Hands-On Lab: NAV on Docker

# Who should complete this HOL?

This Hands-On Lab is designed to help you understand what Docker is and what NAV on Docker can do for you. After completing the HOL, you should be able to determine if Docker and especially NAV on Docker is useful in your organization. The HOL will use the Workshop VMs as a foundation for the HOL to have a uniform platform for all.

When you have completed this HOL, you can find more info on the nav-docker project on github: <http://www.github.com/microsoft/nav-docker>. This is also the place you should be filing issues and comments.

# What is Docker?

If you are new to Docker and Containers, you might want to scan through this document before heading into the workshop:

<https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/>

This should give you a better understanding of what Docker is.

For the remaining of the workshop, you will be going through some scenarios, using Docker, on how to Deploy NAV.

When you connect to your learning environment, you are presented with a website, which looks like the image on the right side. What you might not be aware is, that when you are viewing this, you are already using Docker. This website is hosted inside a Container, which is running inside an Azure Virtual Machine, called the Docker host and in this case, your Workshop VM.

Clicking the Connect link will download the .rdp file, which launches Remote Desktop to the Workshop VM.

Note that all other workshops at Directions are using the same Workshop VM, but they just don’t care that NAV is running in a Container.

Note also that the Workshop VMs are very much like the NAV Developer Preview VMs with the September Update you can get from <http://aka.ms/navdeveloperpreview>.

Note also that you will not and cannot connect to a remote desktop in the Container. The Container is based on WindowsServerCore, which has no UI, no desktop.

Connecting to the Workshop VM (the Docker host) will allow you to interact with the Docker Containers that are available on that machine by using various commands.

First thing we will do is to have a look at the Workshop VM desktop and what we can do with that.

|  |  |
| --- | --- |
| Open your workshop landing page in a browser and press the Connect button to connect to the remote desktop of your workshop environment. Use the provided credentials to login.  Close Server manager and other windows that pop up and you should have a desktop, with a few shortcuts. |  |
| **Command Prompt**  The Command Prompt is the standard CMD.EXE running as Administrator.  This prompt is primarily there for running Docker commands or other executables.  We will be using the Command Prompt throughout this Hands On Lab. |  |
| **Nav Container Helper**  The navcontainerhelper is a set of functions, which will help you working with Nav Containers.  When you start the container helper, it will display a number of the available functions.  **Note** that the navcontainerhelper is an open source project from <http://www.github.com/microsoft/navcontainerhelper> and any issues regarding the navcontainerhelper should be added under issues in the github repo.  We will dive into the container helper later. |  |
| **navserver Command Prompt**  The navserver Command Prompt is the standard CMD.EXE running inside the navserver container.  When you run dir inside the navserver Command Prompt you will see the Container file system. Folders that are shared from the host to the container are shown as symbolic directory links (SYMLINKD).  The file system inside the NAV Docker Image consists of a few special folders/files:  **c:\run** the run folder is the folder containing all the scripts, which are used to set up NAV in the container.  **c:\run\my** is the location, where you can place scripts which can override functionality of the run folder. Typical scenario is to share a folder from the host to the c:\run\my folder, containing various scripts that you want executed during start.  **c:\run\start.ps1** is the entry point for the container.  **c:\run\navstart.ps1** is the main script for setting up NAV and launching other setup scripts. |  |
| **navserver PowerShell Prompt**  The navserver PowerShell Prompt is a PowerShell prompt running inside the container.  All NAV cmdlets are loaded inside the PowerShell prompt, ready to use.  Example:  **Get-NavServerUser NAV**  Will list all users in the NAV server instance (which is the default server instance in the container).  Note that **not** all commands will work inside the container. You cannot create a new server instance, for example – that is done by spinning up another container (the Docker way😊) |  |
| **navserver CSIDE**  C/SIDE a.k.a. the Classic Development Environment for the navserver container.  Note that C/SIDE is not there to support all classic development scenarios.  The primary reason for C/SIDE to be available is for the VS Code developer to be able to see and browse through the source of the base application.  Having said that, you can do the majority of classic development scenarios in C/SIDE.  Note that when you start C/SIDE you will be running Database Authentication and you have to login as **SA** and use the Workshop VM password.  Server name is **navserver\SQLEXPRESS** and the database name depends on which localization you are running. |  |
| **navserver Web Client**  Opens a browser with the Web client for the navserver container. The Web client is installed inside the container on IIS and the ports are exposed on the container and published to the host.  Login user name is **student** and the Workshop VM password is your password. |  |
| **navserver Windows Client**  Opens the Windows client for the navservercontainer.  The Windows client is not installed on the Docker host even though it looks like it.  The Docker host shares a folder to the container called C:\Program Files (x86)\Microsoft Dynamics NAV – and the container then copies the files from that folder to the host.  This gives the best compatibility and allows the folder to be overridden if deploying a new container.  Login user name is **student** and the Workshop VM password is your password.  You can also install the Windows client using ClickOnce. There is a section about this later in the HOL. |  |
| **Landing Page**  The landing page was the starting point of your journey. You will find all info and links here necessary to connect and use the Workshop VM. | |
| **PowerShell ISE**  PowerShell ISE running on the Docker host. This is every IT infrastructure gurus favorite tool and we will be using ISE throughout this Hands On Lab. | |
| **Visual Studio Code**  Visual Studio Code is used for AL development and is not used in this Hands On Lab.  When launching the Workshop VM, the AL Language extension from the landing page is preinstalled. If you deploy a new NAV Container, you will have to uninstall and install a new AL Language extension. | |

# Basic Docker commands

Let’s drill into some of the basic Docker commands to get a better understanding of what Docker is and how it works.

You can run these commands in PowerShell, but Docker is a simple Windows Executable and will run in a command prompt as well.

For simplicity reasons, we will use the Command Prompt.

|  |  |
| --- | --- |
| Open the Command Prompt and write:  **docker ps**  This gives you a list of all the running Docker Containers on your machine.  Take some time to Inspect the info:   * The container name is navserver. * The container ID starts with 755757f82076. * The container is based on the *microsoft/dynamics-nav:devpreview-finus* image. * Ports 443 and 7046-7049 are all exposed on the Docker host, meaning you can access them from the outside. * Port 8080 from the Docker container is published as port 80 on the Docker host. * Ports 80, 1433 and 7045 are open for the host. |  |
| You might wonder why the previous section says: *“The container ID starts with…”*. The reason for this is, that the ID really is a 64 digit globally unique hex identifier, but most time you can refer to the ID by specifying the first digits until your specification isn’t ambiguous.  You will get the full ID by typing:  **docker ps --no-trunc**  but if you only have one image you can identify it by writing the first digit – here: **7** |  |
| The next command to try is:  **docker images**  This gives you a list of all images available for you to run.  In this picture there are the 2 Microsoft base images: Windows Server Core and Nano Server. Beside them, the latest devpreview image with Financials US localization.  A Docker image is really a set of services installed in a box (container) ready to run on demand.  A specific version of the NAV Docker image is a specific version (incl. localization) of NAV installed in a Container ready to run (ex. NAV 2017 CU7 DK).  The NAV Docker images are highly configurable and customizable. |  |
| Now, try to run another instance of the dynamics-nav image you have available:  **docker run microsoft/dynamics-nav:devpreview-finus**  As the error indicates, you will have to accept the End User License Agreement before this container can start.  Use:  **docker inspect --format='{{.Config.Labels.legal}}' microsoft/dynamics-nav:devpreview-finus**  to view the legal documents for the preview. |  |
| Let’s run another instance of the image and accept the EULA:  **docker run -e accept\_eula=Y microsoft/dynamics-nav:devpreview-finus**  Press Ctrl+C in the command prompt to exit the container and leave it running in the background.  Now, run:  **docker ps**  The command will show you two containers running. Inspect the difference in names, ports etc.  Note that Docker automatically assigns a readable name to the container if you don’t assign a name in the Docker run statement.  The original container will have ports exposed on the host, the new container will only have ports exposed on the container. |  |
| Open the Web client in a browser. Ignore the certificate warnings for the self-signed certificate. |  |
| Now try to run  **docker ps -a**  which will show you all containers – running ones and exited ones. If you did try to run a container earlier without specifying the accept\_eula=Y then you will have an exited container in the list.  Remove the dead container using  **docker rm <containerid>**  If you want to remove a running container you either need to stop it first or use the -f parameter:  **docker rm <contianerid> -f** |  |
| Use  **docker inspect navserver**  to inspect settings, status, labels etc. on a container or an image.  You will also find network settings etc. if you look through the emitted JSON. |  |
| Use  **docker stats**  to get statistics from the currently running containers. |  |
| If you dislike the format of Docker stats (if you would like the container name included) you can modify the output by specifying a statsFormat property in the c:\users\student\.docker\config.json file.  In VSCode, create a new file with this content:  {  "statsFormat": "table {{.Name}}\t{{.CPUPerc}}\t{{.MemUsage}}\t{{.NetIO}}\t{{.BlockIO}}"  }  And save it in c:\users\student\.docker\config.json.  Now re-run  **docker stats**  and you will see the info requested.  You can also add a section for psFormat etc. |  |

# Use PowerShell ISE to modify files in the container

|  |  |
| --- | --- |
| If you are new to Docker you might not yet be annoyed over how cumbersome it is to modify files in the container.  You can connect using the PowerShell prompt or the command prompt, but since the file system is remote and you don’t have a UI, you cannot edit files using Notepad.  But…  You can use ISE – it just requires a small trick.  Open ISE and run  **Enter-PSSession -ContainerId (docker ps --no-trunc -qf "name=navserver")**  Now you will enter a remote session in PowerShell (much like the navserver PowerShell Prompt) and inside of this you can use psEdit to edit files remotely without having to share folders and copy back and forth.  PS. The navcontainerhelper introduces a function which is called Enter-NavContainer <containername> which does exactly this.  Note that psEdit is an ISE specific function and does NOT work inside the navserver PowerShell Prompt. |  |

# Advanced parameters

When using Docker run with the NAV image, there are a lot of different parameters you can use. All NAV image specific parameters are specified as environment variables (-e or –env).

|  |  |
| --- | --- |
| There are a number of different parameters you can set when running the NAV Container. This command uses some of them:  **docker run -e accept\_eula=Y -e usessl=N -e auth=Windows -e username=student -e password=<password> --name test microsoft/dynamics-nav:devpreview-finus**  If you specify the password of your student user, then this command will start NAV in a Container without SSL and using Windows Authentication.  Note that this is a known hack, that you can use Windows Authentication between two machines if they share the same username and password.  **Note** that the Web client is now without SSL and if you open it in a browser, you will find that you are logged directly into NAV. Use:  **docker rm test -f**  to remove the container named test. |  |
| In the above example, test is the container name.  Most Docker commands take container ID or container name as parameter.  The container name however is not added to the DNS resolver and you cannot ping the container name.  In order to access the container using TCP or HTTP you need to use the hostname.  The default hostname is the first 10 characters of the container ID.  You can specify your own hostname using:  **--hostname test**  Docker will automatically maintain the IP address in the DNS resolution for the hostname, locally on the host.  You can also specify a public DNS name, which is the CNAME record, which points to your host if you are exposing the container to the world using a trusted certificate. PublicDnsName will default to the hostname.  **-e publicDnsName=s11.navdemo.net**  If you do not use SSL, the publicDnsName is only used for calculating properties like PublicWebBaseUrl, PublicSoapBaseUrl etc. in the config file. |  |

# Pulling new NAV on Docker Images

Up until now, you have only used the images that already exist on the VM. If you want to use non-existing images or if you want to get an update to an existing image, you will have to pull these images. If you pull dynamics-nav:2017 without any specific cu you will always get the latest cu. Pulling dynamics-nav:2017-cu8 will only be updated if there are updates to the Windows Server Core or the NAV Generic Image.

|  |  |
| --- | --- |
| All shipped versions of NAV since NAV 2016RTM are available on the public docker hub, where you also can find the EULA and the supported tags.  <https://hub.docker.com/r/microsoft/dynamics-nav/>  under Tags, you will find a list of all the tags in the public repository.  Docker images are constructed in layers. That means a Docker pull will only need to download those layers that are different from already downloaded layers. Try:  **docker pull microsoft/dynamics-nav:devpreview-findk**  You will see, that the first ~18 layers already exists and only a few layers will need to be downloaded.  The ~5 layers, which are downloaded is the difference between the US localization and the DK localization, so only this difference will have to be downloaded. |  |
| As you might have guessed by now – if US and DK includes W1 (are built on top of W1), pulling W1 should not cause any downloads. Try:  **docker pull microsoft/dynamics-nav:devpreview**  Indeed – nothing to download, all layers already exist. |  |

# Using the navcontainerhelper

The navcontainerhelper is already installed on the workshop VM, but you can easily install it on your local box from the PowerShell Gallery using:

**Install-module navcontainerhelper -force**

|  |  |
| --- | --- |
| Even though Docker is a command line executable, you can use it in PowerShell like other executables and it does have some advantages. For that, we have created the navcontainerhelper, an open source project which is supposed to make it easier to work with containers.  Create a folder called C:\TEST. Start PowerShell ISE, create a new script and p aste in this line:  **New-NavContainer -accept\_eula -containerName myserver -includeCSide**  save it as C:\TEST\start.ps1 and run it.  By default, the New-NavContainer will create a container running Windows Authentication, using the same image as the navserver container. Please supply the Windows Credentials of your workshop VM. |  |
| You should see an output, which is like the output on the right here.  Note that first time you run a specific version and include CSide, the container will automatically export all objects as text (baseline for object handling functions). You can avoid this by adding -doNotExportObjectsToText  You can press F5 again and again and the script will automatically remove the old container and start a fresh, this time without exporting objects.  Note that you will have a new set of shortcuts on the desktop to connect to your myserver container. | PS C:\demo> New-NavContainer -accept\_eula -containerName myserver -includeCSide  Creating Nav container myserver  Using image microsoft/dynamics-nav:devpreview  Using license file C:\DEMO\license.flf  NAV Version: 11.0.19097.0-W1  Generic Tag: 0.0.3.3  Creating container myserver from image microsoft/dynamics-nav:devpreview  Waiting for container myserver to be ready, this shouldn't take more than a few minutes  Time: ½ 1 ½ 2  .........................................................Ready  Create Desktop Shortcuts for myserver  Welcome to the NAV Container PowerShell prompt  Export Objects to C:\DEMO\Extensions\Original-11.0.19097.0-W1.txt  Split C:\DEMO\Extensions\Original-11.0.19097.0-W1.txt to C:\DEMO\Extensions\Original-11.0.19097.0-W1  Export Objects (new syntax) to C:\DEMO\Extensions\Original-11.0.19097.0-W1-newsyntax.txt  Split C:\DEMO\Extensions\Original-11.0.19097.0-W1-newsyntax.txt to C:\DEMO\Extensions\Original-11.0.19097.0-W1-newsyntax  Nav container myserver successfully created |
| If you want to see the actual output of the container, write:  **docker logs myserver** | Initializing...  Hostname is myserver  PublicDnsName is myserver  Running Specific Image  Using Windows Authentication  Starting Local SQL Server  Starting Internet Information Server  Using Database Connection localhost/SQLEXPRESS [CRONUS]  Modifying NAV Service Tier Config File for Docker  Modifying NAV Service Tier Config File with Instance Specific Settings  Start NAV Service Tier  Using license file 'c:\run\my\license.flf'  Import NAV License  Creating DotNetCore NAV Web Server Instance  Creating http download site  Creating Windows user  Enabling SA  Creating NAV user  (6869 rows affected)  Container IP Address: 172.18.236.245  Container Hostname : myserver  Container Dns Name : myserver  Web Client : http://myserver/NAV/WebClient/  Dev. Server : http://myserver  Dev. ServerInstance : NAV  Files:  http://myserver:8080/al-0.11.14434.vsix  Ready for connections! |
| You can use  **help new-navcontainer**  To list all parameters available in the new-navcontainer function. | PS C:\demo> help New-NavContainer  NAME  New-NavContainer    SYNOPSIS  Create or refresh a Nav container      SYNTAX  New-NavContainer [-accept\_eula] [-accept\_outdated] [-containerName] <String> [[-imageName] <String>] [[-licenseFile] <String>]  [[-Credential] <PSCredential>] [[-memoryLimit] <String>] [-updateHosts] [-useSSL] [-includeCSide] [-doNotExportObjectsToText] [[-auth]  <String>] [[-additionalParameters] <String[]>] [[-myScripts] <String[]>] [<CommonParameters>]      DESCRIPTION  Creates a new Nav container based on a Nav Docker Image  Adds shortcut on the desktop for Web Client and Container PowerShell prompt |
| If you want to spin up a new container with NAV 2017, you can write:  **New-NavContainer -accept\_eula -containerName nav2017 -imageName microsoft/dynamics-nav:2017 -includeCSide -doNotExportObjectsToText**  You will see, that the function automatically pulls the NAV 2017 CU12 W1 image and it will take some time to complete the pull as most of the layers are changed.  The Generic Tag here is 0.0.3.2 (previous was 0.0.3.3)  Again you will find shortcuts on the desktop to connect to your nav 2017 container. | ...  b3eeb1f92259: Pull complete  95ca09a479f5: Pull complete  77c622d9410e: Pull complete  550c091a6a4b: Pull complete  c21730148ab6: Pull complete  25eb60f6a3a2: Pull complete  0b1c5b4248fd: Pull complete  5f0ae6248073: Pull complete  e36f63306277: Pull complete  b4ee766d2c06: Pull complete  bd446c382e13: Pull complete  f8fd5c75fab4: Pull complete  Digest: sha256:744004a466b6c3550d23c7794f562b4d9b049c0c07cbbe7fa43867aac8acf47f  Status: Downloaded newer image for microsoft/dynamics-nav:2017  Creating Nav container nav2017  Using image microsoft/dynamics-nav:2017  Using license file C:\DEMO\license.flf  NAV Version: 10.0.18976.0-W1  Generic Tag: 0.0.3.2  Creating container nav2017 from image microsoft/dynamics-nav:2017  Waiting for container nav2017 to be ready, this shouldn't take more than a few minutes  Time: ½ 1 ½ 2  .............................................Ready  Create Desktop Shortcuts for nav2017  Nav container nav2017 successfully created |
| **Remove-NavContainer nav2017**  Will clean up after your Nav 2017 container | PS C:\demo> Remove-NavContainer nav2017  Removing container nav2017  Removing Desktop Shortcuts for container nav2017  Successfully removed container nav2017 |

# Using a different database server

Up until now, we have been using NAV in a Container with the database living inside the same container. That is convenient when doing demos, but frequently you probably want to run the database on a different SQL Server or maybe even on Azure SQL. It is very easy to point out a different database server, instance, and name on the command line and if your containers are set up with gMSA (Group Managed Service Accounts) and Windows Authentication this should be sufficient to connect.

**BUT…**

The Workshop VMs do NOT use gMSA (as this would require the setup of AD servers), meaning that connecting to a different database server requires a little more work. What you will do here is to override the SetupDatabase script. Overriding scripts is an important feature of the NAV on Docker Images, and it really provides the flexibility needed for setting up a system as complex as NAV.

In the end we’ll have our new container ”myserver” running NAV and we’ll just reuse the first container ”navserver” as a database server. We could also use a new container containing only SQL Server, but we’ll be faster this way.

|  |  |
| --- | --- |
| Copy this script, paste it into PowerShell ISE, modify the “<Password>” with the password for your Workshop VM, and save the script as c:\test\SetupDatabase.ps1.  Inside the container, the primary setup runner (**c:\run\navstart.ps1**), will call several scripts in the c:\run folder to setup variables, database, license etc.  For each of these scripts, it will check whether there is a script in **c:\run\my** which overrides the behavior. The overriding script can determine if it is necessary to call the default behavior, but it needs to take over the responsibility of the script.  Specifying scripts in the myscripts parameter for new-navcontainer will automatically copy these scripts to the c:\run\my folder inside the container.  The NAV on Docker documentation talks about all the different scripts you can override and typical scenarios for why you would want to do it. | if (!$RestartingInstance) {    Write-Host "Change Database Connection"  $DatabaseServer = "navserver"  $DatabaseInstance = "SQLEXPRESS"  $DatabaseName = "CRONUS"  $DatabaseUserName = "sa"  $DatabasePassword = "<Password>"  $EncryptionPassword = "1234abcd!1234abcd"  $TrustSQLServerCertificate = $true  $DatabaseSecurePassword = ConvertTo-SecureString -String $DatabasePassword -AsPlainText -Force  $DatabaseCredentials = New-Object PSCredential -ArgumentList $DatabaseUserName, $DatabaseSecurePassword    $EncryptionKeyPath = Join-Path $ServiceTierFolder 'DynamicsNAV.key'  $EncryptionSecurePassword = ConvertTo-SecureString -String $EncryptionPassword -AsPlainText -Force  New-NAVEncryptionKey -KeyPath $EncryptionKeyPath -Password $EncryptionSecurePassword -Force | Out-Null    Write-Host "Import Encryption Key"  Import-NAVEncryptionKey -ServerInstance NAV `  -ApplicationDatabaseServer $DatabaseServer `  -ApplicationDatabaseCredentials $DatabaseCredentials `  -ApplicationDatabaseName $DatabaseName `  -KeyPath $EncryptionKeyPath `  -Password $EncryptionSecurePassword `  -WarningAction SilentlyContinue `  -Force    Set-NAVServerConfiguration -ServerInstance "NAV" -KeyName "EnableSqlConnectionEncryption" -KeyValue "true" -WarningAction SilentlyContinue  Set-NAVServerConfiguration -ServerInstance "NAV" -KeyName "TrustSQLServerCertificate" -KeyValue $TrustSQLServerCertificate.Tostring().ToLowerInvariant() -WarningAction SilentlyContinue  Set-NavServerConfiguration -serverinstance "NAV" -databaseCredentials $DatabaseCredentials  } |
| Replace the content of start.ps1 with this:  New-NavContainer -accept\_eula `  -containerName myserver `  -includeCSide `  -auth NavUserPassword `  -additionalParameters @("-e databaseServer=navserver","-e databaseInstance=SQLEXPRESS","-e databaseName=CRONUS") `  -myScripts @("c:\test\setupdatabase.ps1")  Press F5 and wait for the container to complete. | |
| To inspect the log, write:  **docker logs myserver**  and you will see that the myserver container never starts the local SQL Server, instead it changes the database connection.  Try to connect to the navserver Web client and the myserver Web Client (on the Desktop) at the same time and you will see, that they are using the same database.  The highlighted section on the right are all outputs, which is a result of the script override. | Initializing...  Hostname is myserver  PublicDnsName is myserver  Running Specific Image  Using NavUserPassword Authentication  Starting Internet Information Server  Change Database Connection  Import Encryption Key  WARNING: You should encrypt the connection to the database to help protect  against network attacks, such as man-in-the-middle attacks.  To encrypt the connection, select the Enable Encryption on SQL Server  Connections setting (EnableSqlConnectionEncryption = True) and clear the Enable  Trust of SQL Server Certificate setting (TrustSQLServerCertificate = False).  WARNING: The new settings value will not take effect until you stop and restart  the service.  Modifying NAV Service Tier Config File for Docker  Creating Self Signed Certificate  Self Signed Certificate Thumbprint 46A9D00692C24EAC9F1B826991A6E566143FF422  Modifying NAV Service Tier Config File with Instance Specific Settings  Start NAV Service Tier  Using license file 'c:\run\my\license.flf'  Import NAV License  Creating DotNetCore NAV Web Server Instance  Creating http download site  Creating Windows user  Container IP Address: 172.18.235.116  Container Hostname : myserver  Container Dns Name : myserver  Web Client : http://myserver/NAV/WebClient/  Dev. Server : http://myserver  Dev. ServerInstance : NAV  Files:  http://myserver:8080/al-0.11.14434.vsix  Ready for connections! |
| Running **docker stats** now reveals two containers and the one running SQL Server and NAV uses more memory than the one running NAV only. |  |

# Using the Object Handling Functions in navcontainerhelper

|  |  |  |
| --- | --- | --- |
| Open the Nav Container Helper prompt and run  **New-NavContainer -accept\_eula -containerName myserver -includeCSide**  To create a CSide development environment next to the navserver container.  Use your Workshop VM credentials when asked for credentials. |  | |
| Navigate to the C:\DEMO\Extensions folder and examine the folders:  **myserver** is a folder with files specific for the myserver container  **navserver** is a folder with files specific for the navserver container  **Original-11.0.19097.0-W1** contains all the base objects for build 11.0.19097.0 (w1 version)  **Original-11.0.19097.0-W1-newsyntax** contains all the base objects for build 11.0.19097.0 (w1 version) in new syntax format (for txt2al)  The reason for these base object folders are for being able to create deltas from changes in a container. |  |
| Start the myserver CSIDE client and modify a few objects.  Please only create modifications which are allowed in extensions v1.  Save your modifications and close the classic development environment. |  |

|  |  |
| --- | --- |
| Run  **Export-ModifiedObjectsAsDeltas -containerName myserver -openfolder** |  |
| You should see a folder being opened with TXT files for new objects and DELTA files for changed.  If you navigate to the parent folder, you will find work folders for:   * original * modified * delta |  |
| Try also  **Convert-ModifiedObjectsToAl -containerName myserver -openFolder** |  |
| Inspect other object handling functions, especially import and compile functions. |  |

# Portainer.io

Portainer is a free GUI for maintaining your Docker environment.

|  |  |
| --- | --- |
| Portainer doesn’t work with IE and Edge doesn’t run on Windows Server 2016, so we need to download and install Chrome on the Workshop VM from: | <https://www.google.com/intl/en/chrome/browser/> |
| Copy the PowerShell script, paste it into PowerShell ISE and run it. The script will:  1. Reconfigure Docker deamon  2. Open port 2375 in the firewall  3. Create a Portainer directory  4. Get the IP address  5. Download and run the Portainer Docker image | '{  "hosts": ["tcp://0.0.0.0:2375", "npipe://"]  }' | Set-Content "C:\ProgramData\docker\config\daemon.json"  restart-service docker  netsh advfirewall firewall add rule name="Docker" dir=in action=allow protocol=TCP localport=2375  new-item -Path "C:\Portainer" -ItemType Directory  $ipAddress = (get-netadapter | Select-Object -First 1 | get-netipaddress | ? addressfamily -eq 'IPv4').ipaddress  docker run -d -v C:\Portainer:C:\data --name portainer --hostname portainer portainer/portainer -H tcp://${ipAddress}:2375 |
| Open Google Chrome and navigate to  <http://portainer:9000>  On your first connection, you will have to create an admin password for Portainer.  After that…  **Welcome to a free tool for maintaining your Docker environment** |  |