# RACKWARE SWIFT

INTRODUCTION AND OVERVIEW
VERSION 1.3.17

RACKWARE INC.

# Contents

## What is the RackWare SWIFT solution?

SWIFT is a new product offering from RackWare for replication, backup, and DR of the container orchestration platforms. SWIFT supports Kubernetes and OpenShift clusters for replication, backup, and DR use-cases.

SWIFT supports multiple cloud platforms, including Amazon EKS, Oracle OKE, Azure AKS, IBM IKS, and Google GKE services. SWIFT also supports cross-cloud, cross-platform, and cross-version replications as well as DR scenarios.

## Where can SWIFT help you?

The primary use-cases of SWIFT are:

- Any-App Any-Storage Live Replication: The cross-cloud, cross-platform, and cross-version replications for your containerized applications without any production downtime.
- Any-App Any-Storage Backup and Restore: The cross-cloud, cross-platform, and cross-version backups as well as restores for your containerized applications. Long-term backups can be configured into the cloud object storage.
- Any-App Any-Storage Disaster Recovery: The cross-cloud, cross-platform, and cross-version DR capability with automated failover and failbacks for your containerized applications.

If you work in an IT or a DevOps team and use Kubernetes or OpenShift, SWIFT will help you with multiple use-cases in your day-to-day business activities. SWIFT will help you unlock the real power of your DevOps. Below are some of the common use-cases where SWIFT can help you:

- You need to backup your application before a new rollout, so if something goes south, you have a click-button restore and a plan-B in place.
- You need the capability to restore your production application backups anytime to any other cluster (including to the same production cluster) and any namespaces.
- You need a full-fledged production disaster recovery (DR) solution for your containerized workloads based on Kubernetes or OpenShift platforms.
- You need a full-fledged disaster recovery (DR) solution with fully automated failover and failback capabilities.
- You need a disaster recovery (DR) solution that allows you a choice of pre-provisioned DR clusters (to keep them hydrated in DR steady state) or dynamic provisioning of DR clusters as part of a failover or mix of the two strategies based on cluster priority.
- You want to clone or lift-and-shift your containerized applications from your data center to a cloud, between clouds and regions, or from cloud to the data center.
- You want to pick and choose cloud, platform, and storage for your disaster recovery (DR) site and not worry about storage or any other vendor lock-in.
- You need multiple full and incremental backups for your containerized application with the capability to restore any of those at any point in time.
- You need to replicate your containerized application images between two container registries (on-prem or cloud combinations).
- You need reliable, faster, continuous syncs with a better RPO for your disaster recovery (DR) site.

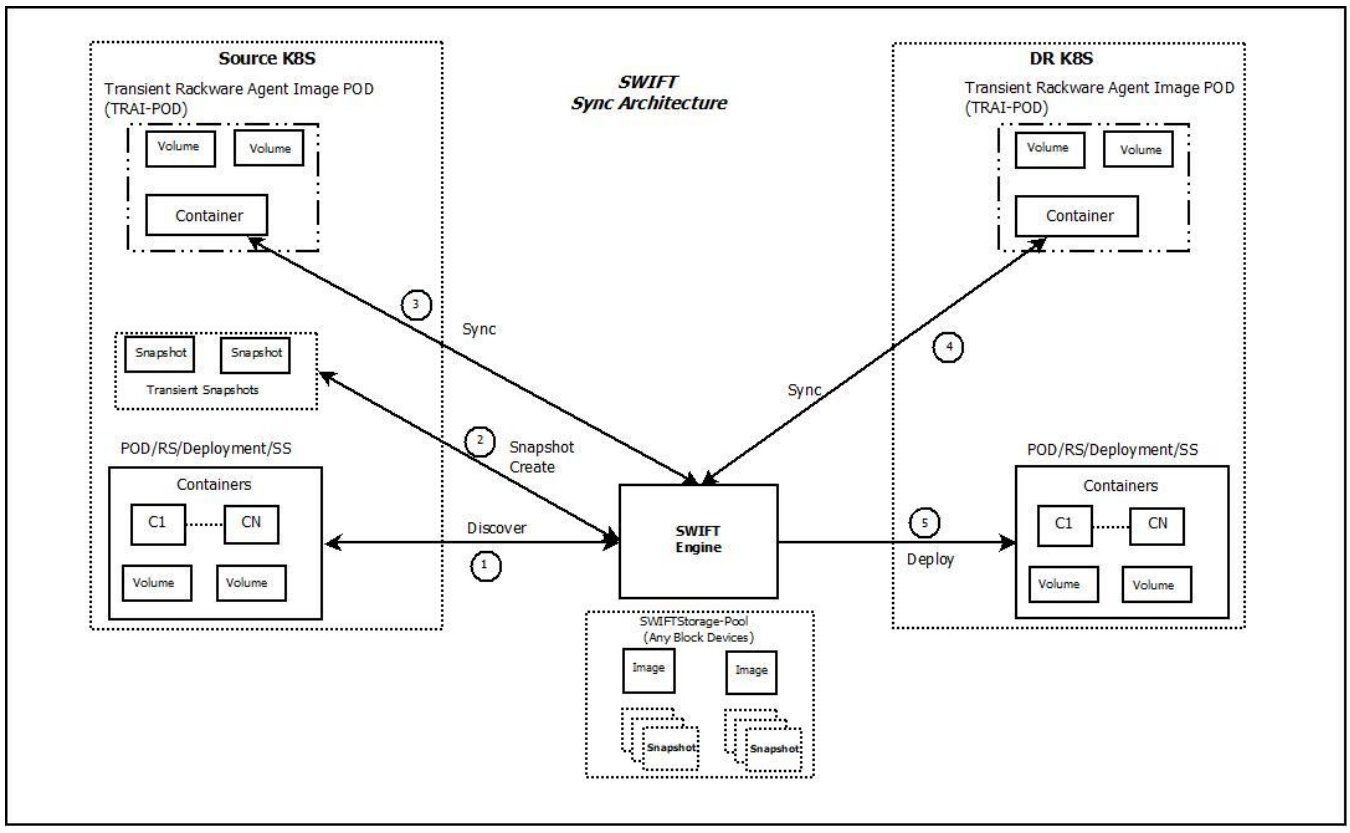
Given SWIFT is purely Any-To-Any replication, backup, and DR solution, you can employ SWIFT in more ways than mentioned above. SWIFT will help you avoid vendor lock-ins and will also allow you to break any existing storage lock-ins.

## How does SWIFT work?

SWIFT employs hybrid syncs for backup and DR. It has a filesystem view for Kubernetes volumes that allows it to do a lot of customizations during sync. SWIFT installs outside all of your managed clusters. It can be installed on any server/VM, which has network connectivity to all managed clusters. The selected namespaces, application Pods, and volumes are backed up by capturing all objects (metadata) and taking a snapshot of storage volumes. SWIFT will automatically discover the entire cluster to find application object dependencies. The volume snapshots are read at the filesystem level by attaching to a transient orchestration Pod. During a SWIFT sync, no production application reconfig is needed or done in any way.

SWIFT supports various storage providers (including all cloud-native volume types) for snapshots and intelligently employs a vendor-native snapshot mechanism that can be at any level (like Kubernetes, storage layer, cloud, etc.) depending on the storage type. Most CSI volumes in Kubernetes are snapshotted using CSI APIs if the corresponding CSI driver supports snapshots.

*SWIFT Sync-Engine architecture*



Once you install a SWIFT, you can add it with any block storage that it manages under one or more virtual storage pools. Configuring additional block storage for SWIFT is optional if you don't need SWIFT to do backup and DR and only use it for one-time replications.

Typically, the below events happen when a SWIFT sync is initiated (Refer to the above diagram)

1. You configure the cluster credentials in the SWIFT through SWIFT CLI or dashboard. For a local cluster, the credentials consist of a service-account token, while for cloud-based clusters, you will input cloud-account credentials. The configuration itself is a one-time step. SWIFT will go ahead and discover all containerized applications using Kubernetes or OpenShift APIs. The discovery is lightning fast and typically takes less than a few seconds. The discovery also captures all application objects and metadata in the SWIFT database.
2. You initiate a sync between production and target cluster namespaces (for migration use-case) or between production cluster, storage pool in SWIFT and DR cluster namespace (for DR use-case).
3. SWIFT will snapshot the auto selected persistent volumes in the production cluster (which could be mounted in Pods directly as PVs or used as PVCs). Depending on the volume type, it could either be snapshotted using a storage-vendor native mechanism (like cloud APIs) or CSI APIs. SWIFT will intelligently employ the proper snapshot mechanism for each volume to be backed up.
4. SWIFT will launch a transient agent-Pod (TRAI-Pod) and attach snapshot volumes to it. These temporary snapshot volumes are cleaned up as soon as sync or replication finishes. The agent-Pod uses RackWare proprietary image, which will be stored inside your cluster-specific private image registry for convenience.
5. Depending on the replication or sync mode, SWIFT will replicate data in the previous step to SWIFT storage (staged-sync) or directly to the target volumes (passthrough-sync). If you selected staged-sync, then at this stage in the flow, SWIFT would replicate data to the target by creating empty volumes and attaching them to a transient agent-Pod. In passthrough-sync mode, the target agent-Pod and empty volumes are created in the previous step, so SWIFT will directly replicate data from the source to the target cluster (i.e., between agent-Pod attached volumes on both sides of clusters). Regardless of sync mode, the data transfer is always encrypted with VPN-grade security and compressed for on-wire transfer.
6. SWIFT will create all required application objects (services, ingress, Pods, PVs, PVCs, etc.) inside the target cluster to restore your application. SWIFT will also run post-sync health validation for your replicated applications. In case of staged-sync mode, the restore happens as part of the stage2 sync.

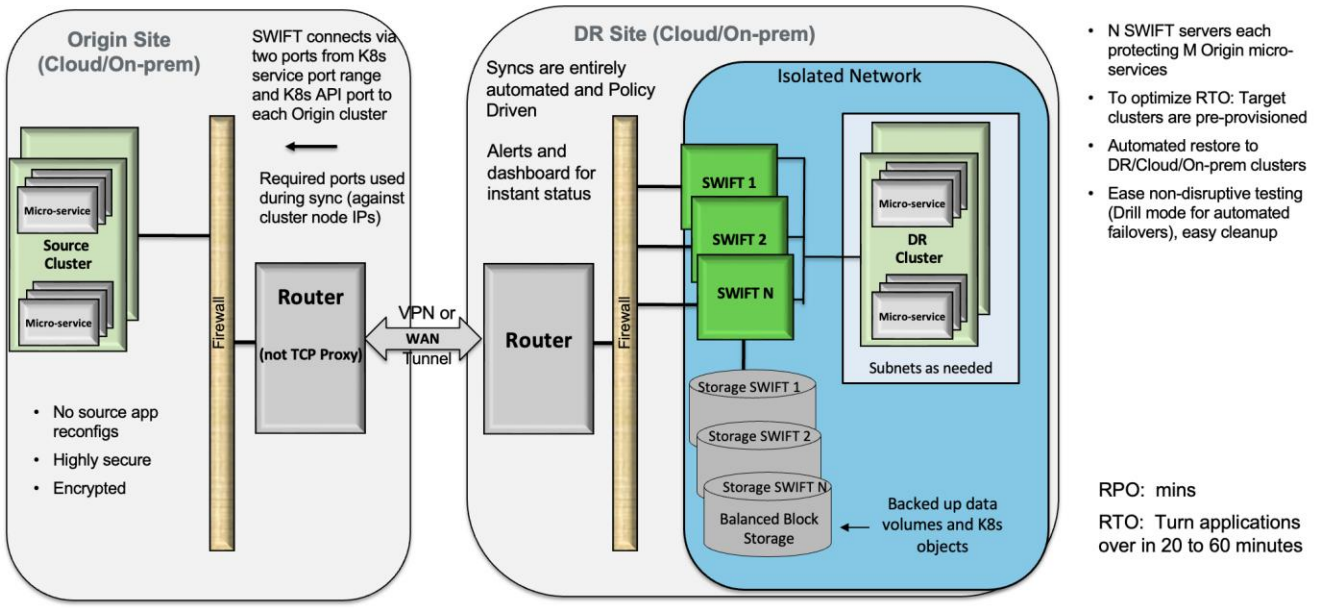
The first sync will replicate the entire application, including all relevant volumes, while subsequent syncs of the same namespace will only transfer delta changes.

If you use staged-syncs for DR use-case, then SWIFT will allow two-step (store-and-forward) syncs, with stage-I sync capturing application data/volumes and objects to the SWIFT CMDB and stage-II sync replicating data/volumes and application objects from the SWIFT CMDB to the target cluster.

You need to use staged-syncs for DR use-case, especially if you want to create more than one full and incremental backup of your applications. Additionally, the staged syncs will allow you to dynamically provision the target (DR) cluster in the cloud only when the actual DR event happens. That is a huge cost-saver for DR without losing much on RTO time!

The container registry replication works differently and not covered here. But SWIFT can allow you to replicate or configure DR for container registries too.

## SWIFT Deployment - DR Architecture




For DR deployments, you can employ one or more SWIFTs to distribute the sync load across servers. SWIFT installs as part of CentOS/RHEL/Oracle Linux 7.x+, and it can be part of a VM or a physical server. It is recommended that you install SWIFT in DR site or environment, so if you lose source as part of a DR event, then you will still have SWIFTs to do any failovers and DR site prep that SWIFT can automate today. SWIFT servers will be installed within your DR firewall boundary.

Load distribution across SWIFT servers on DR site can be based on one or more criteria below.

1. N number of micro-services per SWIFT
2. N number of namespaces per SWIFT
3. N GBs of total persistent volume size per SWIFT
4. N container clusters (Kubernetes or OpenShift or others that SWIFT supports) per SWIFT

The later three criteria, when employed, could mean number of micro-services synced per SWIFT change drastically and unevenly balanced. The actual criteria chosen depends on your environment as well as SWIFT server configuration and network bandwidth available to each SWIFT server.

Each SWIFT server then will be typically configured with DR-local block storage devices that it will manage internally as one or more virtual pools. Optionally, you can also configure remote object storage under each SWIFT that it will manage as one or more remote pools. It is recommended that you do short term backups in SWIFT pool that maps to local block storage, while configure long term backups (weekly/monthly backups) to pools that map to remote object storage, as object storage is cheap but also slower for access.

For DR deployment, you will typically follow below steps:

1. You will first open required network ports that SWIFT will use for its discovery and syncs in source as well as DR site firewalls, and any firewalls in between.

2. Provision one or more SWIFT servers on DR site where SWIFT can be installed. This can be VM or physical server with CentOS/RHEL/Oracle Linux 7.x+. You will work with RackWare professional services team to decide number of SWIFT servers needed for your deployment. Essentially, this number depends on what above mentioned criteria or combination of them you use to distribute sync load across SWIFT servers.
3. Provision DR-local block storage devices and/or object storage locations that can be configured under each SWIFT, where backed up data and objects are stored.
4. Attach required block storage to each SWIFT server. How much storage goes to each SWIFT server will be dependent on what criteria you choose from the above list to divide sync load across SWIFT servers.
5. Set up initial password for all SWIFT servers, login to Dashboard for each SWIFT, and then set up other non-admin users and roles.
6. From each SWIFT's dashboard, create virtual pool(s) for attached storage or object storage locations. The created pool is where SWIFT backups will be stored.
7. From each SWIFT's dashboard, create one or more DR policies with sync frequency, timeline and exclude time windows to meet your RPO requirement. You will then also apply these policies for syncing namespaces or micro-services that are logically assigned/distributed to the SWIFT server (from step #1) to sync them.
8. Anytime DR event happens, or you want to do a DR drill, you will failover required DR policies from respective SWIFT server's dashboard. Such failover operation for SWIFT DR policies also supports a drill mode for testing DR site.
9. Once DR testing is complete for the DR drill, or once original site is rebuilt in case of real DR failover, you will fallback DR policies from respective SWIFT's dashboard so forward syncs can be resumed.

Note: All operations mentioned above for SWIFT dashboard could also be performed with SWIFT CLI commands.

# SWIFT pre-requisites and version matrix

## SWIFT Installation

The SWIFT can be installed on any 64-bit CentOS/RedHat 7.x/8.x machine (VM or baremetal). We recommend having 10GB or more free space under the '/opt' partition. You will also need full connectivity and access to your Kubernetes or OpenShift cluster service-port range and API port access from the SWIFT server. Apart from that, there are no particular pre-requisites for the SWIFT install.

Please refer to the SWIFT installation and pre-requisites guide for more details.

## Supported Storage Types

SWIFT supports the below storage providers for snapshots. These only apply to a source cluster or container platform, and the target/DR cluster can have ANY storage providers configured.

- Any CSI volumes (where the CSI driver supports snapshots)
- Ceph storage (CSI and non-CSI)
- Amazon EBS, EFs, and FSX storage
- Azure disk and file storage (all storage SKUs and classes)
- Google disk and file storage (all storage SKUs and classes)
- Oracle block (BV) storage (all storage SKUs and classes)
- IBM Classic File and block storage as well as VPC block storage
- Red Hat OpenShift Data Foundation (OSDF)
- Rancher Longhorn (CSI)
- Akamai Linode block storage (CSI)
- Digital Ocean block storage (CSI)

The supported storage providers list is rapidly expanding for SWIFT. If you need any specific storage provider added, then please contact RackWare Support. Typically, it takes us only a few weeks to add any new storage provider.

## Supported Container Platforms

SWIFT supports the following container orchestration platforms for replication, backup, and DR.

- Vanilla Kubernetes [v1.14 - Latest]
- Oracle Linux Cloud Native Environment (OLCNE) [v1.14 - Latest]
- Google Kubernetes Engine (GKE) [All available versions]
- Amazon Elastic Kubernetes Service (EKS) [All available versions]
- Azure Kubernetes Service (AKS) - Including community AKS edition [All available versions]
- Oracle OCI Cloud Kubernetes Engine (OKE) [All available versions]
- Red Hat OpenShift Container Platform [v4.5 - Latest]
- Red Hat OpenShift Dedicated for Google Cloud [All available versions]
- Red Hat OpenShift for AWS Cloud (ROSA) [All available versions]
- Azure Red Hat OpenShift (ARO) [All available versions]
- Red Hat OpenShift Origins (Community/OKD) [v4.5 - Latest]
- IBM OpenShift Cloud [All available versions]
- IBM Kubernetes Service/IBM Container Cloud [All available versions]
- Digital Ocean Kubernetes (DOKS) [All available versions]

- Rancher Kubernetes [v1.14 - Latest]
- Akamai Linode Kubernetes Engine (LKE) [All available versions]

The supported container platforms list is rapidly expanding for SWIFT. If you need any specific container platform or version support added, then please contact RackWare Support.

### Supported Container Registries

SWIFT supports the following container image registries for cross-platform replication, backup, and DR.

- Google Container Registry (GCR)
- Azure Container Registry (ACR)
- Oracle Cloud Infrastructure Registry (OCIR)
- Amazon Elastic Container Registry (ECR)
- Docker Hub

The supported list of container registries is rapidly expanding for SWIFT. If you need any specific container registry support added, then please contact RackWare Support.

## SWIFT Licensing

SWIFT employs innovative and liberal per-app-per-GB licenses. Essentially, you will only pay for what you migrate or do DR for. An app stands for a micro-service or Service object in Kubernetes or Openshift. The per-app-per-GB license is application-centric, so charged on an application (micro-services) basis. One SWIFT application license allows you to replicate/DR one micro-service with any number of volumes totaling up to 200GB in size. For any additional storage replication/DR, you can buy additional storage licenses. Each storage license will allow you to replicate/DR 200GB of additional data (Volume sizes).

The storage license is not tied to any specific application, so you can avail it to replicate/DR any application's data. Similarly, the free 200GB storage quota you get with each license can be used across applications. That means if you have two applications (micro-services) with 300GB and 100GB sized volumes each, then you can do DR/replication with only two application licenses and without any additional storage licenses!

Note that licensed and replicated micro-service in Kubernetes/OpenShift can be serving requests with any number of Pods/Containers, and it doesn't impact SWIFT licensing. Similarly, the scale (in terms of number of Pods/Containers) of application micro-services can change anytime without impacting SWIFT licensing as SWIFT only looks at and tracks micro-services for licensing.

With any app licenses that you purchase, you can replicate all container image registries that are used with your application without any additional cost.

Both application and storage licenses aren't perpetual.

For any specific licensing queries as well as discounts, please contact RackWare Licensing Support.

### Is there an evaluation version available for SWIFT?

Yes. You can contact us here or email our Support to download a copy of SWIFT. SWIFT deploys with an evaluation license, which allows you free cross-cluster replications of two applications (micro-services) with a total of up to 400GB of data volumes and up to four cross-container-registry replications . The freetier SWIFT license is valid for 30 days of usage.

## What next?

SWIFT is free to try! Go ahead and request your SWIFT installer here or email our Support to get your free copy. If you already downloaded and evaluated SWIFT and are now ready to go live into production with SWIFT, please contact SWIFT Licensing Support to get your license today.

For any feedback and questions, contact our Support channel, and we will be happy to help you.