

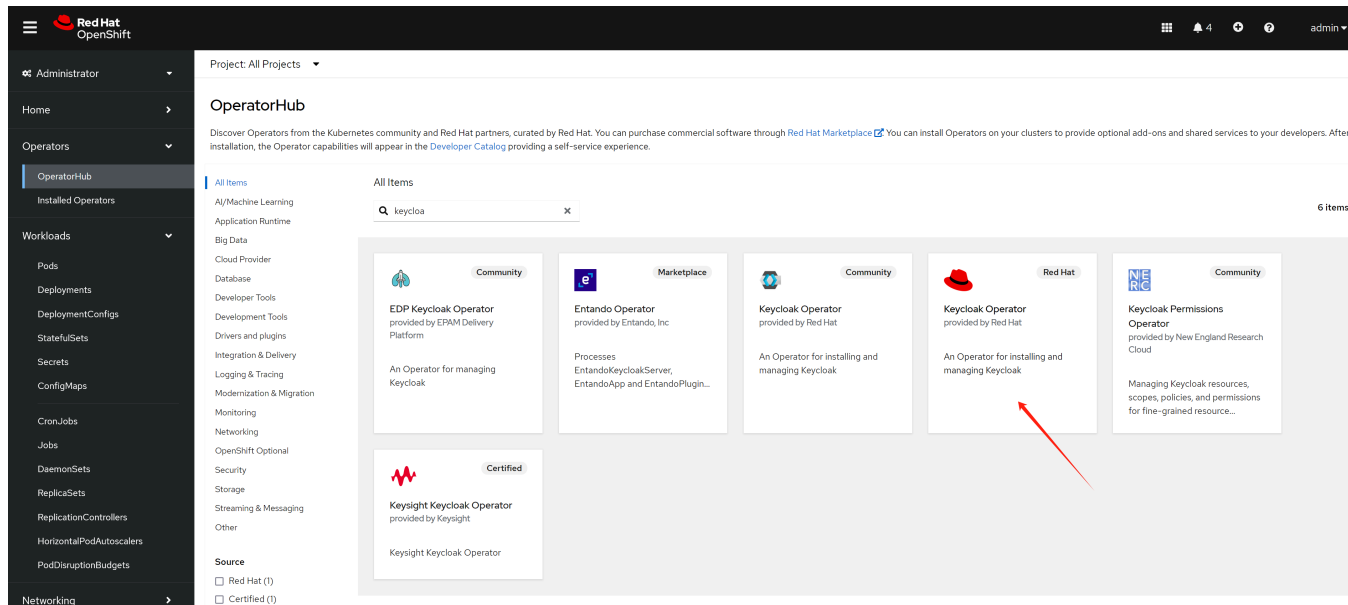
# RHDH 1.4 with Conditional Policy Permission

- Red Hat Developer Hub (RHDH) 1.4 introduces GUI for permission management
- Supports conditional policy for fine-grained access control
- Current implementation has some bugs that require workarounds
- This presentation demonstrates how it works and provides solutions

# Overview

1. Install RHSSO/Keycloak as OAuth2 provider
2. Configure users and groups in Keycloak
3. Install RHDH 1.4
4. Configure RHDH with Keycloak integration
5. Set up conditional policies
6. Test with different user permissions

# Install RHSSO/Keycloak



- Deploy PostgreSQL database for Keycloak
- Configure TLS for secure access
- Deploy Keycloak using Kubernetes operator

# Keycloak Configuration - Key Steps

```
# Create PVC for PostgreSQL
oc create -f keycloak-db-pvc.yaml -n demo-keycloak

# Deploy PostgreSQL database
oc create -f keycloak-db.yaml -n demo-keycloak

# Create TLS certificate
openssl req -subj "/CN=$RHSSO_HOST/O=Test Keycloak./C=US" \
  -newkey rsa:2048 -nodes -keyout key.pem -x509 -days 365 -out certificate.pem

# Create Keycloak instance
oc create -f keycloak.yaml -n demo-keycloak
```

# Keycloak Realm Setup

1. Create a realm named RHDH
2. Create test users (e.g., demo-user )
3. Set non-expiring passwords
4. Create client for RHDH integration

Red Hat | Red Hat build of Keycloak

RHDH

Red Hat build of Keycloak master

RHDH ✓

Create realm

Create realm

### Create realm

A realm manages a set of users, credentials, roles, and groups. A user belongs to and logs into a realm. Realms are isolated from one another and can only manage their own users.

Resource file

Drag a file here or browse to upload

Browse... Clear

1

Upload a JSON file

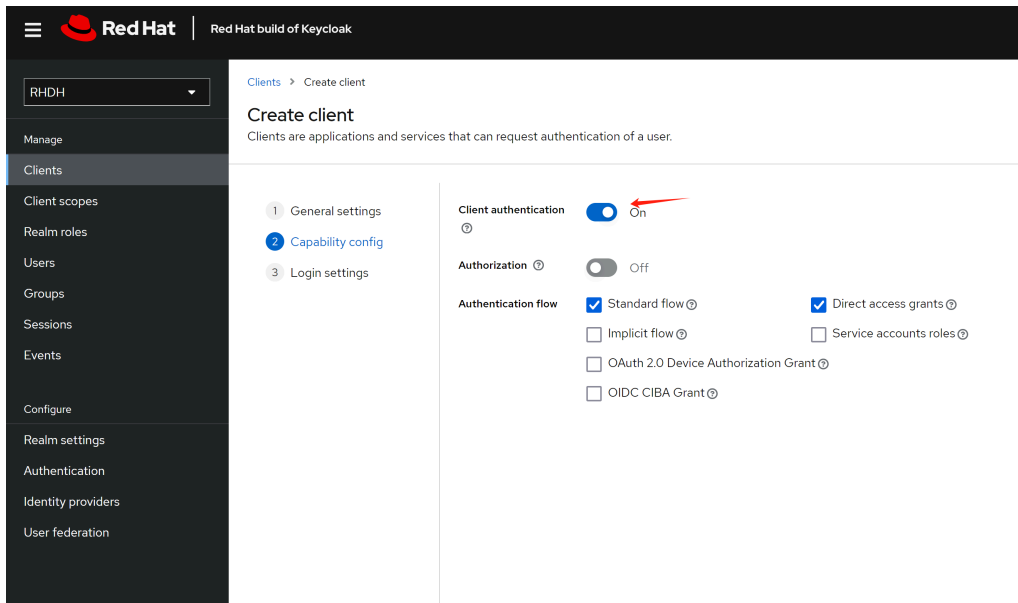
Realm name \*

Enabled ☒ On

Create Cancel

# Keycloak Client Configuration

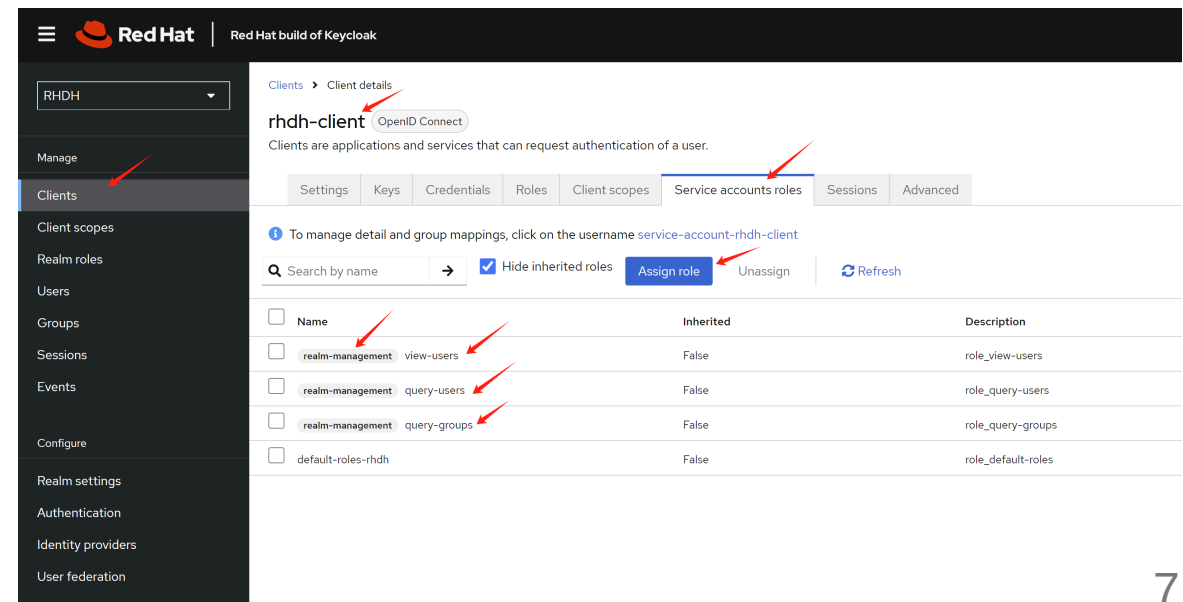
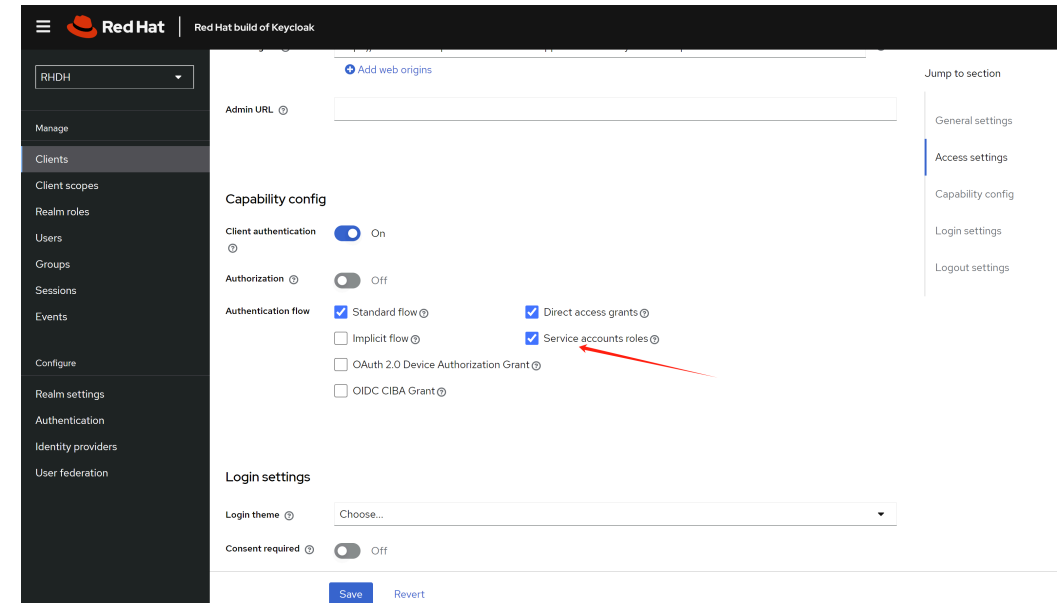
- Set client ID (e.g., `rhdh-client`)
- Configure redirect URL: `https://<RHDH_URL>/api/auth/oidc/handler/frame`
- Set access type to confidential
- Enable service accounts
- Copy client secret for RHDH configuration



# Service Account Roles

For user/group synchronization, add these roles:

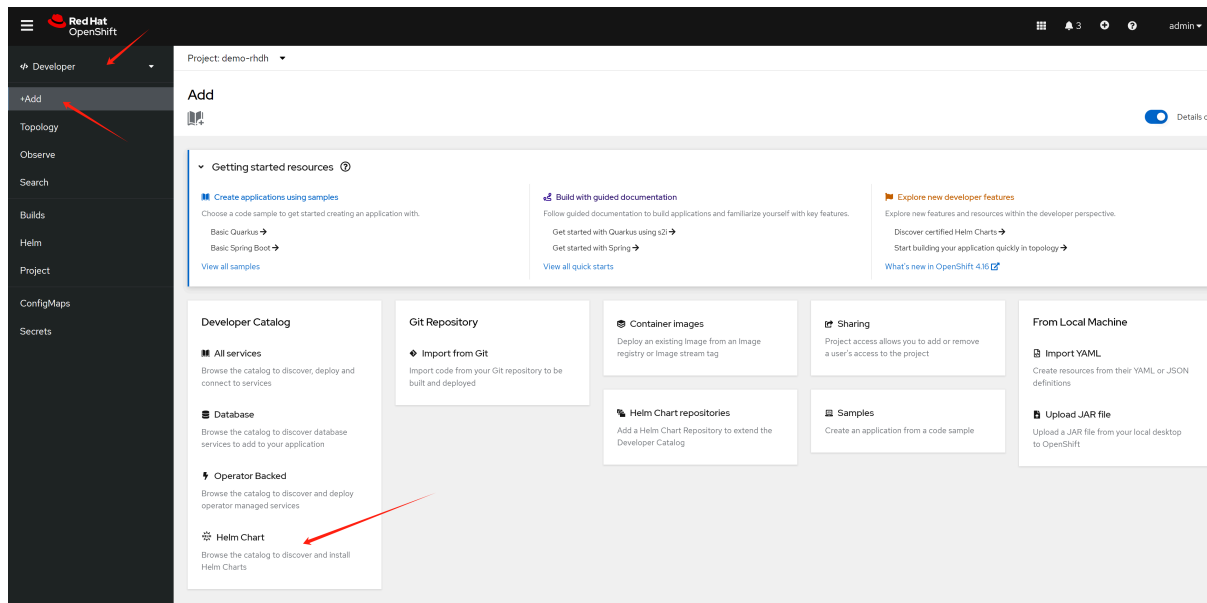
- query-groups
- query-users
- view-users



# Install RHDH 1.4

- Create a new namespace for RHDH
- Install using Helm chart (alternative: operator)

```
oc new-project demo-rhdh
```





# Configure RHDH - Preparation

1. Create service account for Kubernetes access
2. Create PVC for RHDH plugins
3. Generate service account token
4. Set up environment variables

```
# Create service account with appropriate permissions
oc create -f sa-rhdh.yaml -n $NAMESPACES

# Create PVC for plugins
oc apply -f pvc-rhdh.yaml -n $NAMESPACES

# Generate long-lived token
SA_TOKEN=`oc create token backstage-read-only-sa --duration=876000h -n $NAMESPACES`
```

# RHDH Configuration - App Config

Key configuration elements:

- Authentication providers (Keycloak OIDC)
- Catalog sources
- Kubernetes integration
- Permission settings

```
auth:  
  environment: development  
  providers:  
    oidc:  
      development:  
        clientId: ${AUTH_KEYCLOAK_CLIENT_ID}  
        clientSecret: ${AUTH_KEYCLOAK_CLIENT_SECRET}  
        metadataUrl: ${KEYCLOAK_BASE_URL}/realms/${KEYCLOAK_REALM}/.well-known/openid-configuration
```

# RHDH Configuration - Permissions

Enable RBAC and plugin permissions:

```
permission:
  enabled: true
  rbac:
    admin:
      users:
        - name: user:default/static-demo-admin
  pluginsWithPermission:
    - catalog
    - scaffolder
    - permission
    - topology
    - kubernetes
```

# RHDH Configuration - Keycloak Integration

Configure Keycloak as identity provider:

```
catalog:
  providers:
    keycloakOrg:
      default:
        baseUrl: ${KEYCLOAK_BASE_URL}
        loginRealm: ${KEYCLOAK_REALM}
        realm: ${KEYCLOAK_REALM}
        clientId: ${AUTH_KEYCLOAK_CLIENT_ID}
        clientSecret: ${AUTH_KEYCLOAK_CLIENT_SECRET}
      schedule:
        frequency: { minutes: 1 }
        timeout: { minutes: 1 }
        initialDelay: { seconds: 15 }
```

# Helm Configuration

Update Helm chart with:

- Extra app configuration
- Environment variables from secrets
- Dynamic plugins

```
upstream:
  backstage:
    extraAppConfig:
      - configMapRef: app-config-rhdh
        filename: app-config-rhdh.yaml
    extraEnvVarsSecrets:
      - wzh-rhdh-credentials
```

# User and Group Requirements

**Important Workaround:** RHDH 1.4 has bugs where conditional policy doesn't work if users don't belong to groups.

Solution:

- Create multiple test users
- Assign users to appropriate groups
- Test with different permission scenarios

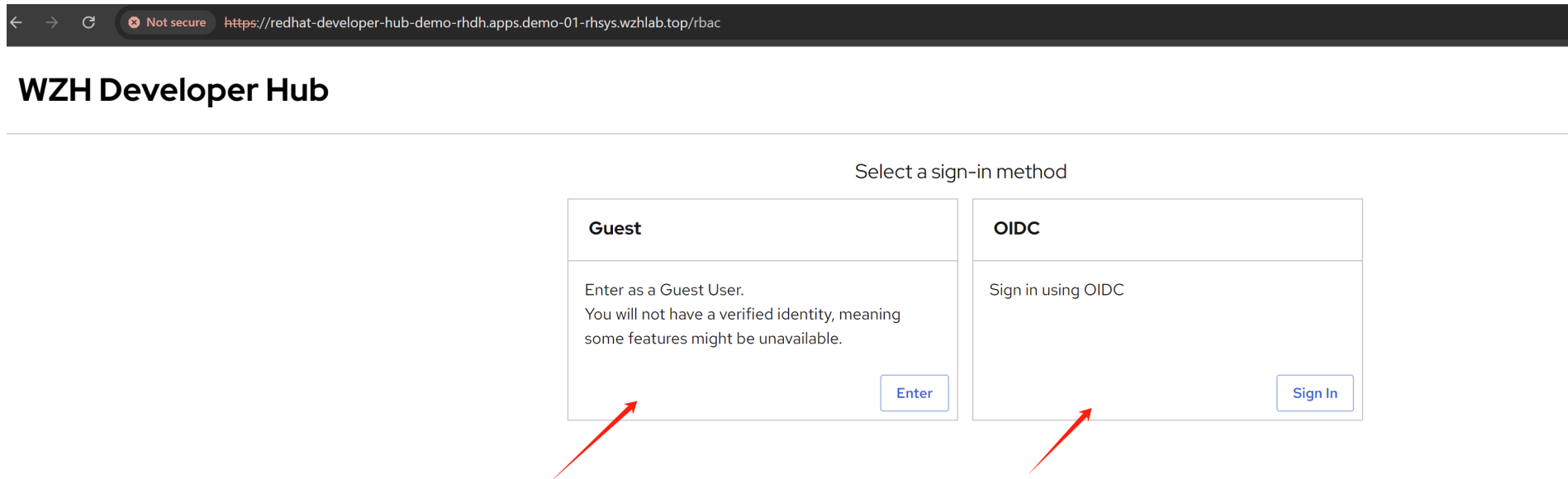
The screenshot displays the Red Hat build of Keycloak Admin Console. The left sidebar shows the navigation menu with 'Groups' highlighted. The main content area shows the 'Group details' for 'kc-group01'. The 'Members' tab is active, displaying a table of group members. Red arrows highlight the 'Groups' menu item, the group name 'kc-group01', and the member names 'kc-user01' and 'kc.user03' in the table.

Name	Email	First name	Last name
kc-user01	kc-user01@wzhlab.top	kc	user01
kc.user03	kc.user03@wzhlab.top	kc.	user03

# Login Options

After configuration, RHDH provides multiple login options:

- Guest login (for admin/troubleshooting)
- OIDC login (normal user access)



The screenshot shows a web browser window with the address bar displaying "Not secure https://redhat-developer-hub-demo-rhdh.apps.demo-01-rhsys.wzhlab.top/rbac". The page title is "WZH Developer Hub". Below the title, there is a heading "Select a sign-in method". There are two login options presented in separate boxes:

- Guest**: Enter as a Guest User. You will not have a verified identity, meaning some features might be unavailable. An "Enter" button is at the bottom right. A red arrow points to this box.
- OIDC**: Sign in using OIDC. A "Sign In" button is at the bottom right. A red arrow points to this box.

# Conditional Policy - Built-in Policies

RHDH comes with built-in policies that cannot be edited:

Red Hat Developer Hub

Search

Home

My Group

Catalog

APIs

Learning Paths

Create...

Docs

Administration

RBAC

Plugins

Settings

Edit Role

Edit name and description of role

Edit users and groups

Edit permission policies

Permission policies can be selected for each plugin. You can add multiple permission policies using +Add option.

What can users/groups access?

Plugin \*  
permission

Resource type \*  
policy-entity

What actions they can do \*

☒ Read

☒ Create

☒ Delete

☒ Update

What can users/groups access?

Plugin \*  
catalog

Resource type \*  
catalog-entity

[Configure access](#)

What actions they can do \*

☒ Read

☐ Delete

☐ Update

[+ Add](#)

Back

Next

Cancel

Red Hat Developer Hub

Search

Home

My Group

Catalog

APIs

Learning Paths

Create...

Docs

Administration

RBAC

Plugins

Settings

RBAC

All roles (4)

Filter

Name	Users and groups	Accessible plugins	Actions
<a href="#">role.default/tbac_admin</a>	1 user	Catalog, Permission	<a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">role.default/static-admin</a>	1 user	Catalog	<a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">role.default/kc.user03</a>	1 user	Catalog	<a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">role.default/demo-user</a>	1 user	Catalog	<a href="#">Edit</a> <a href="#">Delete</a>

5 rows [1-4](#)

[Download User List](#)



# Creating Custom Policies

Custom policies include:

- Name and description
- User/group assignments
- Permission settings
- Conditional rules

The screenshot displays the Red Hat Developer Hub RBAC console interface. The breadcrumb navigation at the top shows the path: RBAC / role:default/kc.user03. The main content area is divided into two panels. The left panel, titled 'Users and groups (1 user)', contains a table with one entry: 'kc.user03' (User) with no members. The right panel, titled 'Permission Policies (1)', contains a table with one entry: 'catalog' (Permission) with 'catalog-entity' (Policy) and 'Read' (Conditional) settings. Red arrows point to the role name in the breadcrumb, the user name in the 'Users and groups' table, and the policy name in the 'Permission Policies' table.

Red Hat Developer Hub

RBAC / role:default/kc.user03

role:default/kc.user03

Overview

About

DESCRIPTION

MODIFIED BY

user:default/static-demo-admin

LAST MODIFIED

24 Feb 2025, 14:27

Users and groups (1 user)

Name	Type	Members
kc.user03	User	-

5 rows | 1 of 1

Permission Policies (1)

Plugin	Permission	Policies	Conditional
catalog	catalog-entity	Read	1 rule

5 rows | 1 of 1

# Permission Configuration

Configure which resources users can access:

Red Hat  
Developer Hub

Search

Home

My Group

Catalog

APIs

Learning Paths

Create...

Docs

Administration

RBAC

Plugins

Settings

RBAC / Edit role

Edit role

Edit Role

✓ Edit name and description of role

✓ Edit users and groups

3 Edit permission policies

Permission policies can be selected for each plugin. You can add multiple permission policies using +Add option.

What can users/groups access?

Plugin \*  
catalog

Resource type \*  
catalog-entity

Configure access (1 rule)

What actions they can do? \*  
☒ Read ☐ Delete ☐ Update

+ Add

Back Next Cancel

# Conditional Rules

Define conditions for resource access:

Red Hat Developer Hub

Search

Home

My Group

Catalog

APIs

Learning Paths

Create...

Docs

Administration

RBAC

Plugins

Settings

RBAC / Edit role

Edit role

Edit name and description of role

Edit users and groups

Edit permission policies

Permission policies can be selected for each plugin. You can add multiple permission policies using +Add option.

What can users/groups access?

Plugin \*

catalog

Resource type

catalog

What actions they can do? \*

☒ Read

☐ Delete

☐ Update

+ Add

Back

Next

Configure access for the catalog-entity

By default, the selected resource type will be visible to the chosen users in step two. If you want to restrict or grant permission to specific plugin resource type rule, select it and add the required parameters.

Condition

Allof

AnyOf

Not

Rule \*

IS\_ENTITY\_KIND

kinds \*

template

List of kinds to match at least one of

Save

Cancel

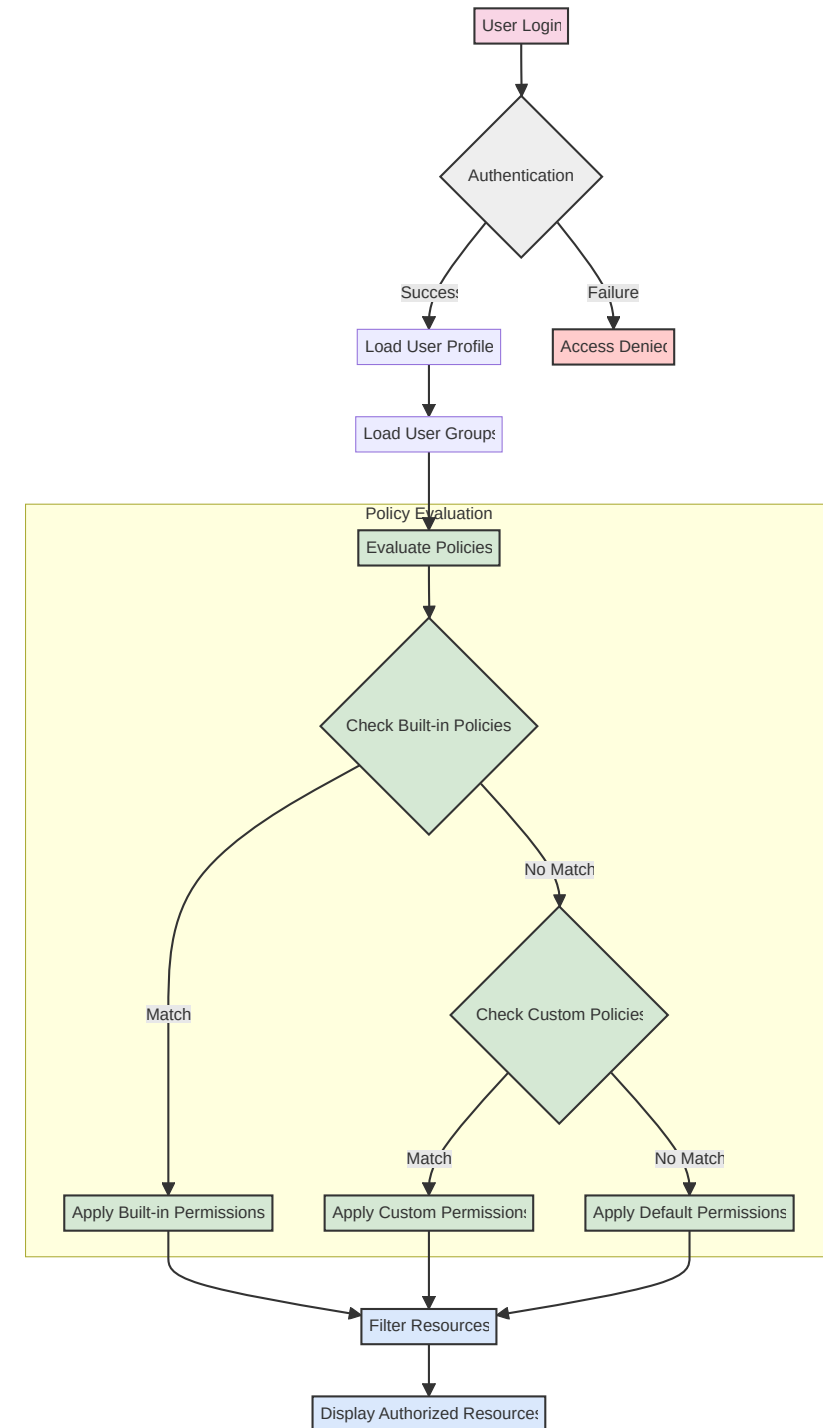
Remove all

19

# Permission Workflow Diagram

RHDH conditional permission workflow:

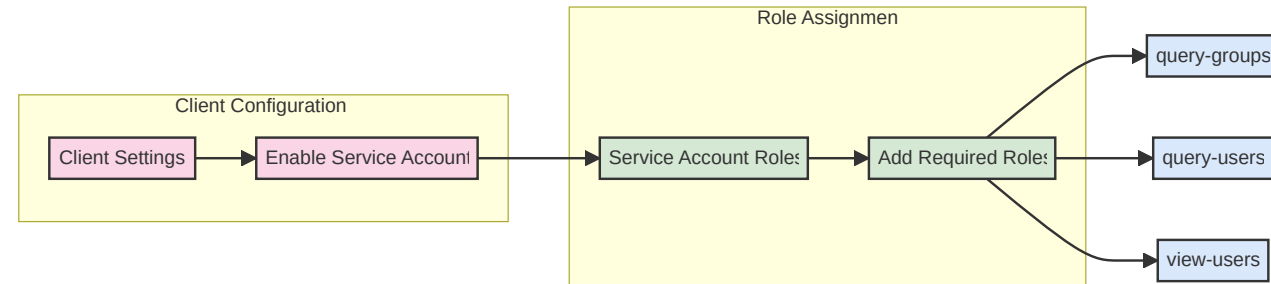
- User Authentication: Verifies user identity and group membership
- Permission Policy Engine: Evaluates user's operation request
- Condition Assessment: Checks if target resources meet defined conditions
- Access Control: Access granted only when all conditions are satisfied



# Service Account Setup

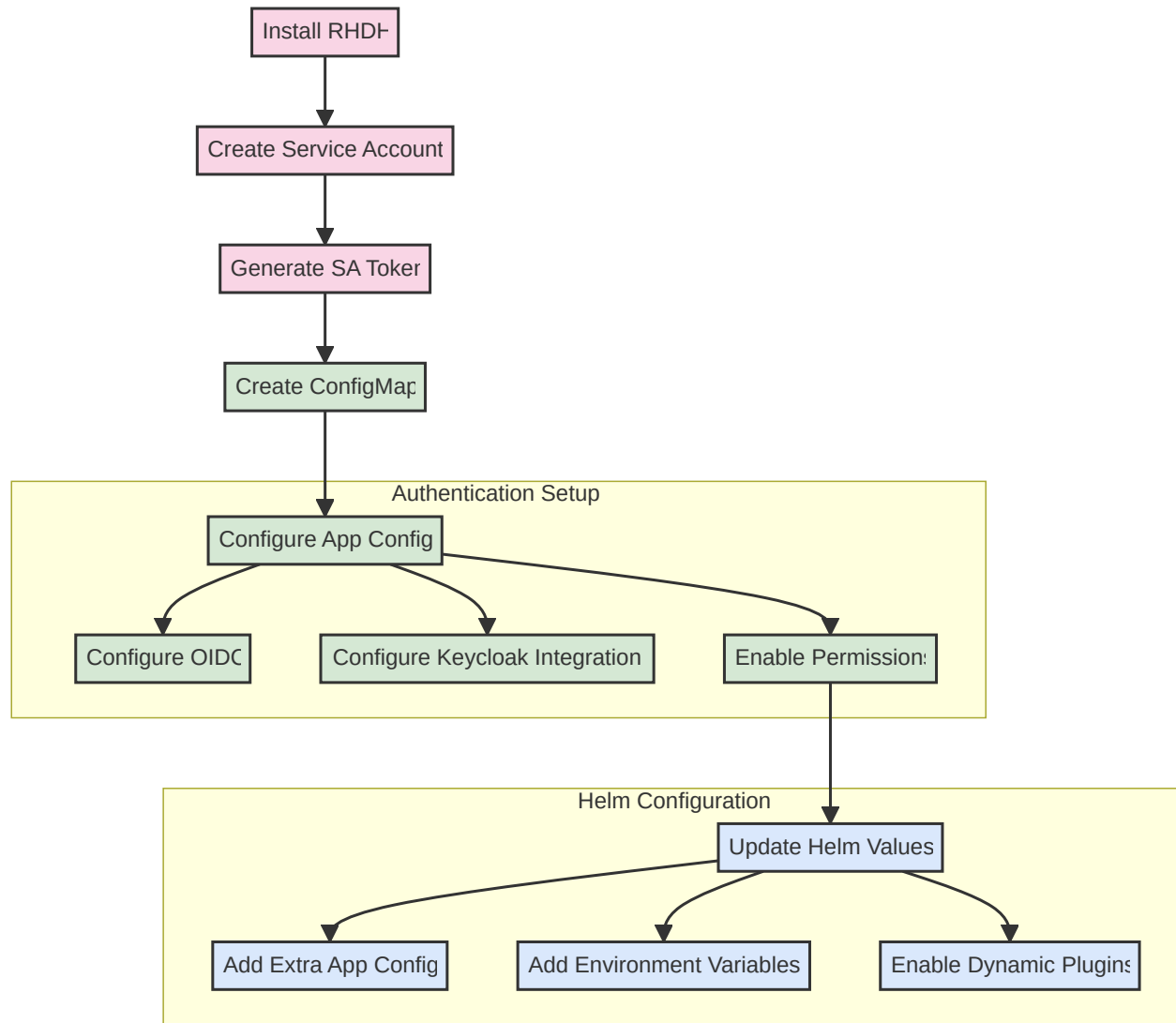
A Keycloak Service Account is essential for RHDH to interact with Keycloak:

- Configure a dedicated client in Keycloak with service account capability
- Enable "Service Account Enabled" option in client settings
- Assign appropriate client roles to access user and group information
- Configure client authentication with client ID and secret credentials



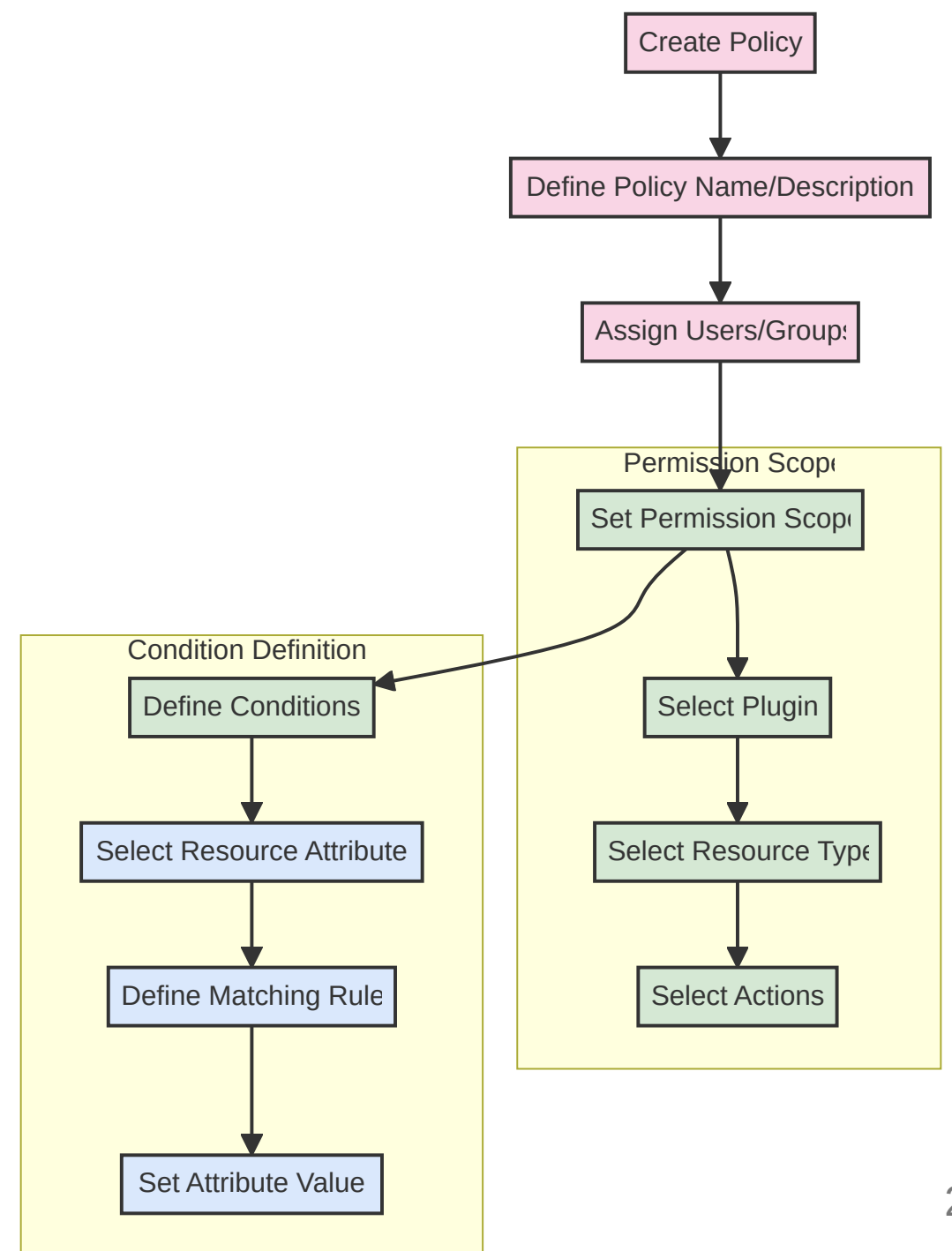
# RHDH Configuration Process

1. **Set up environment variables** - Define all necessary parameters for your RHDH deployment including cluster details, authentication credentials, and service endpoints
2. **Create Kubernetes secrets** - Store sensitive information securely using Kubernetes secrets
3. **Configure app-config YAML** - Define the core RHDH configuration including:
  - Authentication settings
  - Plugin enablement
  - Catalog sources
  - Permission settings
4. **Apply Helm chart configuration** - Complete the setup by:
  - Referencing your custom ConfigMaps
  - Configuring dynamic plugins
  - Setting environment variables from secrets
  - Applying any additional runtime parameters
5. **Deploy and test** - Deploy the updated configuration and verify functionality



# Conditional Policy Workflow

1. **Policy Creation** - Administrators define policies with specific conditions
2. **User Authentication** - Users log in through configured identity providers
3. **Permission Evaluation** - The system evaluates user permissions based on their identity and group memberships
4. **Conditional Filtering** - Resources are filtered according to policy conditions
5. **Resource Access** - Users can only access resources permitted by policies that match their context



# Results

- Users see only resources they have permission to access
- Permissions can be based on:
  - User identity
  - Group membership
  - Resource attributes
  - Custom conditions



# Summary

- RHDH 1.4 provides GUI-based permission management
- Keycloak integration enables robust identity management
- Conditional policies allow fine-grained access control
- Current bugs require workarounds (users must belong to groups)
- Proper configuration enables role-based access to developer resources

# Thank You!

Questions?