OpenShift RBAC, AD Group Integration & Namespace Isolation

# 1. Overview

This document outlines the steps and configuration required to:  
- Integrate Active Directory (AD) groups into OpenShift  
- Assign RBAC roles for Virtual Machine (VM) creation  
- Ensure tenant isolation using namespaces

# 2. Architecture Diagram

The diagram below illustrates how AD groups are integrated and mapped through RoleBindings to isolated VM namespaces:



# 3. Sample YAMLs

## 3.1 Role: VM Management

apiVersion: rbac.authorization.k8s.io/v1  
kind: Role  
metadata:  
 name: vm-manager  
 namespace: dev-project  
rules:  
- apiGroups: ["kubevirt.io"]  
 resources: ["virtualmachines"]  
 verbs: ["create", "get", "list", "delete", "update"]

## 3.2 RoleBinding: AD Group Access

apiVersion: rbac.authorization.k8s.io/v1  
kind: RoleBinding  
metadata:  
 name: devops-vm-access  
 namespace: dev-project  
subjects:  
- kind: Group  
 name: "CN=DevOps-Team,OU=Groups,DC=example,DC=com"  
 apiGroup: rbac.authorization.k8s.io  
roleRef:  
 kind: Role  
 name: vm-manager  
 apiGroup: rbac.authorization.k8s.io

## 3.3 NetworkPolicy: Namespace Isolation

apiVersion: networking.k8s.io/v1  
kind: NetworkPolicy  
metadata:  
 name: deny-all  
 namespace: dev-project  
spec:  
 podSelector: {}  
 policyTypes:  
 - Ingress  
 - Egress

## 3.4 LimitRange: Resource Limits

apiVersion: v1  
kind: LimitRange  
metadata:  
 name: resource-limits  
 namespace: dev-project  
spec:  
 limits:  
 - default:  
 memory: 2Gi  
 cpu: "1"  
 defaultRequest:  
 memory: 1Gi  
 cpu: "500m"  
 type: Container

### ****Step 1: Research available RBAC options within OCP for VM creation and document****

#### Objective:

* Understand what RBAC roles and permissions are needed to allow or restrict VM creation in OpenShift Virtualization.

#### Detailed Plan:

* ✅ Review OpenShift Virtualization official documentation on RBAC.
* ✅ Identify the required API groups for VM creation:
  + kubevirt.io (primary CRDs for VM management)
  + template.openshift.io (if using VM templates)
* ✅ Identify relevant cluster roles:
  + kubevirt.io:edit
  + kubevirt.io:view
  + kubevirt.io:admin (broadest permissions)
* ✅ Document permissions:
  + Which verbs (create, delete, update, patch, get, list, etc.)
  + Which resources (virtualmachines, virtualmachineinstances, templates, datavolumes, persistentvolumeclaims, etc.)
* ✅ Create sample YAMLs for:
  + Role
  + RoleBinding
  + ClusterRoleBinding (if needed)
* ✅ Document how these roles map to operational responsibilities (e.g. Admin, Developer, Read-Only)

#### Deliverable:

* Document with RBAC roles, YAML samples, and permission breakdown.

### ****Step 2: Identify how AD groups can be added and come up with a list****

#### Objective:

* Understand how to integrate AD groups into OpenShift to assign them proper roles.

#### Detailed Plan:

* ✅ Review current OpenShift OAuth Identity Provider configuration:
  + LDAP (typically used for AD integration)
* ✅ Confirm AD groups being synced into OpenShift:
  + Use oc get groups to see synced AD groups.
* ✅ Work with AD team to:
  + Get list of relevant AD groups.
  + Validate group membership.
* ✅ Map AD groups to OpenShift RBAC:
  + oc adm groups add-users <group> <user>
  + Create RoleBinding or ClusterRoleBinding that assigns AD groups to RBAC roles.
* ✅ Create a spreadsheet or table mapping:
  + AD Group Name ➔ OpenShift Role ➔ Namespace/Scope

#### Deliverable:

* Document with list of AD groups, mapped roles, and namespace assignments.

### ****Step 3: Test them in lab environment if possible****

#### Objective:

* Validate that the RBAC roles and AD group assignments work as expected.

#### Detailed Plan:

* ✅ Setup test namespaces/projects.
* ✅ Apply RoleBindings and ClusterRoleBindings.
* ✅ Use test accounts from AD with assigned groups.
* ✅ Perform the following tests:
  + Create VM
  + Delete VM
  + Modify VM
  + View VM
  + Attempt restricted actions (negative test cases)
* ✅ Collect logs and screenshots for documentation.

#### Deliverable:

* Test report with:
  + Test scenarios
  + Expected vs actual results
  + Issues found (if any)
  + Screenshots

### ****Step 4: Research how tenant isolation works within namespaces****

#### Objective:

* Ensure multi-tenancy and isolation between different user groups in OpenShift.

#### Detailed Plan:

* ✅ Review OpenShift Namespace Isolation principles:
  + Default resource isolation
  + NetworkPolicy for network isolation
  + LimitRange and ResourceQuota for resource control
* ✅ Review security controls:
  + SCC (Security Context Constraints)
  + Pod Security Admission
* ✅ Validate VM resource isolation:
  + StorageClass usage
  + PersistentVolume isolation
* ✅ Draft recommendations for:
  + Namespace creation strategy per tenant/group.
  + Recommended policies for isolation (NetworkPolicy, Quota, SCC).
* ✅ Create sample YAML for:
  + NetworkPolicy
  + LimitRange
  + ResourceQuota

#### Deliverable:

* Document explaining tenant isolation architecture with sample YAMLs and recommendations

**f you want, I can prepare:**

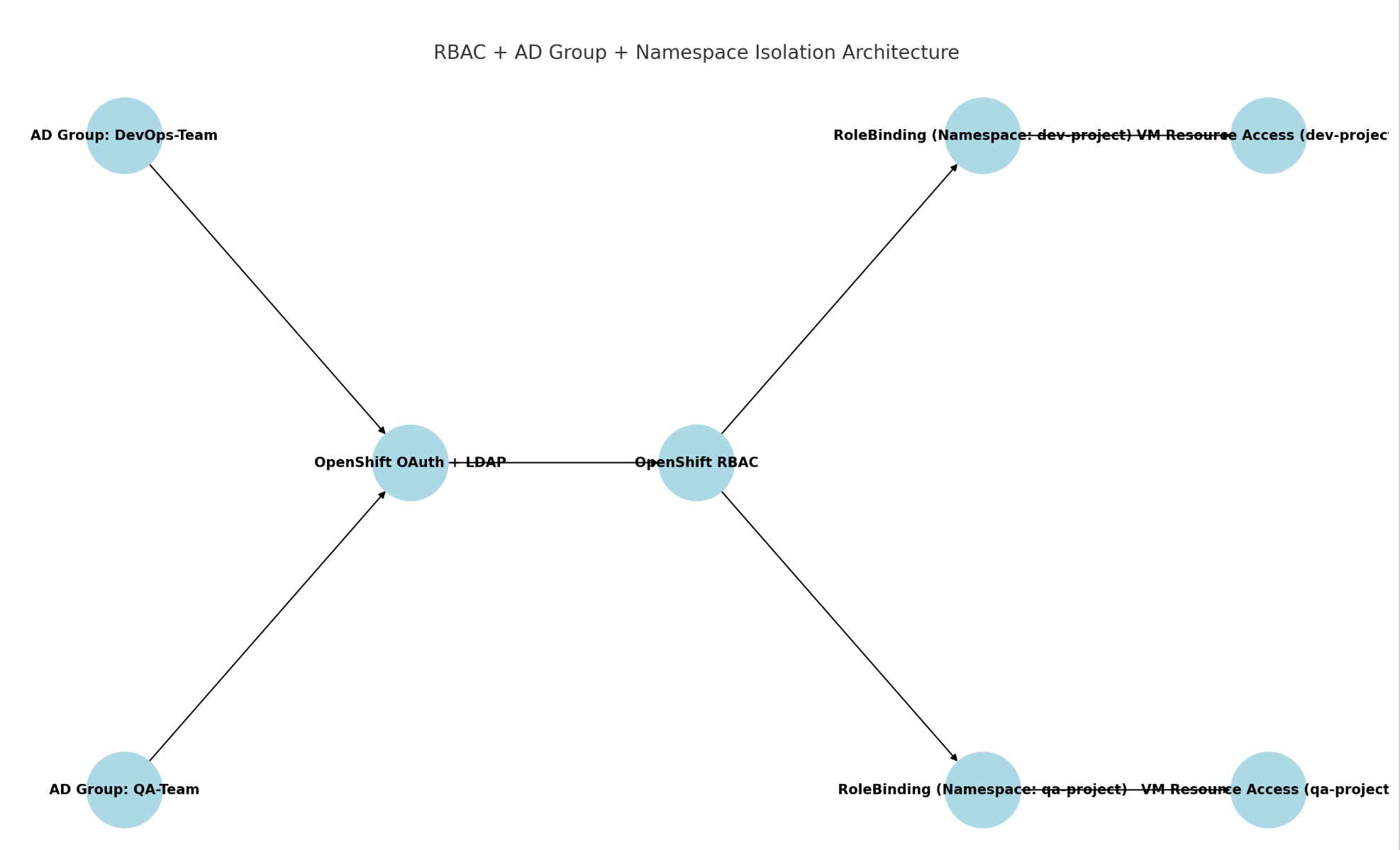
* **Sample YAMLs for each step**
* **Diagram of overall RBAC + AD group + namespace isolation design**

👉 Just say: "**Please create samples and diagram**" if you want me to continue with that.

##### You said:

yes

RBAC + AD Group + Namespace Isolation Architecture

Here is the **architecture diagram** showing the flow of:

* **AD Groups (DevOps & QA Teams)**
* Integration with **OpenShift OAuth via LDAP**
* Application of **RBAC rules through RoleBindings**
* Resulting access to **VM resources within isolated namespaces**

### ✅ ****1. Role for VM Management (Namespace Scoped)****

yaml

CopyEdit

kind: Role

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: vm-manager

namespace: dev-project

rules:

- apiGroups: ["kubevirt.io"]

resources: ["virtualmachines"]

verbs: ["create", "get", "list", "delete", "update"]

### ✅ ****2. RoleBinding for AD Group****

yaml

CopyEdit

kind: RoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: devops-vm-access

namespace: dev-project

subjects:

- kind: Group

name: "CN=DevOps-Team,OU=Groups,DC=example,DC=com"

apiGroup: rbac.authorization.k8s.io

roleRef:

kind: Role

name: vm-manager

apiGroup: rbac.authorization.k8s.io

### ✅ ****3. NetworkPolicy for Namespace Isolation****

yaml

CopyEdit

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: deny-all

namespace: dev-project

spec:

podSelector: {}

policyTypes:

- Ingress

- Egress

### ✅ ****4. LimitRange for Resource Control****

yaml

CopyEdit

apiVersion: v1

kind: LimitRange

metadata:

name: resource-limits

namespace: dev-project

spec:

limits:

- default:

memory: 2Gi

cpu: "1"

defaultRequest:

memory: 1Gi

cpu: "500m"

type: Container

Would you like me to export this into a PDF or Word doc for easier sharing?

##### You said:

yes

Your document is ready. You can download it using the link below:

📄 Download OpenShift\_RBAC\_AD\_Namespace\_Isolation.docx

Let me know if you'd like this converted to PDF or if you want to include test plans or additional scenarios.

##### You said:

tekton pipeline

###### ChatGPT said:

Here’s how you can **integrate a Tekton Pipeline** into your OpenShift setup to automate **VM creation via GitOps or declarative YAML**, especially in coordination with your RBAC and AD group control setup.

## ✅ ****Use Case: VM Creation via Tekton Pipeline****

### 🔧 ****1. Prerequisites****

* OpenShift Pipelines Operator installed (Tekton)
* RBAC in place (as discussed)
* KubeVirt (OpenShift Virtualization) installed
* Git repo containing YAML manifests for VMs

## 🏗️ ****2. Tekton Pipeline Structure****

### ****Resources Required****

* Pipeline
* PipelineRun
* Task: Apply a YAML from Git using kubectl or oc
* ServiceAccount: Bound with appropriate Role (via RoleBinding)

## 📁 ****Folder Structure in Git****

csharp

CopyEdit

vm-deploy/

├── base/

│ └── vm-rhel.yaml

├── overlays/

│ └── dev/

│ └── kustomization.yaml

## 🔨 ****3. Example Tekton**** Task

yaml

CopyEdit

apiVersion: tekton.dev/v1

kind: Task

metadata:

name: apply-vm-yaml

spec:

params:

- name: git-url

type: string

- name: git-revision

type: string

steps:

- name: git-clone

image: alpine/git

script: |

git clone $(params.git-url) -b $(params.git-revision) repo

- name: apply-vm

image: registry.access.redhat.com/openshift4/ose-cli

script: |

oc apply -k repo/overlays/dev

## 📦 ****4. Tekton**** Pipeline

yaml

CopyEdit

apiVersion: tekton.dev/v1

kind: Pipeline

metadata:

name: vm-deployment-pipeline

spec:

params:

- name: git-url

- name: git-revision

tasks:

- name: deploy-vm

taskRef:

name: apply-vm-yaml

params:

- name: git-url

value: $(params.git-url)

- name: git-revision

value: $(params.git-revision)

## ▶️ ****5. Tekton**** PipelineRun

yaml

CopyEdit

apiVersion: tekton.dev/v1

kind: PipelineRun

metadata:

name: vm-deployment-run

spec:

pipelineRef:

name: vm-deployment-pipeline

params:

- name: git-url

value: https://git.example.com/vm-deploy.git

- name: git-revision

value: main

serviceAccountName: vm-creator-sa

Ensure vm-creator-sa has a RoleBinding or ClusterRoleBinding to allow VM creation via kubevirt.io resources.

## 🛡️ ****6. RBAC Sample for Pipeline SA****

yaml

CopyEdit

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

name: tekton-vm-binding

namespace: dev-project

subjects:

- kind: ServiceAccount

name: vm-creator-sa

roleRef:

kind: Role

name: vm-manager

apiGroup: rbac.authorization.k8s.io

### 🚀 Want me to export all this as a .docx or .yaml pack for you to deploy/test?

Just say:

* "Export Tekton docx" — for documentation
* "Bundle as YAML zip" — for deployable manifests

##### You said:

.yaml

###### ChatGPT said:

Your Tekton pipeline YAML bundle is ready. You can download it here:

📦 Download tekton\_vm\_pipeline\_bundle.zip

It includes:

* Task for applying VM manifests
* Pipeline definition
* PipelineRun example
* RBAC RoleBinding for the service account

Let me know if you'd like to add Git clone validation, error handling, or trigger via webhook integration (e.g. GitHub, GitLab).

**Definition of done:**

1. **Research available RBAC options within OCP for VM creation and document**
2. **Identify how AD groups can be added and come up with a list**
3. **Test them in lab environment if possible**
4. **Research how tenant isolation works within namespaces**

If you want, I can help you create:

* A detailed plan for each step
* Example commands or YAMLs for OpenShift
* AD group integration approach
* RBAC role examples for VM creation in OpenShift Virtualization
* Explanation of tenant isolation via namespaces

### ****Step 1: Research available RBAC options within OCP for VM creation and document****

#### Objective:

* Understand what RBAC roles and permissions are needed to allow or restrict VM creation in OpenShift Virtualization.

#### Detailed Plan:

* ✅ Review OpenShift Virtualization official documentation on RBAC.
* ✅ Identify the required API groups for VM creation:
  + kubevirt.io (primary CRDs for VM management)
  + template.openshift.io (if using VM templates)
* ✅ Identify relevant cluster roles:
  + kubevirt.io:edit
  + kubevirt.io:view
  + kubevirt.io:admin (broadest permissions)
* ✅ Document permissions:
  + Which verbs (create, delete, update, patch, get, list, etc.)
  + Which resources (virtualmachines, virtualmachineinstances, templates, datavolumes, persistentvolumeclaims, etc.)
* ✅ Create sample YAMLs for:
  + Role
  + RoleBinding
  + ClusterRoleBinding (if needed)
* ✅ Document how these roles map to operational responsibilities (e.g. Admin, Developer, Read-Only)

#### Deliverable:

* Document with RBAC roles, YAML samples, and permission breakdown.

### ****Step 2: Identify how AD groups can be added and come up with a list****

#### Objective:

* Understand how to integrate AD groups into OpenShift to assign them proper roles.

#### Detailed Plan:

* ✅ Review current OpenShift OAuth Identity Provider configuration:
  + LDAP (typically used for AD integration)
* ✅ Confirm AD groups being synced into OpenShift:
  + Use oc get groups to see synced AD groups.
* ✅ Work with AD team to:
  + Get list of relevant AD groups.
  + Validate group membership.
* ✅ Map AD groups to OpenShift RBAC:
  + oc adm groups add-users <group> <user>
  + Create RoleBinding or ClusterRoleBinding that assigns AD groups to RBAC roles.
* ✅ Create a spreadsheet or table mapping:
  + AD Group Name ➔ OpenShift Role ➔ Namespace/Scope

#### Deliverable:

* Document with list of AD groups, mapped roles, and namespace assignments.

### ****Step 3: Test them in lab environment if possible****

#### Objective:

* Validate that the RBAC roles and AD group assignments work as expected.

#### Detailed Plan:

* ✅ Setup test namespaces/projects.
* ✅ Apply RoleBindings and ClusterRoleBindings.
* ✅ Use test accounts from AD with assigned groups.
* ✅ Perform the following tests:
  + Create VM
  + Delete VM
  + Modify VM
  + View VM
  + Attempt restricted actions (negative test cases)
* ✅ Collect logs and screenshots for documentation.

#### Deliverable:

* Test report with:
  + Test scenarios
  + Expected vs actual results
  + Issues found (if any)
  + Screenshots

### ****Step 4: Research how tenant isolation works within namespaces****

#### Objective:

* Ensure multi-tenancy and isolation between different user groups in OpenShift.

#### Detailed Plan:

* ✅ Review OpenShift Namespace Isolation principles:
  + Default resource isolation
  + NetworkPolicy for network isolation
  + LimitRange and ResourceQuota for resource control
* ✅ Review security controls:
  + SCC (Security Context Constraints)
  + Pod Security Admission
* ✅ Validate VM resource isolation:
  + StorageClass usage
  + PersistentVolume isolation
* ✅ Draft recommendations for:
  + Namespace creation strategy per tenant/group.
  + Recommended policies for isolation (NetworkPolicy, Quota, SCC).
* ✅ Create sample YAML for:
  + NetworkPolicy
  + LimitRange
  + ResourceQuota

#### Deliverable:

* Document explaining tenant isolation architecture with sample YAMLs and recommendations.