CQURE Academy

Windows Server 2019 Workshop

Document generated: 191029.010428

E-mail: info@cqure.pl | Twitter: @CQUREAcademy
https://cqure.pl

1. APPLOCKER: BASIC CONFIGURATION

Summary

AppLocker is an application whitelisting technology introduced with Microsoft's Windows 7 operating system. It allows restricting which programs users can execute based on the program's path, publisher, or hash, and in an enterprise can be configured via Group Policy.

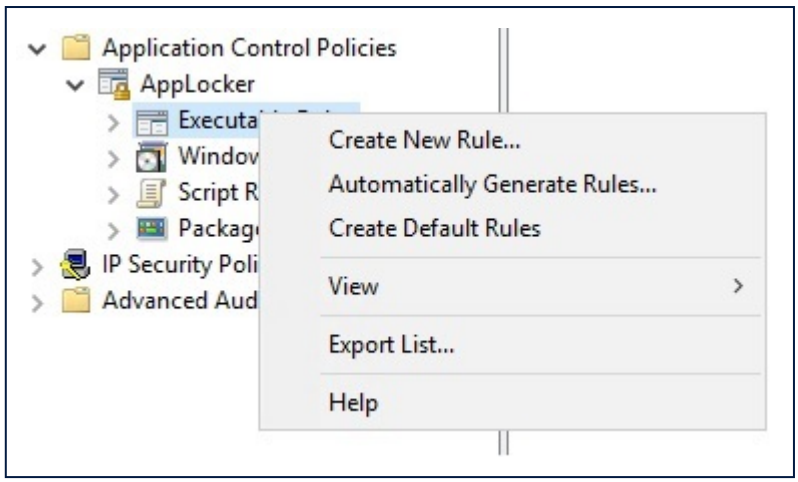
Unlike the earlier Software Restriction Policies, which was originally available for Windows XP and Windows Server 2003,[2] AppLocker rules can apply to individuals or groups. Policies are used to group users into different enforcement levels. For example, some users can be added to an 'audit' policy that will allow administrators to see the rule violations before moving that user to a higher enforcement level.

Lab 1: Enabling AppLocker

1. Using administrator account, open *Local Security Policy* settings.
2. Go to *Application Control Policies -> AppLocker*.
3. By default, there are no defined rules. Let's start by defining some rules - Windows already offers predefined set of basic rules. To enable them, go to subtree of AppLocker and in each, right click and choose *Create Default Rules* option.
4. Inspect the rules and understand why are they needed for system to work correctly.. Notice that ADMIN still has rights to execute anything.
5. To enable new rules, AppLocker service needs to be started. In elevated cmd type:

   ```
   net start appidsvc
   ```

6. Re login as normal user or force group policy update (`gpupdate /force`). Note that policy may still need couple more minutes before it starts working.
7. Open *Event Viewer*. Go to *Applications and Services Logs -> Microsoft -> Windows -> AppLocker*.
8. Search for event ID 8001 - this shows that AppLocker rules were updated.
9. As normal user, try to run some nonstandard binaries versus Microsoft binaries and compare the results.
10. Refresh the logs if necessary, read reported events.
11. Go back to *Local Security Policy* settings and define custom rules. To do that, right click on *Executable Rules* and select *Create New Rule*.



12. Allow running Putty for any user.
13. Test new rule
14. Enforce new rules for Executable
15. Force group policy update (`gpupdate /force`). Note that policy may still need couple more minutes before it starts working.
16. Test new rule, try to download some new software from internet (for example psftp.exe) and run it as standard user.
17. Analyze what has happened and try to remediate this problem.

Lab 2: Simple bypassing

1. As normal user download new software (choose something from https://live.sysinternals.com)
2. Test if software is blocked
3. Open CMD and copy executable to `C:\Windows\Temp\`
4. Try to run executable from `C:\Windows\Temp\` folder
5. Modify rules so that this trick will be blocked

Lab 3: Searching other writable paths

1. Download from https://live.sysinternals.com tool accesschk.exe
2. Open administrator CMD and analyze switches for accesschk.exe
3. Use this tool to find all elements writable by Users in `c:\Windows` folder
4. Add appropriate modifications to the rules

Lab 4: Danger executable

1. Add new DENY rules for path `C:\Windows\system32\WindowsPowerShell`
2. As standard user try to run powershell console (You should fail)
3. From mounted DVD copy payload.csproj to some work folder on C: drive
4. Open CMD as standard user and go to work folder
5. Run command `c:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exe`
6. Wait until your custom PowerShell will start and run commands

```
Get-Commands
$PSVersionTable.PSVersion
Get-Host
$ExecutionContext.SessionState.LanguageMode
[System.Console]::WriteLine("Hello AppLocker")
```

7. Correct rules so that this trick will not work
8. Verify your solution
9. Remember to remove rule that is blocking PowerShell

References

- https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/applocker/applocker-overview
- https://oddvar.moe/2017/12/13/applocker-case-study-how-insecure-is-it-really-part-1/
- https://oddvar.moe/2017/12/21/applocker-case-study-how-insecure-is-it-really-part-2/

2. WINDOWS ADMIN CENTER

Summary

Windows Admin Center (formerly codenamed Project Honolulu) is an evolution of Windows Server in-box management tools; it's a single pane of glass that consolidates all aspects of local and remote server management. As a locally deployed, browser-based management experience, an Internet connection and Azure aren't required. Windows Admin Center gives you full control of all aspects of your deployment, including private networks that aren't Internet-connected.

Lab 1: Installing Admin Center

1. Logon to **Host-0** and **Host-1**
2. Verify that both host can resolve name **cqure.lab**

```
nslookup cqure.lab
```

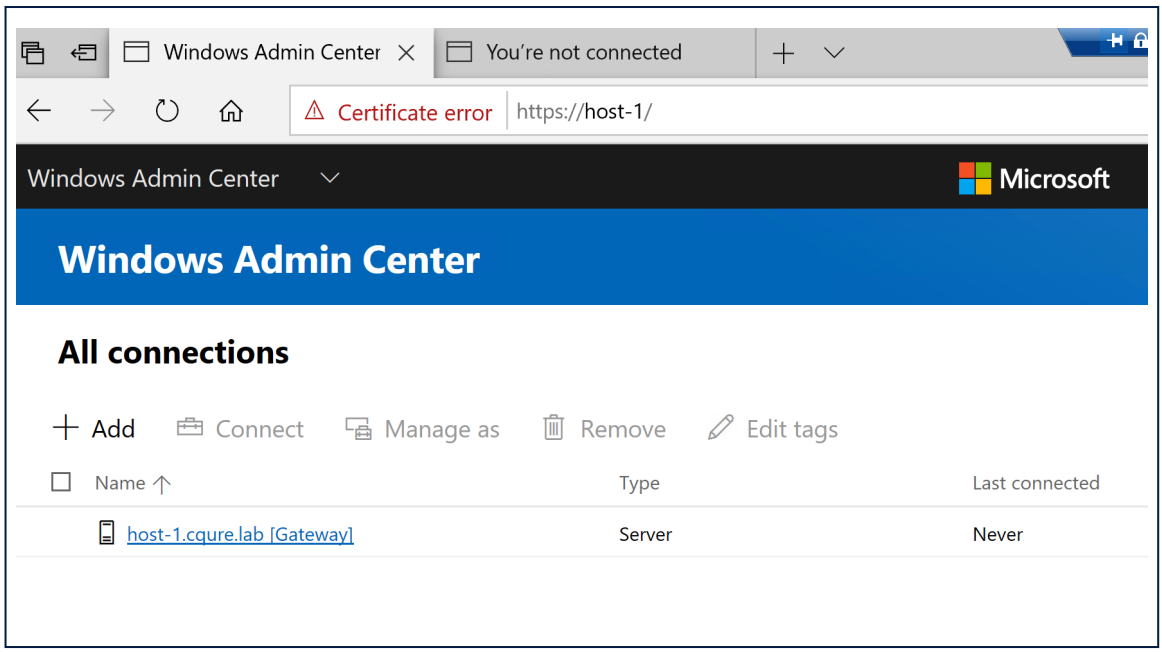
3. Solve any problems with network (IP addresses or DNS - 192.168.0.10)
4. Join both hosts to the domain **cqure.lab**

```
Add-Computer -DomainName cqure.lab
```

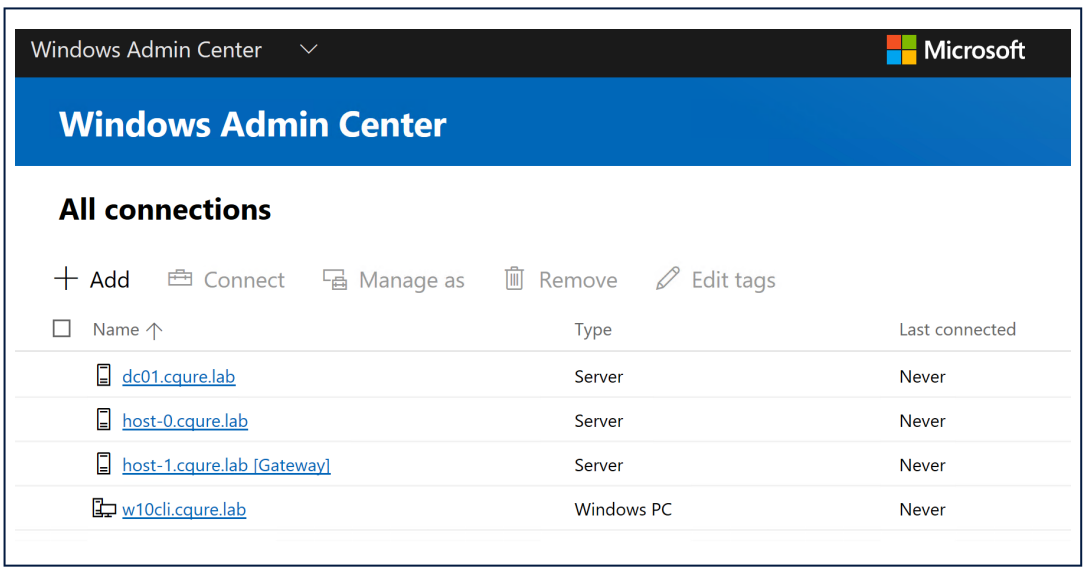
5. Restart machines
6. Logon to **Host-1** as CQURE\Administrator
7. Download Windows Admin Center installation package: https://aka.ms/WACDownload
8. Install WAC using default settings

Lab2 : Managing hosts with WAC

1. Logon to **W10Cli** as cqure\administrator
2. Open browser and go to https://host-1/
3. Confirm certificate error and continue to website



4. Add hosts to manage - do this by searching in Active Directory:
    1. DC1
    2. Host-0
    3. W10CLI



5. Connect to Host-0
6. Explore Admin Center:
    1. Execute PowerShell commands
    2. Connect with RDP
    3. Install IIS
    4. Create Scheduled task
    5. Explore Event Logs
    6. Add CQURE\Alice to Remote Desktop Users on W10Cli
7. Go to settings and install extensions for:
    1. Active Directory
    2. Containers

3. DHCP
4. DNS

8. Verify what you can do with these extensions and when they are appearing

Lab 3 Single Sign On

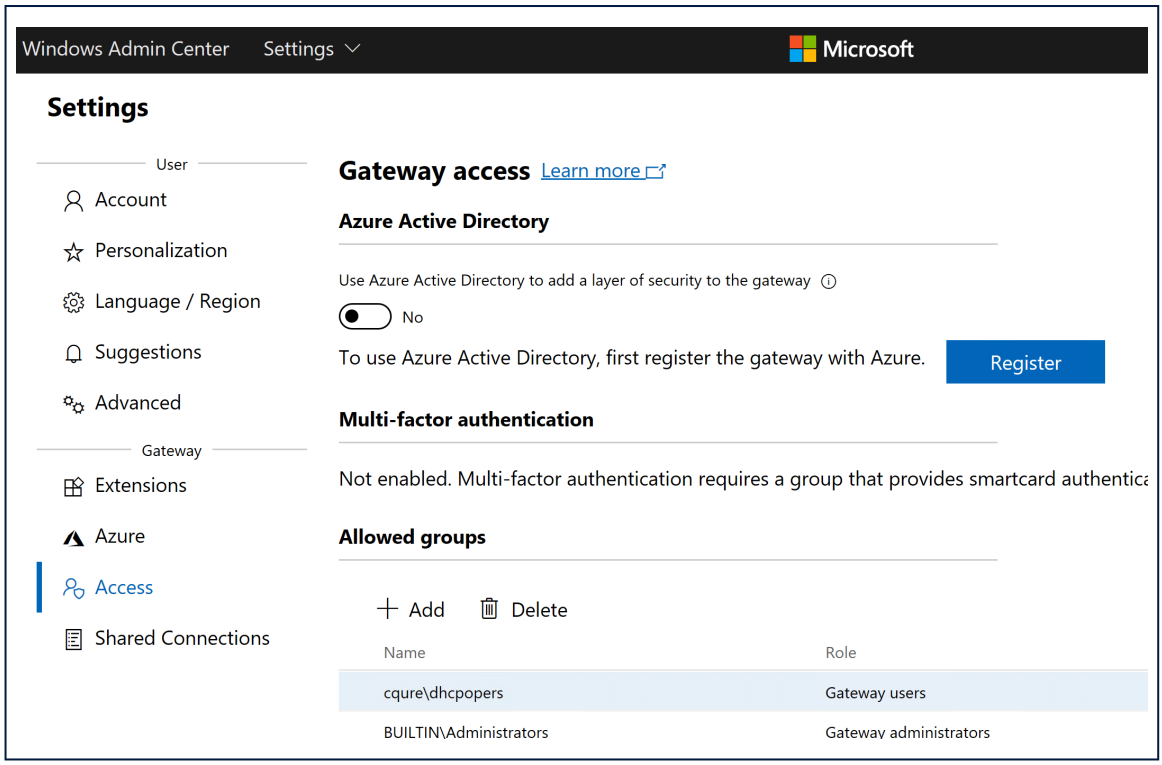
1. Logon to **WS19-DC01**
2. Open PowerShell and grant constrained delegation for WAC

```
$gateway = "HOST-1" # Machine where Windows Admin Center is installed
$node = "ManagedNode" # Machine that you want to manage
$gatewayObject = Get-ADComputer -Identity $gateway
$nodeObject = Get-ADComputer -Identity $node
Set-ADComputer -Identity $nodeObject -
PrincipalsAllowedToDelegateToAccount $gatewayObject
```

3. Verify that this time you are not asked for credentials

Lab 4 Delegation of privileges

1. Go in WAC to Settings
2. Add CQURE\DHCPopers to allowed groups



3. Go to management of Host-0 in WAC
4. On bottom left side click in Settings
5. Select Role-Based Access Controll and Apply
6. After succesfull activation add CQURE\alice to Host-1 local security groups:
    1. Windows Admin Center Readers
    2. Windows Admin Center Hyper-V Administrators
7. Sign out from W10Cli

8. Logon to W10Cli as CQURE\Alice
9. Go to https://Host-1/ and accept certificate
10. Test capabilities of Alice

References

- https://docs.microsoft.com/en-us/windows-server/manage/windows-admin-center/overview

3. WINDOWS CONTAINERS

Summary

Lab 1

Installation

1. Connect to HOST-0
2. Start powershell
3. In powershell:

```
Enable-WindowsOptionalFeature -Online -FeatureName containers -All
```

4. To use hyper-v containers the hyper-v needs too be installed too.

```
Install-WindowsFeature -Name Hyper-V -IncludeManagementTools -Restart
```

5. After you've opened PowerShell, your first step is to install the Microsoft Package Provider for Docker. This is done with the following command:

```
Install-Module -Name DockerMsftProvider -Repository PSGallery -Force
```

6. Now you can install the latest version of Docker with the following command:

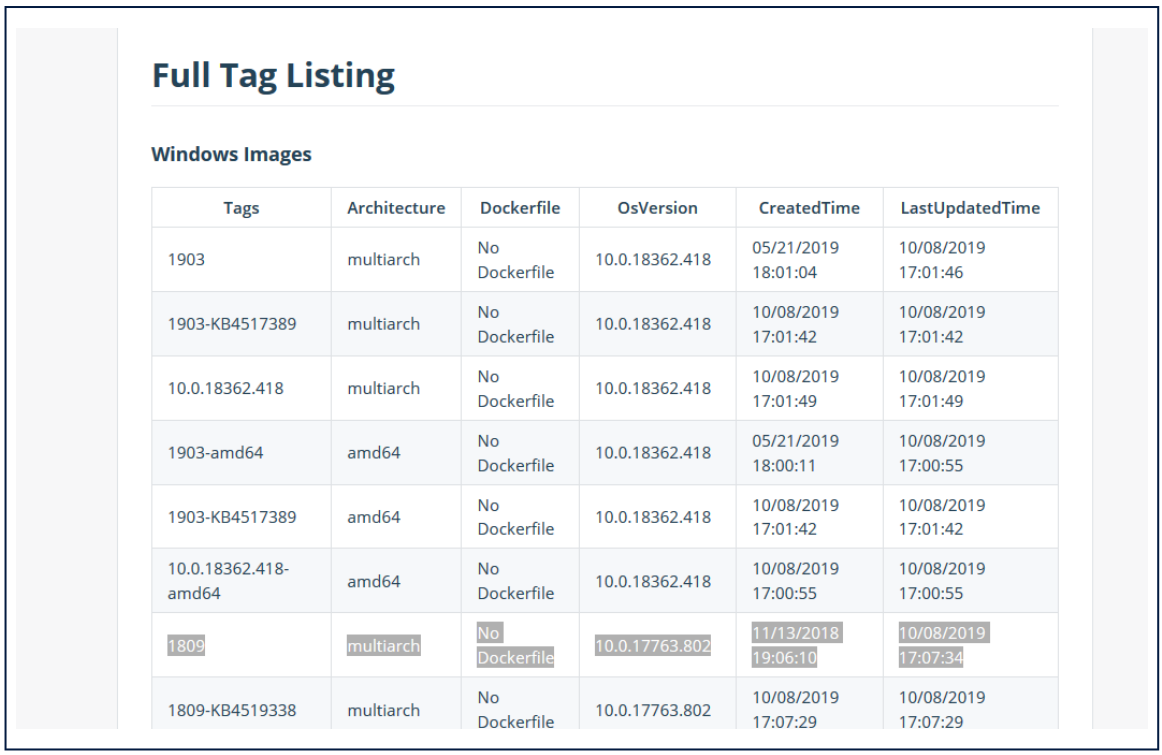
```
Install-Package -Name docker -ProviderName DockerMsftProvider
```

7. After Docker is installed, you need one more restart. You can do this through the graphical user interface (GUI), or you can just type the following into PowerShell:

```
Restart-Computer -Force
```

Pulling the image and running the container

1. Open browser and go to Docker Windows Nano Server image installation: https://store.docker.com/_/microsoft-windows-nanoserver



Full Tag Listing

Windows Images

Tags	Architecture	Dockerfile	OsVersion	CreatedTime	LastUpdatedTime
1903	multiarch	No Dockerfile	10.0.18362.418	05/21/2019 18:01:04	10/08/2019 17:01:46
1903-KB4517389	multiarch	No Dockerfile	10.0.18362.418	10/08/2019 17:01:42	10/08/2019 17:01:42
10.0.18362.418	multiarch	No Dockerfile	10.0.18362.418	10/08/2019 17:01:49	10/08/2019 17:01:49
1903-amd64	amd64	No Dockerfile	10.0.18362.418	05/21/2019 18:00:11	10/08/2019 17:00:55
1903-KB4517389	amd64	No Dockerfile	10.0.18362.418	10/08/2019 17:01:42	10/08/2019 17:01:42
10.0.18362.418-amd64	amd64	No Dockerfile	10.0.18362.418	10/08/2019 17:00:55	10/08/2019 17:00:55
1809	multiarch	No Dockerfile	10.0.17763.802	11/13/2018 19:06:10	10/08/2019 17:07:34
1809-KB4519338	multiarch	No Dockerfile	10.0.17763.802	10/08/2019 17:07:29	10/08/2019 17:07:29

2. To pull the nanoserver 1809 docker image run:

```
docker pull mcr.microsoft.com/windows/nanoserver:1809
```

3. List docker image repository:

```
docker images
```

4. Run cmd.exe in the pulled container:

```
docker run -it mcr.microsoft.com/windows/nanoserver:1809 cmd.exe

Flags explaination:
-t, --tty                                Allocate a pseudo-TTY
-i, --interactive                        Keep STDIN open even if not
attached
```

5. Run procExp and find runing docker
6. List c:/ drive - it's isolated from the host:

```
dir
```

7. Create new file in the docker image and exit:

```
echo "Hello World!" > C:\Users\ContainerUser\Hello.txt
exit
```

8. To see all running or exited containers execute the command:

```
docker ps -a
```

9. Run the following command to create the new 'HelloWorld' image based on the exited container. Replace <containerid> with the id of your container.

```
docker commit <containerid> helloworld
```

10. When completed show docker images one more time:

```
docker images
```

11. Run the new created container

```
docker run --rm helloworld cmd.exe /s /c type C:
\Users\ContainerUser\Hello.txt
```

```
Flags explained:
--rm                                Automatically remove the container
when it exits
```

12. Windows containers running on Windows Server default to running with process isolation. To run cmd.exe in the hyper-v isolated container use the --isolation=hyperv flag as following:

```
docker run --isolation=hyperv -it mcr.microsoft.com/windows/nanoserver:
1809 cmd.exe
```

13. Find container in process explorer
14. Exit the container:

```
exit
```

Dockerfile

1. Run powershell
2. Go to:

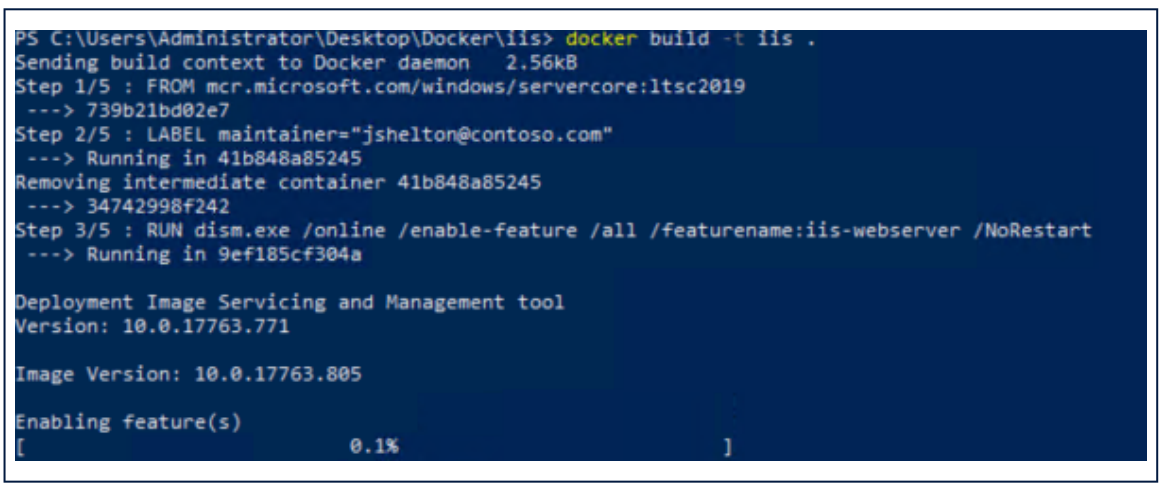
```
cd C:\Users\Administrator\Desktop\Docker\iis
```

3. Analyze the Dockerfile - it installs IIS and creates index.html file

```
ise Dockerfile
```

4. Build a new image called iis based on Dockerfile in the directory:

```
docker build -t iis .
```



5. Show docker images:

```
docker images
```

Networking

The first time the docker engine runs, it will create a default NAT network, 'nat', which uses an internal vSwitch and a Windows component named WinNAT. If there are any pre-existing external vSwitches on the host which were created through PowerShell or Hyper-V Manager, they will also be available to Docker using the transparent network driver and can be seen when you run the docker network ls command.

1. To list networks available for docker run:

```
docker network ls
```

2. Show vSwitches:

```
Get-VMSwitch
```

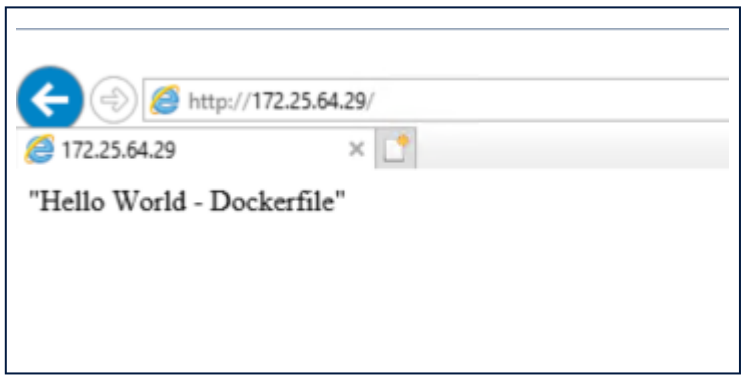
3. Run created iis image:

```
docker run -it iis
```

4. Run ipconfig and route print and compare it with ipconfig on the host - the container is behind the internal nat

```
ipconfig
route print
```

5. Open Internet Explorer and open the page hosted on IIS container, replace <container_ip> with container ip address



6. Exit container
7. To create transparent network first find adapter name:

```
Get-NetAdapter
```

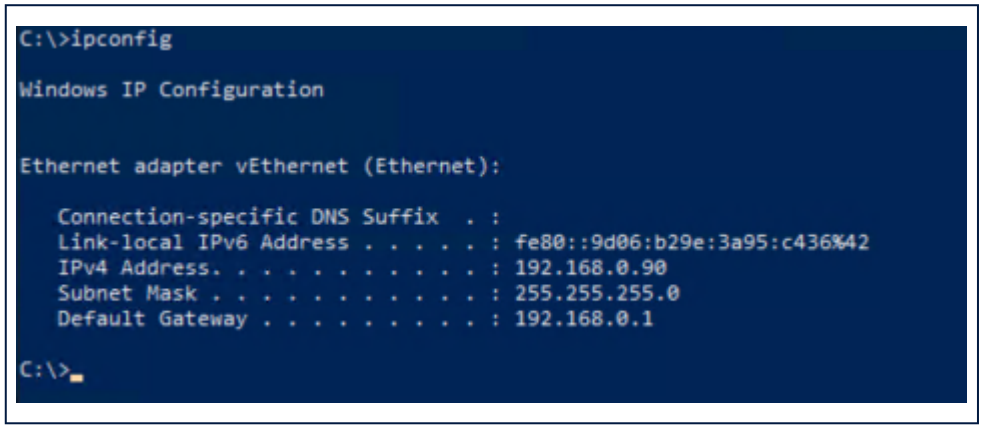
8. Create transparent network TransparentNet

```
docker network create -d transparent -o
com.docker.network.windowsshim.interface="Ethernet" --
subnet=192.168.0.0/24 --gateway=192.168.0.10 TransparentNet
```

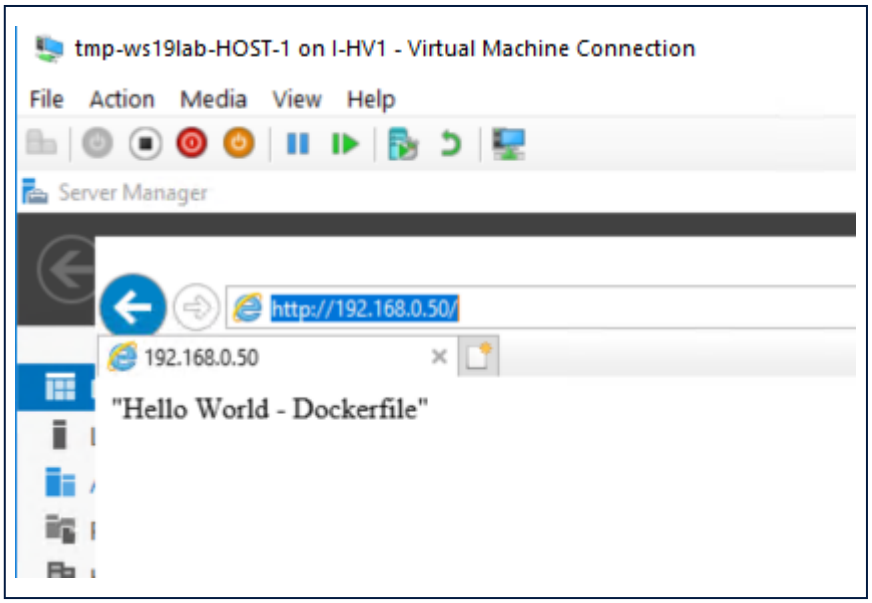
9. Run the container with static IP (192.168.0.50) set:

```
docker run -it --network=TransparentNet --ip 192.168.0.50 iis
```

10. Run ipconfig in the container:



11. From HOST-1 Windows Server 2019 machine in the same network, try to reach the IIS page http://192.168.0.50/



Storage

1. Go back to machine Host-0
2. Use **dir** command to check the available space in the container - by default the storage limit is 20GB.
3. Exit the container
4. Run the container with --storage-opt "size=30GB" param to change the storage limit:

```
docker run -it --storage-opt "size=30GB" iis
```

5. Use dir one more time to check available space:
6. To share a folder on a host with the container use -v switch and permissions switch: RO =Read Only or RW = Read-Write (which is set by default):

```
docker run -it -v C:\Shared_data:C:\Shared_data:RO iis
```

7. Check that it is possible to read a c:\shared_data\cqure.txt file in the container

```
type c:\shared_data\cqure.txt
exit
```

Docker Swarm

Swarm mode is a Docker feature that provides built in container orchestration capabilities, including native clustering of Docker hosts and scheduling of container workloads. A group of Docker hosts form a "swarm" cluster when their Docker engines are running together in "swarm mode." For additional context on swarm mode, refer to Docker's main documentation site.

Manager nodes and worker nodes

A swarm is composed of two types of container hosts: manager nodes, and worker nodes. Every swarm is initialized via a manager node, and all Docker CLI commands for controlling and monitoring a swarm must be executed from one of its manager nodes. Manager nodes can be thought of as "keepers" of the Swarm state—together, they form a consensus group that maintains awareness of the state of services running on the swarm, and it's their job to ensure that the swarm's actual state always matches its intended state, as defined by the developer or admin.

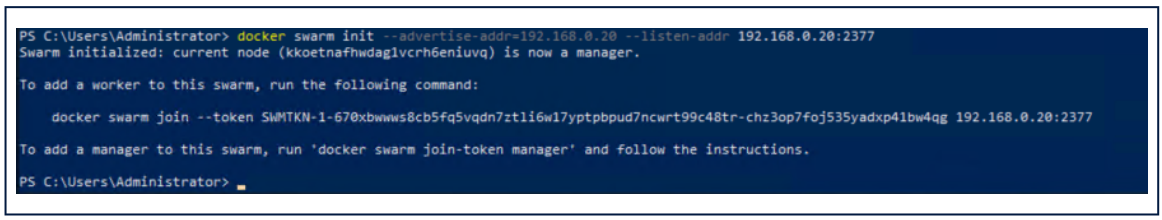
Required open ports:

The following ports must be available on each host.

```
TCP port 2377 for cluster management communications
TCP and UDP port 7946 for communication among nodes
UDP port 4789 for overlay network traffic
```

1. On **Host-0** and **Host-1** disable firewall on all profiles
2. Initialize a swarm, replace <HOSTIPADDRESS> with Host-0 IP (192.168.0.20):

```
docker swarm init --advertise-addr=<HOSTIPADDRESS> --listen-addr
<HOSTIPADDRESS>:2377
```



3. Save the adding worker command with token
4. In case of forgetting the token, use:

```
docker swarm join-token worker
```

5. To check docker status (check if the node is a swarm manager), use:

```
docker info
```

```
PS C:\Users\Administrator\Desktop\Docker\iis> docker info
Client:
 Debug Mode: false
 Plugins:
  cluster: Manage Docker clusters (Docker Inc., v1.2.0)

Server:
 Containers: 7
  Running: 0
  Paused: 0
  Stopped: 7
 Images: 7
 Server Version: 19.03.4
 Storage Driver: windowsfilter
  Windows:
 Logging Driver: json-file
 Plugins:
  Volume: local
  Network: ics internal l2bridge l2tunnel nat null overlay private transparent
  Log: awslogs etwlogs fluentd gcplogs gelf json-file local logentries splunk syslog
 Swarm: active
  NodeID: vrifk1p5tbbmooluw8tdd2hg6
  Is Manager: true
  ClusterID: ycjoh06akicrlb2qvnlx1tz6q
  Managers: 1
  Nodes: 1
  Default Address Pool: 10.0.0.0/8
  SubnetSize: 24
  Data Path Port: 4789
  Orchestration:
   Task History Retention Limit: 5
  Raft:
```

6. Go to Host-1 and run saved command:

```
docker swarm join --token <token> <HOSTIPADDRESS>:2377
```

7. Check details:

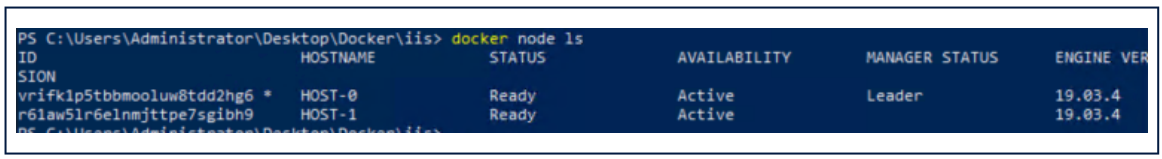
```
docker info
```

```
PS C:\Users\Administrator> docker info
Client:
 Debug Mode: false
 Plugins:
  cluster: Manage Docker clusters (Docker Inc., v1.2.0)

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 19.03.4
 Storage Driver: windowsfilter
  Windows:
 Logging Driver: json-file
 Plugins:
  Volume: local
  Network: ics internal l2bridge l2tunnel nat null overlay private transparent
  Log: awslogs etwlogs fluentd gcplogs gelf json-file local logentries splunk syslog
 Swarm: active
  NodeID: uid81la2baiswga6us5f08gzg
  Is Manager: false
  Node Address: 192.168.0.101
  Manager Addresses:
   192.168.0.20:2377
 Default Isolation: process
 Kernel Version: 10.0 17763 (17763.1.amd64fre.rs5_release.180914-1434)
 Operating System: Windows Server 2019 Datacenter Version 1809 (OS Build 17763.107)
 OSType: windows
 Architecture: x86_64
 CPUs: 2
```

8. On **Host-0** show details about swarm nodes:

```
docker node ls
```



9. On **Ubuntu** machine run (sign in as cqure wih password: P@ssw0rd):

```
sudo tcpdump
```

10. On docker manager host **Host-0** run the command to create a service that will ping the ubuntu machine:

```
docker service create --name=nanoserv_service --replicas=2
mcr.microsoft.com/windows/nanoserver:1809 ping -n 10000 192.168.0.100
```

11. ICMP packets are sent from both **Host-0** and **Host-1** to Ubuntu machine.
12. On Host-0 to check the logs run:

```
docker service logs nanoserv_service
```

13. To remove the service run:

```
docker service rm nanoserv_service
```

14. Add Host-2 to the swarm
15. Repeat steps 10-12
16. After you determine which hosts are running containers shutdown one of the hosts
17. Test if you have still two replicas running

Lab2 - DNS in Windows Container

1. Download Windows 2019 Core edition image
2. Crete your own image which will contains DNS server
3. Use dokerfile to ensure that after container will start it will execute commands

```
Add-DnsServerPrimaryZone -Name company.lab -ZoneFile company.lab.dns -
Verbose
$localip = Get-NetIPAddress -AddressFamily ipv4 -InterfaceAlias
'vEthernet (Ethernet)*'
Add-DnsServerResourceRecord -ZoneName company.lab -A -Name (Get-Content
env:computername) -IPv4Address $localip.ipaddress -Verbose
ping -t localhost > NULL
```

4. Try to use also healthcheck that will test result of:

```
Get-Service dns
```

References

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

4. WINDOWS CONTAINERS - USE CASE

Summary

In this lab we will build containers for:

- SQL Server
- IIS Server

Additionally this will build and deploy images with docker compose

Setup:

1. start powershell and install SSMS

```
md c:\install
cd install
Start-BitsTransfer -Source https://aka.ms/ssmsfullsetup -Destination
SSMS-Setup-ENU.exe
.\SSMS-Setup-ENU.exe
Install-WindowsFeature Web-Mgmt-Tools
# Install-WindowsFeature Web-Mgmt-Tools, Web-Mgmt-Service
Start-BitsTransfer -Source https://cqure.pl/wp-content/uploads/2019/10/
DonutShop-Publish.zip -Destination DonutShop-Publish.zip
Expand-Archive .\DonutShop-Publish.zip
```

Lab 1:

2. Login to **Host-0** and open powershell
3. Create work folder and enter into it

```
md c:\docApp
cd c:\docApp
```

4. Download Sql Server 2017 Express Edition Media

```
Start-BitsTransfer -Source https://go.microsoft.com/fwlink/?
linkid=853017 -Destination sse2017.exe
.\sse2017.exe
```

5. Download Media for later installation and place it in c:\docApp\SSE2017\
6. Execute c:\docApp\SSE2017\SQLEXPR_x64_ENU.exe and confirm default extraction location
7. Create a folder c:\docApp\sqlbuilder\source and copy downloaded and extracted content into it

```
md c:\docApp\sqlbuilder\source
mv c:\docApp\SSE2017\SQLEXPR_x64_ENU c:\docApp\sqlbuilder\source\
```

8. Create and edit dockerfile for sqlbuilder

```
New-Item c:\docApp\sqlbuilder\dockerfile
ise c:\docApp\sqlbuilder\dockerfile
```

9. Copy into docker file:

```
# escape=`
FROM mcr.microsoft.com/windows/servercore:1809
ADD source c:\source
RUN C:\source\SQLEXPR_x64_ENU\SETUP.exe /Q /ACTION=INSTALL /
FEATURES=SQLENGINE /INSTANCENAME=MSSQLSERVER /SECURITYMODE=SQL /
SAPWD="P@ssw0rd" /SQLSVCACCOUNT="NT AUTHORITY\System" /AGTSVCACCOUNT="NT
AUTHORITY\System" /SQLSYSADMINACCOUNTS="BUILTIN\Administrators" /
IACCEPTSQLSERVERLICENSETERMS=1 /TCPENABLED=1 /UPDATEENABLED=False
RUN powershell -Command " `
Remove-Item C:\source -Recurse `
"
ENTRYPOINT "ping -t localhost > NULL"
```

10. Execute docker build command

```
cd c:\docApp\sqlbuilder\
docker image build --tag sqlbuilder .
```

11. Create volumes folder

```
md c:\docApp\volumes\sql
md c:\docApp\volumes\iis
```

12. Run container

```
docker run --detach -it --name sqlbuilder --hostname sqlbuilder --
network TransparentNet --ip 192.168.0.200 --dns 192.168.0.10 --volume c:
\docApp\volumes\sql:c:\volsql sqlbuilder
```

13. Open SSMS and connect as SA with pass P@ssw0rd to 192.168.0.200
14. Create new database MyApp
15. Create new SQL Login MyAppUser with password P@ssw0rd
16. Disable password enforce for this user
17. In User Mapping grant db_datareader and db_datawriter for MyApp database
18. Open new query tab and run SQL batch

```
CREATE DATABASE [MyApp]
CONTAINMENT = NONE
ON  PRIMARY
( NAME = N'MyApp', FILENAME = N'C:\volsql\MyApp.mdf' , SIZE =
8192KB , FILEGROWTH = 65536KB )
LOG ON
( NAME = N'MyApp_log', FILENAME = N'C:\volsql\MyApp_log.ldf' , SIZE =
8192KB , FILEGROWTH = 65536KB )
GO
CREATE LOGIN [MyAppUser] WITH PASSWORD=N'P@ssw0rd',
DEFAULT_DATABASE=[master], CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
GO
USE [MyApp]
GO
CREATE USER [MyAppUser] FOR LOGIN [MyAppUser]
GO
USE [MyApp]
GO
```

```
ALTER ROLE [db_datareader] ADD MEMBER [MyAppUser]
GO
USE [MyApp]
GO
ALTER ROLE [db_datawriter] ADD MEMBER [MyAppUser]
GO
USE MyApp
GO
CREATE TABLE PersonDonuts(
Person varchar(50),
NumDonuts int
)
GO
INSERT INTO PersonDonuts VALUES ('Mike', 100),('Tom', 50),('Charlie',
1)
GO
```

19. Disconnect
20. Stop docker

```
docker rm sqlbuilder -f
```

21. Create new folder and docker file

```
md c:\docApp\CQsql
New-Item c:\docApp\CQsql\dockerfile
ise c:\docApp\CQsql\dockerfile
```

22. Copy to docker file script

```
# escape=`
FROM contsqlbuilder
ENTRYPOINT powershell -Command " `
Invoke-Sqlcmd -Query """CREATE DATABASE MyApp ON (FILENAME='c:
\volsql\myapp.mdf'),(FILENAME='c:\volsql\myapp_log.ldf') FOR ATTACH;""" ;
`
Invoke-Sqlcmd -Query """USE MyApp; EXEC sp_change_users_login
'Auto_Fix', 'myappuser', NULL, 'P@ssw0rd';""" ; `
ping -t localhost > NULL `
"
HEALTHCHECK --interval=2s `
CMD powershell -Command `
try { `
$result = Get-Service MSSQLServer ; `
if ($result.status -eq 'Running') { exit 0 } `
else { exit 1 } ; `
} catch { exit 1 }
```

23. Save file and build image

```
cd c:\docApp\CQsql
docker image build --tag cqsql .
```

24. Test the new image

```
docker run --detach -it --name cqsql --hostname cqsql --network
TransparentNet --ip 192.168.0.202 --dns 192.168.0.10 --volume c:
\docApp\volumes\sql:c:\volsql cqsql
```

25. Reconnect with SSMS this time to 192.168.0.202
26. Check if database is visible
27. **Dont Stop** container
28. Create IISbuilder container

```
New-Item c:\docApp\iisbuilder\dockerfile
ise c:\docApp\iisbuilder\dockerfile
```

29. Copy into docker file:

```
# escape=`
FROM mcr.microsoft.com/windows/servercore:1809
RUN powershell -Command " `
Install-WindowsFeature Web-Server,Web-Asp-Net45,Web-Mgmt-Service -
Verbose ; `
New-ItemProperty -Path HKLM:\software\microsoft\WebManagement\Server -
Name EnableRemoteManagement -Value 1 -Force ; `
Set-Service -Name wmsvc -StartupType Automatic ; `
New-LocalUser iisadmin -Password ( 'P@ssw0rd' | ConvertTo-
SecureString -AsPlainText -Force) ; `
Add-LocalGroupMember -Group Administrators -Member iisadmin `
"
ENTRYPOINT "ping -t localhost > NULL"
```

30. Execute docker build command

```
cd c:\docApp\iisbuilder\
docker image build --tag iisbuilder .
```

31. Copy sourcecode of application into volume

```
Robocopy.exe c:\install\DonutShop-Publish\DonutShop-Publish\ C:
\docApp\volumes\iis\DonutShop-Publish /E
```

32. Run container

```
docker run --detach -it --name cqiis --hostname cqiis --network
TransparentNet --ip 192.168.0.201 --dns 192.168.0.10 --volume c:
\docApp\volumes\iis:c:\voliis iisbuilder
```

33. Open IIS Manager and connect to server 192.168.0.201 with username iisadmin and password P@ssw0rd
34. Test functionality of IIS
35. Close IIS manager
36. Stop and remove container

```
docker rm cqiis -f
```

37. Create new folder and docker file

```
md c:\docApp\cqIIS
New-Item c:\docApp\cqIIS\dockerfile
ise c:\docApp\cqIIS\dockerfile
```

38. Copy to docker file script

```
# escape=`
FROM iisbuilder
ENTRYPOINT powershell -Command " `
Remove-Website -Name 'Default Web Site' ; `
New-Website -Name MyApp -PhysicalPath C:\voliis\donutshop-publish ; `
Stop-Website -Name MyApp ; Start-Website -Name MyApp ; `
ping -t localhost > NULL `
"
HEALTHCHECK --interval=2s `
CMD powershell -Command `
try { `
$result = Get-Service W3SVC ; `
if ($result.status -eq 'Running') { exit 0 } `
else { exit 1 } ; `
} catch { exit 1 }
```

39. Save file and build image

```
cd c:\docApp\cqIIS
docker image build --tag cqiis .
```

40. Test the new image

```
docker run --detach -it --name cqiis --hostname cqiis --network
TransparentNet --ip 192.168.0.201 --dns 192.168.0.10 --volume c:
\docApp\volumes\iis:c:\voliis cqiis
```

41. Open web browser and go to website http://192.168.0.201

Lab 2: Docker compose

1. Download docekr compose

```
[Net.ServicePointManager]::SecurityProtocol =
[Net.SecurityProtocolType]::Tls12
Invoke-WebRequest "https://github.com/docker/compose/releases/download/
1.24.1/docker-compose-Windows-x86_64.exe" -UseBasicParsing -OutFile
$Env:ProgramFiles\Docker\docker-compose.exe
docker-compose --version
```

2. Create folder and yml file

```
md c:\docApp\myapp
new-item c:\docapp\myapp\docker-compose.yml
ise c:\docapp\myapp\docker-compose.yml
```

3. Edit yml file and copy into it

```
version: "3.7"
services:
```

```
    cqsql:
     image: cqsql
     hostname: cqsql
     container_name: cqsql
     volumes:
       - c:\docApp\volumes\sql:c:\volsql
     networks:
       prodnet:
         ipv4_address: 192.168.0.202
    cqiis:
     image: cqiis
     hostname: cqiis
     container_name: cqiis
     depends_on:
       - cqsql
     volumes:
       - c:\docApp\volumes\iis:c:\voliis
     networks:
       prodnet:
         ipv4_address: 192.168.0.201
    networks:
    TransparentNet:
     external: true
```

4. Stop and remove previous containers

```
docker rm cqiis, cqsql -f
```

5. Run docker-compose for myapp

```
cd c:\docApp\myapp
docker-compose up -d
```

6. Test application

5. DEVICE GUARD (WDAC)

Summary

Windows 10 includes a set of hardware and OS technologies that, when configured together, allow enterprises to "lock down" Windows 10 systems so they operate with many of the properties of mobile devices. In this configuration, specific technologies work together to restrict devices to only run authorized apps by using a feature called configurable code integrity, while simultaneously hardening the OS against kernel memory attacks through the use of virtualization-based protection of code integrity (more specifically, HVCI).

Configurable code integrity policies and HVCI are very powerful protections that can be used separately. However, when these two technologies are configured to work together, they present a very strong protection capability for Windows 10 devices.

Using configurable code integrity to restrict devices to only authorized apps has these advantages over other solutions:

1. Configurable code integrity policy is enforced by the Windows kernel itself. As such, the policy takes effect early in the boot sequence before nearly all other OS code and before traditional antivirus solutions run.
2. Configurable code integrity allows customers to set application control policy not only over code running in user mode, but also kernel mode hardware and software drivers and even code that runs as part of Windows.
3. Customers can protect the configurable code integrity policy even from local administrator tampering by digitally signing the policy. This would mean that changing the policy would require both administrative privilege and access to the organization's digital signing process, making it extremely difficult for an attacker with administrative privilege, or malicious software that managed to gain administrative privilege, to alter the application control policy.
4. The entire configurable code integrity enforcement mechanism can be protected by HVCI, where even if a vulnerability exists in kernel mode code, the likelihood that an attacker could successfully exploit it is significantly diminished. Why is this relevant? That's because an attacker that compromises the kernel would otherwise have enough privilege to disable most system defenses and override the application control policies enforced by configurable code integrity or any other application control solution.

Lab 1 - Windows Defender Application Control

1. Logon to **Host-0**
2. Open powershell as administrator
3. Create folder

```
md C:\DeviceGuard
cd c:\DeviceGuard
```

4. Create first initial policy

```
$CIPolicyPath = 'c:\DeviceGuard\'
$InitialCIPolicy = $CIPolicyPath+"InitialScan.xml"
$CIPolicyBin = $CIPolicyPath+"DeviceGuardPolicy.bin"
New-CIPolicy -Level PcaCertificate -Fallback FileName $InitialCIPolicy
-UserPEs 3> InitialCIPolicyLog.txt
```

5. Open and analyze policy:

```
ise $InitialCIPolicy
```

6. Open documentation and check settings https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/select-types-of-rules-to-create
7. Enable options for your initial policy

```
#0 Enabled:UMCI
Set-RuleOption -FilePath $InitialCIPolicy -Option 0
#3 Enabled:Audit Mode (Default)
Set-RuleOption -FilePath $InitialCIPolicy -Option 3
#6 Enabled:Unsigned System Integrity Policy (Default)
Set-RuleOption -FilePath $InitialCIPolicy -Option 6
#9 Enabled:Advanced Boot Options Menu
Set-RuleOption -FilePath $InitialCIPolicy -Option 9
#10 Enabled:Boot Audit on Failure
Set-RuleOption -FilePath $InitialCIPolicy -Option 10
```

8. Open and analyze modified policy:

```
ise $InitialCIPolicy
```

9. Enable audit policy

```
ConvertFrom-CIPolicy $InitialCIPolicy $CIPolicyBin
Copy-Item $CIPolicyBin "C:\Windows\System32\CodeIntegrity\SIPolicy.p7b"
```

10. Restart computer

```
Restart-Computer
```

11. Logon again to **Host-0**
12. Run some application from sysinternals
13. Download 7zip and notepad++ and putty.exe from internet and install it and run it
14. Prepare second policy which will include the new programs

```
$CIPolicyPath = 'c:\DeviceGuard\'
$InitialCIPolicy = $CIPolicyPath+"InitialScan.xml"
$CIPolicyBin = $CIPolicyPath+"DeviceGuardPolicy.bin"
```

15. Open and analyze modified policy (try to find rules responsible for 3 new programs installed in previous point):

```
ise $InitialCIPolicy
```

16. Remove rule for putty.exe

```
cd $CIPolicyPath
Remove-CIPolicyRule -FilePath $InitialCIPolicy -id
"ID_ALLOW_SOMENUMBERHERE_FOR_PUTTY"
```

17. Open and analyze modified policy (try to confirm that there is no rule for putty.exe):

```
ise $InitialCIPolicy
```

18. Test policy

```
ConvertFrom-CIPolicy $InitialCIPolicy $CIPolicyBin
Copy-Item $CIPolicyBin "C:\Windows\System32\CodeIntegrity\SIPolicy.p7b"
Restart-Computer
```

19. Try to run putty and notepad++
20. Test if msbuild.exe is starting properly

```
& 'C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe'
```

21. Find events in Event log for Device Guard
22. Download Microsoft recommended block list https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/microsoft-recommended-block-rules
23. Save it or copy it to c:\DeviceGuard
24. Merge it with last policy and apply

```
$CIPolicyPath = 'c:\DeviceGuard\'
$MSSuggestedPolicy = 'c:\DeviceGuard\MSrecomended.xml'
$MergedCIPolicy = $CIPolicyPath+"MergedPolicy_MSrecomended.xml"
$InitialCIPolicy = $CIPolicyPath+"InitialScan.xml"
$CIPolicyBin = $CIPolicyPath+"NewDeviceGuardPolicy_MSrecomended.bin"
Merge-CIPolicy -PolicyPaths $InitialCIPolicy,$MSSuggestedPolicy -
OutputFilePath $MergedCIPolicy
ConvertFrom-CIPolicy $MergedCIPolicy $CIPolicyBin
Copy-Item $CIPolicyBin "C:\Windows\System32\CodeIntegrity\SIPolicy.p7b"
```

25. Restart computer to test it
26. Logon to **Host-0**
27. Open powershell as administrator and test MSBUILD.exe

```
& 'C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe'
```

28. Find events for device guard
29. Prepare new policy with all audited events

```
$CIPolicyPath = 'c:\DeviceGuard\'
cd $CIPolicyPath
$CIAuditedPolicy = 'c:\DeviceGuard\Audited.xml'
$MergedCIPolicy = $CIPolicyPath+"MergedPolicy_Audited.xml"
$InitialCIPolicy = $CIPolicyPath+"MergedPolicy_MSrecomended.xml"
$CIPolicyBin =
$CIPolicyPath+"NewDeviceGuardPolicy_Audited_and_recomended.bin"
New-CIPolicy -Audit -Level Hash -Fallback FileName -FilePath
$CIAuditedPolicy -UserPEs 3> CIAuditedPolicyLog.txt
```

30. Open and analyze modified policy (try to find for putty):

```
ise $CIAuditedPolicy
```

31. Remove rule for putty.exe

```
cd $CIPolicyPath
Remove-CIPolicyRule -FilePath $CIAuditedPolicy -id
"ID_ALLOW_SOMENUMBERHERE_FOR_PUTTY"
```

32. Merge policy and apply it

```
Merge-CIPolicy -PolicyPaths $InitialCIPolicy,$CIAuditedPolicy -
OutputFilePath $MergedCIPolicy
ConvertFrom-CIPolicy $MergedCIPolicy $CIPolicyPath
Copy-Item $CIPolicyBin "C:\Windows\System32\CodeIntegrity\SIPolicy.p7b"

Restart-computer
```

33. Logon again and try to run putty.exe (should still run)
34. Open powershell and copy merged policy

```
$CIPolicyPath = 'c:\DeviceGuard\'
cd $CIPolicyPath
$CIAuditedPolicy = 'c:\DeviceGuard\Audited.xml'
$InitialCIPolicy = $CIPolicyPath+"MergedPolicy_Audited.xml"
$EnforcedCIPolicy = $CIPolicyPath+"EnforcedPolicy.xml"
$CIPolicyBin = $CIPolicyPath+"EnforcedDeviceGuardPolicy.bin"
copy $InitialCIPolicy $EnforcedCIPolicy
```

35. Open policy and look at options (search for option 3)

```
ise $EnforcedCIPolicy
```

36. Remove option 3 and enforce policy

```
dir $CIPolicyPath
Set-RuleOption -FilePath $EnforcedCIPolicy -Option 3 -Delete
ConvertFrom-CIPolicy $EnforcedCIPolicy $CIPolicyBin
Copy-Item $CIPolicyBin "C:\Windows\System32\CodeIntegrity\SIPolicy.p7b"
```

37. Restart machine

```
Restart-computer
```

38. Sign in and try to run MSBuild and putty

References

- https://docs.microsoft.com/en-us/windows/security/threat-protection/device-guard/introduction-to-device-guard-virtualization-based-security-and-windows-defender-application-control
- https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/windows-defender-application-control

6. DEVICE GUARD: NETWORK PROTECTION

Summary

Network Protection helps reduce the attack surface of your devices from Internet-based events. It prevents users from using any application to access dangerous domains that may host phishing scams, exploits, and other malicious content on the Internet.

Setup

Make sure that Network Protection is disabled before the demo.

```
Set-MpPreference -EnableNetworkProtection 0
```

Network connection is required.

Demo

1. Use Chrome and navigate to: https://smartscreentestratings2.net/ - page is accessed
2. Use Edge and navigate to: https://smartscreentestratings2.net/ - alert is displayed, but user can ignore it and access the page
3. In powershell, as administrator, enable the feature:

   ```
   Set-MpPreference -EnableNetworkProtection Enabled
   ```

4. Revisit the page in Edge and Chrome. Both attempts should fail.

7. POWERSHELL: JUST ENOUGH ADMINISTRATION

Summary

Just Enough Administration (JEA) is a security technology that enables delegated administration for anything managed by PowerShell. With JEA, you can:

- Reduce the number of administrators on your machines using virtual accounts or group-managed service accounts to perform privileged actions on behalf of regular users.
- Limit what users can do by specifying which cmdlets, functions, and external commands they can run.
- Better understand what your users are doing with transcripts and logs that show you exactly which commands a user executed during their session.

Lab 1: Powershell basics

1. Logon to **WS19-DC**
2. Execute commands one by one and analyze output:

```
get-help get-help
get-module
get-command
get-command | measure
alias
Get-PSDrive
cd cert:\
ls
cd .\\CurrentUser\My
ls
Get-PackageProvider -ListAvailable
Get-PackageSource
Get-PackageProvider Chocolatey
Find-Package Astley
Install-Package Astley
Find-package PowerShellCookbook
Install-Package PowerShellCookbook -AllowClobber
Get-Service Spooler | Show-Object
get-service | ogv -PassThru | Stop-Service
```

3. Create a new file test1.ps1 with content:

```
write-host  "Script successfully executed"
```

4. Try to execute file:

```
Set-ExecutionPolicy -ExecutionPolicy All
C:\awsc\test.ps1
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy Unrestricted
C:\awsc\test.ps1
```

```
5  Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy Default
   Set-ExecutionPolicy -ExecutionPolicy Default
```

Lab 2 : JEA

1. Logon to WS19-DC
2. Prepare users and group

```
New-ADUser -Name Alice -PasswordNeverExpires $true -AccountPassword
(ConvertTo-SecureString -AsPlainText 'P@ssw0rd' -force) -enabled $true
New-ADGroup -Name DHCPopers -GroupScope Global
Get-ADGroup DHCPopers
Add-ADGroupMember DHCPopers Alice
```

3. Set variables:

```
$modpath = "$env:ProgramFiles\WindowsPowerShell\Modules\JeaDhcpOper"
$jeaConfigPath = "$env:ProgramData\JEAConfiguration"
```

4. Create folders:

```
mkdir $modpath
mkdir $modpath\RoleCapabilities
mkdir $jeaConfigPath -Force
```

5. Create JEA files:

```
New-ModuleManifest -Path $modpath\JeaDhcpOper.psd1
New-PSRoleCapabilityFile -Path
$modpath\RoleCapabilities\JeaDhcpOper.psrc
New-PSSessionConfigurationFile -Path $jeaConfigPath\SCF_DHCPopers.pssc
```

6. Analyze module description:

```
ise $modpath\JeaDhcpOper.psd1
```

7. Modify role capability file

```
ise $modpath\RoleCapabilities\JeaDhcpOper.psrc
```

8. Add commandlets to manage DHCP Server

```
VisibleCmdlets = 'DHCPServer\*'
```

9. Add hostname and ipconfig externall commands

```
VisibleExternalCommands = 'hostname.exe', 'ipconfig.exe'
```

10. Modify session configuration file

```
ise $jeaConfigPath\SCF_DHCPopers.pssc
```

11. Adjust content of pssc file:

```
SessionType = 'RestrictedRemoteServer'
RunAsVirtualAccount = $true
```

```
    RoleDefinitions = @{ 'CQURE\DHCPopers' = @{ RoleCapabilities =
'JeaDhcpOper' }}
```

12. Register profile:

```
    Register-PSSessionConfiguration -Name `DHCPopers' -Path
$jeaConfigPath\SCF_DHCPopers.pssc
```

13. Restart WinRM:

```
    Restart-Service WinRM
```

14. Logon to **W10CLI** as CQURE\Alice with password P@ssw0rd
15. Start powershell and try to connect with PSremoting

```
    Enter-PSSession -ComputerName DC01.cqure.lab
```

16. Try the same command with profile

```
    Enter-PSSession -ComputerName DC01.cqure.lab -ConfigurationName
DHCPopers
```

17. Display all commands which are avaiable and test ping and hostname

Lab 3

1. Logon to **WS19-DC01**
2. Prepare new JEA configuration for:
    1. Read only commands for DNS
    2. Restart service but only DNS Server
    3. Allowed ping command
3. Test configuration by using **W10CLI**

References

- https://docs.microsoft.com/en-us/powershell/scripting/learn/remoting/jea/overview?view=powershell-6

8. GUARDED FABRIC AND SHIELDED VMS

Summary

To help protect against compromised virtualization fabric, Windows Server 2016 Hyper-V introduced shielded VMs. A shielded VM is a generation 2 VM (supported on Windows Server 2012 and later) that has a virtual TPM, is encrypted using BitLocker, and can run only on healthy and approved hosts in the fabric. Shielded VMs and guarded fabric enable cloud service providers or enterprise private cloud administrators to provide a more secure environment for tenant VMs.

Setup - very important

1. Shutdown all machines except **HGS, HOST-2, DC01**
2. Set or check IP configuration for:
    1. HGS: IP 192.168.0.150/24 DNS 192.168.0.10
    2. HOST-1: IP 192.168.0.151/24 DNS 192.168.0.10
    3. Check if ping is working between machines and DC
3. Check that firewalls are disabled on all machines
4. Logon to **DC1** and execute command

```
netdom trust cqure.lab /domain:bastion.cqure.lab /remove /force
```

Lab

1. Logon to **HGS**
2. Install windows feature and after restart logon again

```
Install-WindowsFeature -Name HostGuardianServiceRole -
IncludeManagementTools -Restart
```

3. Start powershell and execute script to prepare certifictes and install HGS Server with a bastion domain

```
cd c:\
md awsc
cd awsc
$secPass = ConvertTo-SecureString  -AsPlainText "Password1." -Force
$signCert = New-SelfSignedCertificate -Subject "CN=HGS Signing
Certificate"
Export-PfxCertificate -FilePath .\signCert.pfx -Password $secPass -
Cert $signCert
Remove-Item $signCert.PSPath
$encCert = New-SelfSignedCertificate -Subject "CN=HGS Encryption
Certificate"
Export-PfxCertificate -FilePath .\encCert.pfx -Password $secPass -
Cert $encCert
Remove-Item $encCert.PSPath
```

```
Install-HgsServer -HgsDomainName 'bastion.cqure.lab' -
SafeModeAdministratorPassword $secPass -Restart
```

4. Wait for the reboot to complete and sign in again
5. Open powershell legacy mode

```
$secPass = ConvertTo-SecureString  -AsPlainText "Password1." -Force
#Initialize-HgsServer -HgsServiceName 'MyHgsDNN' -
SigningCertificatePath 'c:\awsc\signCert.pfx' -SigningCertificatePassword
$secPass -EncryptionCertificatePath 'c:\awsc\encCert.pfx' -
EncryptionCertificatePassword $secPass -TrustActiveDirectory
Initialize-HgsServer -HgsServiceName 'MyHgsDNN' -
SigningCertificatePath 'c:\awsc\signCert.pfx' -SigningCertificatePassword
$secPass -EncryptionCertificatePath 'c:\awsc\encCert.pfx' -
EncryptionCertificatePassword $secPass -TrustHostkey
```

6. Create one way trust between bastion and cqure.lab domain

```
Add-DnsServerConditionalForwarderZone -Name "cqure.lab" -
ReplicationScope "Forest" -MasterServers 192.168.0.10
netdom trust bastion.cqure.lab /domain:cqure.lab /
userD:cqure.lab\Administrator /passwordD:P@ssw0rd /add
Get-ClusterNetwork
(Get-ClusterNetwork).Role = 3
Get-HgsTrace -RunDiagnostics -Detailed
```

7. Restart computer and wait for full reboot
8. Logon to the Domain Controller **DC1**
9. Open powershell and add DNS zone and group membership (deprecated mode)

```
Add-DnsServerZoneDelegation -Name "cqure.lab" -ChildZoneName "bastion"
-NameServer 192.168.0.150  -IPAddress 192.168.0.150
New-ADGroup -Name GuardedHosts -GroupScope Global -GroupCategory
Security
Add-ADGroupMember -Identity GuardedHosts -members (Get-ADComputer -
Identity Host-2)
Get-ADGroupMember -Identity GuardedHosts
```

10. Logon to **HOST-2** and open powershell
11. Install Windows feature and after reboot logon again

```
Install-WindowsFeature Hyper-V, HostGuardian -IncludeManagementTools -
Restart
```

12. Logon to **HOST-2** and open powershell

```
$tpmBoundCert = New-SelfSignedCertificate -Subject "Host Key
Attestation ($env:computername)" -Provider "Microsoft Platform Crypto
Provider"
Set-HgsClientHostKey -Thumbprint $tpmBoundCert.Thumbprint
Get-HgsClientHostKey -Path "C:\awsc\HostKey.cer"
```

13. Copy file HostKey.cer to HGS server \192.168.0.150\c$\awsc
14. Logon to **HGS** and add key to guarded hosts

```
#Add-HGSAttestationHostGroup -HostGroup GuardedHosts -Name GuardedHosts
Add-HgsAttestationHostKey -Name MyHost01 -Path "C:\awsc\HostKey.cer"
Get-HgsTrace -RunDiagnostics -Detailed
```

15. Logon to **HOST-2**
16. Test name resolution

```
nslookup myhgsdnn.bastion.cqure.lab
Set-HgsClientConfiguration -AttestationServerUrl 'http://
MyHGSDNN.bastion.cqure.lab/Attestation' -KeyProtectionServerUrl 'http://
MyHGSDNN.bastion.cqure.lab/KeyProtection'
Get-HgsClientConfiguration
Get-HgsTrace -RunDiagnostics -Detailed
```

17. Verify output

This lab cannot be successfully completed in virtualized environment.

References

- https://docs.microsoft.com/en-us/windows-server/security/guarded-fabric-shielded-vm/guarded-fabric-and-shielded-vms

9. WINDOWS SUSBSYSTEM FOR LINUX

Summary

Lab 1

Installation

1. Connect to HOST-0
2. Enable WSL feature:

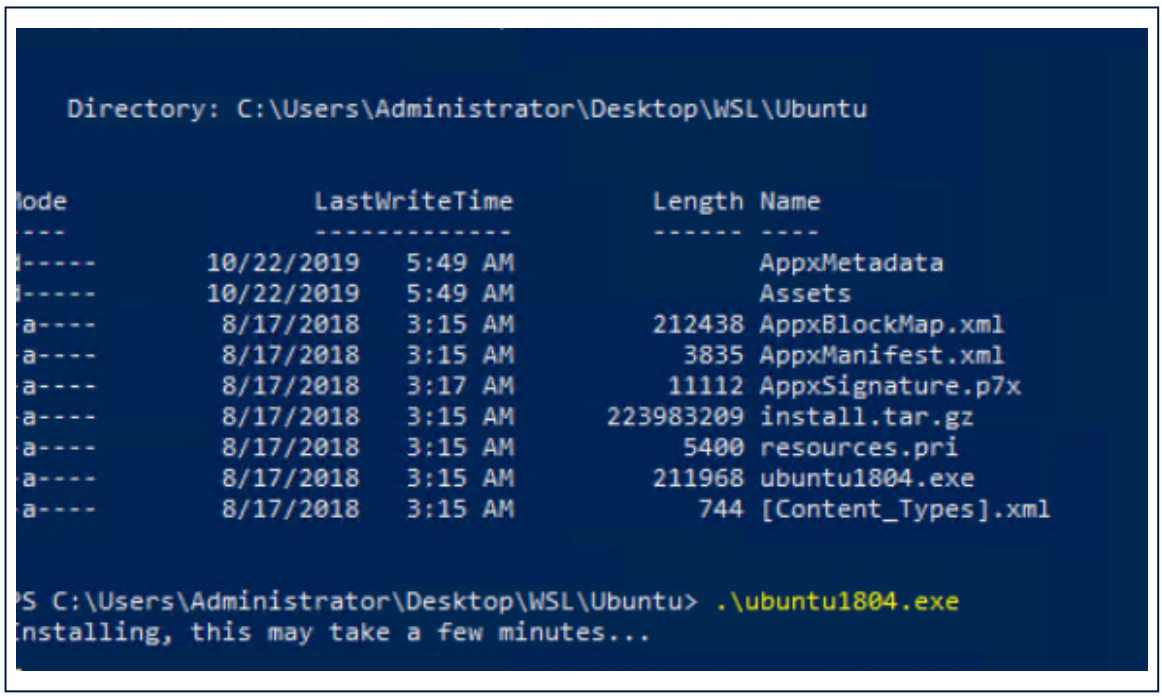
```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
```

3. Restart the machine
4. Download your distrubution from MS (more info: https://docs.microsoft.com/en-us/windows/wsl/install-manual):

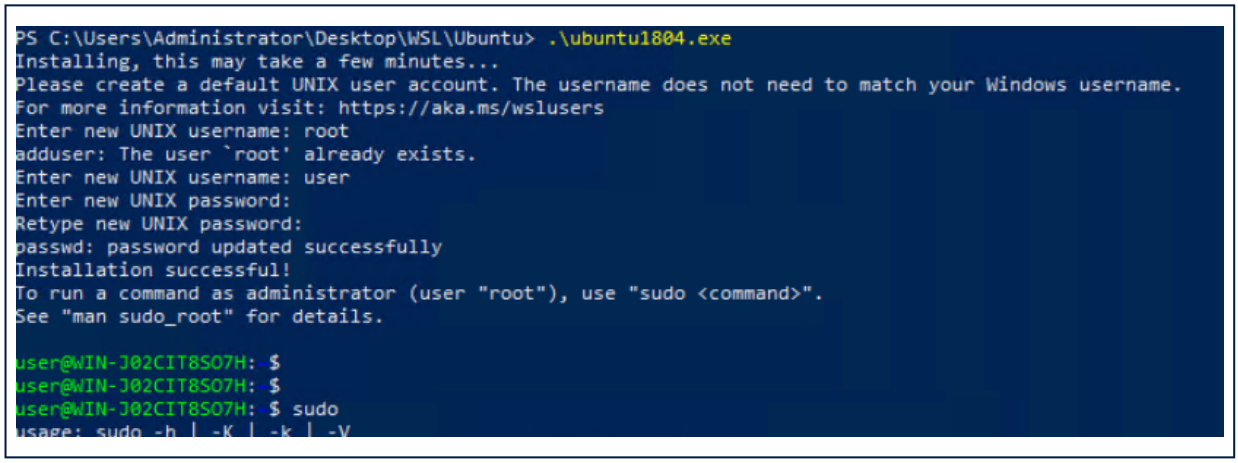
```
cd C:\Users\Administrator\Desktop\WSL
Import-Module BitsTransfer
Start-BitsTransfer -Source https://aka.ms/wsl-ubuntu-1804 -Destination Ubuntu.appx
```

5. Extract package:

```
Rename-Item ./Ubuntu.appx ./Ubuntu.zip
Expand-Archive ./Ubuntu.zip ./Ubuntu
cd Ubuntu
.\ubuntu1804.exe
```



6. During installation provide credentials username: cqure password: P@ssw0rd



1. Exit from ubuntu `exit`

Using WSL, interoperability between Windows and Linux instance

1. To run WSL in interactive mode, execute:

```
wsl
```

2. Show path - you get access to all local files:

```
pwd
```

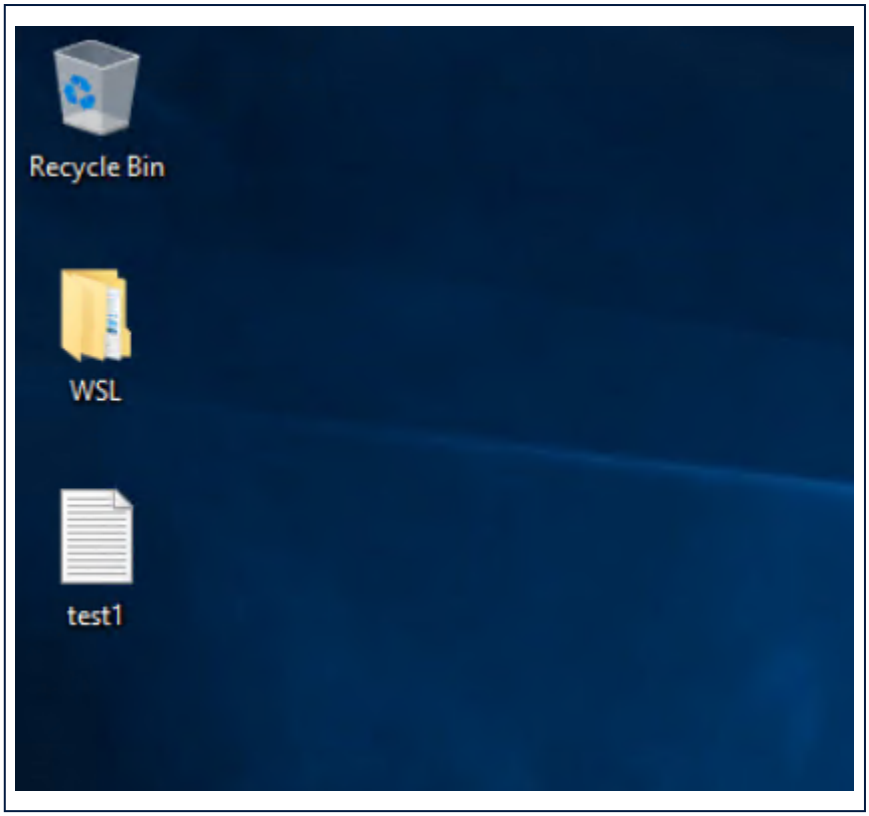
3. Go to desktop folder:

```
cd /mnt/c/Users/Administrator/Desktop
```

4. Create new file:

```
touch test1.txt
```

5. The file was created on desktop:



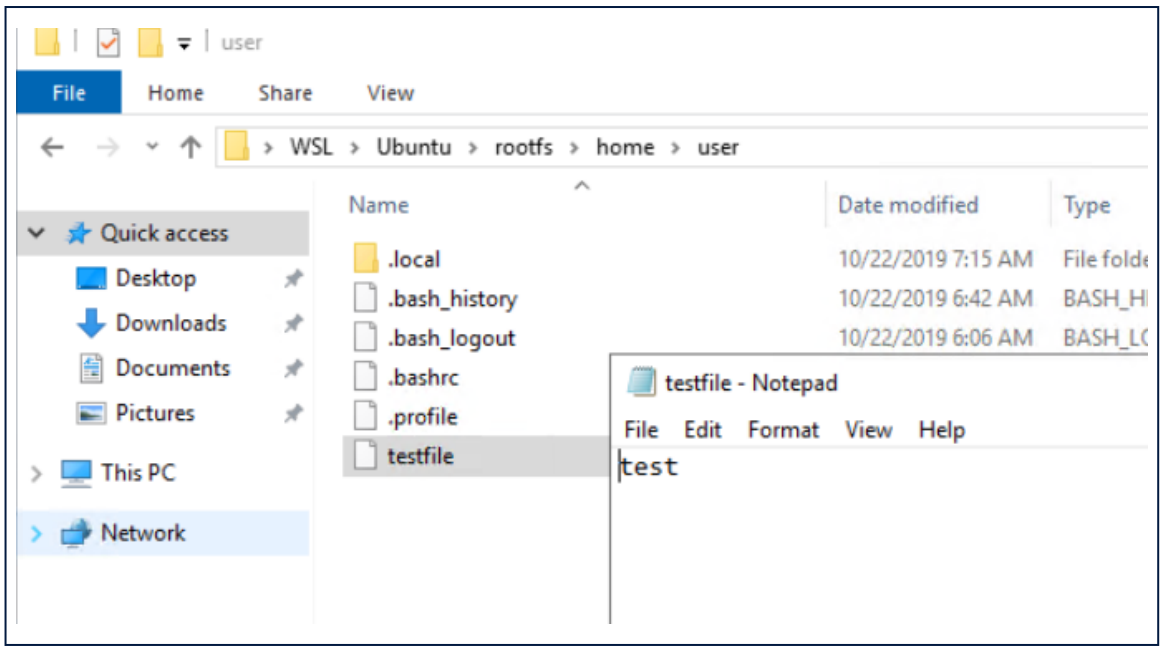
6. Go to home folder (use <space> if ~ sign is not working):

```
cd ~
```

7. Run nano editor

```
nano test2.txt
```

8. Write some text, save file and exit : **Ctrl +x**
9. Exit from WSL
10. Goto the following path in windows explorer to see that the file was created: `C:\Users\Administrator\Desktop\WSL\Ubuntu\rootfs\home\user`



Working with WSL

1. List distributions:

```
wslconfig /l
```

2. To run chosen distribution:

```
wsl -d Ubuntu-18.04
exit
```

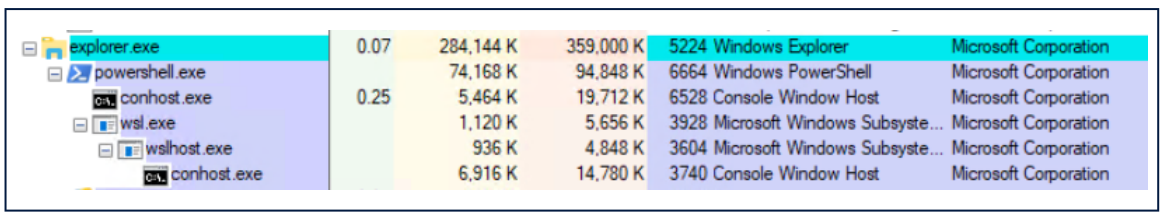
3. To run ifconfig command with wsl:

```
wsl ifconfig
```

4. Run calculator from wsl and htop:

```
wsl
/mnt/c/Windows/System32/calc.exe
htop
```

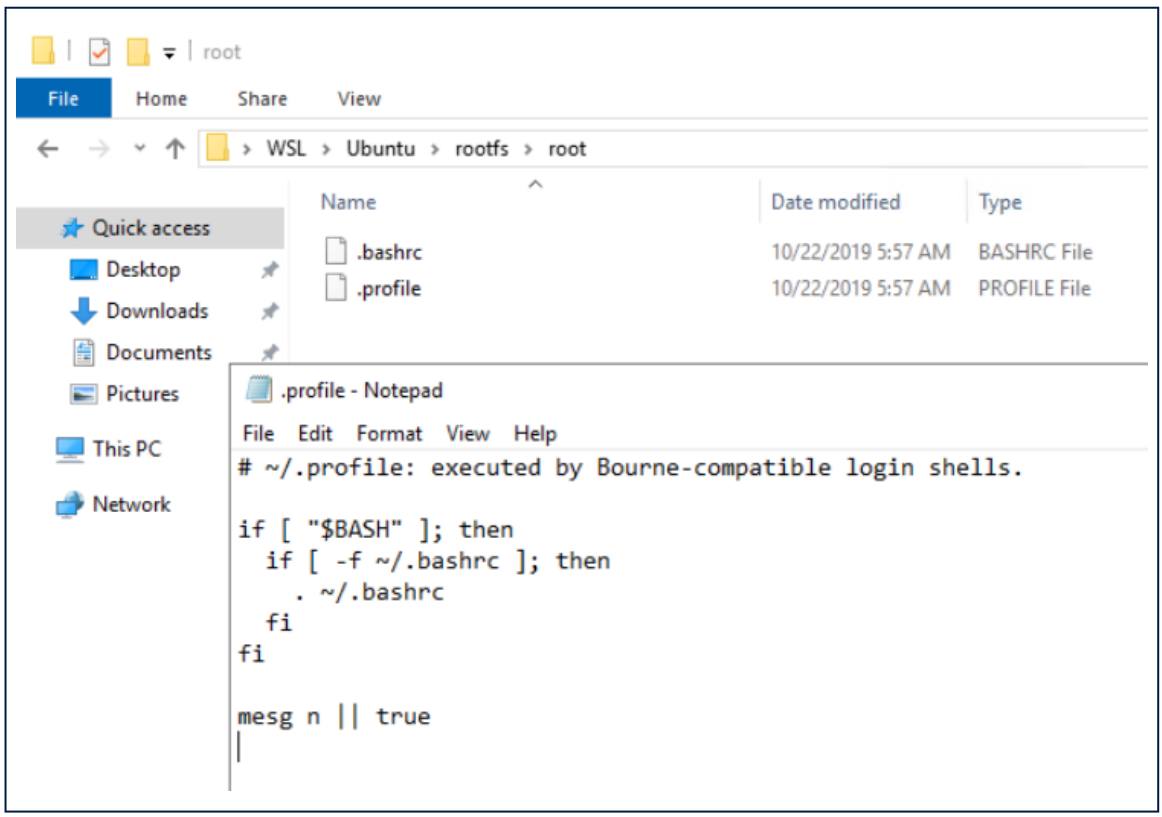
5. Find the wsl, win32calc.exe and htop processes in Process Explorer("C:\Users\Administrator\Desktop\SysinternalsSuite\procexp64.exe"):



6. Click **F10** to quit htop in WSL
7. Try to access /root in WSL - not possible without root permissions:

```
cd /root
```

8. From Windows as Admin you are able to get access to the files without additional privileges (C:\Users\Administrator\Desktop\WSL\Ubuntu\rootfs\root):



Updates and software installation:

1. Ensure to update your WSL distribution - without it it may be vulnerable, to update the Ubuntu Linux, use the following command:

```
wsl sudo apt update
```

Warning! `wsl sudo apt dist-upgrade` (it takes a long time so you don't need to run it in the labs)

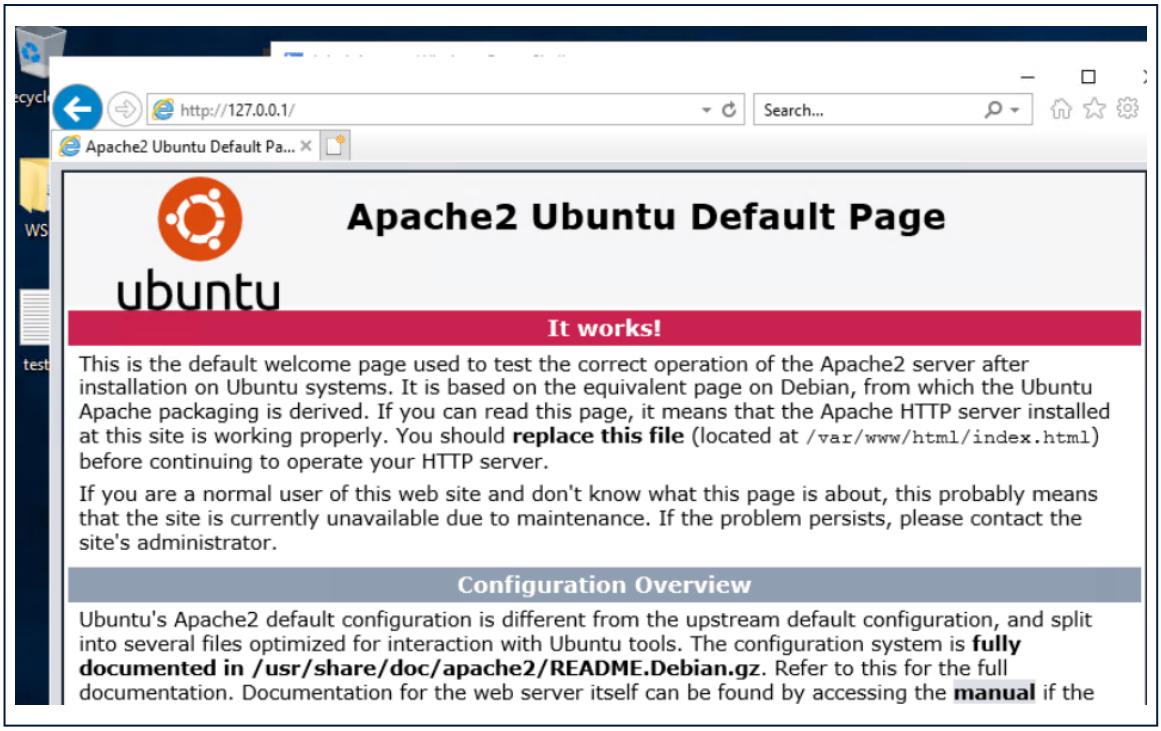
1. Install Apache and g++:

```
wsl sudo apt install apache2 g++
```

2. To start apache, run and use password: P@ssw0rd (an error may occur, but it may be ignored)

```
wsl sudo apachectl start
```

3. Open browser: http://127.0.0.1



4. Show netstat in WSL - no info is available - not all system calls are implemented - they will be implemented in WSL2 - planned date: spring 2020 in Windows 10:

```
wsl sudo netstat -antlp
```

5. In Windows netstat, the port 80 with WSL Apache is open:

```
netstat -a
```

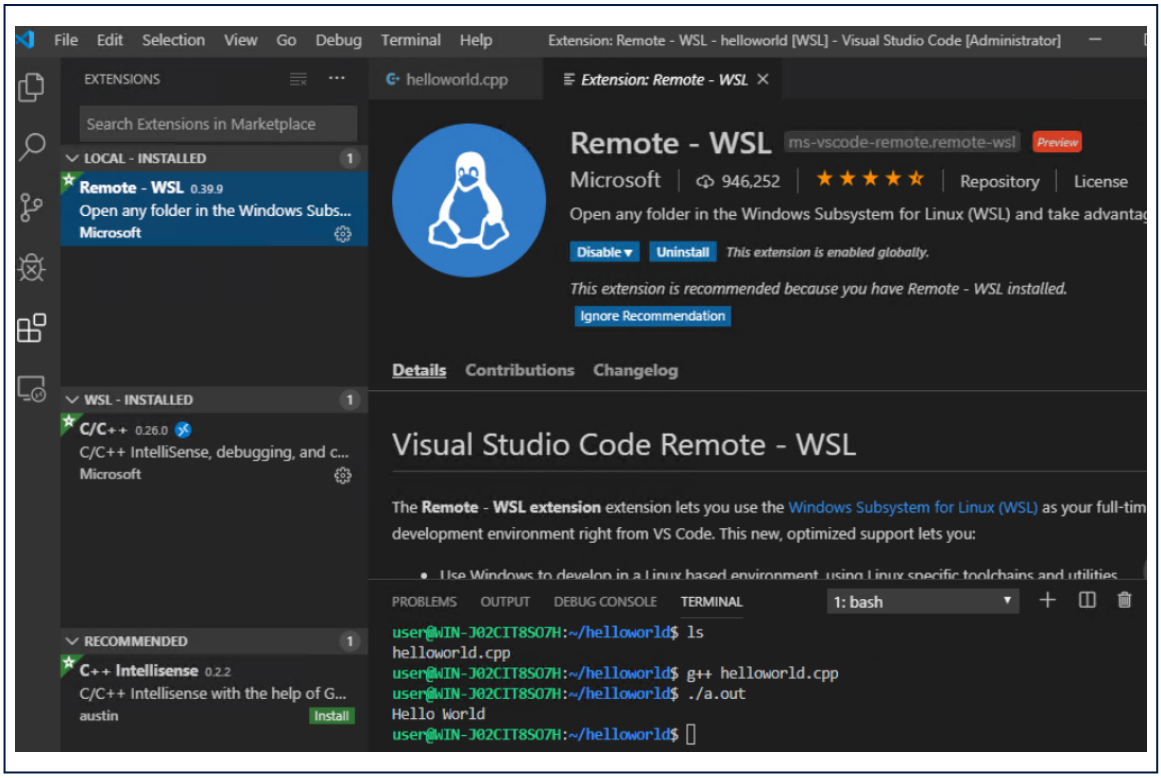
```
PS C:\Users\Administrator\Desktop\WSL\Ubuntu\rootfs\home\user\helloworld> netstat -a

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:80             WIN-J02CIT8SO7H:0      LISTENING
  TCP    0.0.0.0:135            WIN-J02CIT8SO7H:0      LISTENING
  TCP    0.0.0.0:445            WIN-J02CIT8SO7H:0      LISTENING
  TCP    0.0.0.0:5985           WIN-J02CIT8SO7H:0      LISTENING
  TCP    0.0.0.0:47001          WIN-J02CIT8SO7H:0      LISTENING
  TCP    0.0.0.0:49664          WIN-J02CIT8SO7H:0      LISTENING
  TCP    0.0.0.0:49665          WIN-J02CIT8SO7H:0      LISTENING
  TCP    0.0.0.0:49666          WIN-J02CIT8SO7H:0      LISTENING
  TCP    0.0.0.0:49667          WIN-J02CIT8SO7H:0      LISTENING
  TCP    0.0.0.0:49668          WIN-J02CIT8SO7H:0      LISTENING
  TCP    0.0.0.0:49669          WIN-J02CIT8SO7H:0      LISTENING
  TCP    0.0.0.0:49684          WIN-J02CIT8SO7H:0      LISTENING
  TCP    0.0.0.0:50366          WIN-J02CIT8SO7H:0      LISTENING
  TCP    127.0.0.1:80           WIN-J02CIT8SO7H:50438  TIME_WAIT
  TCP    127.0.0.1:50366        WIN-J02CIT8SO7H:50368  ESTABLISHED
  TCP    127.0.0.1:50366        WIN-J02CIT8SO7H:50369  ESTABLISHED
  TCP    127.0.0.1:50368        WIN-J02CIT8SO7H:50366  ESTABLISHED
  TCP    127.0.0.1:50369        WIN-J02CIT8SO7H:50366  ESTABLISHED
  TCP    169.254.123.175:139    WIN-J02CIT8SO7H:0      LISTENING
  TCP    192.168.251.182:139    WIN-J02CIT8SO7H:0      LISTENING
  TCP    192.168.251.182:50421  yukinko:http           CLOSE_WAIT
  TCP    [::]:80                WIN-J02CIT8SO7H:0      LISTENING
  TCP    [::]:135               WIN-J02CIT8SO7H:0      LISTENING
  TCP    [::]:445               WIN-J02CIT8SO7H:0      LISTENING
  TCP    [::]:5985              WIN-J02CIT8SO7H:0      LISTENING
  TCP    [::]:47001             WIN-J02CIT8SO7H:0      LISTENING
  TCP    [::]:49664             WIN-J02CIT8SO7H:0      LISTENING
  TCP    [::]:49665             WIN-J02CIT8SO7H:0      LISTENING
  TCP    [::]:49666             WIN-J02CIT8SO7H:0      LISTENING
  TCP    [::]:49667             WIN-J02CIT8SO7H:0      LISTENING
  TCP    [::]:49668             WIN-J02CIT8SO7H:0      LISTENING
  TCP    [::]:49669             WIN-J02CIT8SO7H:0      LISTENING
  TCP    [::]:49684             WIN-J02CIT8SO7H:0      LISTENING
  TCP    [::]:50366             WIN-J02CIT8SO7H:0      LISTENING
  UDP    0.0.0.0:123            *:*
```

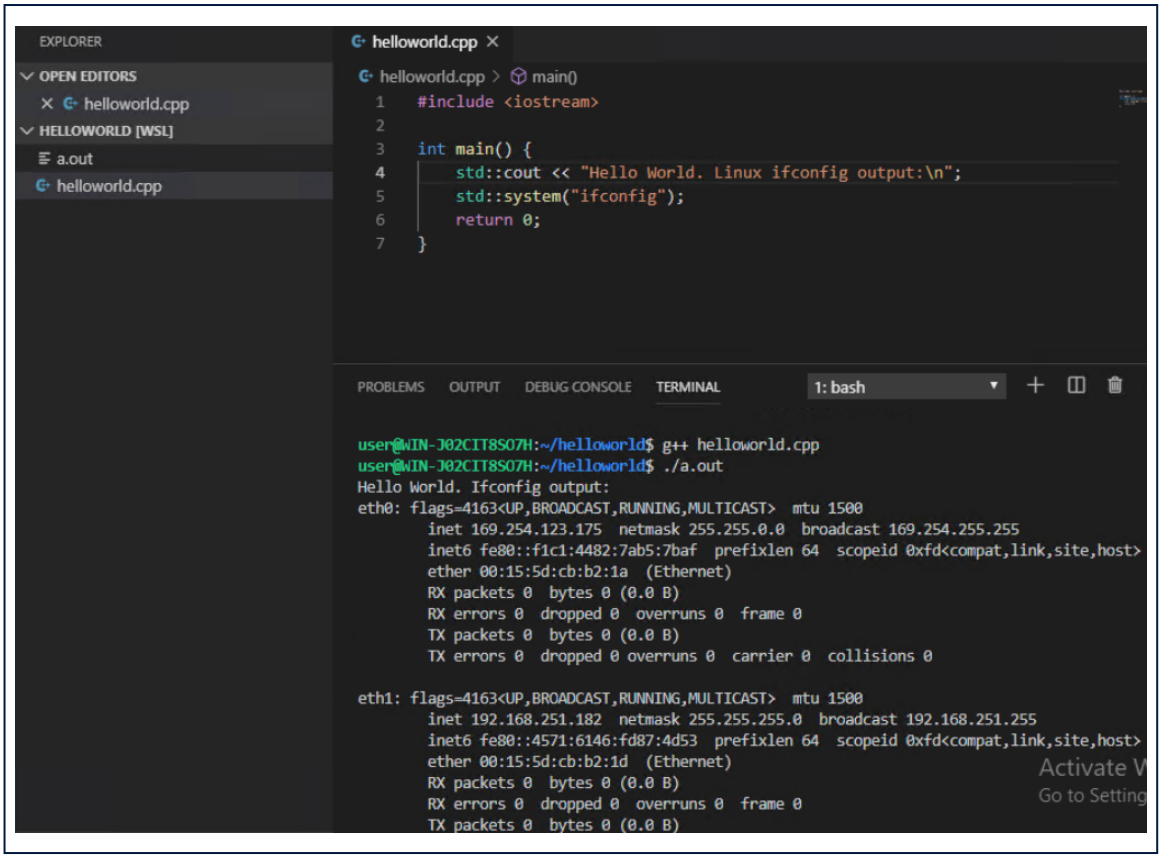
WSL in Visual Studio Code:

1. Run Visual Studio Code
2. Go to extenstions
3. Check if WSL support is installed



4. Click: View -> Command palette -> Remote-WSL:New Window
5. Open file: File -> Open folder -> /mnt/c/Users/Administrator/Desktop/WSL/HelloWorld/
6. Click Terminal -> New Terminal
7. In terminal compile the code and run it:

```
g++ HelloWorld.cpp
./a.out
```



Lab2 - WSL vs Sysmon

1. Install Sysmon
2. Download config for sysmon: https://github.com/SwiftOnSecurity/sysmon-config
3. Update sysmon configuration and test if you see programs started in windows
4. Try to start calc from WSL
5. Try to start some linux based programs in WSL
6. Modify some files in Windows
7. Modify windows files in WSL
8. Verify which action you see which are not registered by Sysmon

References

- https://docs.microsoft.com/en-us/windows/wsl