VMware Advanced Customer Engagements (ACE) Team

Enterprise PKS Identity Management

Authored by Riaz Mohamed & Raghu Pemmaraju
April 2020

vmware

Table of Contents

Introduction

In this document, we provide an overview of Enterprise PKS Roles and Responsibility and detail steps to configure the supported Enterprise PKS (PKS) identity providers. It is essential to understand the options available when configuring Identity management for PKS (PKS) to ensure the principle of least privileges are applied, and authorized access is provided to the users to ensure the integrity and security of PKS and PKS managed Kubernetes clusters. The steps provided are based on PKS 1.6 and above. The steps here apply to installing and configuring PKS using PKS Management Console (EPMC) or a standalone installation.

PKS consists of several components that require careful consideration to provide authentication and authorization. PKS users can gain access to PKS Control Plane, and PKS managed Kubernetes clusters using the OpenID Connect provider called User Account and Authentication (UAA). UAA is an OAuth2 provider issues tokens to PKS and Kubernetes cluster administrators and users with the appropriate roles. Roles can be cluster administrator, developer, or someone with custom access.

We discuss all the necessary configuration steps in this document, starting with a table that shows all of the PKS components that require user access. PKS supports three types of identity management: endpoints, local accounts, LDAP and SAML integrations.

Component	User Access	API	User
Enterprise PKS Management Console (EPMC)	Local only (EPMC 1.7 will support LDAP and SAML)		root
OpsMan	Local[1] or LDAP or SAML		admin (Found in EPMC metadata)
Bosh	Local or LDAP or SAML		admin (Found in EPMC metadata)
PKS	1. Local Account 2. Integrate to LDAP or SAML	pks	LDAP users For local accounts admin (Found in EPMC metadata)
K8 clusters	Integrate to LDAP or SAML	K8 API	LDAP users

[1] Local Only when using EPMC as all metadata is stored in EPMC

Harbor	1. Local Account 2. Integrate to LDAP or SAML	Harbor	LDAP users For local accounts admin (Found in EPMC metadata)

EMPC - VMware Enterprise PKS Management Console provides a unified installation experience for deploying Enterprise Pivotal Container Service (Enterprise PKS) to vSphere.

OpsMan - Platform provides a set of APIs and a graphical interface to manage the deployment and upgrade of PKS components

Bosh - Unifies release engineering, deployment, and lifecycle management of small and large-scale PKS clusters

PKS - is an enterprise Kubernetes platform, architected for rapid results, scaling, and reliability on any infrastructure

K8 Cluster access - The recommended approach to access Kubernetes clusters provisioned via PKS is to use Role-based access control (RBAC) to regulate access to computer or network resources based on the roles of individual users within an enterprise. More information can be found *here*.

Harbor - Harbor is an open source container image registry that secures images with role-based access control, scans images for vulnerabilities, and signs images as trusted

Personas and Roles

An organization that deploys PKS should keep in mind three types components and personas:

1. PKS Control Plane
    a. Administrators
    b. Users/Developers

2. Kubernetes Clusters
    a. Administrators
    b. Developers
    c. Users

3. Harbor Registry
    a. Administrators
    b. Users/Developers

Let us go into the details.

Role Definitions

Roles are a set of defined rights that are assumed by users associated to it to perform specific operations. Role-based access control (RBAC) is a method of regulating access to computer or network resources based on the roles of individual users within your organization.

Enterprise PKS roles

PKS defines a set of roles that are applicable to the PKS control plane. PKS access gives users the ability to deploy and manage Kubernetes clusters. This includes creating clusters, upgrading clusters, deleting clusters, creating network profiles, scaling cluster nodes etc. There are three roles or UAA profiles within PKS:

pks.clusters.manage: Accounts with this scope can create and access their own clusters

pks.clusters.admin: Accounts with this scope can create and access all clusters.

pks.clusters.admin.read: Accounts with this scope can access any information about all clusters except for cluster credentials.

It is straightforward that pks.clusters.admin should be assigned to an individual or group responsible for all of the clusters created by the PKS control plane.

You can assign pks.clusters.manage is role that you want them to create clusters, but not let them view clusters they have not created or those created by someone with pks.clusters.admin role. Be careful who is given access to pks.clusters.manage and pks.clusters.admin

pks.clusters.admin.read role is best for a developer. With this role, they can have readonly access to all of the clusters and they can issue 'pks get-kubeconfig' command to get kubeconfig for the Kubenetes cluster they have access.

Kubernetes Roles:

An RBAC Role or ClusterRole contains rules that represent a set of permissions. Permissions are purely additive (there are no "deny" rules).

A Role always sets permissions within a particular namespace; when you create a Role, you have to specify the namespace it belongs in.

ClusterRole, by contrast, is a non-namespaced resource. The resources have different names (Role and ClusterRole) because a Kubernetes object always has to be either namespaced or not namespaced; it can't be both.

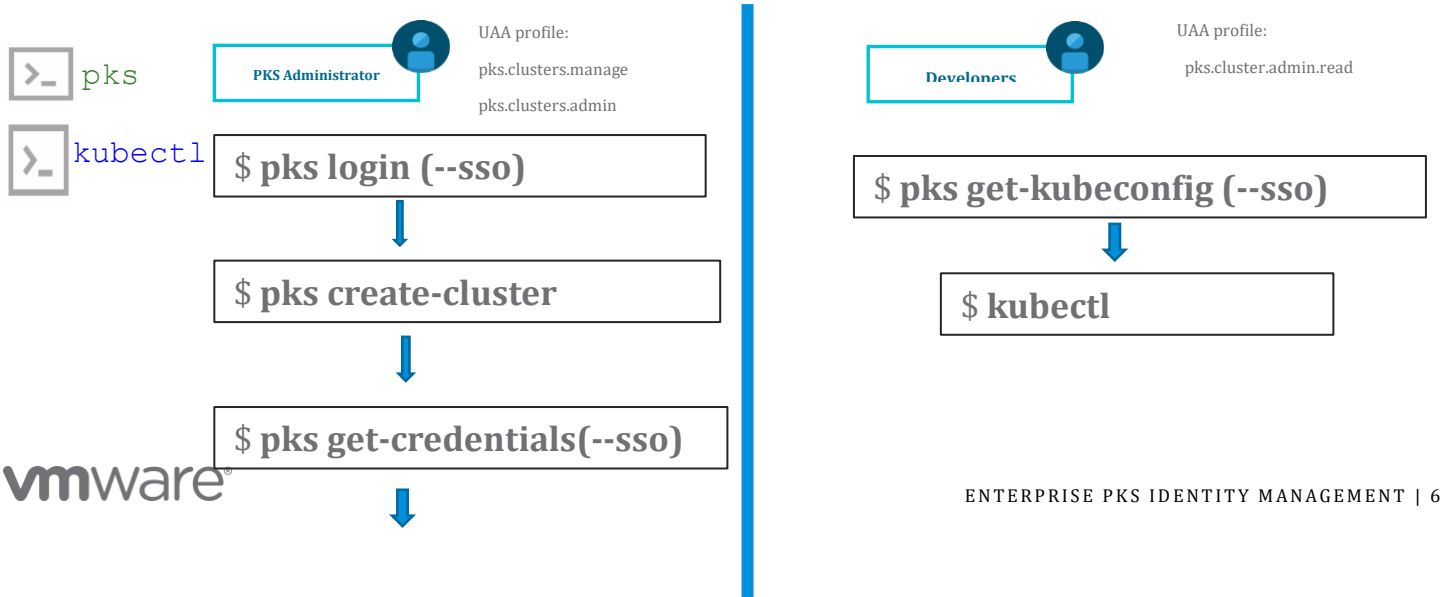
ClusterRoles have several uses. You can use a ClusterRole to:

- define permissions on namespaced resources and be granted within individual namespace(s)
- define permissions on namespaced resources and be granted across all namespaces
- define permissions on cluster-scoped resources

If you want to define a role within a namespace, use a Role; if you want to define a role cluster-wide, use a ClusterRole.

PKS RBAC

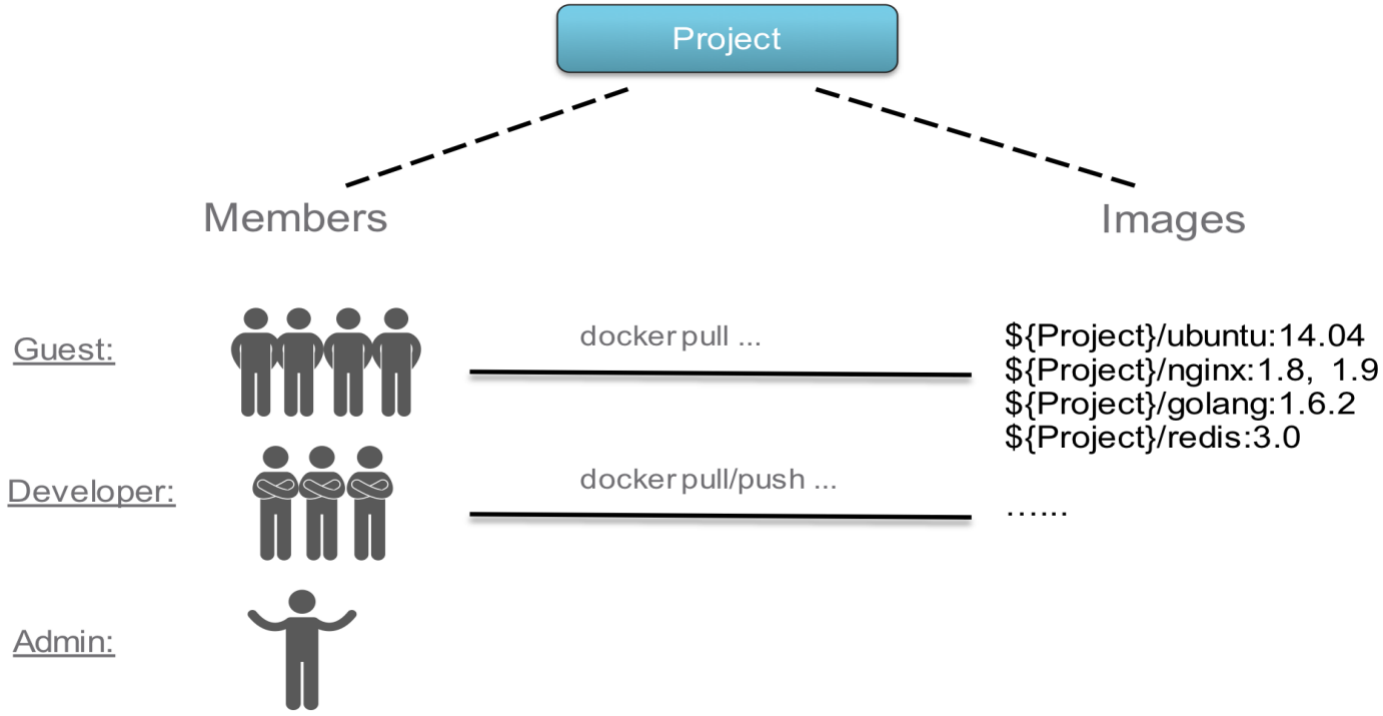
The diagram below shows how different personals gain access to PKS and Kubernetes.



```
$ kubectl
```

Harbor Roles:

Harbor manages images through projects. You provide access to these images to users by including the users in projects and assigning one of the following roles to them.



Limited Guest: A Limited Guest does not have full read privileges for a project. They can pull images but cannot push, and they cannot see logs or the other members of a project. For example, you can create limited guests for users from different organizations who share access to a project.

Guest: Guest has read-only privilege for a specified project. They can pull and retag images, but cannot push.

Developer: Developer has read and write privileges for a project.

Master: Master has elevated permissions beyond those of 'Developer' including the ability to scan images, view replications jobs, and delete images and helm charts.

ProjectAdmin: When creating a new project, you will be assigned the "ProjectAdmin" role to the project. Besides read-write privileges, the "ProjectAdmin" also has some management privileges, such as adding and removing members, starting a vulnerability scan.

Besides the above roles, there are two system-level roles:

Harbor system administrator: "Harbor system administrator" has the most privileges. In addition to the privileges mentioned above, "Harbor system administrator" can also list all projects, set an ordinary user as administrator, delete users and set vulnerability scan policy for all images. The public project "library" is also owned by the administrator.

Anonymous: When a user is not logged in, the user is considered as an "Anonymous" user. An anonymous user has no access to private projects and has read-only access to public projects.

Personas

To help you understand how we plan the various personas found in an organization, we will use the following personas.

Alana
Alana is a PKS Administrator and Operator. This person is involved in the following aspects of Pivotal Container Service:

- Initial PKS Deployment and Installation activities
- PKS platform upgrades and systems patches
- PKS Tile Administration and the PKS control plans
- Setting PKS Role-Based Access Control for the PKS control plane. This person provides access and authorization to any underlying PKS Cluster Administrators.

Cody
Cody could create a PKS cluster and become its PKS cluster Admin. This person is involved in the following aspects of Pivotal Container Service:

- Can create a PKS cluster through mechanism of pks cli: pks create-cluster ...
- Can be the PKS Cluster Administrator.
- Sets Kubernetes Role-Based Access Control to a specific PKS Cluster for End-users and Developers within the Kubernetes control plane.
- Assume the role of a Harbor system administrator
- Setup User access to harbor projects

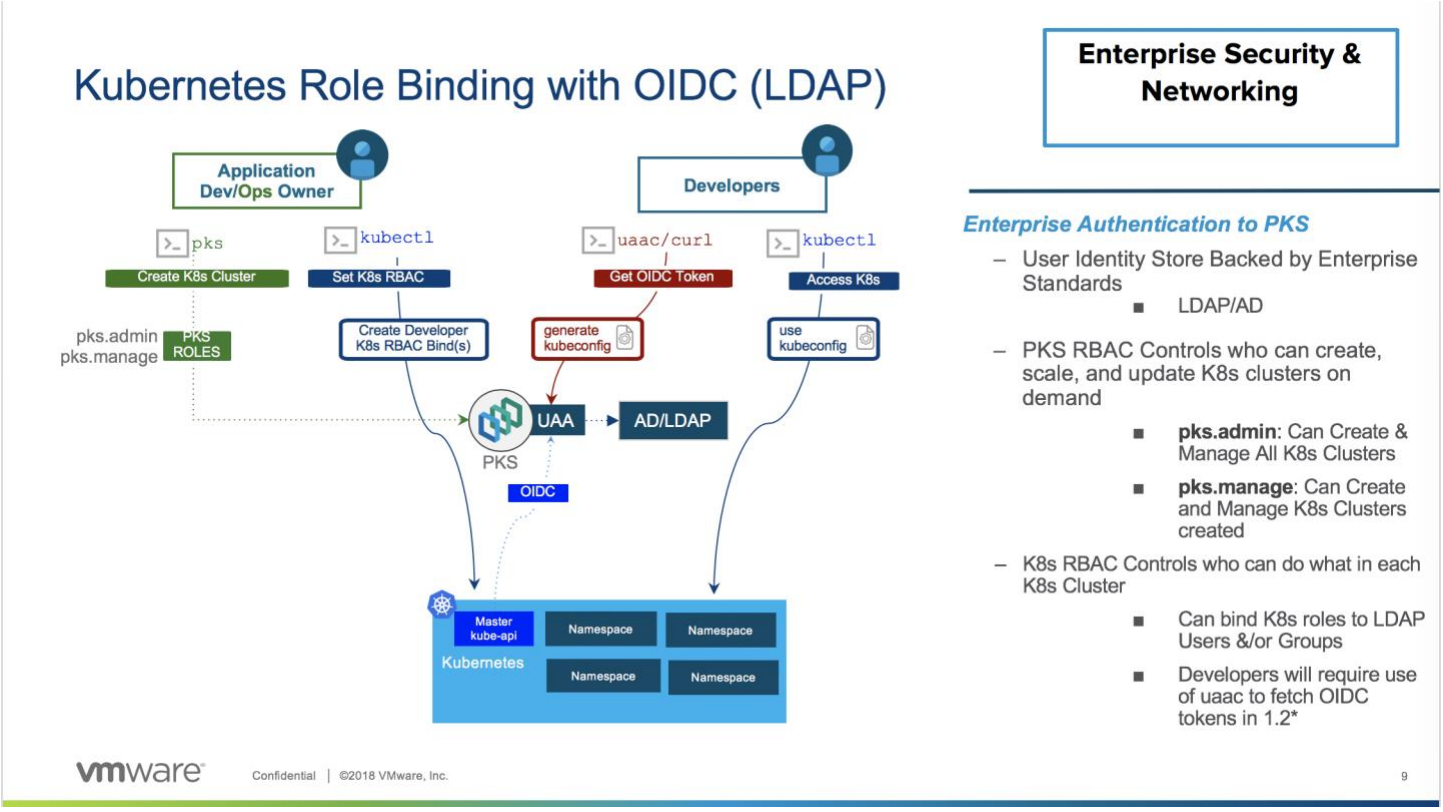
Naomi
Naomi is an end-user or developer on a Kubernetes cluster. This person is involved in the following aspects of Pivotal Container Service:

- Access a Kubernetes API for the cluster they have access to
- Can use the kubectl cli, *Kubernetes Dashboard*, or other Kubernetes API clients to interact with a specific PKS Cluster
- Harbor Developer access

Scott
Scott is an application developer. This person is involved in the following aspects of Pivotal Container Service:

- Access a Kubernetes API for the cluster they have access to with specific granular roles
- Can use the kubectl cli, *Kubernetes Dashboard*, or other Kubernetes API clients to interact with a specific PKS Cluster
- Harbor Developer access

Identity Management Endpoints



PKS supports the following identity management endpoints

- Local User Database – Uses UAA

- AD/LDAP
- SAML Identity provider

Harbor supports the following identity management endpoints

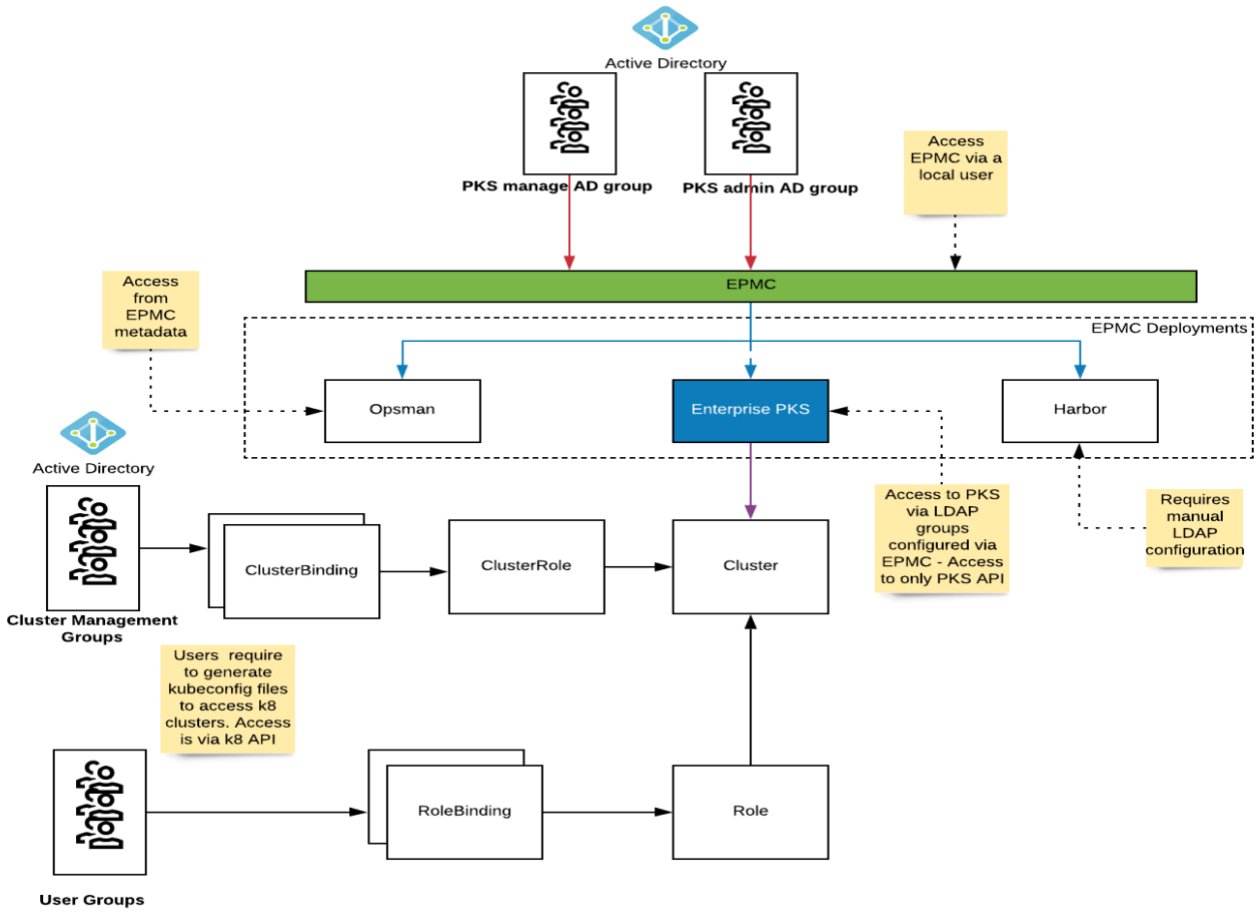
- Harbor internal user management
- AD/LDAP
- UAA in Pivotal Container Service

Group Definitions

This section defines the usergroups we will be creating within the different identity management endpoints to associate the different personas

Group	Persona	Role
pksclusteradmin	alana	pks.cluster.admin
pksclustermanage	cody	pks.cluster.manage
pksclusterread	naomi scott	pks.cluster.read
k8clusteradmins	naomi	cluster-admin
k8developers	scott	create, list , get on pods in namespace dataengg
harboradmins	cody naomi	Harbor system administrator
harborusers	scott	Harbor developer

LDAP – As an Identity Endpoint



This section provides the steps to set up LDAP that we would be using in the later sections to configure PKS components.
For this setup, we would be using Microsoft Active Directory (AD) as the LDAP provider.

The document assumes to have all users under

CN=Users,DC=corp,DC=local

The groups definitions are created under

OU=pks,OU=Infra,DC=corp,DC=local

NOTE: You could have different OU where your user's exists eg. IT(OU)/Offices(OU)/Boston

Required LDAP Groups Configuration

This section provides an overview on setting up LDAP groups and users that we would be using in the later sections. The LDAP Groups settings will be as follows:

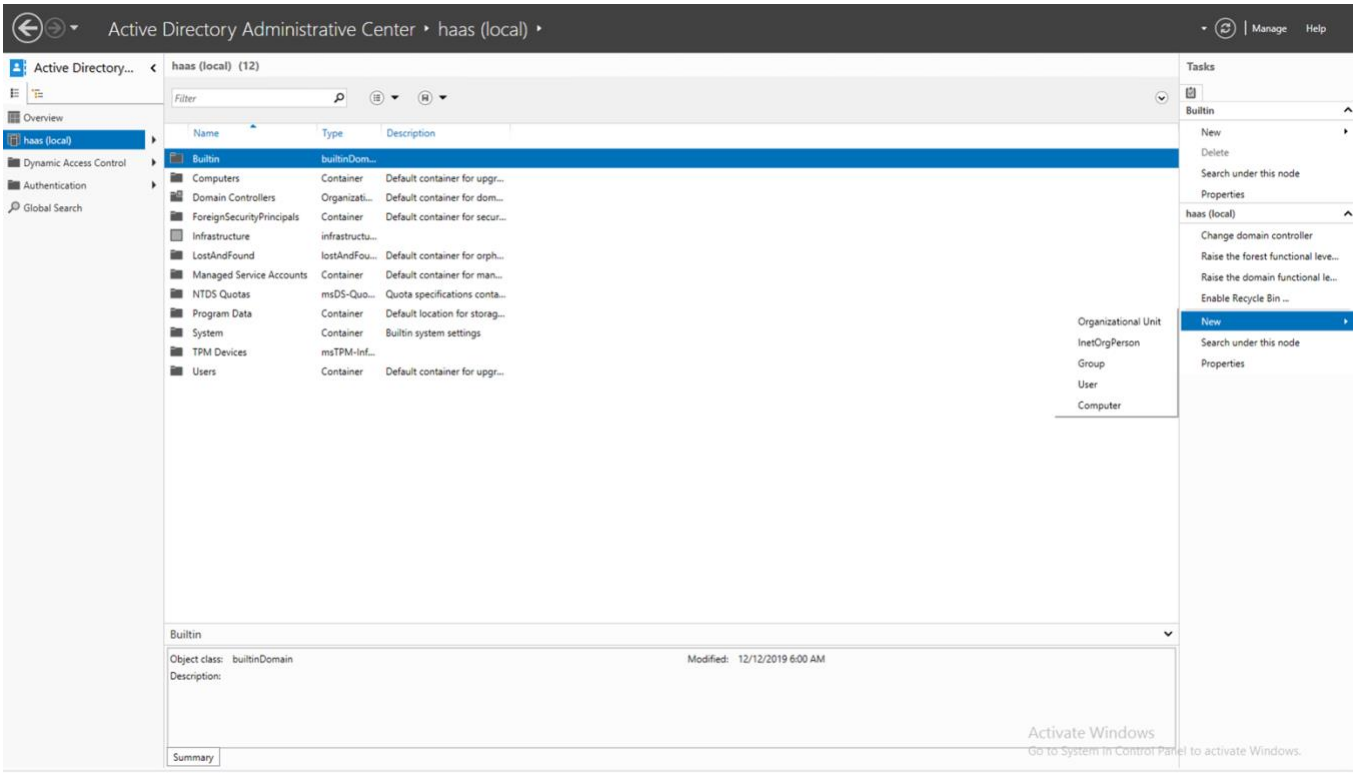
Setup Users and Groups

This section provides the steps to create ou's, groups, users and assign users to groups.

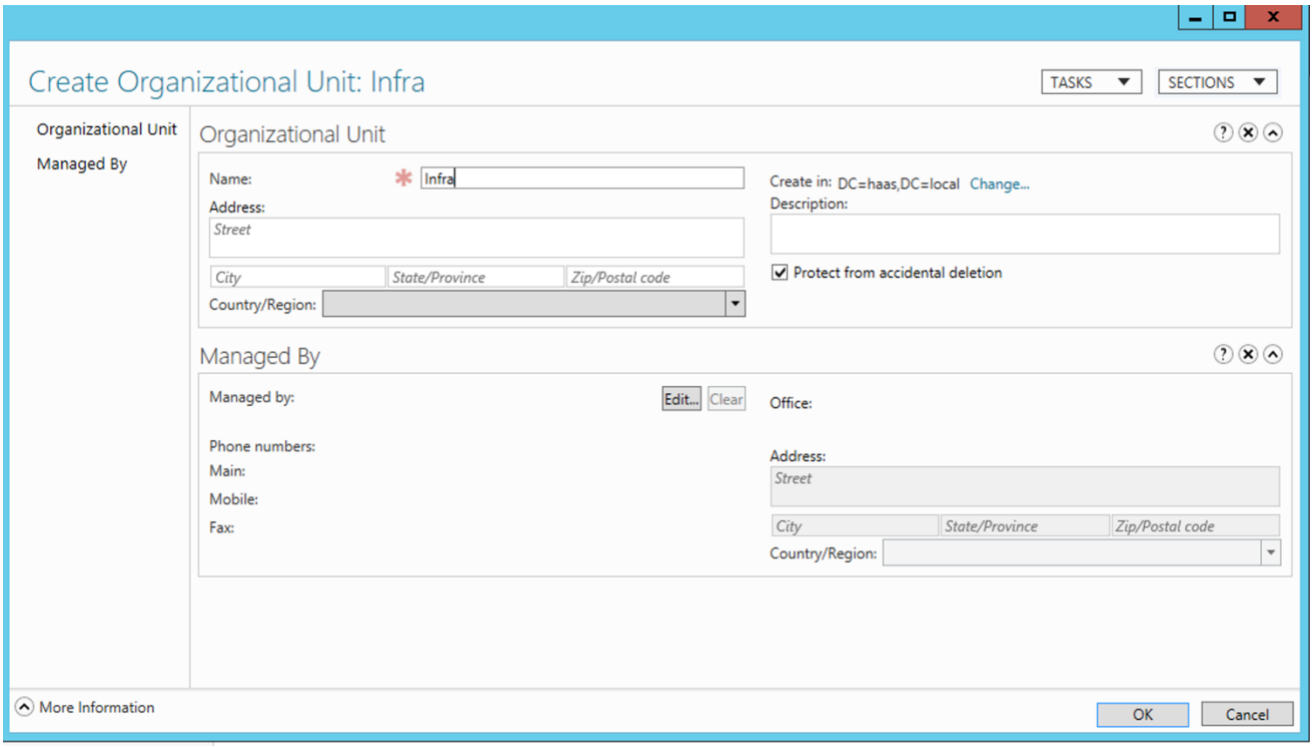
Create Groups

These groups will be used to configure access to different components within the PKS infrastructure.

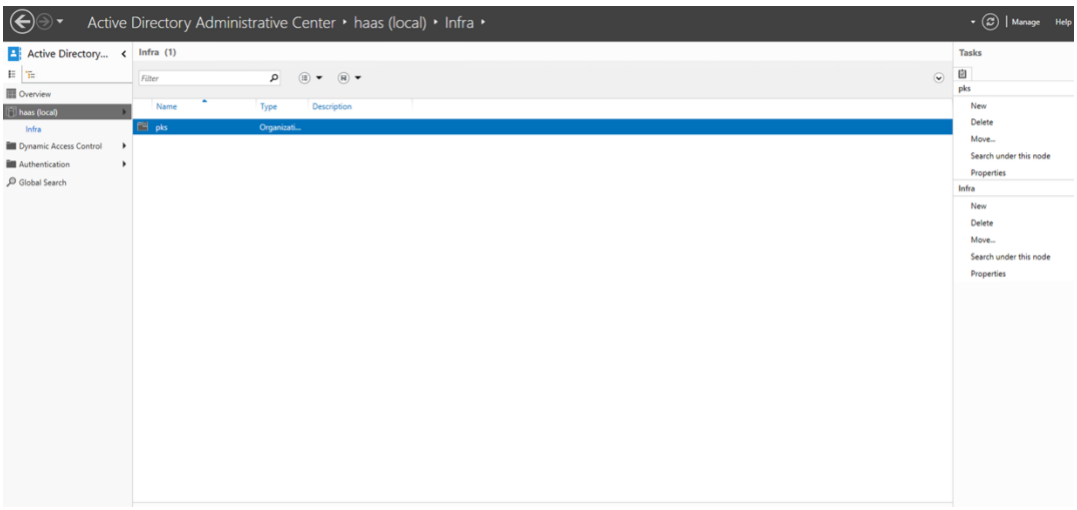
Step 1: Select New Organizational Unit:



Step 2: Enter Name e.g. Infra and click OK. The OU must be created:

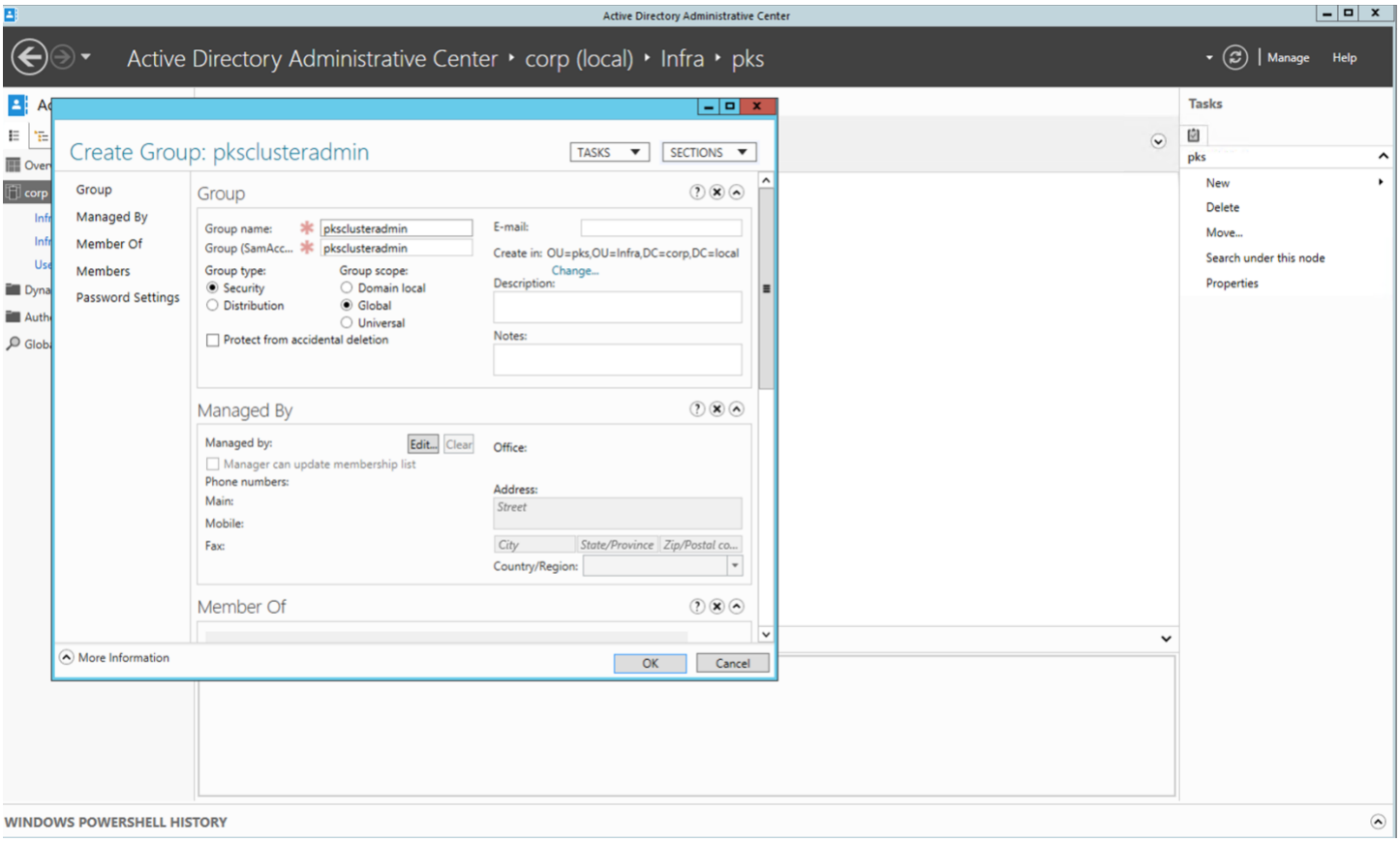


Step 3: Drill down to Infra and create another OU pks:

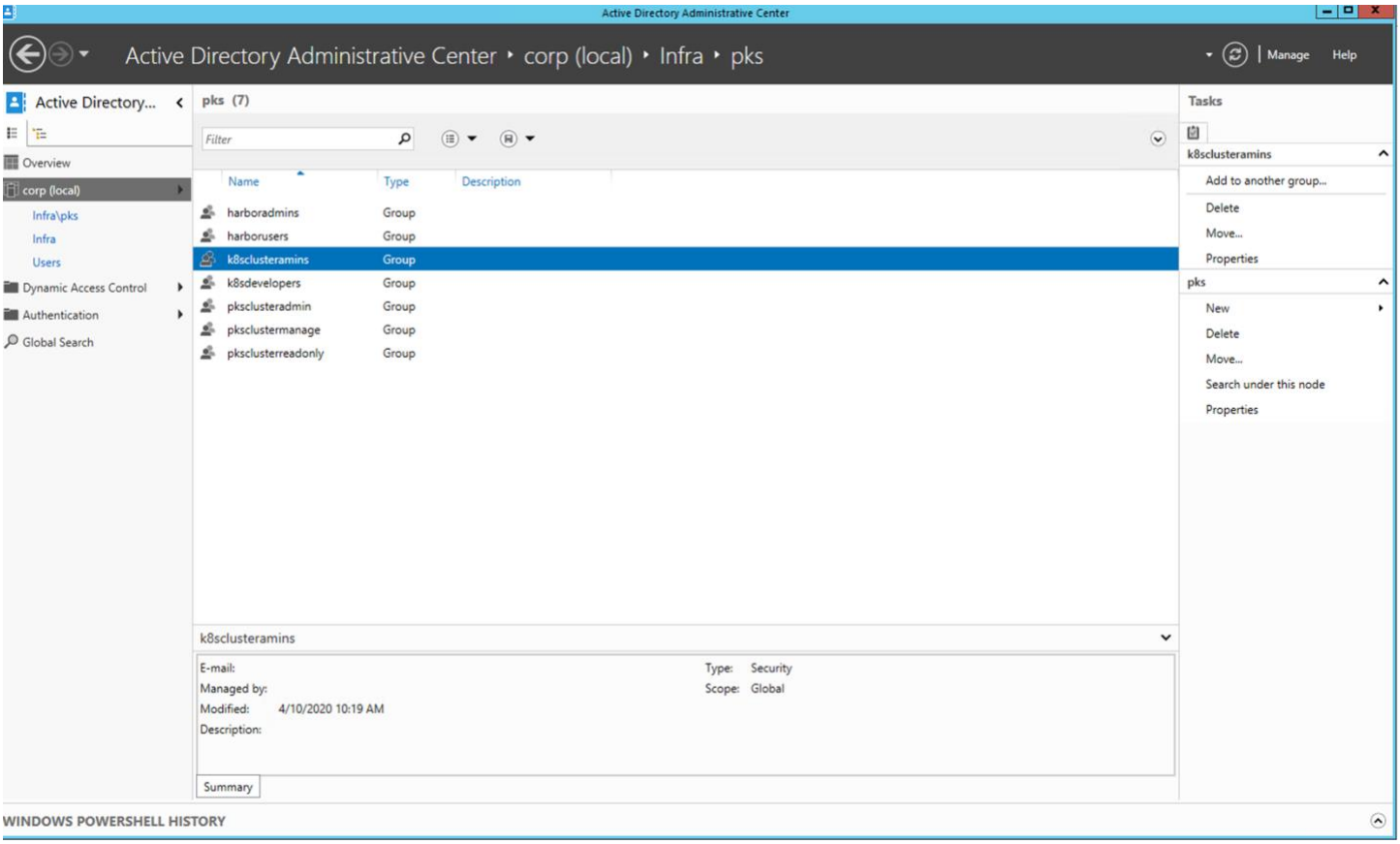


Step 4: Create Groups under pks – Select pks and click on New Group

Step 5: Create the pksclusteradmin group



Step 6: Create the other groups under the pks OU – pksclustermanage, pksclusterread, k8clusteradmins, k8developers, harboradmins, harborusers



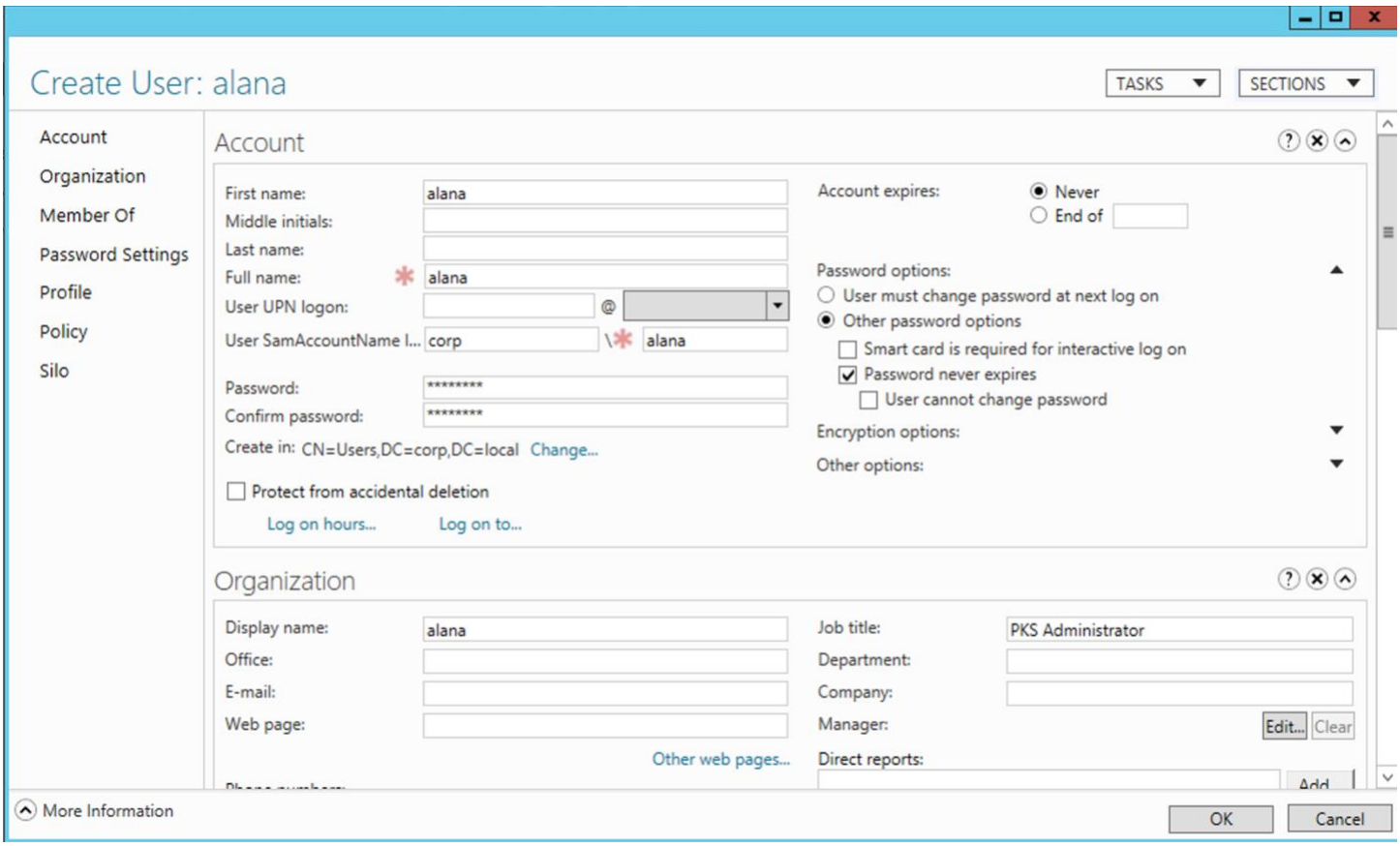
Create Users

Create users under each BU as per the following table

Step 1: Create the following users under CN=Users,DC=corp,DC=local

NOTE: You could create your own org structure for users.

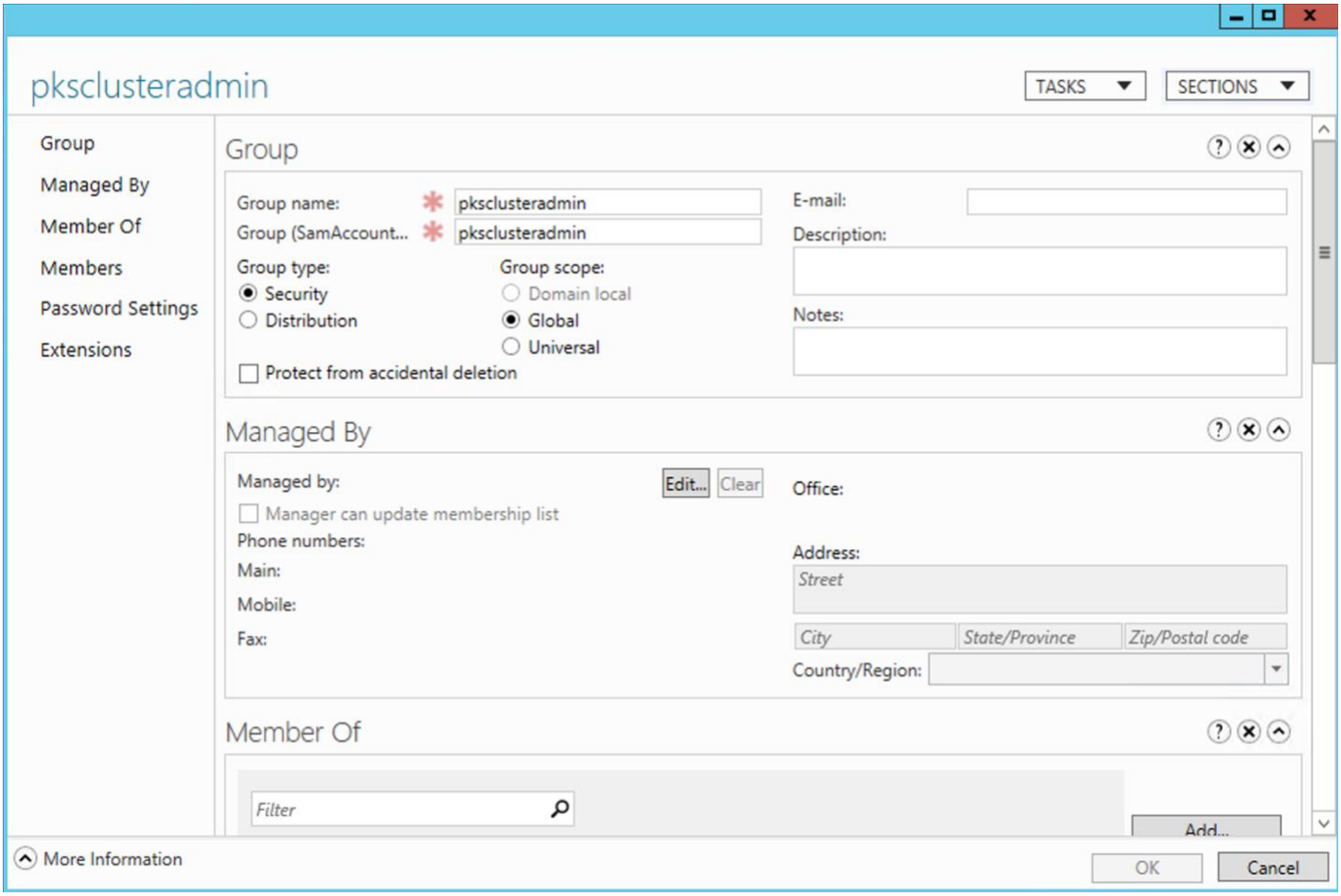
Step 2: Enter 'alana' for both Full Name and User SamAccountName, Select the option of password never expires and enter a password (E.g. <password>)



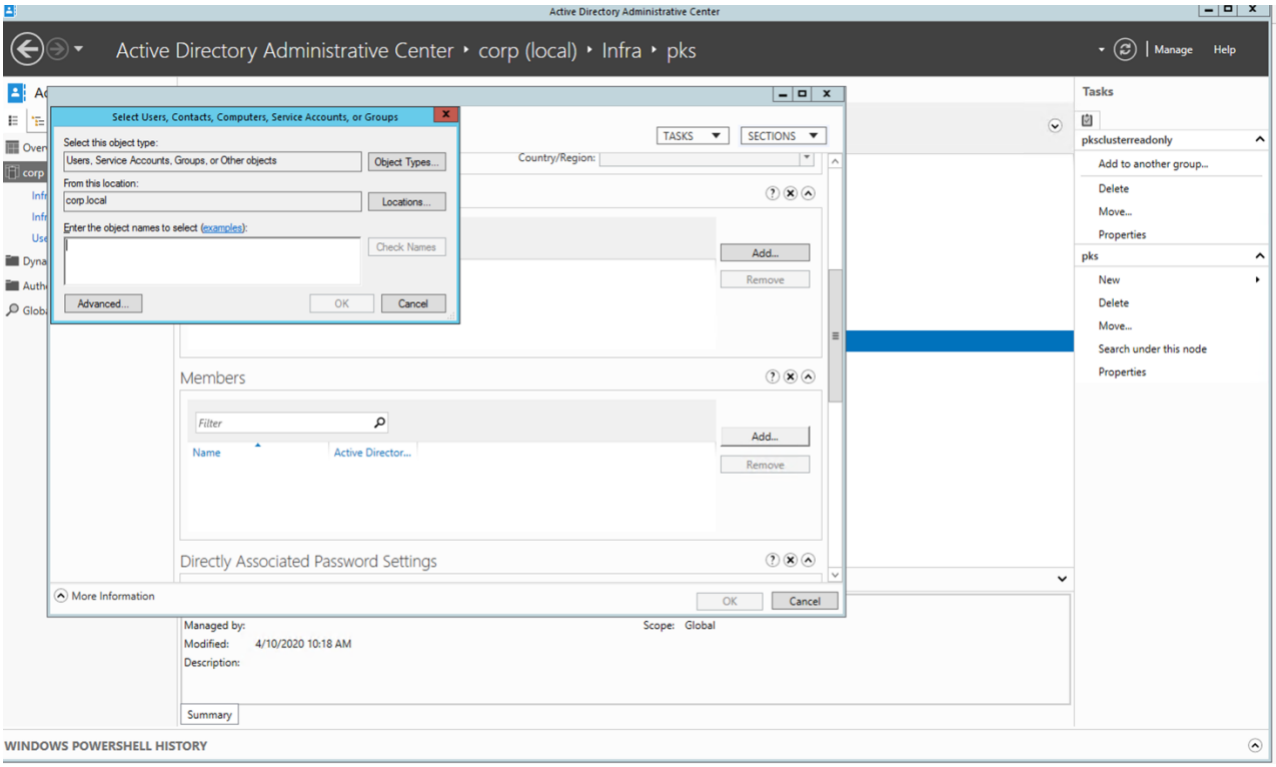
Step 3: Repeat the above steps to create the other personas cody, naomi, scott,

Assign Users to Groups

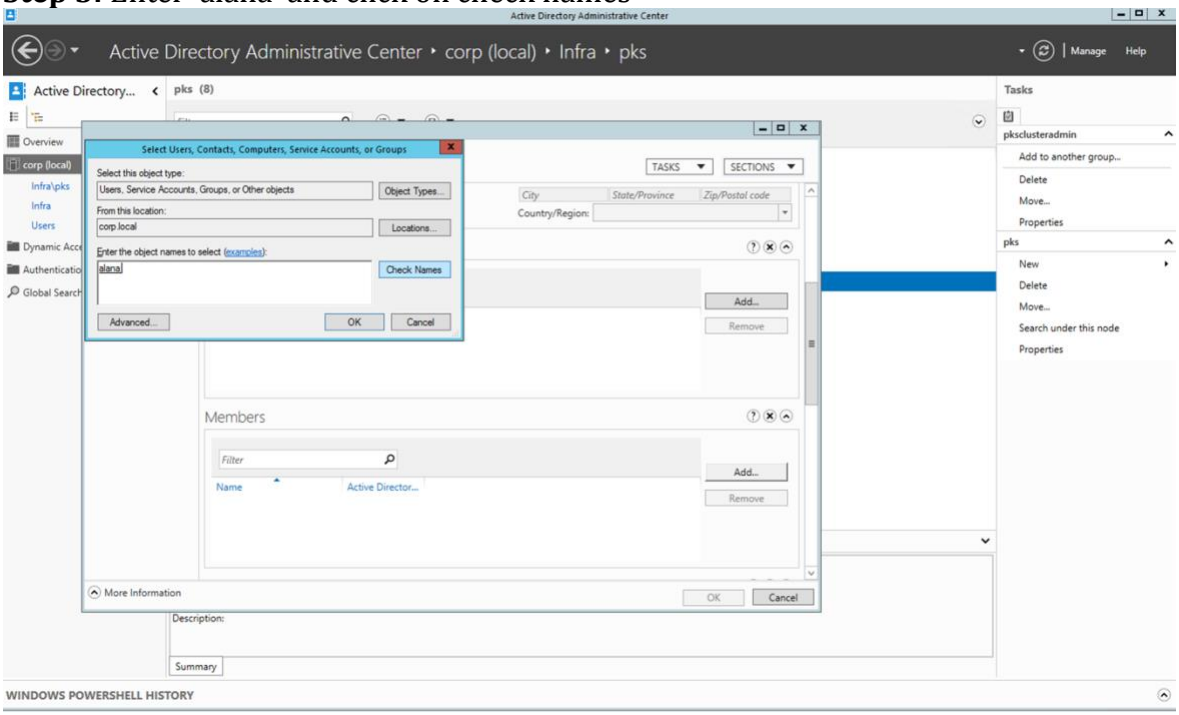
Step 1: Drill down to Infra/pks/pksclusteradmin group



Step 2: Under members click on Add



Step 3: Enter 'alana' and click on check names



Step 4: Click on OK

Step 5: Select the appropriate group and add users as per the table

Group	Persona
pksclusteradmin	alana
pksclustermanage	cody
pksclusterread	naomi scott
k8clusteradmins	naomi
k8developers	scott
harboradmins	cody
harborusers	scott

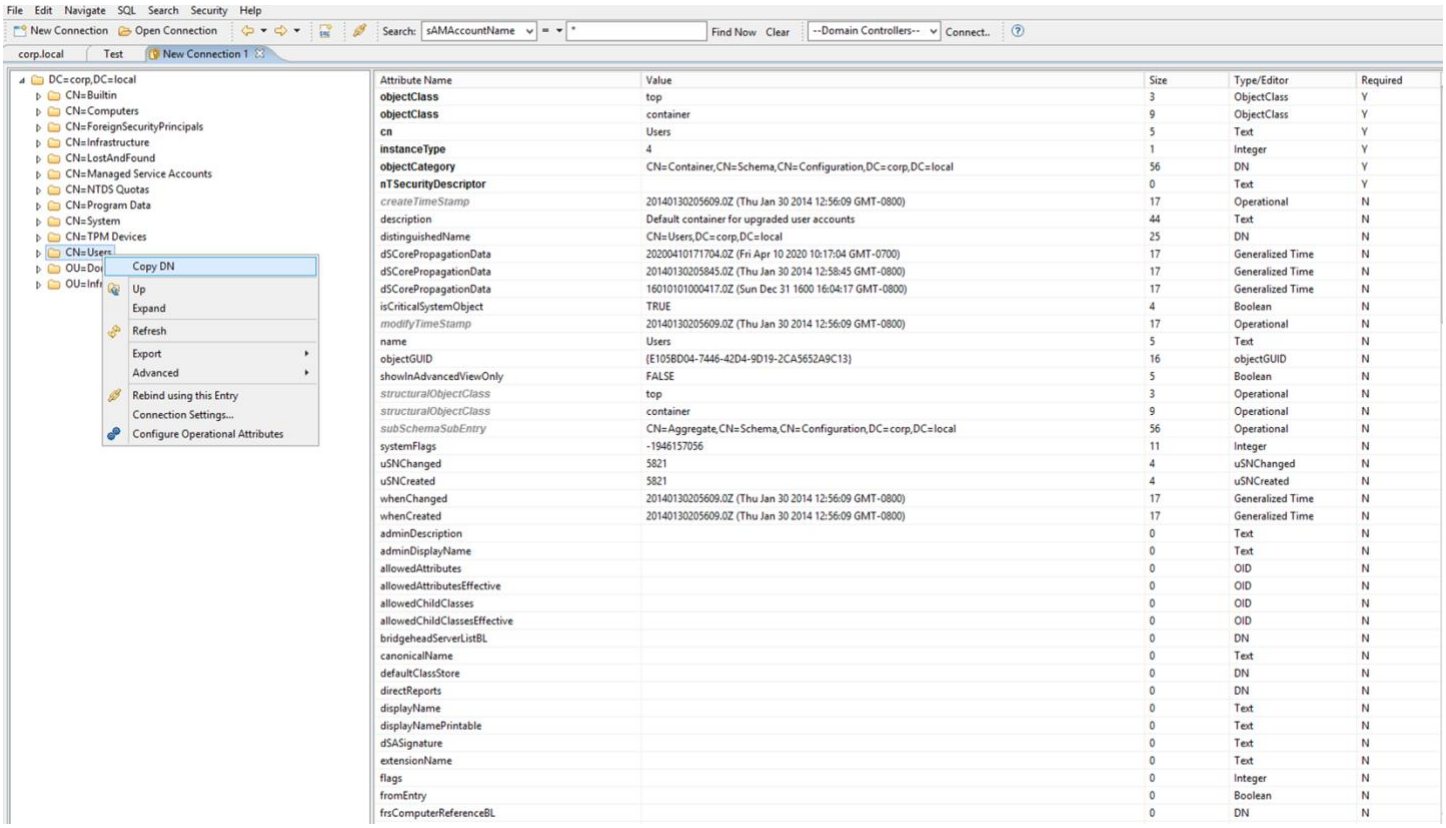
PKS LDAP Attributes

This section covers the basic LDAP attributes required to configure PKS for LDAP integration. Use an LDAP Browser like LDAPSoft AD Browser if not familiar with how to get values for some of these attributes.

LDAP Username : CN=Administrator,CN=Users,DC=corp,DC=local

User Search Base:

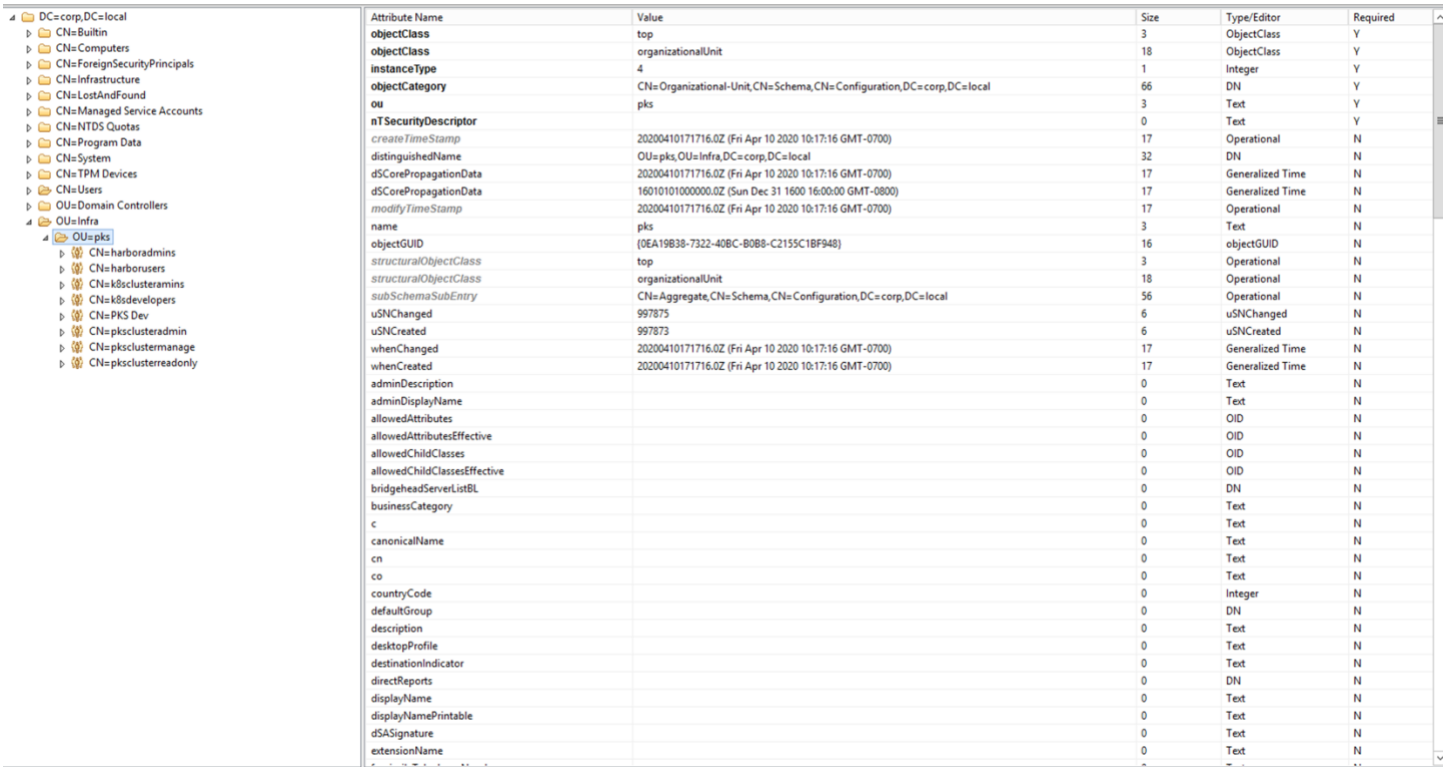
Select 'Users', right click and Copy DN
CN=Users,DC=corp,DC=local


File Edit Navigate SQL Search Security Help
New Connection Open Connection
Search: sAMAccountName = *
Find Now Clear
--Domain Controllers--
Connect..
corp.local
Test
New Connection 1
DC=corp,DC=local
CN=Builtin
CN=Computers
CN=ForeignSecurityPrincipals
CN=Infrastructure
CN=LostAndFound
CN=Managed Service Accounts
CN=NTDS Quotas
CN=Program Data
CN=System
CN=TPM Devices
CN=Users
OU=Do
OU=Infr
Copy DN
Up
Expand
Refresh
Export
Advanced
Rebind using this Entry
Connection Settings...
Configure Operational Attributes
Attribute Name
Value
Size
Type/Editor
Required
objectClass
top
3
ObjectClass
Y
objectClass
container
9
ObjectClass
Y
cn
Users
5
Text
Y
instanceType
4
1
Integer
Y
objectCategory
CN=Container,CN=Schema,CN=Configuration,DC=corp,DC=local
56
DN
Y
nTSecurityDescriptor
0
Text
Y
createTimeStamp
20140130205609.0Z (Thu Jan 30 2014 12:56:09 GMT-0800)
17
Operational
N
description
Default container for upgraded user accounts
44
Text
N
distinguishedName
CN=Users,DC=corp,DC=local
25
DN
N
dSCorePropagationData
20200410171704.0Z (Fri Apr 10 2020 10:17:04 GMT-0700)
17
Generalized Time
N
dSCorePropagationData
20140130205845.0Z (Thu Jan 30 2014 12:58:45 GMT-0800)
17
Generalized Time
N
dSCorePropagationData
16010101000417.0Z (Sun Dec 31 1600 16:04:17 GMT-0800)
17
Generalized Time
N
isCriticalSystemObject
TRUE
4
Boolean
N
modifyTimeStamp
20140130205609.0Z (Thu Jan 30 2014 12:56:09 GMT-0800)
17
Operational
N
name
Users
5
Text
N
objectGUID
{E105BD04-7446-42D4-9D19-2CA5652A9C13}
16
objectGUID
N
showInAdvancedViewOnly
FALSE
5
Boolean
N
structuralObjectClass
top
3
Operational
N
structuralObjectClass
container
9
Operational
N
subSchemaSubEntry
CN=Aggregate,CN=Schema,CN=Configuration,DC=corp,DC=local
56
Operational
N
systemFlags
-1946157056
11
Integer
N
uSNChanged
5821
4
uSNChanged
N
uSNCreated
5821
4
uSNCreated
N
whenChanged
20140130205609.0Z (Thu Jan 30 2014 12:56:09 GMT-0800)
17
Generalized Time
N
whenCreated
20140130205609.0Z (Thu Jan 30 2014 12:56:09 GMT-0800)
17
Generalized Time
N
adminDescription
0
Text
N
adminDisplayName
0
Text
N
allowedAttributes
0
OID
N
allowedAttributesEffective
0
OID
N
allowedChildClasses
0
OID
N
allowedChildClassesEffective
0
OID
N
bridgeheadServerListBL
0
DN
N
canonicalName
0
Text
N
defaultClassStore
0
DN
N
directReports
0
DN
N
displayName
0
Text
N
displayNamePrintable
0
Text
N
dSASignature
0
Text
N
extensionName
0
Text
N
flags
0
Integer
N
fromEntry
0
Boolean
N
frsComputerReferenceBL
0
DN
N

Group Search Base

The groups definitions are created under

OU=pks,OU=Infra,DC=corp,DC=local

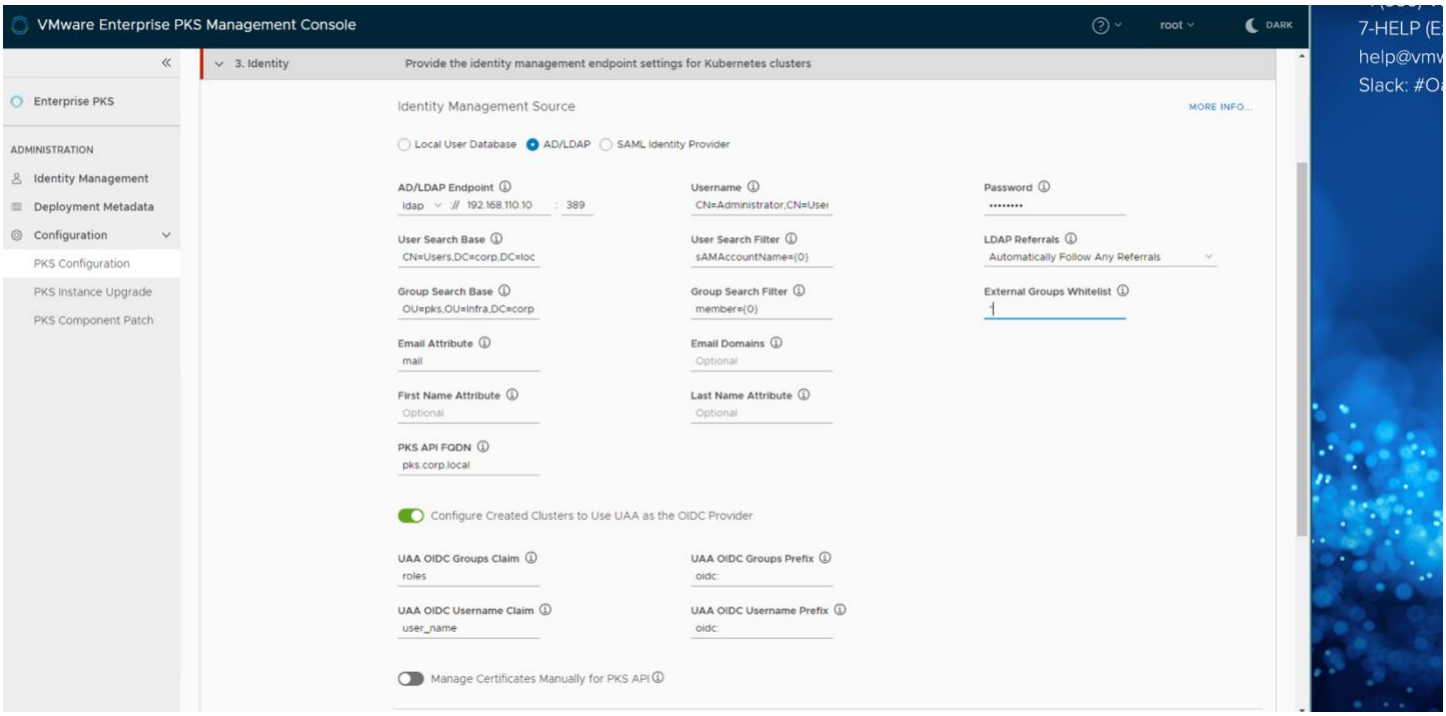


Attribute Name	Value	Size	Type/Editor	Required
objectClass	top	3	ObjectClass	Y
objectClass	organizationalUnit	18	ObjectClass	Y
instanceType	4	1	Integer	Y
objectCategory	CN=Organizational-Unit,CN=Schema,CN=Configuration,DC=corp,DC=local	66	DN	Y
ou	pks	3	Text	Y
nTSecurityDescriptor		0	Text	Y
createTimeStamp	20200410171716.0Z (Fri Apr 10 2020 10:17:16 GMT-0700)	17	Operational	N
distinguishedName	OU=pks,OU=Infra,DC=corp,DC=local	32	DN	N
dSCorePropagationData	20200410171716.0Z (Fri Apr 10 2020 10:17:16 GMT-0700)	17	Generalized Time	N
dSCorePropagationData	16010101000000.0Z (Sun Dec 31 1600 16:00:00 GMT-0800)	17	Generalized Time	N
modifyTimeStamp	20200410171716.0Z (Fri Apr 10 2020 10:17:16 GMT-0700)	17	Operational	N
name	pks	3	Text	N
objectGUID	{0EA19B38-7322-40BC-B0B8-C2155C1BF948}	16	objectGUID	N
structuralObjectClass	top	3	Operational	N
structuralObjectClass	organizationalUnit	18	Operational	N
subSchemaSubEntry	CN=Aggregate,CN=Schema,CN=Configuration,DC=corp,DC=local	56	Operational	N
uSNChanged	997875	6	uSNChanged	N
uSNCreated	997873	6	uSNCreated	N
whenChanged	20200410171716.0Z (Fri Apr 10 2020 10:17:16 GMT-0700)	17	Generalized Time	N
whenCreated	20200410171716.0Z (Fri Apr 10 2020 10:17:16 GMT-0700)	17	Generalized Time	N
adminDescription		0	Text	N
adminDisplayName		0	Text	N
allowedAttributes		0	OID	N
allowedAttributesEffective		0	OID	N
allowedChildClasses		0	OID	N
allowedChildClassesEffective		0	OID	N
bridgeheadServerListBL		0	DN	N
businessCategory		0	Text	N
c		0	Text	N
canonicalName		0	Text	N
cn		0	Text	N
co		0	Text	N
countryCode		0	Integer	N
defaultGroup		0	DN	N
description		0	Text	N
desktopProfile		0	Text	N
destinationIndicator		0	Text	N
directReports		0	DN	N
displayName		0	Text	N
displayNamePrintable		0	Text	N
dSASignature		0	Text	N
extensionName		0	Text	N

EPMC LDAP settings

This section shows the configuration settings on EPMC to install PKS. To configure LDAP through EPMC, select AD/LDAP as the Identity management Service during PKS installation through EPMC.

NOTE: Since EPMC is the management console, we would require to only configure LDAP for PKS. In 'traditional' (tiles based) deployments LDAP would need to be configured for OpsMan as well.

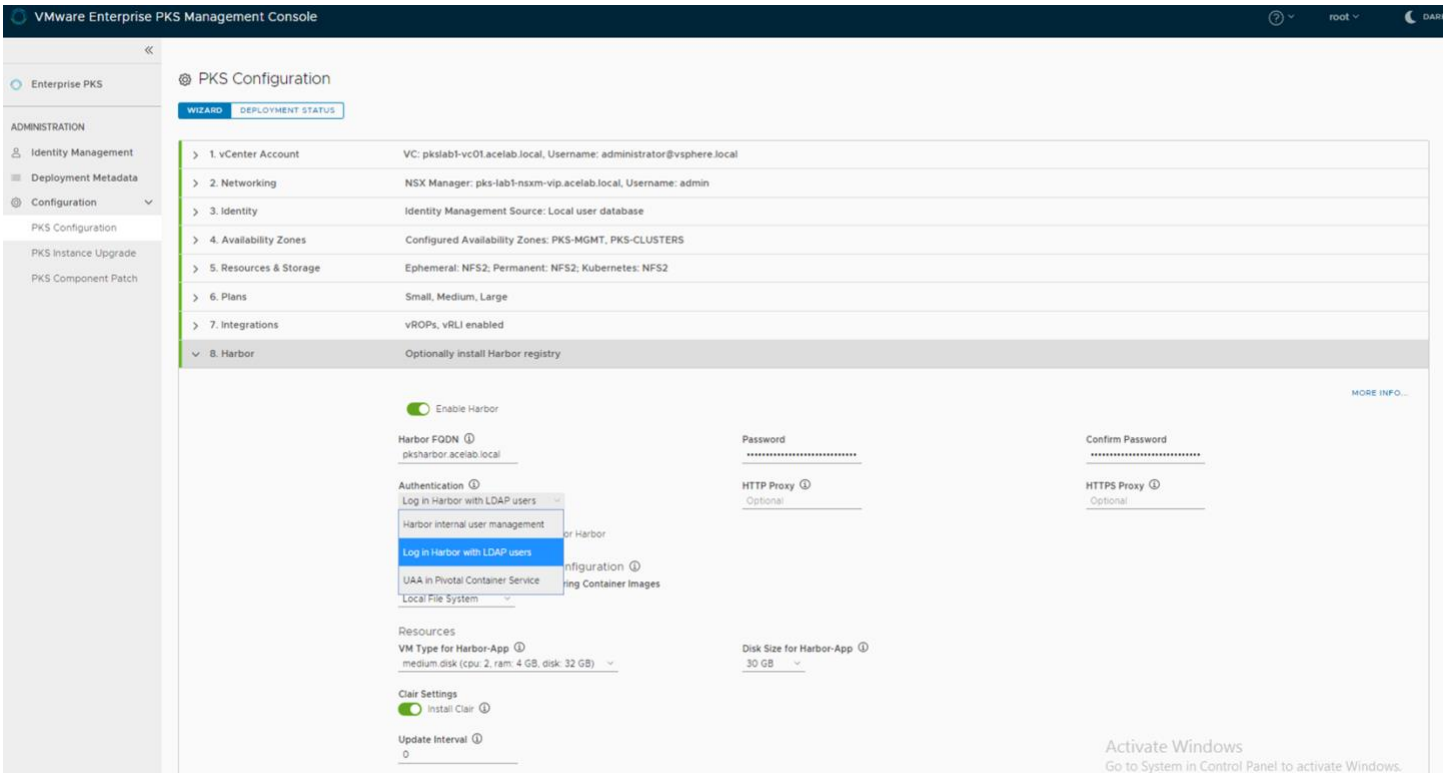


Example LDAP settings provided to EPMC. Please update with your values and use during installation.

Description	Value
Server URL (Only takes an IP)	ldap://192.168.110.10
LDAP Username	CN=Administrator,CN=Users,DC=corp,DC=local
LDAP Password	<password>
User Search Base	CN=Users,DC=corp,DC=local
User Search Filter	sAMAccountName={0}
Group Search Base	OU=pks,OU=Infra,DC=corp,DC=local
Email	mail

- Configure Created Clusters to use UAA as the OIDC provider – Enabled
- Once LDAP is configured generate configuration and apply to deploy and configure PKS with LDAP

For Harbor LDAP integration on the Harbor section of EPMC, select Log in Harbor with LDAP users.



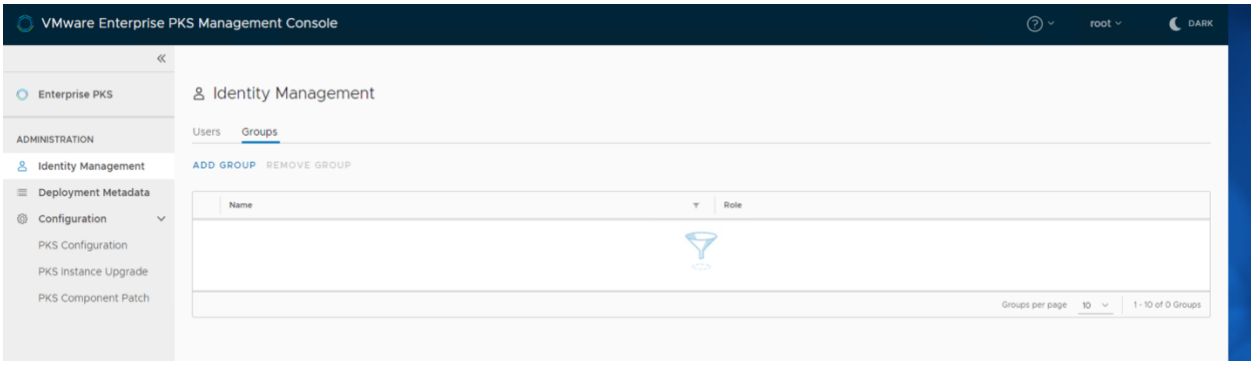
NOTE: If this is not selected in the initial install, the Harbor tile will need to be deleted through Opsman and redeployed with the LDAP integration enabled.

A UAA admin user can assign the following UAA scopes to Enterprise PKS users

pks.clusters.manage: Accounts with this scope can create and access their own clusters.
pks.clusters.admin: Accounts with this scope can create and access all clusters.
pks.clusters.admin.read: Accounts with this scope can access any information about all clusters except for cluster credentials.

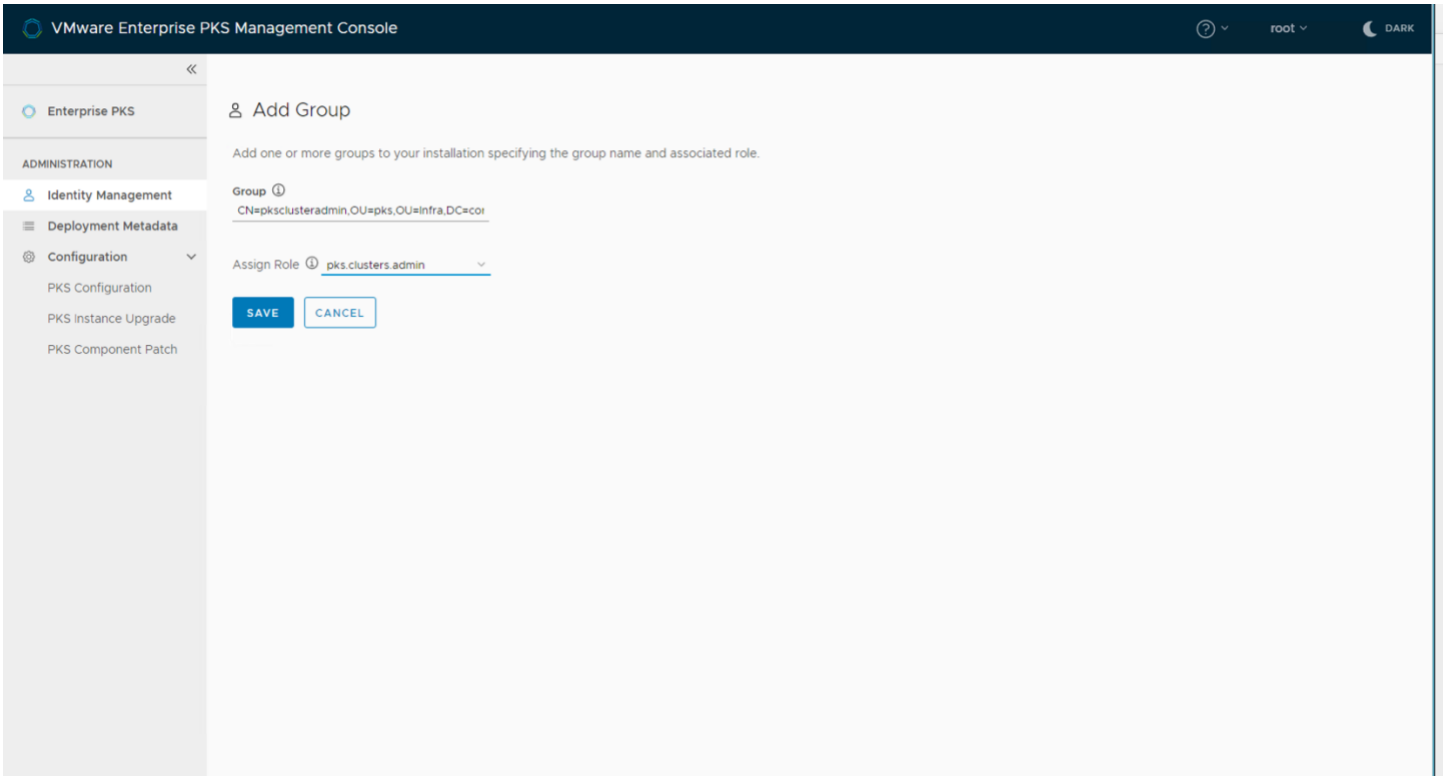
Configure Users and Groups from EPMC

Once the deployment is complete, click on Identity Management on the EPMC Console and select the groups tab



Map LDAP Group to PKS Admin roles

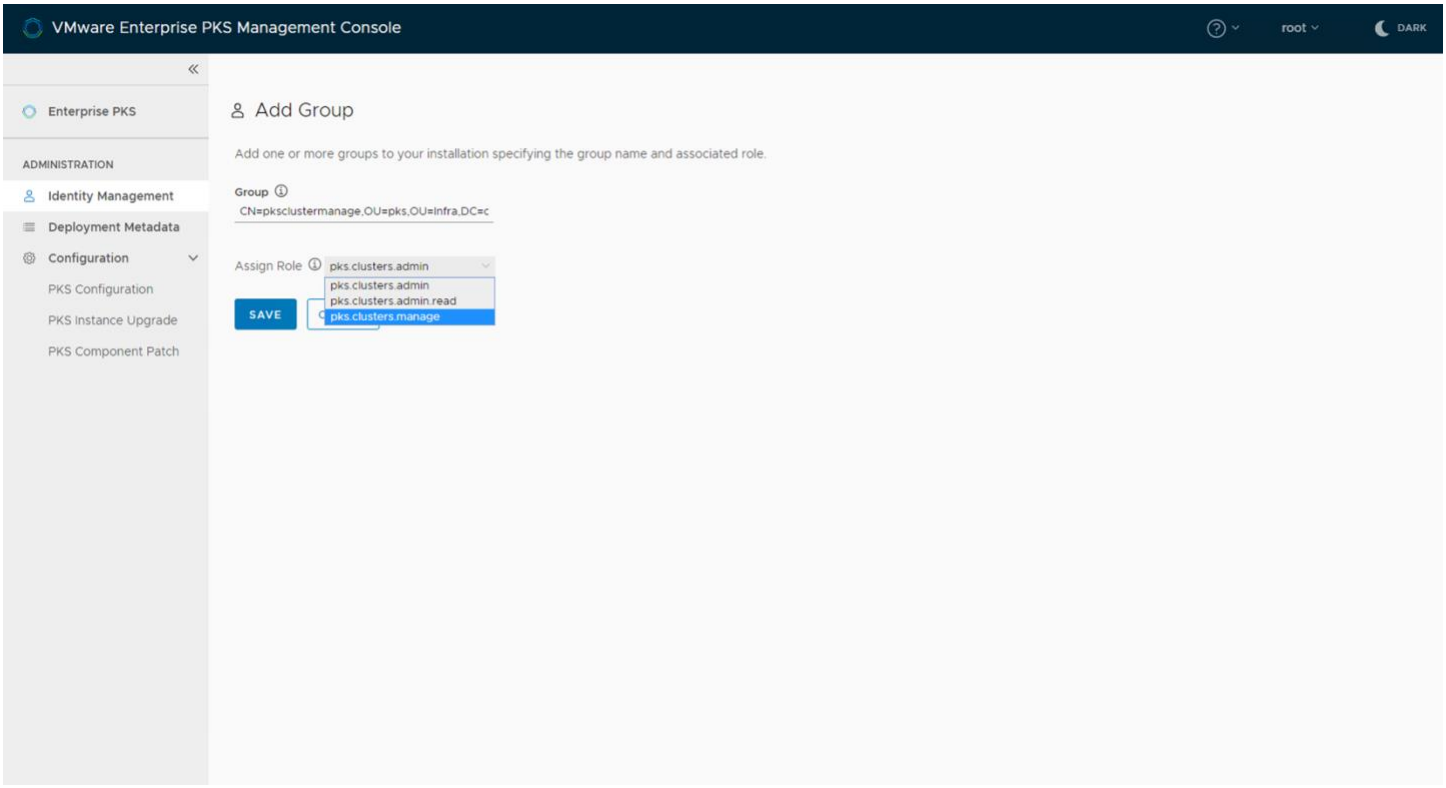
Click on Add Group



Group Name - DN of the group that need to be added with pks admin roles - CN=pksclusteradmin,OU=pks,OU=Infra,DC=corp,DC=local

Map LDAP Group to PKS Manage roles

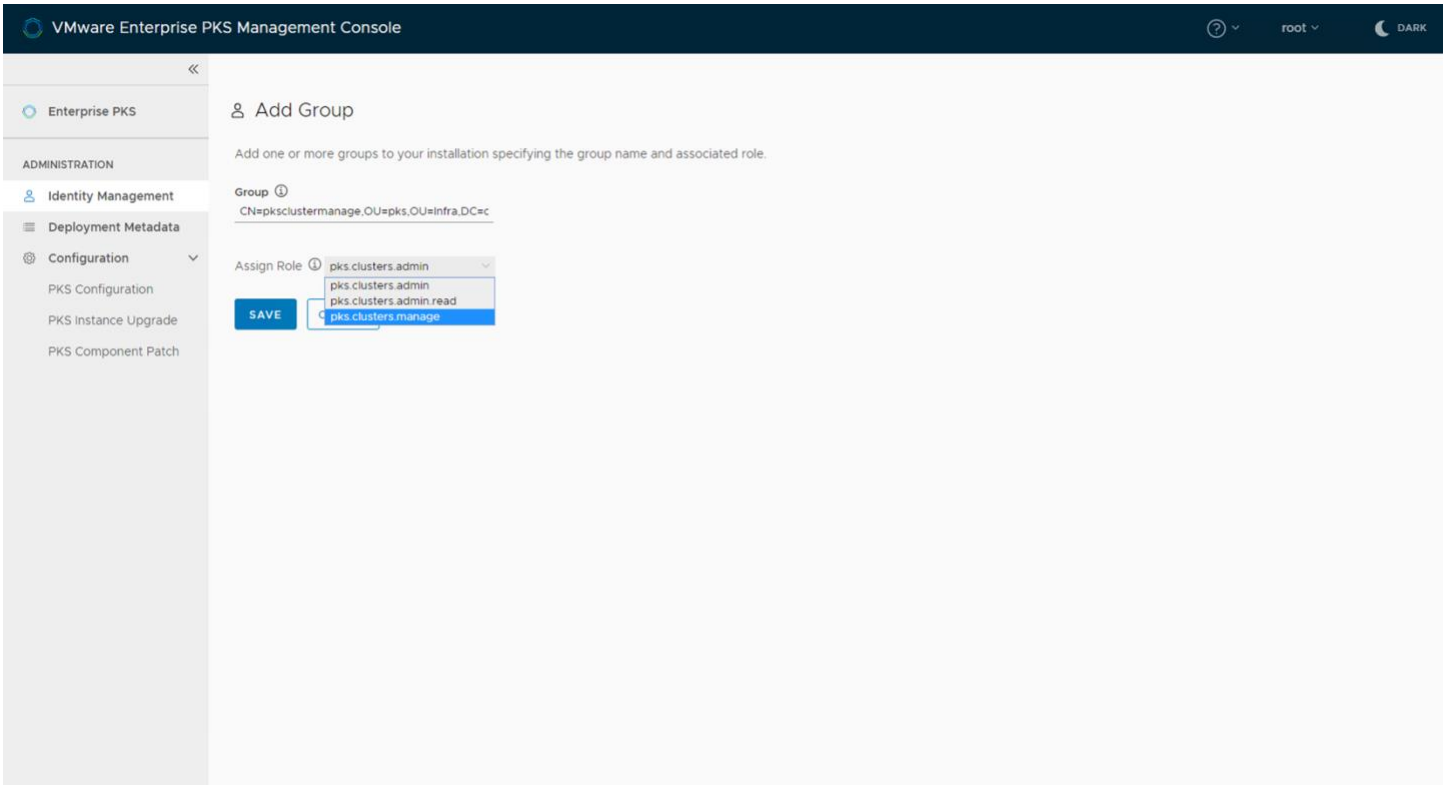
Click on Add Group, select pks.cluster.manage as role



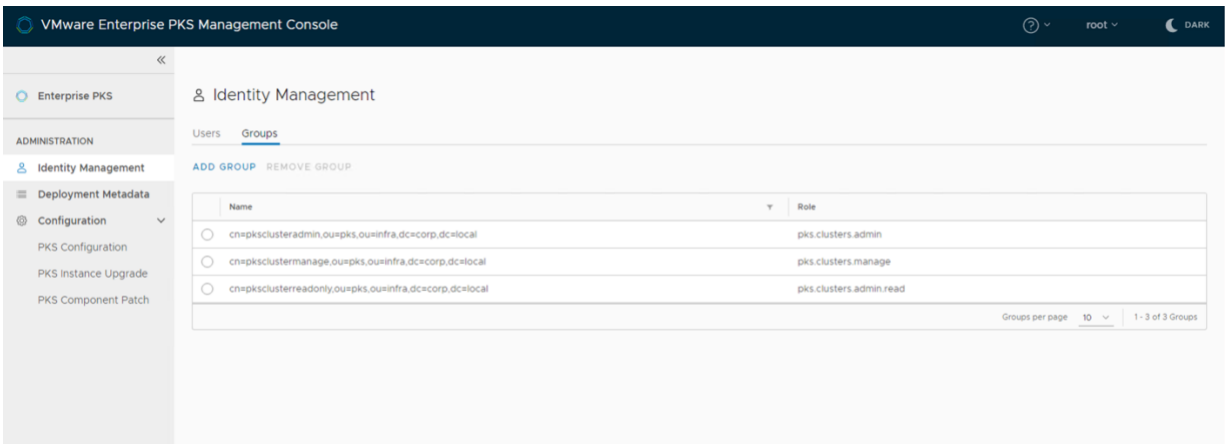
Group Name - DN of the group that need to be added with pks manage roles. - CN=pksclustermanage,OU=pks,OU=Infra,DC=corp,DC=local

Map LDAP Group to PKS Read roles

Click on Add Group, select pks.cluster.read as role



Group Name - DN of the group that need to be added with pks manage roles. - CN=pksclusterreadonly,OU=pks,OU=Infra,DC=corp,DC=local



Login to a PKS (Revisit)

pks login -a pks.corp.local -u <user-in-ldap> --skip-ssl-validation

Login to pks as alana and create a cluster, this should be possible

pks login -a pks.corp.local -u alana -k

Login to pks as cody and create a cluster, should be possible but cody
pks login -a pks.corp.local -u cody -k

Login to pks as scott or naomi. The k8 developers will only be able to view clusters in the system but not do any admin tasks on the clusters.

NON EPMC DEPLOYMENT

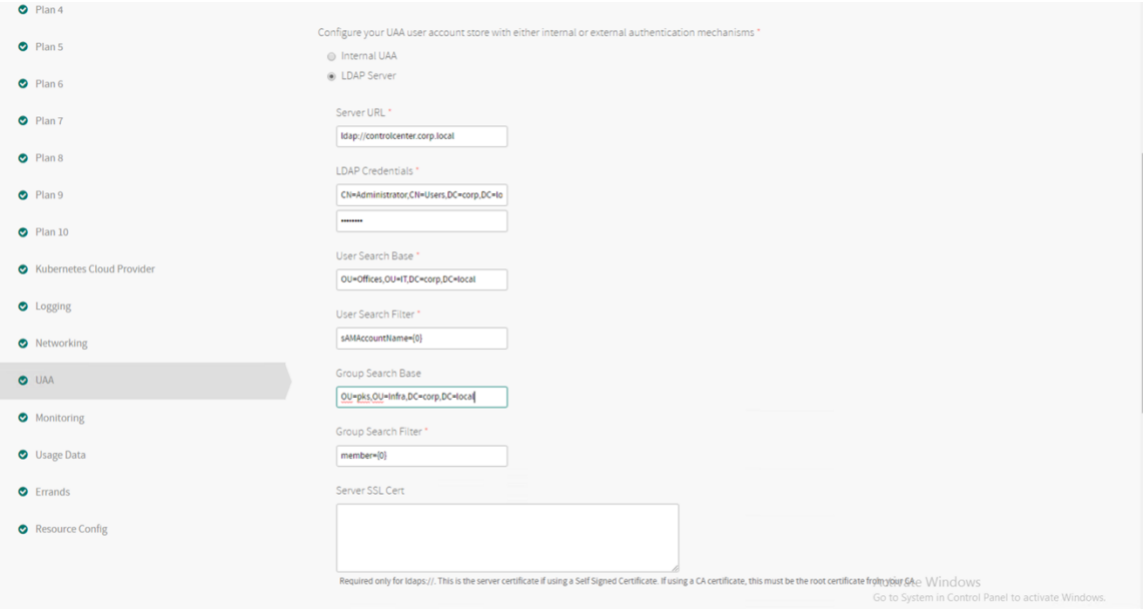
In a non EPMC deployment each product on the installation requires to be separately configured to use LDAP.

PKS LDAP Integration

The following section covers LDAP integration on the PKS tile:

- Log into OpsMan and select the Enterprise PKS Tile
- Select the UAA section and configure the following

Note: Enable UAA as OIDC Provider



Example LDAP settings provided to EPMC. Please update with your values and use during installation.

Description	Values
Current Decryption Password	<password>
Server URL	ldap://controlcenter.corp.local
LDAP Username	CN=Administrator,CN=Users,DC=corp,DC=local
LDAP Password	<password>
User Search Base	CN=Users,DC=corp,DC=local
User Search Filter	sAMAccountName={0}
Group Search Base	OU=pks,OU=Infra,DC=corp,DC=local

Group Search Filter	Member{0}

Configuring Groups in UAA

A UAA admin user can assign the following UAA scopes to Enterprise PKS users

pks.clusters.manage: Accounts with this scope can create and access their own clusters.
pks.clusters.admin: Accounts with this scope can create and access all clusters.
pks.clusters.admin.read: Accounts with this scope can access any information about all clusters except for cluster credentials.

To grant Enterprise PKS access to an external LDAP group, perform the following steps:

1. Log in as the UAA admin using the procedure in Log in as a UAA Admin.

2. To assign the pks.clusters.manage scope to all users in an LDAP group, run the following command:

   uaac group map --name pks.clusters.manage GROUP-DISTINGUISHED-NAME

   Where GROUP-DISTINGUISHED-NAME is the LDAP Distinguished Name (DN) for the group.
   For example:
   uaac group map --name pks.clusters.manage cn=pksclustermanage,ou=pks,ou=infra,dc=corp,dc=local

   For more information about LDAP DNs, see the LDAP DNs and RDNs in the LDAP documentation.

3. To assign the pks.clusters.admin scope to all users in an LDAP group, run the following command:
   uaac group map --name pks.clusters.admin GROUP-DISTINGUISHED-NAME

   Where GROUP-DISTINGUISHED-NAME is the LDAP Distinguished Name (DN) for the group.
   For example:
   uaac group map --name pks.clusters.admin cn=pksclusteradmin,ou=pks,ou=infra,dc=corp,dc=local

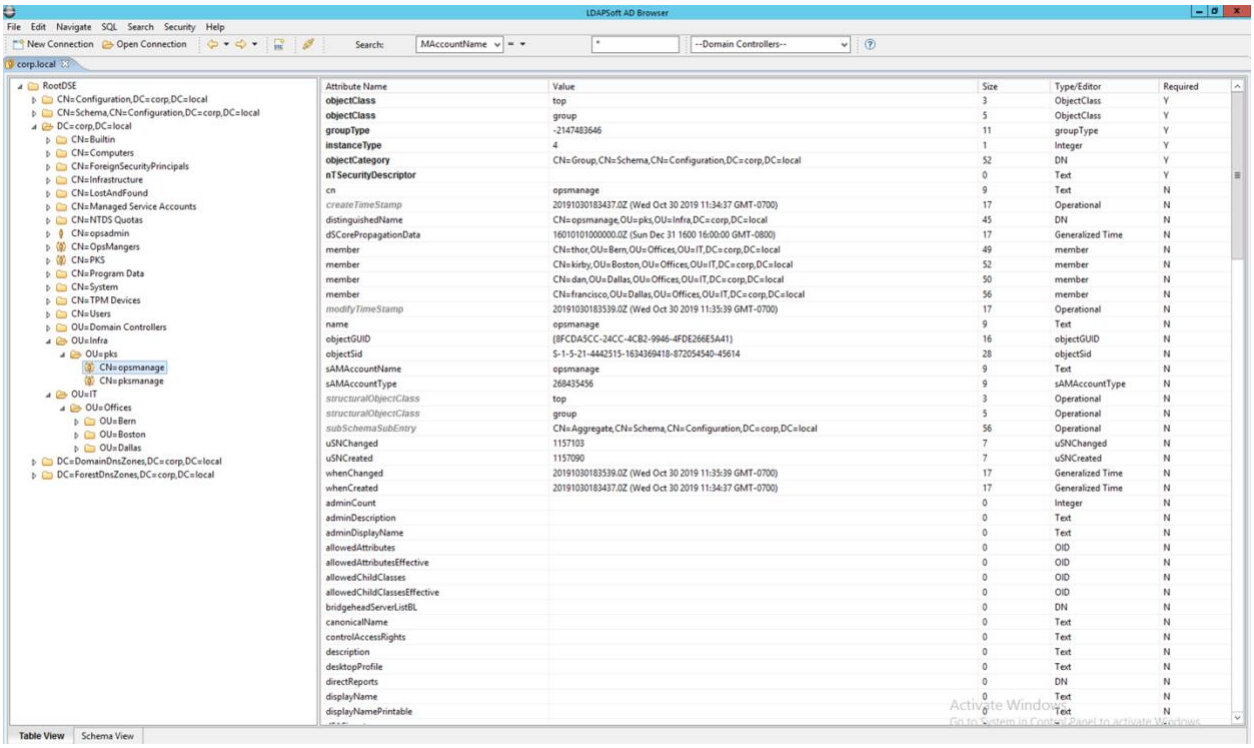
4. To assign the pks.clusters.read scope to all users in an LDAP group, run the following command:
   uaac group map --name pks.clusters.read GROUP-DISTINGUISHED-NAME

   Where GROUP-DISTINGUISHED-NAME is the LDAP Distinguished Name (DN) for the group.
   For example:
   uaac group map --name pks.clusters.read cn=pksclusterreadonly,ou=pks,ou=infra,dc=corp,dc=local

OpsMan LDAP Integration

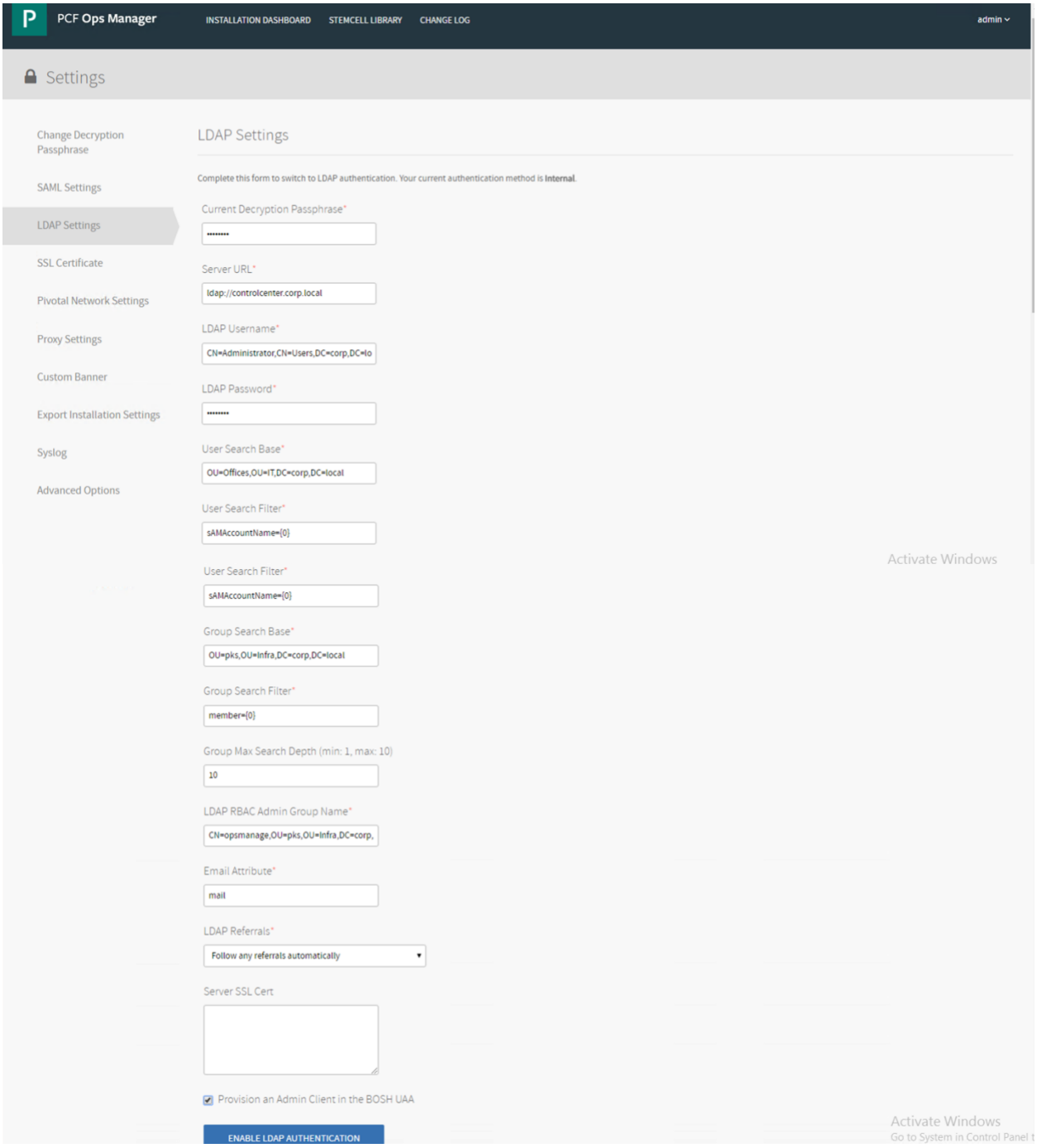
OpsMan LDAP integration requires an OpsMan RBAC group, which we had configured when setting up LDAP. Copy the DN to this group similar to what was done for the 'pksadmin' group.

OpsMan RBAC group



OpsMan Tile LDAP settings

Access the OpsMan configuration pane via Admin setting that can be found on the right corner



Description	Values
Current Decryption Password	<password>
Server URL	ldap://controlcenter.corp.local
LDAP Username	CN=Administrator,CN=Users,DC=corp,DC=local
LDAP Password	<password>
User Search Base	OU=Offices,OU=IT,DC=corp,DC=local
User Search Filter	sAMAccountName={0}
Group Search Base	OU=pks,OU=Infra,DC=corp,DC=local
Group Search Filter	Member{0}
Group Max Search Depth	10
LDAP RBAC Admin Group	CN=opsmanage,OU=pks,OU=Infra,DC=corp,DC=local The group where the opsman users are present

Azure Active Directory as a SAML Identity Endpoint

Given below are the detailed steps of configuring single-sign-on (SSO) between Azure Active Directory and VMware Enterprise PKS

These steps are found *here*

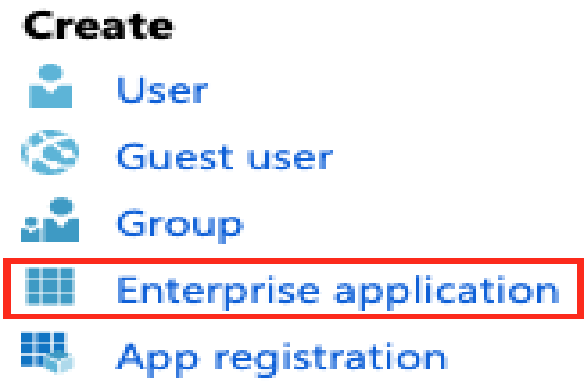
Prerequisites

To configure Azure AD to designate Enterprise PKS as a service provider, you must have an Azure AD Global Administrator account.

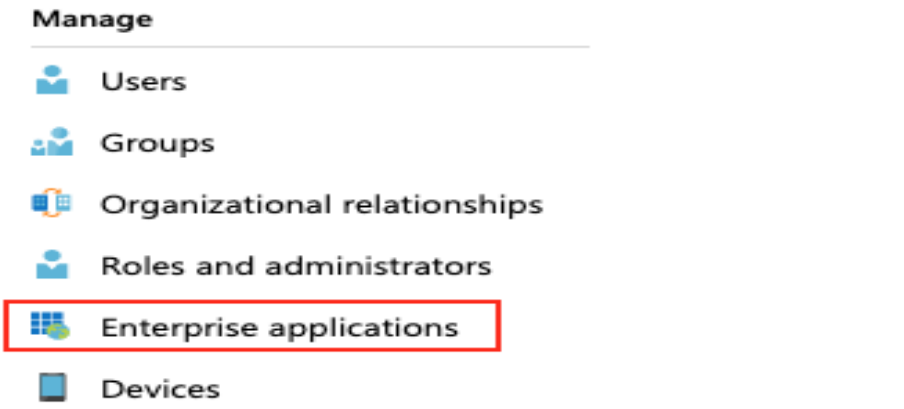
Configure SAML in Azure AD

To configure Azure AD as a SAML identity provider for Enterprise PKS, do the following:

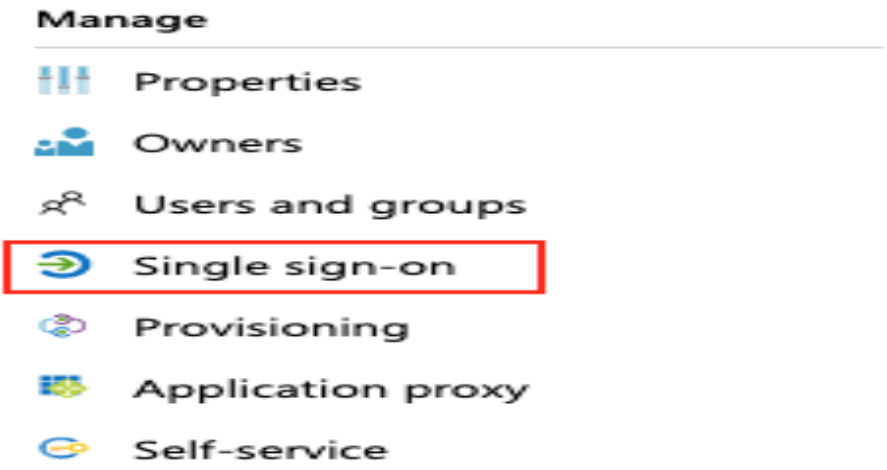
- Log in to Azure AD as a Global Administrator.
- Navigate to **Azure Active Directory**.
- Under **Create**, click **Enterprise application**.



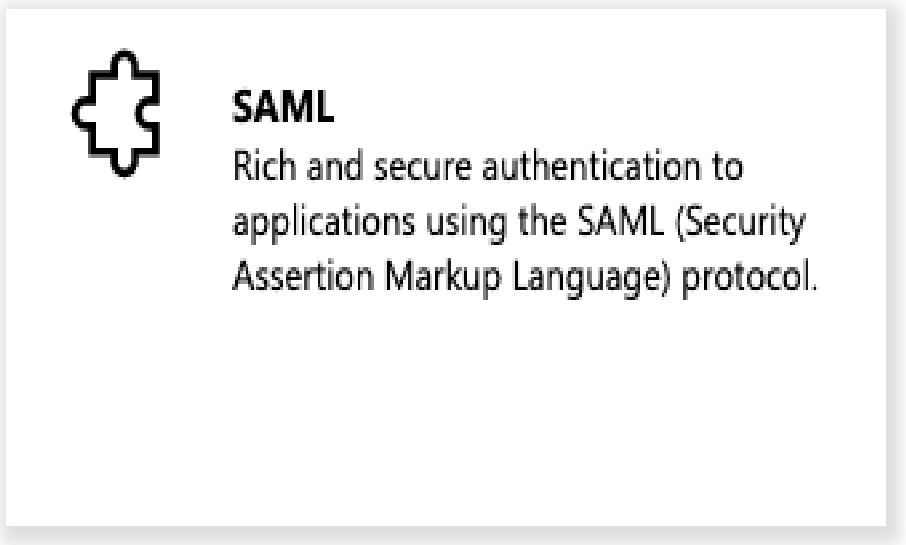
- Under Add your own app, select Non-gallery application.
- Enter a Name and click Add.
- Navigate to Azure Active Directory > Enterprise applications.



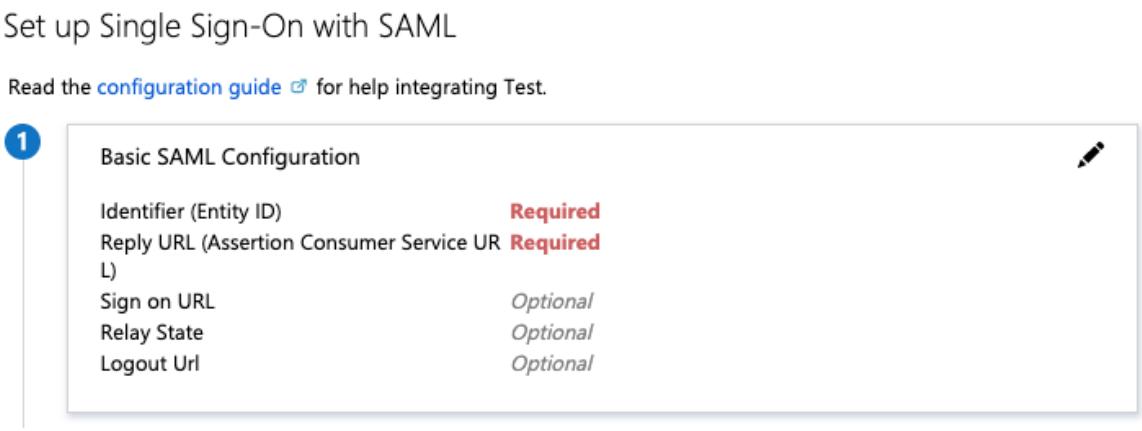
Click your app and then click Single sign-on.



- Under Select a single sign-on method, select SAML.



- Under Set up Single Sign-On with SAML, click the pencil icon for Basic SAML Configuration.

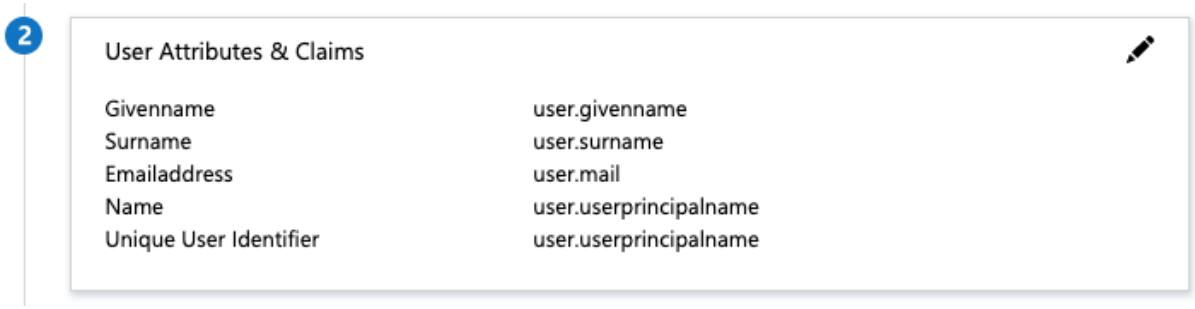


Configure as per table below

Description	Values
Identifier (Entity ID)	Enter PKS-API:8443. For example: api.pks.example.com:8443
Reply URL	Enter https://PKS-API:8443/saml/SSO/alias/PKS-API:8443. For example: https://api.pks.example.com:8443/saml/SSO/alias/api.pks.example.com:8443
Sign on URL	Enter https://PKS-API:8443/saml/SSO/alias/PKS-API:8443. For example: https://api.pks.example.com:8443/saml/SSO/alias/api.pks.example.com:8443
Identifier (Entity ID)	Enter PKS-API:8443. For example: api.pks.example.com:8443
Reply URL	Enter https://PKS-API:8443/saml/SSO/alias/PKS-API:8443. For example: https://api.pks.example.com:8443/saml/SSO/alias/api.pks.example.com:8443
Sign on URL	Enter https://PKS-API:8443/saml/SSO/alias/PKS-API:8443. For example: https://api.pks.example.com:8443/saml/SSO/alias/api.pks.example.com:8443

Note: VMware recommends that you use the default settings for the fields that are not referenced in the above table.

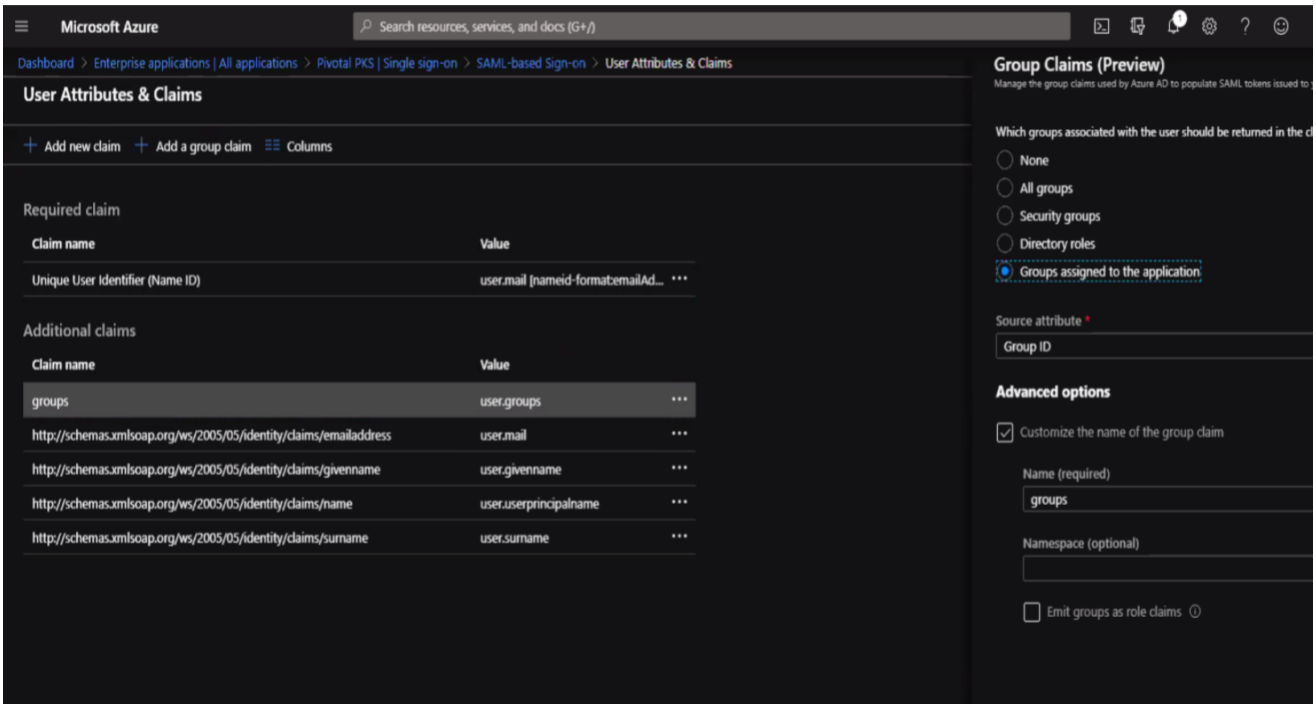
- Click the pencil icon for User Attributes &



Claims

- Configure your user attributes and claims by doing the procedures in

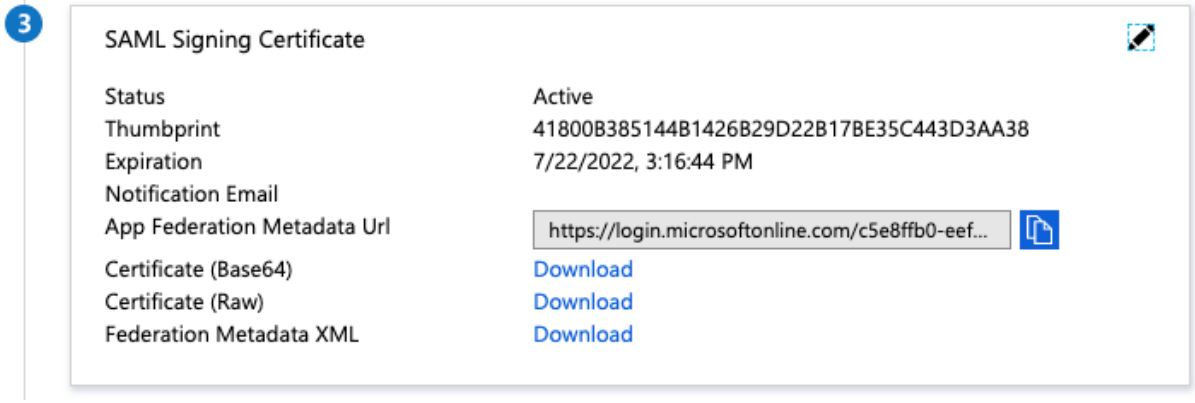
How to: Customize claims issued in the SAML token for enterprise applications in the Microsoft Azure documentation. By default, Enterprise PKS uses the EmailAddress name identifier format.



As shown above, do the following:

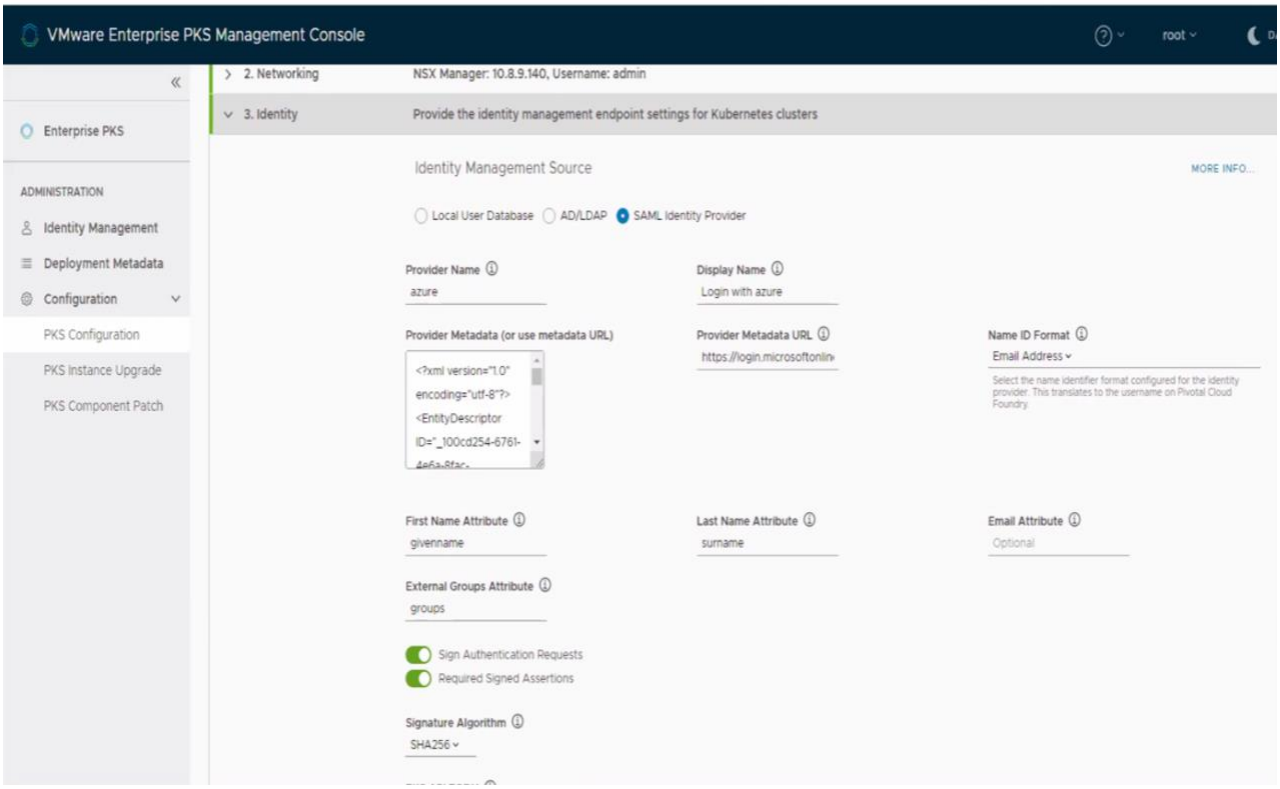
- Click “Add a group claim”
- Choose the option for Group Claims as “Groups assigned to the application”
- If you chose “Source Attribute” as GroupID, then you will need to map GroupID(s) of the group(s) to the right objectID (as shown later).
- Click on “Advanced Options” and “Customize the name of the group claim” with the Name “groups”.
- Save.

- Under SAML Signing Certificate, copy and save the link address for App Federation Metadata Url or download Federation Metadata XML. You use the Azure AD metadata to configure SAML in the Enterprise PKS tile. For more information, see the Configure SAML as an Identity Provider section of Installing Enterprise PKS on



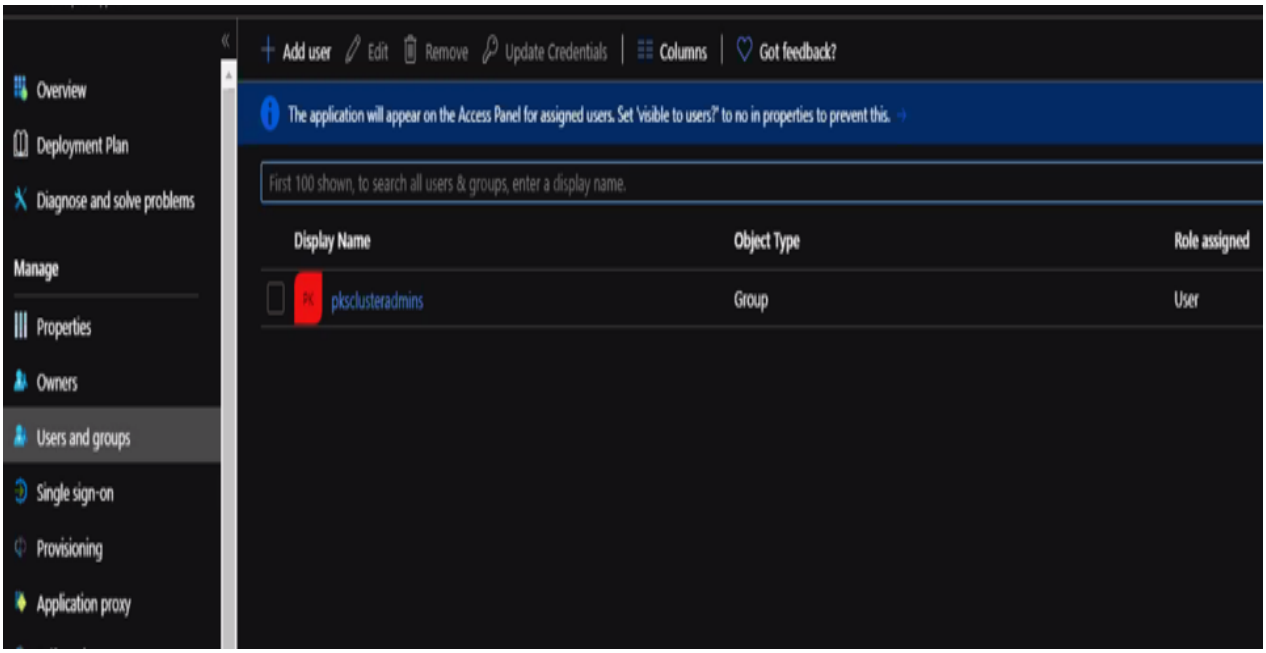
Azure.

- If using Enterprise PKS Management Console or Opsman's PKS Title, use the following options to configure SAML Identity provider. Make sure that "External Group Atribute" is set to "groups" as configured in Azure. Once you save this, you will need to update PKS.

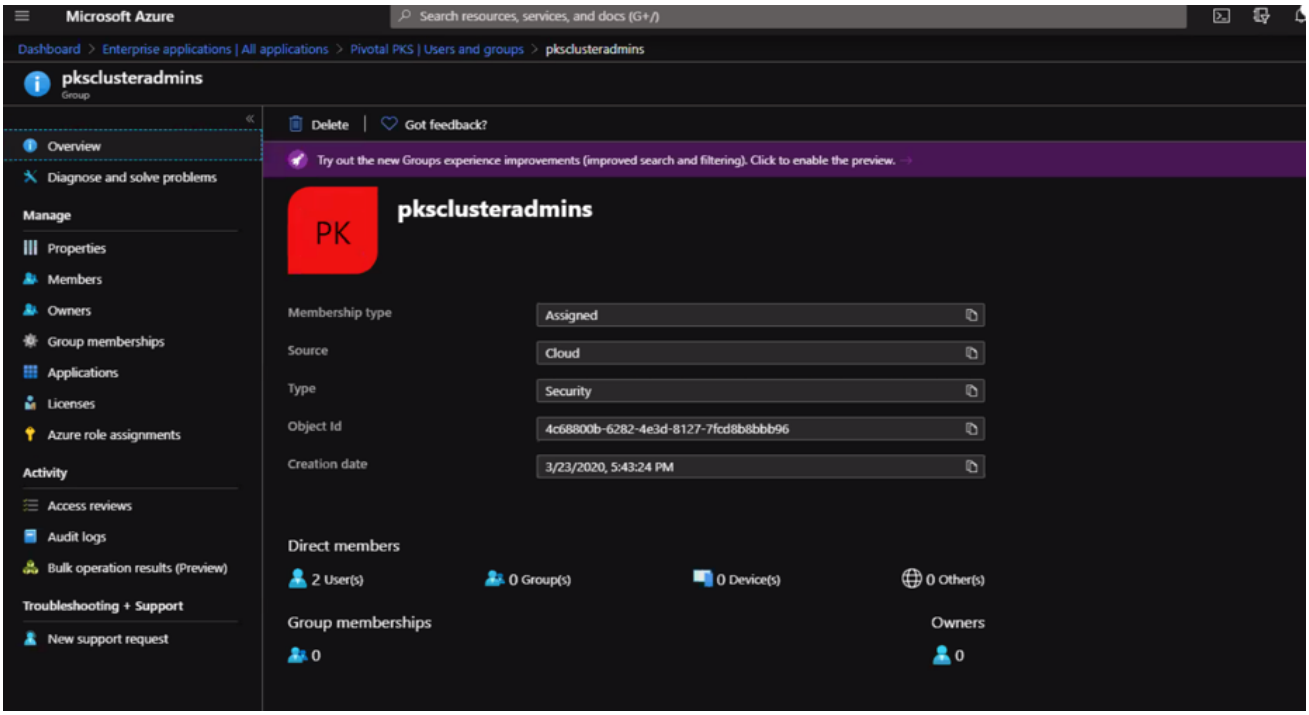


- Configure your group attributes and claims by doing the procedures in the Configure group claims for SAML applications using SSO configuration section of Configure group claims for applications with Azure Active Directory (Public Preview) in the Microsoft Azure documentation.

In the example below, we're associating all PKS Admins to the AD group, "pksclusteradmins"



Below are details of pksclusteradmins attributes. This includes the Object ID, that you will need to map in UAA to the appropriate group (see below).



UAA Scopes for Enterprise PKS Users

Now that SAML configuration is complete in PKS and Azure, we need to map the UAA scope to the right Azure AD Group. A UAA admin user can assign the following UAA scopes to Enterprise PKS users:

pks.clusters.manage: Accounts with this scope can create and access their own clusters.

pks.clusters.admin: Accounts with this scope can create and access all clusters.

pks.clusters.admin.read: Accounts with this scope can access any information about all clusters except for cluster credentials.

To grant Enterprise PKS access to a SAML group, perform the following steps:

5. Log in as the UAA admin using the procedure in Log in as a UAA Admin.

6. Assign a PKS cluster scope to all users in a SAML group by running the following command:
   - uaac group map --name UAA-SCOPE SAML-GROUP --origin SAML-ORIGIN

Where:

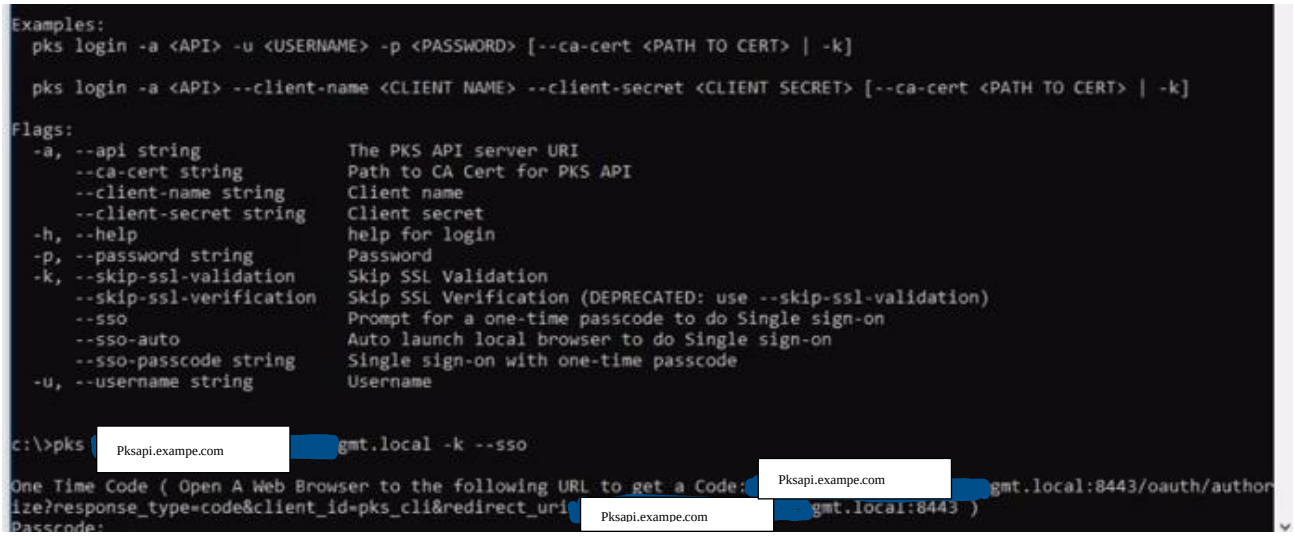
- UAA-SCOPE is one of the UAA scopes described above
- SAML-GROUP is name of your SAML identity provider group. In our example it is the object ID for pksclusteradmins which is 4809484d-dc00-419a-83f4-f1301ae0765b
- SAML-ORIGIN is the domain name for your SAML identity provider. To find SAML-ORIGIN, click on the PKS tile, select Settings > UAA > SAML, and locate the Provider Name. In our example, it is azure (see above)

For example:
$ uaac group map --name pks.clusters.admin 4809484d-dc00-419a-83f4-f1301ae0765b -–origin azure

Sign on using Enterprise PKS CLI

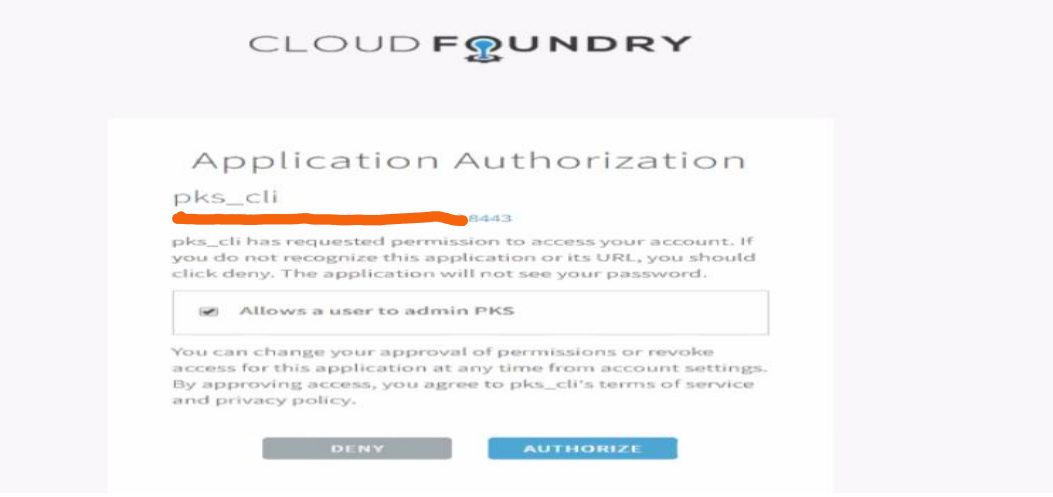
Now you can issue the following command to sign on using SAML Single Sign-on

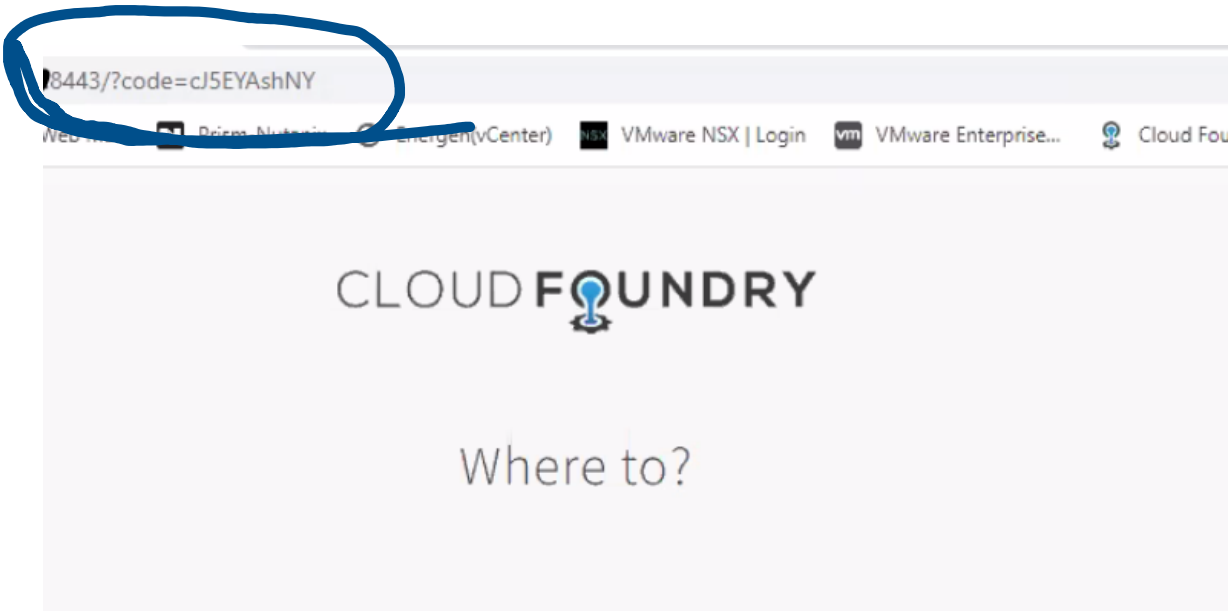
pks login -a {pksapi.example.com} –-sso



Use a web browser and copy the URL to get the code as following and click on “login with azure”

The first time you sign on you will get the following message:





Use the code above to sign on…

Troubleshooting:

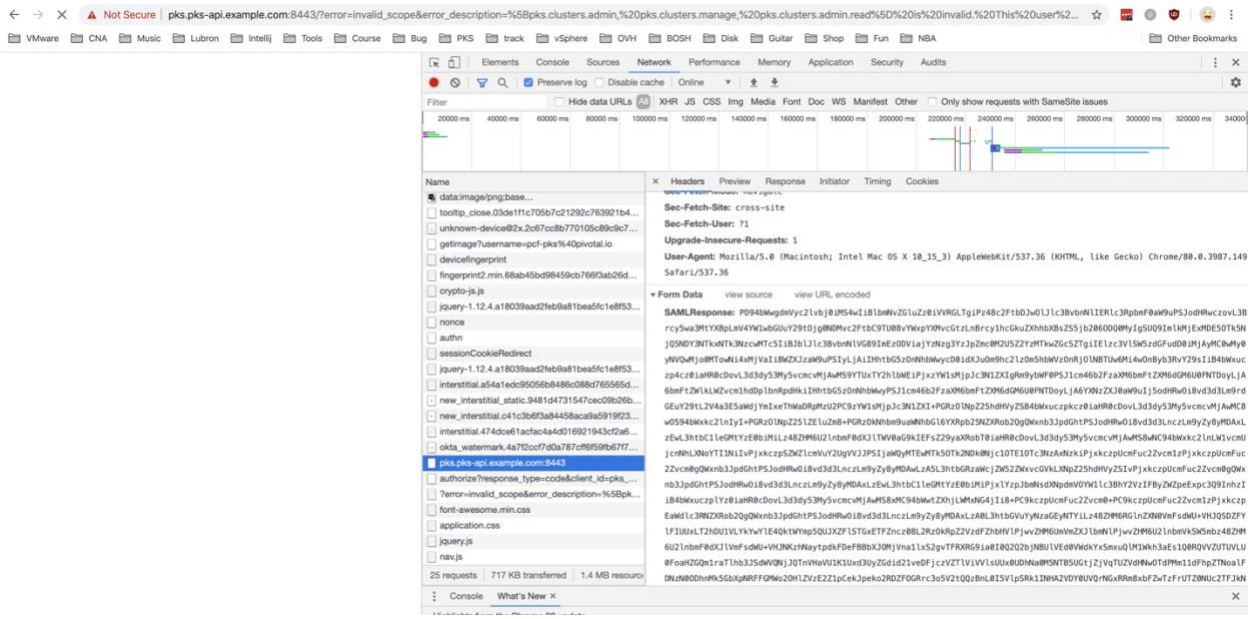
If you see the following error message, the mapping for UAA is not correct or what is coming back in the SAML token is not correct.

```
https://pksapi.example.com:8443/?error=invalid_scope&error_description=%5Bpks.clusters.admin,%20pks.clusters.admin.read,%20pks.clusters.manage%5D%20is%20invalid.%20This%20user%20is%20not%20allowed%20any%20of%20the%20requested%20scopes
```

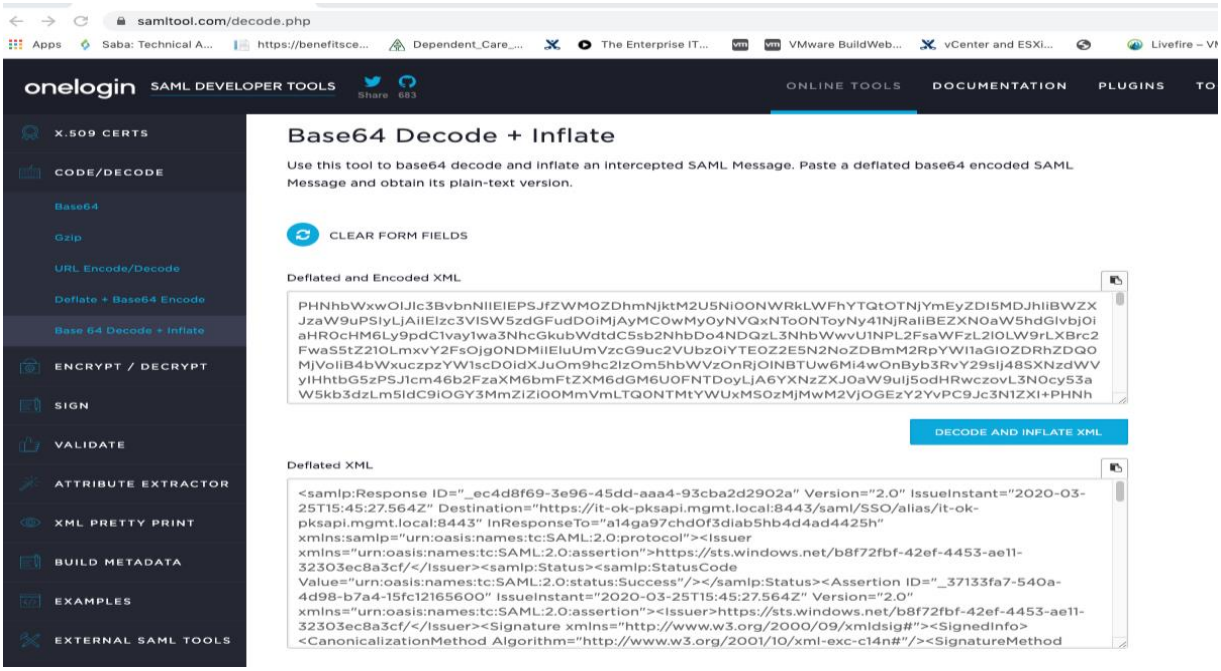
```
error=invalid_scope
error_description=pks.clusters.admin,pks.clusters.admin.read,pks.clusters.manage
Userid is invalid. This user is not allowed any of the requested scopes
```

The following steps can help determine how to resolve this:

Sign on using Chrome. Before hitting enter, right click and click on ‘inspect’ on the page.



Copy the SAML Respose and go to samltool.com/decode.php


samltool.com/decode.php
onelogin SAML DEVELOPER TOOLS
ONLINE TOOLS
DOCUMENTATION
PLUGINS
X.509 CERTS
CODE/DECODE
Base64
Gzip
URL Encode/Decode
Deflate + Base64 Encode
Base 64 Decode + Inflate
ENCRYPT / DECRYPT
SIGN
VALIDATE
ATTRIBUTE EXTRACTOR
XML PRETTY PRINT
BUILD METADATA
EXAMPLES
EXTERNAL SAML TOOLS
Base64 Decode + Inflate
Use this tool to base64 decode and inflate an intercepted SAML Message. Paste a deflated base64 encoded SAML Message and obtain its plain-text version.
CLEAR FORM FIELDS
Deflated and Encoded XML
DECODE AND INFLATE XML
Deflated XML

Here you can see the attributes that are being returned including the groups being returned and evaluated.

```
<Attribute
Name="http://schemas.microsoft.com/ws/2008/06/identity/claims/groups"><AttributeValue>4c68800b-6282-4e3d-8127-7fcd8b8bbb96</AttributeValue></Attribute>
```

For example, In this case we noticed the groups attribute value is being returned as ObjectID instead of the Name of the group. Given this information, we were able to change the mapping to use the Object ID instead of Name.

Harbor Identity Management

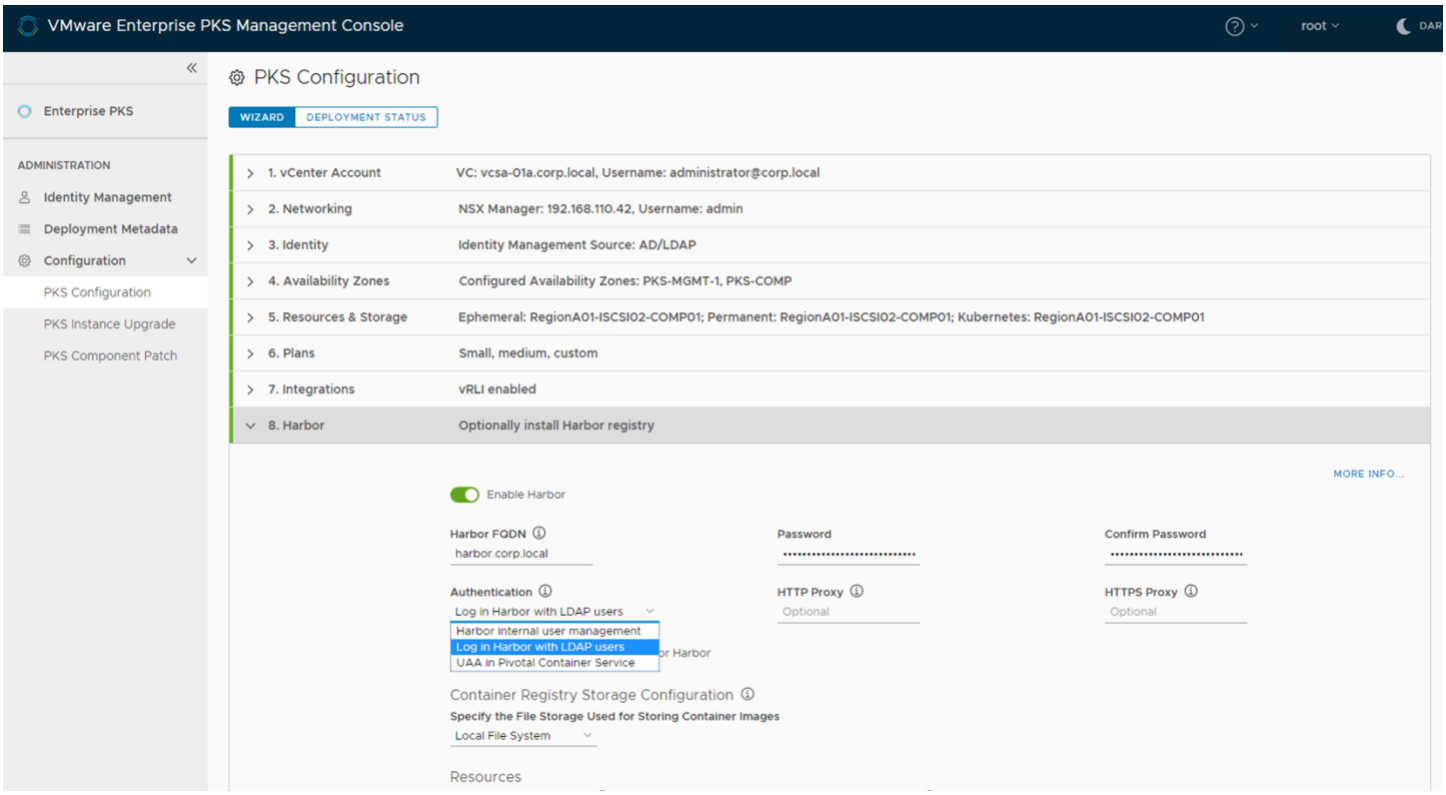
Harbor supports the following identity management endpoints

- Harbor internal user management
- AD/LDAP
- UAA in Pivotal Container Service

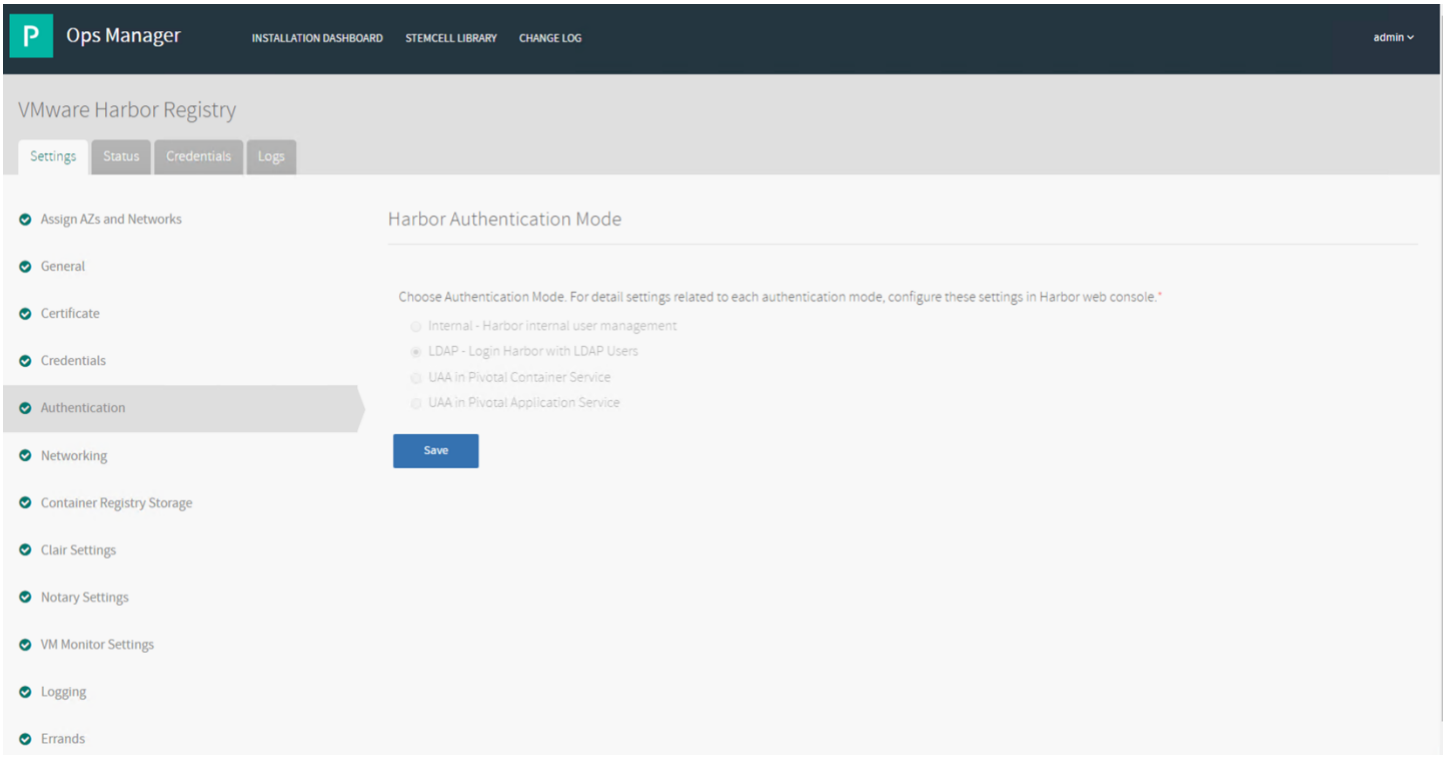
Harbor AD/LDAP as an identity Management end point

Harbor Authentication mode needs to be enabled when installing PKS.

EPMC configuration, select Login harbor with LDAP users



If not using EPMC, change the Authentication mode to LDAP : Login harbor with LDAP users in the authentication section of the Harbor Registry tile within opsman.



Group definitions

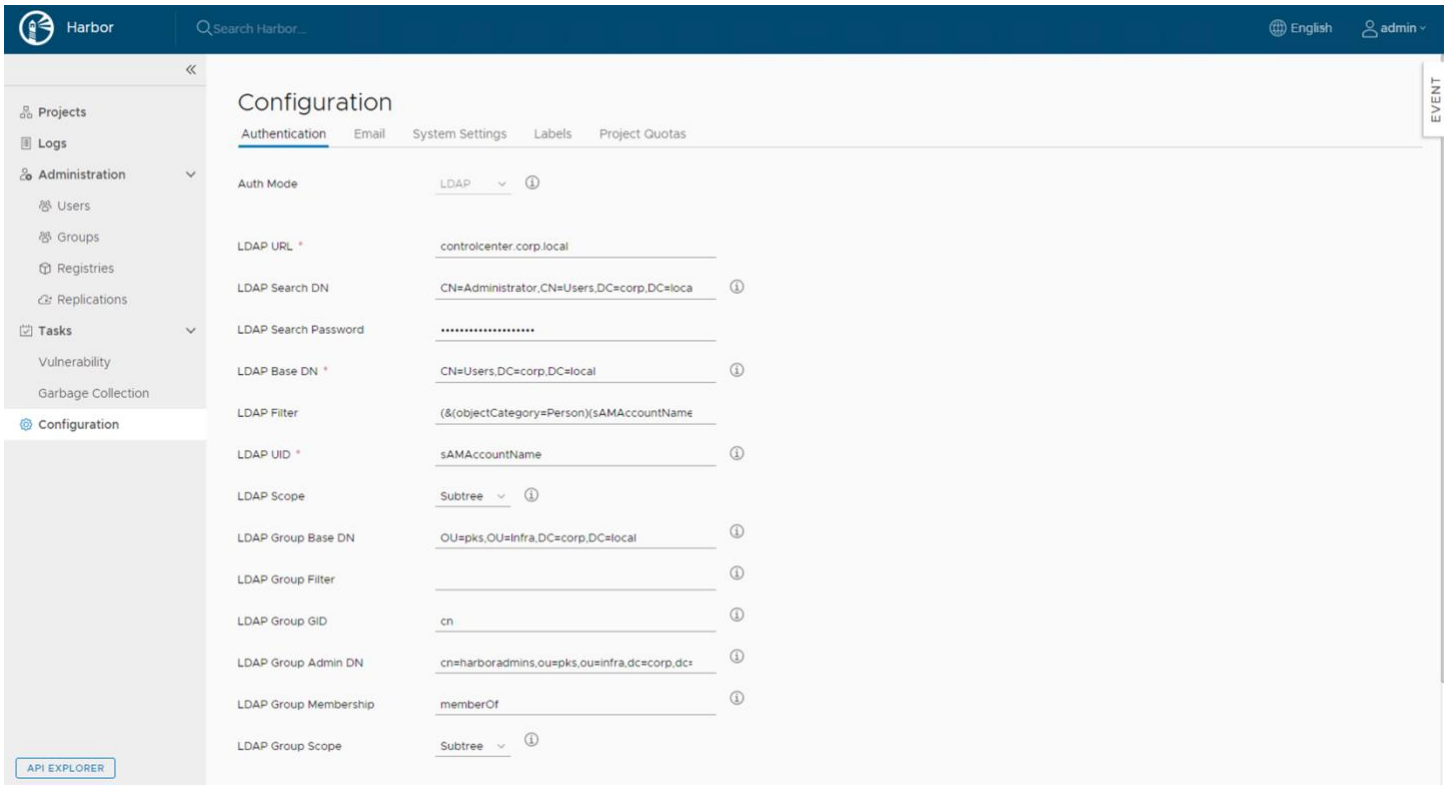
This section defines the usergroups we will be creating within the different identity management endpoints to associate the different personas to

Group	Persona	Role
harboradmins	cody naomi	Harbor system administrator
harborusers	scott	Harbor developer

The groups are created under OU=pks,OU=Infra,DC=corp,DC=local

Harbor LDAP configuration

- Login to harbor as the admin user.
- Navigate to the Configuration section
- Select Authentication and the Authentication Mode as LDAP



- Configure LDAP as per the below table

Description	Values	Description
LDAP URL	controlcenter.corp.local	LDAP hostname
LDAP Search DN	CN=Administrator,CN=Users,DC=corp,DC=local	Service account that has read rights to LDAP
LDAP Search Password	<password>	Service account password
LDAP Base DN	CN=Users,DC=corp,DC=local	DN where Users exist
LDAP Filter*	(&(objectCategory=Person)(sAMAccountName=*)(|(memberOf=CN=harboradmins,OU=pks,OU=Infra,DC=corp,DC=local)(memberOf=CN=harborusers,OU=pks,OU=Infra,DC=corp,DC=local)))	Add a filter that only queries users that belong to specific groups. Eg. harborusers and harboradmins
LDAP UID	sAMAccountName	Attribute name used to search user
LDAP Scope	Subtree	
LDAP Group Base DN	OU=pks,OU=Infra,DC=corp,DC=local	Base DN under which the harbor usergroups exist
LDAP Group Filter		Filter to look up specific LDAP groups
LDAP Group GID	cn	Attribute used in a search to match users , can be sAMAccountName
LDAP Group Admin DN	cn=harboradmins,ou=pks,ou=infra,dc=corp,dc=local	LDAP group that will have admin rights to Harbor
LDAP Group Membership	memberOf	All members of this group will have admin access to harbor
LDAP Group Scope	Subtree	

- Save and Test LDAP server
- Logout of harbor and try logging in as cody and scott

*for a full list of LDAP Filter go to *http://www.ldapexplorer.com/en/manual/109010000-ldap-filter-syntax.htm*

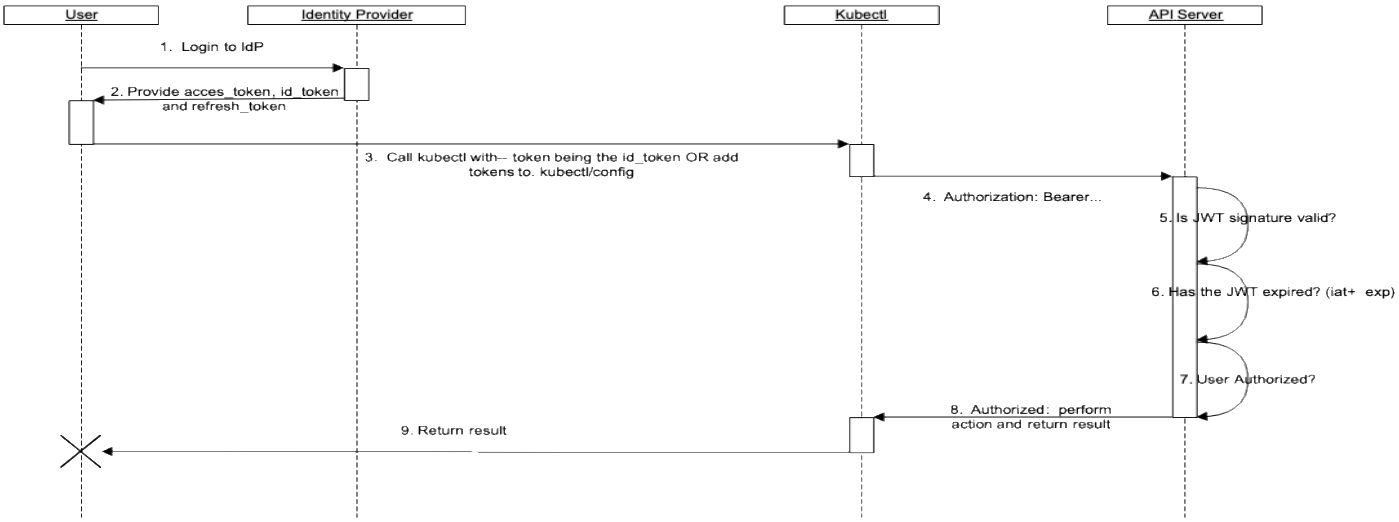
RBAC Authorization to K8 Clusters

This section describes the steps to configure PKS to perform authentication against LDAP server so Developer can access their K8s clusters (globally or via namespace).

PKS UAA integration with LDAP (Windows Active Directory) in order to generate token-id (that's why the functionality is called UAA as OIDC provider).

This token-id then allows the developer to access K8s Cluster using kubectl.

PKS Control Plane VM is interacting with LDAP server here (K8s cluster don't interact with LDAP server at all)
NOTE: Make sure that UAA as an OIDC provider is enabled



Create Bindings

NOTE: Change the values as per the environment

- Login to the PKS API using the following command. Login as alana/cody who are pks api admins

```
pks login -a pks.corp.local -u cody -k
```

- Get credentials to the cluster

```
pks get-credentials my-cluster
```

- Create a namespace called ‘dataengg’ and run 5 replicas of nginix in it

```
kubectl create ns dataengg

kubectl run nginx --image=nginx --replicas=5 --namespace=dataengg
```

Roles and Role Bindings

A Role contains rules that represent a set of permissions. Permissions are purely additive (there are no “deny” rules). A Role can be defined within a namespace with a Role, or cluster-wide with a ClusterRole.

A Role can only be used to grant access to resources within a single namespace.

A Role binding grants the permissions defined in a Role to a user or set of users. It holds a list of subjects (users, groups, or service accounts), and a reference to the Role being granted. Permissions can be granted within a namespace with a RoleBinding, or cluster-wide with a ClusterRoleBinding.

Cluster Roles

- Create a cluster role for certain resources with get, list and watch permissions

  The following is an example of a cluster role for with permissions to all resources
  PKS by default has a cluster-admin cluster role configured, this could be used instead.

  Copy the contents below to a file cluster-all-role.yaml

---

kubectl apply -f cluster-all-role.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-all-admin
rules:
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - '*'
- nonResourceURLs:
  - '*'
  verbs:
  - '*'
```

The following is an example of a cluster role for certain resources with get, list and watch permissions

Copy the contents below to a file cluster-reader-role.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: reader
rules:
- apiGroups: [""]
  resources: ["deployments", "configmaps", "pods", "secrets", "services"]
  verbs: ["get", "list", "watch"]
```

kubectl apply -f cluster-reader-role.yaml

Cluster Rolebinding

- Create a cluster role binding for k8 cluster administrators. The persona associated with this role is naomi.

  The cluster role referred here is cluster-admin that exists in pks by default.

  Copy the contents to a file admin-cluster-binding.yaml

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: dev-cluster-role-binding
subjects:
- kind: Group
  name: oidc:k8clusteradmins # Name is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: cluster-admin
  apiGroup: rbac.authorization.k8s.io
```

NOTE: We are assigning the group ‘k8clusteradmins’ that is present in our LDAP configuration.

- Logout of the pks using the pks cli

  pks logout

```
ubuntu@cli-vm:~/roles$ pks logout

You have successfully logged out
```

- Retrieve the kube config for user

  To obtain kubeconfig for a selected cluster and a user, run the following command:

```
pks get-kubeconfig <CLUSTER-NAME> -u <USER> -p <password> -a <PKS API Server> -k
```

  Eg. pks get-kubeconfig my-cluster -u naomi -a pks.corp.local -k

Test OIDC based access for 'k8administrators'

- Retrieve the kubeconfig for persona naomi

```
pks get-kubeconfig my-cluster -u naomi -a pks.corp.local -k
```

- Set context

```
kubectl config set-context my-cluster --user=naomi
kubectl get po --all-namespaces
```

  All pods running in the cluster should be visible

```
riazm-a01:roles riazm$ kubectl config set-context riaz-cluster-sd --user=thor
Context "riaz-cluster-sd" modified.
riazm-a01:roles riazm$ kubectl get po --all-namespaces
NAMESPACE     NAME                                                    READY   STATUS      RESTARTS   AGE
dataengg      nginx-7bb7cd8db5-8mpmc                                  1/1     Running     0          14h
dataengg      nginx-7bb7cd8db5-fhjpc                                  1/1     Running     0          14h
dataengg      nginx-7bb7cd8db5-g65lw                                  1/1     Running     0          14h
dataengg      nginx-7bb7cd8db5-ltpc7                                  1/1     Running     0          14h
dataengg      nginx-7bb7cd8db5-rs9sd                                  1/1     Running     0          14h
kube-system   coredns-6f9bcd8956-7tzlz                                1/1     Running     0          16h
kube-system   coredns-6f9bcd8956-djcr2                                1/1     Running     0          16h
kube-system   coredns-6f9bcd8956-zjwbk                                1/1     Running     0          16h
kube-system   kubernetes-dashboard-5fc4ccc79f-9fzz6                   1/1     Running     0          16h
kube-system   metrics-server-7f85c59675-xdrbx                         1/1     Running     0          16h
pks-system    cert-generator-11b35c51b71ea3086396a780dbf20b5cd695b25d-rwxh7   0/1     Completed   0          16h
pks-system    event-controller-7b96987577-m58k5                       2/2     Running     0          16h
pks-system    fluent-bit-684pr                                        2/2     Running     0          16h
pks-system    fluent-bit-l29p6                                        2/2     Running     0          16h
pks-system    fluent-bit-z4mzz                                        2/2     Running     0          16h
pks-system    kube-state-metrics-65bcd5f775-bd4pv                     2/2     Running     0          16h
pks-system    metric-controller-66b8b66498-jbrkk                      1/1     Running     0          16h
pks-system    observability-manager-7dd6c4c6d-fw8ql                   1/1     Running     0          16h
pks-system    sink-controller-5d76d8d546-cphp8                        1/1     Running     0          16h
pks-system    telegraf-kf5g2                                          1/1     Running     0          16h
pks-system    telegraf-ndj49                                          1/1     Running     0          16h
pks-system    telegraf-tzlgn                                          1/1     Running     0          16h
pks-system    telemetry-agent-58797bf64d-whrfm                        2/2     Running     0          16h
pks-system    validator-847cb99cc-hpllj                               1/1     Running     0          16h
pks-system    vrops-cadvisor-jvx29                                    1/1     Running     0          16h
pks-system    vrops-cadvisor-m969s                                    1/1     Running     0          16h
pks-system    vrops-cadvisor-wsf2h                                    1/1     Running     0          16h
pks-system    wavefront-collector-5c9fcfff8-k5cmw                     1/1     Running     0          16h
pks-system    wavefront-proxy-66d6445f6c-s258c                        1/1     Running     0          16h
```

Create Roles

Create a Role on the namespace dataengg which has reader permissions for pods and pod-logs.(get and list permissions). Copy the contents to a file role-dataengg.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: dataengg
  name: dataengg-pod-and-pod-logs-reader
rules:
- apiGroups: [""]
  resources: ["pods", "pods/log"]
  verbs: ["get", "list"]
```

kubectl apply -f role-dataengg.yaml

Role binding for the Developer Role

- Create a Role binding for the pksdev group to access the Roles defined in role-dataengg.yaml for the namespace dataengg

  Copy the contents to a file dataengg-role-binding.yaml

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: dataengg-pod-RoleBinding
  namespace: dataengg
subjects:
- kind: Group
  name: oidc:k8developers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: dataengg-pod-and-pod-logs-reader
  apiGroup: rbac.authorization.k8s.io
```

NOTE: We are assigning the group ‘pksdev’ that is present in our LDAP configuration

kubectl apply -f dataengg-role-binding.yaml

Test OIDC based access for 'k8developers

- Retrieve the kubeconfig for persona naomi

```
pks get-kubeconfig my-cluster -u naomi -a pks.corp.local -k
```

- Set context

```
kubectl config set-context my-cluster --user=scott
```

- List all pods running

```
    kubectl get po --all-namespaces
```

This will result in an error since 'scott' is not an admin and has specific permissions to on namespace dataengg

Error from server (Forbidden): pods is forbidden:

```
riazm-a01:roles riazm$ kubectl get po --all-namespaces
Error from server (Forbidden): pods is forbidden: User "oidc:visiondev" cannot list resource "pods" in API group "" at the cluster scope
```

- List pods running in the dataengg namespace

kubectl get po --namespace dataengg

```
riazm-a01:roles riazm$ kubectl get po --namespace dataengg
\NAME                     READY   STATUS    RESTARTS   AGE
nginx-7bb7cd8db5-8mpmc   1/1     Running   0          14h
nginx-7bb7cd8db5-fhjpc   1/1     Running   0          14h
nginx-7bb7cd8db5-g65lw   1/1     Running   0          14h
nginx-7bb7cd8db5-ltpc7   1/1     Running   0          14h
nginx-7bb7cd8db5-rs9sd   1/1     Running   0          14h
```

KubeConfig

SOLUTION 1: (Recommended)
Retrieve kubeconfig file

In the LDAP configuration we have 'visiondev' assigned to the 'pksdev' group with a password <password>
To obtain kubeconfig for a selected cluster and a user, run the following command:

```
pks get-kubeconfig <CLUSTER-NAME> -u <USER> -p <password> -a <PKS API Server> -k
```

NOTE: -k option used to avoid "missing certificate" warning

E.G. pks get-kubeconfig my-cluster -u naoimi -p <password> -a pks.corp.local -k

This will generate kubeconfig file which can be distributed to Developer to access the cluster.

Admin users can review user related context in the kubeconfig file using **kubectl config get-contexts** command like shown below:

```
kubectl config get-contexts

CURRENT   NAME                                          CLUSTER                                       AUTHINFO
NAMESPACE
……….

riaz-cluster-vra8_1                             riaz-cluster-vra8_1                          0dfc190f-0a6e-45c8-afde-
9dbd098d4deb
raghu-cluster-ace1                               raghu-cluster-ace1                                87930898-554f-
49fb-a689-0d6148b6fc51
raghu-cluster-vra8                              raghu-cluster-vra8                           34ad0f04-5e2d-
4baf-902b-11df2d7c10ca
*     riaz-cluster-sd                                   riaz-cluster-sd                                visiondev

        Currently active context is marked with '*'
```

END SOLUTION 1

SOLUTION 2:

Retrieve token for Developer

In the LDAP configuration we have visiondev assigned to the pksdev group with a password <password>

Copy the contents to a file getToken.sh

```
curl 'https://pks-ace-api.eng.vmware.com:8443/oauth/token' -k -XPOST -H 'Accept: application/json' -d
"client_id=pks_cluster_client&client_secret=""&grant_type=password&username=visiondev&password=<password>&respo
nse_type=id_token"
```

Run getToken.sh

./getToken.sh

```
This will result in both a id token and a refresh token

        E.g.
        {"access_token":"eyJhbGciOiJSUzI1NiIsImprdSI6Imh0dHBzOi8vcGtzLWFjZS1hcGkuZW5nLnZtd2FyZS5jb
        ……
        cn6Bxg","expires_in":599,"scope":"openid roles","jti":"a7c42e36061b44ab809920016237f3
```

Create kubeconfig File

Create the kube config file or Update an existing one with the id token and refresh token. (visiondev). If visiondev does not exist in your existing config file, add a new entry.
Note : config file can be found under ~/.kube/config
If it does not exist create one

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: Ci0tLS0tQkVHSU4gQ0VSVElGSUNBVEUtLS0tLQpNSUlDK3pDQ0FlT2dBd0lCQWdJVWEvZjdGWWVUS1VFTE9
…..
uEU0MTlkSG9YUHlJa3NiUWNjMEVIMVI4dm5tUzVsYmZZdCtBREE9Ci0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0K
    server: https://my-cluster.corp.local:8443
  name: my-cluster
- name: thor
  user:
    auth-provider:
      config:
        client-id: pks_cluster_client
        client-secret: ""
        id-token:
eyJhbGciOiJSUzI1NiIsImprdSI6Imh0dHBzOi8vcGtzLWFjZS1hcGkuZW5nLnZtd2FyZS5jb206ODQ0My90b2tlbl9rZXlzIiwia2lkIjoia2V5LTEiLCJ
0eXAiOiJKV1QifQ.
…….
of_ObK2-RJN9LeGsynnuBgD8qfZfUZoiV2EkySf2OnEhXXSSB8JIH4oDvhEPA
        idp-certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURqRENDQW5TZ0F3SUJBZ0lWQ
…..
hdC8vMk41UXg0dGkvWE4rSE4vUDMKLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQo=
        idp-issuer-url: https://pks-ace-api.eng.vmware.com:8443/oauth/token
        refresh-token: eyJhbGciOiJSUzI1NiIsImprdSI6Imh0dHBzOi8vcGtzLWFjZS1hcGkuZW5nLnZtd2FyZS5jb2
…..
uOzlQBv6Dm2fw-MMP3IDBKuuA
      name: oidc
- name: visiondev
  user:
    auth-provider:
      config:
        client-id: pks_cluster_client
        client-secret: ""
        id-token:
eyJhbGciOiJSUzI1NiIsImprdSI6Imh0dHBzOi8vcGtzLWFjZS1hcGkuZW5nLnZtd2FyZS5jb206ODQ0My90b2tlbl9rZXlzIiwia2lkIjoia2V5LTEiLCJ
0eXAiOiJKV1QifQ.
…..
zu107ERplGSp4zw60syn52439L7OOb4ZjnK7c8VvOXNuw9Q5NT2GAJou3tNBgfQjLxUAgLxx7H0Rg
        idp-certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURqRENDQW5TZ0F3SUJBZ0lWQ
….
THBuTFhXCnN0VnAxVVI3UkZqS3NoOUJ4VnhMRldJanV3ZjlNdGhMQ2lOZEJnd3Q2aFZhdC8vMk41UXg0dGkvWE4rSE4vUDMKLS0tLS1FTkQgQ0VSVElGSUN
BVEUtLS0tLQo=
        idp-issuer-url: https://pks-ace-api.eng.vmware.com:8443/oauth/token
        refresh-token:
eyJhbGciOiJSUzI1NiIsImprdSI6Imh0dHBzOi8vcGtzLWFjZS1hcGkuZW5nLnZtd2FyZS5jb206ODQ0My90b2tlbl9rZXlzIiwia2lkIjoia2V5LTEiLCJ
0eXAiOiJKV1QifQ.
…..

_0e2LzMVT04uF9VllvwT74jjssmpXolMsdJvWs50g9qMVKouvsBIHPYl5xWVh0GCfkTvaqKamr2qLZg2mHz53uzrpzBl8Vc
zd3BKDO6ysDM_WNi_JjoNZKHhI6xI-NTe8Xxcyy2n9a7d0lpr7Ai6V-imOsfy8fdvh5RKwuq-
      name: oidc
```

END SOLUTION 2

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com.