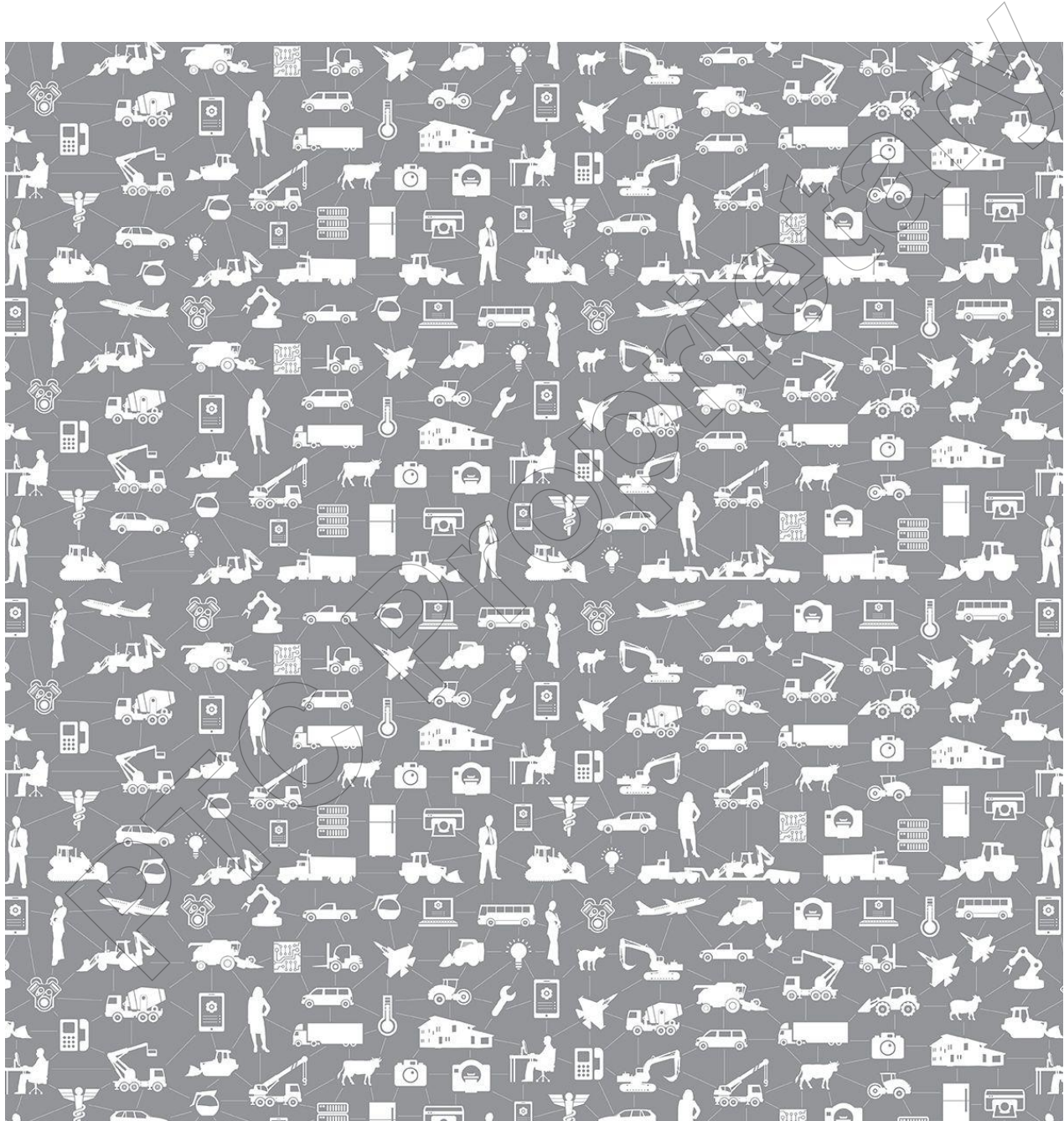


DevOps Playbook for ThingWorx



Copyright © 2021 PTC Inc. and/or Its Subsidiary Companies. All Rights Reserved.

Training guides and related documentation from PTC Inc. and its subsidiary companies (collectively "PTC") are subject to the copyright laws of the United States and other countries.

No part of this guide or documentation may be reproduced in any form without permission from PTC.

PRINTING HISTORY

Document no.
TE-00042

Date

08-04-2021

Description

Initial Printing of:
DevOps Playbook for ThingWorx

Table of Contents

Module 1 Introduction	6
Exercise 1: Creating OR Accessing VM Environment	7
Module 3: Source Control System Configuration	13
Exercise 1: Setting Up Source Control System Using GitHub.....	14
Exercise 2: Setting Up Source Control System Using Tortoise SVN	18
Module 4: CI/CD Configuration.....	21
Exercise 1: Creating a Simple Pipeline Using Jenkins.....	22
Exercise 2: Enhanced Pipeline With Automated Tests	26

Course Overview

In this course, you will review the DevOps process going through Source Control, CI/CD and Docker. You will apply it by configuring ThingWorx and Jenkins, identifying a relevant pipeline, and automated tests.

PTC Proprietary

Course Objectives

After completing this course, you will be able to:

- Install GitBackup extension
- Create a Git thing and check connection
- Configure an SVN repository
- Identify and implement a pipeline
- Identify and implement automatic tests

Module 1

Introduction

Module Overview

- Welcome to the class! This module is a discussion about course logistics and expectations. Please review them with your instructor.
- The PTC Technical Enablement team delivers this training to the PTC Internal Customer Success team and PTC Partner community.
- **PTC Internal Training – Students** should follow the instructions in Task 1 to create a VM instance on PTC Cloud Portal

Module Objective

After completing this module, you will be able to:

- Get acquainted with the Instructor and other class participants
- Be aware of classroom logistics for virtual or in-class sessions
- Get an overview of tools and systems logistics to participate in the class effectively
- Understand class expectations and actively participate in the class

Exercise 1: Creating OR Accessing VM Environment

Objective:

- Creating course specific VM instance on PTC Cloud Portal for the PTC Internal training and on Lighthouse for PTC Partner Training.

Note

Images are for reference only.

Task 1: Create VM instance on Cloud Portal - PTC Internal Training

1. Browse [Technical Enablement Student Zone Page](#) and login using your credentials.
2. Search your Course ID and note the VM template name mentioned in front of 'PTC Internal – Cloud Portal' under the VM Information column.

Example - for course ID TE-00040, 'ThingWorx Manufacturing Apps 8.3 – Training VM-1 – Windows' is the VM template name.

ILT/Virtual ILT Training Delivery Instructions

Select the details link for the course (ID) you are scheduled to deliver to find all necessary information on how to prepare, deliver and assess the course.

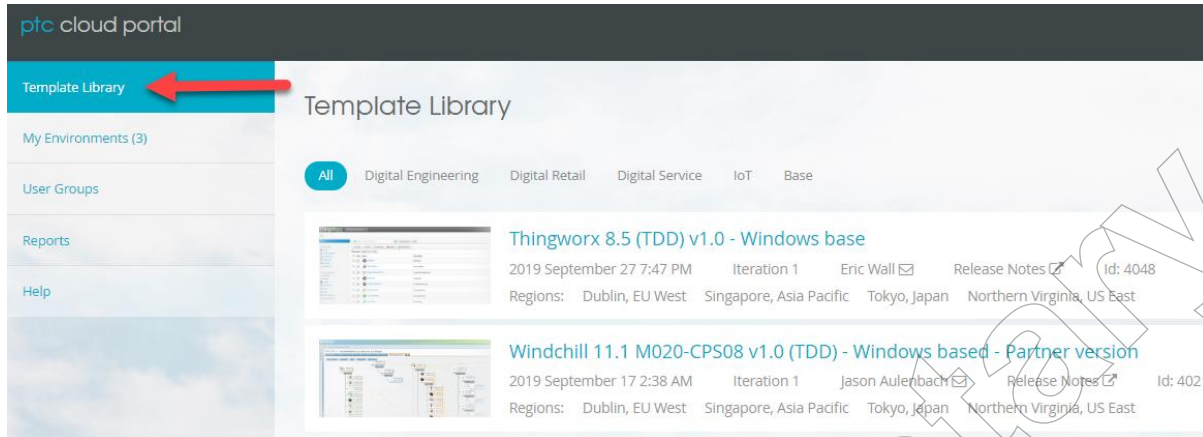
Course List

#	Course/Assessment	Course ID	Format	Duration (Hours)	Details	Curriculum/s	VM Information
4	ThingWorx Manufacturing Apps – Custom App Development	TE-00040	Virtual ILT	8	Link	• ThingWorx SCO C1 Curriculum for	<ul style="list-style-type: none"> • PTC Internal – Cloud Portal: ThingWorx Manufacturing Apps 8.3 – Training VM-1 – Windows • PTC Partners – Azure: ThingWorx_Manufacturing_Apps_8.3_SCO-02

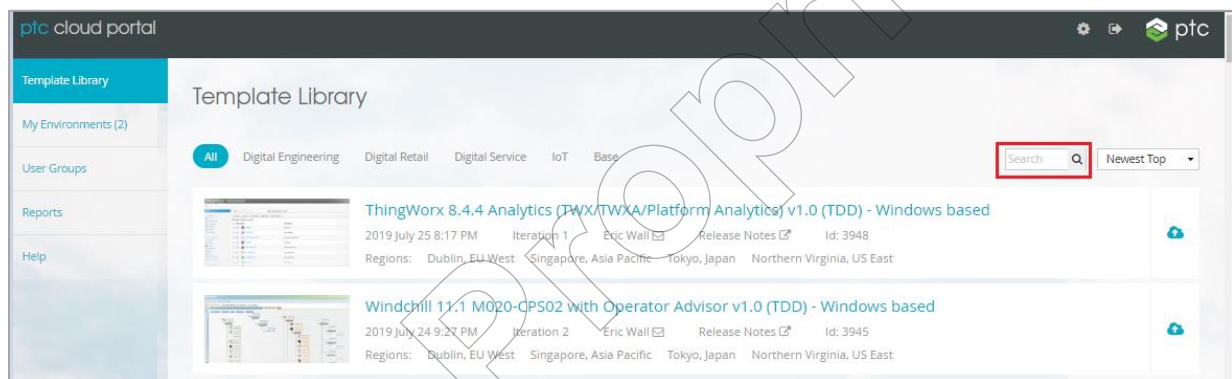
3. Browse [PTC Cloud portal](#) and sign in using your credentials. The cloud portal relies on the Amazon Web services (AWS) infrastructure to provide the VM instances templates.

4. Go through the usage policies if you are signing in for the first time and click **Agree**.

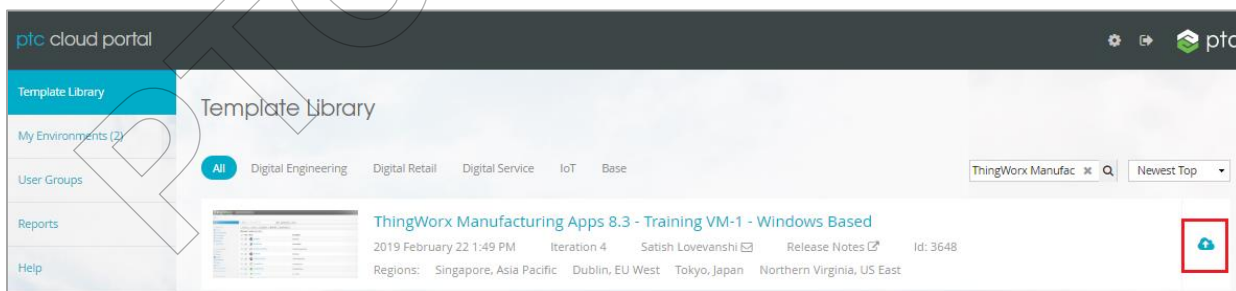
5. Ensure that the Template Library is selected in the left panel. It lists the available templates in the portal.



6. Type/paste the template name noted in step number 2 in search field as shown in below image and click search button.



7. Filtered template will be displayed. Click the **Create Environment** icon as shown in the below image.



8. Create a New Environment window as displayed in the figure. Then provide the details that are highlighted in the below image. For example:
 - o Name – For example, SCO Training
 - o Business Rationale – Training
 - o Region – Singapore, Asia Pacific (select the Region as per your geographic location)

- Time Zone
- Daily Shutdown Time
- Set **Daily Startup and Shutdown** time. Select the **Enable** checkbox to automatically start the VMs at the given time. Startup time should be 30 min before the scheduled class/training time.
- Finally, click the **Create New Environment** button.

The screenshot shows the 'Create New Environment' form with several fields and a red box highlighting the 'Options' section. Annotations include yellow arrows pointing to the 'Name', 'Purpose', 'Region', and 'Timezone' fields, and a purple arrow pointing to the 'Create New Environment' button.

Create New Environment

Name: SCO Training

Size: ☒ Medium (2CPU, 8GB RAM) ☐ Large (4CPU, 16GB RAM)

Business Rationale: Purpose: Training

Notes: Project Code, Customer Name, etc.

Region: Singapore, Asia Pacific

Start / Stop Settings: Timezone: Kolkata, UTC +05:30

Options: **Schedule:** ☒ Weekdays ☐ Everyday ☐ 24 / 5 ☐ 24 / 7
Monday-Friday, scheduled daily shutdown (optional scheduled daily startup)

Daily Shutdown: 6 PM

Daily Startup: 8 AM ☒ Enable

Estimated Cost: per hour \$0.22

Create New Environment Cancel

9. A new environment is generated and is listed under My Environments. It will take approximately 15 mins to start. Once VM starts, its turns green with status running.

The screenshot shows the 'My Environments' page with a list of environments. The 'SCO Training' environment is highlighted, and its status is 'running' (indicated by a green bar). The page includes a search bar and a 'Newest Top' dropdown.

My Environments

All Digital Engineering Digital Retail Digital Service IoT Base

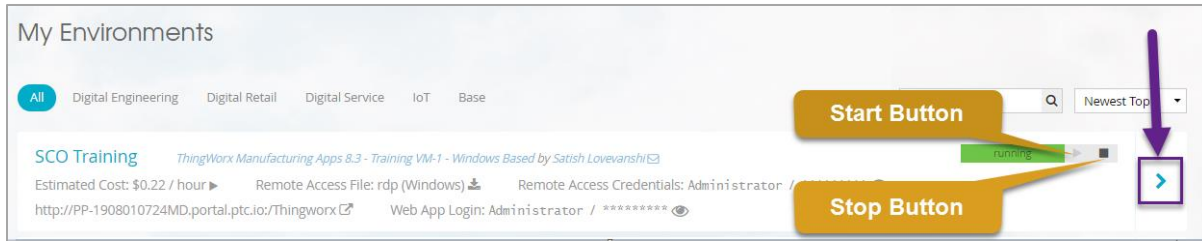
Search Newest Top

SCO Training ThingWorx Manufacturing Apps 8.3 - Training VM-1 - Windows Based by Satish Lovevanshi

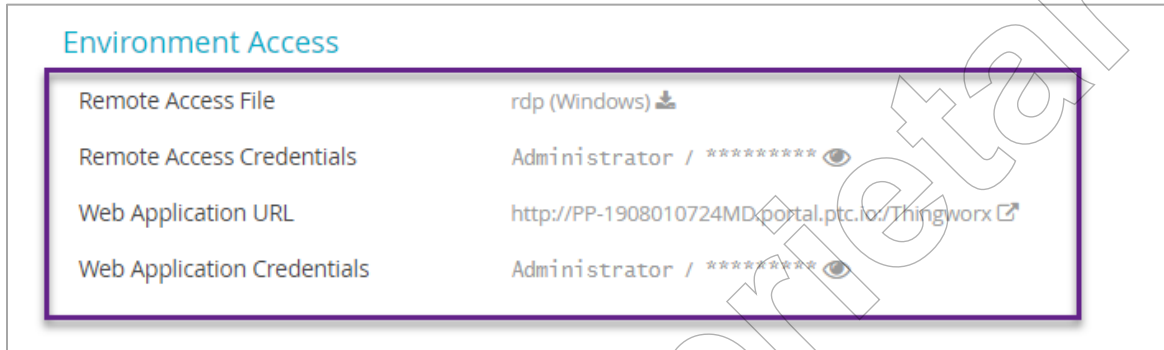
Estimated Cost: \$0.22 / hour Remote Access File: rdp (Windows) Remote Access Credentials: Administrator / ***** Web App Login: Administrator / *****

running

10. You can start and stop the VM environment using the buttons as shown in the below image.



11. Click on the arrow '>' icon as shown by arrow in the image above and refer to the **Environment Access** section in the image below.



12. The following are the details of the Environment Access:

- Remote Access File : Download the rdp file
- Remote Access Credentials : rdp credentials
- Web Application URL : Web Application URL such as ThingWorx or Windchill
- Web Application Credentials : Web Application Credentials

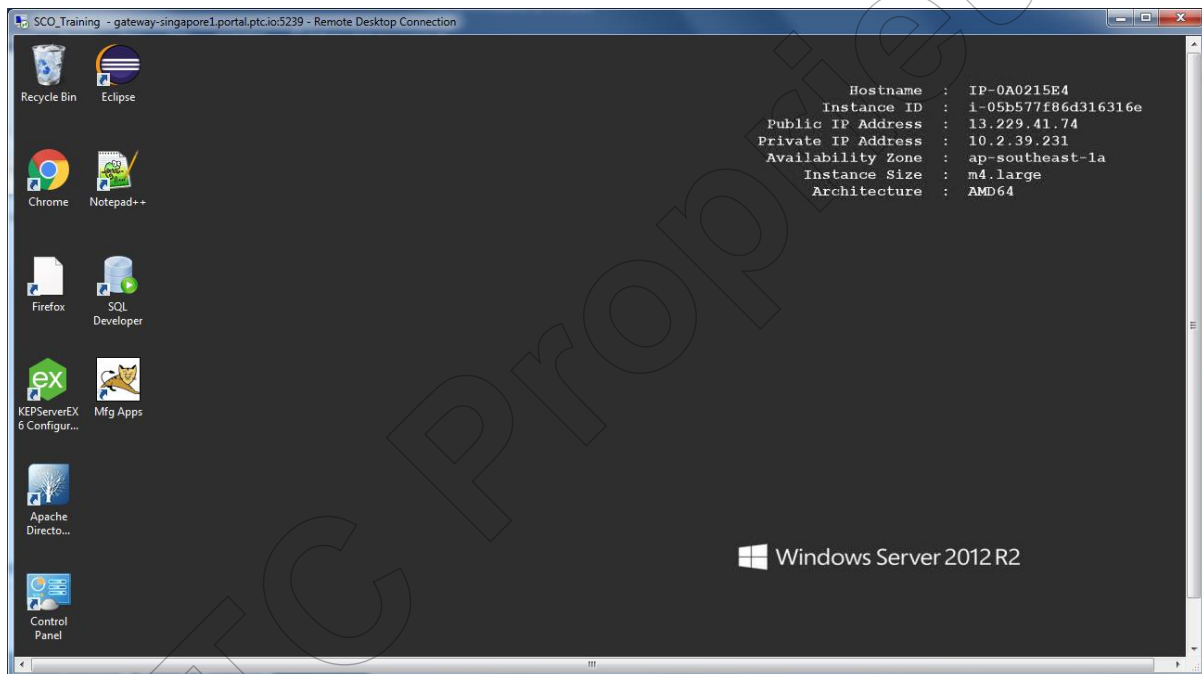
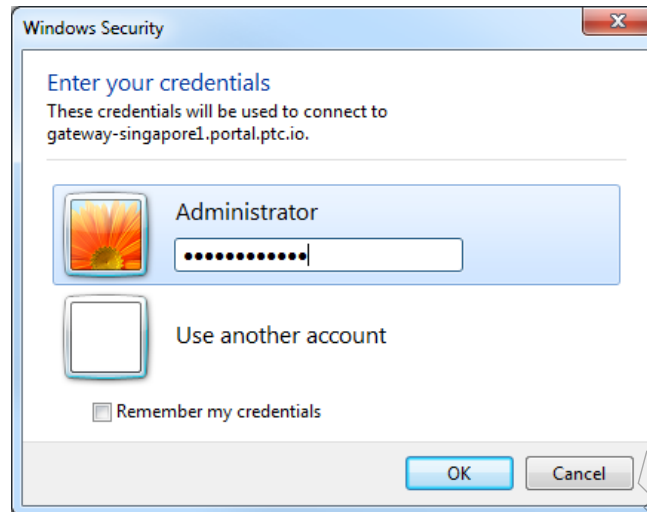
13. Download the RDP file and open it in a text editor such as Notepad ++. Edit the value of 'prompt for credentials' setting from 0 to 1. Save the file.

```
20 keyboardhook:i:2
21 negotiate security layer:i:1
22 prompt for credentials:i:1
23 redirectclipboard:i:1
24 redirectcomports:i:0
25 redirectposdevices:i:0
```

Note

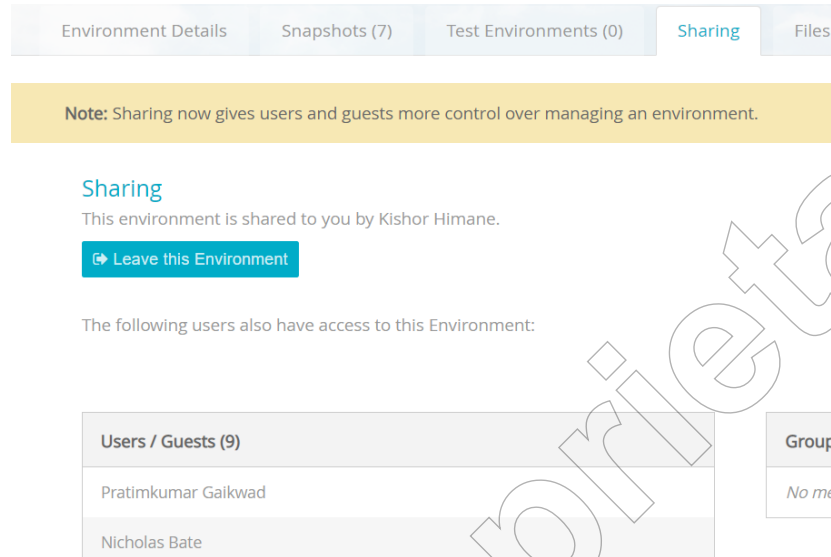
Edit the value of 'screen mode id' settings to 2 if you want monitor size screen.

14. Double-click the rdp. Copy the uniquely generated password from the Remote Access Credentials, refer step 12-b and paste in password field. Click **OK**.



15. To access the application using URL refer step 12-c and for application login details refer step 12-d.
16. You can also refer to the template release notes to get further details of different application installed on the VM including their install location, admin credentials.
17. Delete the VM instance once ALL your activity related to the course/training is complete.
- 18. Best Practices and Recommendations:**
 - Ensure you delete the VMs once ALL your activity related to the course/training delivery is complete. Any VM existing in a stopped/Archived state will incur cost and the same will be charged to your department cost center.

- For the VM size, select Small or Medium whenever possible based on the solution or the requirement. This will save on storage and processing cost.
- Use the Sharing tab to give access to your instance for any VM related issue. The shared VM will then be listed under My Environments tab on the portal of the shared user's login.



- Snapshots tab allow you to preserve the VM state. You can take the snapshot from the snapshot tab on the VM details page from the cloud portal after any major configuration change or new/update installation.

Note
Snapshots involve additional disk usage and hence additional costs. Use it cautiously and delete immediately once the required task is complete.

Note
Reach out to csportal@ptc.com for any VM accessibility issue from the portal.

Task 2: PTC Partner users – Access the VM instance on PTC Cloud Portal assigned to you by the Instructor

Important Notes:
Please refer the Exercise Guide or [VM Information](#) page for detailed steps.
Use the allocated time during the training to complete the exercises. The VMs will be deleted after the class. Extending the VM time is subject to additional approvals.
VMs are to be used for Training purpose only. Any usage outside training will incur cost to the user organization

This completes the exercise.

Module 3:

Source Control System Configuration

Module Overview

Source Control ensures you view exactly what was applied to an environment and tracks changed files over time.

Thus, you can come back anytime to a former version of the file, do automatic comparison, and check changes among others.

Module Objectives

After completing this module, you will be able to:

- Install GitBackup extension
- Create a Git repository
- Create and check a Git thing
- Create an SVN repository
- Commit changes in SVN

Exercise 1: Setting Up Source Control System Using GitHub

Objectives

After completing this module, you will be able to:



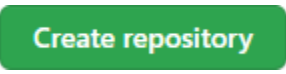
- Install GitBackup extension
- Create a Git repository
- Create and check a Git thing

Scenario

After installing ThingWorx, DevOps toolchain will be installed. The Source Control system is carried with the GitHub and GitBackup extensions. The GitHub account is a prerequisite.

In this case, we have two VMs based on the same template.

Task 1: Create a GitHub Repository.

1. Go to <https://github.com>.
2. Connect to your GitHub account.
3. Create a new repository by clicking the **New**  button or from the **+** top-right drop-down menu .
4. Enter the following details:
 - Repository name: ThingWorxDevOps
 - Description: repository for ThingWorx DevOps
 - Select radio button for Private
 - Check "Add a README file"
5. Click **Create repository** .

Note:

The newly created repository is automatically opened. Let the GitHub repository Web page open; It will be reused soon.

Task 2: Create a Git Thing on Dev Server.

1. Open your VM dedicated as the Dev server.
2. For ThingWorx URL, check the Thingworx-Credentials.txt file on the desktop.
3. Enter the credentials:
 - Username: Administrator
 - Password: portaldemo12345
4. Open the **Import/Export** menu.
5. Click **Import**.
6. In the Import Option drop-down menu, select **Extension**.
7. Click **Browse**.
8. Navigate to E:\LabFiles\TE-00042_DevOps_Practices_For_ThingWorx\Source Control System Configuration Ex1 task2 and select the GitBackup zip **Extension**.
9. Click **Open**.
10. Click **Import**. On successful import, you will get the "Import Successful" message.
11. Close the **Import** window and click **Yes** to refresh the Composer.

Note:

The GitBackup extension is now installed, so we can focus on creating the Git thing. A repository has been previously created.

12. Navigate to **Browse > Thing**, and click **+** to create a new Thing.
13. Enter the following details:
 - Name: DeploymentRepository
 - Project: DevOps
 - Base Thing Template: FileRepository
14. Click **Save**.
15. Search and open the GitBackup.Main.Mashup mashup.

Note:

There is a warning for deprecated widgets, asking to Yes/NO as per the requirement. Ignore this message or click NO.

16. Click **View Mashup**.



17. Click the top grey +.

18. For the first “Login” step, enter the following details:

- Name: GitDeployment
- For Git server, select Other
- Git username: your GitHub ID
- Git account password: your GitHub password

19. Click **Next**.

20. Go back to your GitHub repository for the following:

- Click **Code** .
- Copy the HTTPS clone. The  icon enables you to copy the URL in the clipboard.

21. Go back to the pending Git thing creation step.

22. For the second “Repo settings” step, enter the following details:

- Committer Name: free to choose yours
- Commit e-mail: your e-mail used for GitHub
- Git repo URL: paste the URL previously copied and remove the final “.git”
- File repository: remove the standard one and select DeploymentRepository
- File repository path: /pipelineJob
 - Don’t miss the “/” sign. The name is coherent with the future exercises.
- Initial branch: let the default **master** value
- Click **Add**.

23. Observe the Branch/Commit drop-down menu is empty.

24. Click **Pull**.

25. Observe the Branch/Commit drop-down menu now contains a main branch.

26. Open the **Advanced** tab.

27. Observe the folder structure:

- It only contains a pipelineJob top folder and a .git subfolder.

28. Click **Export**.

29. For the Export entities field, select the **DevOps** project.

30. Click **Export** (on the same row).

31. Close the **Export** pop-up menu.
32. Observe a new DevOps subfolder has been created.
33. Select this folder and click **delete** (garbage icon).

Task 3: Create a Git Thing on Test Server.

Note:

We do the same work on the Test server to prepare the CI/CD exercises.

1. Open your VM dedicated as the Test server.
2. Open a Web browser and connect to ThingWorx.
3. Enter the credentials:
 - Username: Administrator
 - Password: portaldemo12345
4. Redo all steps from 4 to 32 from the previous task.

Note:

Another way to do that is to unitary export (as XML) the DeploymentRepository and GitDeployment things from the Dev server, copy the files, and then open the Test server VM, and paste the files. Before importing them, you will need to update the GitDeployment XML file. Open it in Notepad and remove the encrypted password. If you don't do that, the import will fail because the password format is not recognizable. You can now import the two files. Edit the configuration of the GitDeployment thing to enter the password. The Universal export option overcomes this password trouble by writing the file in plain text. This operation is a security risk.

Completion Criteria

At the end of this exercise, your Dev and Test servers both have a working Git thing connected to the same GitHub repository.

This completes the exercise

Exercise 2: Setting Up Source Control System Using Tortoise SVN

Objectives

After completing this module, you will be able to:

- Create an SVN repository
- Commit changes in SVN

Scenario

After installing ThingWorx, Tortoise SVN can be configured to Source Control ThingWorx entities.

Task 1: Initialize SVN Repository.

1. Open your VM dedicated as the Dev server.
2. Navigate to C: root.
3. Create a new folder and name it SVNRepository.
4. Right-click this new folder.
5. In the contextual menu, open the **TortoiseSVN** submenu.
6. Click **Create repository here**.
7. In the repository created, click the Create folder structure.
8. Click **OK** to close the Create Repository window.
9. Click **Start Repobrowser**.
10. On the left panel, select the trunk folder.
11. Check that the URL is now file:///C:/SVNRepository/trunk (if not, you can type it).
12. Click **OK** to close the Start Repobrowser window.
13. Click **OK** to close the Repository created window.
14. Navigate to the C:\ThingworxStorage\repository.
15. Create a new folder named VersionControlExport.

Note:

The name of this folder must match the name of a FileRepository thing in ThingWorx. This thing will be created later.

16. Right-click this new folder.
17. In the contextual menu, select **SVN Checkout...**
18. In the Checkout window, browse the URL of the repository.
19. In the Repository Browser window, select the trunk in the left folder structure. The URL is now file:///C:/SVNRepository/trunk.
20. Click **OK**.
21. In the Checkout window, notice that the Checkout directory is the newly created folder C:\ThingworxStorage\repository\VersionControlExport.
22. Click **OK**.
23. In the Checkout Finished windows, click **OK**.

Note:

The folder is now at version one with an  icon.

24. Open **ThingWorx**.
25. Create a new Thing with the following details:
 - Name: VersionControlExport
 - Template: FileRepository

Task 2: Commit Changes in SVN.

Note:

We assume that creations and updates are already done. We are going to commit in SVN.

1. Open the **Import/Export** menu.
2. Click **Export**.
3. In the Export Option drop-down list, select **Source Control Entities**.
4. Select the **DevOps** project.
5. Remove the default repository.
6. Select the **VersionControlExport** repository.
7. Click **Export**.

Note:

Other filters may be applied for the export. The Update version tag or the start/end date of modified entities are two good examples.

8. Navigate to the ThingWorx directory.
9. Navigate to the \ThingworxStorage\ repository.
10. Observe the content of VersionControlExport.
11. Right-click the VersionControlExport.
12. In the contextual menu, open the TortoiseSVN submenu.
13. Click **Add....**
14. In the Add – Tortoise SVN window, select the **Select/deselect all** check box.
15. Click **OK**.
16. In the Add Finished window, click **OK**.

Note:

The folders and files in the VersionControlExport folder now have the  and  icons.

17. Right-click the VersionControlExport folder.
18. In the contextual menu, select **SVN Commit....**
19. In the Commit – TortoiseSVN window, enter the message **First commit**.
20. In the **Changes made** table, check that all paths are selected.
21. Click **OK**.
22. In the Commit Finished window, click **OK**.

Note:

The folders and files in the VersionControlExport folder now have the  and  icons.

Completion Criteria

At the end of this exercise, SVN repository is configured to source control entities exported from ThingWorx.

Note:

During the exercise, the Source Control Entities export was used. The To file export or unitary file export would also work. It requires to move the exported file to the VersionControlExport folder.

This completes the exercise

Module 4: CI/CD Configuration

Module Overview

CI/CD refers to Continuous Integration Continuous Deployment (also called Delivery). It is a set of practices for the developers to continuously integrate and deploy their updates. We will practice a simple example to show how CI/CD can be put in place including automated tests.

Module Objectives

After completing this module, you will be able to:

- Create, configure, and test a simple pipeline in Jenkins
- Identify and create accurate tests
- Enhance the pipeline with tests
- Check the pipeline with automated tests

Exercise 1: Creating a Simple Pipeline Using Jenkins

Objectives

After completing this module, you will be able to:

- Create and configure a simple pipeline in Jenkins
- Test the simple pipeline

Scenario

Git things are well configured in both Dev and Test servers. Now the deployment will be automated with a Jenkins pipeline. First, the deployment will be in two stages, a Package application stage and a Deploy application stage.

Task 1: Create a Thing to Manage Deployment.

1. Open your VM dedicated as the Dev server.
2. Open a Web browser and connect to ThingWorx.
3. If needed, enter the credentials:
 - Username: Administrator
 - Password: portaldemo12345
4. Create a new thing.
5. Enter the following details:
 - Name: PipelineManagerThing
 - Project: DevOps
 - Template: GenericThing
6. Save it.
7. Select the **Services** tab.
8. Create a new service.
9. Enter the following details:
 - Name: ExportSimulationApp
 - Output: String
10. Navigate to E:\LabFiles\TE-00042_DevOps_Practices_For_ThingWorx\CICD Configuration Ex1 task2, and copy the content of Script ExportSimulationApp.txt.

11. Paste it in the script handler.

12. Observe the script.

Note:

The script is simple and only exports entities of a project named Simulator.

13. Save the service. Click **Done**.

14. Create a new service.

15. Enter the following details:

- Name: ImportSimulationApp
- Output: String

16. From the LabFiles, copy the content of Script ImportSimulationApp.txt.

17. Paste it in the script handler.

18. Observe the script.

Note:

The script is simple and only imports entities of a project named Simulator.

19. Save the service.

20. Import the Simulator entities. Use the ThingWorx entity import and the file from the folder E:\LabFiles\TE-00042_DevOps_Practices_For_ThingWorx\CICD Configuration Ex1 task2.

21. Execute the ExportSimulationApp service and observe the following consequence:

- Result: Success
- In GitBackup.Main.Mashup > View Mashup> Advanced tab, the directory structure now has a Simulator subfolder with a Source Control export type.
- In GitHub, the repository now has a master branch; select it and observe inside all folders like in the Thingworx repository.

22. From the GitBackup.Main.Mashup, delete the Simulator folder.

23. From the Composer, delete all entities that belong to the Simulator project.

Note:

To quickly delete entities belonging to a project, you can browse to all entities, then filter on the project, and then select all entities and delete. You will get error messages for each entity that can't be deleted due to dependency. Repeat the select all entities and delete actions to delete all entities from this project.

BECAREFUL: this way of deleting all entities is at risk, and you must take care of the filter applied to the list.

24. Execute the ImportSimulationApp service and observe the following consequence:

- Result: Success
- In GitBackup.Main.Mashup, the directory structure now has a Simulator subfolder with an export to Source Control (coming from GitHub repository)
- In the Composer, you can notice that you once again have the Simulator entities.

25. In the PipelineManagerThing thing, open the More drop-down menu and select **Export**.

26. Copy the exported PipelineManagerThing XML file.

27. Open your VM dedicated as the Test server.

28. Paste the PipelineManagerThing XML file on the desktop.

29. Import the PipelineManagerThing XML file.

Task 2: Create the Pipeline to Automate the Deployment.

1. Open your VM dedicated as the Dev server.
2. Create a text file to gather the following information:
 - URL address of the Dev server
 - URL address of the Test server
 - Key id of the DevOps appKey from the Dev server
 - The Test server also has this appKey, and you can check that the key id is the same. That's why we are not going to distinguish Dev and Test appKey.
3. Open a Web browser and connect to Jenkins. Find Jenkins URL in the Thingworx-Credentials.txt file placed on the desktop.
4. If needed, enter the credentials:
 - Username: Administrator
 - Password: Jenkins@12345
5. Click the **New Item** and enter the following details:

- Name: DeployFromDevToTest
 - Type: pipeline
6. Click **OK**.
 7. Observe that after creation, you are redirected to the configuration page of the pipeline.
 8. Scroll down to view the pipeline script frame.
 9. From the LabFiles, copy the content of Pipeline1-BasicImportExport.txt.
 10. Paste it into the pipeline script.
 11. At the beginning of the script, adapt the value for the DEV, TEST, and APPKEY environment variables.
 12. Take time to review the content of the script.

Note:

The script is in the Groovy language. Thanks to plugins added into Jenkins, the script can use the REST API request. It is a convenient way to easily understand the script, what does it do, and make it easy to adapt.

13. At the end of the page, click **Save**.
14. Observe that you are redirected to the page of the pipeline.
15. Click **Build Now**.
16. Observe the pipeline stages are running. Observe the result. The two stages should be green. If the red frame appears, browse for error.

Completion Criteria

At the end of this exercise, you succeed to create a simple pipeline to automatically deploy the Simulator application from the Dev server to the Test server.

This completes the exercise

Exercise 2: Enhanced Pipeline With Automated Tests

Objectives

After completing this module, you will be able to:

- Create services for tests in ThingWorx
- Update the pipeline to integrate automated tests

Scenario

After a first pipeline approach with two simple stages for deployment, the script will be updated with the capability to trigger automatic tests.

Task 1: Create a Thing to Manage Deployment Tests.

1. Open your VM dedicated as the Test server.
2. Open a Web browser and connect to ThingWorx.
3. If needed, enter the credentials:
 - Username: Administrator
 - Password: portaldemo12345
4. Open the **Import/Export** menu.
5. Click **Import**.
6. Click **Browse**.
7. Navigate to E:\LabFiles\TE-00042_DevOps_Practices_For_ThingWorx\CICD Configuration Ex2 task1 and select DataShapes_TestResultData.xml.
8. Click **Open**.
9. Click **Import**.

Note:

This datashape defined the field of the table that we will display for the test results.

10. Create a new thing.
11. Enter the following details:
 - Name: DeploymentTest
 - Project: DevOps
 - Template: GenericThing

12. Select the **Properties and Alerts** tab.

13. Add a property with the following data:

- Name: LastTestResults
- Base type: infotable, datashape: TestResultData

14. Add the property.

15. Select the **Services** tab.

16. Create a new service.

17. Enter the following details:

- Name: Test1
- Output: Infotable, datashape: TestResultData

18. From the LabFiles, copy the content of Script Test1.txt.

19. Paste it in the script handler.

20. Observe the script.

Note:

This service simulates a service to test a user service. It generates a random number between 0 to 1. The test is considered as fail if the value is below 0.15.

21. Save the service.

22. Create a new service.

23. Enter the following details:

- Name: RunAllTests
- Output: String

24. From the LabFiles, copy the content of Script RunAllTests.txt.

25. Paste it in the script handler.

26. Observe the script.

Note:

The RunAllTests service is a focal point to trigger all test services. In this case, it triggers many times the Test1 service and stores the stacked results in the LastTestResults property.

27. Save the service.

28. Create a new service.

29. Enter the following details:

- Name: FormatResults
- Output: html

30. From the LabFiles, copy the content of Script FormatResults.txt.

31. Paste it in the script handler.

32. Observe the script.

Note:

This service formats the content of the LastTestResults property to be displayed as an html page.

33. Save the service.

34. Create a new service.

35. Enter the following details:

- Name: RunAllTestsFormatResult
- Output: html

36. From the LabFiles, copy the content of Script RunAllTestsFormatResult.txt.

37. Paste it in the script handler.

38. Observe the script.

Note:

This service triggers all tests and the html result in a row.

39. Save the service.

40. Execute RunAllTestsFormatResult.

41. Observe the result. You can execute it a few times to check randomness of Test1.

Task 2: Update the Pipeline with Automated Tests.

1. Open your VM dedicated as the Dev server.
2. Open a Web browser and connect to Jenkins.
3. If needed, enter the credentials:
 - Username: Administrator
 - Password: Jenkins@12345
4. Click the DeployFromDevToTest pipeline.
5. Click **Configure**.
6. Scroll down to view the pipeline script frame.

7. From the LabFiles, copy the content of Pipeline2-WithTests.txt.
8. Paste it into the pipeline script.
9. At the beginning of the script, adapt the value for the DEV, TEST, TEST_THING, and APPKEY environment variables. (TEST_RESULT_PATH remains unchanged)
10. Take time to review the content of the script.

Note:

It is based on the first pipeline. In addition, we have two new environment variables and a new ExecuteTests stage.

11. At the end of the page, click **Save**.
12. Notice that you are redirected to the page of the pipeline.
13. Click **Build Now**.
14. Notice the pipeline stages are running and observe the result. For the first time, the last stage should be in red due to a permission issue.
15. On the left side, in the Build History, cursor over the last number next to the red dot.
16. Select the drop-down menu icon ▼.
17. Select **Console Output**.
18. Observe the last stage. An error occurred due to script permission issues.
19. Click the **Administrators can decide whether to approve or reject this signature** link.
20. Click **Approve** for the three permissions.
21. Go back to the pipeline.
22. Click **Build Now**.
23. Observe the three stages are successful and in green.
24. After successful results, press F5 to refresh the pipeline page.
25. Click the new link Result ThingWorx tests.
26. Observe the HTML result.

Completion Criteria

At the end of this exercise, you succeed to create deployment tests and automatically trigger them within the pipeline.

Note:

In the LabFiles, you have an Example folder with entities of a more complete export/import script, more complete tests, and a Pipeline script including a Docker part to deploy and test automatically on a fresh new image.

This completes the exercise