Role-Based Authorization in Jenkins with Team-Based Job Management

This documentation outlines the configuration of a Jenkins setup for an organization with three teams: **Developer**, **DevOps**, and **Testing**. Each team has dedicated users and jobs, with access restricted to their specific roles.

---

1. Organizational Structure

Teams and Users

1. **Developer Team**:
    - **Users**: `developer-1`, `developer-2`
    - **Jobs**: `dev-1`, `dev-2`, `dev-3`
2. **Testing Team**:
    - **Users**: `testing-1`, `testing-2`
    - **Jobs**: `test-1`, `test-2`, `test-3`
3. **DevOps Team**:
    - **Users**: `devops-1`, `devops-2`
    - **Jobs**: `devops-1`, `devops-2`, `devops-3`
4. **Administration**:
    - **User**: `admin-1`
    - **Role**: Full administrative access to Jenkins.

---

2. Job Configuration

Job Naming and Views

1. **Developer Jobs**:
    - Job Names: `dev-1`, `dev-2`, `dev-3`
    - Visible in the **Developer View**.
2. **Testing Jobs**:
    - Job Names: `test-1`, `test-2`, `test-3`
    - Visible in the **Testing View**.
3. **DevOps Jobs**:
    - Job Names: `devops-1`, `devops-2`, `devops-3`
    - Visible in the **DevOps View**.

---

3. Access Control

Access Permissions

Team/Role	Visible Jobs	Permissions
Developer	`dev-*`	Can view, build, configure, and see workspace for `dev-*` jobs only.
Tester	`test-*`	Can view, build, configure, and see workspace for `test-*` jobs only.
DevOps	`devops-*`	Can view, build, configure, and see workspace for `devops-*` jobs only.
Administrator	All jobs	Full access to all jobs and configurations.

---

4. Step-by-Step Setup

1. Create Jobs

1. Navigate to **New Item** in Jenkins.
2. Create jobs with the names:
   - `dev-1`, `dev-2`, `dev-3` for Developer jobs.
   - `test-1`, `test-2`, `test-3` for Testing jobs.
   - `devops-1`, `devops-2`, `devops-3` for DevOps jobs.
3. Add the build script mentioned above to each job.

2. Create Views (Optional)

1. Go to **Manage Jenkins → Views**.
2. Create views for each team: **Developer View**, **Testing View**, **DevOps View**.
3. Assign jobs to views based on their prefixes (`dev-*`, `test-*`, `devops-*`).

3. Install and Configure Role-Based Authorization Strategy

1. **Install Plugin**:
   - Go to **Manage Jenkins → Manage Plugins**.
   - Search for **Role-Based Authorization Strategy** and install it.
2. **Enable Plugin**:
   - Go to **Manage Jenkins → Configure Global Security**.
   - Select **Role-Based Authorization Strategy** under the Authorization section and save.
3. **Define Roles**:
   - Navigate to **Manage Jenkins → Manage and Assign Roles → Manage Roles**.
   - Create roles and permissions as described below:

Roles and Permissions

Role	Permissions
Developer	View, Build, Configure, See Workspace for `dev-*`.
Tester	View, Build, Configure, See Workspace for `test-*`.
DevOps	View, Build, Configure, See Workspace for `devops-*`.
Admin	Full permissions.

4. **Assign Roles to Users**:
    - Navigate to **Manage Jenkins → Manage and Assign Roles → Assign Roles**.
    - Assign users to their respective roles:

User	Assigned Role
`developer-1`	Developer
`developer-2`	Developer
`testing-1`	Tester
`testing-2`	Tester
`devops-1`	DevOps
`devops-2`	DevOps
`admin-1`	Admin

---

5. Summary

This setup provides:

- **Isolated access** for Developers, Testers, and DevOps teams based on their roles.
- **Granular permissions** to ensure each team interacts only with its designated jobs.
- A streamlined **job management process** with team-specific views.

By adhering to the **least privilege principle** and **role-based access**, this configuration ensures an efficient and secure CI/CD pipeline tailored to organizational requirements.