



Swimlane (10x) Platform Installer Guide

CONTENTS

1. Swimlane Platform Installer Guide	2
1.1. Embedded Cluster Installation	1
1.1.1. Install Swimlane on an Embedded Kubernetes Cluster	1
1.1.2. System Requirements for an Embedded Cluster Install	1
1.1.2.1. Swimlane Platform Installer Embedded Cluster Architecture Diagrams	1
1.1.2.2. Infrastructure Examples	1
1.1.2.2.1. AWS Application Load Balancer	1
1.1.2.2.2. AWS Network Load Balancer	1
1.1.2.2.3. Azure Load Balancer	1
1.1.2.2.4. Azure Standard Application Gateway	1
1.1.2.2.5. Azure Standard_v2 Application Gateway	3
1.1.2.2.6. GCP HTTPS Load Balancer	1
1.1.2.2.7. GCP TCP Load Balancer	1
1.1.2.2.8. HAProxy Load Balancer	1
1.1.3. Overriding Installer Settings	1
1.1.3.1. Define High Availability and Load Balancer Settings	1
1.1.3.2. Define Custom Pod and Service Subnets	1
1.1.3.3. Define Proxy Settings	1

1. Swimlane Platform Installer Guide

When you need to access the SPI UI after the initial install you can access it by access port 8800 over HTTPS on any node IP (e.g. <https://:8800>).

All configuration of the Swimlane Platform Installer and Swimlane platform must be done through the SPI admin console config page. Editing or manipulating the underlying Kubernetes resources is not supported and will not be permanent since they are managed and controlled by the Swimlane Platform Installer.

Configure the Swimlane Platform for an Embedded Cluster Install

To configure the Swimlane platform for an embedded cluster install:

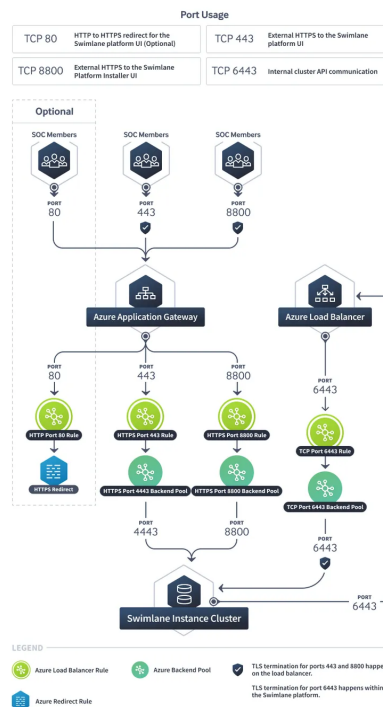
Swimlane Settings (Ingress & Web)

- On Swimlane Settings, you'll begin setting up your configuration for Swimlane. Review and set the following fields as necessary:
 - **Expose the Swimlane Web Service Externally?** – Enable this option when using a Layer 7 load balancer. The Swimlane web service will be directly exposed from each node in the cluster on TCP port 4443.
 - **Enable the Ingress Controller** – Enable this option when using a Layer 4 load balancer or for single node lab/test environments. The included ingress controller will be used for routing web requests to Swimlane on TCP port 443.
 - **Swimlane Hostname** – The DNS record pointing to the Swimlane Platform Installer and Swimlane platform load balancer.
 - **Upload a certificate for Swimlane Web?** – Enable this option to upload a certificate and key to be used for the included ingress controller. If no certificate is uploaded a self-signed one will be generated and used. The certificate must be ASCII encoded X.509 format. The private key cannot be password protected.
 - **Upload a certificate for Swimlane Web backend?** – Enable this option to upload a certificate and key to be used by the backend Swimlane web service. If no certificate is uploaded a self-signed one will be used.
 - **Swimlane Web CORS Headers** – Enable this option to set HTTP Cross-Origin Resource Sharing Headers for Swimlane Web. A list of options relevant to CORS Headers will appear below and if left blank, they will be set with the default values specified next to each option. If this option is disabled, no CORS headers will be set for Swimlane Web.
 - **Swimlane Web CORS Header Access-Control-Allow-Origin** – Determine whether to override the value for the HTTP Access-Control-Allow-Origin header.
 - **Swimlane Web CORS Header Access-Control-Allow-Headers** – Determine whether to override the value for the HTTP Access-Control-Allow-Headers header.

1.1.2.2.5. Azure Standard_v2 Application Gateway

This topic explains how to use an Azure Standard_v2 Application Gateway (Layer 7) for your Swimlane deployment. Two load balancers are required for this deployment. The Azure Standard_v2 Application Gateway (Layer 7) is used for external access to the Swimlane platform and the Swimlane Platform Installer. An additional Azure Load Balancer (Layer 4) is still required for the internal cluster communication.

Architecture Diagram



Standard_v2 Application Gateway for the Swimlane Platform and the Swimlane Platform Installer

- Create an **Azure Standard_v2 Application Gateway**
 - **Resource Group** should be set according to your organization's standards
 - **Region** should match that of the Virtual Machines that Swimlane will be installed in
 - **Tier** set to **Standard V2**
 - **Enable Autoscaling** should be set according to your organization's standards
 - **Availability Zone** set to select all zones that will have virtual machines that Swimlane will be installed on.
 - **Virtual Network** should be set according to your organization's standards
 - The virtual network that the Application Gateway is in needs to be able to communicate to the virtual network that the Swimlane virtual machines are in
 - **Frontend IP Address Type** should be set according to your organization's standards

1.1.4. Add Additional Nodes to Create an HA Cluster

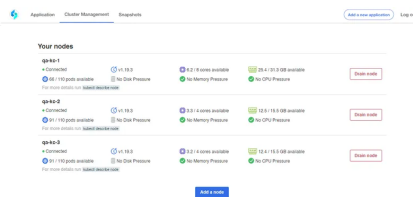
This topic provides instructions on how to create a High Availability (HA) cluster and assumes you have already run the Swimlane Platform Installer on the first node. Please see [Install Into an Embedded Kubernetes Cluster](#) if you have not yet installed on the first node.

HA considerations

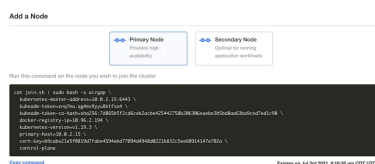
- HA clusters require at least 3 nodes and the total number of nodes must be an odd number.
- Only one MongoDB pod can run on a node. For example, you need 3 nodes in a cluster if you want to go to 3 MongoDB pods to enable a replica set.
- The first node that the initial install is running on should be in the target group to start. The additional nodes should only be added to the target groups after they've been joined to the cluster.

To add additional nodes to a cluster:

1. From the Swimlane Platform Installer UI, click the Cluster Management tab.



2. The Cluster Management page displays your installed nodes.
3. Click **Add a node**.
4. Under Add a Node, ensure that you have selected **Primary node** and then click **Copy command**.



5. **Important!** The code generated here expires after time. Swimlane recommends that you always copy the code as you work to add additional nodes. Do not save the code for use later.
6. Go to your Command-Line Interface (CLI) and paste the copied command for your additional nodes.
7. **Note:** You can add multiple nodes simultaneously.
8. If your environment requires a proxy then the `additionalNoProxyAddresses` override option must include an IP range in CIDR notation that encompasses the new node's private IP. If it does not then you will need to update the installer override file to add the new node's private IP to the `additionalNoProxyAddresses` list and specify the override file when running the `join node` command on the new node, and then re-run the `install` command on every other

1.1.12.2. Automate an Offline Embedded Installation

This topic provides automation steps for setting up an offline, single-node Swimlane installation on embedded clusters.

Installation Preparation and Customization

Before you begin, review the [System Requirements for an Embedded Cluster Install](#) to confirm your system's compliance. The Swimlane Platform Installer (SPI) performs several checks in the initial phases of the installation to ensure the underlying host is compatible with the application. To bypass any applicable checks, an installer patch YAML file can be applied at installation time to account for several of these settings. Refer to [Overriding Installer Settings](#) for more options.

Installer Patch File Example

Here is an example installer patch file:

```
apiVersion: "cluster.kurl.sh/v1beta1"
kind: "Installer"
metadata:
  name: "patch"
spec:
  kubernetes:
    HACluster: true
    loadBalancerAddress: "<Load Balancer FQDN or IP>:6443"
  firewallConfig:
    disableFirewalld: true
  selinuxConfig:
    disableSelinux: true
```

You also need the following files:

- Swimlane Platform Installer license file (ends with `.yaml`)
- Swimlane license file (ends with `.lic`)
- Swimlane tar archive (ends with `.tar.gz`)
- Swimlane airgap bundle (ends with `.airgap`)

Embedded Cluster Installation

Once you've accounted for any customizations and system checks, install an embedded cluster by running the following command:

```
tar -zxvf <Swimlane Airgap Bundle>.tar.gz
cat install.sh | bash -s installer-spec-file=<Path to Patch File>.yaml
```

Swimlane Installation and Deployment

After the embedded Kubernetes cluster has been installed, install Swimlane with the following command:

1.2.7. Backup and Restore on an Existing Cluster with Snapshots

The Swimlane Platform Installer (SPI) uses Velero and Restic as the underlying technology for snapshots and restores.

Note the following bulleted items that list limitations with snapshots:

Taking a snapshot requires enough free disk space for a compressed archive of the Swimlane database to be saved in ephemeral storage before it is uploaded to the snapshot destination. Free disk space on the cluster at `/var/lib/kubelet` should be greater than or equal to the size of the uncompressed database to ensure there is no disk pressure during the snapshot process.

Instance snapshots are considered complete and are usable in disaster recovery scenarios. Application snapshots are the legacy snapshot method and are not usable for disaster recovery. Instance snapshots can be used in place of the legacy application snapshots.

Snapshots cannot be restored to a different namespace than when the snapshot was taken.

Snapshots cannot be restored to different installation methods. For example, online cluster snapshots cannot be restored to offline clusters.

AWS S3 buckets with a bucket policy that requires the server-side encryption header are not supported. If you need to require server-side encryption for objects, Swimlane recommends that you enable default encryption on the bucket itself instead.

Cleanup and removal of snapshots can only be done through the SPI admin console snapshots tab. Removing data from the snapshot storage itself will result in data corruption and loss of snapshots.

Install Velero

See Velero's [Customize Resource Requests and Limits](#) if you need to set the resource requests and limits for the Velero and Restic pods.

Install Velero (Online)

Velero install instructions if your cluster has access to the internet:

1. Download Velero 1.9.0 for the OS version from which you run `kubectl` commands:

```
https://github.com/vmware-tanzu/velero/releases/tag/v1.9.0
```

1. Untar the file and then change directory into the uncompressed directory:

```
tar xzf <FILENAME>.tar.gz cd <DIRECTORY>
```

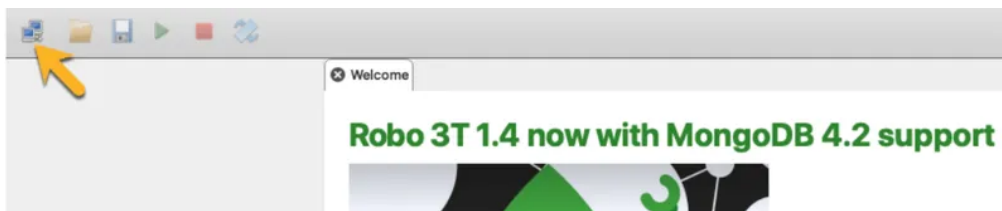
1.3.2. Installing Robo 3T and running a query for MongoDB

Step 1: Installing Robo 3T

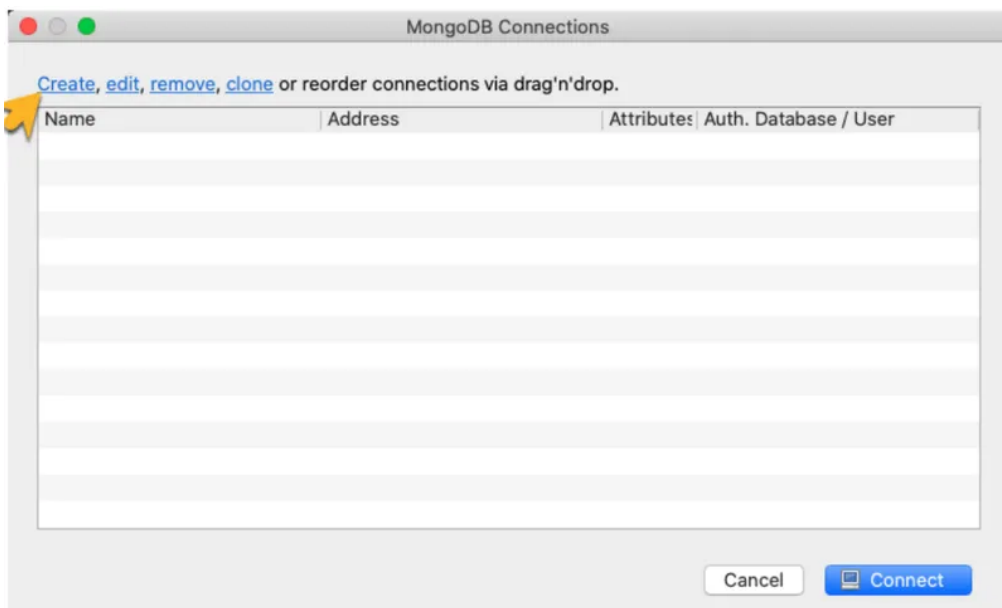
1. Download Robo 3T version on your respective OS (we are using Mac OS for this article) from <https://robomongo.org/download>.

Step 2: Setting up and Connecting to a single host Linux CentOS system

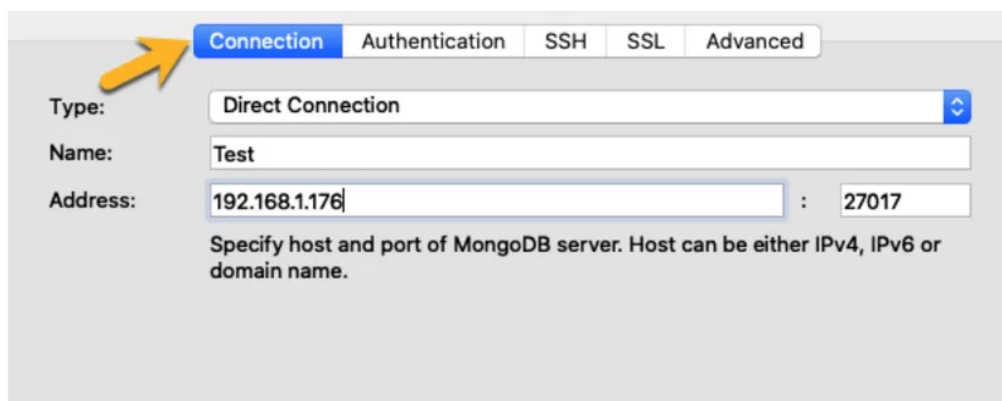
1. Click on the two computers at the upper left-hand corner as shown in the screenshot below:



2. This will open the connection window then click on create as shown below:



3. Click the connection tab:



1.3.12. 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
2023/06/28 05:00:37 Generated debug information bundle: /root/bundle-2023-06-28-05-00-18.tar.gz
```

To check the status and get information on the backup:

```
velero backup describe <BackupName or ID>
velero backup logs <BackupName or ID>
```

To fetch the logs of the Velero server pod. This provides the output of the Velero server processes:

```
kubectl logs deployment/velero -n velero
```

Getting velero debug logs. You can increase the verbosity of the Velero server by editing your Velero deployment to look like this:

How to Reduce the Restic Prune Interval to Retrieve Deleted Velero Backup Disc Space

When deleting velero backups, the 'delete backup' commands are asynchronous: they are queued and executed later by restic like garbage collection. The default restic prune interval is seven days, so it can take up to seven days before a velero backup is actually deleted from disc.

To retrieve disc space from deleted backups more quickly, you can force restic to prune 'delete backup' commands within five to ten minutes by executing:

```
kubectl -n velero get resticrepositories -oname \
| xargs -I{} kubectl -n velero patch {} \
--type='json' \
-p ' [{"op": "replace", "path": "/status/lastMaintenanceTime", "value": "2020-01-01T00:00:00Z"} ] '
```

Resetting Swimlane Platform 1.3.23. Installer (SPI) Console Admin Password

Once the admin password is stored it cannot be read back, the only way to recover is to reset it.

```
kubectl kots reset-password <namespace>
```

This is also valid syntax to change the password:

```
kubectl kots reset-password -n <namespace>
```

NOTE: for embedded clusters the namespace is default

1.3.24. How to Work Around SPI Installer Incompatible Version Problem

Occasionally, an SPI upgrade preflight check may fail with "Platform Installer (x.x.x) is incompatible with this release."



If this happens, then perform this workaround to proceed with the upgrade.

1. Delete the kurl-last-config configmap:

```
kubectl delete pod -l app=kotsadm-fs-minio
```

1.3.32. kubectl commands are failing due to proxy

Kubectl commands may fail with the following error:

```
E0403 14:04:05.944359 3116197 memcache.go:265] couldn't get current server API group list: Get "https://swimlane.dev.swimlane.example.com:6443/api?timeout=32s": Service Unavailable
```

1. Check if there is a proxy enabled in the environment

```
env | grep -i proxy
```

The resulting output should be as below if there is a proxy enabled in place:

```
https_proxy=http://192.168.x.x:8080
http_proxy=http://192.168.x.x:8080
```

2. Disable proxy from the node

```
export
no_proxy=127.0.0.1,192.168.x.x,.default,kubernetes,.local,localhost,.svc,load
balancer fqdn
```

The above should also be added to **\$HOME/.bashrc** otherwise a node reboot will wipe it.

3. Check to make sure the proxy is disabled.

```
env | grep -i proxy
```

The output from above should now be as below.

```
https_proxy=http://192.168.x.x:8080
http_proxy=http://192.168.x.x:8080
no_proxy=127.0.0.1,192.168.x.x,.default,kubernetes,.local,localhost,.svc,load
balancer fqdn
```

4. kubectl commands should work now.

1.3.33. Upload a New TLS Certificate

If you've already gone through the setup process once, and you want to upload new TLS certificates, run this command to restore the ability to upload new TLS certificates:

```
kubectl -n default annotate secret kotsadm-tls acceptAnonymousUploads=1
```

Adding this annotation temporarily creates a vulnerability for an attacker to maliciously upload TLS certificates. Once TLS certificates have been uploaded again, the vulnerability goes away.