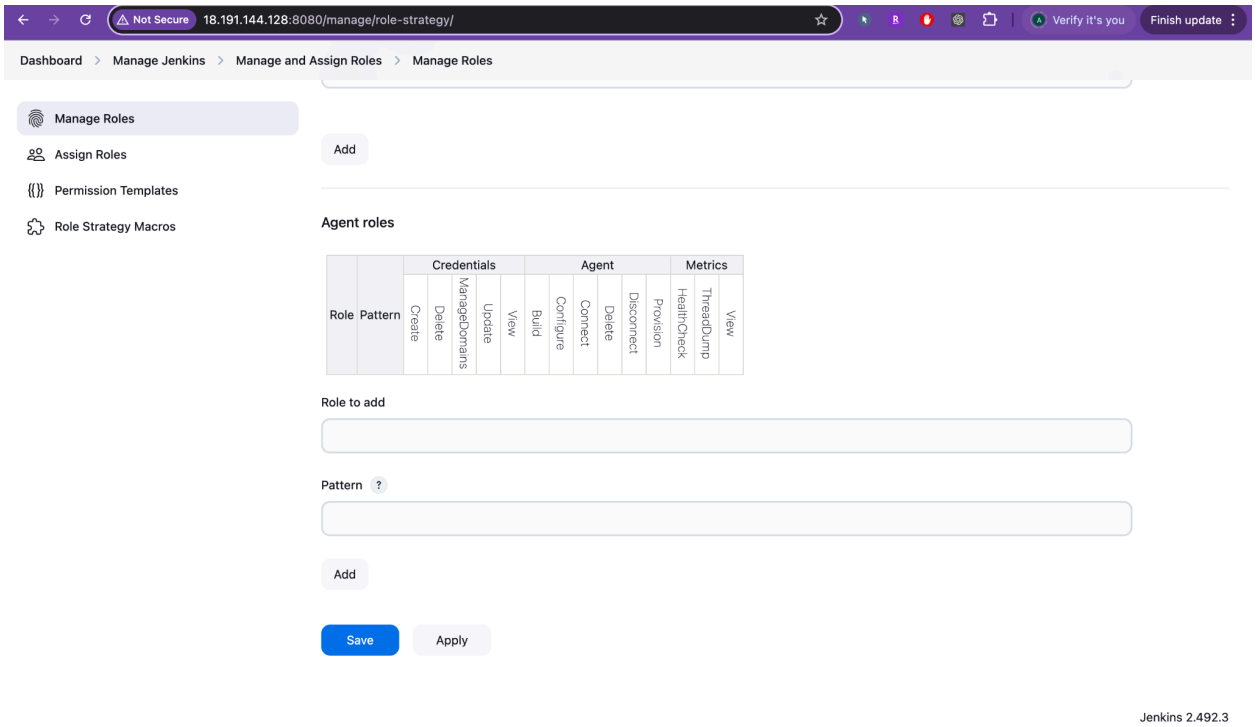
REST API Jenkins 2.492.3

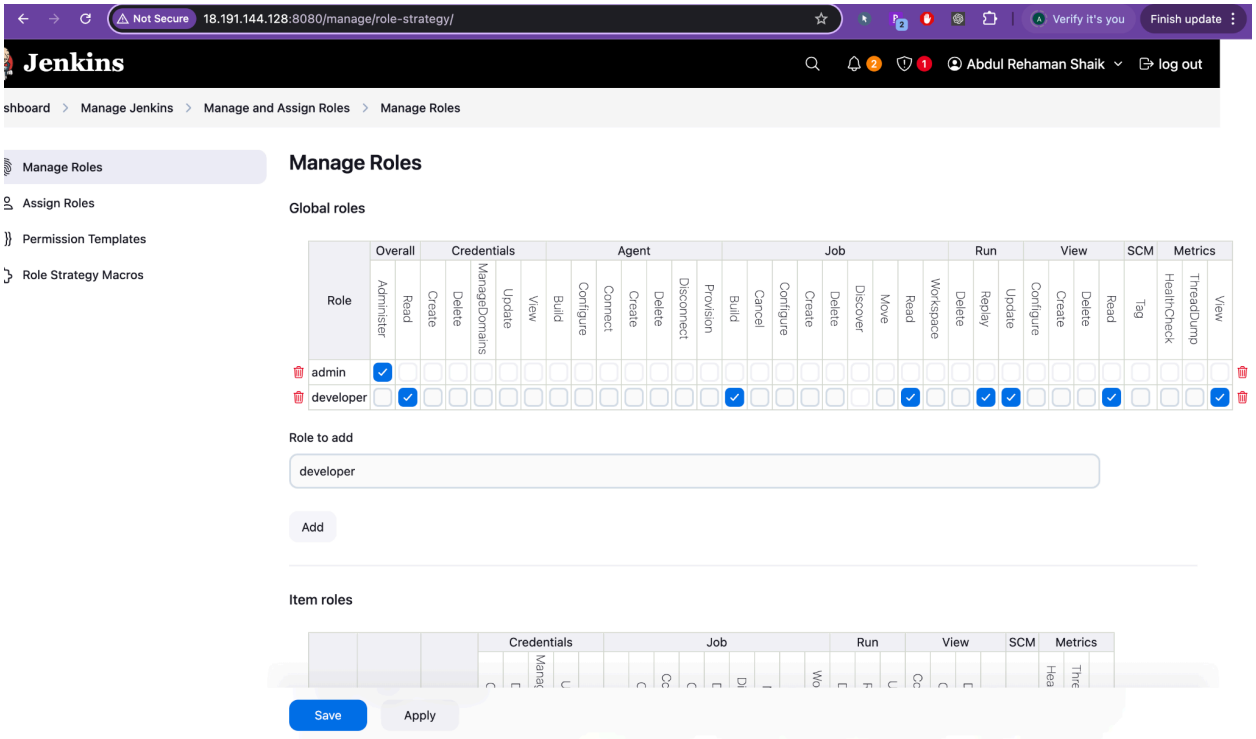


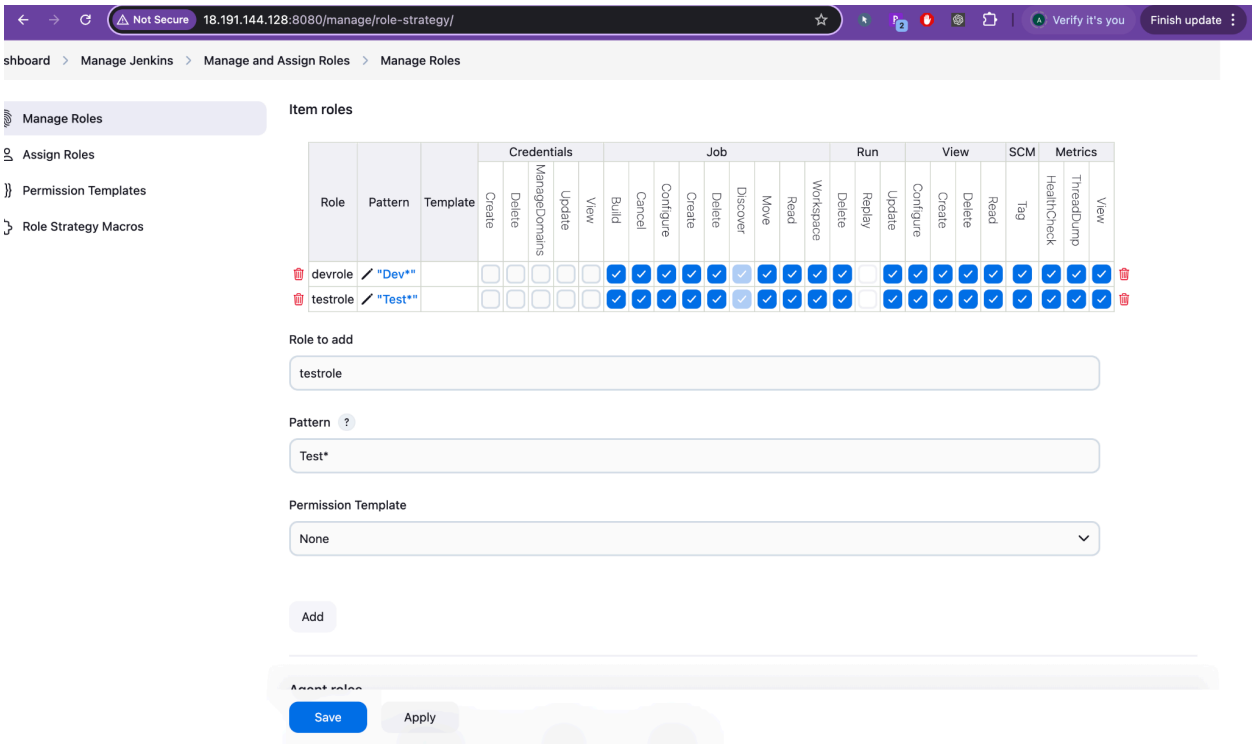

Not Secure
18.191.144.128:8080/manage/configureSecurity/
Verify it's you
Finish update
Dashboard
Manage Jenkins
Security
Security Realm
Jenkins' own user database
Allow users to sign up
Authorization
Role-Based Strategy
Markup Formatter
Markup Formatter
Plain text
Shows descriptions mostly as written. 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

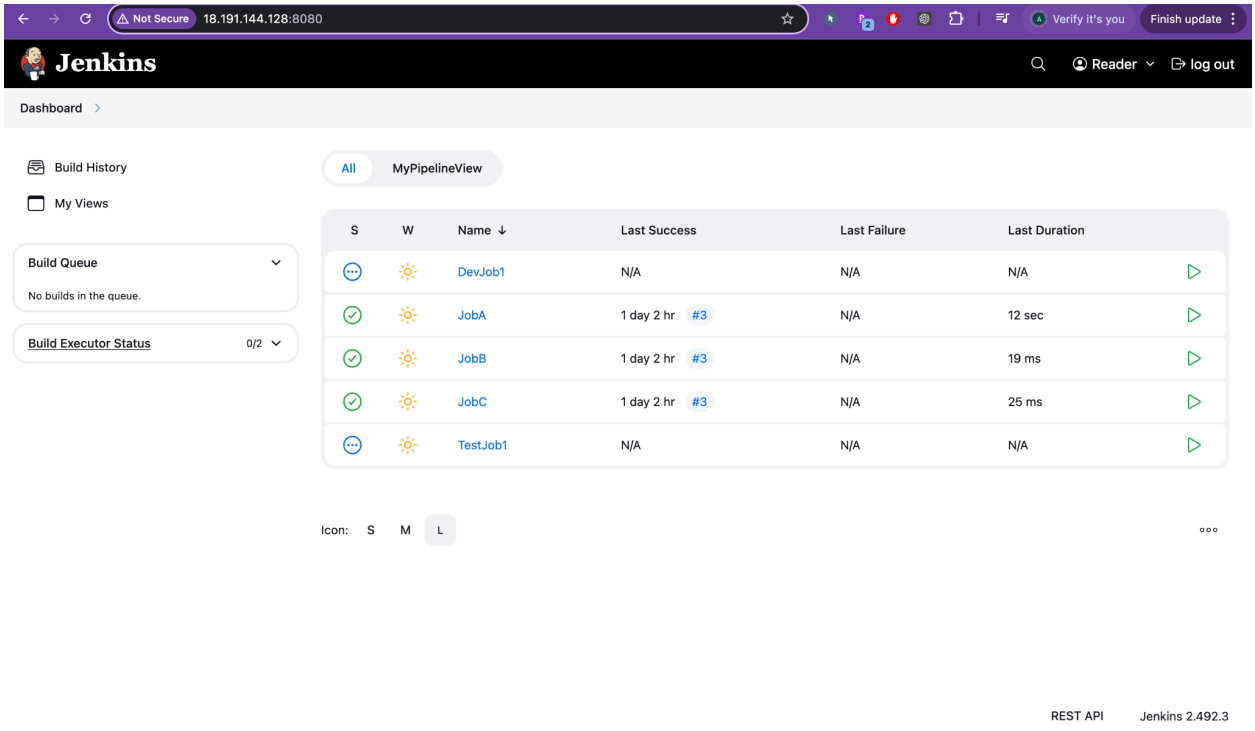

Not Secure 18.191.144.128:8080/manage/role-strategy/
Verify it's you
Finish update
Dashboard > Manage Jenkins > Manage and Assign Roles > Manage Roles
Manage Roles
Assign Roles
Permission Templates
Role Strategy Macros
Manage Roles
Global roles
The delete permission is necessary to remove credentials stored in a credentials provider.
Overall
Credentials
Agent
Job
Run
View
SCM
Metrics
Role
Administer
Read
Create
Delete
ManageDomains
Update
View
Build
Configure
Connect
Create
Delete
Disconnect
Provision
Build
Cancel
Configure
Create
Delete
Discover
Move
Read
Workspace
Delete
Replay
Update
Configure
Create
Delete
Read
Tag
HealthCheck
ThreadDump
View
admin
user01
Role to add
user01
Add
Item roles
Role
Pattern
Template
Credentials
Create
Delete
ManageDomains
Update
View
Job
Build
Cancel
Configure
Create
Delete
Discover
Move
Read
Workspace
Run
Delete
Replay
Update
View
Configure
Create
Delete
Read
SCM
Tag
Metrics
HealthCheck
ThreadDump
View
Save
Apply


Not Secure 18.191.144.128:8080/manage/role-strategy/
Verify it's you
Finish update
Dashboard > Manage Jenkins > Manage and Assign Roles > Manage Roles
Manage Roles
Assign Roles
Permission Templates
Role Strategy Macros
Item roles
The view permission is necessary to view the credentials stored in a credentials provider.
Role
Pattern
Template
Credentials
Create
Delete
ManageDomains
Update
View
Job
Build
Cancel
Configure
Create
Delete
Discover
Move
Read
Workspace
Run
Delete
Replay
Update
View
Configure
Create
Delete
Read
SCM
Tag
Metrics
HealthCheck
ThreadDump
View
Role to add
Pattern
Permission Template
None
Add
Agent roles
Credentials
Agent
Metrics
Save
Apply


Not Secure
18.191.144.128:8080/manage/role-strategy/
Verify it's you
Finish update
Dashboard
Manage Jenkins
Manage and Assign Roles
Manage Roles
Manage Roles
Assign Roles
Permission Templates
Role Strategy Macros
Add
Agent roles
Role
Pattern
Credentials
Create
Delete
ManageDomains
Update
View
Agent
Build
Configure
Connect
Delete
Disconnect
Provision
Metrics
HealthCheck
ThreadDump
View
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
Abdul Rehaman Shaik
log out
shboard > Manage Jenkins > Manage and Assign Roles > Manage Roles
Manage Roles
Assign Roles
Permission Templates
Role Strategy Macros
Manage Roles
Global roles
Role
Overall
Administer
Read
Credentials
Create
Delete
ManageDomains
Update
View
Agent
Build
Configure
Connect
Create
Delete
Disconnect
Provision
Job
Build
Cancel
Configure
Create
Delete
Discover
Move
Read
Workspace
Run
Delete
Replay
Update
View
Configure
Create
Delete
Read
SCM
Tag
Metrics
HealthCheck
ThreadDump
View
admin
developer
Role to add
developer
Add
Item roles
Credentials
Job
Run
View
SCM
Metrics
Save
Apply


Not Secure 18.191.144.128:8080/manage/role-strategy/
Verify it's you
Finish update
shboard > Manage Jenkins > Manage and Assign Roles > Manage Roles
Manage Roles
Assign Roles
Permission Templates
Role Strategy Macros
Item roles
Role
Pattern
Template
Credentials
Create
Delete
ManageDomains
Update
View
Job
Build
Cancel
Configure
Create
Delete
Discover
Move
Read
Workspace
Run
Delete
Replay
Update
View
Configure
Create
Delete
Read
SCM
Tag
Metrics
HealthCheck
ThreadDump
View
devrole
"Dev*"
testrole
"Test*"
Role to add
testrole
Pattern
Test*
Permission Template
None
Add
Save
Apply


Not Secure
18.191.144.128:8080
Verify it's you
Finish update
Jenkins
Reader
log out
Dashboard
Build History
My Views
Build Queue
No builds in the queue.
Build Executor Status
0/2
All
MyPipelineView
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
REST API
Jenkins 2.492.3