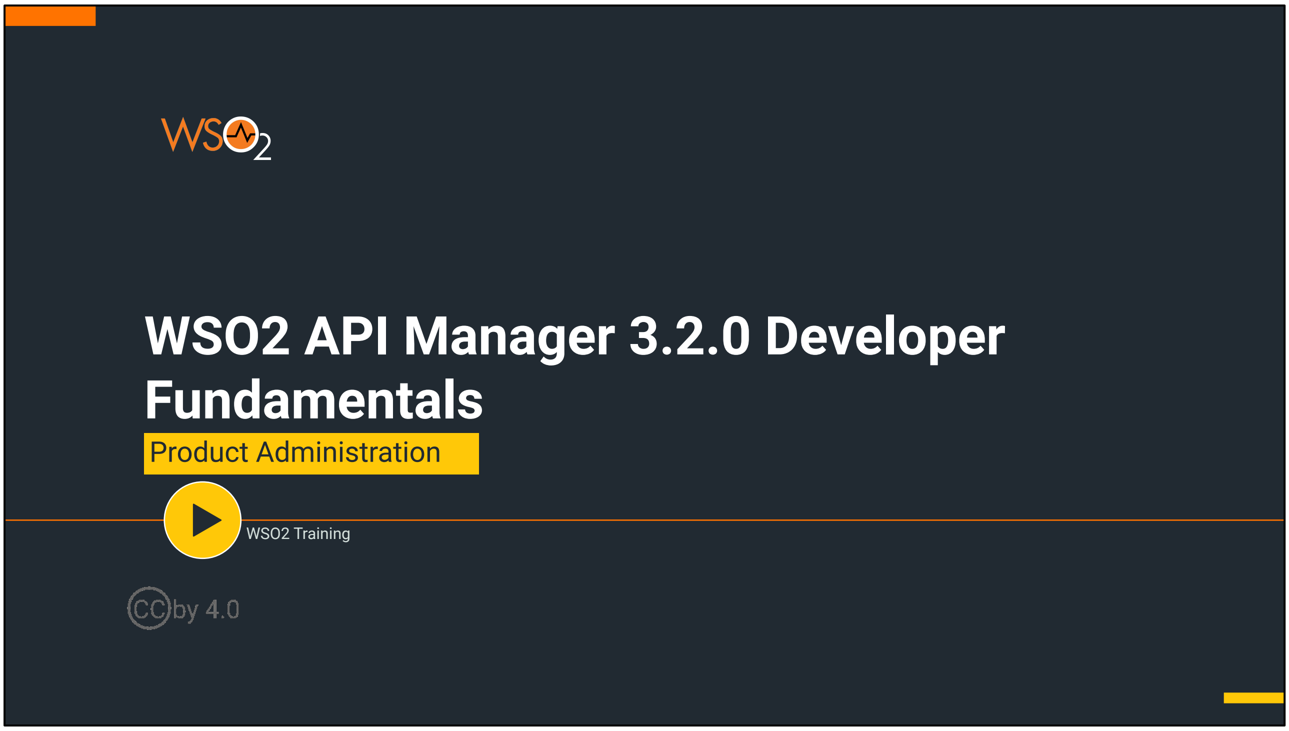
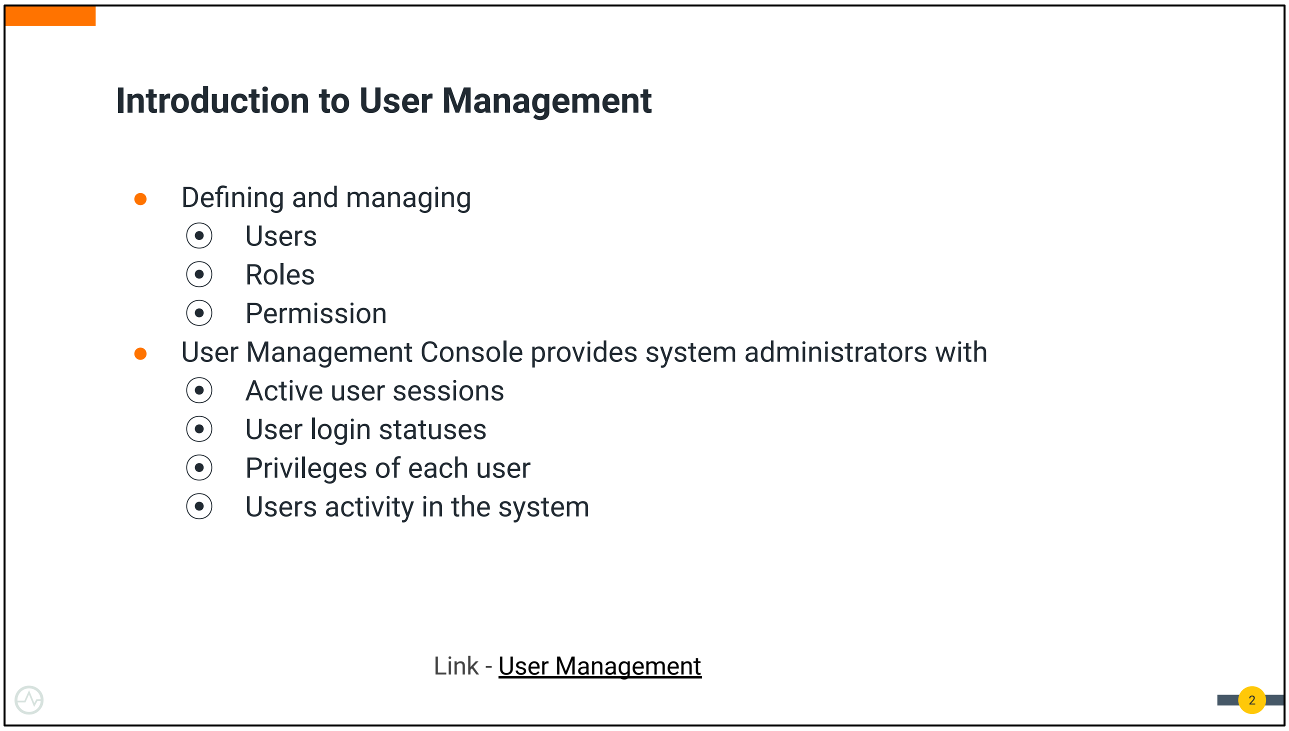


WSO2 API Manager 3.2.0 Developer Fundamentals

Product Administration

WSO2 Training

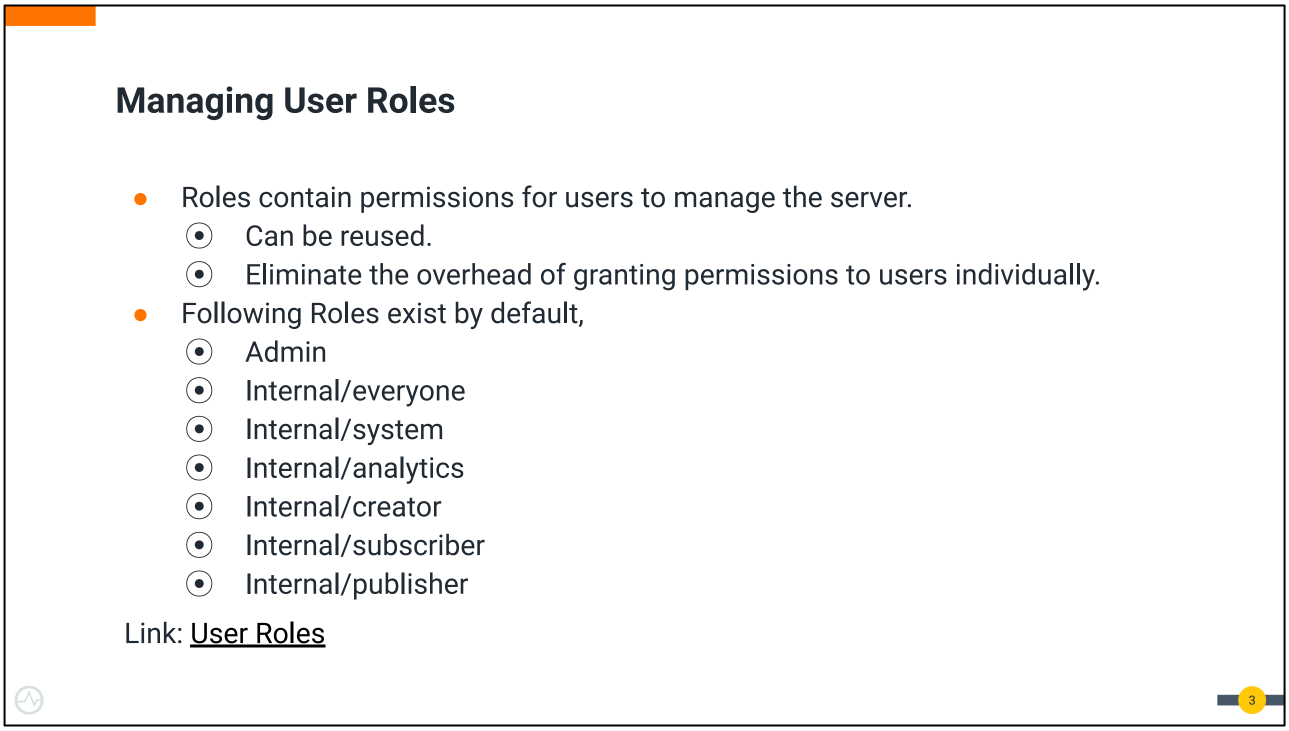


Introduction to User Management

- Defining and managing
  - Users
  - Roles
  - Permission
- User Management Console provides system administrators with
  - Active user sessions
  - User login statuses
  - Privileges of each user
  - Users activity in the system

Link - User Management

A **permission** is a delegation of authority or a right to perform an action on a system



Managing User Roles

- Roles contain permissions for users to manage the server.
  - Can be reused.
  - Eliminate the overhead of granting permissions to users individually.
- Following Roles exist by default,
  - Admin
  - Internal/everyone
  - Internal/system
  - Internal/analytics
  - Internal/creator
  - Internal/subscriber
  - Internal/publisher

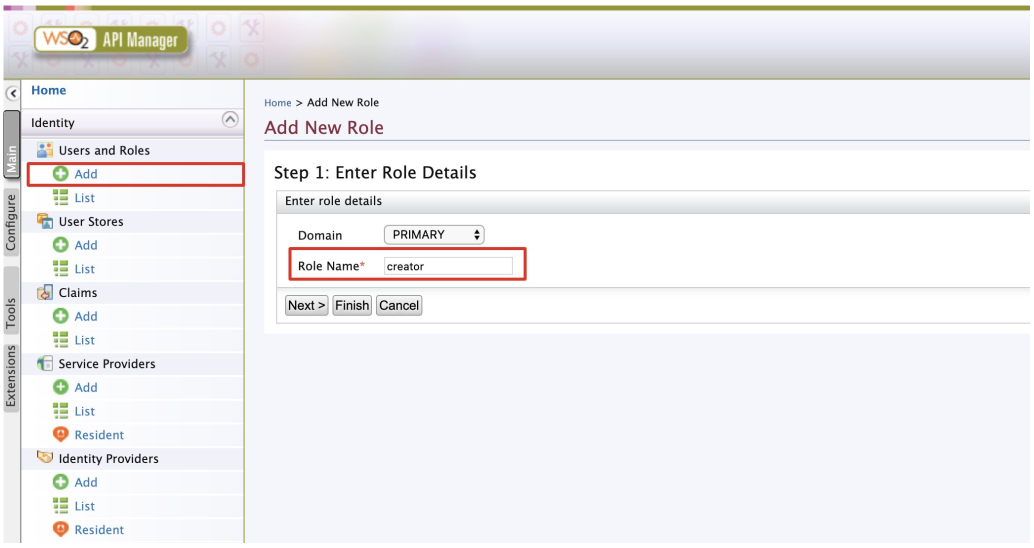
Link: User Roles

3

- **admin** - Provides full access to all features and controls. By default, the admin user is assigned to both the admin and the Internal/everyone roles.
- **Internal/everyone** - This is a predefined role that is used to group all the users (across the user stores) together. When you create a new user, automatically the user belongs to the Internal/everyone role. It does not include any permissions. This role can be used to identify all logged in users.
- **Internal/system** - This is another pre defined role which does not include any permissions. Unlike the Internal/everyone role, this role is **not assigned** to a user by default.
- **Internal/analytics** - This role can be assigned to users who do not have the publisher or subscriber roles assigned but need permission to view the analytics dashboards.
- **Internal/creator:** A creator is typically a person in a technical role who understands the technical aspects of the API (interfaces, documentation, versions etc.) and uses the **API publisher** to provision APIs into the Developer Portal. The creator uses the Developer Portal to consult ratings and feedback provided by API users. Creator can add APIs to the Developer Portal but cannot manage their lifecycle. Governance permission gives a creator permission to govern, manage and configure the API artifacts.
- **Internal/publisher:** A person in a managerial role and overlooks a set of APIs

- across the enterprise and controls the API lifecycle, subscriptions and monetization aspects. The publisher is also interested in usage patterns for APIs and has access to all API statistics.
- **Internal/subscriber:** A user or an application developer who searches the **Developer Portal** to discover APIs and use them. S/he reads the documentation and forums, ratings/comments on the APIs, subscribes to APIs, obtains access tokens and invokes the APIs.

Create a New User Role



Add New Role

Step 1: Enter Role Details

Enter role details

Domain: PRIMARY

Role Name*: creator

Log into the API Manager carbon console (https://localhost:9443/carbon) as admin user. (username: admin, password: admin)

Go to,

Users and Roles -> Add -> Add new Role

Provide the name of the role and click finish.

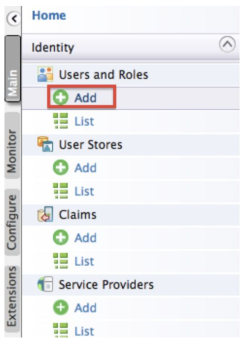
Managing Role Permissions

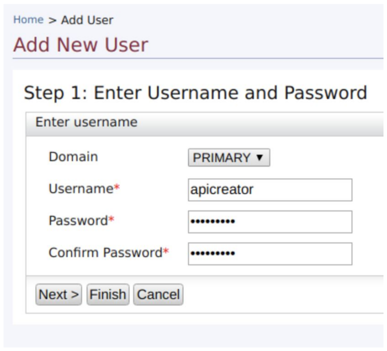
- Permissions can be granted to a role at two levels:
  - **Super Tenant level**: A role with super tenant permissions is used for managing all the tenants in the system and also for managing the key features in the system, which are applicable to all the tenants.
  - **Tenant level**: A role with tenant level permissions is only applicable to individual tenant spaces.

Link : Role Permissions

Managing Users

- Adding User
  - Log in to the Management Console (https://<hostname>:9443/carbon) and click **Add** under **Users and Roles** in the **Main** menu.



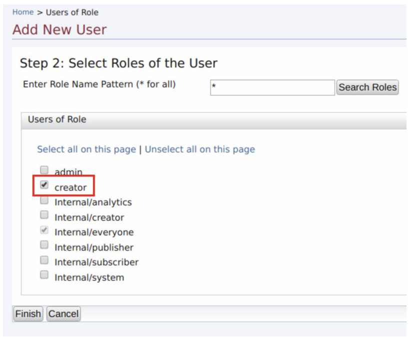

Home > Add User

Add New User

Step 1: Enter Username and Password

Enter username

Domain	PRIMARY ▾
Username*	apicreator
Password*	•••••••••
Confirm Password*	•••••••••


Home > Users of Role

Add New User

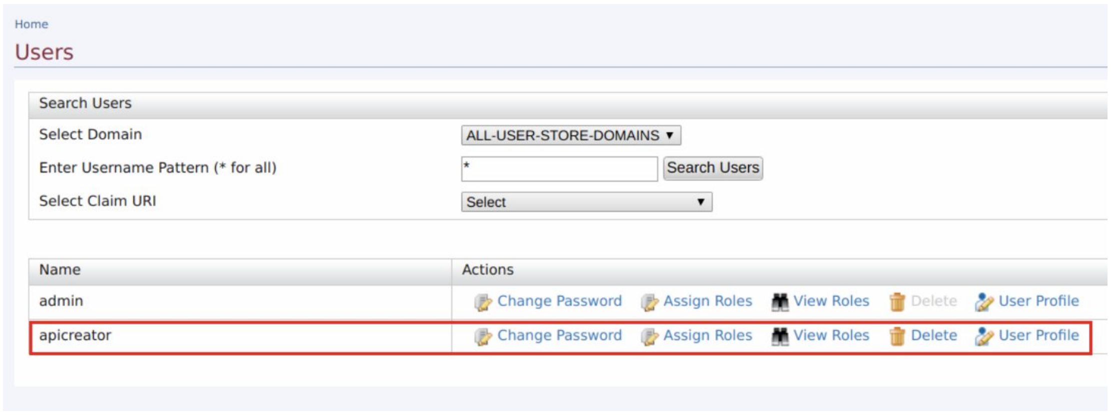
Step 2: Select Roles of the User

Enter Role Name Pattern (* for all) *

Users of Role

- [ ] admin
- [✓] creator
- [ ] Internal/analytics
- [ ] Internal/creator
- [✓] Internal/everyone
- [ ] Internal/publisher
- [ ] Internal/subscriber
- [ ] Internal/system

Managing Users


Home

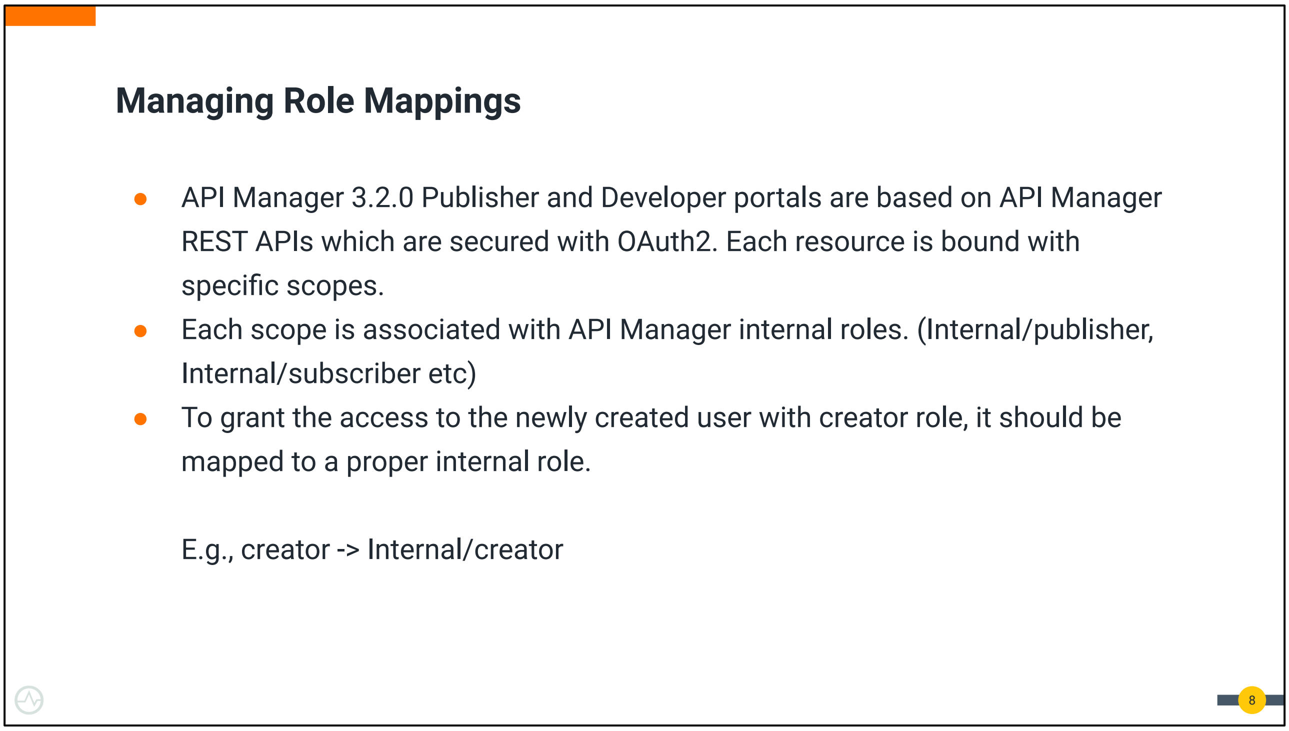
Users

Search Users

Select Domain	ALL-USER-STORE-DOMAINS
Enter Username Pattern (* for all)	*
Select Claim URI	Select

Search Users

Name	Actions
admin	Change Password · Assign Roles · View Roles · Delete · User Profile
apicreator	Change Password · Assign Roles · View Roles · Delete · User Profile



Managing Role Mappings

- API Manager 3.2.0 Publisher and Developer portals are based on API Manager REST APIs which are secured with OAuth2. Each resource is bound with specific scopes.
- Each scope is associated with API Manager internal roles. (Internal/publisher, Internal/subscriber etc)
- To grant the access to the newly created user with creator role, it should be mapped to a proper internal role.

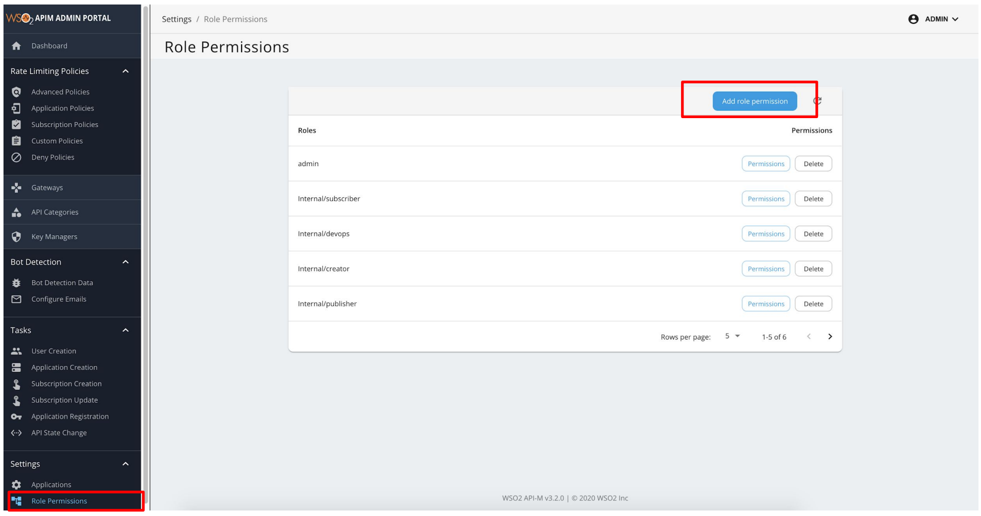
  E.g., creator -> Internal/creator

Scope: Scopes are Role based access control mechanism which can be enforced to API Resources.

If a scope is associated with a resource, the access token used to invoke the resource must have the required scope.

Managing Role Mappings

- Log in to the API Manager Admin Portal (https://<hostname>:9443/admin) and go to Settings -> Role Permissions.

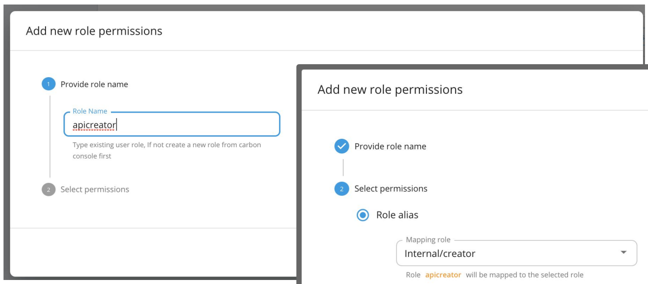


Role Permissions

Roles	Permissions
admin	Permissions Delete
Internal/subscriber	Permissions Delete
Internal/devops	Permissions Delete
Internal/creator	Permissions Delete
Internal/publisher	Permissions Delete

Managing Role Mappings

- Click on Add Role Permission.



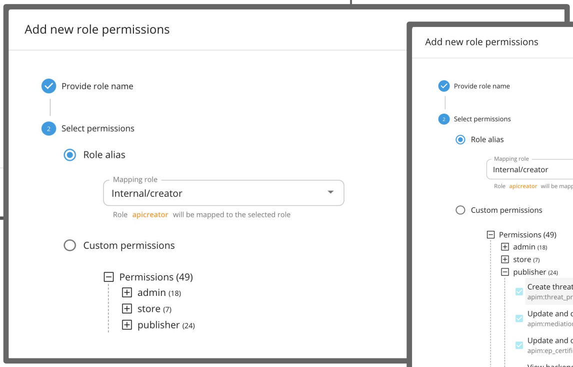
Add new role permissions

1 Provide role name

Role Name: apicreator

Type existing user role, If not create a new role from carbon console first

2 Select permissions



Add new role permissions

✓ Provide role name

2 Select permissions

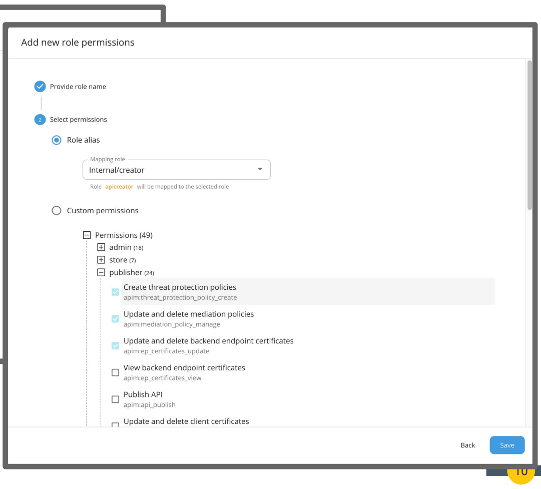
(●) Role alias

Mapping role: Internal/creator

Role apicreator will be mapped to the selected role

( ) Custom permissions

- Permissions (49)
  - admin (18)
  - store (7)
  - publisher (24)



Add new role permissions

✓ Provide role name

2 Select permissions

(●) Role alias

Mapping role: Internal/creator

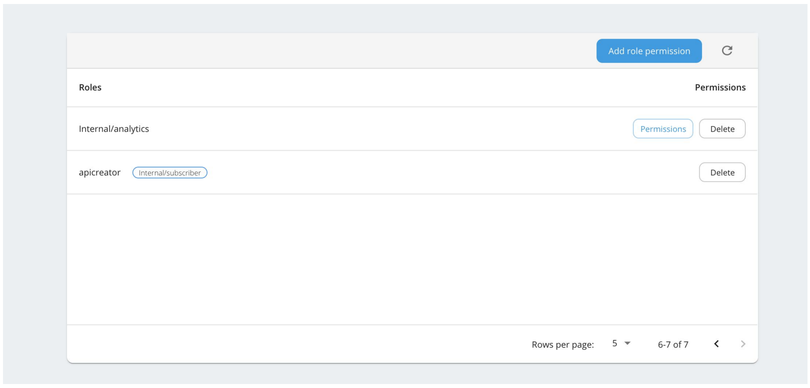
Role apicreator will be mapped to the selected role

( ) Custom permissions

- Permissions (49)
  - admin (18)
  - store (7)
  - publisher (24)
    - ✓ Create threat protection policies — apim:threat_protection_policy_create
    - ✓ Update and delete mediation policies — apim:mediation_policy_manage
    - ✓ Update and delete backend endpoint certificates — apim:ep_certificates_update
    - ☐ View backend endpoint certificates — apim:ep_certificates_view
    - ☐ Publish API — apim:api_publish
    - ☐ Update and delete client certificates

Managing Scope Mappings

- Role Mapping affects all the scopes that the Original Role is associated with.
- If you only need to selectively add the role to specific scopes, Scope Mapping can be used.



Roles	Permissions
Internal/analytics	Permissions Delete
apicreator Internal/subscriber	Delete

Introduction to User Stores

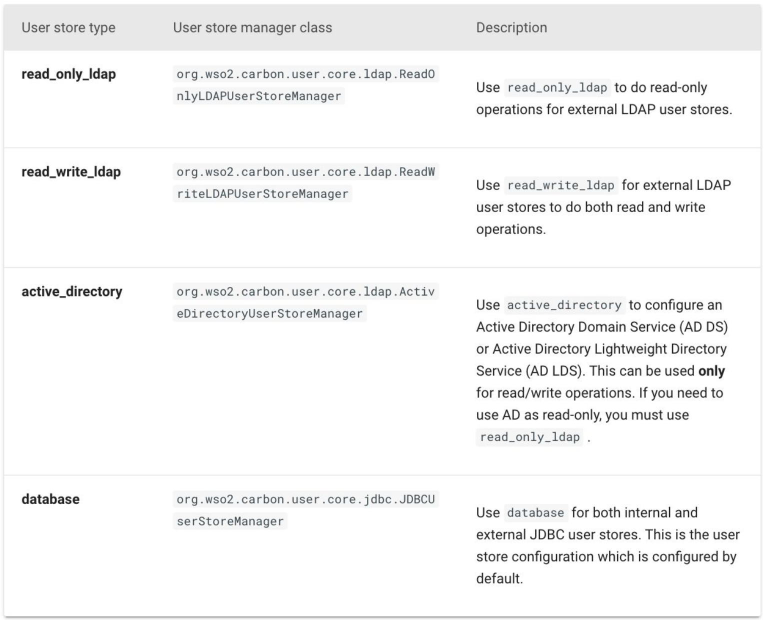
- A user store is the database where information of the users and/or user roles are stored.
- User Information:
  - Usernames, Passwords, First Name, Last Name, Email, etc.
- By default it uses an embedded h2 database.
- Permissions and other authorization related information is stored in a separate database called the user management database, which by default is H2 as well.
- You can also
  - Configure a **primary user store** instead of using embedded h2 database.
  - Configure several **secondary user stores** as well.
  - Configure your own **customized user stores** and connect them with the products as secondary stores.

Docs Link: User Stores

User Store Types

There are four user store types.

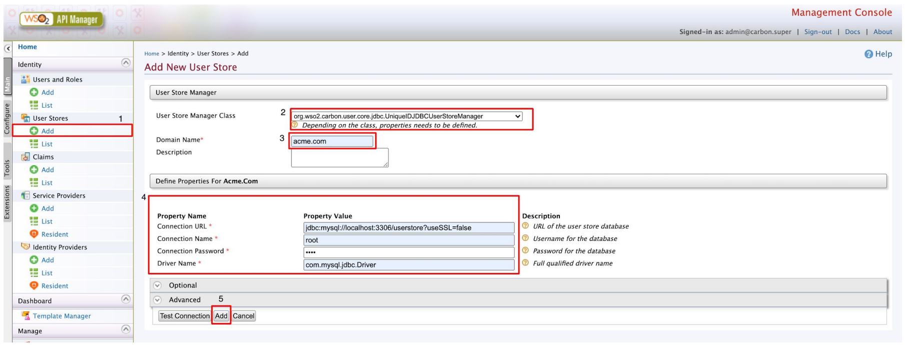
User Store Manager types.



User store type	User store manager class	Description
read_only_ldap	`org.wso2.carbon.user.core.ldap.ReadOnlyLDAPUserStoreManager`	Use `read_only_ldap` to do read-only operations for external LDAP user stores.
read_write_ldap	`org.wso2.carbon.user.core.ldap.ReadWriteLDAPUserStoreManager`	Use `read_write_ldap` for external LDAP user stores to do both read and write operations.
active_directory	`org.wso2.carbon.user.core.ldap.ActiveDirectoryUserStoreManager`	Use `active_directory` to configure an Active Directory Domain Service (AD DS) or Active Directory Lightweight Directory Service (AD LDS). This can be used **only** for read/write operations. If you need to use AD as read-only, you must use `read_only_ldap` .
database	`org.wso2.carbon.user.core.jdbc.JDBCUserStoreManager`	Use `database` for both internal and external JDBC user stores. This is the user store configuration which is configured by default.

Configuring a Secondary User Store

1. Log in to the carbon console and, go to User Stores > Add
2. Select the user store type
3. Enter the user store domain.
4. Enter the user store connection properties
5. Click Add button to add the user store.



Home > Identity > User Stores > Add

Add New User Store

User Store Manager

User Store Manager Class 2 org.wso2.carbon.user.core.jdbc.UniqueIDJDBCUserStoreManager

Depending on the class, properties needs to be defined.

Domain Name* 3 acme.com

Description

Define Properties For **Acme.Com**

4

Property Name	Property Value	Description
Connection URL *	jdbc:mysql://localhost:3306/userstore?useSSL=false	URL of the user store database
Connection Name *	root	Username for the database
Connection Password *	••••	Password for the database
Driver Name *	com.mysql.jdbc.Driver	Full qualified driver name

Optional

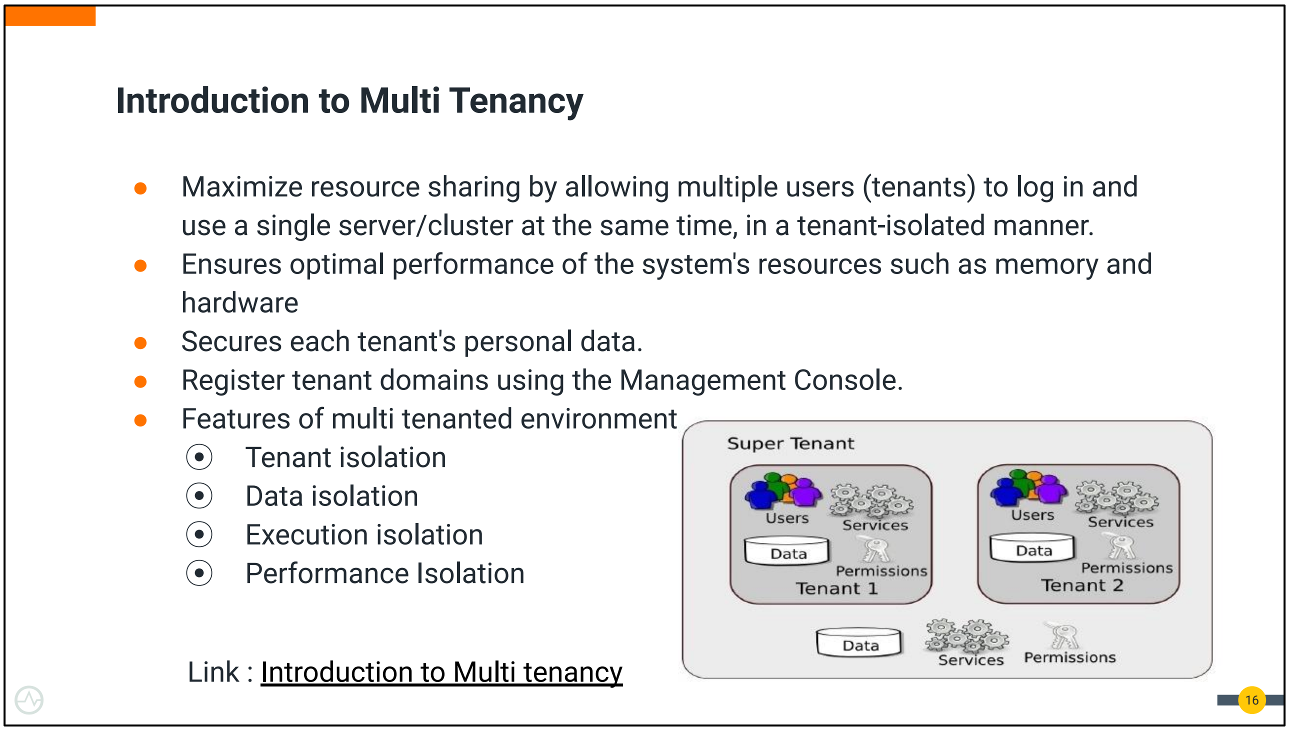
Advanced 5

Test Connection | Add | Cancel

Configuring a Secondary User Store Manually

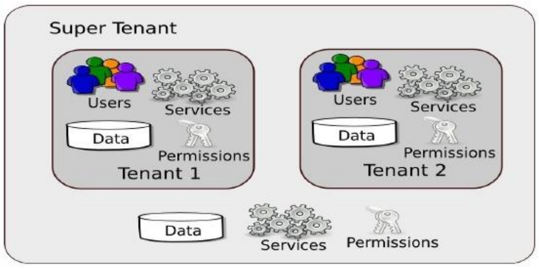
- When you configure multiple user stores, you must **give a unique domain name to each user store** in the `<DomainName>` element.
- If it is the configuration of a super tenant, save the secondary user store definitions in `<APIM_HOME>/repository/deployment/server/userstores` directory.
- If it is a general tenant, save the configuration in `<APIM_HOME>/repository/tenants/<tenantid>/userstores` directory
- The secondary user store configuration file must have the same name as the domain with an underscore (_) in place of the period. For example, if the domain is `wso2.com,` name the file as `wso2_com.xml`
- Only one file should contain the definition for one user store domain.

Configuring a Secondary User Store Manually



Introduction to Multi Tenancy

- Maximize resource sharing by allowing multiple users (tenants) to log in and use a single server/cluster at the same time, in a tenant-isolated manner.
- Ensures optimal performance of the system's resources such as memory and hardware
- Secures each tenant's personal data.
- Register tenant domains using the Management Console.
- Features of multi tenanted environment
  - ⦿ Tenant isolation
  - ⦿ Data isolation
  - ⦿ Execution isolation
  - ⦿ Performance Isolation



Link : Introduction to Multi tenancy

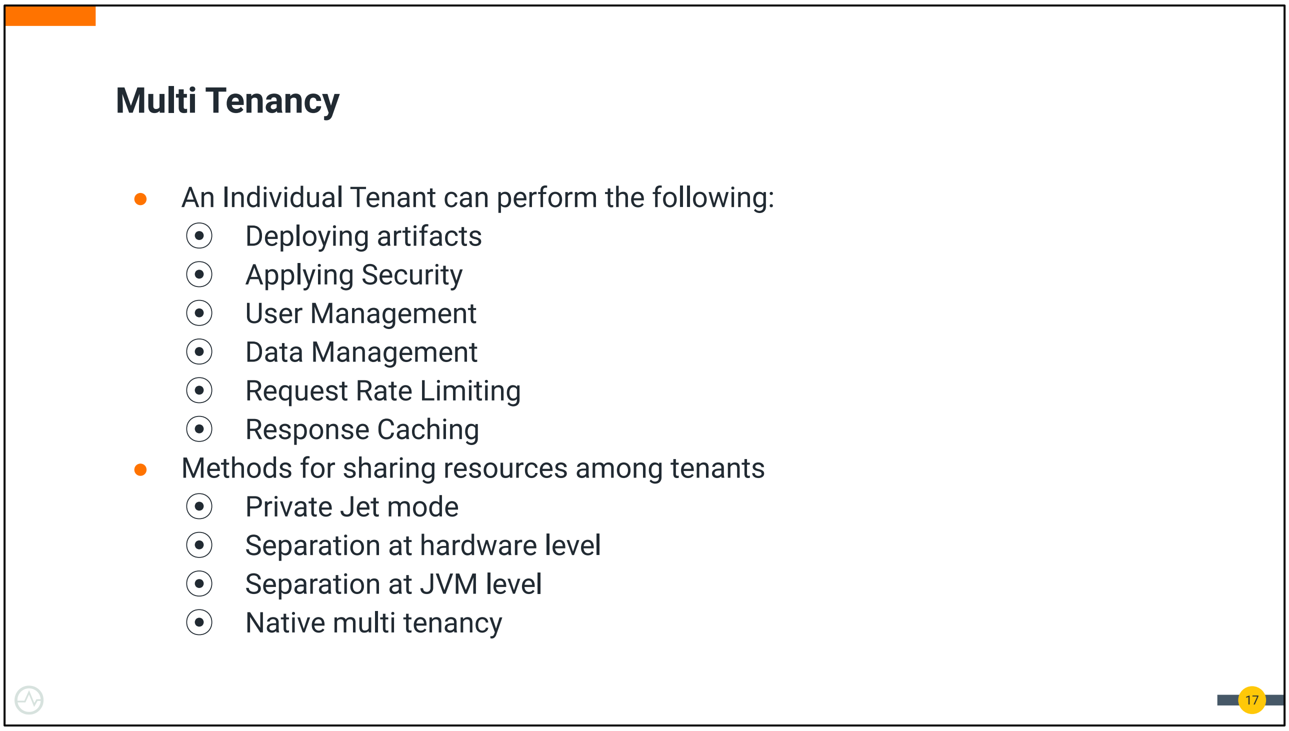
16

What is Multi Tenancy?

Logically isolated entities, sharing the same infrastructure.

All tenants share the same database, user store and other services. But, one tenant's information cannot be accessed by another tenant.

- **Tenant isolation** Each tenant has its own domain, which the other tenants cannot access.
- **Data isolation** Each tenant can manage its data securely in an isolated manner.
- **Execution isolation** Each tenant can carry out business processes and workflows independent of the other tenants. No action of a tenant is triggered or inhibited by another tenant.
- **Performance Isolation** No tenant has an impact on the performance of another tenant.



Multi Tenancy

- An Individual Tenant can perform the following:
  - Deploying artifacts
  - Applying Security
  - User Management
  - Data Management
  - Request Rate Limiting
  - Response Caching
- Methods for sharing resources among tenants
  - Private Jet mode
  - Separation at hardware level
  - Separation at JVM level
  - Native multi tenancy

- **Private Jet mode** : This method allows the load of a tenant ID to be deployed in a single tenant mode. A single tenant is allocated an entire service cluster. The purpose of this approach is to allow special privileges (such as priority processing and improved performance) to a tenant.
- **Separation at hardware level** : This method allows different tenants to share a common set of resources, but each tenant has to run its own operating system. This approach helps to achieve a high level of isolation, but it also incurs a high overhead cost.
- **Separation at JVM level** : This method allows tenants to share the same operating system. This is done by enabling each tenant to run a separate JVM instance in the operating system.
- **Native multi tenancy** : This method involves allowing all the tenants to share a single JVM instance. This method minimises the overhead cost.

Updating WSO2 API Manager

- **WSO2 Update Manager**(WUM) : command-line utility that allows you to get the latest updates that are available for a particular product release.
  - Updates includes the latest bug fixes and security fixes that are released by WSO2 after a particular product version is released.

- **WSO2 in-place updates** : allows you to update your currently used product by fetching updates from the server and merging all configurations and files.
  - Backup and Restore capabilities.

Let’s Try it Out

Working with Tenants