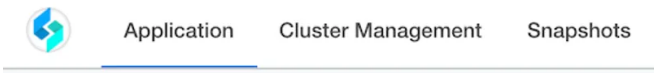
Turbine Platform Installer Guide

1. Turbine Platform Installer Guide

> This guide is for Turbine on-premise installs. If you are on 11.8.x, please refer to the **11.x Turbine Platform Installer Guide**.

The Turbine Platform Installer (TPI) is an administrative console where you can access and manage your Turbine instance. It is also where you will go to access upgrades to new Turbine releases, as well as any updates to the installer itself. The installer consists of an Application tab where you manage your application, version history, configuration settings, as well as a Cluster Management tab and a Snapshots tab.



Cluster Management is where you view, drain, and add nodes to your instance. Follow the installation steps based on how you are managing your clusters.

Cluster Type	Description
Embedded	Install Docker, Kubernetes, and the components required for a working cluster onto your servers. Turbine is then installed into that Kubernetes cluster, referred to here as an embedded cluster.
Existing	Install Turbine into an existing Kubernetes cluster.

You can then use Snapshots to backup and restore your instance of Turbine.

1.1. Embedded Cluster Installation

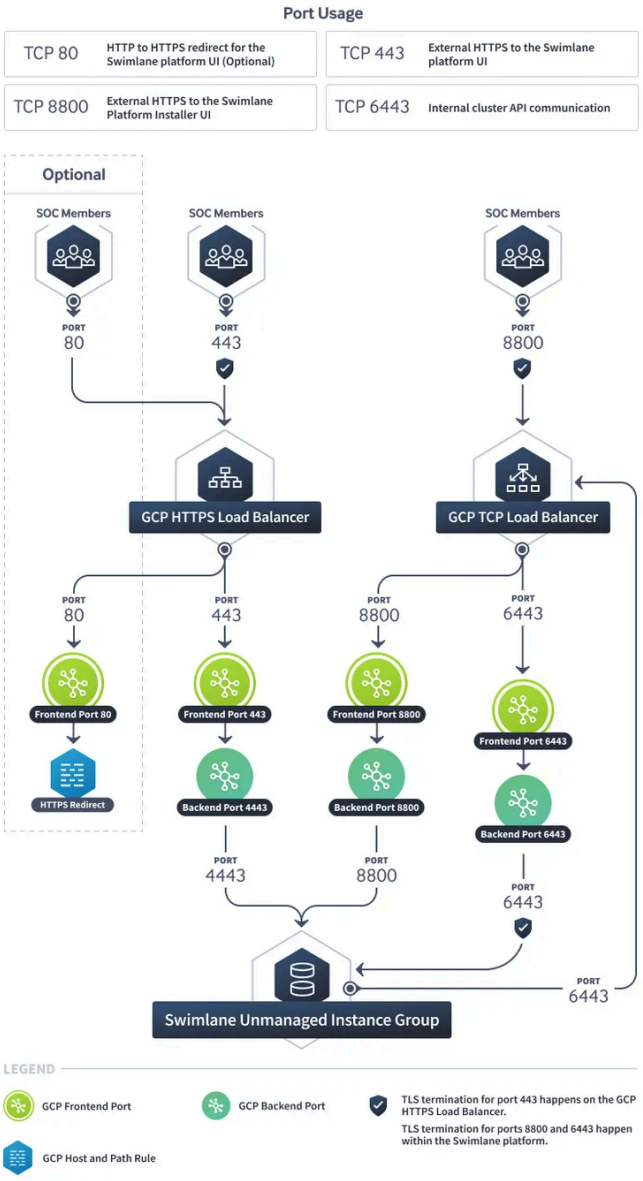
This section covers the installation of Swimlane Turbine on an embedded cluster. You will install Docker, Kubernetes, and the components required for a working cluster onto your servers. Turbine is then installed into that Kubernetes cluster, referred to here as an embedded cluster.

System Requirements for an Embedded

1.1.1.1.7. GCP HTTPS Load Balancer

This topic explains how to use an GCP HTTPS Load Balancer (Layer 7) for your Swimlane Web UI deployment.

Architecture Diagram



Limitations

Due to the limit of ports that may be used with a GCP HTTPS Load Balancer, this type of load balancer may only be used to load balance the traffic of the Swimlane Web UI. A separate TCP load balancer must be used to load balance the other necessary ports for the Turbine Platform Installer(TPI) Console and Kubernetes API.

1.1.5. Air Gap Installation with kURL

This topic walks through the process of installing Turbine in an air-gapped environment using kURL. Air gap refers to a system that resides in a private network, air-gapped from external networks or the internet. kURL is the Kubernetes installer used to initialize the environment in Swimlane's embedded cluster installation option.

Swimlane provides the following:

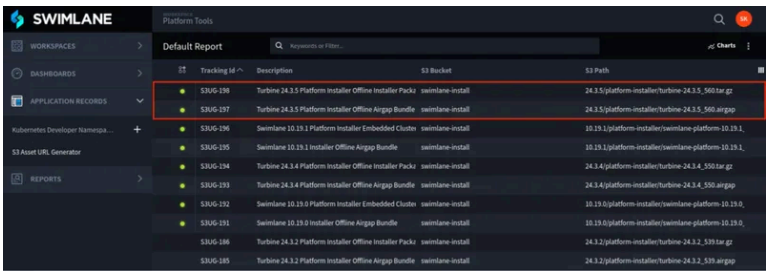
- The offline installer package
- The air gap bundle
- A KOTS application license (.yaml)
- A Swimlane license (.lic)

The offline installer package contains the install script and all the software dependencies, such as Kubernetes, to initialize the environment. The airgap bundle contains the dependencies for the actual Turbine application. When it comes time to update your Turbine version, you will need a new installer package and airgap bundle. Reach out to your Swimlane support representative for access to the new installer package and airgap bundle.

On your end, you need:

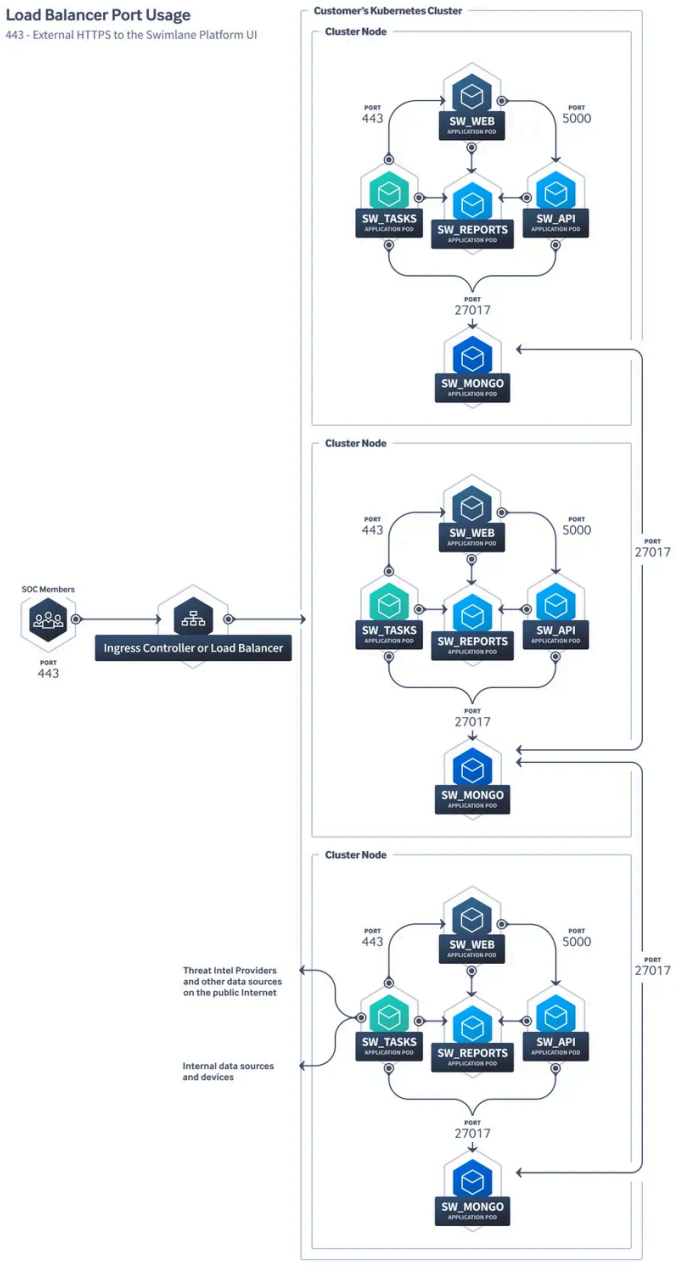
- a jumpbox with access to both the internet and the private network.
- a set of VMs that will be the nodes of your Kubernetes cluster. They must meet the **Prerequisites for Airgap Installation on Rocky Linux or RHEL 9.x** mentioned in **system requirements**.
- the four dependencies listed above from Swimlane.

Before you begin, copy the offline installer package to the jumpbox, and then to each node in the cluster. Copy the airgap bundle to the jumpbox.



The goal is to initialize a Kubernetes cluster, usually with 3 control plane nodes to create high

1.2.1.1. Turbine Platform Installer Existing Cluster Architecture Diagrams



1.2.2. Install Into an Online Existing Kubernetes Cluster

Prepare the Turbine Installer Kubectl Add-on

1.3.1. Endpoint Security Exclusions for Kubernetes Nodes

Overview

Kubernetes nodes perform continuous file system operations, process creation, and network activity. Endpoint security products such as Trellix ePO use on access scanning that can interfere with these operations if not properly scoped.

To prevent performance degradation, file lock issues, and unexpected pod or node instability, specific directories, files, and processes must be excluded from on access scanning.

This guidance applies to Kubernetes clusters deployed using containerd, CNI plugins, and Replicated kURL based installations on RHEL and similar Linux distributions.

Scope

These exclusions apply to:

• Kubernetes worker and control plane nodes

• Hosts running containerd

• Systems using flannel or similar CNI plugins

• Systems using OpenEBS

• Replicated and kURL based Kubernetes installations

These exclusions should be configured for on access or real time scanning only.

Scheduled or on demand scans may still be performed during maintenance windows.

Directory exclusions

Exclude the following directories from on access scanning.

```
/var/openebs
/etc/cni
/etc/kubernetes
/etc/kurl
/opt/cni
```

1.3.11. Using Velero commands for SPI Backup and Restore

Velero is a tool for managing disaster recovery, specifically for Kubernetes (K8s) cluster resources. It provides a simple, configurable, and operationally robust way to back up your application state and associated data. If you're familiar with kubectl, Velero supports a similar model, allowing you to execute commands such as 'velero get backup' and 'velero create schedule'. The same operations can also be performed as 'velero backup get' and 'velero schedule create'. Instead of using the SPI admin UI to do backup and restore, one can also use velero CLI to achieve the same.

Create SPI backup

```
velero create backup <BackupName> OR <BackupID>
```

Schedule a backup:

```
velero create schedule <name> --schedule [flags]
Examples:
# Create a backup every 6 hours
velero create schedule NAME --schedule="0 */6 * * *"
# Create a backup every 6 hours with the @every notation
velero create schedule NAME --schedule="@every 6h"
# Create a daily backup of the web namespace
velero create schedule NAME --schedule="@every 24h" --include-namespaces web

# Create a weekly backup, each living for 90 days (2160 hours)
velero create schedule NAME --schedule="@every 168h" --ttl 2160h0m0s
```

Troubleshooting a failed backup:

Please use velero debug --backup <backupname> to generate the support bundle, and attach to the ticket, more options please refer to velero debug --help

Sample:

```
velero debug  --backup instance-2hg5z
2023/06/28 05:00:19 Collecting velero resources in namespace: velero
2023/06/28 05:00:23 Collecting velero deployment logs in namespace: velero
2023/06/28 05:00:31 Collecting log and information for backup: instance-2hg5z
```

1.3.21. Generate a Support Bundle

If you're having issues with the Turbine Platform Installer (TPI), your Swimlane support representative will likely ask you to generate a support bundle to identify the diagnostics for your issue.

Support bundles contain logs from all relevant pods, as well as other useful information from your deployment.

Airgapped Deployments

Use these instructions for an airgapped deployment.

Before you begin generating a support bundle you must first ensure that you have the support-bundle command installed.

Install the Support-Bundle Command

If you have already installed the support-bundle command, you can skip these steps.

To install the support-bundle command:

1. From a computer that is connected to the internet run:

```
curl -o support-bundle https://krew.sh/support-bundle
curl -o spec.yaml https://kots.io -H 'User-
agent:Replicated_Troubleshoot/v1beta1'
```

1. Move the files to the server:

```
scp support-bundle <server>:/path/
scp spec.yaml <server>:/path/
```

1. Next, connect to the airgapped server and make the support-bundle file executable: `chmod 777 support-bundle`
2. Execute the installer for support-bundle: `./support-bundle`
3. Finally, generate the support bundle: `kubectl support-bundle /path/to/spec.yaml`
4. Give this support bundle to Swimlane support.