Solution-Soft®

WHITE PAPER

USING TIME MACHINE® POD TO TIME TRAVEL ON OPENSHIFT

USING VIRTUAL CLOCKS FOR TIME SHIFT TESTING

Time Machine creates software virtual clocks that allow you to time travel applications into the future or the past, and in that way perform time shift testing of date and time sensitive application logic, such as month-end, quarter-end, year-end processing, billing cycle, debt aging, regulation change date, etc.

TIME MACHINE POD/DEPLOYMENT

Time Machine pod/deployment (created from a Time Machine container image), enables time travel of desired target OpenShift pods/deployments. To time travel a pod where your application is, you just need to create a Time Machine deployment in the same namespace, without the need of changing the container images of the application you're using, or adding containers to the pod with your application.

Time travelling is done on the namespace level where all the (configured) target pods/deployments see the same virtual time.

Please note, if you need to time travel pods running in multiple namespaces, you need to create a different Time Machine deployment in each of those namespaces, as will be explained later in the document.

BENEFITS OF USING TIME MACHINE POD/DEPLOYMENT

Using Time Machine pod/deployment is the single most convenient way to time travel your applications running in an OpenShift cluster. The benefits offered by this approach are numerous:

1. No changes are made to the existing container/app images you are using. The way to time travel them is simply to add Time Machine pod/deployment to the namespace where the target pods/deployments you want to time travel are running.

2. Time Machine is running as non-root user, which is in line with default security restrictions in OpenShift, and will adhere to any security constraints of an enterprise environment.

3. All the other supplementary Time Machine products (see next section: "*Setting up the Environment*") are typically installed and running on separate system(s) outside the cluster.

4. Time travelling is done on the namespace level - virtual clock affects all the pods of any deployment configured to use Time Machine in that namespace, which is especially convenient for multi-pod deployments and multi-container pods.

5. Connecting to Time Machine service to create/remove virtual clocks is done via respective OpenShift Route, regardless of the actual pod iteration running.

6. Licensing of newly created pods is done automatically by the Time Machine Floating License Server (see next section: "*Setting up the Environment*"), which provides flexibility to support scaling in a dynamic environment.

7. Simultaneous time travelling of multiple pods across different namespaces is done using Time Machine Sync Server (see next section: "*Setting up the Environment*"), which is also capable of broadcasting virtual clocks outside the OpenShift Cluster.

   This means that besides synchronizing time travelling your pods inside the cluster, you can also synchronize them with other targets, such as pods in a different cluster, standalone Docker containers, or any other Time Machine virtual clock running on a physical or virtual server, on premise or in the cloud (assuming you have network visibility from the Sync Server host to the desired target).

   Time Machine Sync Server is also the enabler for test automation, since it facilitates fully automating time traveling via the built-in URL API.

SETTING UP THE ENVIRONMENT

Beside the Time Machine pod/deployment, you'll need the following supplementary Solution-Soft products in order to successfully manage Time Machine in your OpenShift cluster:

1. **Time Machine Floating License Server** (TMFLS), which will provide licensing for each Time Machine deployment/pod created (the number of licenses available will depend on the size of the license unit pool of the TMFLS).

   TMFLS needs to be installed on a system which can be accessed by OpenShift cluster nodes.

2. **Time Machine Enterprise Console** (TMEMC), Java based GUI that will allow you to create and remove virtual clocks from the desired pods/deployments, and also serve as GUI for TM Sync Server (see next bullet).

   TMEMC can be installed on any Windows system able to access routes of your OpenShift Cluster (even your laptop).

3. *Optionally – **Time Machine Sync Server** (TMSS), in case you need to simultaneously time travel pods across different namespaces to the same virtual time and/or automate time travelling via URL API.

Please note that all of these products are typically installed on system(s) outside the OpenShift cluster, but with network visibility to the cluster. While TMFLS and TMSS are available for both Windows and Linux, TMEMC is only available for Windows OS (though it can manage Linux targets, such as containers running in OpenShift pods).

The suggested setup could be to have all three products (TMFLS, TMEMC and TMSS) on the same Windows system (any physical or virtual Windows machine, on premise or in the cloud, or even a PC, or your laptop), having network access to the cluster, but you could choose to put them on separate systems as well, if needed.

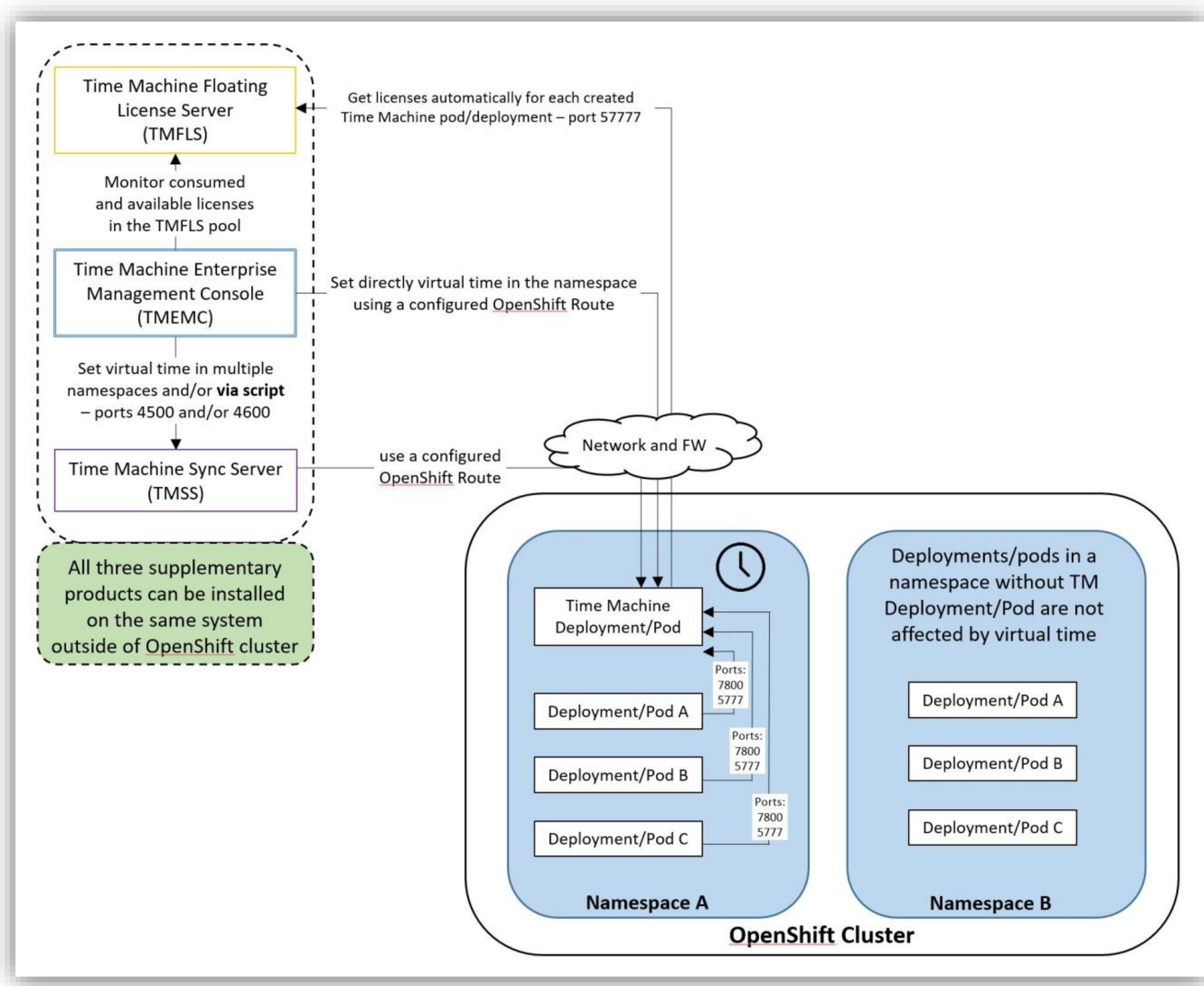
For more details on how to install and license these supplementary products, please refer to their respective user manuals, available upon request from Solution-Soft.

Solution-Soft also provides an AWS image (BYOL AMI), available on the AWS Marketplace, with all three supplementary products preinstalled (and only need to be licensed), which you can locate on the following URL:

https://aws.amazon.com/marketplace/pp/B08X6RVZBV

To obtain licenses for the pre-installed products on the AWS image, please follow the instructions provided in respective documents available on the launched instance, or simply reach out to: support@solution-soft.com

The diagram below explains a typical environment setup when using Time Machine pod:



GRAPHIC 1

CREATING A TIME MACHINE DEPLOYMENT/APPLICATION

Create a new application/deployment in the existing namespace/project, by using a TM container image.

Time Machine container image can be downloaded from the Docker Hub, from the repository:

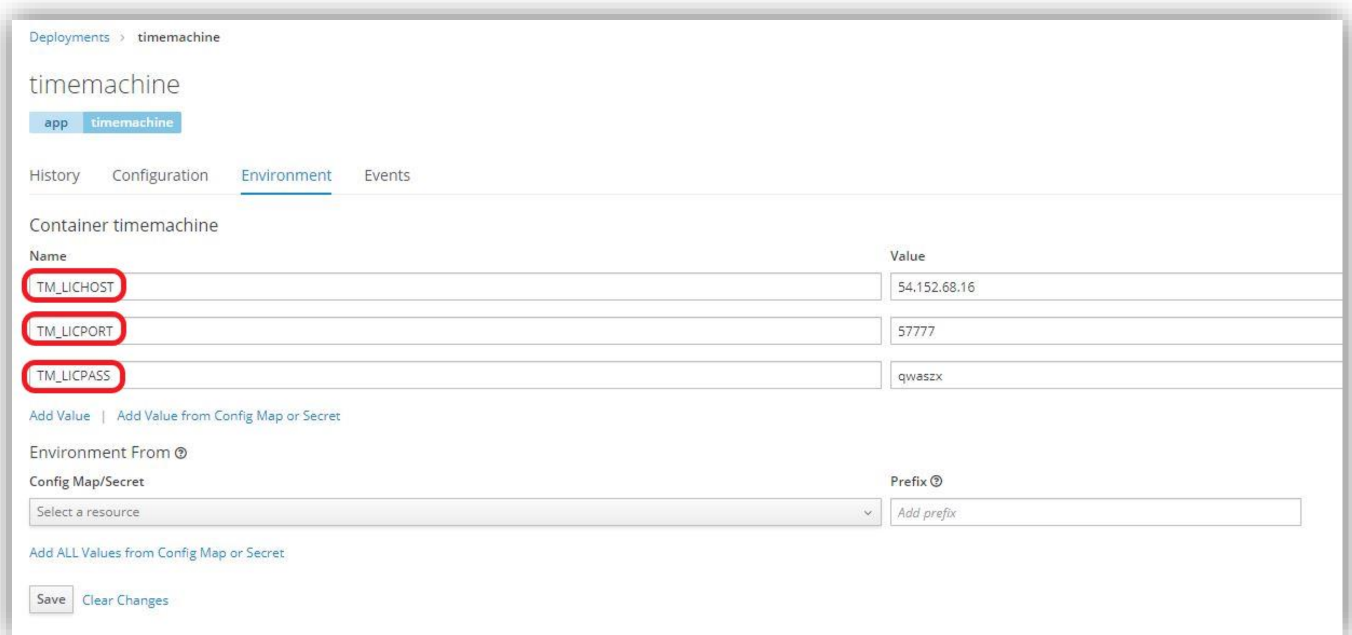
solutionsoft/timemachine-sidecar

Depending on the version of OpenShift you're using, you can either directly create a new deployment via respective yaml file in GUI, where you'll specify the container image above, or run the following via command line:

oc new-app solutionsoft/timemachine-sidecar --name=timemachine

Once the deployment becomes active, we'll need to specify the environment variables for the Time Machine deployment, with the necessary information about TM Floating License Server (TMFLS) we'll use to provide the license.

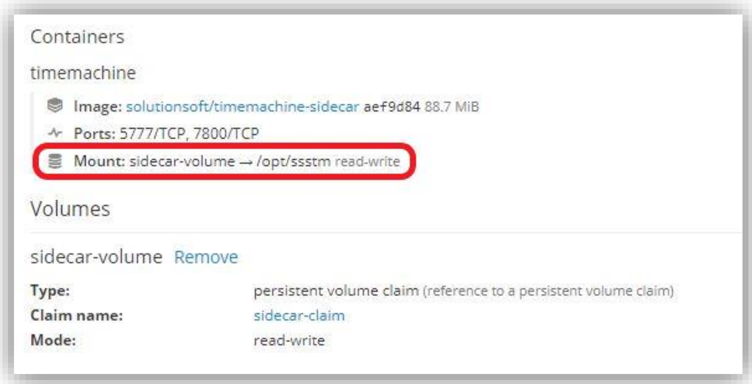
There are three environment variables that need to be specified: **TM_LICHOST**, **TM_LICPORT** and **TM_LICPASS** (depending on TMFLS that's available to you). TM_LICHOST is the IP address of the host on which TMFLS is running. TM_LICPORT is the TCP port number on which TMFLS listens for requests. TM_LICPASS is the Security Key that must match the value configured in TMFLS. The values for the security key and the listening port are configured in the TMFLS configuration file that's located on the TMFLS host under **opt/solutionsoft/tmlicserver/conf/licserver.cf**



GRAPHIC 2

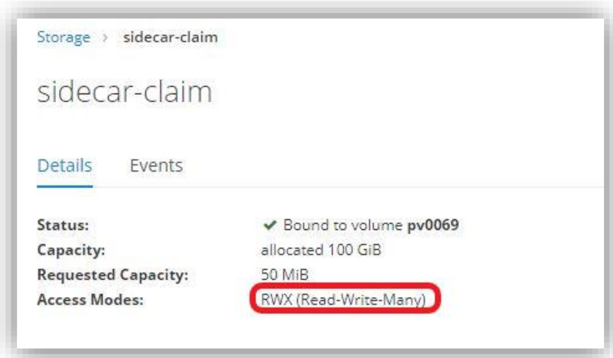
You can read more about TMFLS installation and configuration in the TMFLS documentation, available on request from Solution-Soft.

Next, we will create a new persistent volume claim with Shared Access mode (RWX), and then add storage (using this claim) to Time Machine deployment under mount path **/opt/ssstm**



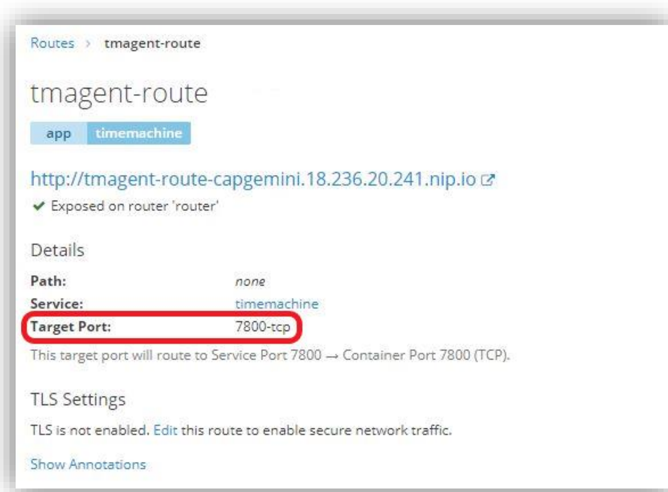
GRAPHIC 3

Please note that the size of the claim does not need to exceed more than 50 MB.



GRAPHIC 4

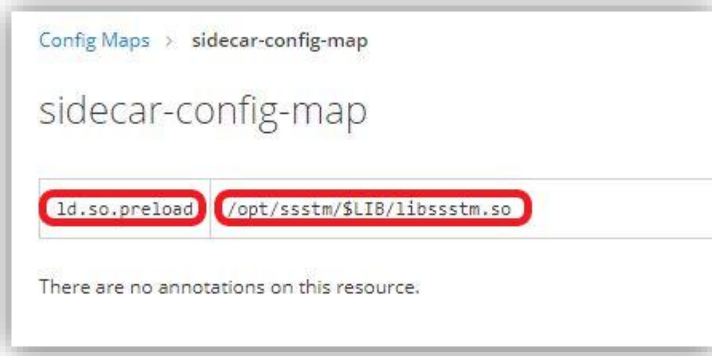
Finally, in order to expose the Time Machine service (only on port 7800, while it is also running on port 5777 which does not need to be exposed in a route), so that you can connect to it via **Time Machine Enterprise Console** (GUI) or TM Sync Server, you'll need to create a Route.



GRAPHIC 5

Once finished, we'll get a URL route that will be used as a connection string in the TM Enterprise Management Console.

At this point, we should also create a **Config Map** (we can name it *sidecar-config-map*) with the key: **ld.so.pre-load** and the value: **/opt/ssstm/$LIB/libssstm.so**



GRAPHIC 6

We'll add this config map later on to a target deployment that we want to time travel, by editing the deploy-ment yaml file, as explained in the next section.

CONFIGURING A TARGET DEPLOYMENT TO BE AFFECTED BY VIRTUAL TIME

To configure a deployment you want to be affected by virtual time, you need to complete the following steps:

1. Add Storage to the deployment, using the persistence volume claim we created in the previous section.

   The mount path should be: **/opt/ssstm**

2. Mount the ConfigMap we created in the previous section (named 'timemachine-configmap') to the deploy-ment.

   The mount path should be: **/etc/ld.so.preload**

   The subPath should be: **ld.so.preload**

   For example, you can edit your yaml file like below:

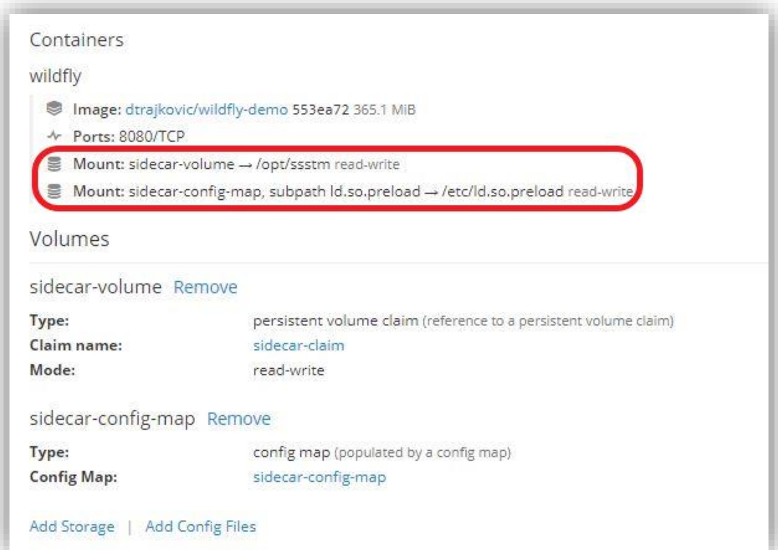
   Under volumeMounts for the container that is to be time travelled, we're adding a mountPath as below:

```
- mountPath: /etc/ld.so.preload
  name: sidecar-config-map
  subPath: ld.so.preload
```

Also, under volumes, we're specifying the config map as another available volume:

- configMap:
defaultMode: 420
name: sidecar-config-map
name: sidecar-config-map

You can see an example of added storage and config map in the image below:



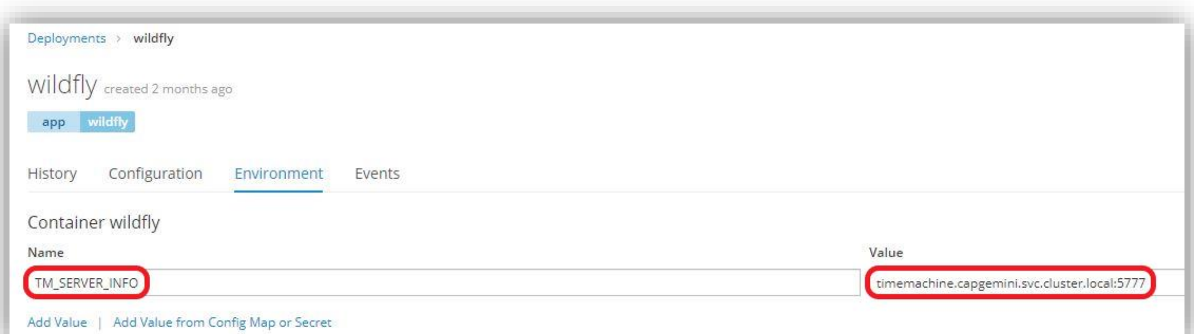
GRAPHIC 7

3. Add Environment variable to the deployment, as shown below:

NAME: **TM_SERVER_INFO**

VALUE: **<service>.<pod_namespace>.svc.cluster.local:5777**

where **<service>** is the name of the service created for Time Machine Deployment, and **<pod_namespace>** is the actual namespace where the deployment/pods you want affected by virtual time are.



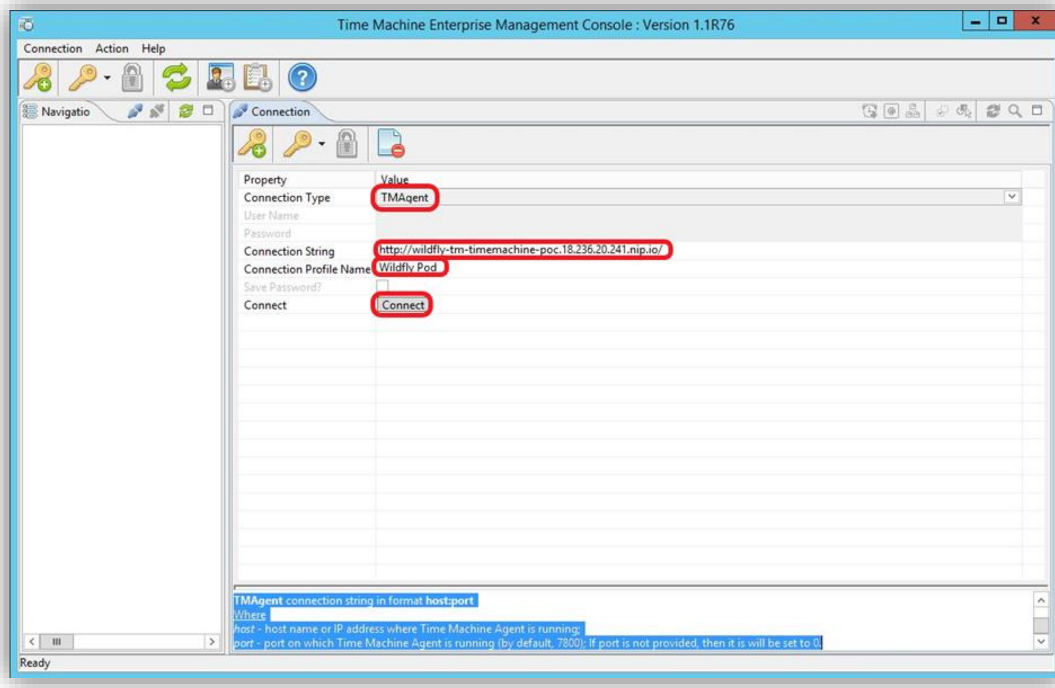
GRAPHIC 8

After you go through the listed steps, your deployment/pod should be ready to be time travelled with Time Machine.

CONNECTING TO TIME MACHINE AND CREATING VIRTUAL CLOCKS

We can now go ahead and create a connection from the TM Enterprise Management Console (TMEMC) that was previously installed on a different system, as mentioned in the earlier section "Setting up the Environment".

To do so, just click the **New Connection** button (icon of a key with a green plus sign) in the TM Enterprise Management Console toolbar and the connection panel similar to below will be displayed:

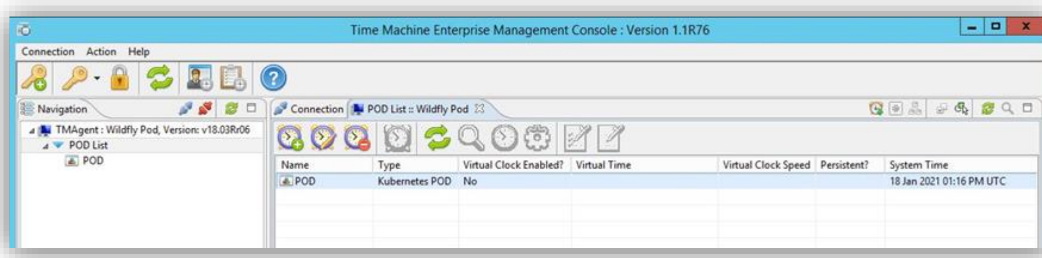


GRAPHIC 9

Configure the following connection settings for remote Time Machine:

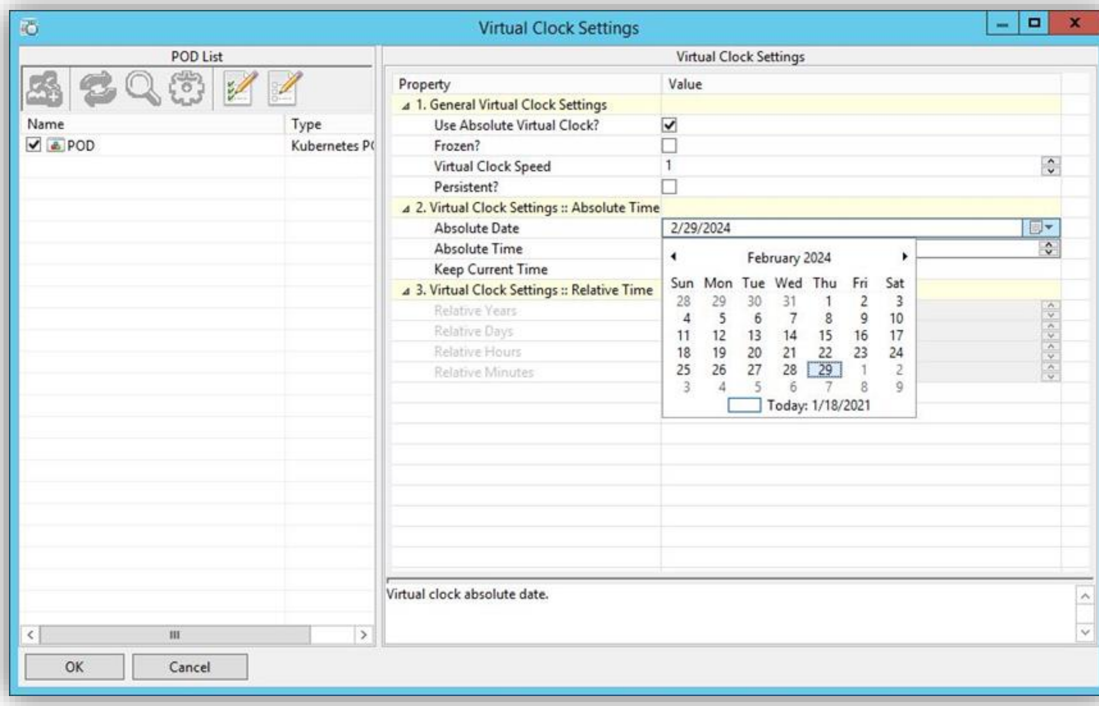
- **Connection Type** – choose **TMAgent** connection type;
- **Connection String** – use the address specified as the Route that exposes Time Machine service for your OpenShift deployment.
- **Connection Profile Name** – optional name of the connection profile

Finally, click Connect button to connect to Time Machine service in the pod, which should look similar to below:



GRAPHIC 10

To create a virtual clock for the pod, either click on the **Add Virtual Clock** button (icon of a clock with a green plus sign), or simply double click on the listed POD.



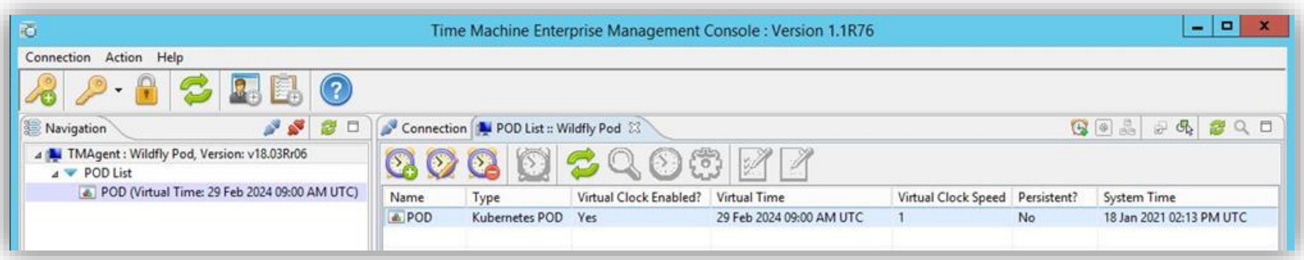
GRAPHIC 11

You can configure the following parameters for pod virtual clocks:

- **Use Absolute Virtual Clock?** – enables an absolute date and time for the virtual clock. If unchecked, a relative time offset from the system time can be used.
- **Frozen?** – determines if the virtual clock is frozen or running.
- **Clock Speed** – configures the speed of virtual clock. This parameter is always 0 for frozen virtual clocks (e.g. value of 3 means that the clock is three times faster than system clock, while -3 means that it is 3 times slower than the system clock).
- **Absolute Date & Time** – sets the exact virtual time for virtual clock.
- **Relative Years, Days, Hours, Minutes** – enables you to use relative time for virtual clock. The virtual time is calculated from the current time using the specified relative parameters. Positive values increment virtual time, negative values decrement virtual time.

Please note that you are allowed to configure either Absolute Time or Relative Time parameters, but not both for a single virtual clock.

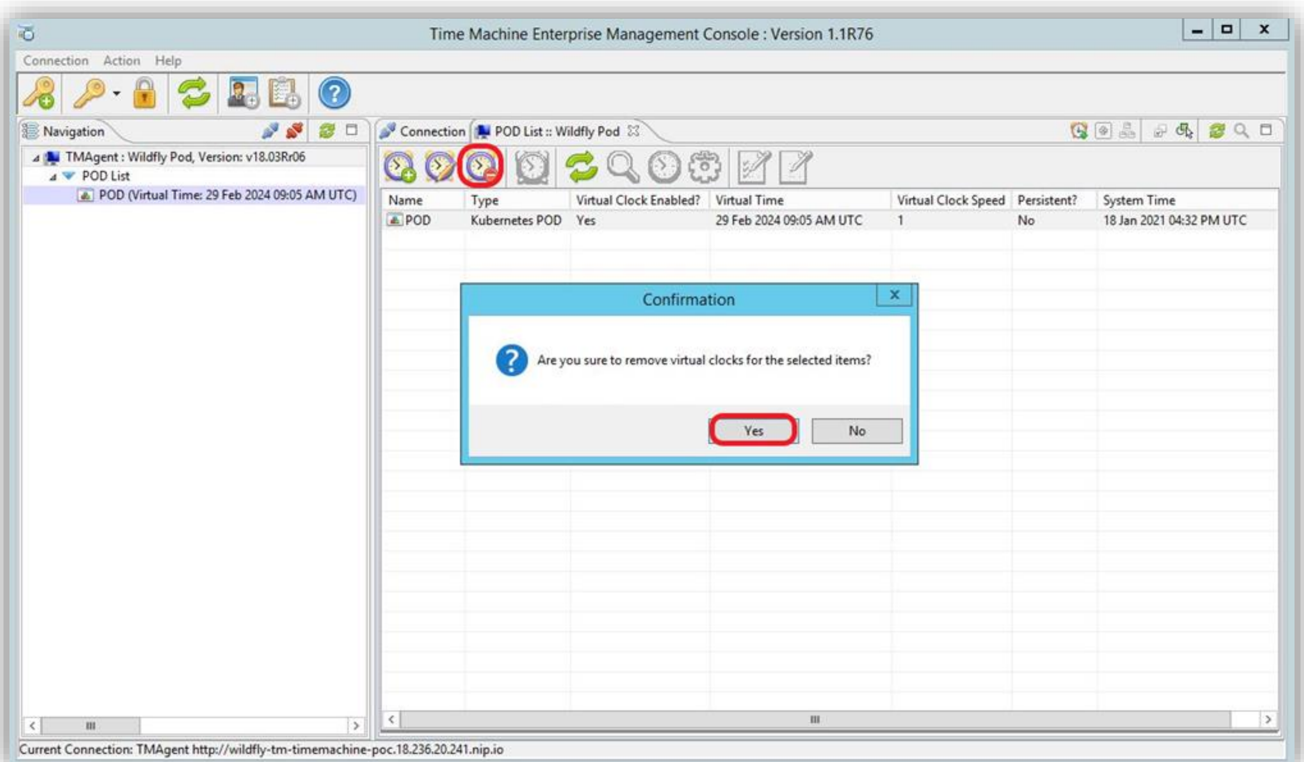
Click OK to enable the virtual clock for all the selected items. After that, you will see the pod displayed with new virtual time in the Pod List panel. You can also click Refresh list toolbar button to refresh the list with active virtual clocks, as seen below:



GRAPHIC 12

In order to confirm whether the target pods configured to see virtual time, actually see virtual time correctly, we can to go to the respective pods, and check in the terminal window whether virtual time is visible.

Finally, if we want to revert back to the system time we just need to remove the virtual clock, and to do that select the pod in the Pod List Panel and click Remove Virtual Clock toolbar button (icon of a clock with a red minus sign):



GRAPHIC 13

CREATING MULTIPLE DIFFERENT VIRTUAL CLOCKS IN THE SAME NAMESPACE

In the previous section we saw that by creating a deployment with Time Machine container image, we can time travel any other target deployment (configured accordingly) in the same namespace, and all of the targets would then see the same virtual time.

In case there is a need to have multiple different virtual clocks in the same namespace, we can achieve that by creating multiple Time Machine deployments.

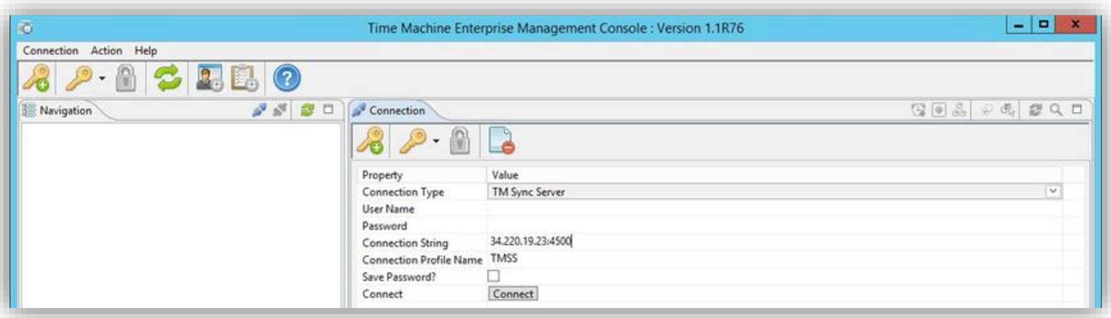
That means that you will need to go through the steps described earlier (and create separate persistent volume claims and routes).

You would then connect to multiple Time Machine deployments via TM Enterprise Management Console, creating multiple connections, and as connection strings use respective routes.

SIMULTANEOUSLY TIME TRAVELLING PODS ACROSS DIFFERENT NAMESPACES USING TMSS

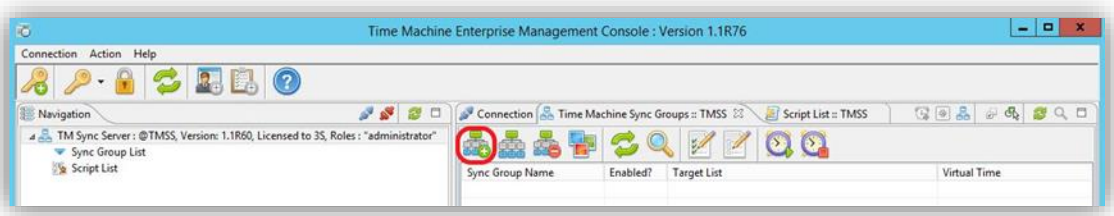
If there is a need to simultaneously time travel pods across different namespaces, or to be more precise, different Time Machine deployments and the target pods configured to get virtual time from them (be it in the different or same namespace), we need to use Time Machine Sync Server (TMSS).

To connect to a previously deployed TMSS (as mentioned in the earlier section “Setting up the Environment”), we’ll use the same TMEMC GUI, by clicking on the **New Connection** button, choosing the TM Sync Server connection type, and specifying the **Connection String** (depending on the details of the actual TMSS that’s available to you):



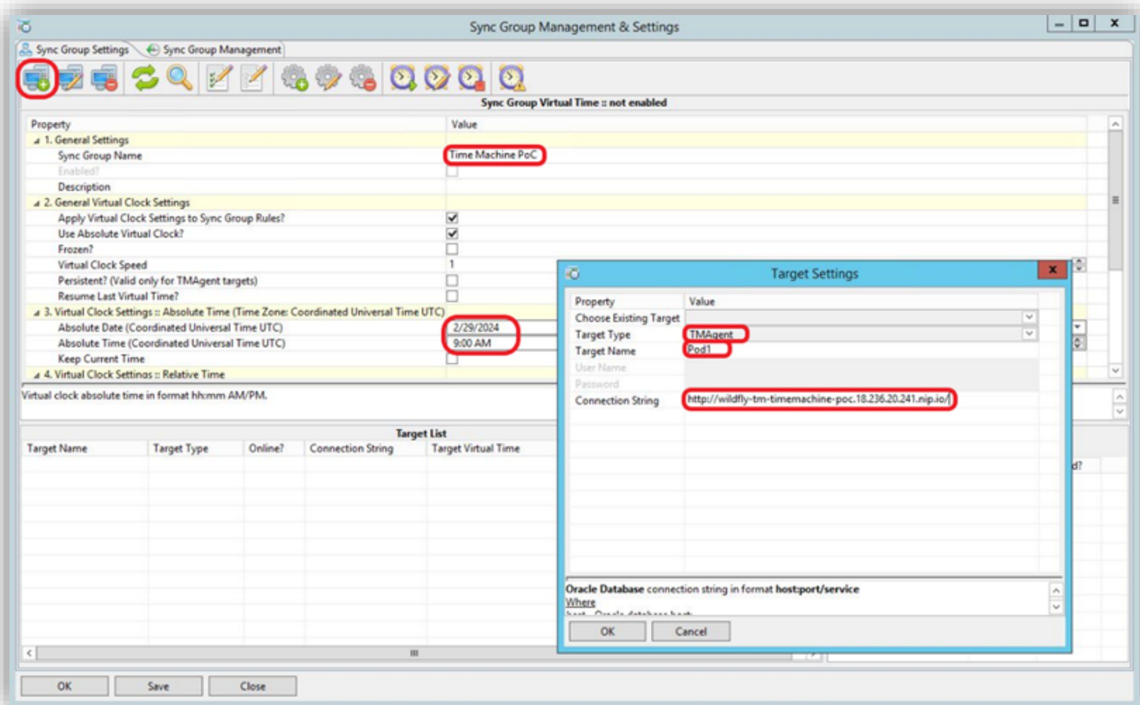
GRAPHIC 14

TMSS uses the concept of “Sync Groups” to group together the targets that you simultaneously want to time travel. Once connected to TMSS, we’ll click on the Add New Sync Group button:



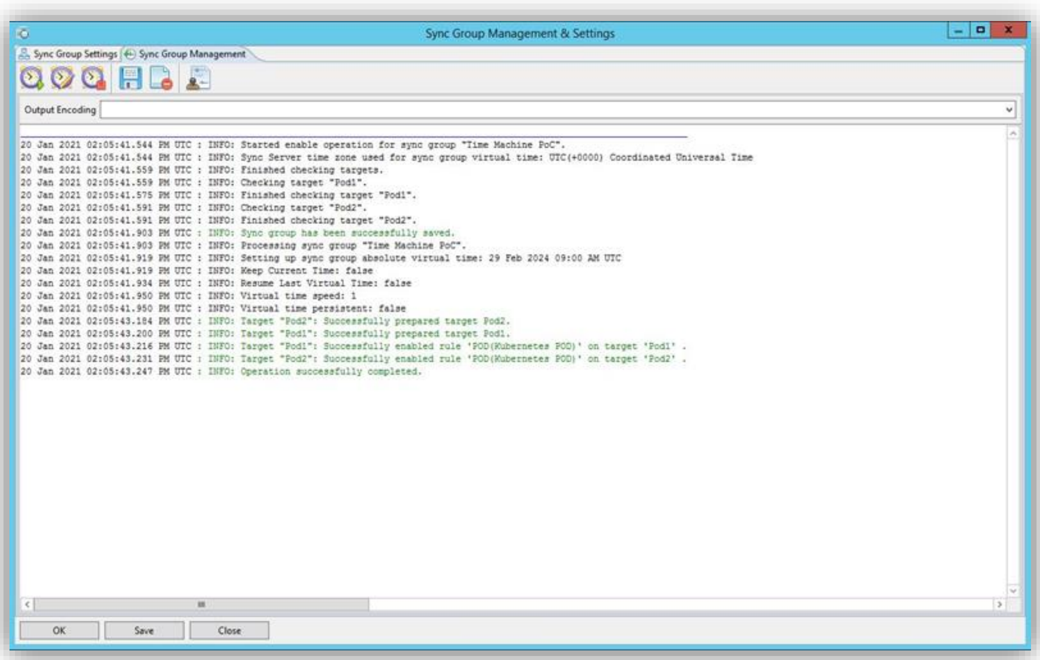
GRAPHIC 15

In the new Sync Group Management & Settings window, we should specify the sync group name and the virtual time desired, and then click on the **Add Target** button in the top left corner to add desired Time Machine deployment(s) to the group, by specifying the respective route(s) for Time Machine service(s) on port 7800:



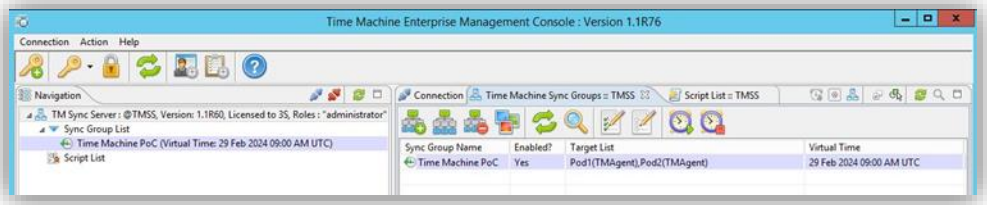
GRAPHIC 16

Finally, to create virtual clocks in both pods at the same time, we'll click on the **Enable Sync Group** button, which is also marked in the picture above (at the middle of the button ribbon). TMSS will check if the targets are available and set virtual time on both of them, as seen in the picture below:



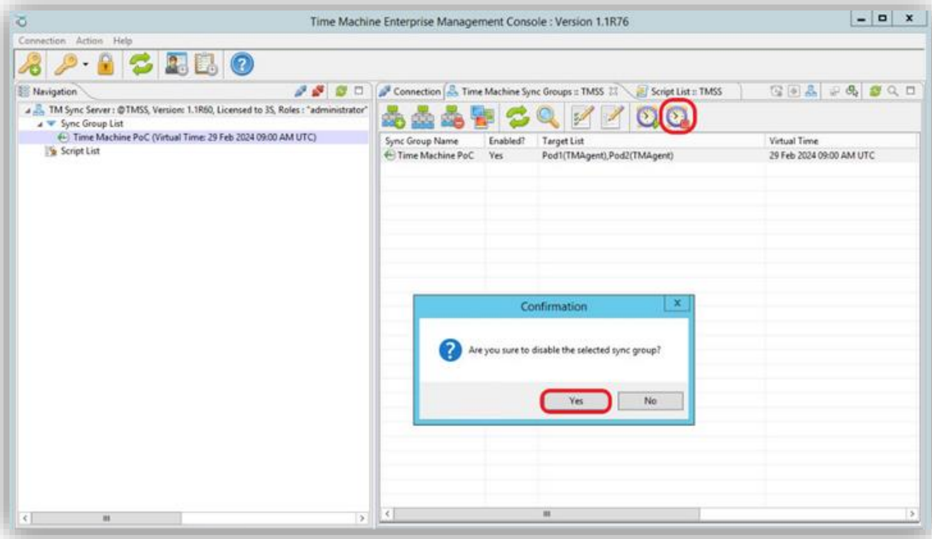
GRAPHIC 17

After we click on the **Close** button, we'll see that our sync group is displayed in the Console with virtual time:



GRAPHIC 18

To remove created virtual clocks and revert back to the system time, we just need to disable the sync group:



GRAPHIC 19

Please note that TMSS offers the possibility of fully automating time traveling via the built-in URL API, using any programming language that can make simple web service calls (via http or https), and automation frameworks of your choice.

For more details on TMSS API, please refer to the TMSS Manual, available in the installation folder, or upon request from the Solution-Soft team.

Solution-Soft®

SolutionSoft Systems, Inc.
2350 Mission College Blvd., Suite #777
Santa Clara, CA 95054, U.S.A.
Phone: 1.408.346.1400
Sales: 1.408.346.1415

Europe
SolutionSoft Systems, Inc.
Trnska 8, Suite 7
Belgrade, Serbia, 11000
Phone: +1381 11 403 1523

www.solution-soft.com

www.facebook.com/solution-soft
www.twitter.com/solution-soft
www.linkedin.com/solution-soft

WP149_LM20210723