SWIFT - Prerequisites & Installation Guide

Version 1.3.17

RACKWARE INC.

Contents

Preparing for the SWIFT installation

The SWIFT is a container orchestration, backup, and disaster recovery (DR) product. It works on top of your existing container platforms like Kubernetes and OpenShift for replications, DR, and backup. You will need the below resources before you start the SWIFT installation:

1. A 64-bit Centos/RedHat 7.x/8.x x86 host*
2. Minimum 3GB free space**
3. 8GB or more RAM
4. Two or more (V)CPUs
5. Internet connectivity on the host (yum needs to be working with EPEL repo enabled)

Once the SWIFT server is ready, subsequent sections point on any other configuration steps that need to be done before actual SWIFT package installation.

**This is where SWIFT will be installed, and it can be a VM or a physical server. CentOS Stream is also supported.*

***For detailed partition space requirements on the SWIFT server, see the next section. If you plan to do backups along with DR, then you will need additional storage during SWIFT operations. The extra storage need not be ready while installing the SWIFT and it can be added later.*

SWIFT Server Storage Requirements

Partition Free Space Requirements

SWIFT installation requires below partitions or paths to have the mentioned minimum free space. If the required minimum free space doesn't exist for one or more paths below, then the installation will fail with an error.

While the required minimum free space for below partitions/paths will let the installation continue, we suggest you provision partitions such way that the recommended free space (from the third column below) for each partition/path can be achieved during installation to make sure of smooth functioning of the solution in the longer run.

Partition or path	Required minimum free space (GB)	Recommended free space (GB)
/opt	3 GB	50 GB
/tmp	3 GB	10 GB
/var/log	3 GB	10 GB

SWIFT Server DR/Backup Storage Requirements

If you are using SWIFT for backup or Disaster Recovery (DR) use cases, then apart from free space for the above partition/paths on the SWIFT server, you will also require one or more block devices attached to the SWIFT server that it will use for backup of source applications' data/volumes. Recommended free block storage that you should attach to the SWIFT server is same as total of your source clusters'/applications' volume sizes. For example, if you plan to backup three clusters with SWIFT that are running 10 applications where applications are using total Kubernetes volumes of 3TB in size, then you need to attach 3TB worth of block storage volumes to SWIFT server. The attached volumes can be of any size and in any number, just they need to be 3TB in total in the example scenario.

Please refer to SWIFT Operations Guide later to know how to configure attached block storage/volumes inside SWIFT dashboard.

Note that you can optionally attach these backup/DR storage block volumes to SWIFT server later and they are not required to complete the installation of the SWIFT software.

Kernel Package Install

SWIFT installation requires 'kernel-devel' package installation on CentOS/RHEL OS. Later it is used by the SWIFT installer to build and deploy the ZFS storage modules on the server.

For every running/available kernel, there is a matching kernel-devel package available in the yum repo. Run the following command to install the kernel-devel package for the running kernel.

```
# yum install kernel-devel-$(uname -r) -y
```

If the above step fails on missing required kernel-devel package, then you must upgrade to the latest available kernel version, reboot the server, and then install the latest kernel-devel package.

```
# yum upgrade kernel

# reboot

# yum install kernel-devel -y
```

After this, the kernel running on the server will have the equivalent kernel-devel package installed on the server.

SWIFT port usage

Various operations are supported for the SWIFT managed container platform. The required network ports will change depending on the type of the container platform and the type of operation performed. The next section highlights port usage per platform and operation type.
If there is any intermediate firewall between the SWIFT server and your remote container platform, then these below ports also need to be opened in the intermediate firewall.

Kubernetes Discover

Port Number (Default)	Direction	Can it be changed?	Purpose
<Kubernetes-API-Server-IP>:443	SWIFT to cluster	Yes (specify during discover and change from configure operation for the cluster)	Talk to API service

Kubernetes Configure

Port Number (Default)	Direction	Can it be changed?	Purpose
<Kubernetes-API-Server-IP>:443	SWIFT to cluster	Yes (specify during discover and change from configure operation for the cluster)	Talk to API service

Kubernetes Sync

Port Number (Default)	Direction	Can it be changed?	Purpose
<Kubernetes-API-Server-IP>:443	SWIFT to cluster	Yes (specify during discover and change from configure operation for the cluster)	Talk to API service
<TRAI-Service-IP>:<port-1> [Default port-range is typically: 32000-34000)	SWIFT to cluster	Yes (specify during sync else auto picked)	Connect to transient sync staging POD/service within the cluster over HTTPS (Management port)
<TRAI-Service-IP>:<port-2> [Default port-range is typically: 32000-34000)	SWIFT to cluster	Yes (specify during sync else auto picked)	Pull/push data from/to transient sync staging POD/service within the cluster over an encrypted tunnel (Data port)

SWIFT sync run will typically use two unique ports from the cluster's service port range on both sides of clusters. If you do sync on a namespace basis and the selected namespace has more than one region

or zone set for persistent volumes on the source cluster side, then SWIFT will run its transient TRAI-Pod for each region/zone. In such cases, you will need 2xTRAI-Pod number of ports per sync. E.g., if your synced source namespace has two regions or zones for persistent volumes and you are syncing the entire namespace as part of a sync run, then the sync will deploy two TRAI-Pods (one for each region or zone of source namespace) and so the sync will need 2x2=4 ports from the source cluster's service port range. These ports can be fixed or can be auto picked by SWIFT. In all cases, SWIFT will always need only two ports per sync run for the target or DR cluster.

Note that any transient TRAI service/Pod ports, either auto picked by SWIFT or specified manually, need to be opened in all intermediate firewalls between the SWIFT server and your cluster. If you want SWIFT to auto pick required ports from the service port range of the cluster, then it is recommended that you whitelist the entire cluster service port range in all intermediate firewalls between the SWIFT server and your cluster.

The <TRAI-Service-IP> can be any of the supported IP types for the cluster services (E.g., ClusterIP, NodePort, LoadBalancer, etc.) and can be optionally specified during the sync.
Depending on the selected TRAI service type, sync will default to auto picked IP for its transient TRAI Kubernetes Service. If you employ NodePort service type for SWIFT's transient TRAI service, then you need to open required sync ports against all cluster node IPs (Also known as NodePort IPs).

OpenShift Discover

Port Number (Default)	Direction	Can it be changed?	Purpose
<OpenShift-API-Server-IP>:443	SWIFT to cluster	Yes (specify during discover and change from configure operation for the cluster)	Talk to API service

OpenShift Configure

Port Number (Default)	Direction	Can it be changed?	Purpose
<OpenShift-API-Server-IP>:443	SWIFT to cluster	Yes (specify during discover and change from configure operation for the cluster)	Talk to API service

OpenShift Sync

Port Number (Default)	Direction	Can it be changed?	Purpose
<OpenShift-API-Server-IP>:443	SWIFT to cluster	Yes (specify during discover and change from configure operation for the cluster)	Talk to API service

<TRAI-Service-IP>:<port-1> [Default port-range is typically: 32000-34000)	SWIFT to cluster	Yes (specify during sync else auto picked)	Connect to transient sync staging POD/service within the cluster over HTTPS (Management port)
<TRAI-Service-IP>:<port-2> [Default port-range is typically: 32000-34000)	SWIFT to cluster	Yes (specify during sync else auto picked)	Pull/push data from/to transient sync staging POD/service within the cluster over an encrypted tunnel (Data port)

SWIFT sync run will typically use two unique ports from the cluster's service port range on both sides of clusters. If you do sync on a namespace basis and the selected namespace has more than one region or zone set for persistent volumes on the source cluster side, then SWIFT will run its transient TRAI-Pod for each region/zone. In such cases, you will need 2xTRAI-Pod number of ports per sync. E.g., if your synced source namespace has two regions or zones for persistent volumes and you are syncing the entire namespace as part of a sync run, then the sync will deploy two TRAI-Pods (one for each region or zone of source namespace) and so the sync will need 2x2=4 ports from the source cluster's service port range. These ports can be fixed or can be auto picked by SWIFT. In all cases, SWIFT will always need only two ports per sync run for the target or DR cluster.

Note that any transient TRAI service/Pod ports, either auto picked by SWIFT or specified manually, need to be opened in all intermediate firewalls between the SWIFT server and your cluster. If you want SWIFT to auto pick required ports from the service port range of the cluster, then it is recommended that you whitelist the entire cluster service port range in all intermediate firewalls between the SWIFT server and your cluster. In all cases, SWIFT will always need only two ports per sync run for the target or DR cluster.

The <TRAI-Service-IP> can be any of the supported IP types for the cluster services (E.g., ClusterIP, NodePort, LoadBalancer, etc.) and can be optionally specified during the sync.
Depending on the selected TRAI service type, sync will default to auto picked IP for its transient TRAI OpenShift Service. If you employ NodePort service type for SWIFT's transient TRAI service, then you need to open required sync ports against all cluster node IPs (Also known as NodePort IPs).

Image Registry Discover

Port Number (Default)	Direction	Can it be changed?	Purpose
<Container-Image-Registry-API-Server-IP>:443	SWIFT to API server	No	Talk to API service

Image Registry Configure

Port Number (Default)	Direction	Can it be changed?	Purpose

<Container-Image-Registry-API-Server-IP>:443	SWIFT to API server	No	Talk to API service

Image Registry Sync

Port Number (Default)	Direction	Can it be changed?	Purpose
<Container-Image-Registry-API-Server-IP>:443	SWIFT to Container Image Registry API server	No	Talk to Container Image Registry API service

The above ports for Image Registry sync are only required if you plan to migrate or set up Disaster Recovery (DR) for your Container Image Registries with SWIFT.

DR Policy Operations

The port usage for DR policy remains the same as respective platform type (discovery and sync) ports documented in the earlier sections, depending on source and/or target platform type selected for the policy and the type of the policy. Additionally, for successful email alerts, the below ports need to be opened from the SWIFT server to the target email server for all configured email accounts where alerts are configured.

Port Number (Default)	Direction	Can it be changed?	Purpose
TCP/587 or custom SMTP port used by the target email server.	SWIFT to remote Email server	Yes (If non-default port used, then specify it in the SWIFT options file for email server options)	Send DR email alerts using local Linux email client

SWIFT uses local Linux email client APIs to relay email alert messages to the configured email accounts for DR policy alerts. If your environment uses non-standard or custom SMTP port, then you should whitelist that port instead of above port. Make sure to specify the custom SMTP port in the SWIFT options file too under the below options, so SWIFT can use that for relaying email alerts for policies. (The SWIFT options file is located on the SWIFT server at: /opt/swift/data/options).

dev_swiftAlertOptions_emailAlertOptions_emailServerUrl=smtp://127.0.0.1:587
dev_swiftAlertOptions_emailAlertOptions_localEmailServer=smtp://127.0.0.1:587

Note: It is important you uncomment the line (by removing # at the start of the line) for set options in the options file, and then restart SWIFT service for the newly set options to take effect.

Doing SWIFT install or upgrade

Fresh install

Once you have identified the host for SWIFT install, the next step is a breeze. Just run the installer for the SWIFT version that you have downloaded. You will be given a EULA prompt, accept it with 'yes' and then accept all defaults for subsequent prompts.

```
Verifying archive integrity...  100%   All good.
Uncompressing RackWare SWIFT Package  100%
Installation log file: /var/log/swift/installer.log


You must accept the terms of the license agreement to install and use this software.
END-USER LICENSE AGREEMENT FOR RACKWARE.

IMPORTANT PLEASE READ THE TERMS AND CONDITIONS OF THIS LICENSE
AGREEMENT
CAREFULLY BEFORE INSTALLING OR USING THIS SOFTWARE:

RackWare's End-User License Agreement ("EULA") is a legal agreement between
you (either an individual or a single entity) and RackWare, Inc. ("RACKWARE")
for the RackWare software product identified above which may include
associated software components, media, printed materials, and "online" or
electronic documentation ("SOFTWARE PRODUCT").  By installing, copying, or
otherwise using the SOFTWARE PRODUCT, you agree to be bound by the terms of
this EULA.  This license agreement represents the entire agreement concerning
the program between you and RackWare, (referred to as "licenser"), and it
supersedes any prior proposal, representation, or understanding between the
parties.  If you do not agree to the terms of this EULA, do not install or use
the SOFTWARE PRODUCT.

The SOFTWARE PRODUCT is protected by copyright laws and international
copyright treaties, as well as other intellectual property laws and treaties.
The SOFTWARE PRODUCT is licensed, not sold.

1. GRANT OF LICENSE.
Rackware grants you, subject to the terms and conditions of this EULA, a
limited, non-exclusive, non-assignable license to the SOFTWARE PRODUCT as
follows:
(a) Installation and Use.
    RackWare grants you the right to install and use copies of the
    SOFTWARE PRODUCT on computers running a validly licensed copy of the
    operating system for which the SOFTWARE PRODUCT was designed.
(b) Backup Copies.
    You may also make copies of the SOFTWARE PRODUCT as may be necessary for
    backup and archival purposes.
```

2. DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS.
(a) Maintenance of Copyright Notices.
You must not remove or alter any copyright notices on any and all copies of the SOFTWARE PRODUCT.
(b) Distribution.
You may not distribute copies of the SOFTWARE PRODUCT to third parties.
(c) Prohibition on Reverse Engineering, Decompilation, and Disassembly.
You may not attempt to reverse engineer, decompile, or disassemble all or any portion of the SOFTWARE PRODUCT, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation.
(d) Rental.
You may not rent, lease, or lend the SOFTWARE PRODUCT.
(e) Support Services.
RackWare may provide you with support services related to the SOFTWARE PRODUCT ("Support Services"). Any supplemental software code provided to you as part of the Support Services shall be considered part of the SOFTWARE PRODUCT and subject to the terms and conditions of this EULA.
(f) Compliance with Applicable Laws.
You must comply with all applicable laws regarding use of the SOFTWARE PRODUCT.

3. TERMINATION
Without prejudice to any other rights, RackWare may terminate this EULA if you fail to comply with the terms and conditions of this EULA. In such event, you must destroy all copies of the SOFTWARE PRODUCT in your possession.

4. COPYRIGHT
All title, including but not limited to copyrights and other intellectual property rights, in and to the SOFTWARE PRODUCT and any copies thereof are owned by RackWare or its suppliers. All title and intellectual property rights in and to the content which may be accessed through use of the SOFTWARE PRODUCT is the property of the respective content owner and may be protected by applicable copyright or other intellectual property laws and treaties. This EULA grants you no rights to use such content. All rights not expressly granted are reserved by RackWare.

5. NO WARRANTIES
RackWare expressly disclaims any warranty for the SOFTWARE PRODUCT. The SOFTWARE PRODUCT is provided 'As Is' without any express or implied warranty of any kind, including but not limited to any warranties of merchantability, noninfringement, or fitness for a particular purpose. RackWare does not warrant or assume responsibility for the accuracy or completeness of any information, text, graphics, links or other items contained within the SOFTWARE PRODUCT. RackWare makes no warranties respecting any harm that may

be caused by the transmission of a computer virus, worm, time bomb, logic bomb, or other such computer program. RackWare further expressly disclaims any warranty or representation to Authorized Users or to any third party.

6. LIMITATION OF LIABILITY
In no event shall RackWare be liable for any damages (including, without limitation, lost profits, business interruption, or lost information) rising out of 'Authorized Users' use of or inability to use the SOFTWARE PRODUCT, even if RackWare has been advised of the possibility of such damages. In no event will RackWare be liable for loss of data or for indirect, punitive, special, incidental, consequential (including lost profit), or other damages based in contract, tort or otherwise. RackWare shall have no liability with respect to the content of the SOFTWARE PRODUCT or any part thereof, including but not limited to errors or omissions contained therein, libel, infringements of rights of publicity, privacy, trademark rights, business interruption, personal injury, loss of privacy, moral rights or the disclosure of confidential information.

7. MISCELLANEOUS
In the event of a legal dispute arising out of or related to this Agreement, the prevailing party in any litigation will be entitled to recover its attorney fees and court costs from the non-prevailing party. If any installation or use of the SOFTWARE PRODUCT requires a click-through agreement in recognition of any third party rights related to the SOFTWARE PRODUCT, you agree that any provisions of such click-through agreement shall be binding upon you in addition to the terms of this EULA or any other agreement you have with RackWare.
Please type 'yes' then [enter] to accept the terms of the license agreement, or simply press [enter] to abort installation.

Enter your acceptance: yes
Beginning installation ...

Extracting new SWIFT install files ...

…
…

One specific input you may want to customize for your environments is the NTP server name or IP. SWIFT needs an NTP server name or IP during its operation. By default, SWIFT would configure the 'pool.ntp.org' server, but you can change it to something local you use in your environment.

```
Configuring NTP ...
Configured NTP server for the SWIFT: pool.ntp.org
 (System level NTP settings are not changed.)

Do you want to configure a different NTP SERVER for the SWIFT? (Type: y/yes/n/no)  [no]: yes

Enter the NTP server name or IP  [pool.ntp.org]:      myntp.local
Configuring the NTP server for the SWIFT use: myntp.local
(System level NTP settings are not changed.)
Configuring all SWIFT components and services ...
Installing dependency packages ...
…
…
```

You will also be prompted to set a password for the 'admin' user at the end of the install, which is a built-in superuser for SWIFT. You can skip this step and set up this password later by running the below command over SSH to the SWIFT server. The installer will also print the same command if you skip setting the password during the fresh install. Note that you will need this user's password set to log in to the SWIFT dashboard and create other users and groups initially.

```
sudo swiftcli user modify admin –password <password>
```

If a restart is needed, the installer will prompt that in the end. If prompted, it will be highly recommended to complete the restart of the SWIFT server immediately for the correct functioning of all the SWIFT services.

Upgrade

For upgrade, steps remain the same as a fresh installation.
You may see an error in an upgrade if any of the SWIFT services are running, and you run the new installer package.

```
Verifying archive integrity...  100%   All good.
Uncompressing RackWare SWIFT Package  100%
Installation log file: /var/log/swift/installer.log

ERROR: The existing SWIFT processes are still running. Please stop those with the 'swiftadm stop
all' command, and re-run this installer.

Make sure no SWIFT operation is running before stopping any of its processes.

Aborting installation.

------------------------------------------------------------
```

```
Installation log file: /var/log/swift/installer.log

------------------------------------------------------------
```

Stop the services with 'swiftadm stop all' as the prompt mentions, and then re-run the installer package. Make sure none of the SWIFT operations are running before you stop services. The ongoing operations resume after the SWIFT starts back, but it would be recommended to do an upgrade only when the SWIFT server is idle.

Once the installer runs to an end, it will complete the upgrade, and the SWIFT services would be automatically started back. If a restart is needed for the upgrade, the installer will prompt that in the end. If prompted, it will be highly recommended to complete the restart of the SWIFT server for the correct functioning of the upgraded SWIFT services.

Creating cluster credentials for SWIFT use

Below sections highlight steps for creating cluster credentials for use with the SWIFT. Note that steps will change for every cloud and non-cloud installs.

The generated credentials are something you would use while configuring cluster details in the SWIFT or using object-storage for the cloud with SWIFT. The same credentials will also be used for the container image registry discover and administration for the respective clouds or platform types.

Adding local Kubernetes cluster service-account for SWIFT use

Before you can add your local (non-cloud) Kubernetes cluster to SWIFT and start managing it, you will need to have a cluster service account created with the necessary permissions.

Create a YAML for the new service account:

```
$ vi swift-admin-sa.yaml
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: swift-admin
  namespace: kube-system
```

Apply the YAML file

```
$ kubectl apply -f swift-admin-sa.yaml
```

Next, add the ‘cluster-admin’ role to the newly created account.

```
$ vi swift-admin-roles.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: swift-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: swift-admin
  namespace: kube-system
---
apiVersion: v1
kind: Secret
metadata:
  name: swift-admin
  namespace: kube-system
  annotations:
        kubernetes.io/service-account.name: swift-admin
type: kubernetes.io/service-account-token
```

Apply the YAML file

```
$ kubectl apply -f swift-admin-roles.yaml
```

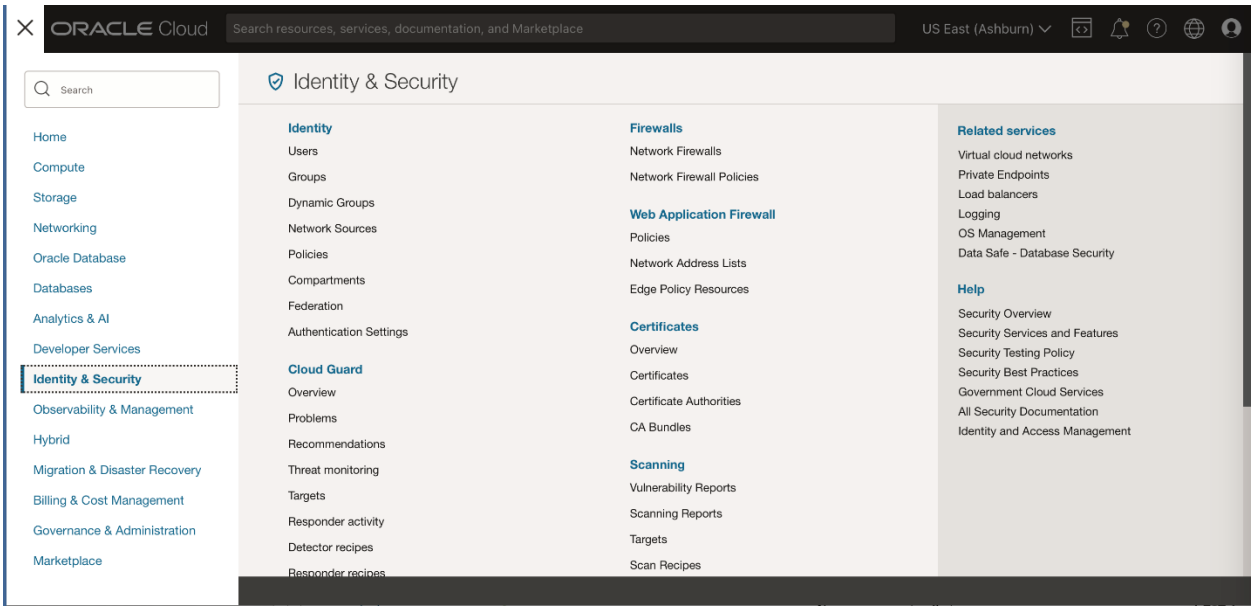
To get the service-account token, you can use a command as below. The command would print the ‘token’ key. You will use this output token later while adding the cluster to the SWIFT.

```
$ kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep "swift-admin" | awk '{print $1}')
```

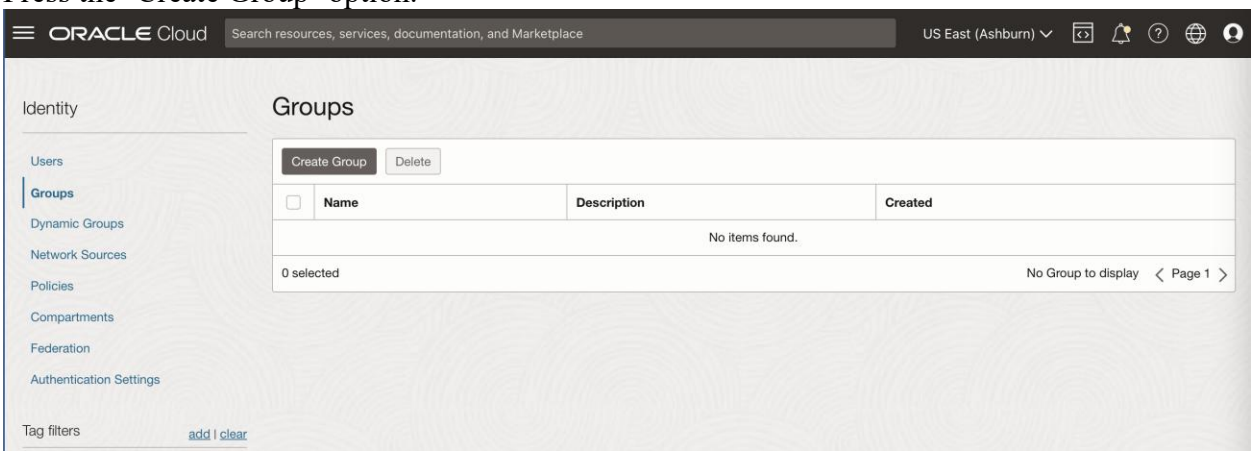
Adding Oracle Cloud Infrastructure (OCI) user for SWIFT use

This section highlights the steps to create an account under your OCI cloud tenancy, which you can use later to configure the cluster details under your installed SWIFT. The same credentials can also be used later to discover an Oracle Cloud Infrastructure Container Registry (OCIR) instance or add an OCI cloud object storage under your SWIFT.

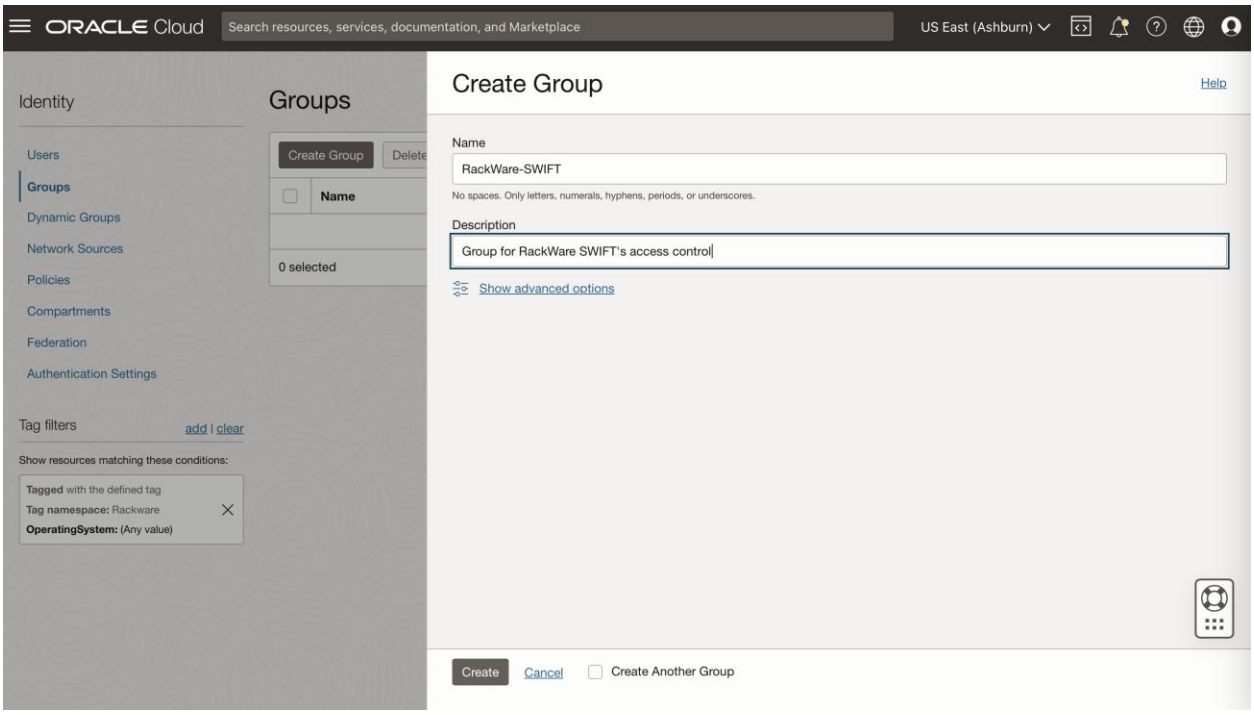
Login to OCI console. Select the ‘Identity & Security’ submenu from the top left menu and then select ‘Groups’ option. We will create a Group, a Policy, and then finally a User.



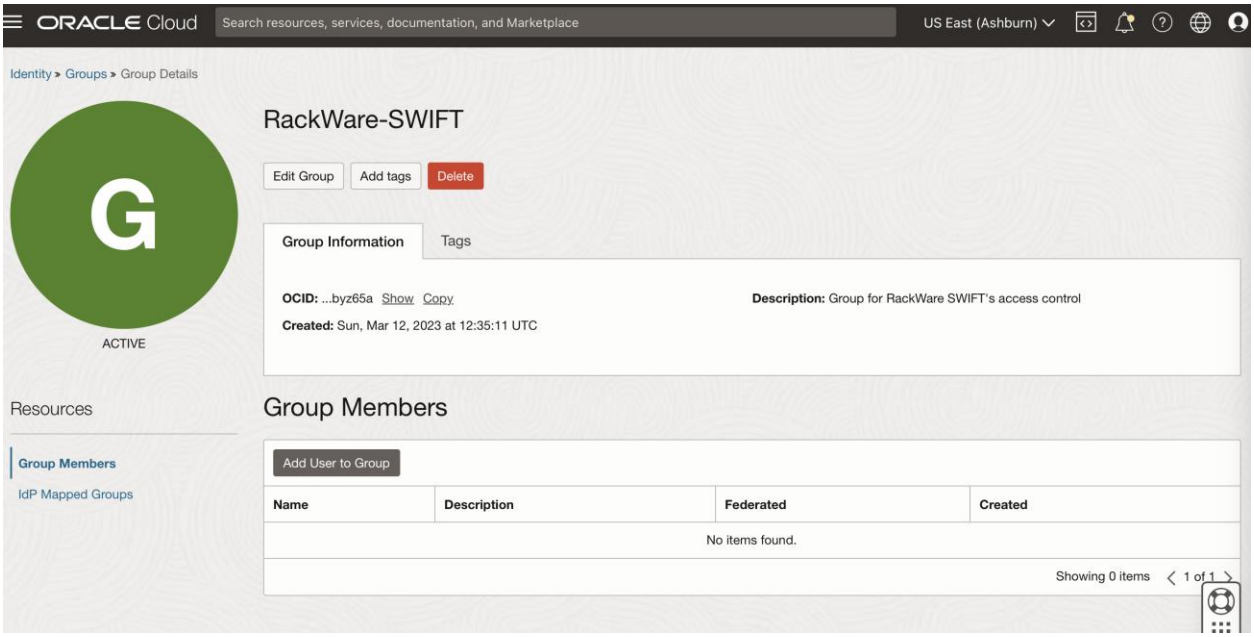
Press the ‘Create Group’ option.



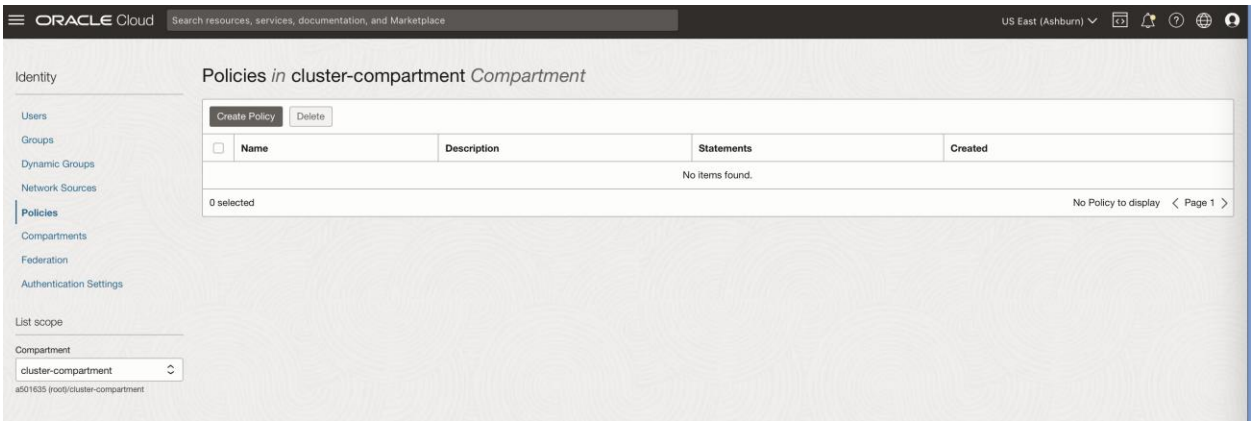
In the new ‘Create Group’ wizard, set appropriate name and description. In this example case, we will name it ‘RackWare-SWIFT.’



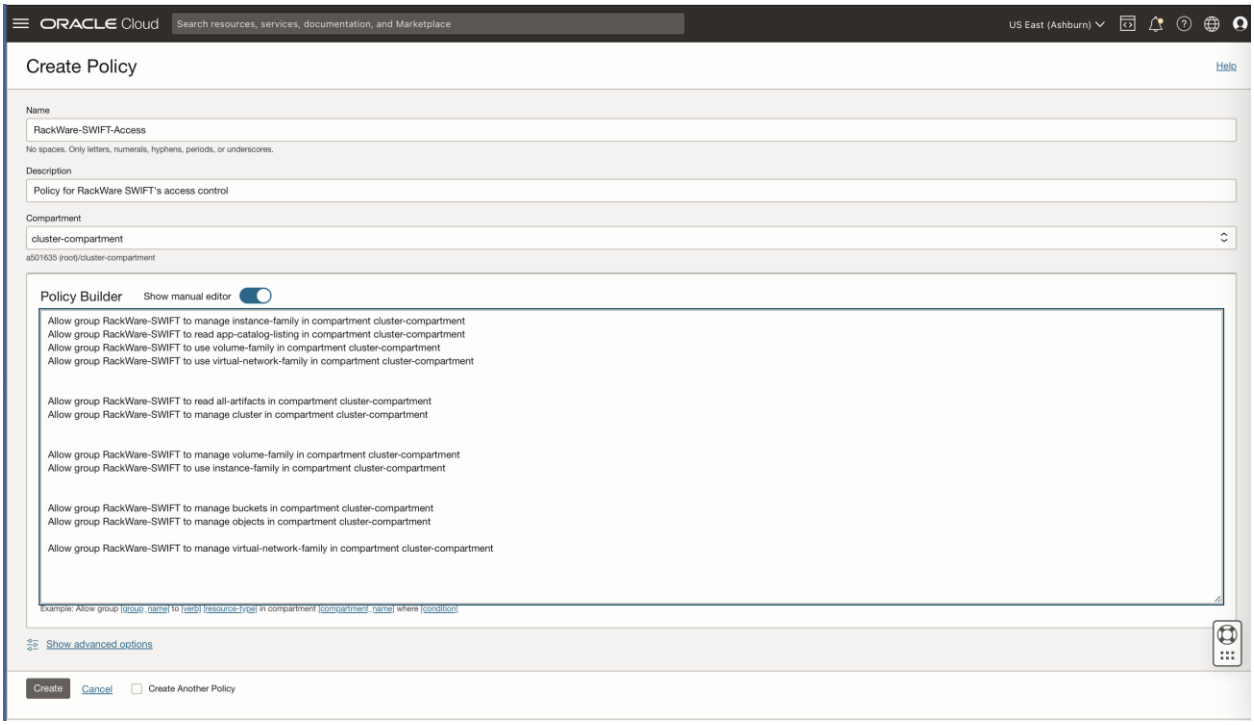
Once the group is created, we are ready for the next step of creating a Policy.



From the Identity menu, select ‘Policies’ submenu. Select compartment where OKE clusters or OCIR registries are located. Then press the ‘Create Policy’ button.



For the new ‘Create Policy’ wizard, give policy a name and description. Then enable ‘Show manual editor’ slider. It will show you a textbox to edit policy rules that you can use to enter rules shown below.



In the policy editor textbox for rules, you will enter below rules. Note that some rules are optional depending on use-cases needed with the SWIFT. Replace ‘{group name}’ and ‘{compartment name}’ in below rules with user group created earlier and compartment name where OKE clusters are located respectively.

Instance access control rules - Mandatory

```
Allow group {group name} to manage instance-family in compartment {compartment name}
Allow group {group name} to read app-catalog-listing in compartment {compartment name}
Allow group {group name} to use volume-family in compartment {compartment name}
Allow group {group name} to use virtual-network-family in compartment {compartment name}
```

Storage access control for snapshots rules - Mandatory

```
Allow group {group name} to manage volume-family in compartment {compartment name}
Allow group {group name} to use instance-family in compartment {compartment name}
```

Sync to/from OKE cluster rules – Mandatory

```
Allow group {group name} to read all-artifacts in compartment {compartment name}
Allow group {group name} to manage cluster in compartment {compartment name}
Allow group {group name} to manage instance-family in compartment {compartment name}
Allow group {group name} to manage volume-family in compartment {compartment name}
Allow group {group name} to use virtual-network-family in compartment {compartment name}
Allow group {group name} to manage objects in compartment {compartment name}
Allow group {group name} to inspect instance-family in tenancy
```

Backup to Object storage control rules – Only needed if you are planning to backup to OCI Object Storage with SWIFT

```
Allow group {group name} to manage volume-family in compartment {compartment name}
Allow group {group name} to manage buckets in compartment {compartment name}
Allow group {group name} to manage objects in compartment {compartment name}
Allow group {group name} to manage virtual-network-family in compartment {compartment name}
```

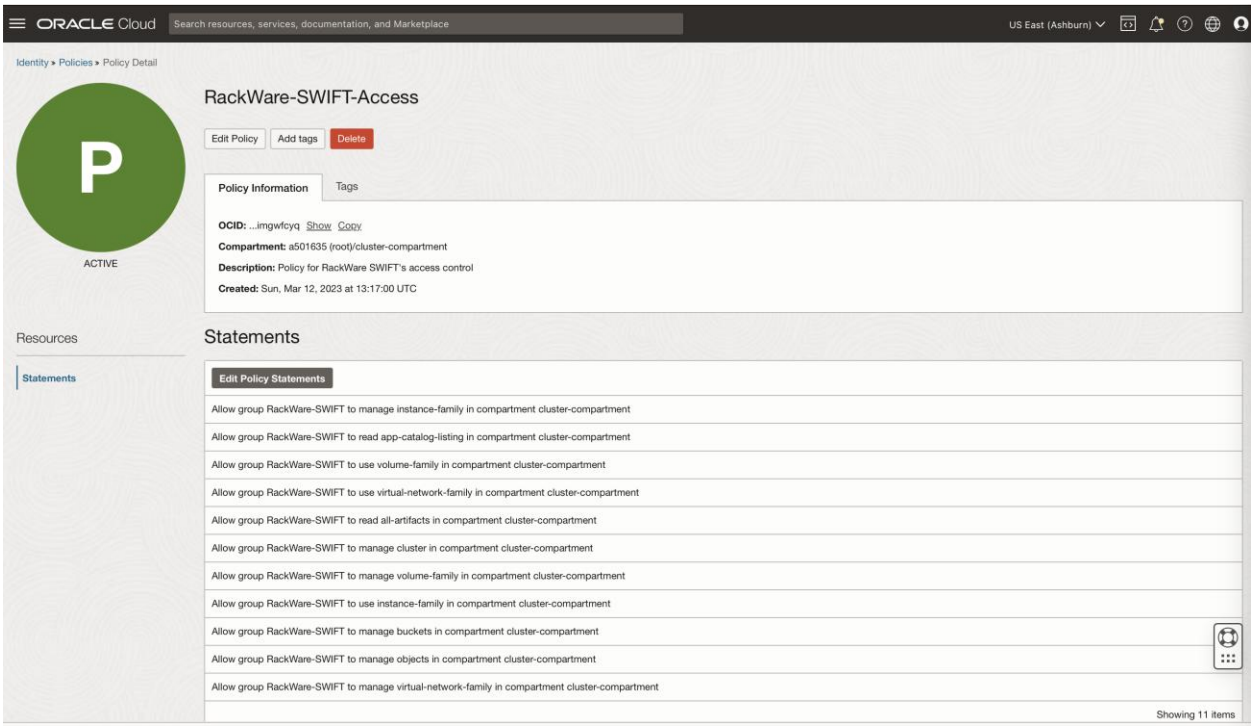
OKE Dynamic cluster provisioning support rules – Only needed if you are planning to dynamically provision DR OKE clusters with SWIFT

```
Allow group {group name} to manage compartments in tenancy
Allow group {group name} to manage vcns in compartment {compartment name}
Allow group {group name} to manage subnets in compartment {compartment name}
Allow group {group name} to use vnics in compartment {compartment name}
Allow group {group name} to use private-ips in compartment {compartment name}
Allow group {group name} to manage public-ips in compartment {compartment name}
Allow group {group name} to use cluster-node-pools in compartment {compartment name}
Allow group {group name} to inspect instance-family in tenancy
Allow group {group name} to manage cluster-family in compartment {compartment name}
```

Oracle Container Registry (OCIR) sync rules – Only needed if you are planning to sync to/from Oracle OCI Container Registries (OCIR) with SWIFT

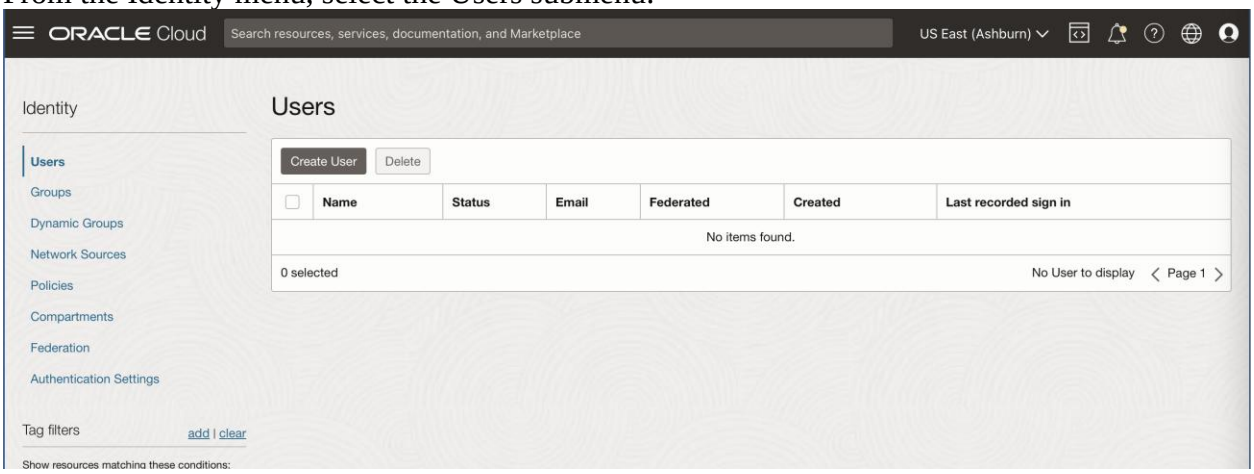
```
Allow group {group name} to manage volume-family in compartment {compartment name}
Allow group {group name} to manage buckets in compartment {compartment name}
Allow group {group name} to manage objects in compartment {compartment name}
Allow group {group name} to manage virtual-network-family in compartment {compartment name}
```

Once you enter required rules above, create the policy.

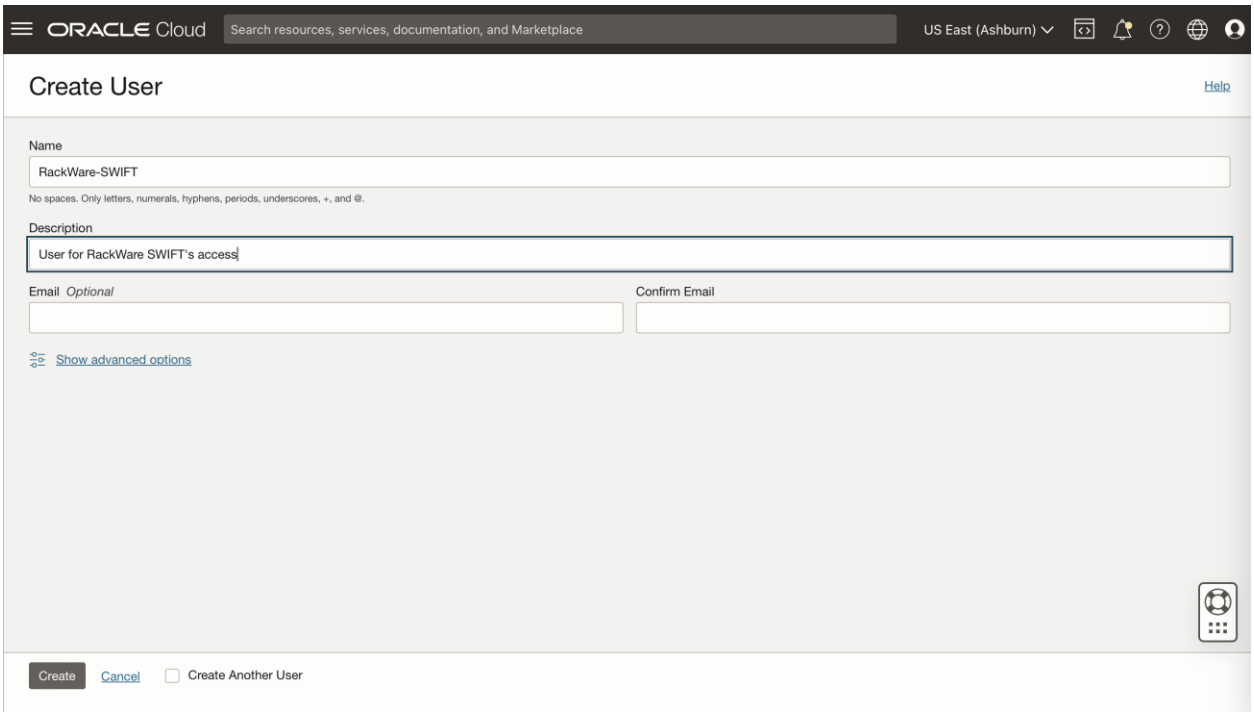


Let's now create a User and add it to the Group created earlier, where we also applied access policy now.

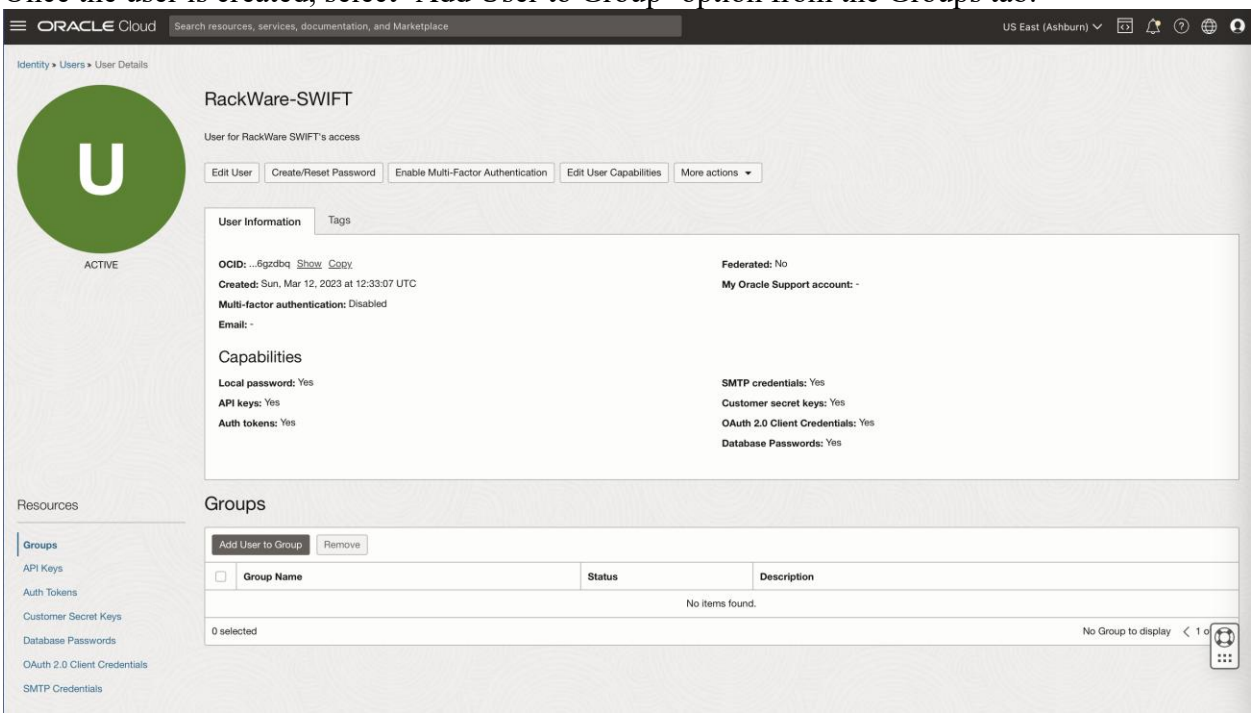
From the Identity menu, select the Users submenu.



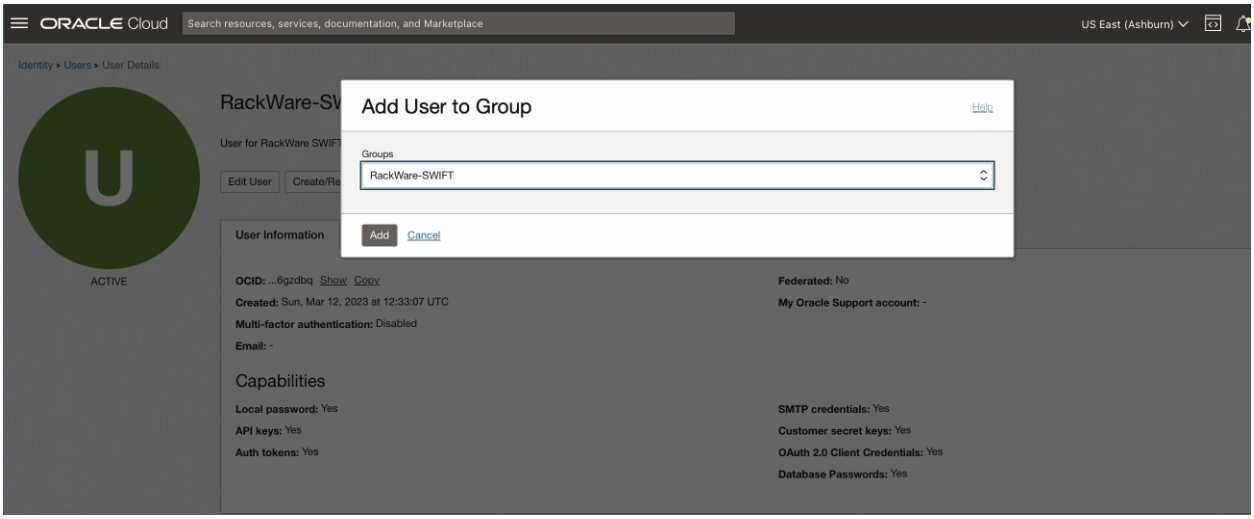
Select the 'Create User' option. Then on new user creation wizard, set name and description for the User. We will use 'RackWare-SWIFT' as a name in the example case.



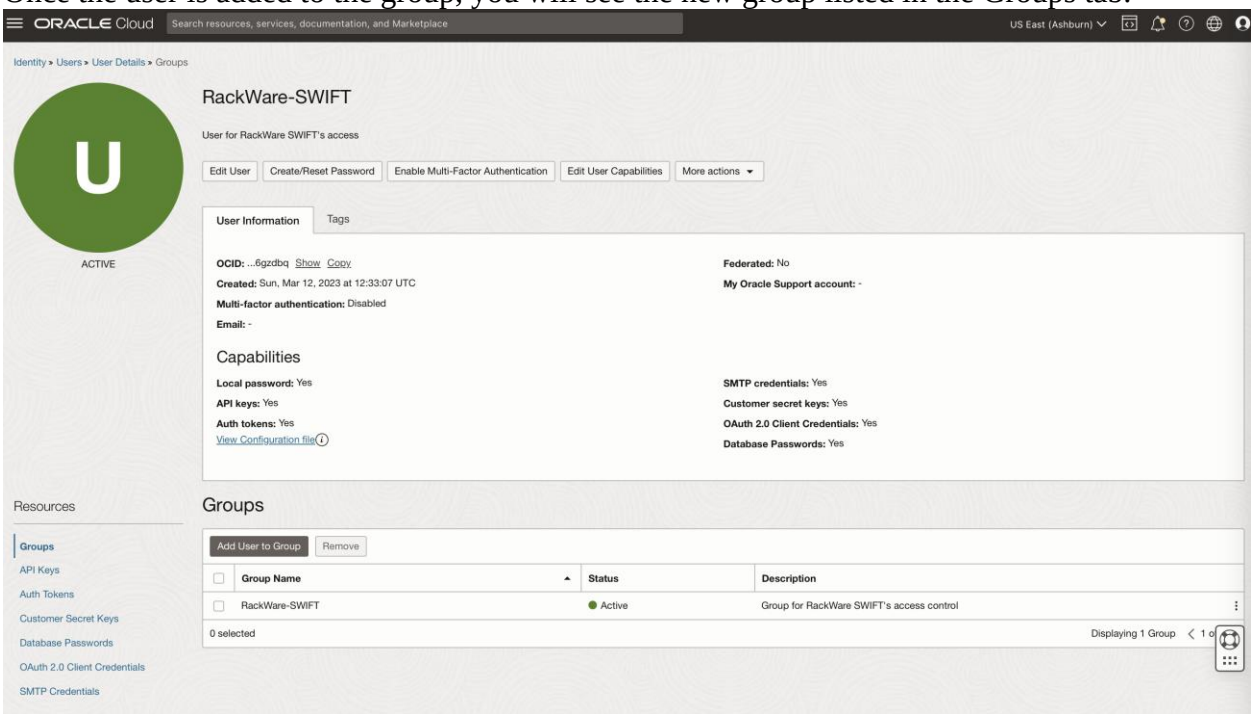
Once the user is created, select 'Add User to Group' option from the Groups tab.



Add it to the 'RackWare-SWIFT' group created earlier in the flow, where new policy is also applied. Adding user to this group would restrict this user's access to OCI with the earlier applied restrictive policy.

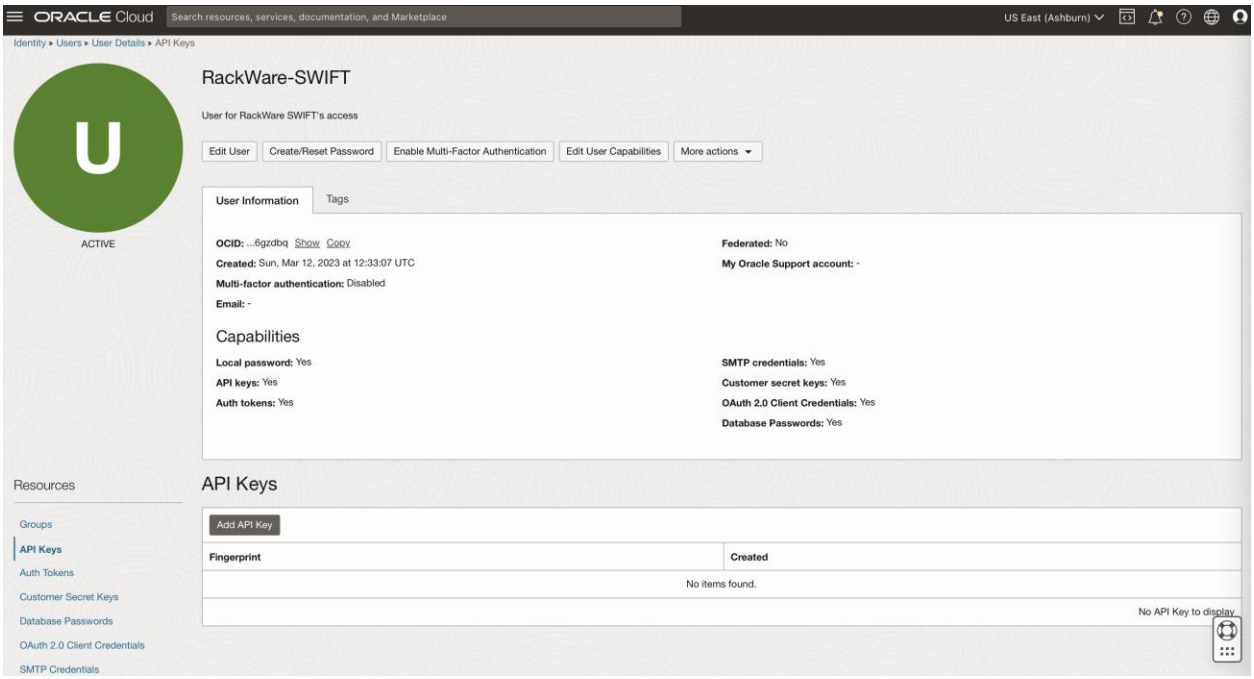


Once the user is added to the group, you will see the new group listed in the Groups tab.

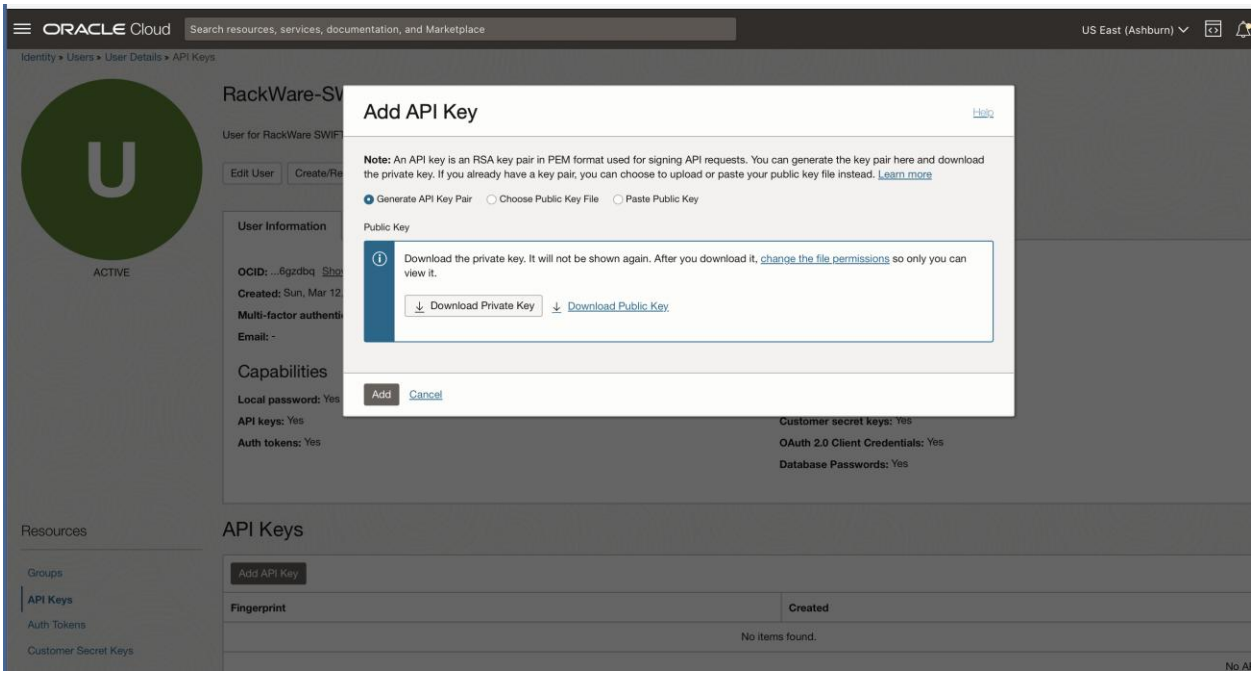


Now, let's generate an API key for the user that later you will use with SWIFT for OKE cluster discovery, OCIR syncs, OCI object storage addition to SWIFT, or OKE dynamic provisioning configuration in SWIFT.

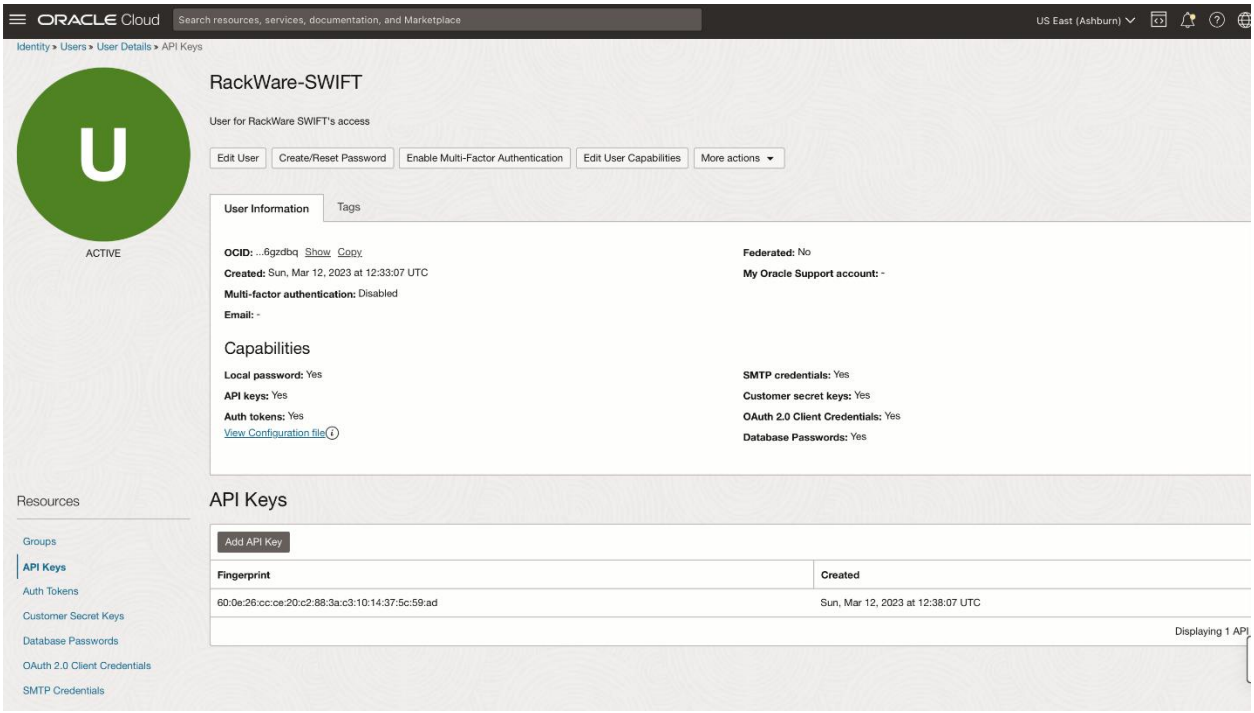
Click on the 'API Keys' tab under user configuration page.



Select ‘Add API Key’ button. On the new key wizard, you can either upload keypair or let OCI generate one for you. Make sure to download both keys if you let OCI generate keypair for you, as you can’t later download these keys later.



Note the generate key’s fingerprint too, as you will later need it for various OKE/OCIR specific configurations in SWIFT, including OKE/OCIR cluster discovery.

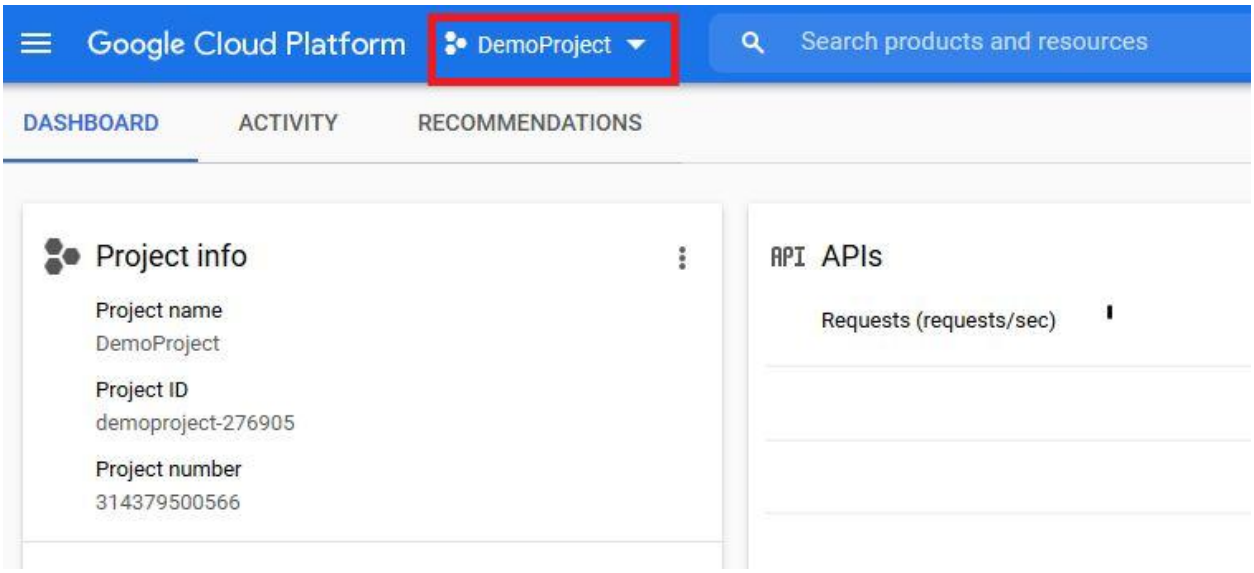


That's it! You can now use the generated API key and fingerprint along with other details like compartment id and user id with SWIFT to discover an OKE cluster or sync OCIR registries under the OCI account. The new API key will have same access that access policy we created and applied above for the new user allows.

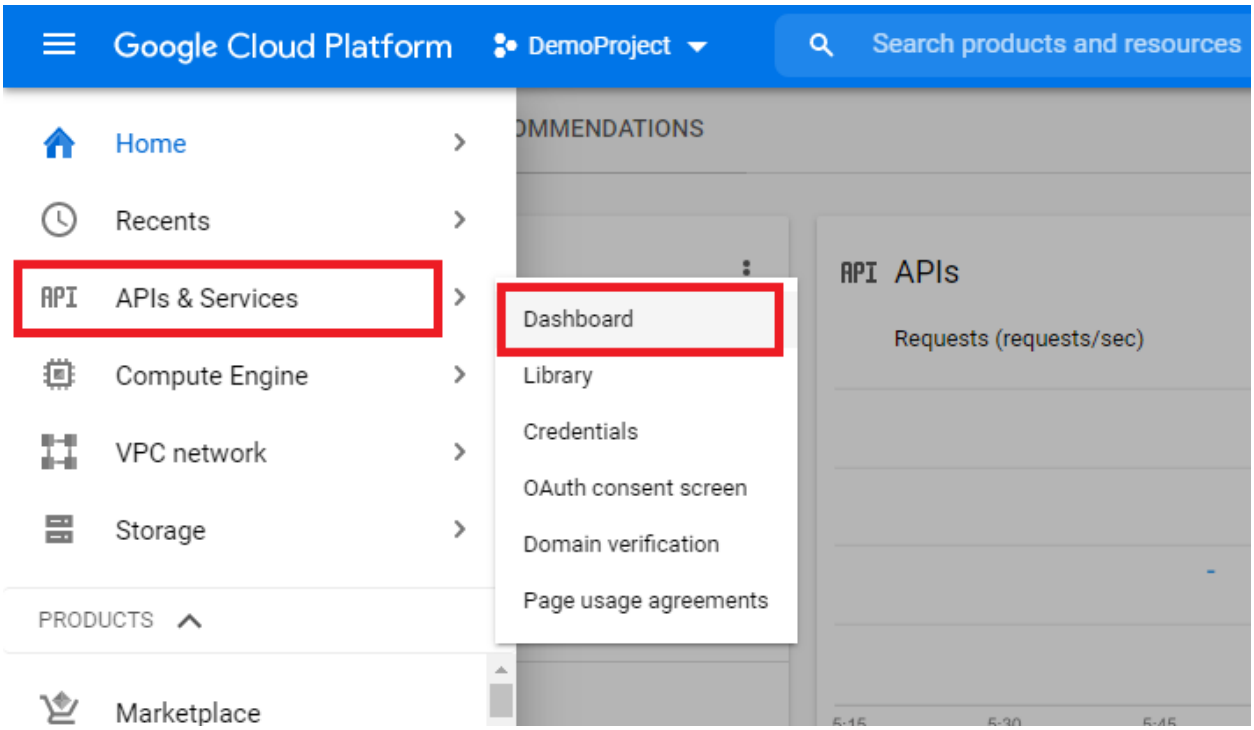
Adding Google GCP service-account for SWIFT use

This section highlights the steps to create an account under your Google cloud project, which you can use later to configure the cluster details under your installed SWIFT. The same credentials can also be used later to discover a Google Container Registry (GCR) instance or add a GCP cloud object storage under your SWIFT.

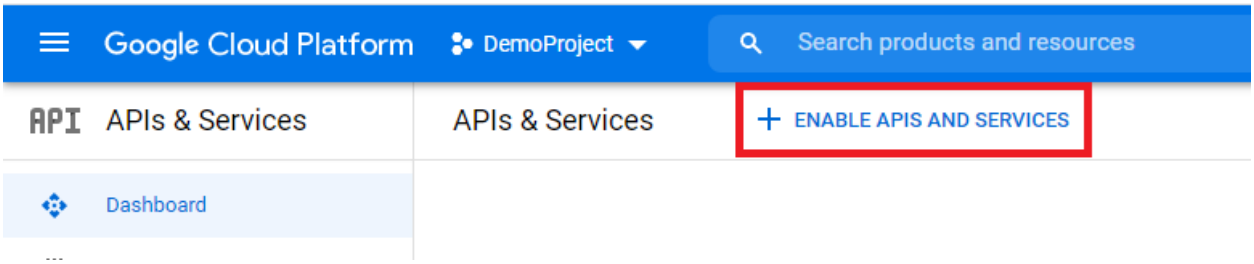
Login to Google cloud console, and select the required project. Note that you will have to repeat these steps for every project, where you have a cluster located and which you want to manage under the SWIFT.



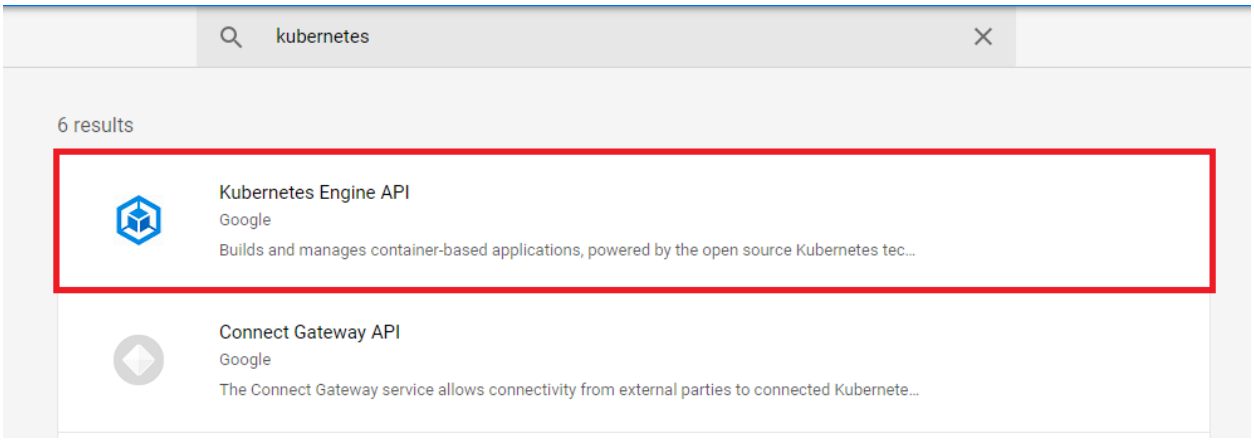
From your menu options, click on the ‘APIs & Services’ option and then on the ‘Dashboard’ submenu.



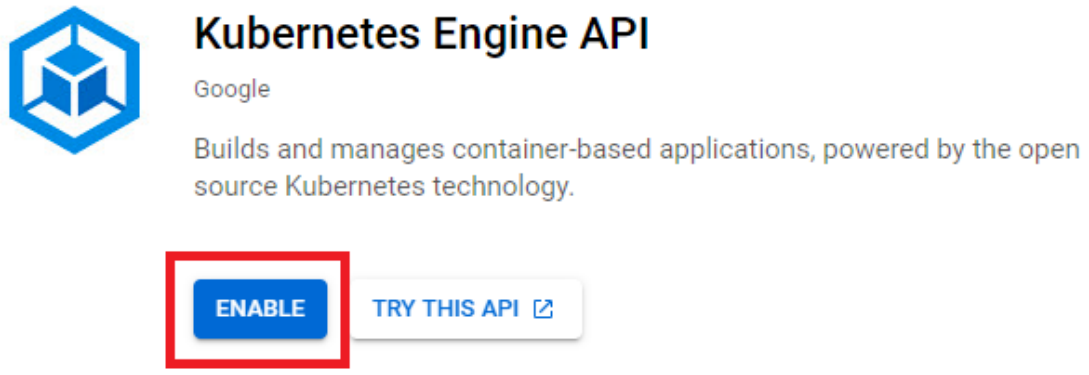
Click on the ‘Enable APIs and Services’ option.



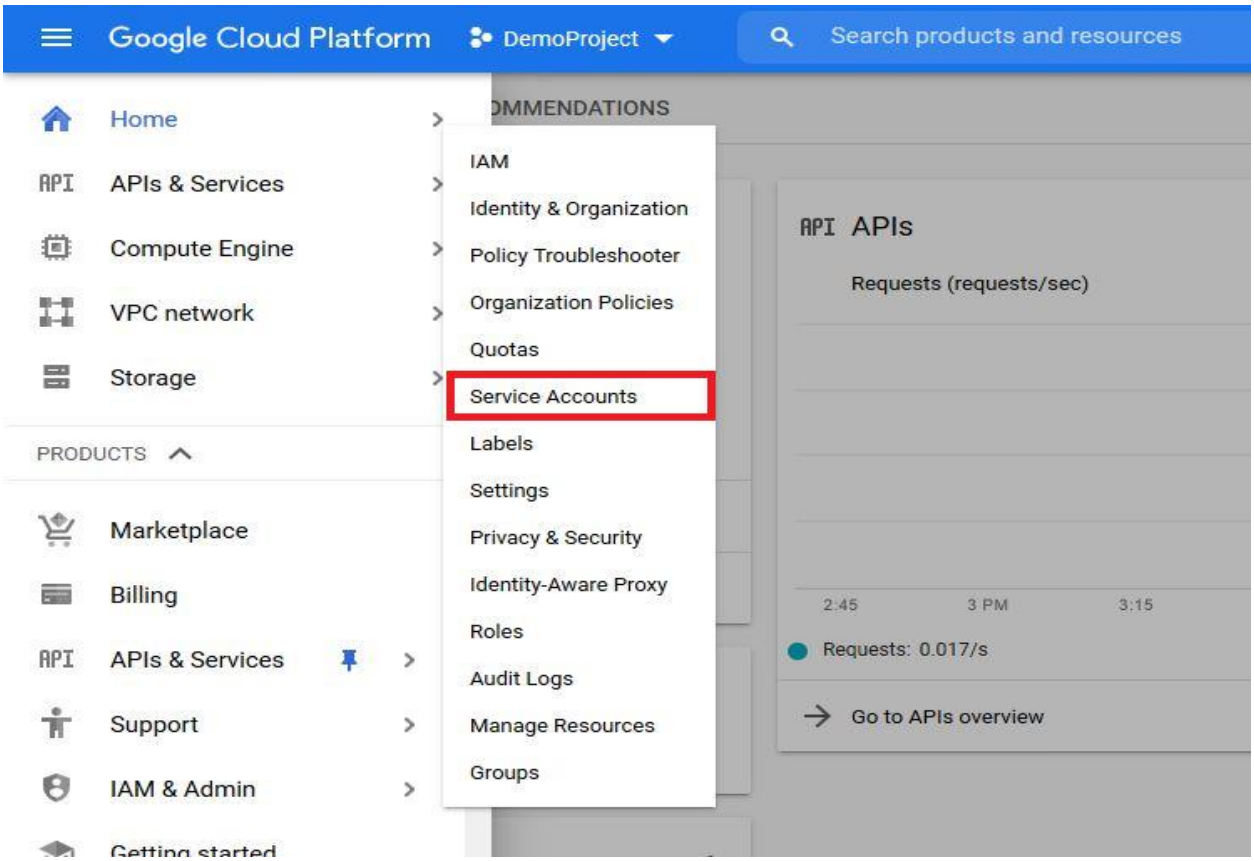
Search For ‘Kubernetes Engine API’ option.



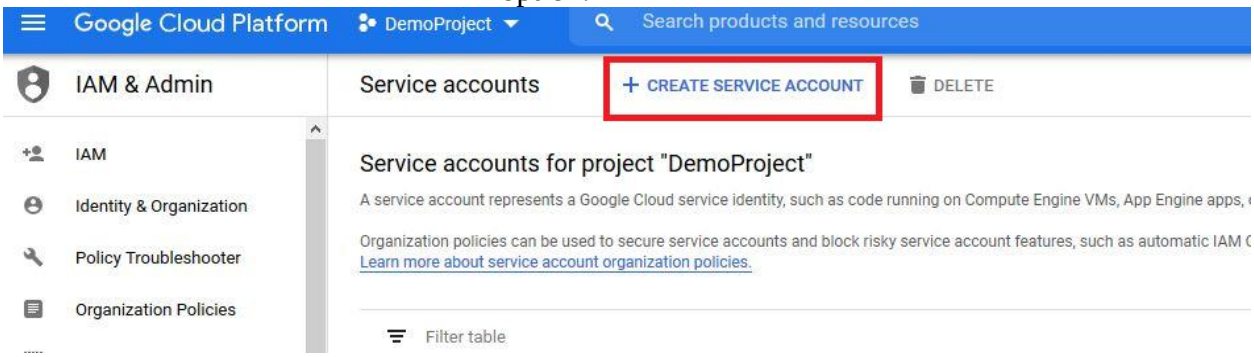
Click on the ‘Enable’ button.



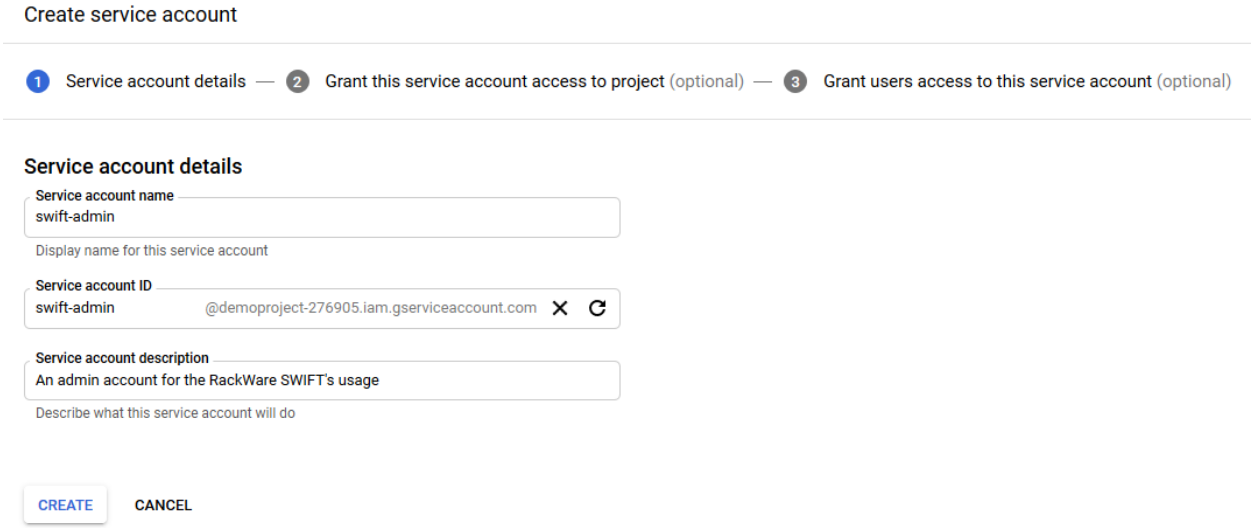
From your menu options, click on the ‘IAM & Admin’ option and then on the ‘Service Accounts’ submenu.



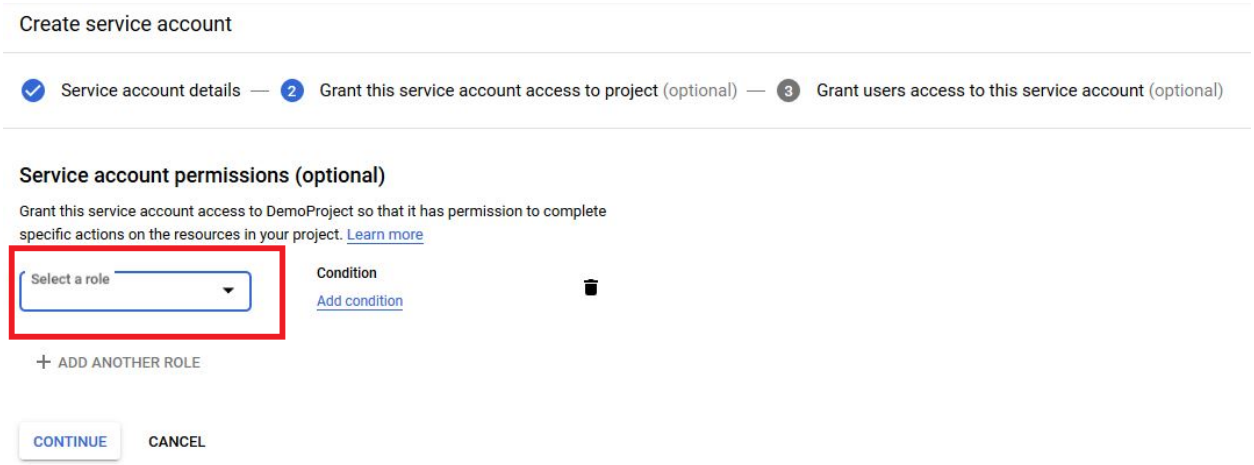
Click on the 'Create Service Account' option.

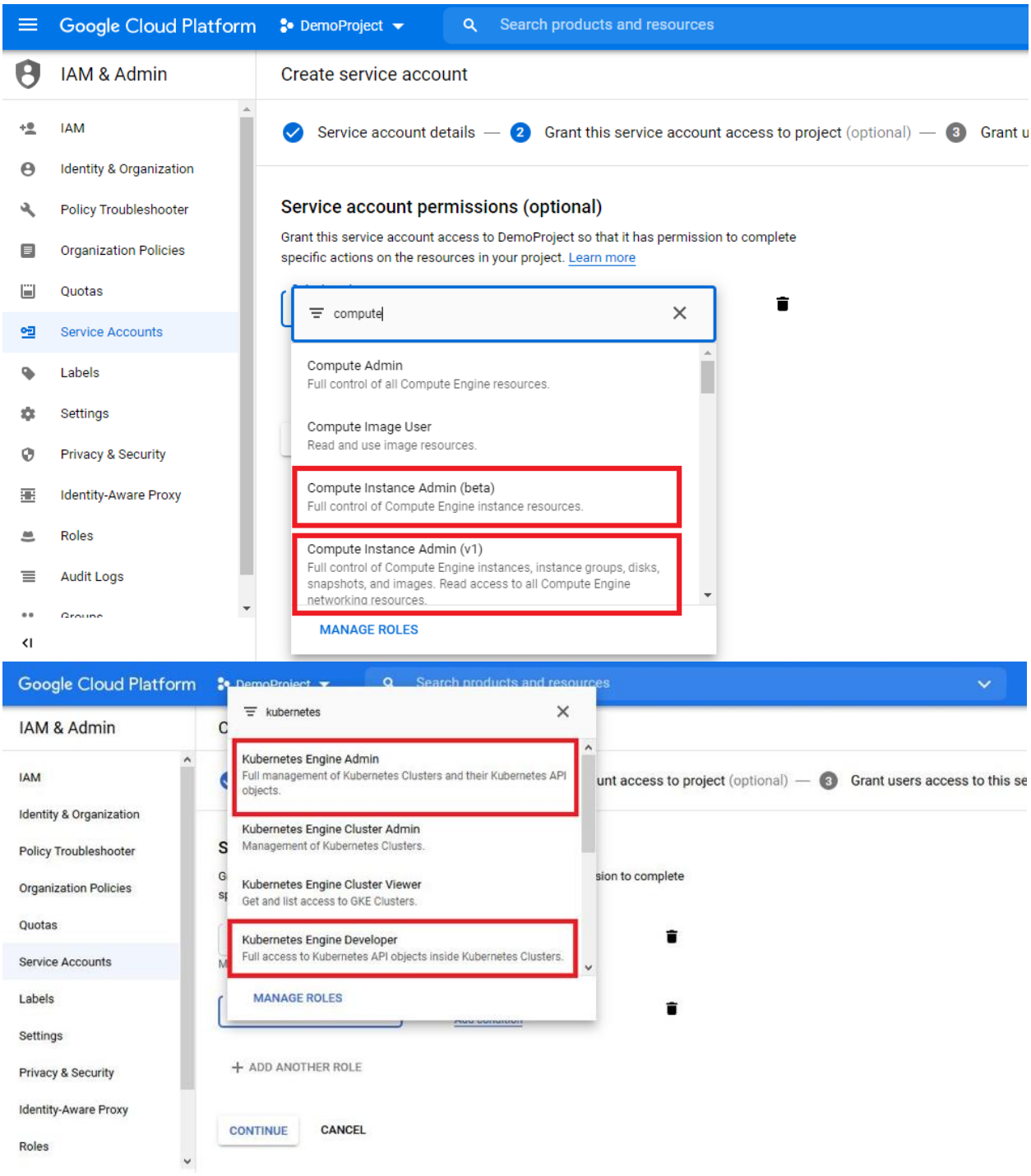


Fill out the necessary account name and description details.

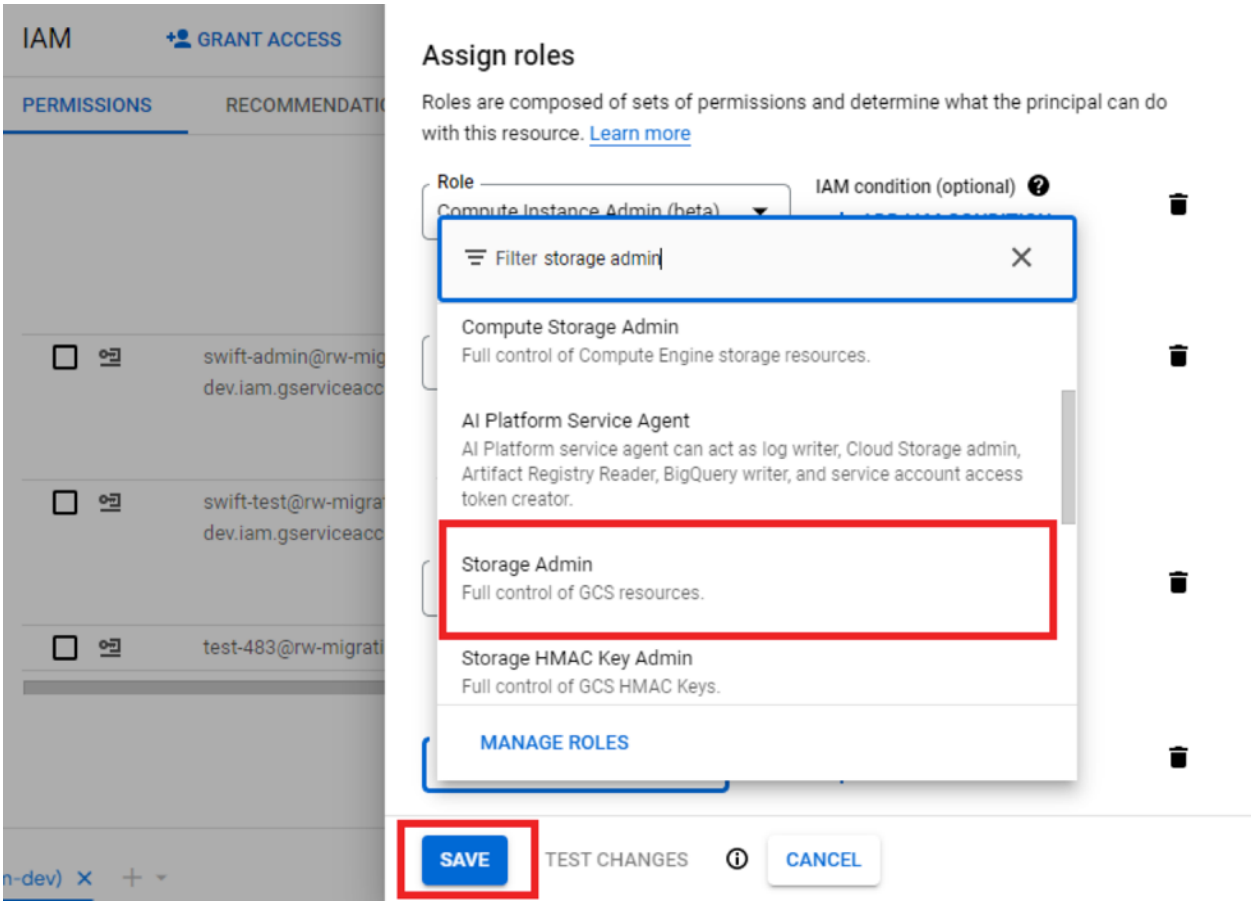


After the account is created, select roles to add for the new account, as shown below. You need to add Kubernetes engine developer, Kubernetes Engine admin, Compute Instance Admin (beta) and Compute Instance Admin (v1).

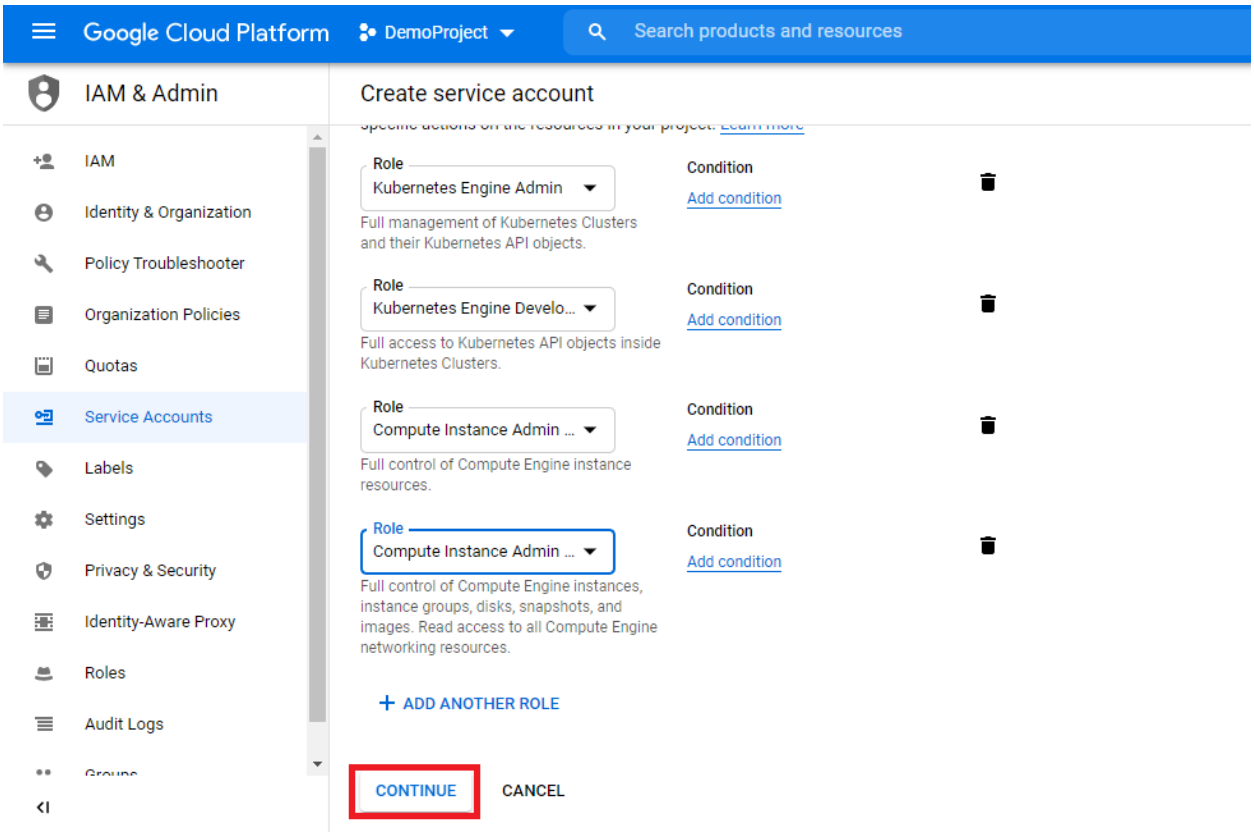




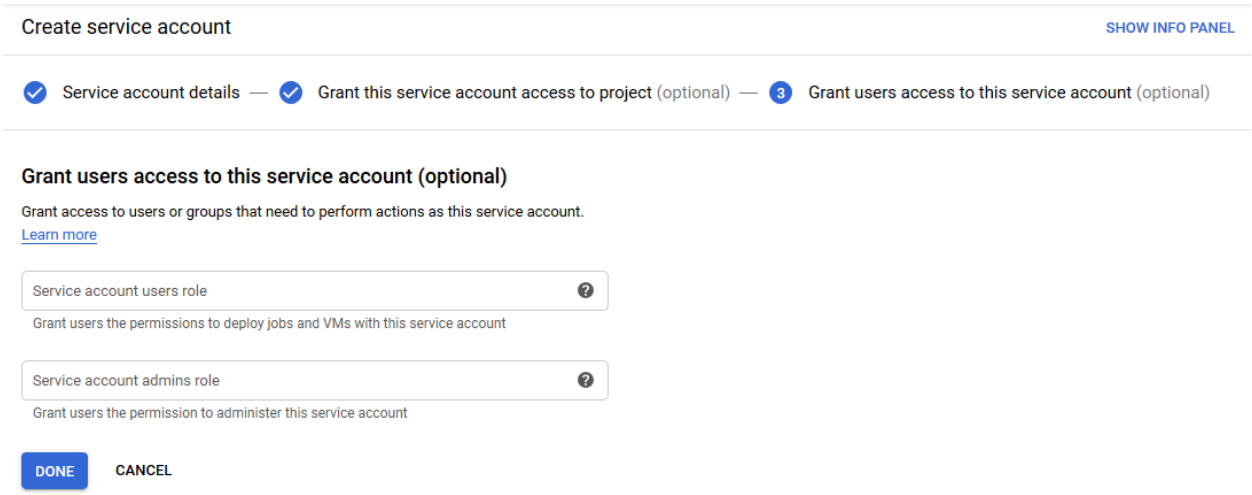
Also add the ‘Storage Admin’ role to the service account if you plan to use this account with the SWIFT for adding a GCP object storage in the SWIFT. If you do not plan to use GCP object storge with SWIFT for backups, then this role addition is an optional step.



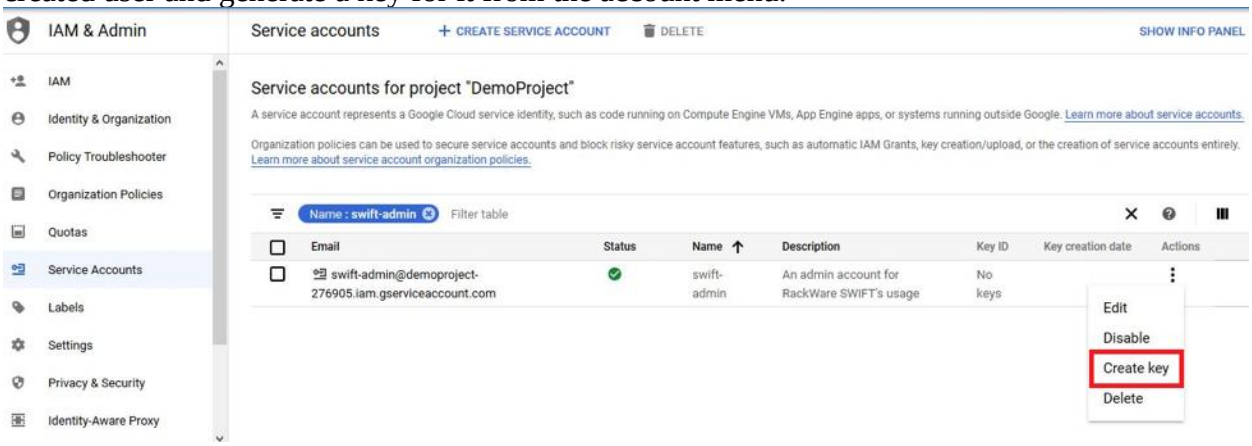
Once the required roles are added, press the ‘Continue’ button.



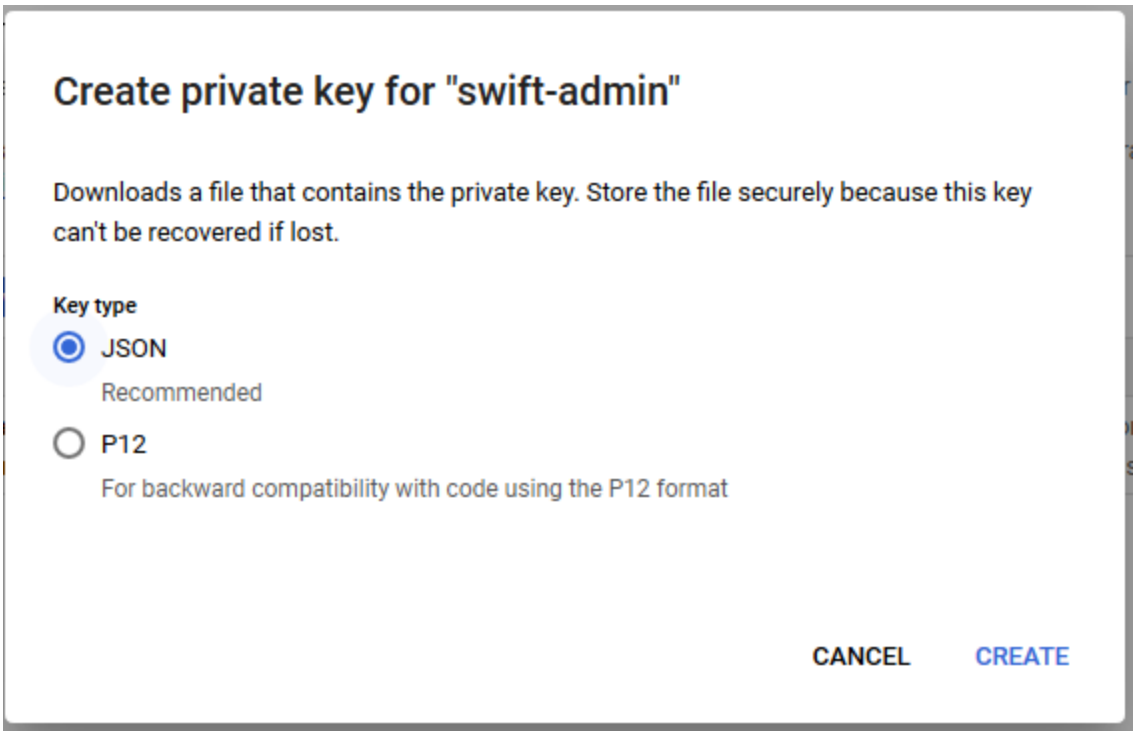
Add any user roles for the account. Adding any user roles is optional for this account. Press the ‘Done’ button to complete the creation of the user.



Now the new account will appear on the service accounts page. Filter, if necessary, to find the newly created user and generate a key for it from the account menu.

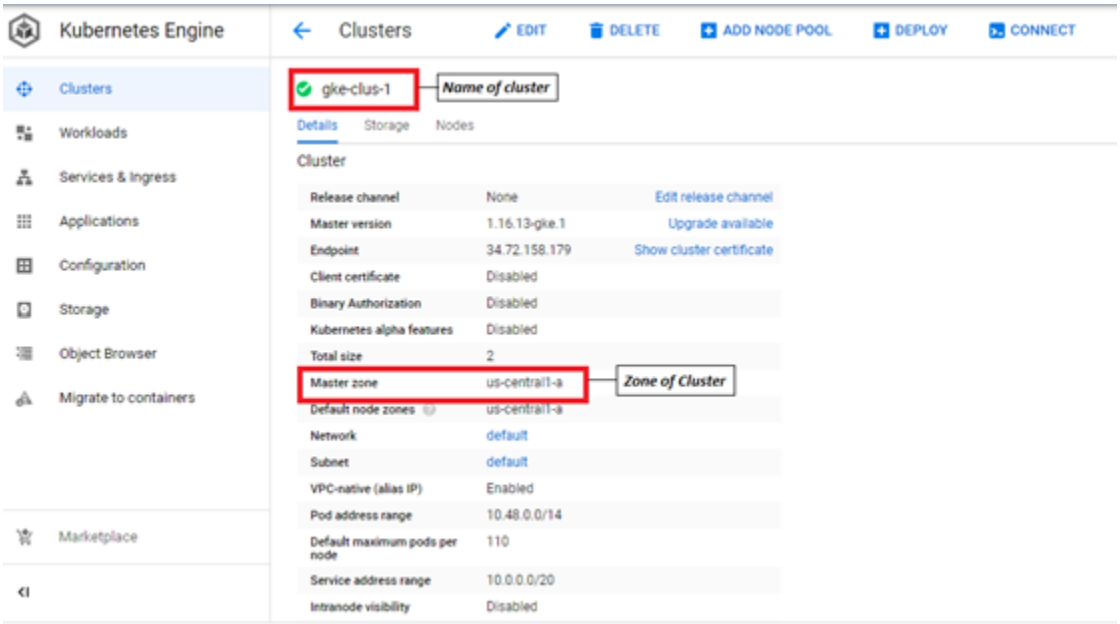


Select the JSON format for the key, and then download the JSON key file.



Store the downloaded JSON file securely, as it will contain API private key and a few other sensitive details. Later, when you are ready to add your cluster to the SWIFT, you will need to upload this downloaded JSON file.

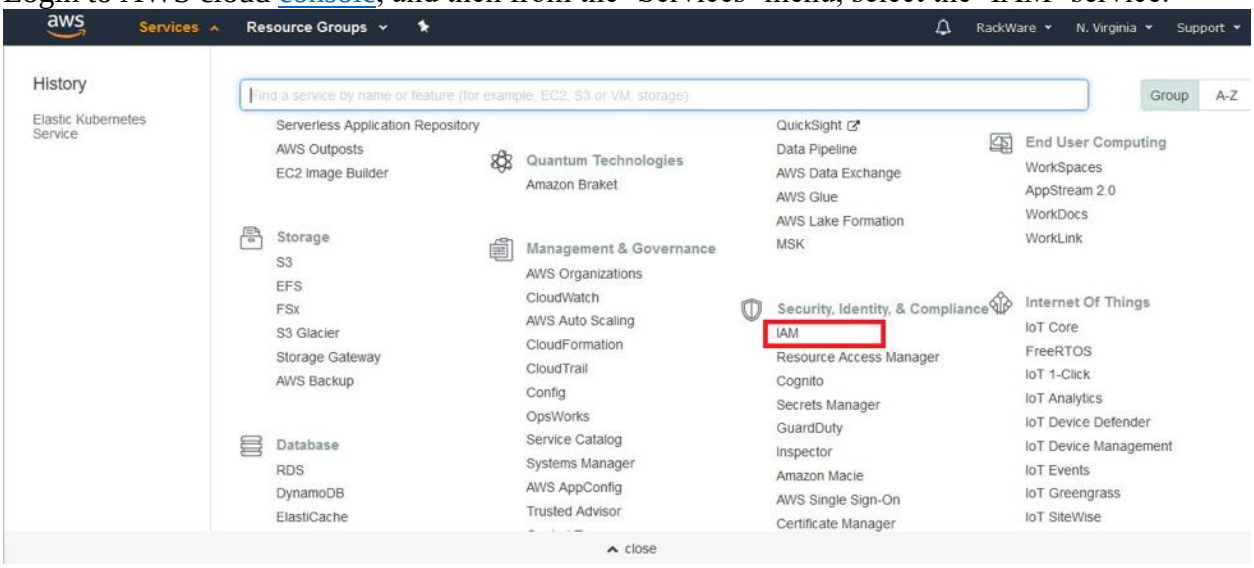
Additionally, you will also need to specify your cloud cluster details to the SWIFT while configuring the GKE cluster under SWIFT. You will need a few necessary details, like the cluster name, the project under which this cluster is created, etc. You can find those on the cluster details page, as shown below.



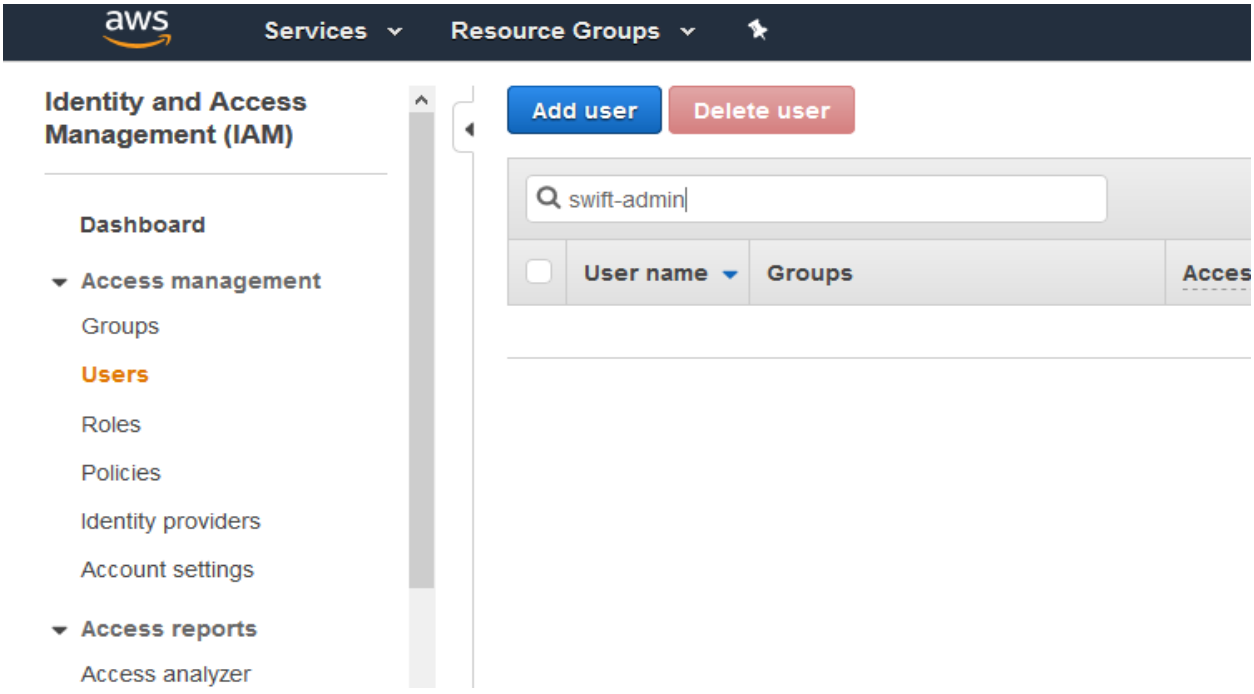
Adding Amazon AWS user for SWIFT use

This section highlights the steps to create an account under your Amazon AWS cloud, which you can use later to configure the EKS cluster details or discover Elastic Container Registry (ECR) instance, or AWS cloud object storage under your installed SWIFT.

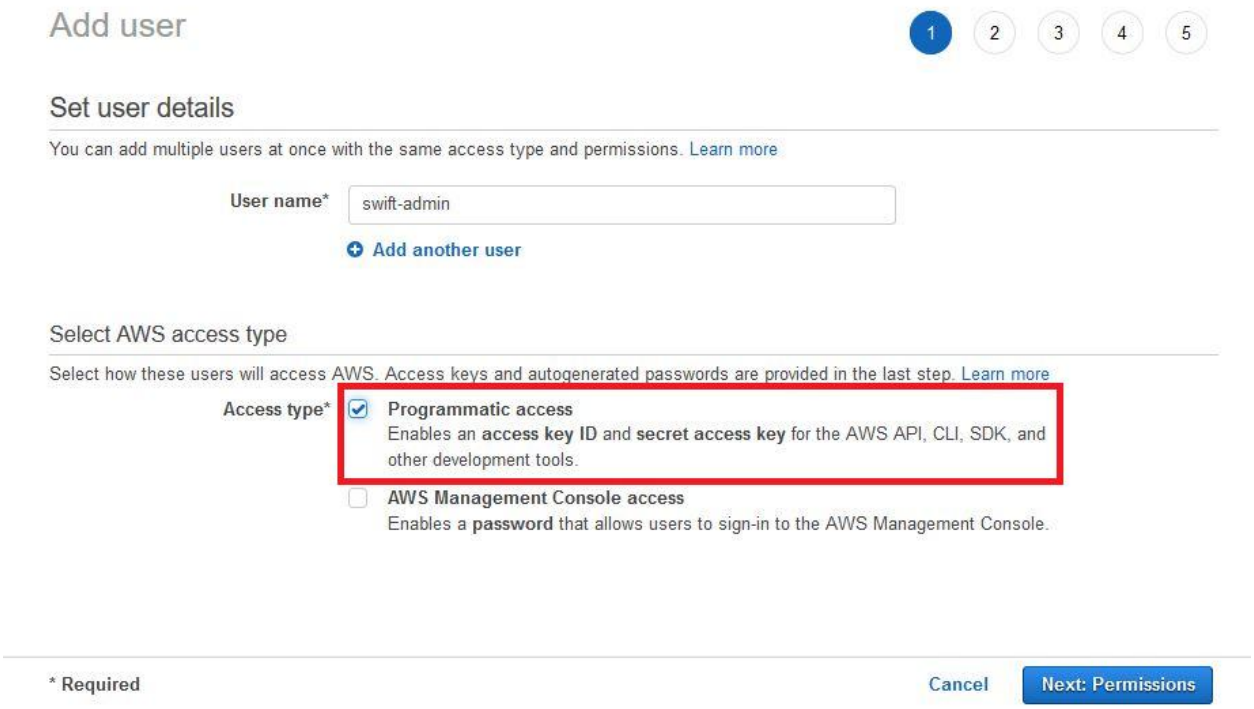
Login to AWS cloud console, and then from the 'Services' menu, select the 'IAM' service.



Traverse to the 'Users' menu on the left and then click on the 'Add User' button.

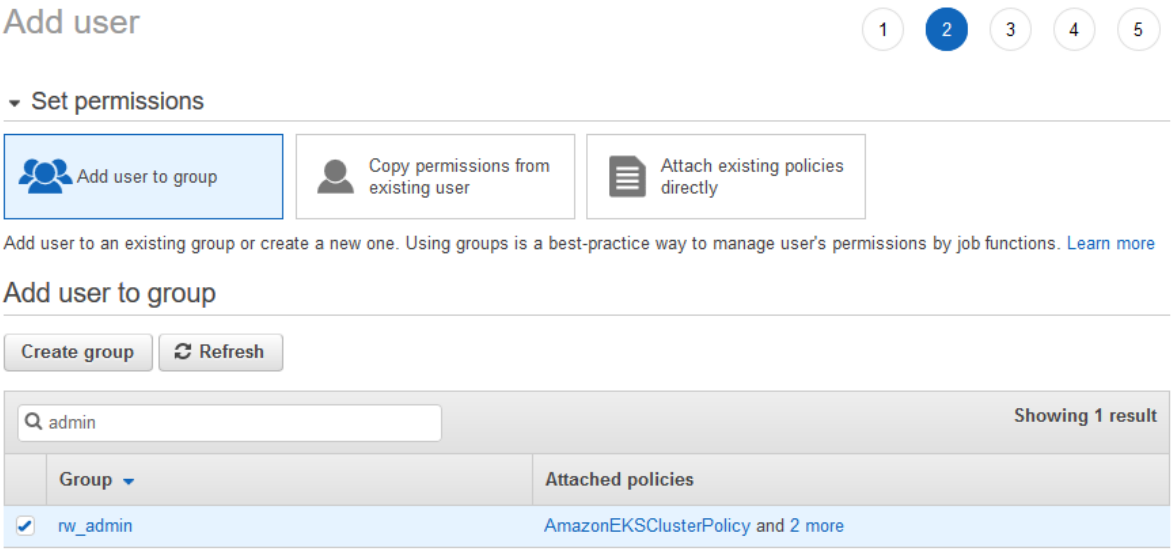


Fill up the user name and then select the 'Programmatic Access' for the access-type.

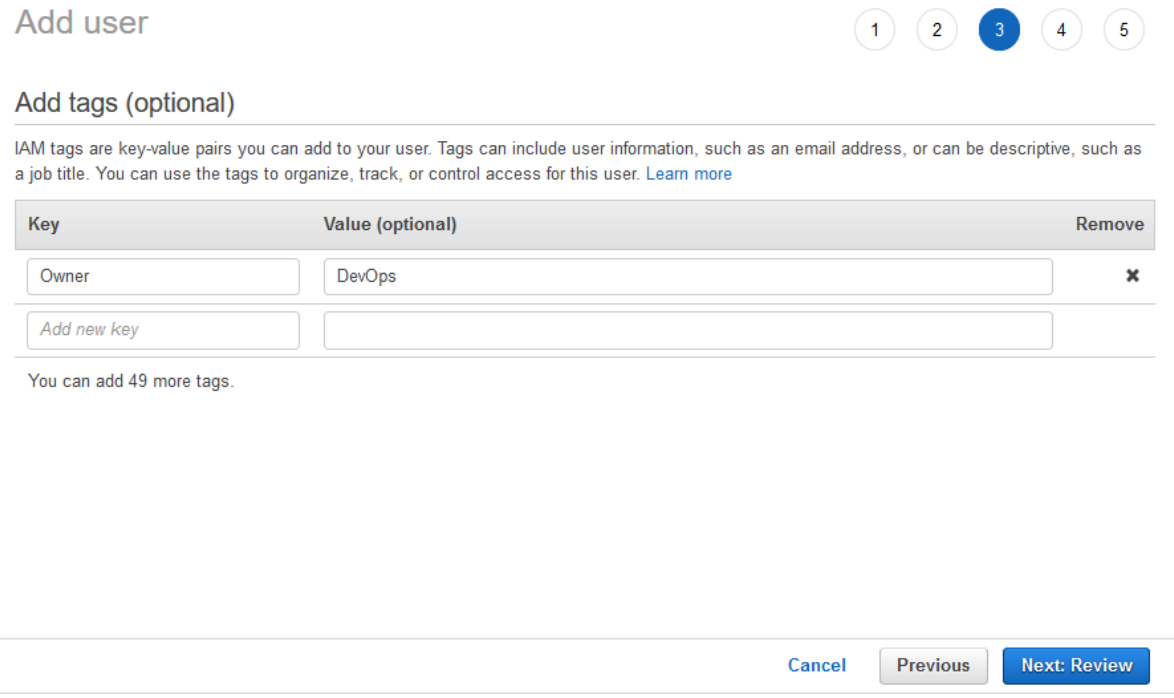


On the next page, select necessary admin groups where this new user would need to be added. In this example case below, the 'rw-admin' group allows access to below essential policies, which in turn enable the admin access to EKS clusters.

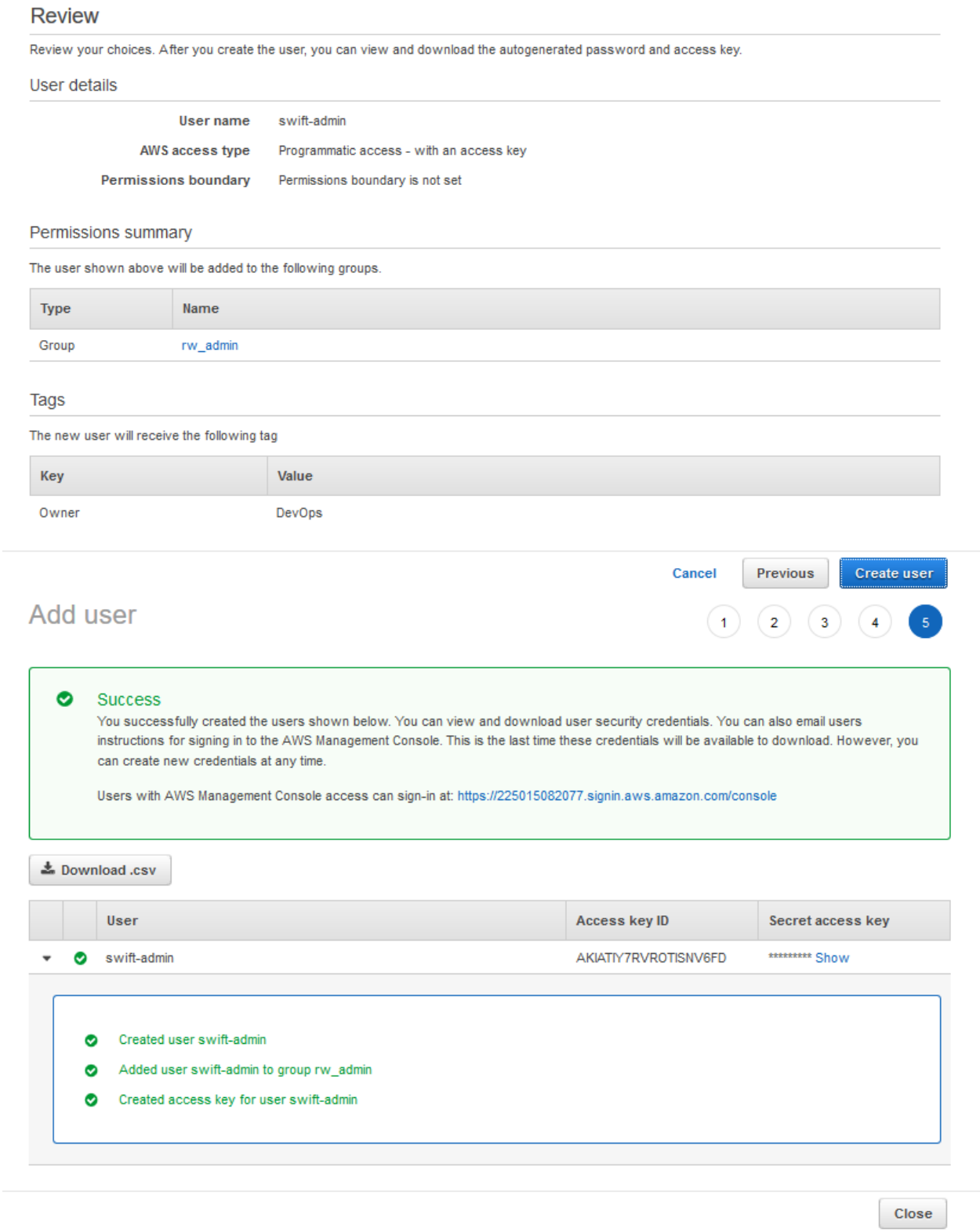
- AmazonEBSCSIDriverPolicy
- AmazonEC2ContainerRegistryFullAccess
- AmazonEKS_CNI_Policy
- AmazonEKSWorkerNodePolicy
- AmazonEKSVPCResourceController
- AmazonEKSClusterPolicy



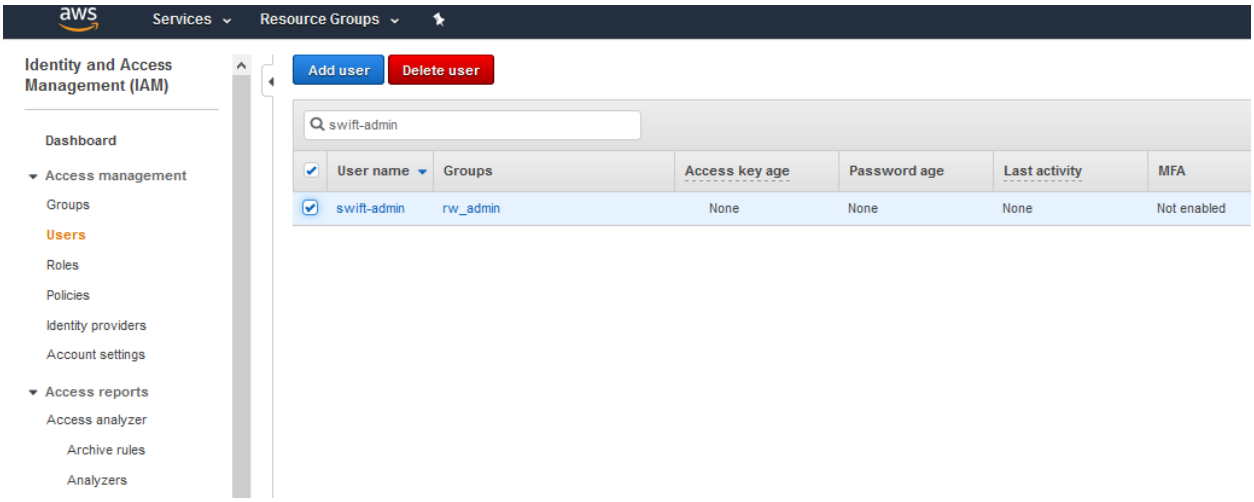
On the next page, select any tags you want to set for the new user.



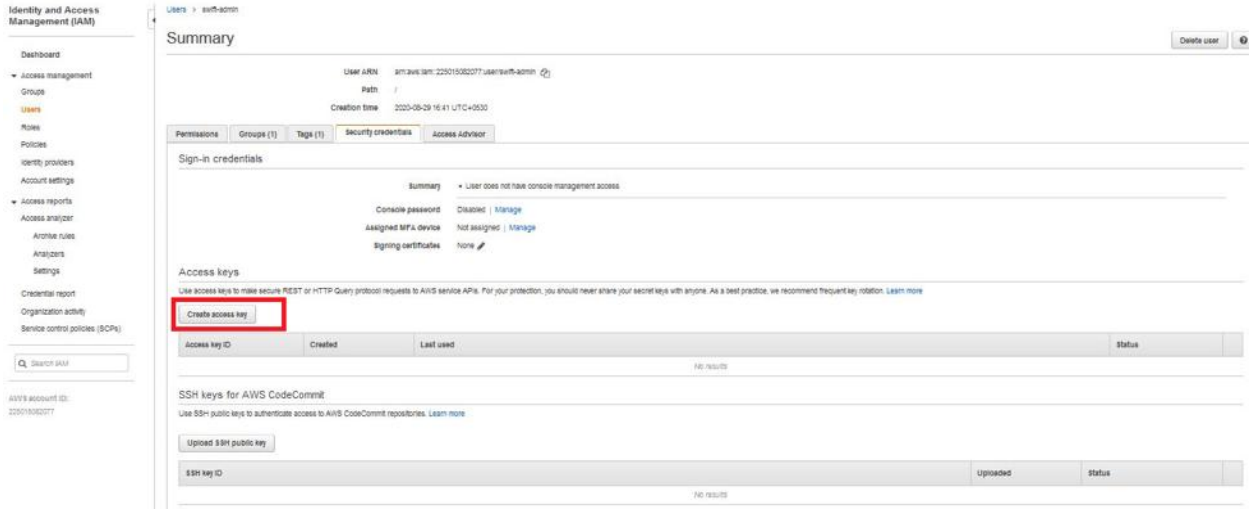
Next, then review all settings and create the user.



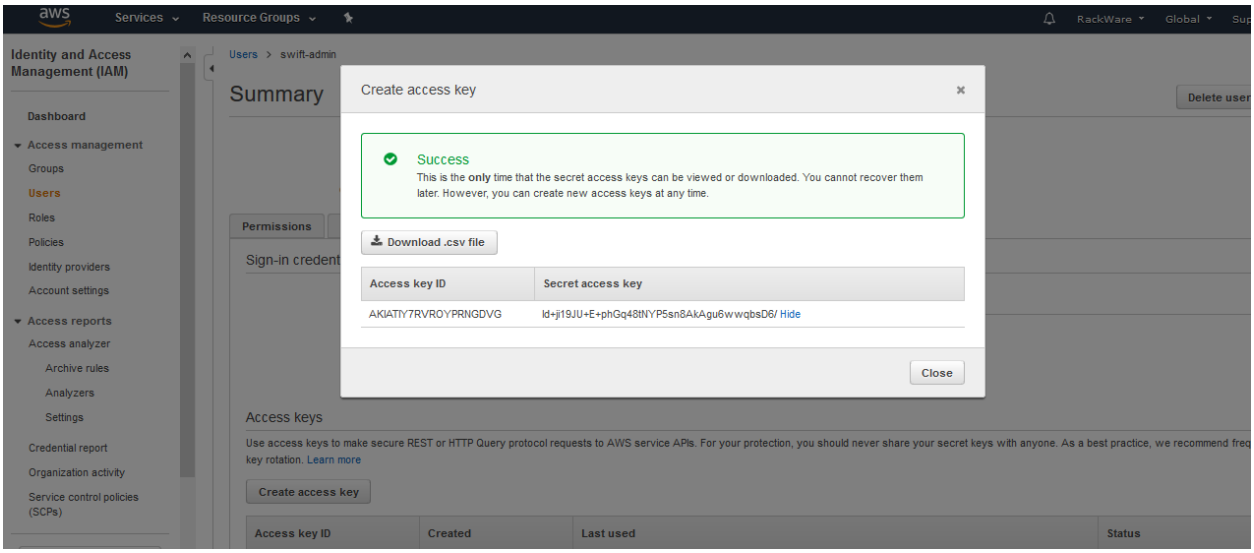
Next, click on the generated user from the 'Users' menu. You will create an access key for it.



Traverse to the 'Security Credentials' tab under user details. You would use the 'Create access key' button to generate a new access key.



Note the generated access key and store it securely. You will need it later while configuring your EKS cluster details under the SWIFT.



Note:
Once you create a new user and set its credentials/access-key, you will also need to whitelist this user for access to the Kubernetes (EKS) service instance, which is managed with the SWIFT. To do that, you have to edit the ConfigMap named 'aws-auth,' which is located under the 'kube-system' namespace on your EKS instance, and then add the ARN of the newly created user there in the below format:

```
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::225015082077:role/eksctl-EKSCluster1-nodegroup-ng-a-NodeInstanceRole-1QTOCK6OPERZS
      username: system:node:{{EC2PrivateDNSName}}
  mapUsers: |
    - userarn: arn:aws:iam::225015082077:user/swift-admin
      username: swift-admin
      groups:
        - system:masters
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
```

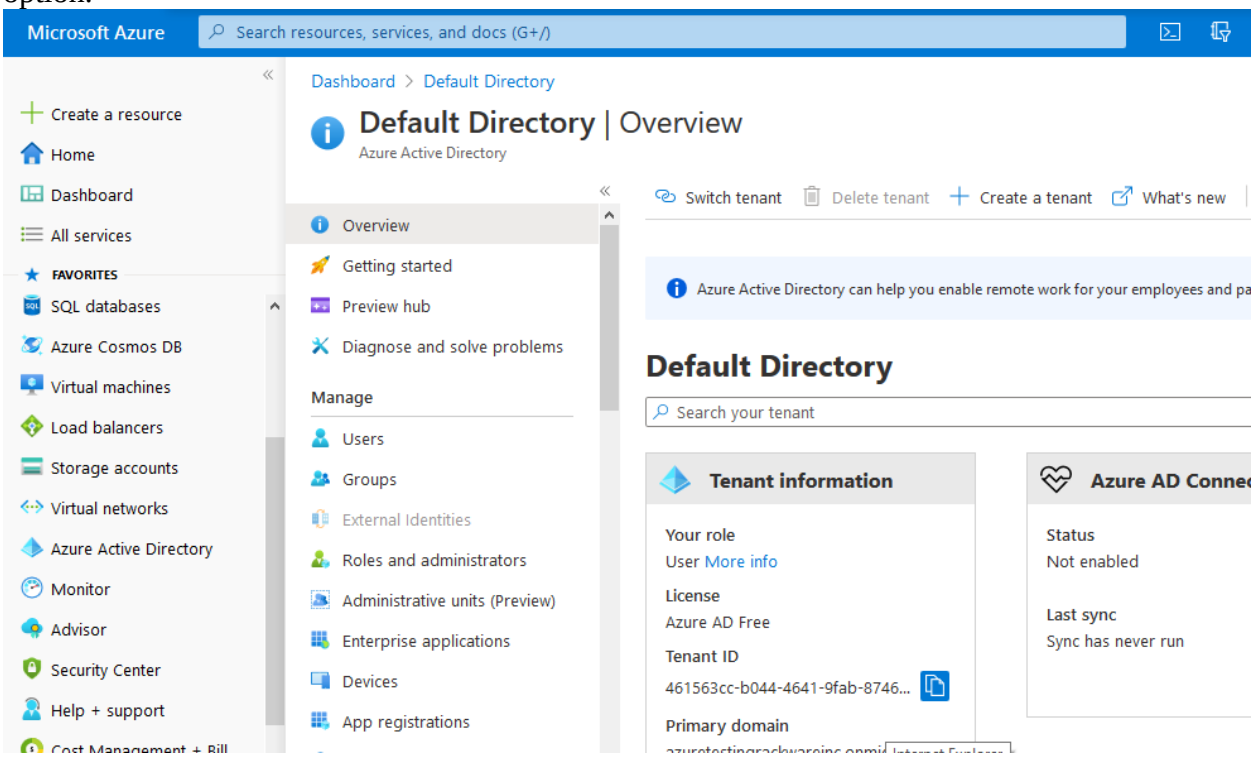
You will need to do the ConfigMap edit step above using the existing cluster-creator user's credentials. You can use the kubectl utility (along with cluster creator credentials) to do the edit. The detailed steps can be found in this KB.

Note that you will have to repeat this whitelisting step for every EKS instance, which you want the SWIFT to manage with the newly created user. If you skip this step, your newly created user will not be able to access the corresponding EKS service instance.

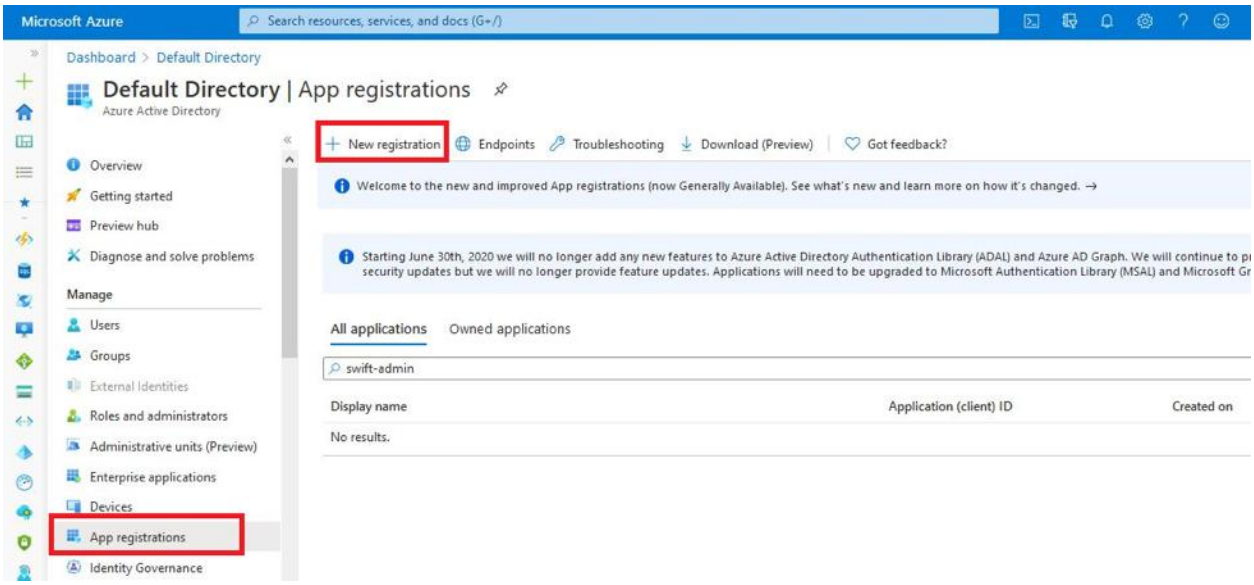
Adding Azure AAD application for SWIFT use

This section highlights the steps to create an AAD application under your Azure cloud. You will later use the AAD application credentials to configure the AKS cluster details under your installed SWIFT. The same credentials could also be used for Azure Container Registry (ACR) or Azure Object Storage discovery in SWIFT.

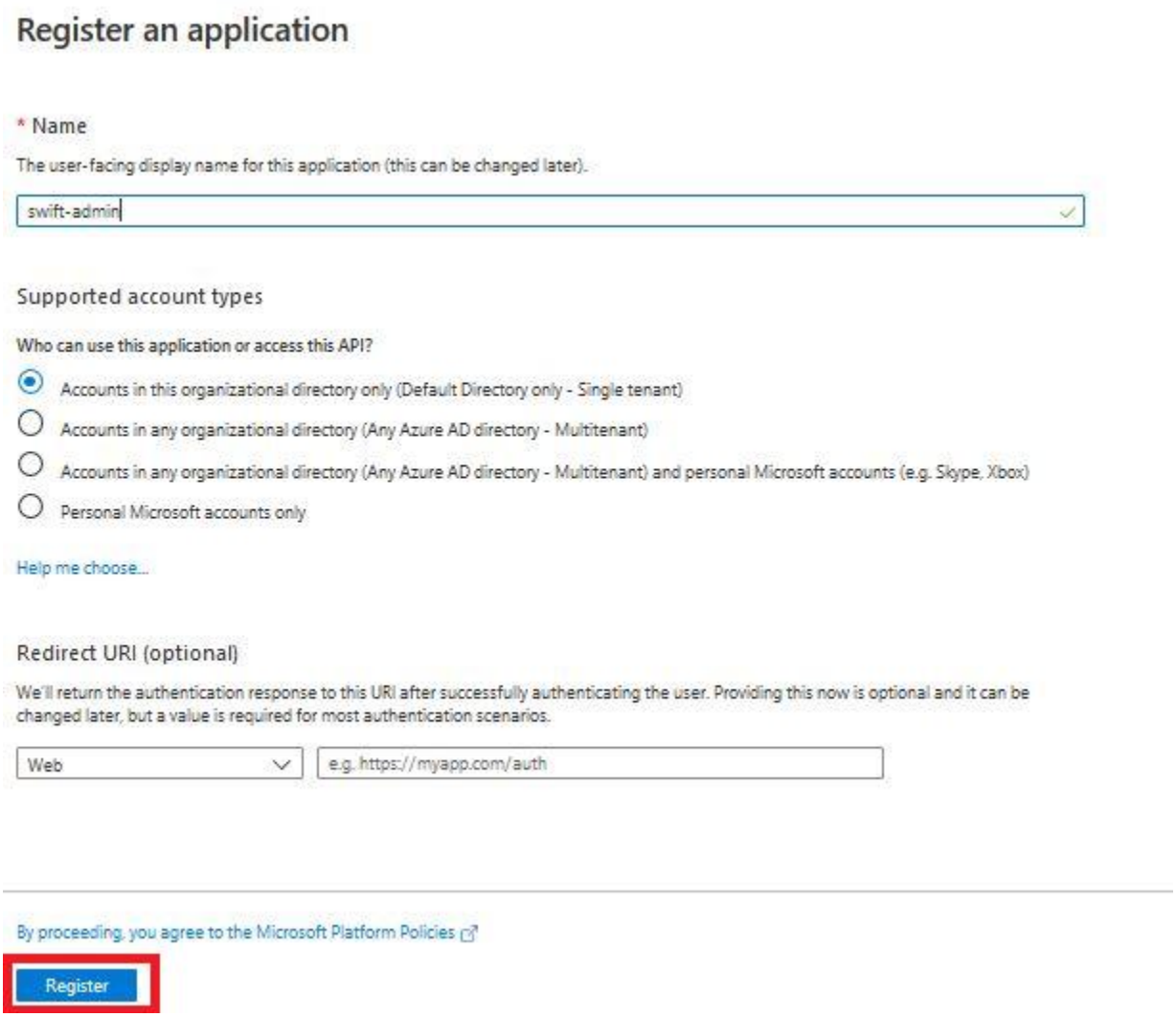
Log in to the Azure cloud console, and from menus on the left, select the 'Azure Active Directory' menu option.



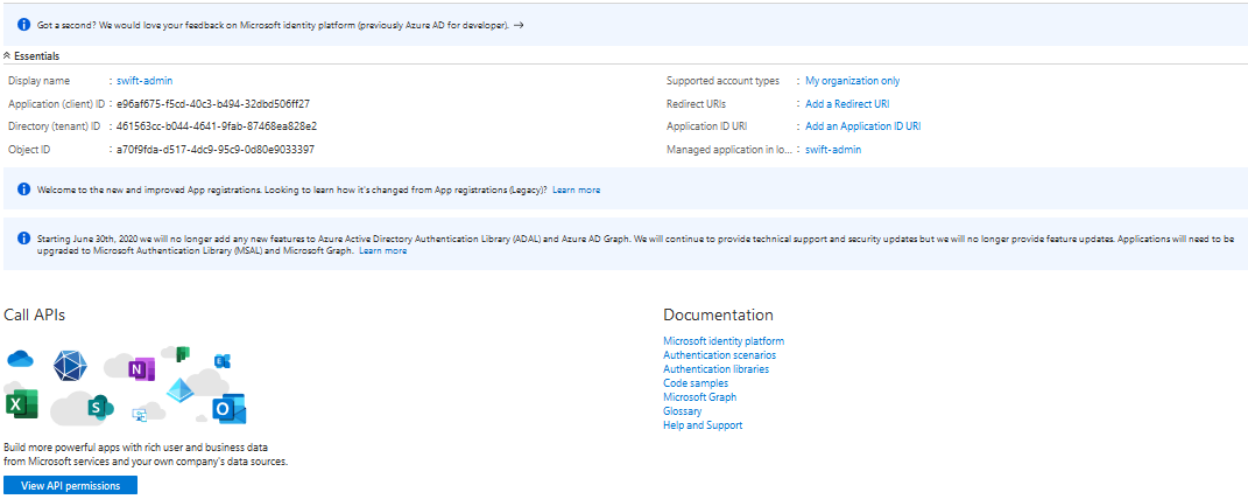
Click on the 'App Registrations' menu and then the 'New Registration' option.



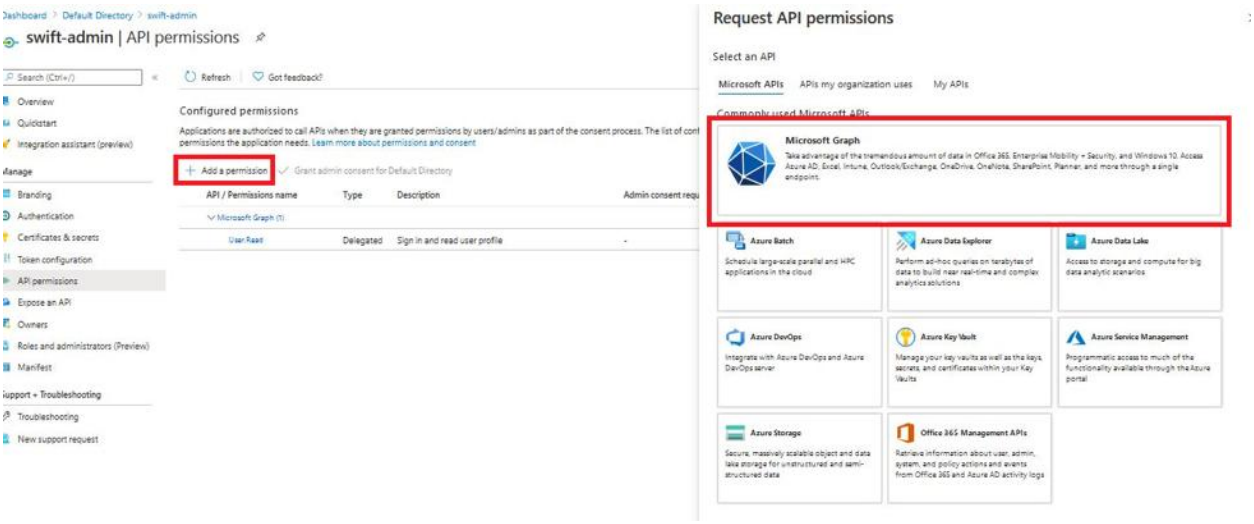
Create a new app registration and give it a name.



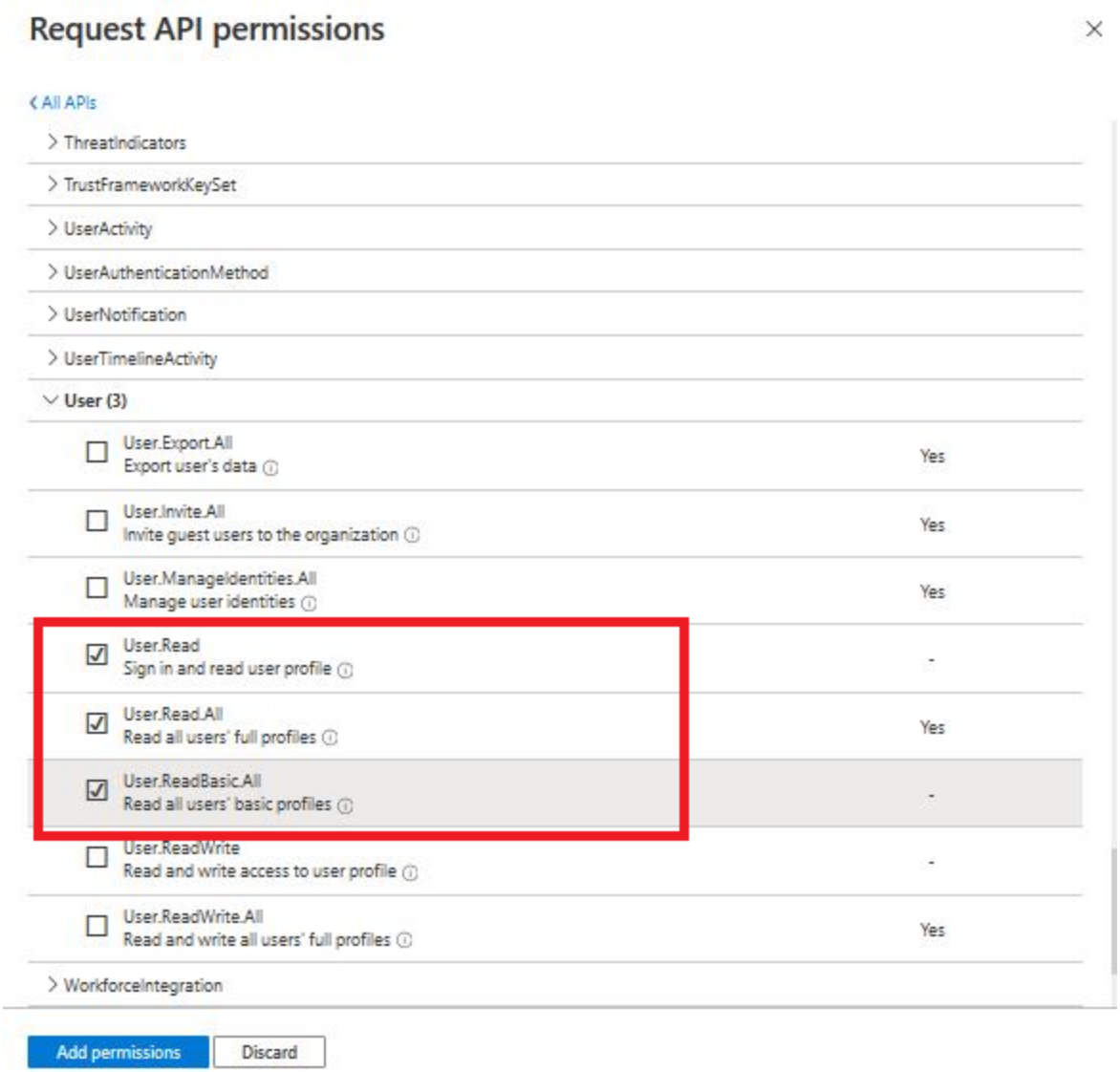
Once the AAD application is created, note the app-id and directory-id. You will need these details later while configuring your AKS clusters under the SWIFT.



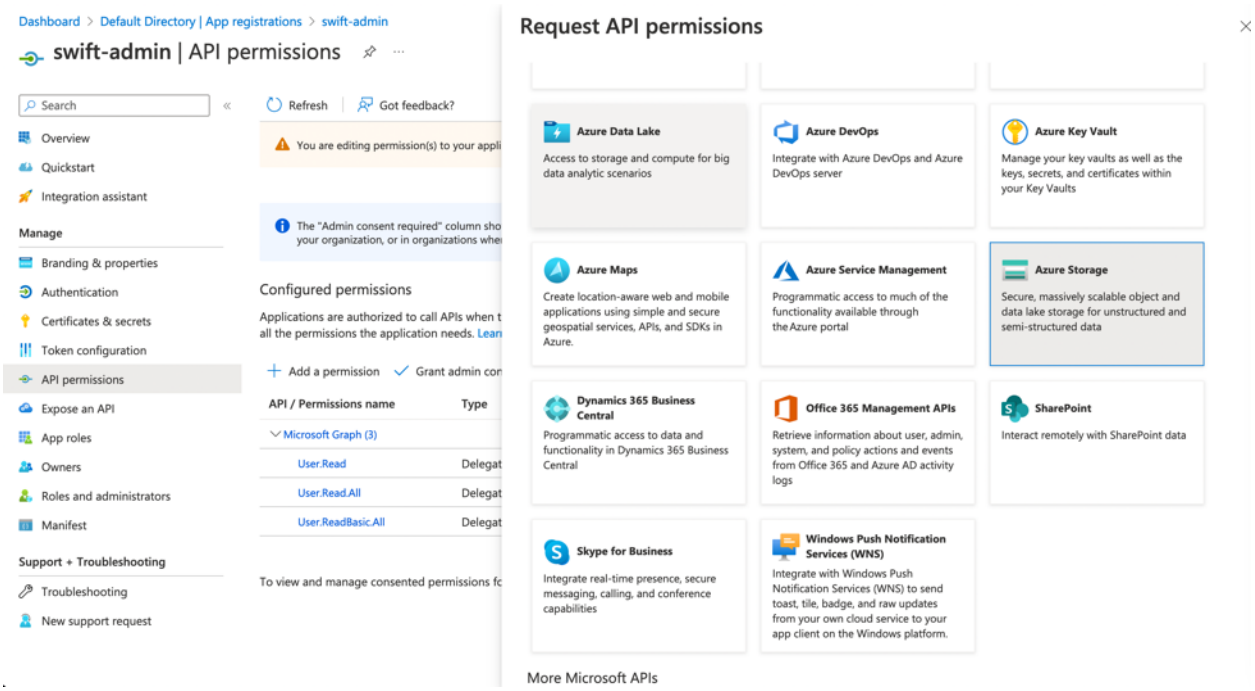
For the new application, click on the 'API permissions' menu on the left, and then select the 'Add Permission' and the 'Microsoft Graph' API permissions.



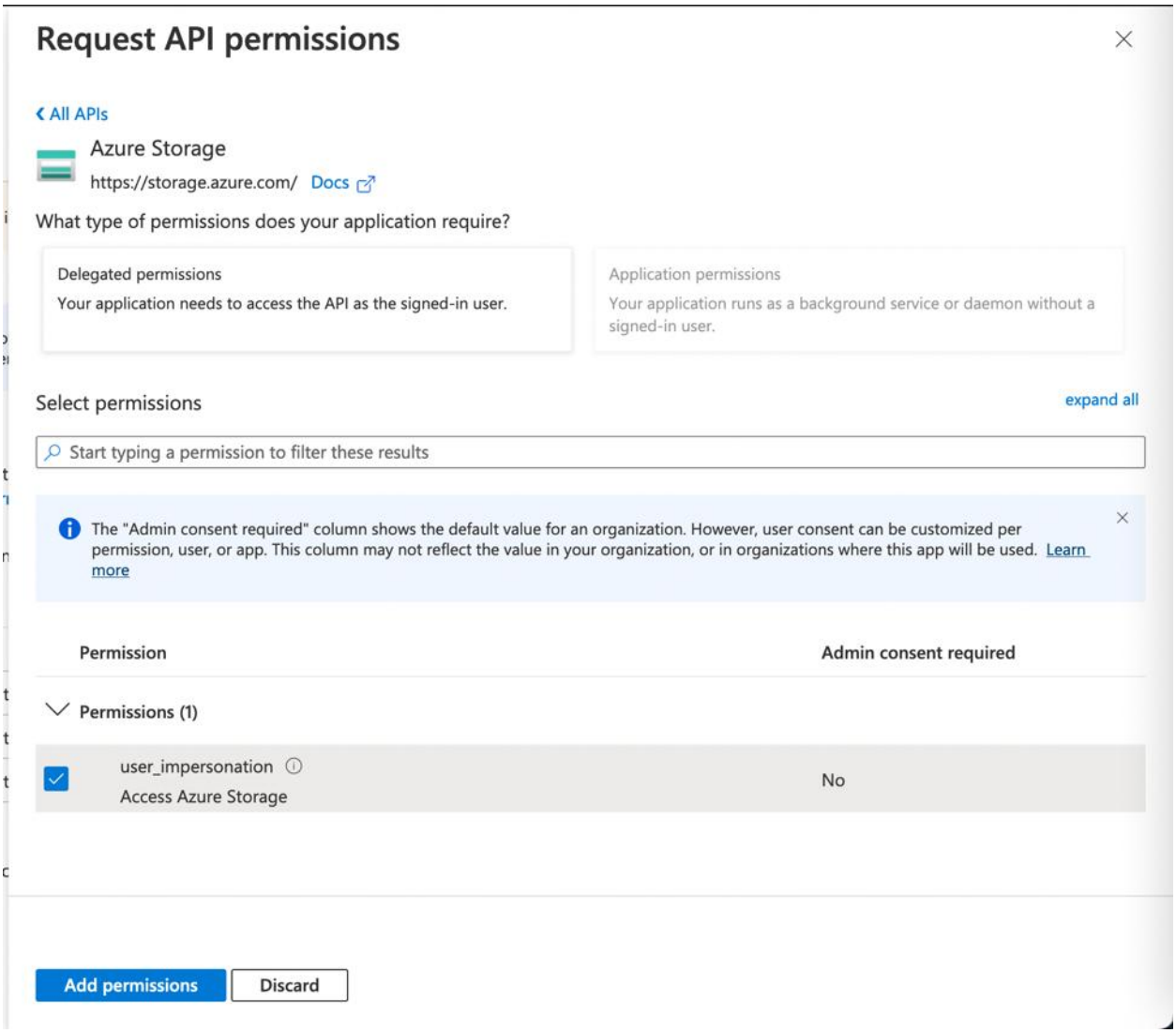
Under the 'Request API permissions' section, select 'Delegated permissions' and check all Read permissions as shown in the below screenshot, and then click on the 'Add permission' button to finish adding those.



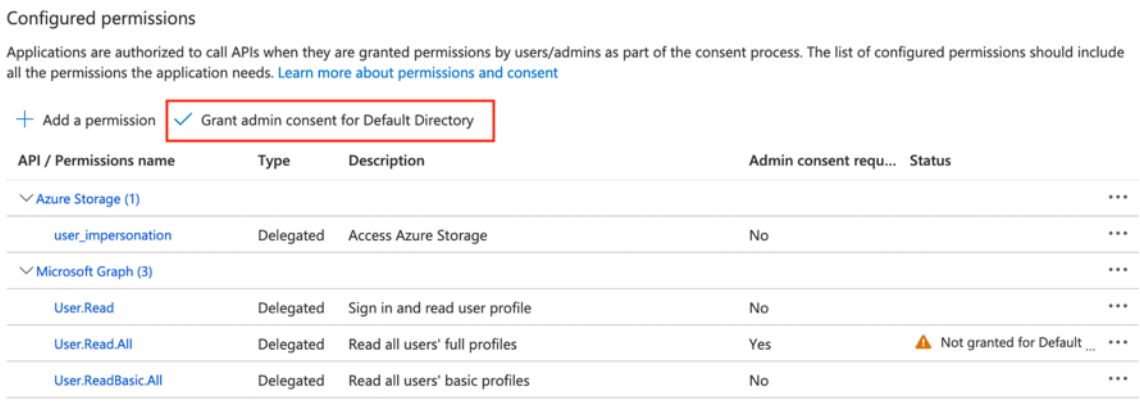
Select ‘Add a permission’ again and this time select ‘Azure Storage’ category.



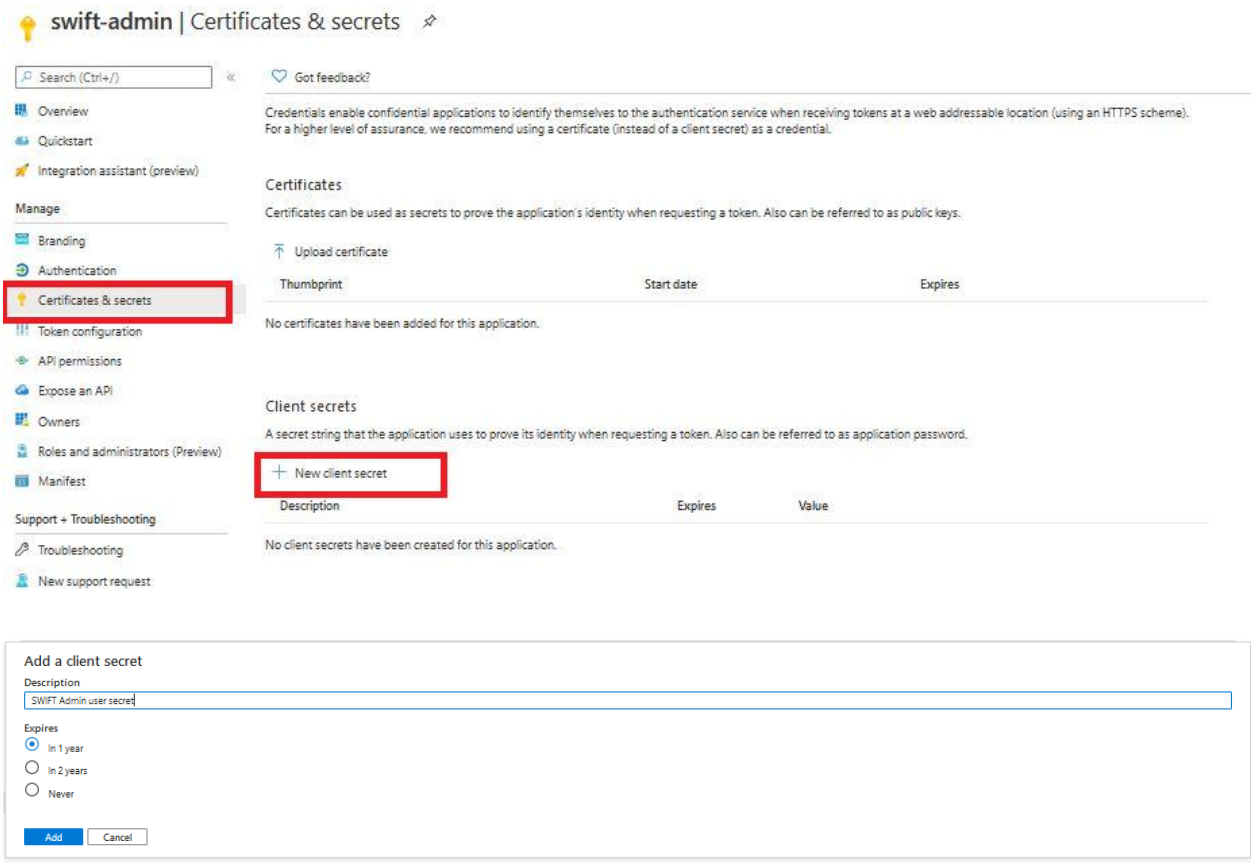
Select permissions shown below and then add it.



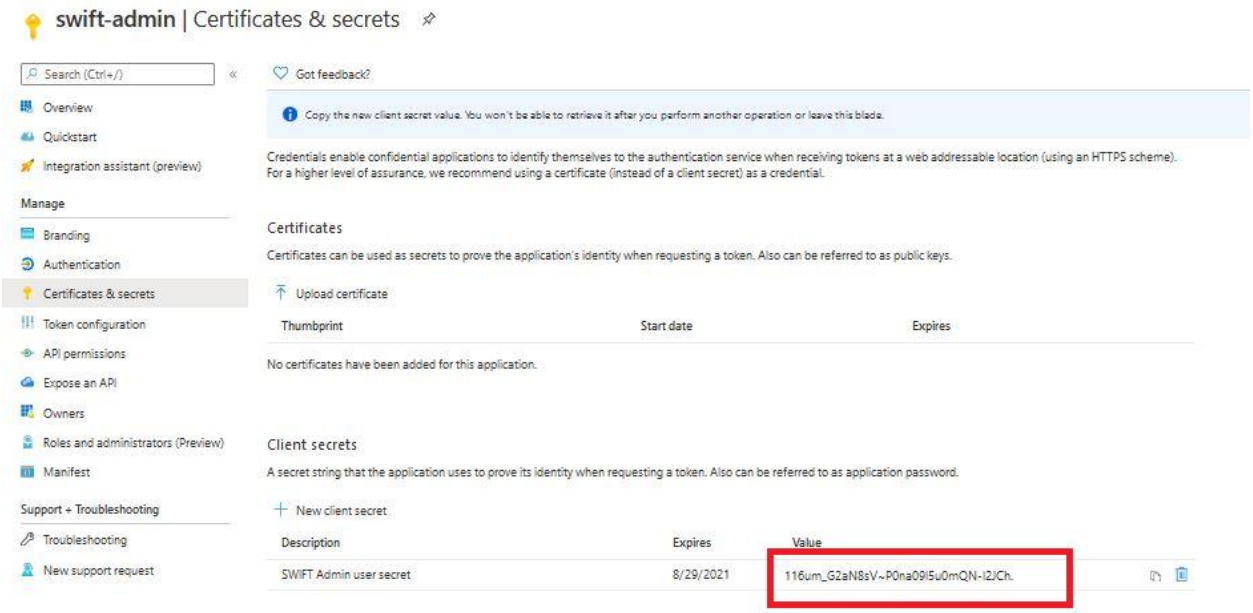
Once the permissions are added, select the admin consent grant option, and then grant the permissions.



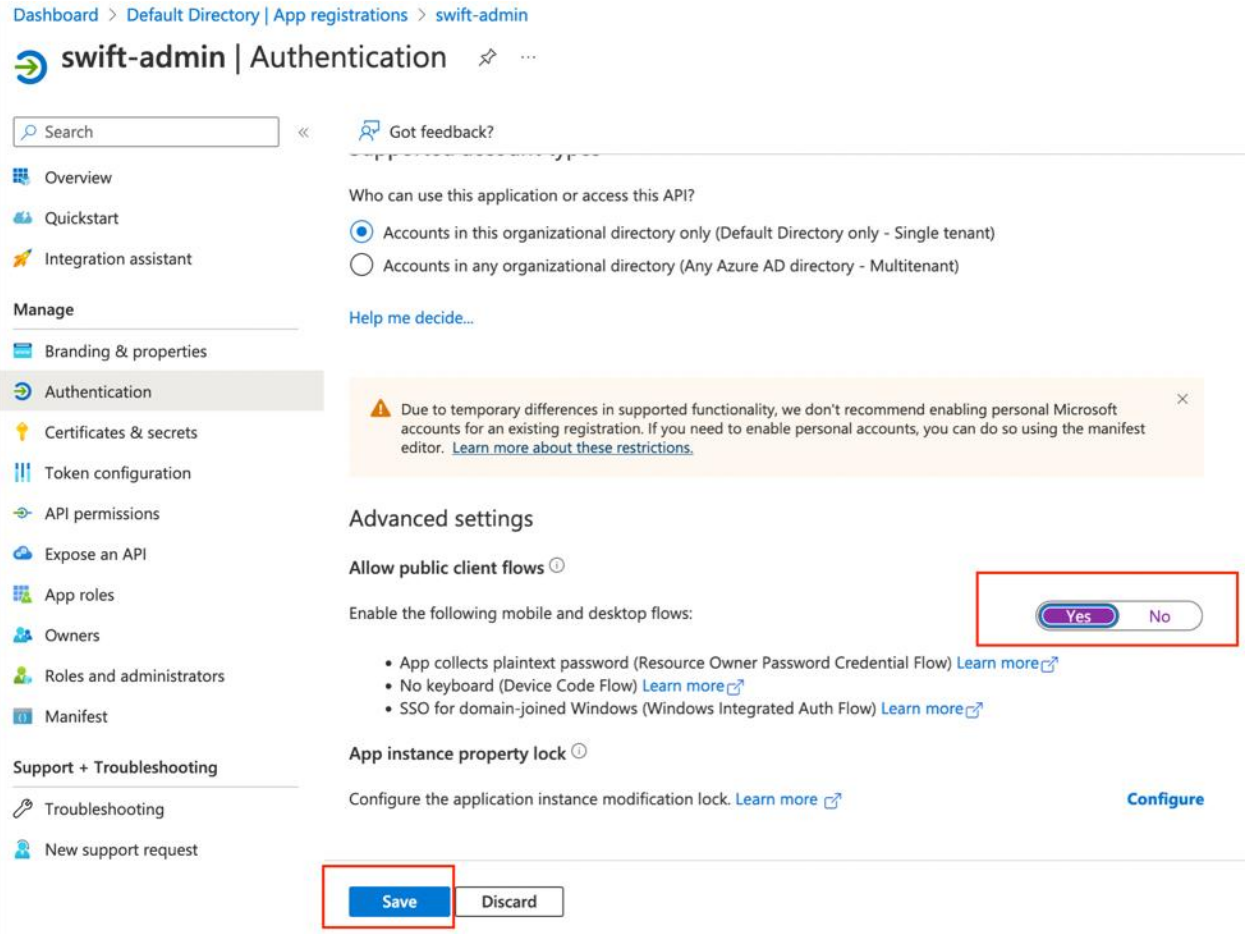
Next, click on the ‘Certificates & secrets’ menu, and then select the ‘New client secret’ option.



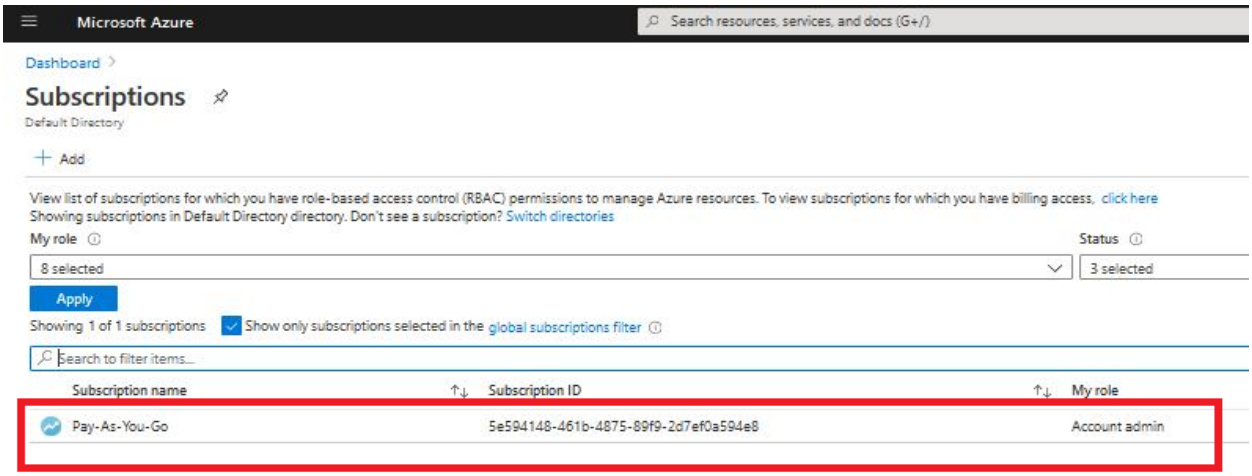
Note the generated secret key and store it safely. The key will not be shown later for security reasons. Also, you will need this key later while configuring your AKS cluster or ACR registry or Azure object storage details under the SWIFT.



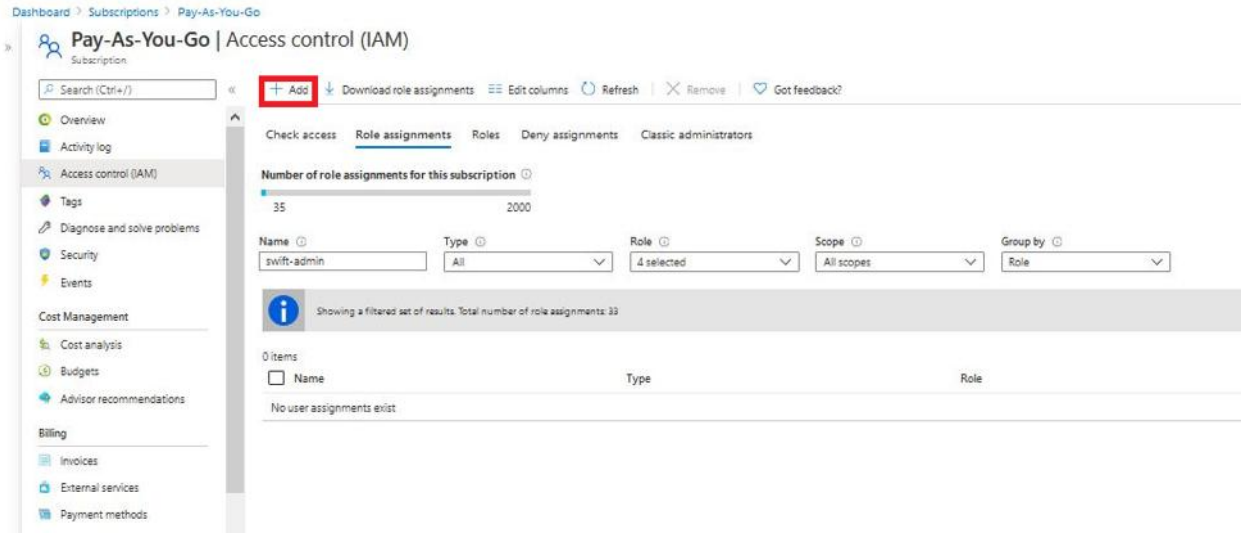
For the new application, from the ‘Authentication’ menu, enable public client flows. Make sure to press the save button to save the changes.



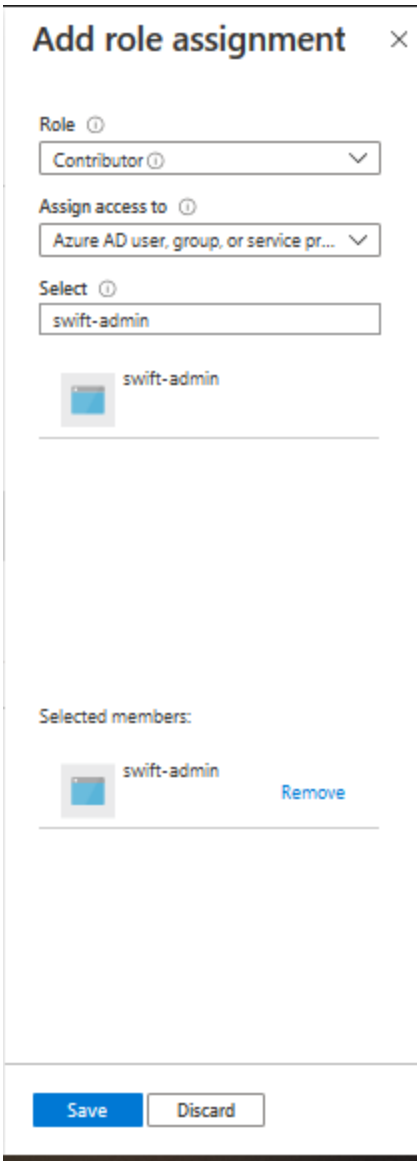
Grant subscription ‘Contributor’ role to the new application. To do that, click on the ‘Subscriptions’ menu and select your subscription where you have your AKS clusters provisioned.

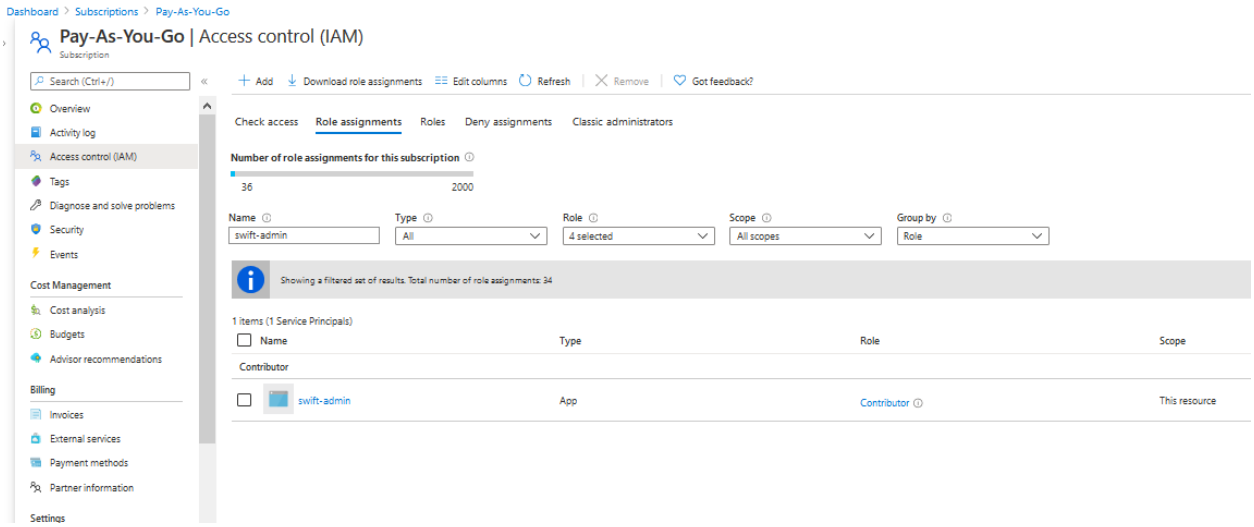


While on the above page, you may also want to note down your subscription id, as you will need it later while configuring your AKS cluster details under the SWIFT.



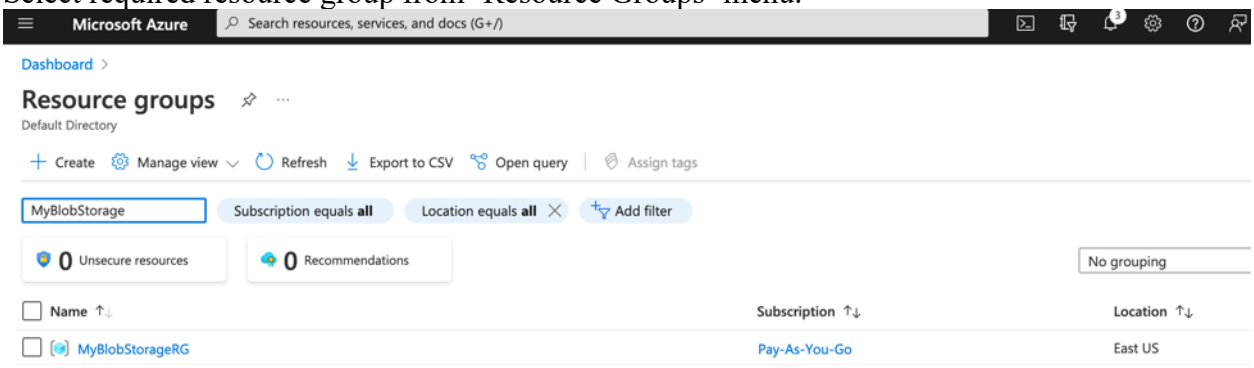
On add dialog, select the newly created application and the contributor role, and then save it.


Add role assignment
Role
Contributor
Assign access to
Azure AD user, group, or service pr...
Select
swift-admin
swift-admin
Selected members:
swift-admin
Remove
Save
Discard

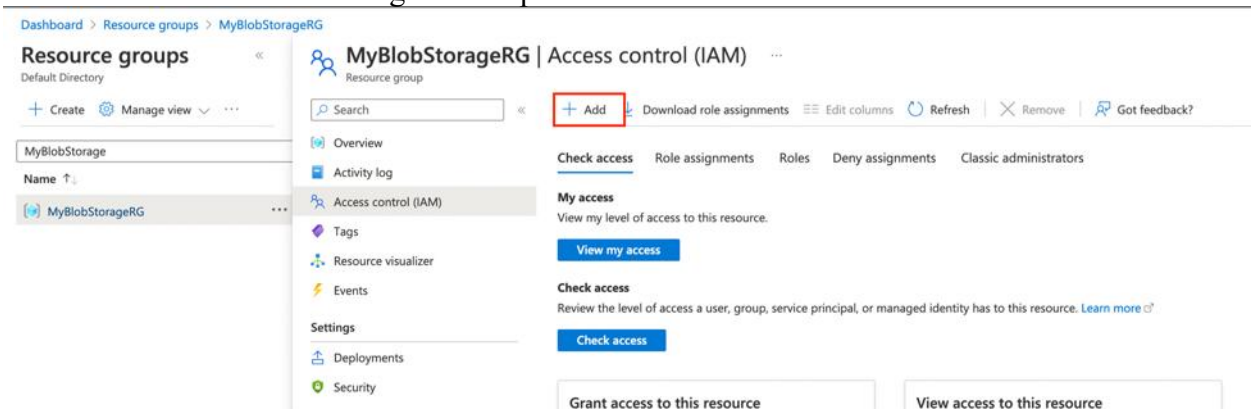

Dashboard > Subscriptions > Pay-As-You-Go
Pay-As-You-Go | Access control (IAM)
Subscription
Search (Ctrl+/)
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Security
Events
Cost Management
Cost analysis
Budgets
Advisor recommendations
Billing
Invoices
External services
Payment methods
Partner information
Settings
Add
Download role assignments
Edit columns
Refresh
Remove
Got feedback?
Check access
Role assignments
Roles
Deny assignments
Classic administrators
Number of role assignments for this subscription
36
2000
Name
swift-admin
Type
All
Role
4 selected
Scope
All scopes
Group by
Role
Showing a filtered set of results. Total number of role assignments: 34
1 items (1 Service Principals)
Name
Type
Role
Scope
Contributor
swift-admin
App
Contributor
This resource

Lastly, you need to assign ‘Storage Blob Data Contributor’ role to the new app or client we created above at all resource group levels where you have either cloud object storage created or want SWIFT to use those resource groups for longer term backups. Note that this is optional step if you do not want SWIFT to use Azure object storage for backups.

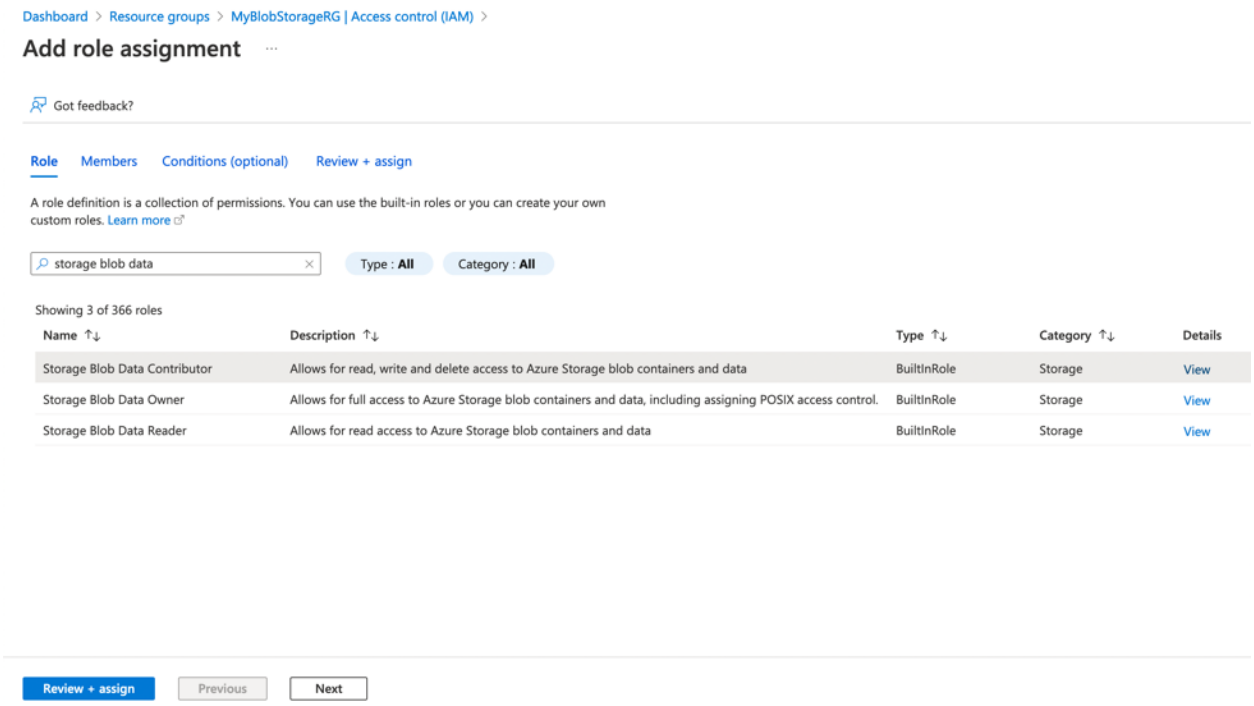
Select required resource group from ‘Resource Groups’ menu.



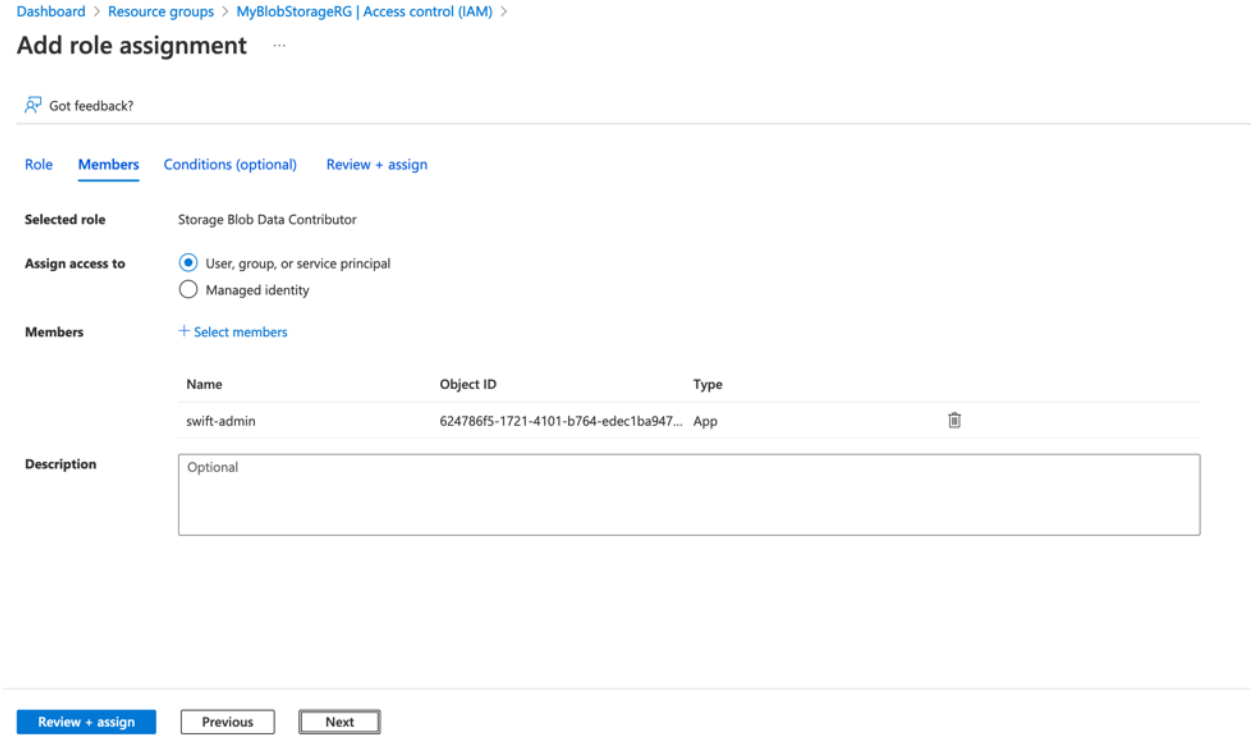
Select the Resource Group and then ‘Access Control (IAM)’ properties for it. Then select the ‘Add’ button and then ‘Add role assignment’ option.



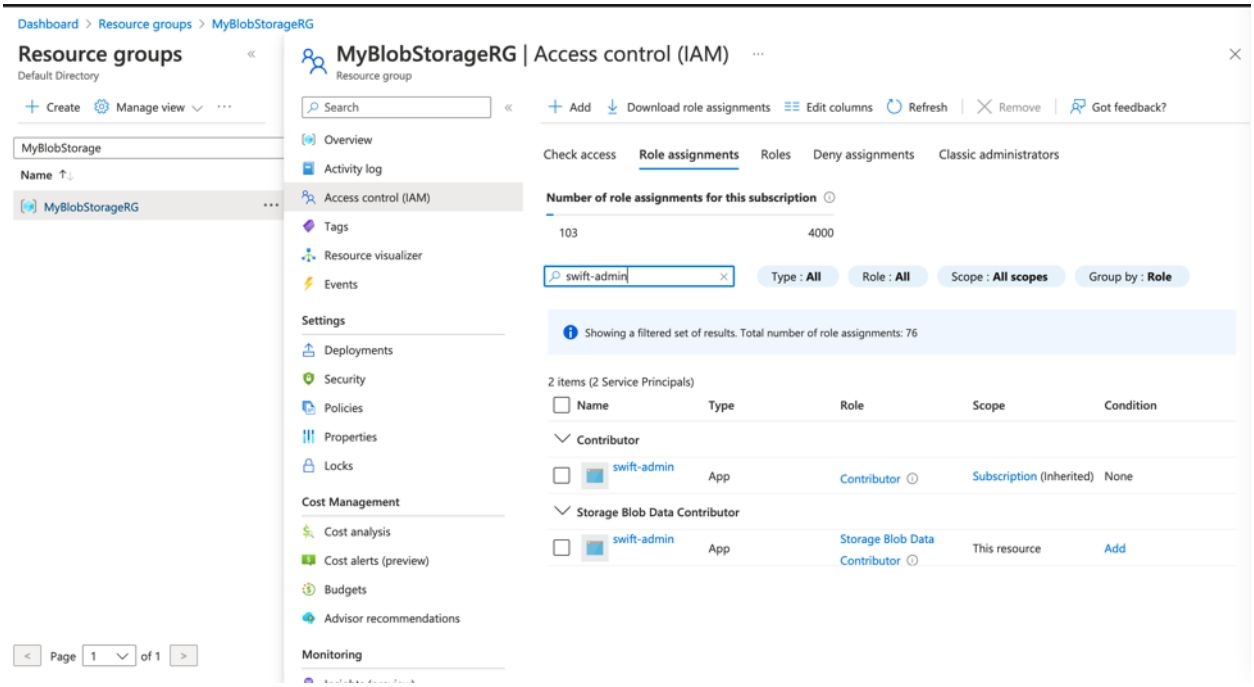
Select the ‘Storage Blob Data Contributor’ role.



Select the app or client you created in earlier steps. For this example case, that is ‘swift-admin’ app.



Press the ‘Next’ button and add permissions after a review. You will see newly added permissions under IAM menu for the resource group.



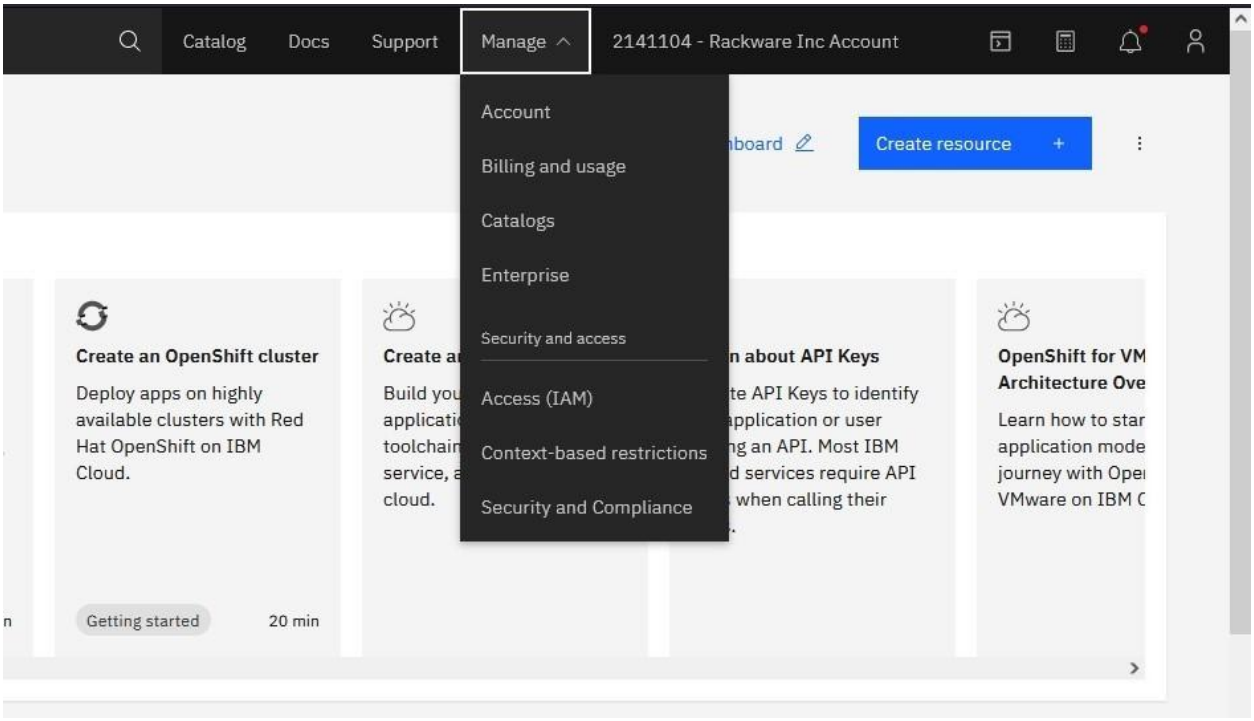
Please now note the below details for the account and the newly created AAD application, which you will need later while configuring your AKS clusters, object storages, or container image registries under the SWIFT:

- Subscription id
- Tenant id (Default directory id)
- Newly created app name and id (client id)
- Newly created app secret (client secret)

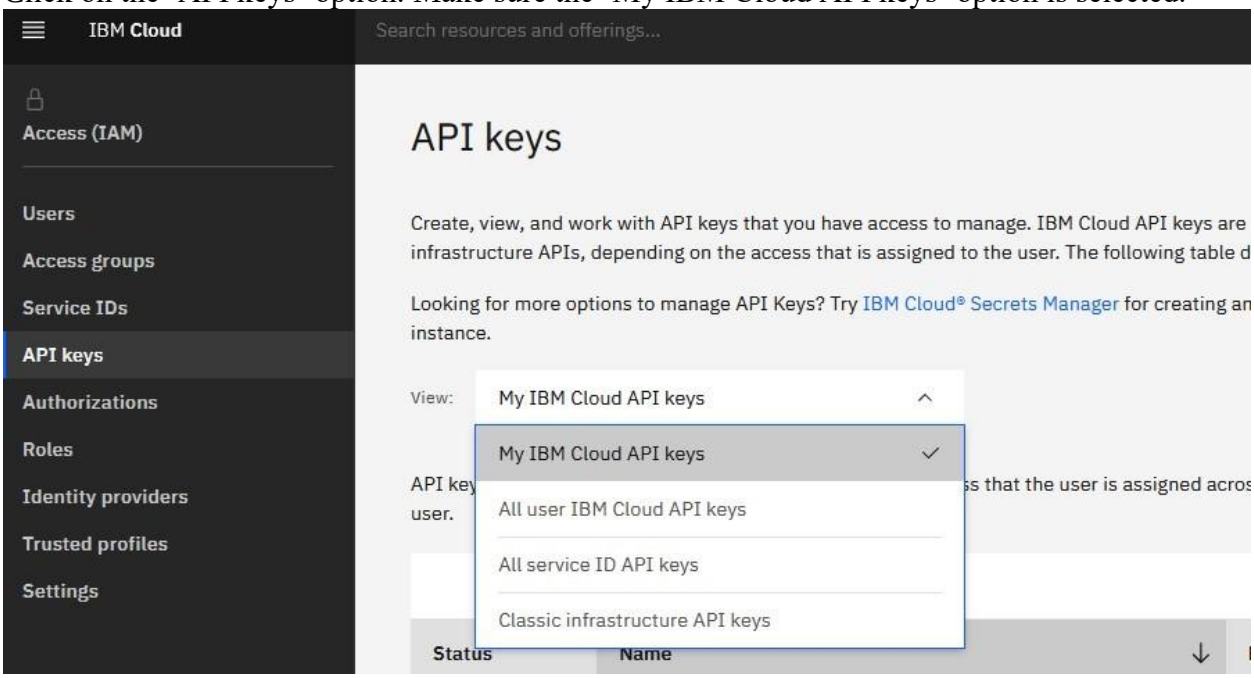
Adding IBM cloud user for SWIFT use

This section highlights the steps to create a user account under your IBM cloud. You will later use these user credentials to configure the IBM Kubernetes Service (IKS) as well as IBM OpenShift cluster details under your installed SWIFT. The same credentials could also be used later to discover an IBM Cloud Container Registry or an object storage under SWIFT.

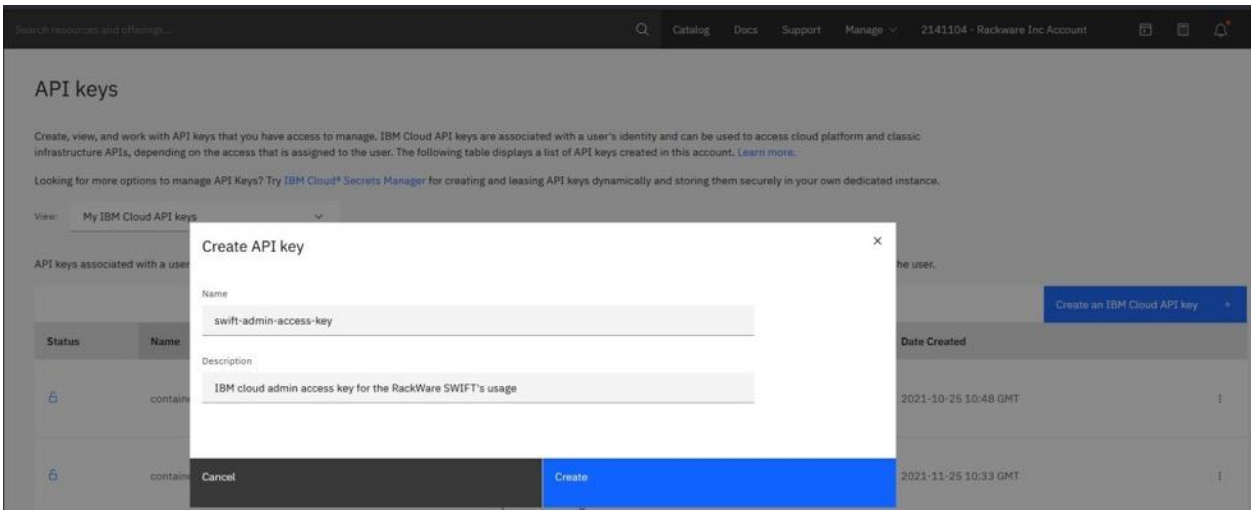
Log in to the IBM cloud console, and from the top section, select the ‘Manage’ and then the ‘Access (IAM)’ menu option.



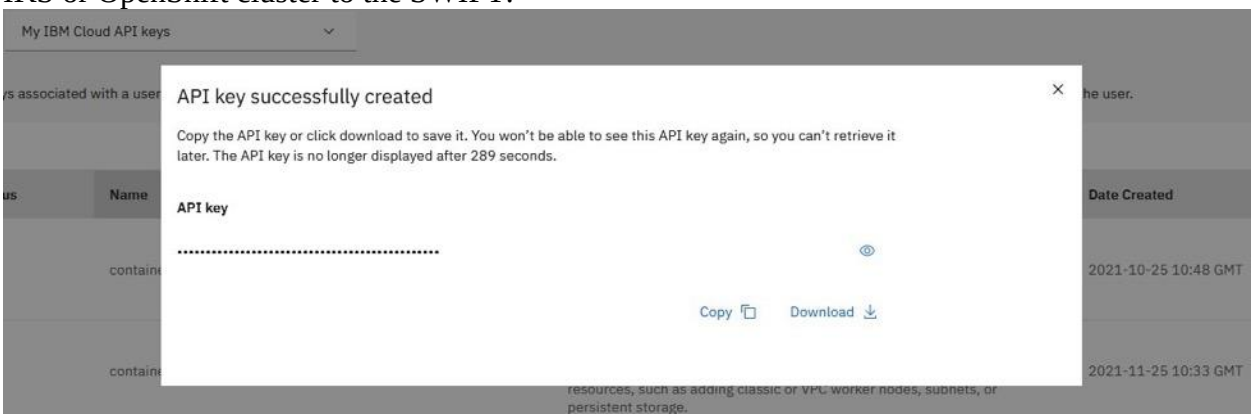
Click on the 'API keys' option. Make sure the 'My IBM Cloud API keys' option is selected.



To create an API key, select the 'Create an IBM Cloud API Key' button on the right side of the webpage. Fill in the name of the key file name and relevant description, and then click on the 'Create' button.

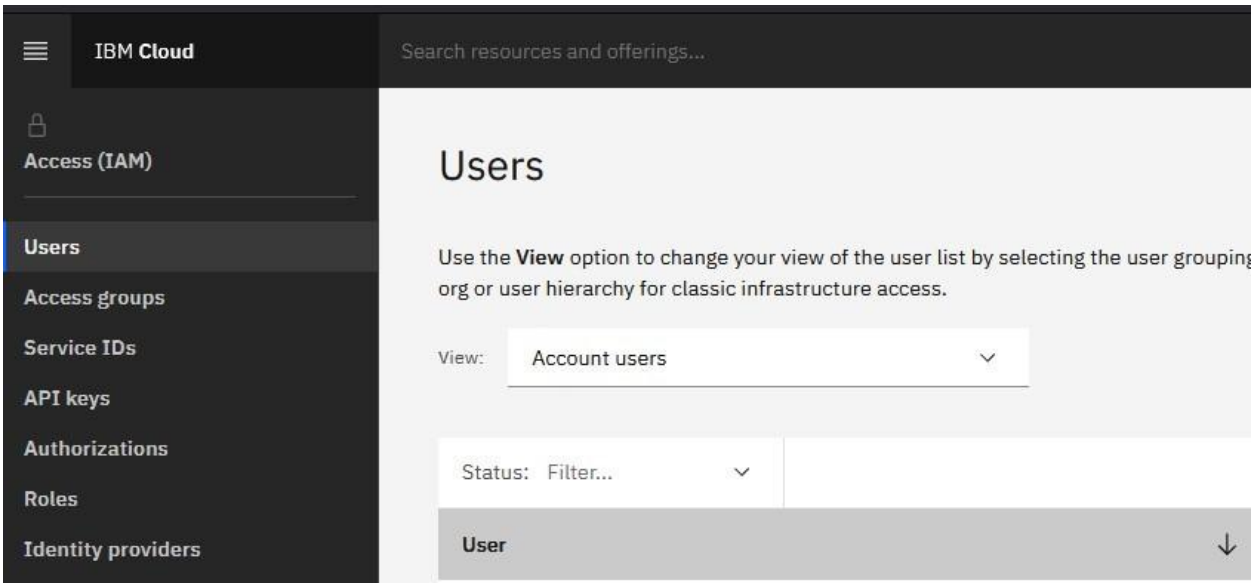


Download and save the generated API key safely. This key will be needed later while adding an IBM IKS or OpenShift cluster to the SWIFT.



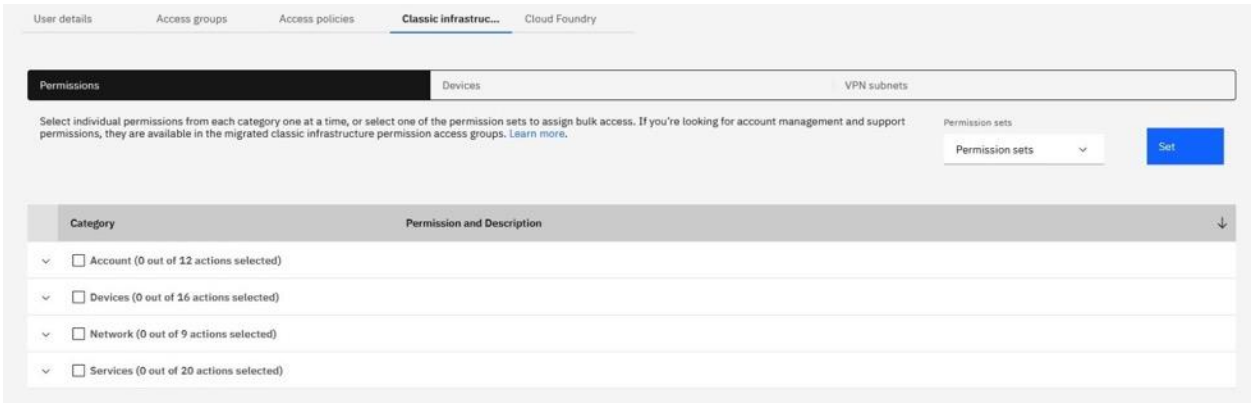
Note: The generated IBM API key inherits all the privileges of the logged in user. If you're logged in with an 'owner' role user and then generated a key, then the generated API key will have full owner rights and if that is the case then you can skip the next set of steps.
However, if you're logged in as a regular user and created a key, then you need to assign some specific privileges for the user that are mentioned in the next section. Without those privileges for the current user, the generated key will not work correctly with your SWIFT server.

Set permissions of the IBM cloud user for SWIFT usage
Select the 'Users' option from the left side menu and then select the current user that generated the key. Additional information can be found here.



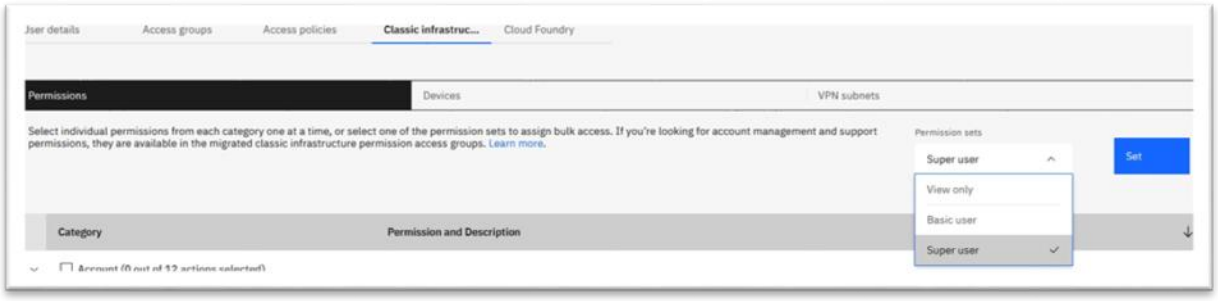
Add the Classic Infrastructure permissions

To assign the classic infrastructure permissions to the user, select the 'Classic Infrastructure' tab. Additional information can be found here.



There are 2 options for assigning classic infrastructure access:

1. Add super user access



2. Add individual permissions as below:

Under the ‘Permissions’ tab, expand and check the following permissions:
Account:

- Add/Upgrade Storage (Storage Layer)

Devices:

- Edit Hostname/Domain
- Manage Port Control

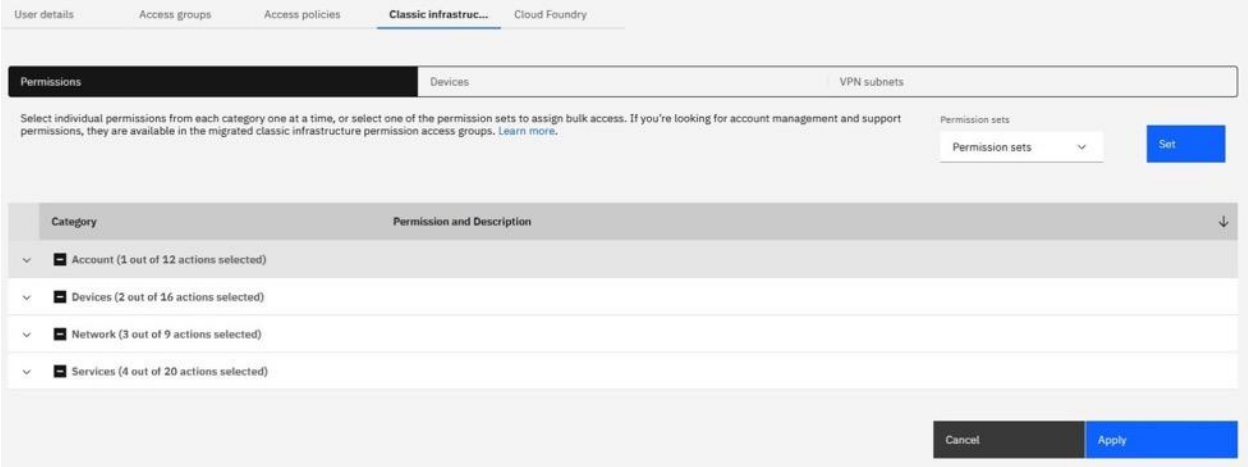
Network:

- Add IP Addresses
- Manage Network Subnet Routes
- Add Compute with Public Network Port

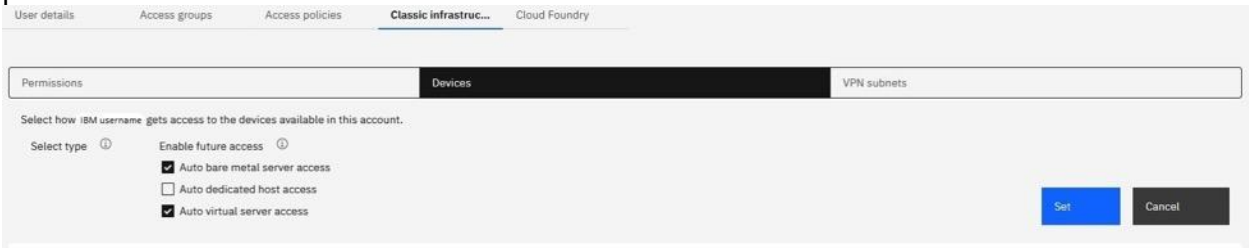
Services:

- Manage DNS
- Manage Certificates (SSL)
- View Certificates (SSL)
- Storage Manage

After selecting the above permissions, click on the ‘Apply’ button.



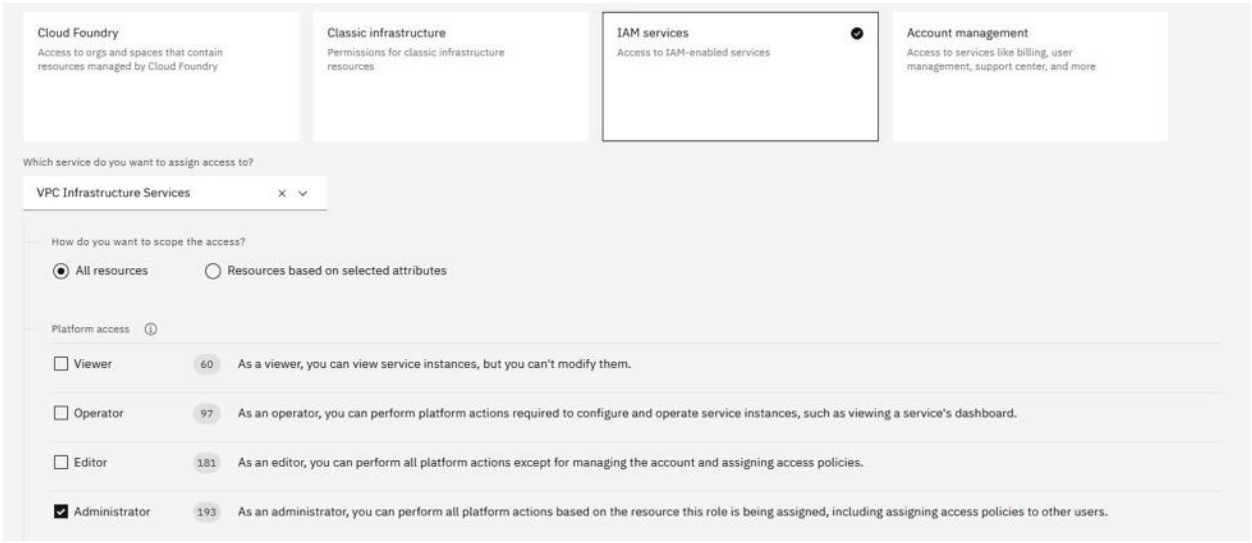
To add/check worker node specific permissions, go to the ‘Devices’ tab and make sure the following permissions are set.



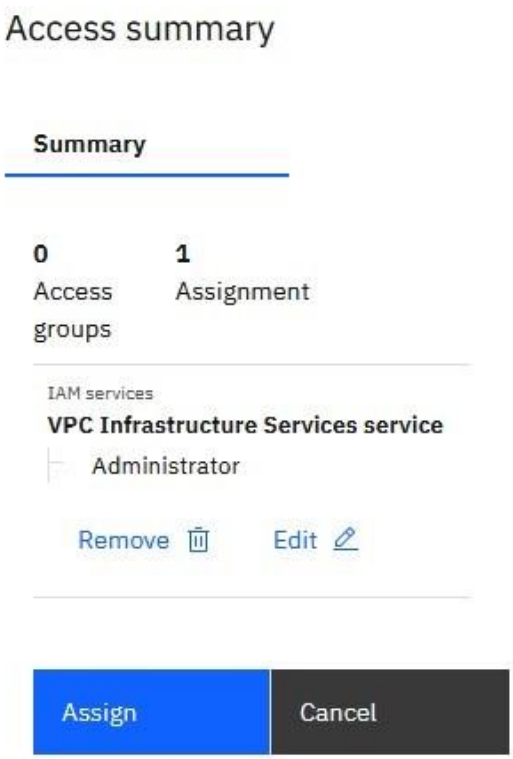
Add VPC Infrastructure permissions

If you have a VPC cluster then the user needs to have the ‘Administrator’ platform access. To assign the access, click on ‘Assign access’ button under the ‘Access policies’ tab.

Search for ‘VPC Infrastructure Services’ and select the related drop-down option. Select the ‘Administrator’ role and click on the ‘Add’ button below.



Click on the ‘Assign’ button on the right to assign the selected role.



Now you can add your IBM IKS or OpenShift cluster to SWIFT by using the username and generated API key.

Adding OpenShift cluster service-account for SWIFT use

Before you can add your local or cloud based OpenShift cluster to SWIFT and start managing it, you will need to have a cluster service account created with the necessary permissions.
Create a YAML for the new service account:

```
$ vi swift-admin-sa.yaml
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: swift-admin
  namespace: kube-system
```

Apply the YAML file

```
$ oc apply -f swift-admin-sa.yaml
```

Next, add the ‘cluster-admin’ role to the newly created account.

```
$ vi swift-admin-roles.yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: swift-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: swift-admin
    namespace: kube-system
```

Apply the YAML file

```
$ oc apply -f swift-admin-roles.yaml
```

To get the service-account token, you can use a command as below. The command would print the ‘token’ key. You will use this output token later while adding the cluster to the SWIFT.

```
$ oc -n kube-system describe secret $(oc -n kube-system get secret | grep "swift-admin-token" | awk '{print $1}')
```