

שלב א' - הורדת קוברנטיס וחיבור מכונות

קודם נבין כמה מושגים :

- node** - הינו יחידת חישוב המריצה את רכיבי Kubernetes הנדרשים ומאחסנת את ה-pods.
- cluster** - אוסף של כמה nodes שעובדים יחד לעבר מטרה מסוימת.
- Kubeadm** - בעזרתו ניתן לייצר את ה cluster ולצרף את worker nodes.
- Kubctl** - מאפשר לשלוח פקודות ל Api server ולנהל את cluster.
- Kubelet** - רץ על כל נוד ותפקידו לוודא שכל ה-pods שצריכים לרוץ - באמת רצים.
- Kubeconfig** - זהו קובץ שמכין פרטי חיבור לcluster, כולל כתובת api server ותעודת אבטחה. בעזרתו kubectl יודע איך לגשת לcluster.
- Pod network** - פתרון רשת שמאפשר תקשורת בין ה-pods בכל node בcluster.
- Kernel** - הליבה של מערכת ההפעלה, מנהלת את החומרה והמשאבים עבור כל התכונות שרצות.
- Cri-dockerd** - תוסף שמאפשר לdocker לדבר עם kubelet.

התבקשנו ליצור בסך הכל 4 מכונות וירטואליות:

הראשונה למנהל (master)

השנייה והשלישית לnodes (workers)

והרביעית - החיצונית (manager)

אנו צריכים להשתמש בפורמט שנקרא: vagrant.

זהו כלי שיכול לייצר או להרוס באופן אוטומטי דרך סקריפט **vagrantfile** מכונות וירטואליות שונות.

על כן, נתקין vagrant על המחשב הפיזי (על windows) ונעשה כמה צעדים :

ניכנס לאתר <https://developer.hashicorp.com/vagrant/install#windows> ונלחץ על התקנה :

Windows

Binary download

AMD64
Version: 2.4.7

[Download](#) 📄

- ניכנס ל-vagrantfile ונרשום חוקים למכונות הווירטואליות שאנו רוצים ליצור (הנה ההרכב למשל של master) :

Vagrant.configure("2") do |config| ----> opening line in **vagrantfile**

Config.vm.define "master" do |master| -----> define the name of the vm as "master"

Master.vm.box= "ubuntu/jammy64" -----> define the box as the base to the vm

Master.vm.hostname = "master" -----> define the name to the computer in the vm

Master.vm.provider "virtualbox" do |vb|-----→ to put the machines in VMware
virtualBox

Vb.memory = 4096 -----→ RAM size

Vb.cpus = 2 -----→ CPU amount

End

Master.vm.network "private_network", ip "192.168.56.10"-----→ define each vm it's
own private network so they can talk to each other.

End

לכל מכונה וירטואלית קבענו :

- מה שמה
- איפה היא תיווצר
- מה כמות הזיכרון בram.
- מה מספר הcpu
- ורשת פרטית ייחודית שבזכותה היא תוכל לדבר אם שאר המכונות הוירטואליות שניצור.

נרשום את הפקודה הבאה בשביל להרים את vagrant file : **vagrant up**

אחרי שהמכונות נוצרו, ניכנס למכונה "master" דרך הפקודה **vagrant ssh master**.

שניכנס למכונה הוירטואלית דרך power shell :

Vagrant up

Vagrant ssh master

יש להתקין על כל המכונות docker-engine הנה הפקודות :

```
for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker  
containerd runc; do sudo apt-get remove $pkg; done
```

```
sudo apt-get update
```

```
sudo apt-get install ca-certificates curl
```

```
sudo install -m 0755 -d /etc/apt/keyrings
```

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o  
/etc/apt/keyrings/docker.asc
```

```
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

```
echo \
```

```
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]  
https://download.docker.com/linux/ubuntu \
```

```
$(. /etc/os-release && echo "${UBUNTU_CODENAME:-  
$VERSION_CODENAME}") stable" \
```

```
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt-get update
```

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
sudo docker run hello-world
```

לאחר מכן, צריך להתקין את cri-dockerd על ארבעת המכונות על ידי השלבים פה:

קודם נבין,

מדוע צריך את cri-dockerd?

כלי זה הוא פתרון המאפשר לקוברנטיס להשתמש בדוקר כrunners אך בצורה שתואמת את ה cri של הקוברנטיס. הוא כמו מתווך בין הדוקר לקוברנטיס.

הנה הפקודות להורדת הכלי:

```
sudo apt-get install git-all
git clone https://github.com/Mirantis/cri-dockerd.git
wget https://dl.google.com/go/go1.24.4.linux-amd64.tar.gz
sudo tar -C /usr/local -xvzf go1.24.4.linux-amd64.tar.gz
echo "export PATH=$PATH:/usr/local/go/bin" >> ~/.bashrc
source ~/.bashrc
go version
cd cri-dockerd
mkdir bin
go build -o bin/cri-dockerd
sudo mkdir -p /usr/local/bin
sudo install -o root -g root -m 0755 bin/cri-dockerd /usr/local/bin/cri-dockerd
sudo cp -a packaging/systemd/* /etc/systemd/system
sudo sed -i -e 's,/usr/bin/cri-dockerd,/usr/local/bin/cri-dockerd,'
/etc/systemd/system/cri-docker.service
sudo systemctl daemon-reload
sudo systemctl enable cri-docker.service
sudo systemctl start cri-docker.service
sudo systemctl enable --now cri-docker.socket
```

בשביל לבדוק אם cri-dockerd פעיל כותבים את הפקודה הבאה:

```
sudo systemctl status cri-docker.service
```

לאחר הורדות ה cri-dockerd ניתן להוריד קוברנטיס (יש להתקין את זה בשלושת המכונות במכונה הרביעית אין צורך)

הנה הפקודות להורדת kubdeam, kubelet:

```

sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
sudo mkdir -p -m 755 /etc/apt/keyrings
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.33/deb/Release.key | sudo gpg --
dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.33/deb/ /' | sudo tee
/etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubelet kubeadm
sudo apt-mark hold kubelet kubeadm
sudo systemctl enable --now kubelet

```

שלבים לחיבור בין כל המכונות:

קודם כל, מוודים ש-kubeadm ו-kubelet מותקנים ב-master.

בתוך master מבצעים את הפעולות הבאות:

1. הורדת ה-swap : `sudo swapoff -a`
2. יצירת הקלאסטר דרך `kubeadm` במסטר נוד:


```

sudo kubeadm init --pod-network-cidr=192.168.56.10/24 --cri-socket
unix:///var/run/cri-dockerd.sock

```

הסבר על הפקודה – שמים לב שהטווח של ה ip זה הטווח של ה ip של כל המכונות שאנו רוצים לחבר אחד לשני, ושבחרים את ה cli שדרכו המכונות יתחברו.
3. יש להריץ את הפקודה הבאה בשביל להעתיק את קובץ ההגדרות להתחברות למכונה הרביעית (המכונה החיצונית) והיא מכילה את כל פרטי ההתחברות כדי שהמכונה החיצונית תוכל להתחבר ולנהל את הקלאסטר של קוברנטיס דרך api של master.

הנה הפקודה:

```

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
4. הורדת ה `kubctl` במכונה הרביעית (ששמה `manager`) דרך התקנה של `kubctl`

מ-snap:

```

sudo snap install kubectl --classic

```
5. העברת קובץ `etc/kubernetes/admin.conf` ממכונה master על פי הפקודה הבאה:
6. `scp vagrant@192.168.56.10:/home/vagrant/.kube/config ~/.kube/config`
7. התקנת `calico` בתוך `manager` בשביל לאפשר לפודים לדבר אחד עם השני:
8. הנה הפקודה:


```

kubectl apply -f
https://docs.projectcalico.org/manifests/calico.yaml

```
9. ואז לחכות דקה ולהריץ: `kubectl get nodes`
10. אחרי הרצת הפקודה ב-`manager`, עלינו לקשר גם את `worker1` ואת `worker2` דרך `join`

`kubeadm join`, קודם כל נוודא מספר צעדים:

האם `kubeadm`, `kubectl` מותקנים על 2 המכונות?

האם התקנו `cri-dockerd` ודוקר כמו שצריך והם פעילים?

האם הסרנו `swap` (sudo swapoff -a) swap

האם הפעלנו את כל הכלים:

```

sudo systemctl enable docker
sudo systemctl start docker
sudo systemctl enable cri-docker.service
sudo systemctl enable --now cri-docker.socket

```

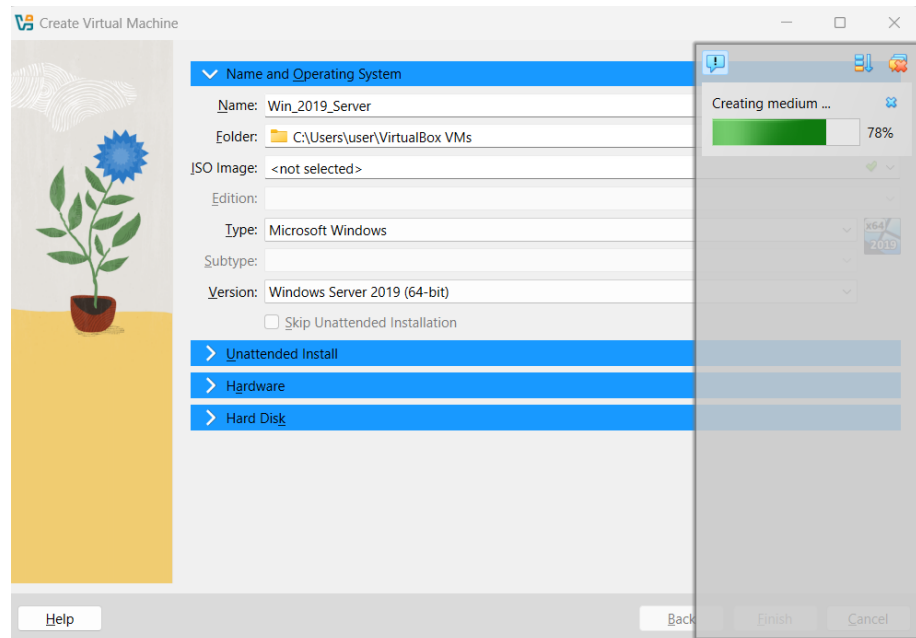
sudo systemctl start cri-docker.service
 -האם הפעלנו את forwarding ip) sudo sysctl net.ipv4.ip_forward=1
 ?(net.bridge.bridge-nf-call-iptables=1
 -האם הורדנו את חומת האש? sudo systemctl stop ufw sudo systemctl disable
 ?(ufw
 -האם אחרי שעשינו "kubeadm join" הkubeteln עובד כמו שצריך?
 11. ניגשים למכונת manager ובודקים אם אכן כל הnodes במצב ready והסטטוס של כל
 הפודים במצב "ready", המשימה הושלמה!

```

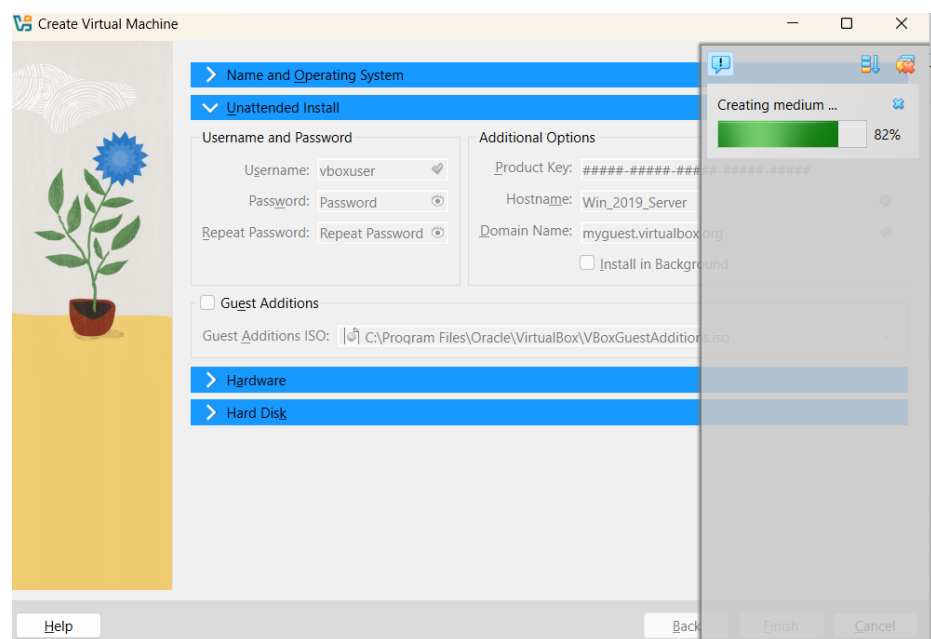
^Cvagrant@manager:~$ kubectl get nodes
NAME          STATUS    ROLES          AGE      VERSION
master        Ready     control-plane   89m      v1.33.2
worker1       Ready     <none>          54m      v1.33.2
worker2       Ready     <none>          9m31s    v1.33.2
  
```

שלב ב'- הורדת windows server 2019 והטמעה של כלים בפנים.

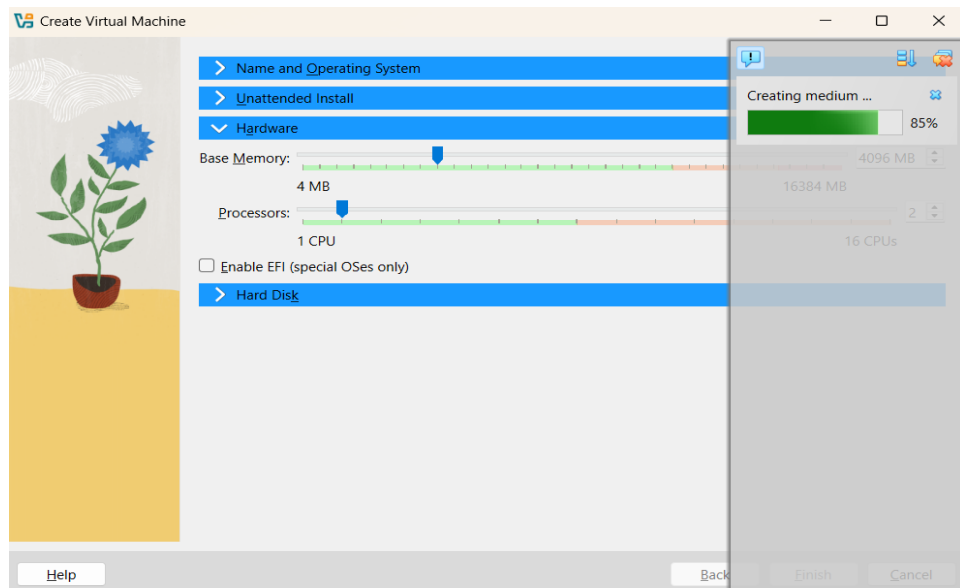
- נכנסים לעמוד הרשמי של Microsoft ומורידים את קובץ הISO שמתאים לגרסה של ווינדוס שאנו רוצים להתקין.
 - נכנסים ל VirtualBox ופותחים מכונה וירטואלית של windows 2019 :
- שלב א'- בחירת שם למכונה הווירטואלית , הגדרת סוג המערכת והתאמת גרסה של windows.



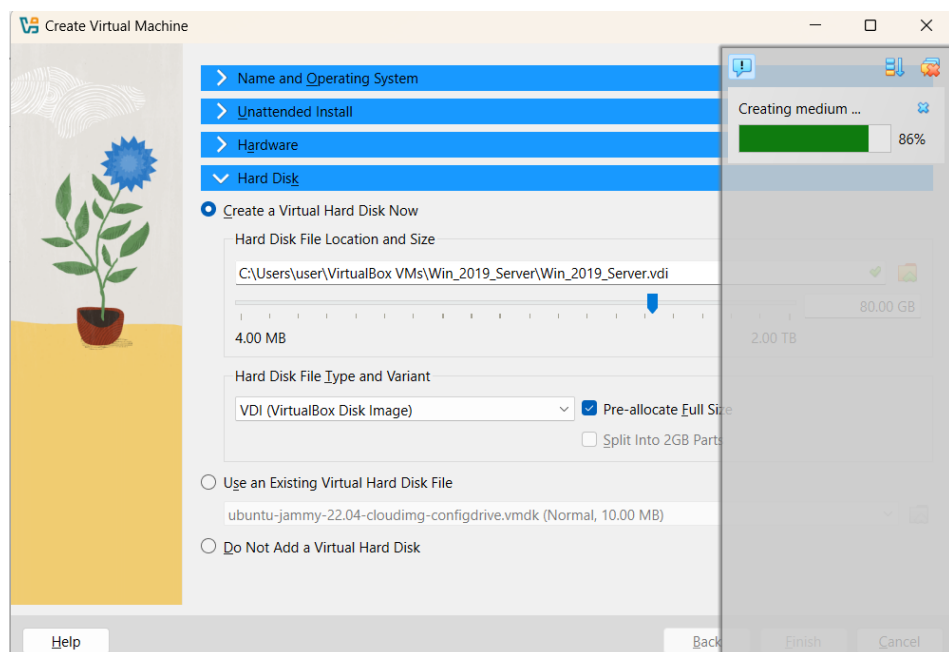
שלב ב'- פה משאירים דיפולטיבי, החלונית "unattended install" מקנה אפשרות להגדיר מראש פרטים כמו שם משתמש או סיסמא, זה דרך לייעל את התהליך ולחסוך התערבות ידנית במהלך תהליכים בהתקנה, אנו בחרנו לעשות את פעולה זו באופן ידני.



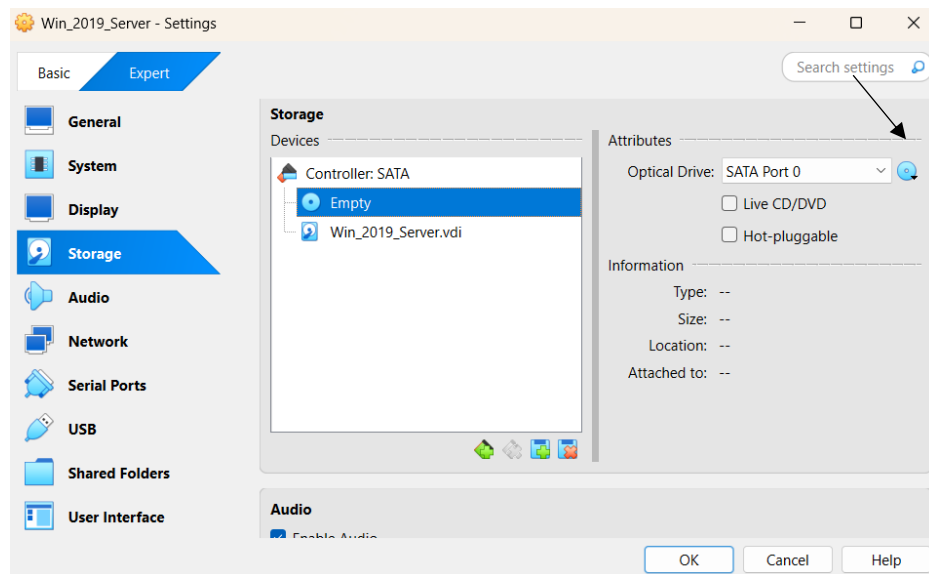
שלב ג'-הגדרת זיכרון בסיסי וליבות מעבד (RAM,CPU)



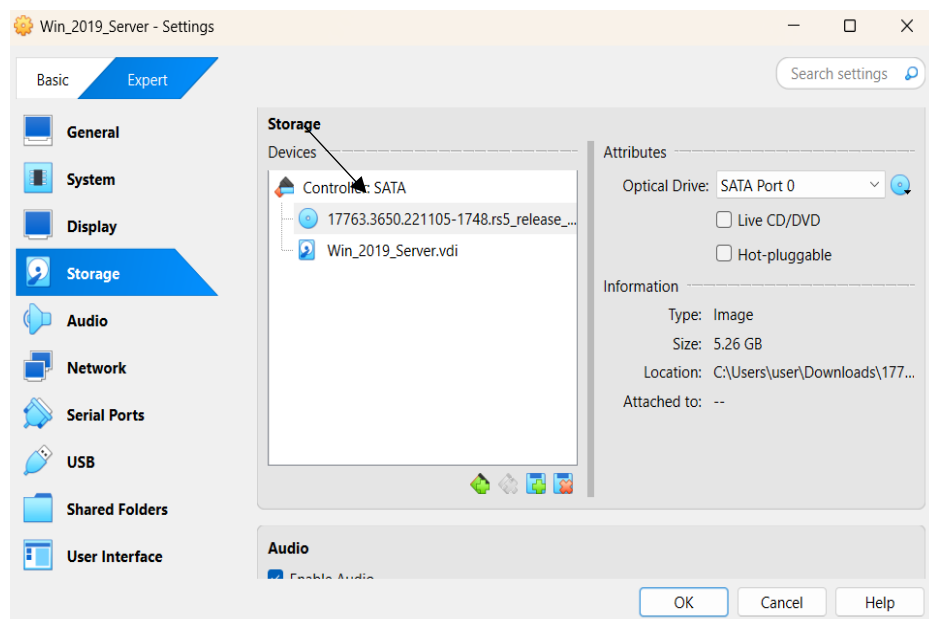
שלב ד'- יצירת דיסק קשיח וירטואלי חדש מקנה סביבה נקייה ומבודדת עבור ההתקנה, גודל הדיסק צריך להיות מספיק כדי להכיל את מערכת ההפעלה ושירותים נוספים בהתקנה. בדרך כלל (GB 80)



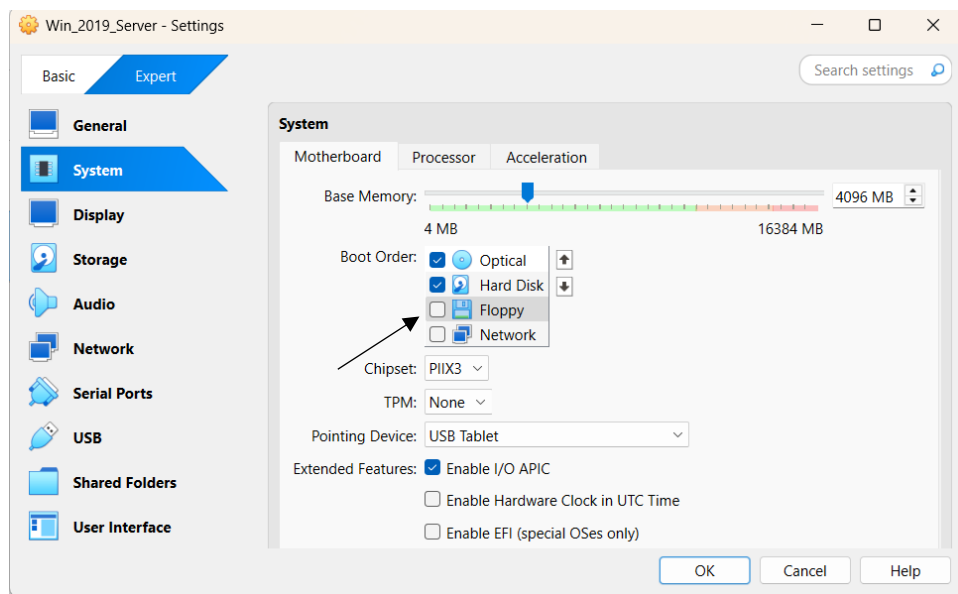
- אחרי שיצרנו את המכונה הווירטואלית, נכנסים להגדרות של המכונה--> לשונית STORAGE:



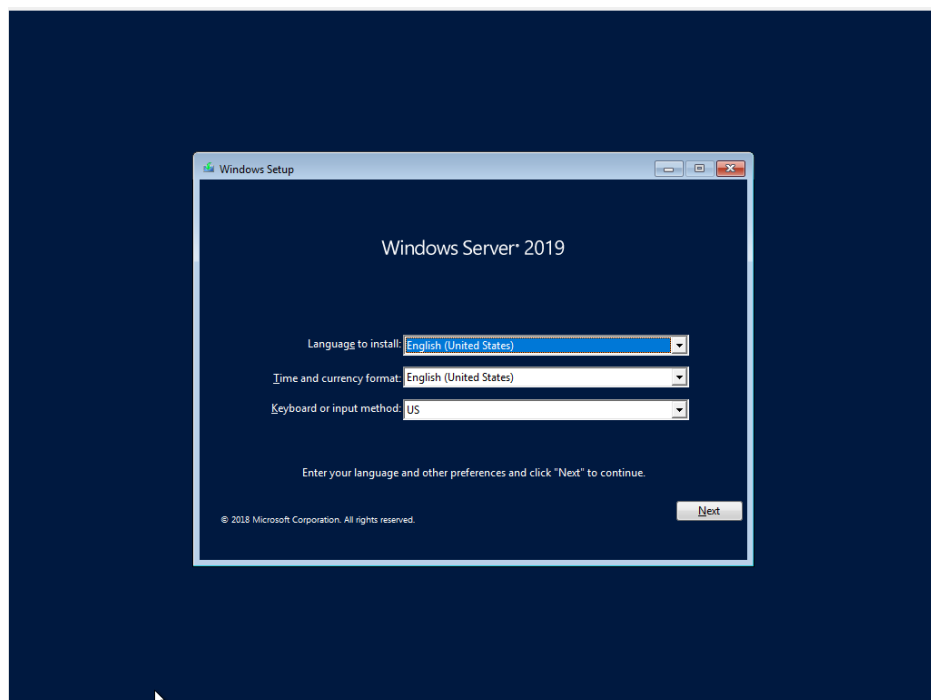
- ואז לוחצים על הסימון של הדיסק, ובחרים את קובץ ISO שהתקנו.



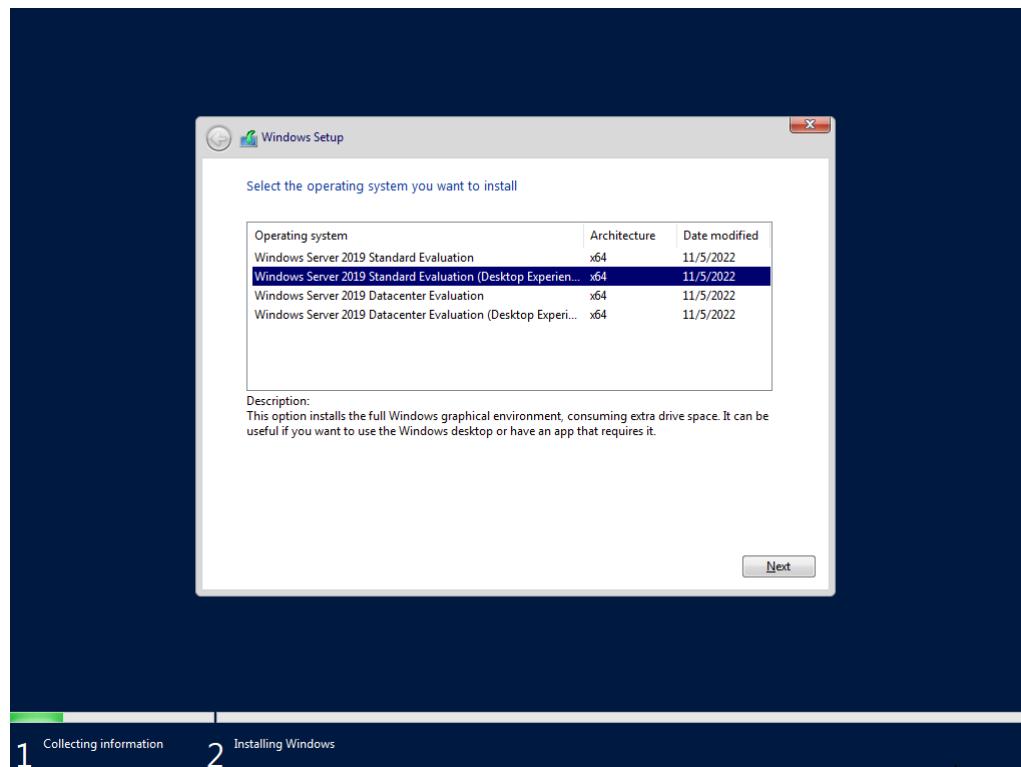
- הזזנו את floppy דיסק למטה כי הוא כלי ישן ולא משתמשים בו כבר.



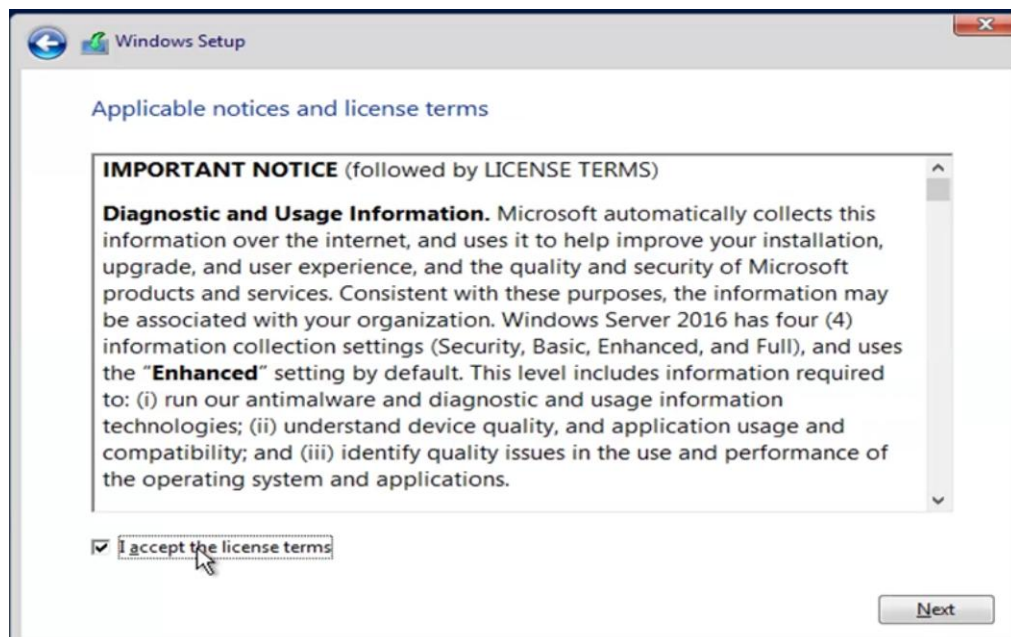
- נכנסים למכונה שהתקנו.



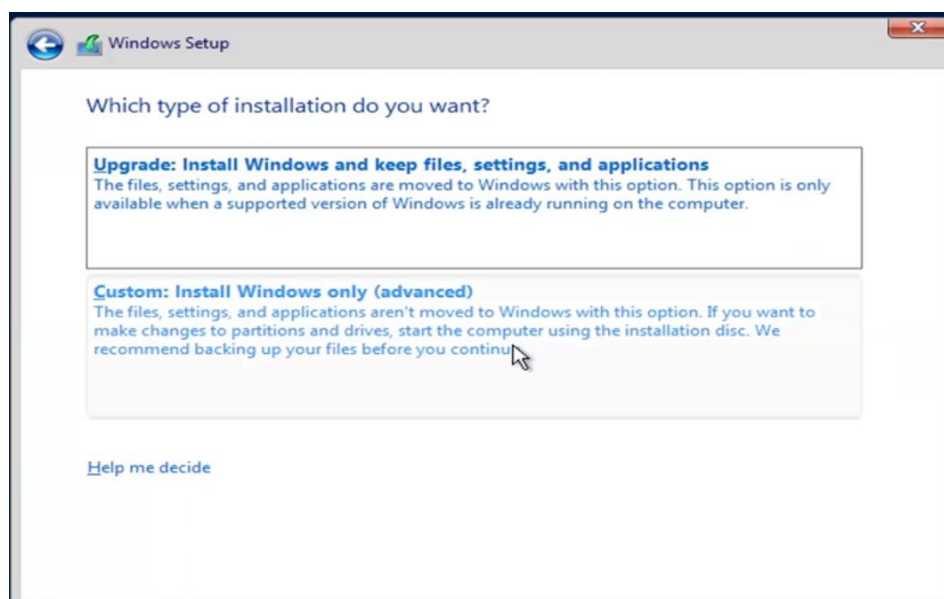
- בוחרים את האופציה השנייה לצורך תצוגת Desktop , תפריט התחלה ודפדפן אינטרנט לתצוגה נוחה יותר.



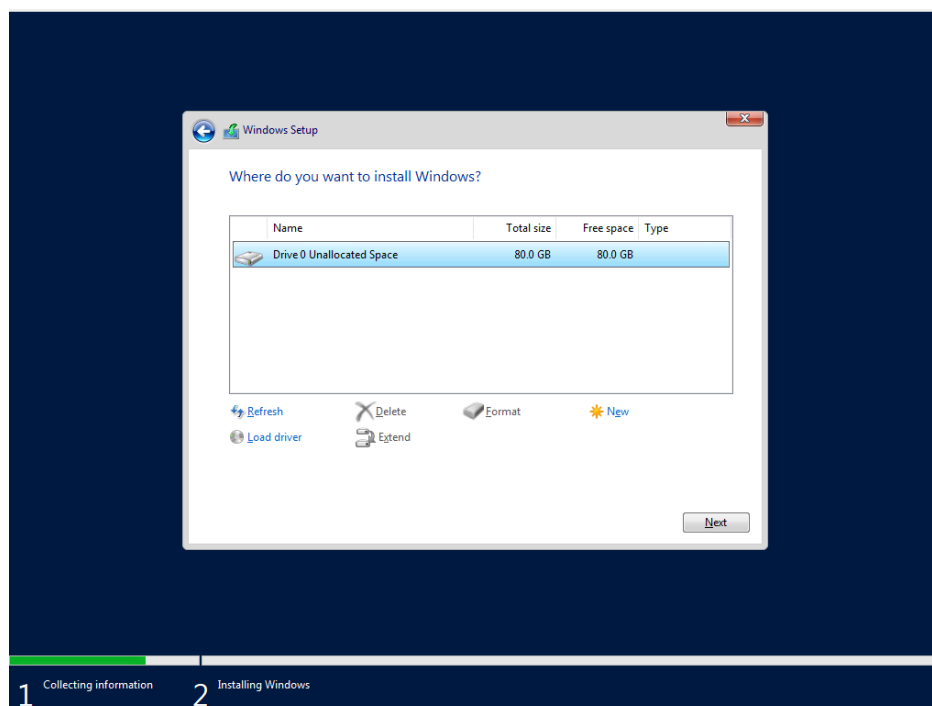
- מאשרים שקראנו את התנאים והחוקים של השימוש במערכת ההפעלה.



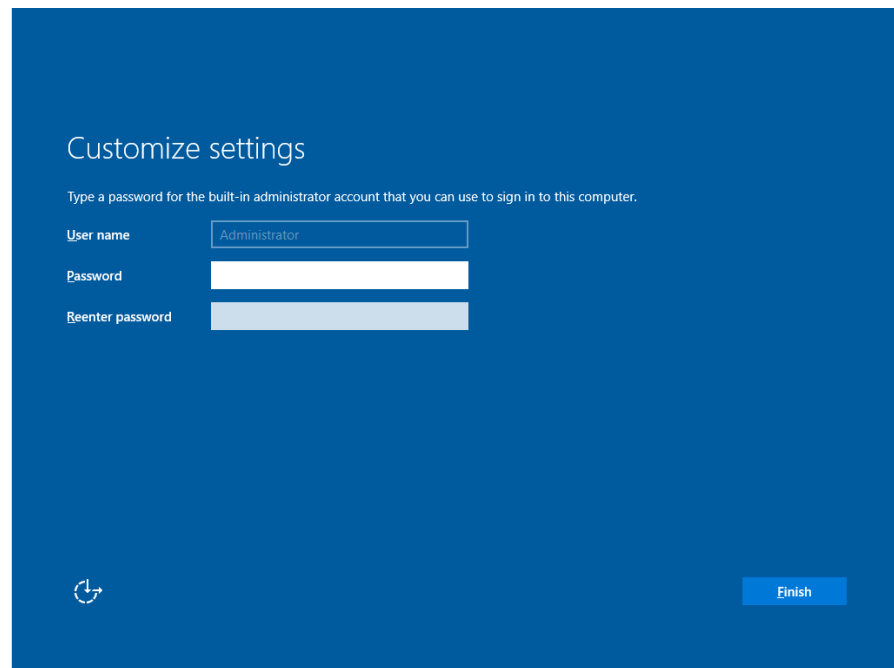
- לוחצים על custom בשביל שיבוצע התקנה נקייה של מערכת ההפעלה, ללא שדרוג של מערכת קיימת.



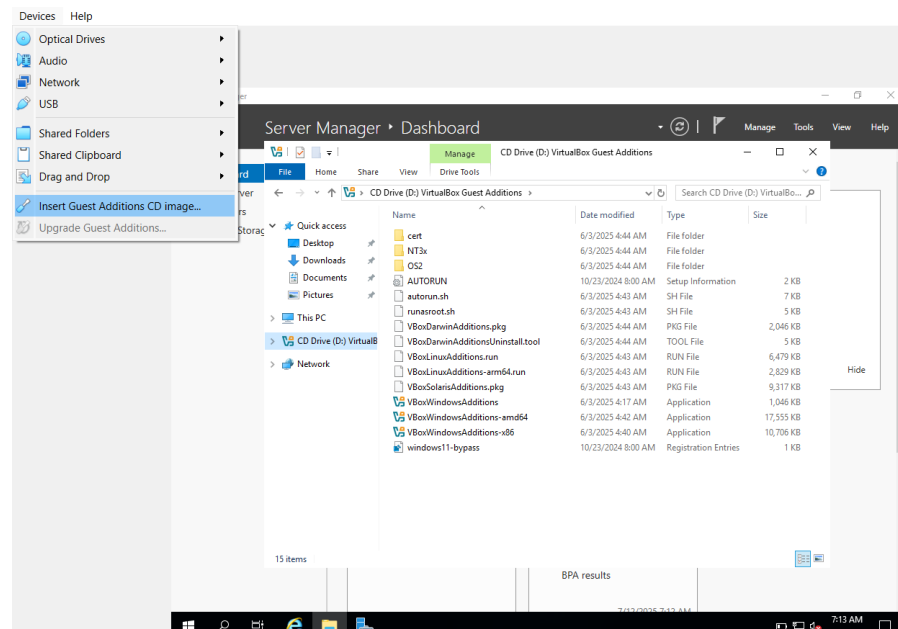
- בוחרים היכן פיזית תותקן מערכת ההפעלה על הדיסק הקשיח הווירטואלי שיצרנו קודם.



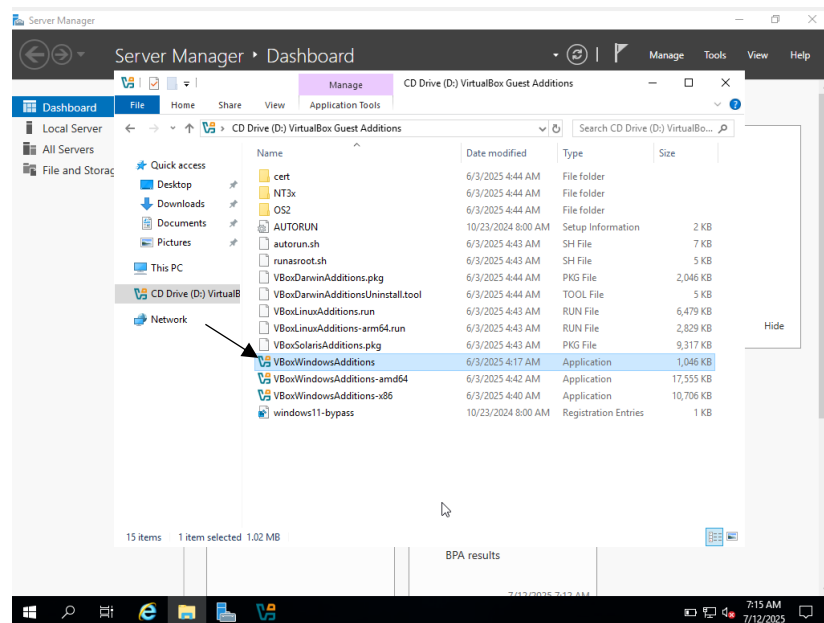
- הגדרנו ססמא חזקה עבור חשבון המנהל.



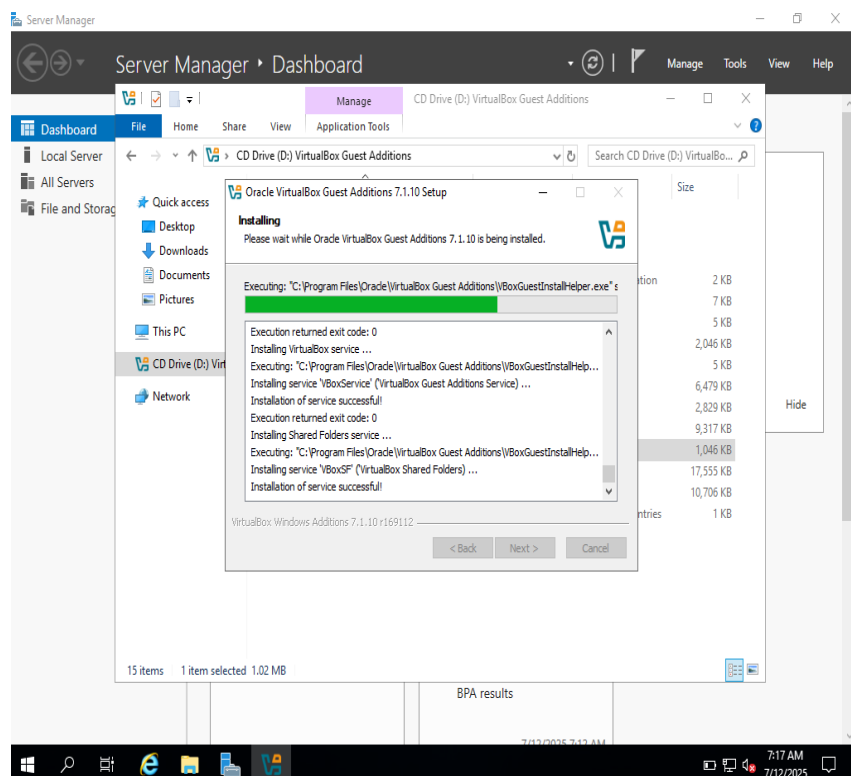
- לאחר שנכנסנו למכונה שיצרנו, נלחץ על "insert guests additions" כלי זה שימושי בעיקר לחוויית תצוגה טובה יותר, שיתוף קבצים, העתקה והדבקה ועוד..



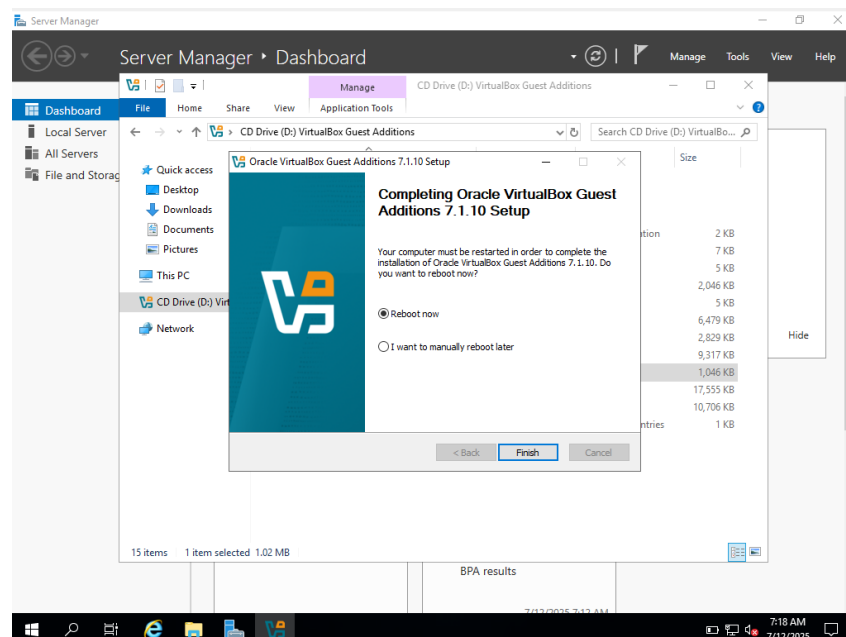
- נתקין את VBwindoesAdditions בשביל לראות את המכונה בתצוגה מלאה עם גישה לכלים למעלה.



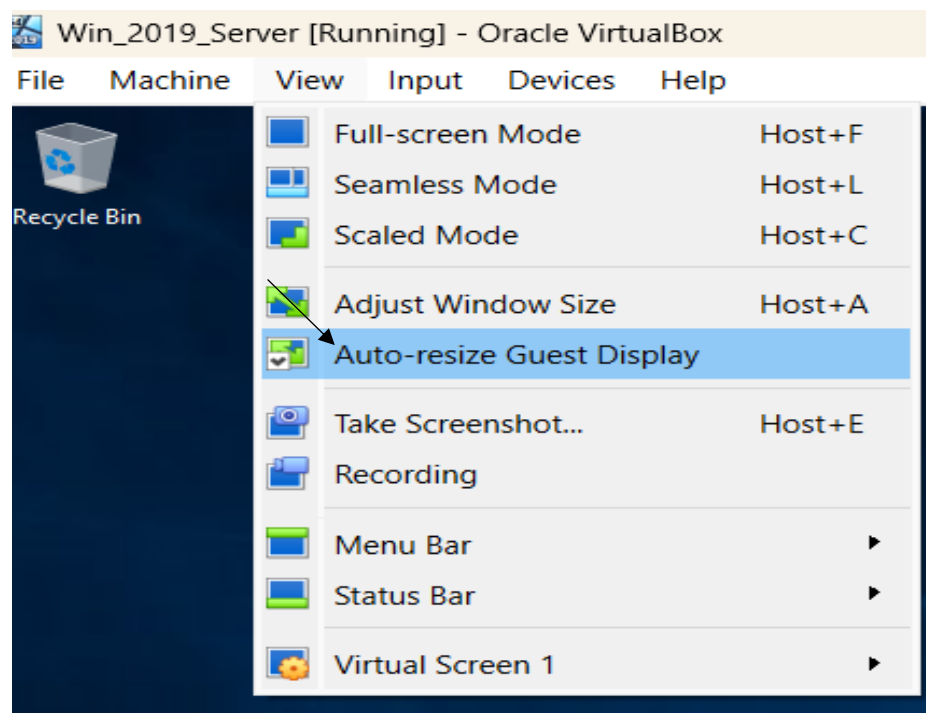
- מאשרים את הכל ולוחצים על install.



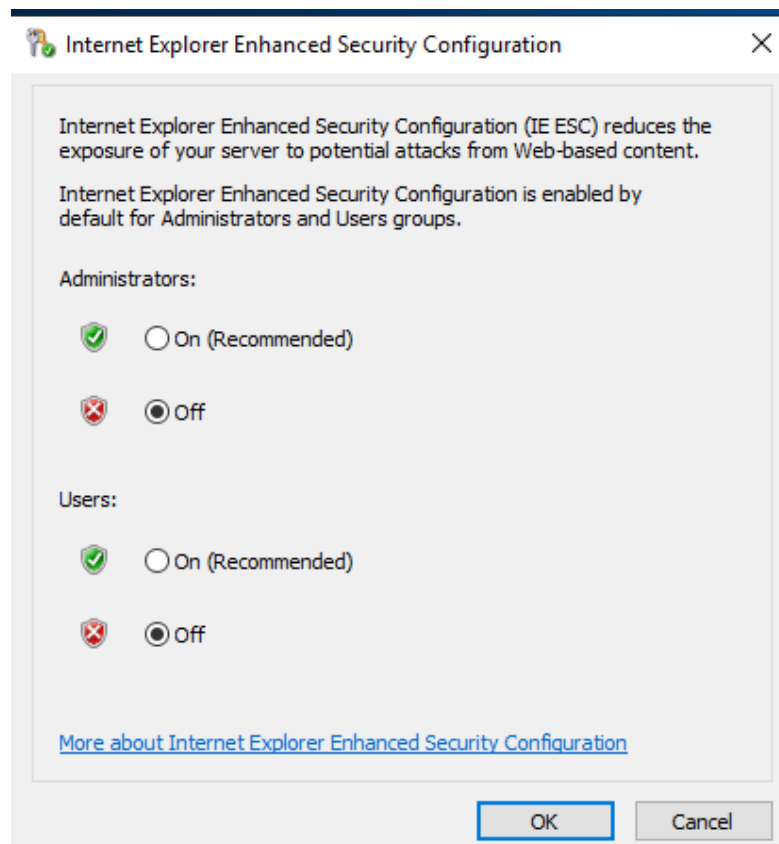
- ועושים ריבוט למכונה.



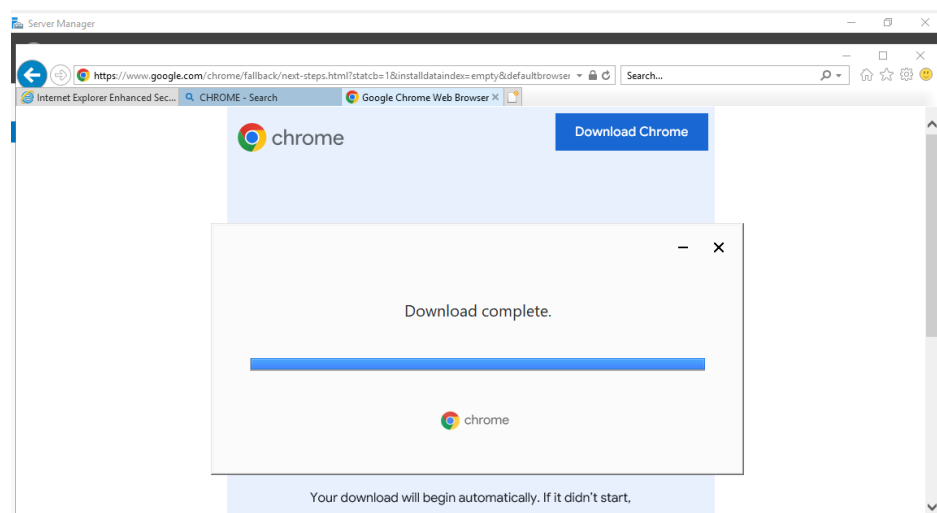
- אחרי ה"ריבוט", פותחים את המכונה ולוחצים על האופציה הזאת לתצוגה מלאה:



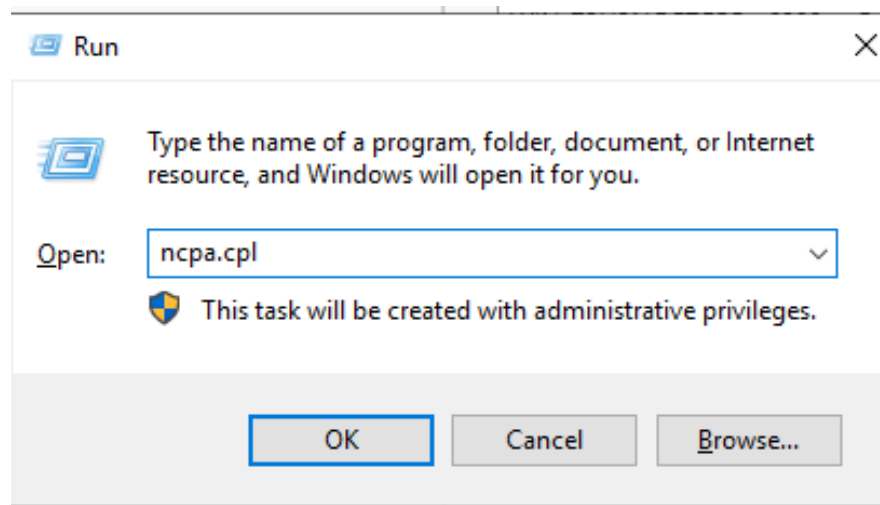
- מורידים את תכונה זו לטובת גלישה חופשית יותר באינטרנט, מה שנוח לצורך הורדת כלים שונים וגלישה למשאבים.



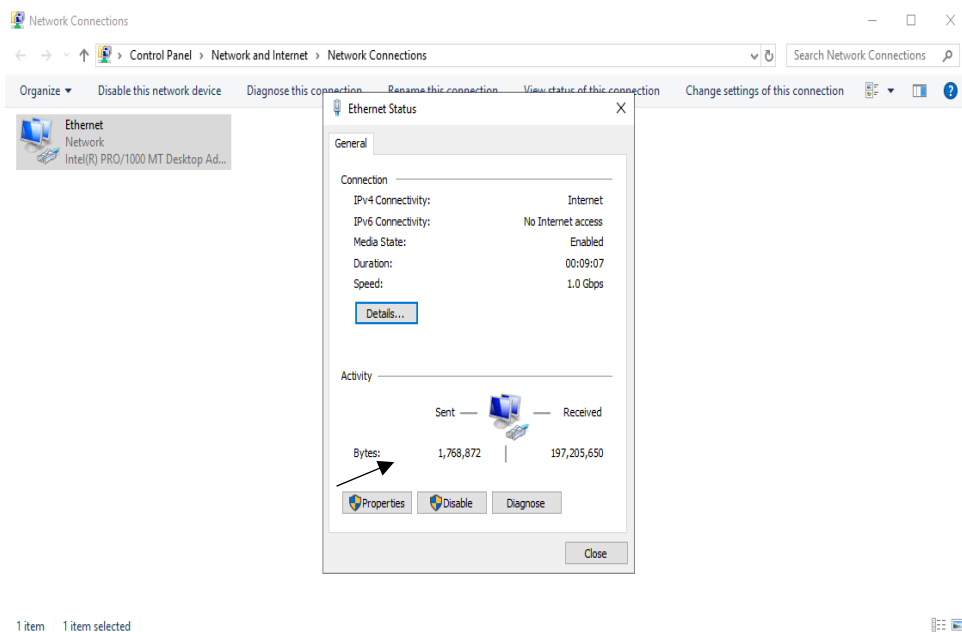
- הורדנו "כרום" בשביל הנוחות.



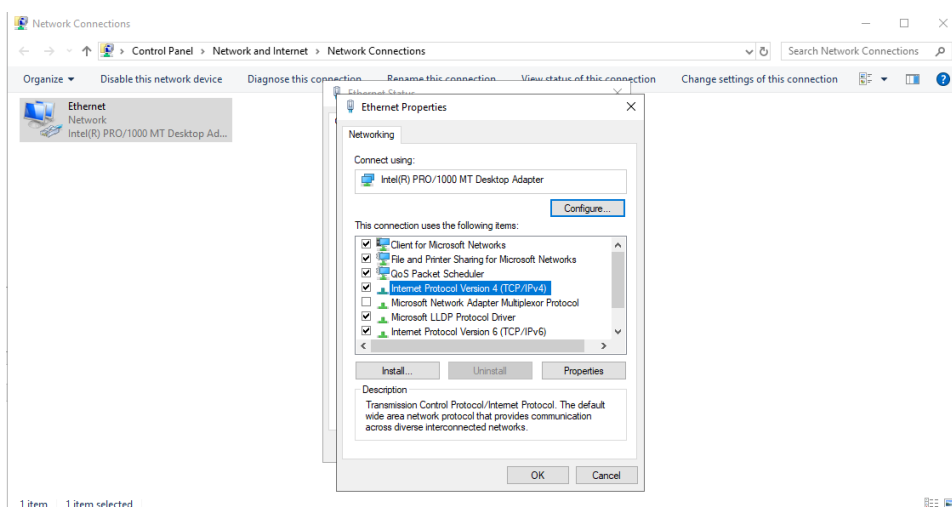
- לחצנו win+R ורשמנו את הפקודה הבאה :



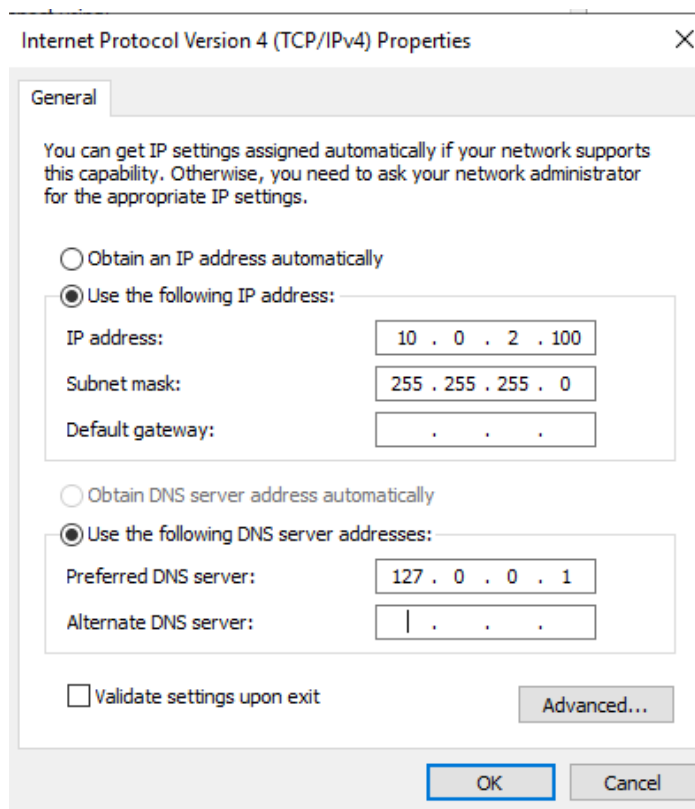
- לחצנו על ethernet והגענו לחלונית הבאה, נלחץ על "properties" :



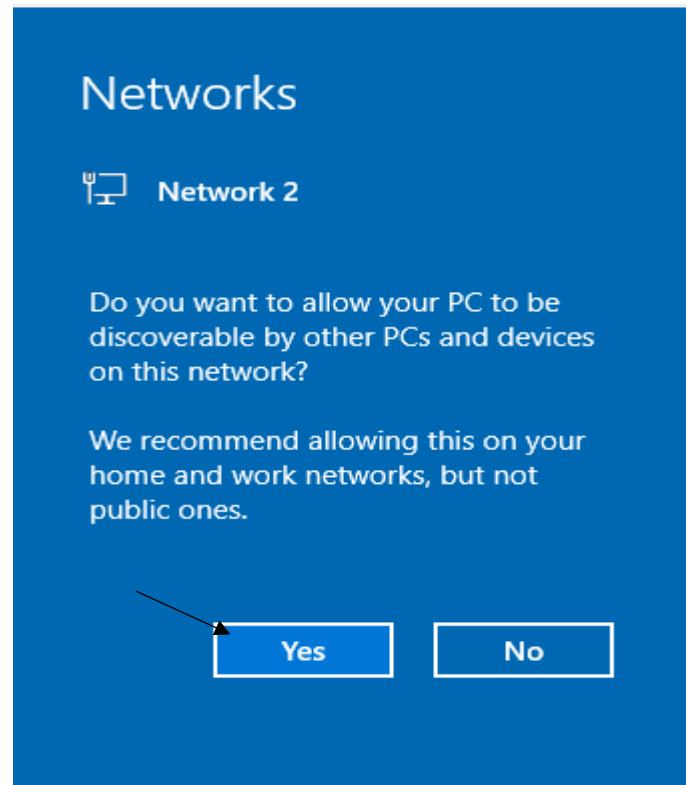
- לוחצים על ipv4 :



- הגדרנו IP קבועה כדי שהשרת תמיד יהיה זמין באותה כתובת, הגדרנו subnet mask בשביל להגדיר את גודל תת-רשת שהשרת נמצא בה וכך, מאפשר לו לתקשר עם מכשירים אחרים בטווח של הכתובות שהוגדר, הגדרנו גם כתובת IP ל-dns server – צעד חיוני כשהשרת מתקדם לבקר דומיין ויכלול את שירותי ה-DNS, הוא יפנה לעצמו עבור פתרונות לשמות בתוך הדומיין.



- לאחר שלחצנו "OK" על מה שהגדרנו, תצוץ חלונית ולוחצים עליה "yes" אישור זה מאפשר למחשב להיות גלוי למחשבים או בקשות חיבורים שונות ברשת.



- נכנסים לCMD ונותנים את הפקודה: `ipconfig /all`

הנה הפלט:

```
Administrator: Command Prompt

Windows IP Configuration

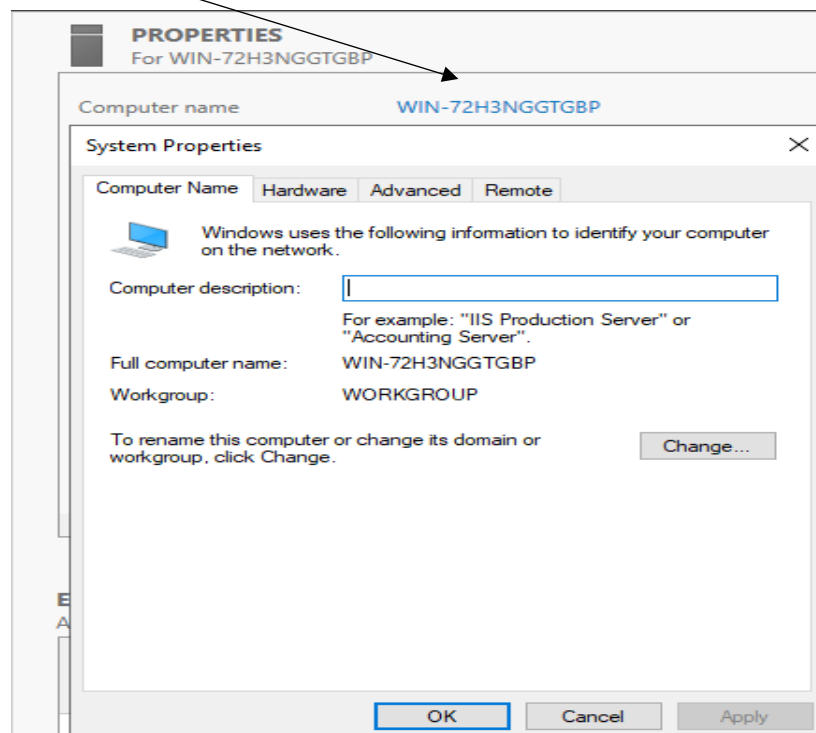
Host Name . . . . . : WIN-72H3NGGTGBP
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No

Ethernet adapter Ethernet:

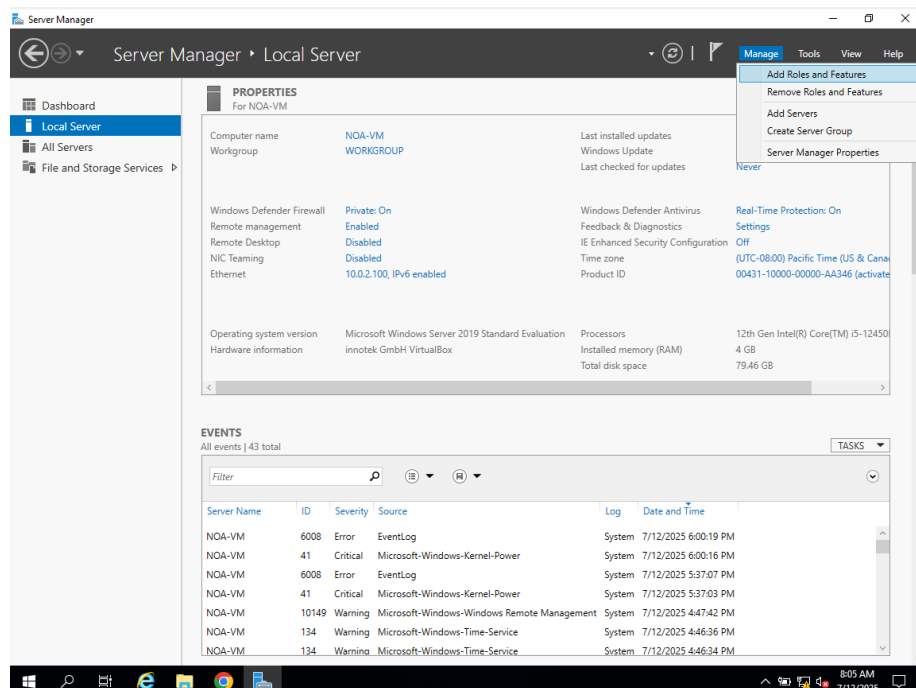
Connection-specific DNS Suffix . :
Description . . . . . : Intel(R) PRO/1000 MT Desktop Adapter
Physical Address. . . . . : 08-00-27-25-69-37
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
IPv6 Address. . . . . : fd17:625c:f037:2:fc95:9a50:e34c:7850(Preferred)
Link-local IPv6 Address . . . . : fe80::8b3:a460:9b8a:cbc%6(Preferred)
IPv4 Address. . . . . : 10.0.2.100(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : fe80::2%6
DHCPv6 IAID . . . . . : 101197623
DHCPv6 Client DUID. . . . . : 00-01-00-01-30-04-AC-3C-08-00-27-25-69-37
DNS Servers . . . . . : 127.0.0.1
NetBIOS over Tcpip. . . . . : Enabled

C:\Users\Administrator>
```

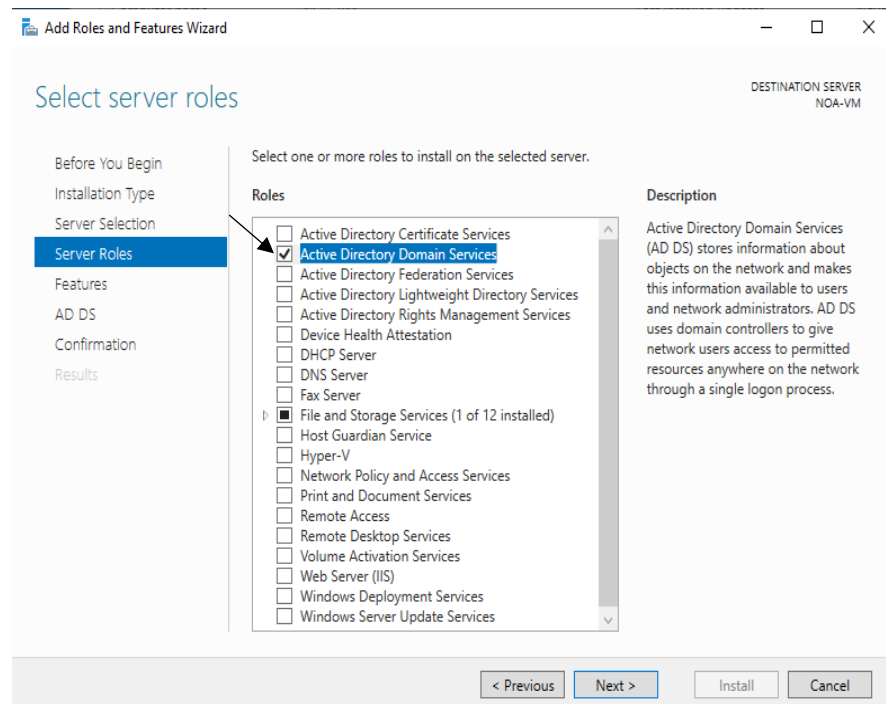
- לאחר מכן, נשנה את שם השרת: נלחץ עליו



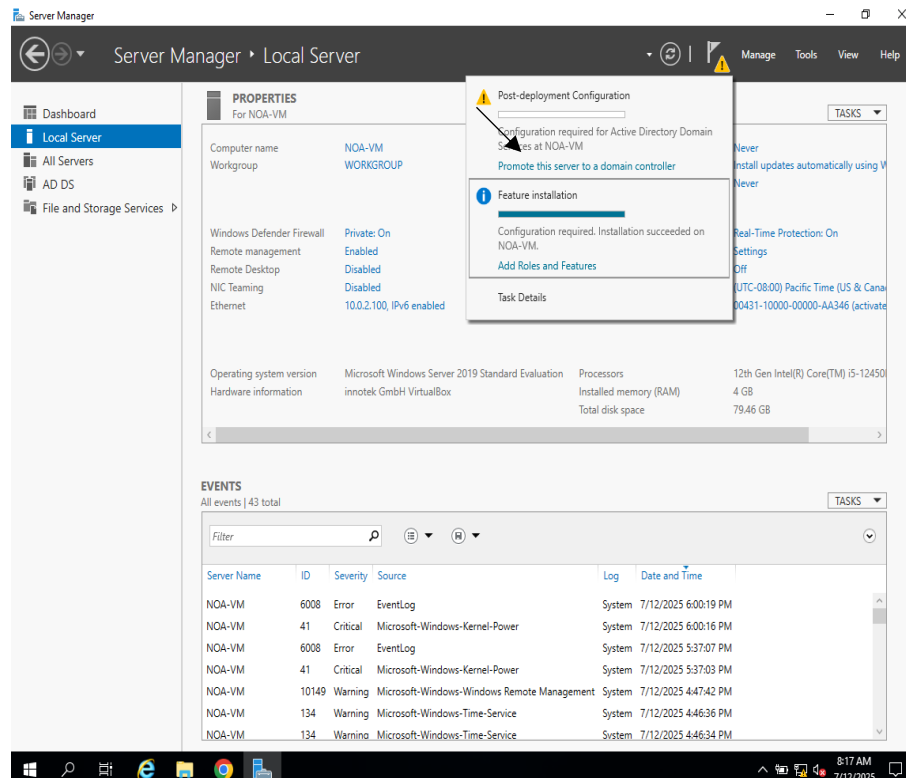
- נלחץ על "change" נשנה את השם על פי העדפה אישית ונלחץ "apply", המכונה תבקש לעשות "ריסטרט", ניתן לה לבצע זאת. לאחר מכן, נראה שהשם של המכונה השתנה ונלחץ על manage→add roles and features



- לעשות "next" עד שהחלונית מגיעה ל"server roles", ושם בוחרים את אופציה: Active Directory Domain Services. שירות זה הינו תפקיד השרת המרכזי שמאפשר לשרת לתפקד כבקר דומיין.



- ואז עושים "next" על הכל ואז "install" לאחר מכן, לוחצים על הדגל עם משולש האזהרה ולוחצים על הפקודה הבאה:



- לאחר מכן, בחרנו באופציה של יצירת FOREST חדש מכיוון שאין לנו forest קיים ואז בחרנו שם וססמא לדומיין.
Forest - על שמו הוא, כמו יער שמורכב מכמה עצים שיש את העץ הראשי כביכול שהוא הדומיין הראשי וכולם מחוברים תחתיו ביחסי גומלין.

- ברוב המקרים, נשאיר את שם NetBIOS כברירת מחדל.

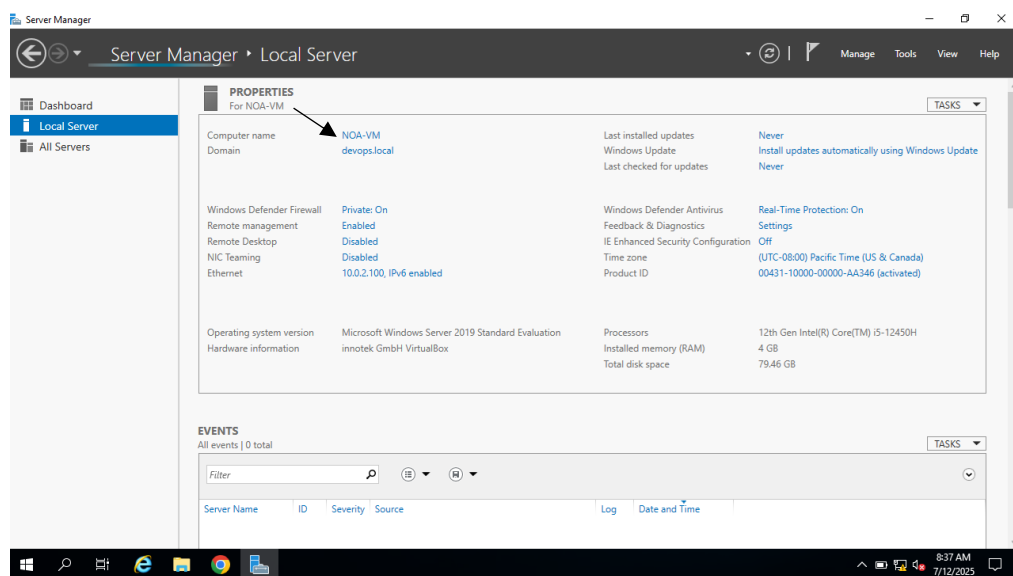
- בחלונית הזו, נשאר את האופציות הדיפולטיות ונעשה next.

The screenshot shows the 'Paths' step of the Active Directory Domain Services Configuration Wizard. The left sidebar contains a list of steps: Deployment Configuration, Domain Controller Options, DNS Options, Additional Options, Paths (selected), Review Options, Prerequisites Check, Installation, and Results. The main area is titled 'Specify the location of the AD DS database, log files, and SYSVOL'. It contains three input fields: 'Database folder:' with the value 'C:\Windows\NTDS', 'Log files folder:' with the value 'C:\Windows\NTDS', and 'SYSVOL folder:' with the value 'C:\Windows\SYSVOL'. Each field has a browse button (three dots) to its right. At the bottom right, there are buttons for '< Previous', 'Next >', 'Install', and 'Cancel'. The top right corner indicates 'TARGET SERVER NOA-VM'.

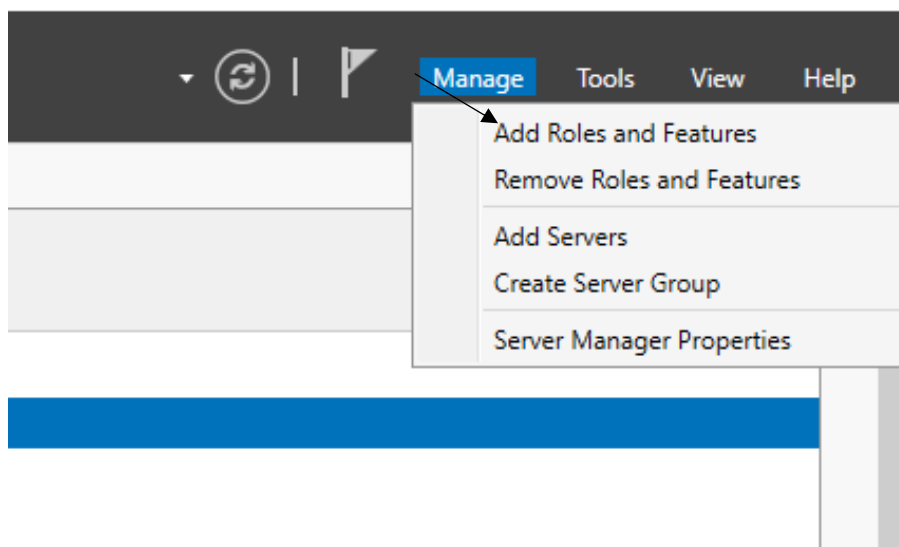
- לפני שנלחץ בחלונית "install", המערכת מבצעת בדיקה לוודא שכל התנאים הקודמים שהגדרנו תקינים, במידה וכן, תינתן האפשרות לבצע התקנה.

The screenshot shows the 'Prerequisites Check' step of the Active Directory Domain Services Configuration Wizard. The left sidebar is the same as in the previous screenshot, with 'Prerequisites Check' now selected. The main area has a title bar 'Prerequisites Check' and a status bar at the top indicating 'All prerequisite checks passed successfully. Click 'Install' to begin installation.' Below this, there is a section titled 'Prerequisites need to be validated before Active Directory Domain Services is installed on this computer' with a link 'Rerun prerequisites check'. A 'View results' section is expanded, showing a list of checks: 'cryptography algorithms when establishing security channel sessions.', 'For more information about this setting, see Knowledge Base article 942564 (http://go.microsoft.com/fwlink/?LinkId=104751).', 'A delegation for this DNS server cannot be created because the authoritative parent zone cannot be found or it does not run Windows DNS server. If you are integrating with an existing DNS infrastructure, you should manually create a delegation to this DNS server in the parent zone to ensure reliable name resolution from outside the domain "devops.local". Otherwise, no action is required.', 'Prerequisites Check Completed', and 'All prerequisite checks passed successfully. Click 'Install' to begin installation.' At the bottom, there is a warning icon and text: 'If you click Install, the server automatically reboots at the end of the promotion operation.' and a link 'More about prerequisites'. At the bottom right, there are buttons for '< Previous', 'Next >', 'Install', and 'Cancel'. The top right corner indicates 'TARGET SERVER NOA-VM'.

- לאחר ההפעלה מחדש, אנחנו אמורים לראות ששם הדומיין השתנה.



- נכנסים כעת לאופציה "manage" <--- "add roles and features":



- בוחרים את אופן ההתקנה הראשונה, זוהי שיטת ההתקנה הנפוצה ביותר.

Add Roles and Features Wizard

DESTINATION SERVER
NOA-VM.devops.local

Select installation type

Select the installation type. You can install roles and features on a running physical computer or virtual machine, or on an offline virtual hard disk (VHD).

- ☒ **Role-based or feature-based installation**
Configure a single server by adding roles, role services, and features.
- ☐ **Remote Desktop Services installation**
Install required role services for Virtual Desktop Infrastructure (VDI) to create a virtual machine-based or session-based desktop deployment.

< Previous Next > Install Cancel

- בוחרים את server שהקמנו, לאחר מכן, נלחץ על "next".

Add Roles and Features Wizard

DESTINATION SERVER
NOA-VM.devops.local

Select destination server

Select a server or a virtual hard disk on which to install roles and features.

- ☒ Select a server from the server pool
- ☐ Select a virtual hard disk

Server Pool

Filter:

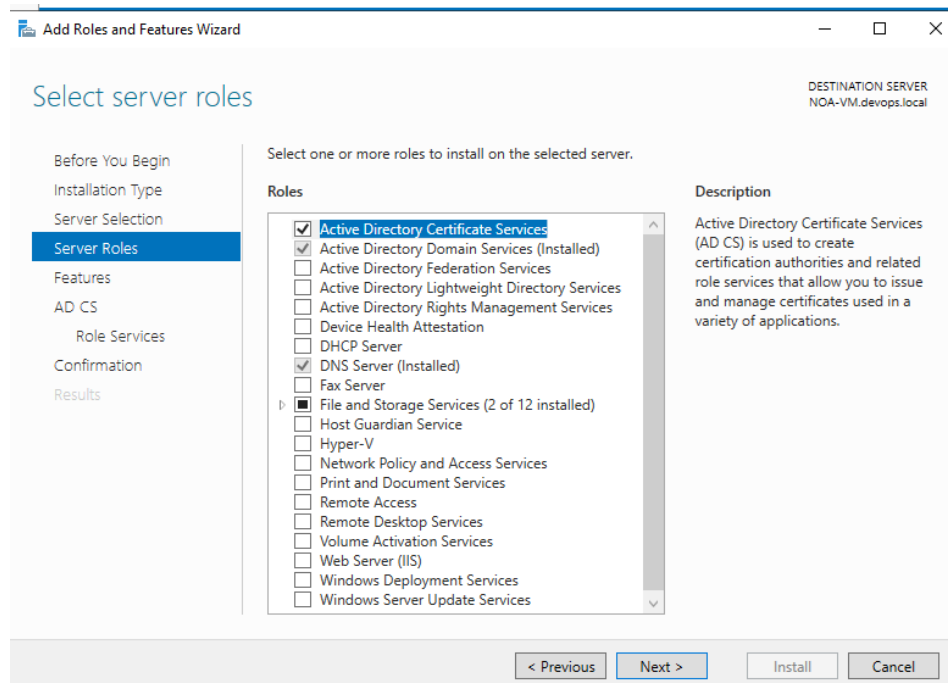
Name	IP Address	Operating System
NOA-VM.devops.local	10.0.2.100	Microsoft Windows Server 2019 Standard Evaluation

1 Computer(s) found

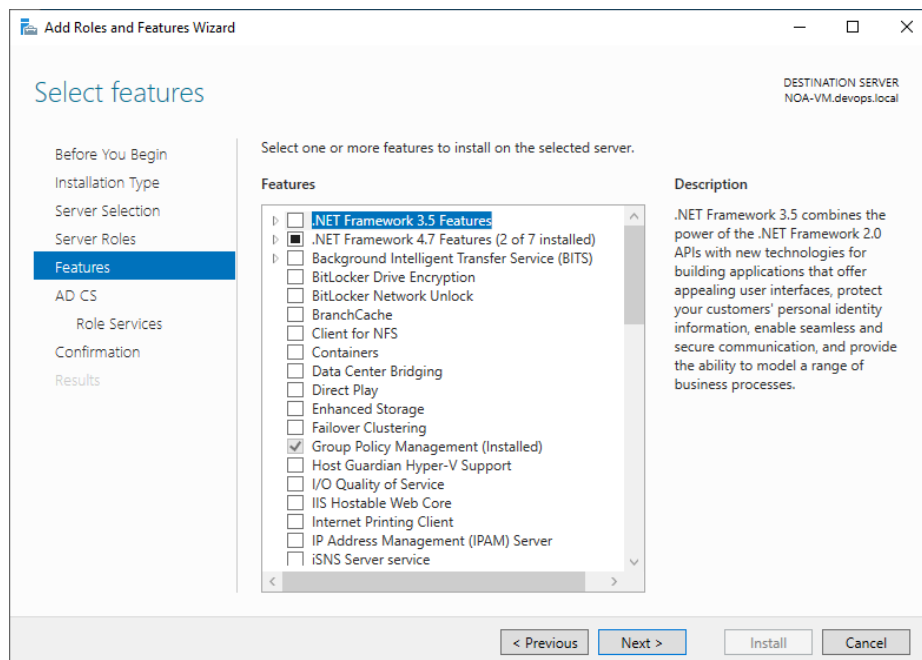
This page shows servers that are running Windows Server 2012 or a newer release of Windows Server, and that have been added by using the Add Servers command in Server Manager. Offline servers and newly-added servers from which data collection is still incomplete are not shown.

< Previous Next > Install Cancel

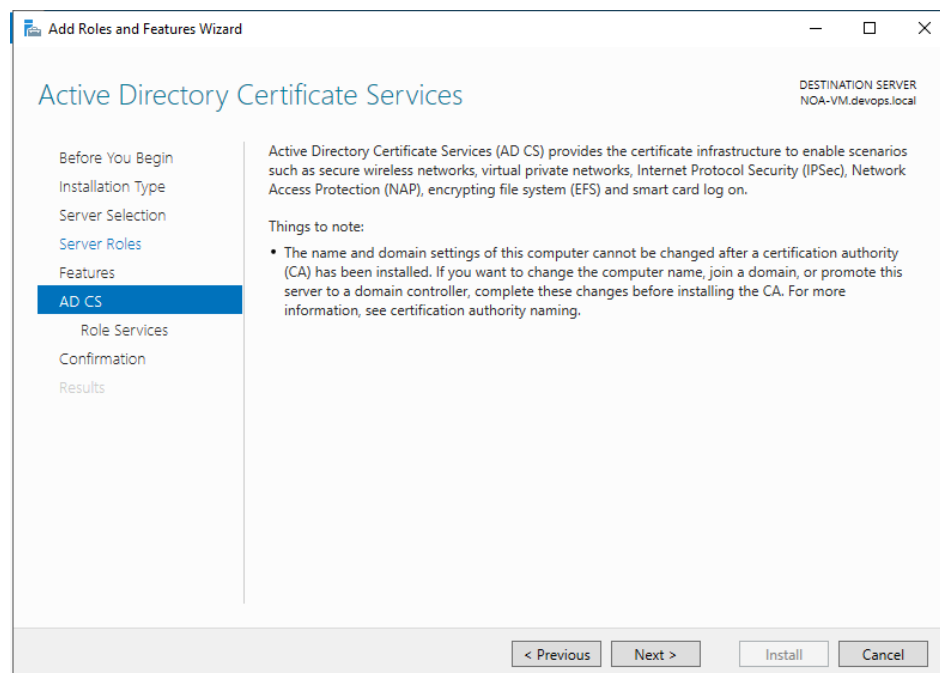
- נסמן את האופציה הראשונה שהיא: "Active Directory Certificate Services", שירות זה ממלא תפקיד של השרת שאחראי על הנפקה, ניהול וביטול אישורים דיגיטליים.



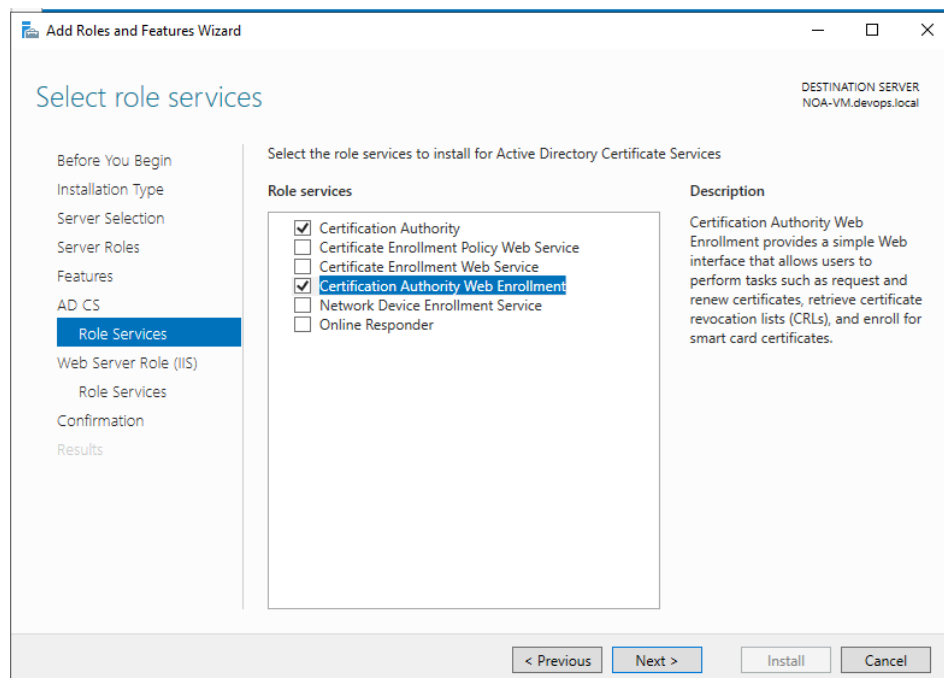
- יש ללחוץ "next" בשלב הבא, בדרך כלל אין צורך להוסיף תכונות אחרות מלבד הדיפולטיות.



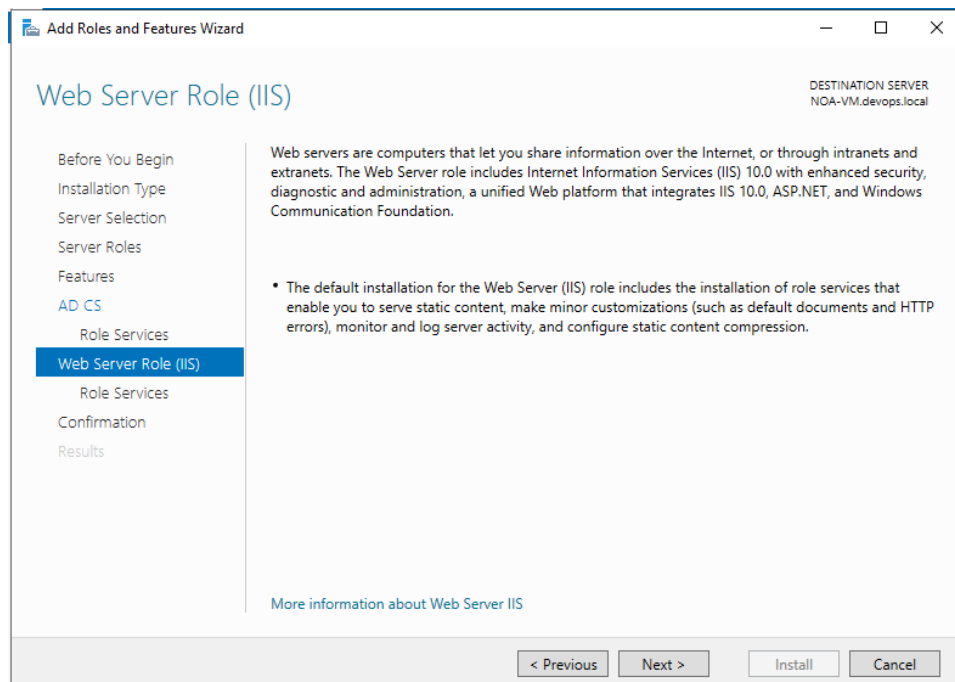
- כאן מוצג הסבר על ad CS , לקרוא ואז ללחוץ "next".



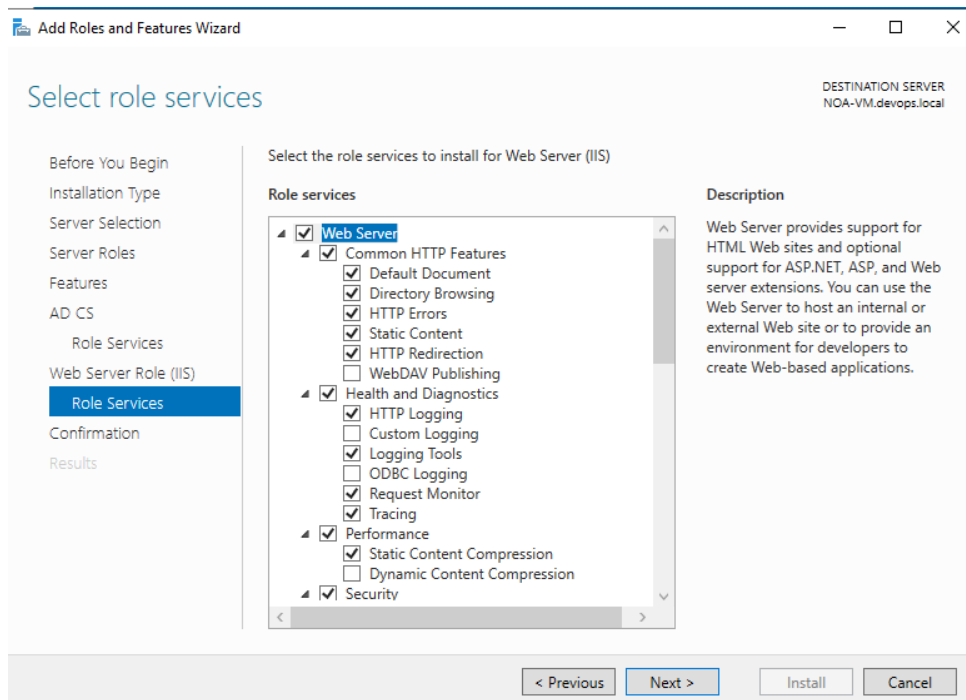
- כאן נבחר את האפשרויות Certification Authority Web Enrollment ו Certification Authority . CA הינו כלי שמאפשר להנפיק ולנהל אישורים, Web Enrollment זה כלי שמאפשר למשתמשים ומחשבים לבקש אישורים ולחדש אותם דרך דפדפן האינטרנט.



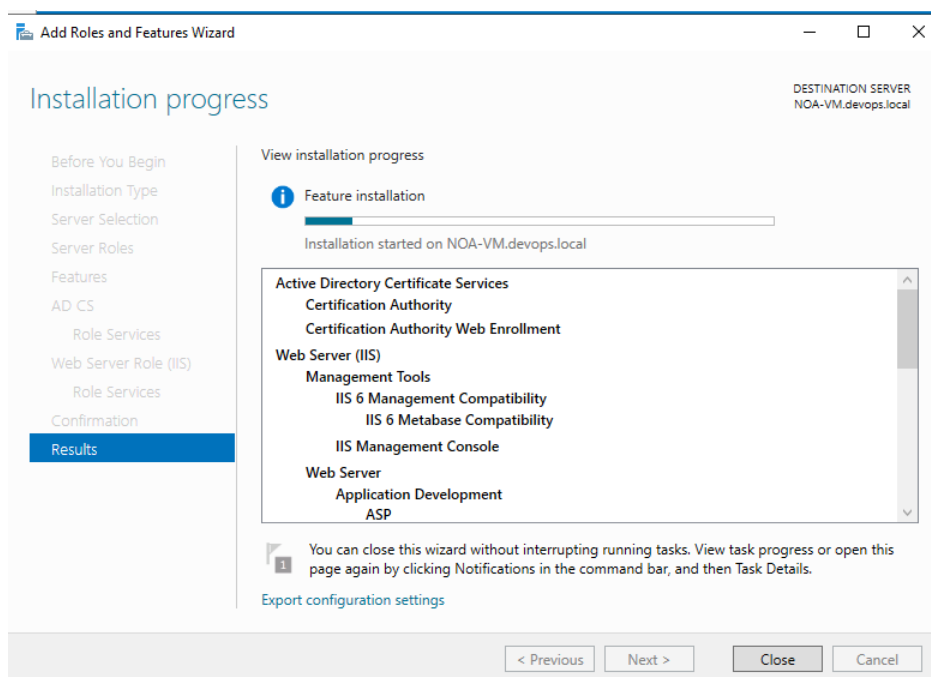
- מידע נוסף על ההתקנה, ללחוץ פשוט "next".



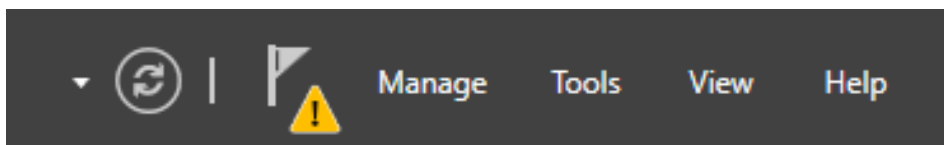
- משאירים את האופציות הדיפולטיות ואז "next".



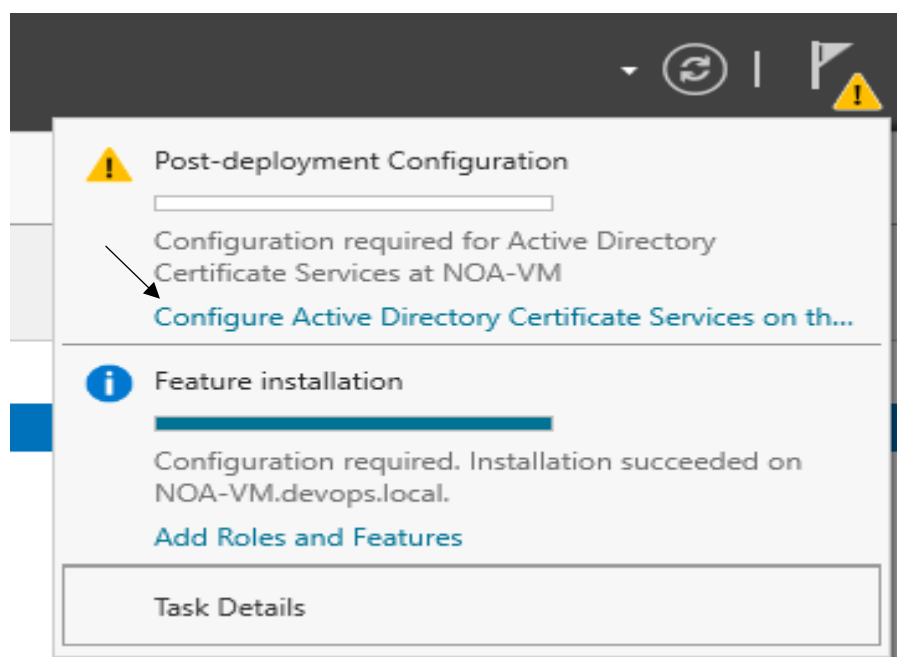
- ולוחצים "install".



- לאחר ההתקנה, יופיע לנו בדף למעלה את הדגל+ משולש אזהרה :



- נלחץ על האופציה הראשונה :



- יש לוודא שמגדירים אישורי מנהל מערכת מתאימים (חשבון מנהל הדומיין שהגדרנו לפני), ואז ללחוץ על "next"

AD CS Configuration

DESTINATION SERVER
NOA-VM.devops.local

Credentials

Specify credentials to configure role services

To install the following role services you must belong to the local Administrators group:

- Standalone certification authority
- Certification Authority Web Enrollment
- Online Responder

To install the following role services you must belong to the Enterprise Admins group:

- Enterprise certification authority
- Certificate Enrollment Policy Web Service
- Certificate Enrollment Web Service
- Network Device Enrollment Service

Credentials:

[More about AD CS Server Roles](#)

< Previous Next > Configure Cancel

- כאן, בוחרים את האופציות Certification Authority ו Certification Authority Web Enrollment.

AD CS Configuration

DESTINATION SERVER
NOA-VM.devops.local

Role Services

Select Role Services to configure

- ☒ Certification Authority
- ☒ Certification Authority Web Enrollment
- ☐ Online Responder
- ☐ Network Device Enrollment Service
- ☐ Certificate Enrollment Web Service
- ☐ Certificate Enrollment Policy Web Service

[More about AD CS Server Roles](#)

< Previous Next > Configure Cancel

- מאחר שיש לנו active directory נסמן בחלונית הבאה את האופציה Enterprise CA.

AD CS Configuration

DESTINATION SERVER
NOA-VM.devops.local

Setup Type

Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results

Specify the setup type of the CA

Enterprise certification authorities (CAs) can use Active Directory Domain Services (AD DS) to simplify the management of certificates. Standalone CAs do not use AD DS to issue or manage certificates.

☒ Enterprise CA
Enterprise CAs must be domain members and are typically online to issue certificates or certificate policies.

☐ Standalone CA
Standalone CAs can be members or a workgroup or domain. Standalone CAs do not require AD DS and can be used without a network connection (offline).

[More about Setup Type](#)

< Previous Next > Configure Cancel

- נבחר את סוג הCA, בוחרים את האופציה Root CA מכיוון שזוהי ההתקנה הראשונה של Certificate Authority בפורסט החדש שיצרנו קודם.

AD CS Configuration

DESTINATION SERVER
NOA-VM.devops.local

CA Type

Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results

Specify the type of the CA

When you install Active Directory Certificate Services (AD CS), you are creating or extending a public key infrastructure (PKI) hierarchy. A root CA is at the top of the PKI hierarchy and issues its own self-signed certificate. A subordinate CA receives a certificate from the CA above it in the PKI hierarchy.

☒ Root CA
Root CAs are the first and may be the only CAs configured in a PKI hierarchy.

☐ Subordinate CA
Subordinate CAs require an established PKI hierarchy and are authorized to issue certificates by the CA above them in the hierarchy.

[More about CA Type](#)

< Previous Next > Configure Cancel

- לאחר מכן, יש ליצור מפתח פרטי, לכן, נלחץ על האופציה הראשונה.

AD CS Configuration

DESTINATION SERVER
NOA-VM.devops.local

Private Key

Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results

Specify the type of the private key

To generate and issue certificates to clients, a certification authority (CA) must have a private key.

☒ Create a new private key
Use this option if you do not have a private key or want to create a new private key.

☐ Use existing private key
Use this option to ensure continuity with previously issued certificates when reinstalling a CA.

☒ Select a certificate and use its associated private key
Select this option if you have an existing certificate on this computer or if you want to import a certificate and use its associated private key.

☐ Select an existing private key on this computer
Select this option if you have retained private keys from a previous installation or want to use a private key from an alternate source.

[More about Private Key](#)

< Previous Next > Configure Cancel

- עבור רוב סביבות העבודה, ברירות המחדל שמוצגות בחלונית מספקות רמה טובה של אבטחה ויעילות, על כן נשאיר את התכונות הדיפולטיות מסומנות ונלחץ "next".

AD CS Configuration

DESTINATION SERVER
NOA-VM.devops.local

Cryptography for CA

Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results

Specify the cryptographic options

Select a cryptographic provider: RSA#Microsoft Software Key Storage Provider Key length: 2048

Select the hash algorithm for signing certificates issued by this CA:

- SHA256
- SHA384
- SHA512
- SHA1
- MD5

☐ Allow administrator interaction when the private key is accessed by the CA.

[More about Cryptography](#)

< Previous Next > Configure Cancel

- משאירים את CA name דפולטיבי.

AD CS Configuration

DESTINATION SERVER
NOA-VM.devops.local

CA Name

Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results

Specify the name of the CA

Type a common name to identify this certification authority (CA). This name is added to all certificates issued by the CA. Distinguished name suffix values are automatically generated but can be modified.

Common name for this CA:
devops-NOA-VM-CA

Distinguished name suffix:
DC=devops,DC=local

Preview of distinguished name:
CN=devops-NOA-VM-CA,DC=devops,DC=local

[More about CA Name](#)

< Previous Next > Configure Cancel

- משאירים את משך התוקף הדיפולטיבי (בדרך כלל 5 שנים)

AD CS Configuration

DESTINATION SERVER
NOA-VM.devops.local

Validity Period

Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results

Specify the validity period

Select the validity period for the certificate generated for this certification authority (CA):

5 Years

CA expiration Date: 7/13/2030 1:03:00 AM

The validity period configured for this CA certificate should exceed the validity period for the certificates it will issue.

[More about Validity Period](#)

< Previous Next > Configure Cancel

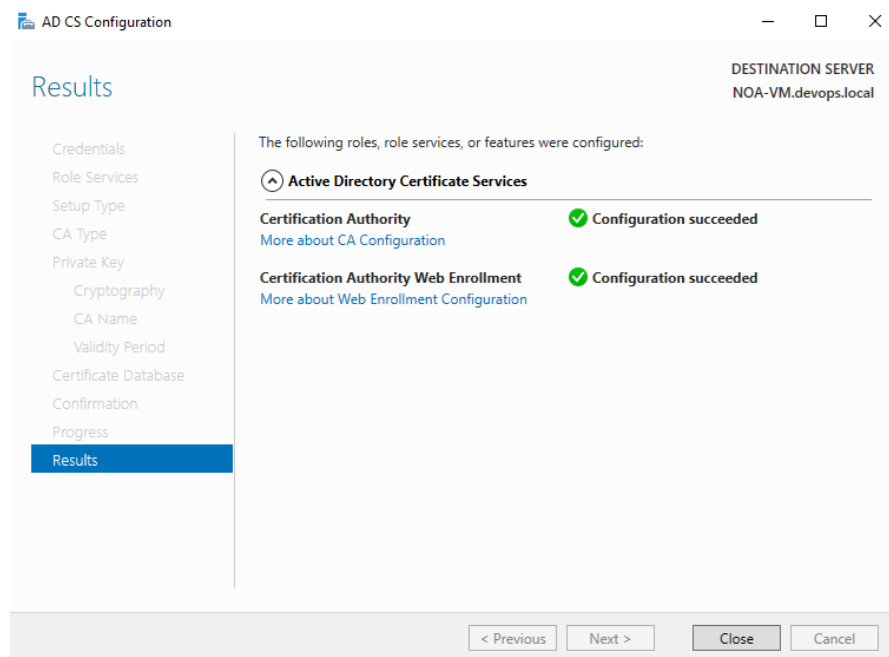
- משאירים את מיקום ה-DATABASE כדיפולטיבי מהסיבה שאין דרישות אחסון ספציפיות.

The screenshot shows the 'AD CS Configuration' wizard at the 'CA Database' step. The left sidebar lists steps: Credentials, Role Services, Setup Type, CA Type, Private Key, Cryptography, CA Name, Validity Period, **Certificate Database**, Confirmation, Progress, and Results. The main area is titled 'Specify the database locations'. It contains two text boxes: 'Certificate database location:' with the value 'C:\Windows\system32\CertLog' and 'Certificate database log location:' also with 'C:\Windows\system32\CertLog'. A 'More about CA Database' link is at the bottom. The top right shows 'DESTINATION SERVER NOA-VM.devops.local'. The bottom navigation bar has '< Previous', 'Next >', 'Configure', and 'Cancel' buttons.

- מאשרים את כל התכונות שהגדרנו בהגדרה, וכשווידינו שכל מה שאנחנו צריכים כלול, נלחץ על "configure".

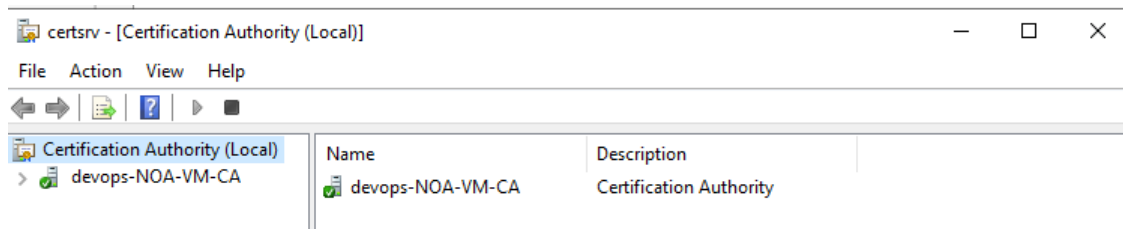
The screenshot shows the 'AD CS Configuration' wizard at the 'Confirmation' step. The left sidebar is the same as the previous step, but 'Confirmation' is now highlighted. The main area is titled 'Confirmation' and contains the text 'To configure the following roles, role services, or features, click Configure.' Below this is a section 'Active Directory Certificate Services' with a sub-section 'Certification Authority' listing various settings: CA Type (Enterprise Root), Cryptographic provider (RSA#Microsoft Software Key Storage Provider), Hash Algorithm (SHA256), Key Length (2048), Allow Administrator Interaction (Disabled), Certificate Validity Period (7/13/2030 1:03:00 AM), Distinguished Name (CN=devops-NOA-VM-CA,DC=devops,DC=local), Certificate Database Location (C:\Windows\system32\CertLog), and Certificate Database Log Location (C:\Windows\system32\CertLog). There is also a sub-section 'Certification Authority Web Enrollment'. The bottom navigation bar has '< Previous', 'Next >', 'Configure', and 'Cancel' buttons.

- אם ההתקנה צלחה, נקבל את הפלט הבא :



- בשביל לבדוק אם Certification Authority בוצע בהצלחה, ניתן לבצע כמה שלבים :

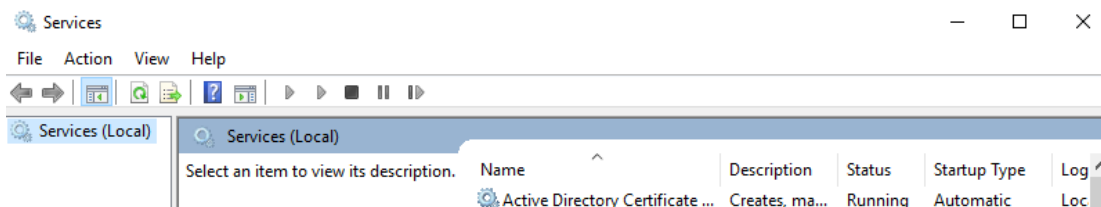
בדיקה 1- לפתוח tools <--- Certification Authority , אמור להיראות ככה :



בדיקה 2- לפתוח "command prompt" <--- ולהריץ את הפקודה הבאה :

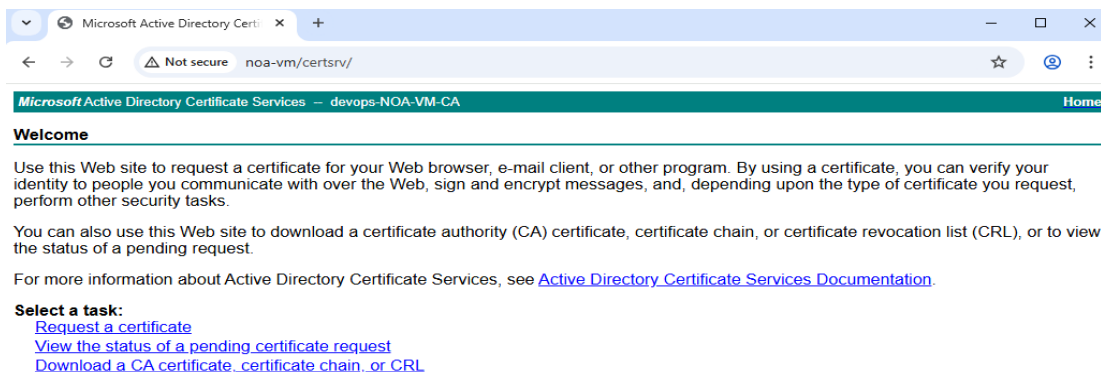


אמור להיפתח חלונית עם כל השירותים וסימון לידם אם הם רצים או לא, אנחנו אמורים לראות את שירות Certification Authority במצב "running" ולוודא שהוא במצב "automatic" ממש כמו פה :



בדיקה 3- נבדוק גם אם Web Enrollment שהתקנו עובד כראוי, ננסה לגשת אליו דרך דפדפן האינטרנט דרך השרת עצמו, הכתובת אמורה להיראות ככה : <http://NOA-VM/certsrv/>

במידה ונראה את הפלט המוצג, המשימה הושלמה!



שלב ג' - הרצת פרויקט בקוברנטיס

לפני שנתחיל במשימה, נגדיר כמה מושגים:

helm - כשצריך לעבוד עם קוברנטיס, צריך שיהיה פריסה של יישומים באמצעות קבצי yml שמתארים את הרכיבים השונים של היישום, עבור יישומים מורכבים, יש הרבה קבצים של yml לכן, קל יותר לנהל אותם עם helm, הכלי הזה גם מתקין ומשדרג את הקבצים הללו.

namespace - הינו דרך לסדר את האפליקציות והמשאבים שבתוך האשכול, וליצור בידוד בניהול ובהרשאות. הכלי הזה בעצם מחלק כל קבוצת משאבים לקבוצה כך שיהיה יותר קל לבקר ולשפר ביצועים.

במכונת manager עלינו לוודא שהאשכול מחובר וכל החיבורים מתבצעים כמו שצריך.

יש להתקין את helm על ידי הפקודה:

```
curl -fsSL https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash
```

לאחר מכן, יש ליצור Namespace חדש בשם "TBD":

נשים את המלל הבא בתוך namespace-tbd.yml:

```
apiVersion: v1
kind: Namespace
metadata:
  name: tbd
```

יש ליצור את Namespace על ידי הפקודה הבאה:

```
kubectl apply -f namespace-tbd.yml
```

לוודא שאכן נוצר namespace:

```
kubectl get namespaces
```

כעת, יש לפתוח תיקייה לפרויקט החדש של האתר:

```
Mkdir my-ui-app
```

```
Cd my-ui-app
```

ואז להעתיק לתיקיית הפרויקט את כל הקבצים שמצורפים לאתר:

```
GNU nano 6.2 index
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>open_page</title>
</head>
<body>
<label for="title" style="color: red;">TASK COMPLETE!</label>
</body>
</html>
```

index.html זה נמצא בתוך my-ui-app/templates/~

```

GNU nano 6.2
FROM python:3.8.3-slim-buster
RUN python -m pip install --upgrade pip
WORKDIR /app
COPY . /app
RUN pip install -r requirements.txt
CMD ["python", "app.py"]

```

~/my-ui-app Dockerfile נמצא

```

GNU nano 6.2
Flask

```

~/my-ui-app Requirement.txt נמצא

```

GNU nano 6.2
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/')
def hello_page():
    return render_template("index.html")

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)

```

App.py נמצא גם הוא ב: ~/my-ui-app

לאחר מכן, יש להתקין דוקר במכונה של manager (על פי הפקודות שצירפתי לפני)

בניתי image docker על ידי הפקודה הבאה:

. sudo docker build -t noa10203040/my-ui-app-python: v1.0

(noa10203040) זהו שם המשתמש שלי לדוקר האב.

ניתן לראות את הimage החדש שבנינו כאן:

```

vagrant@manager:~/my-ui-app$ sudo docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
noa10203040/my-ui-app-python  v1.0       2b154a34db71  About a minute ago  184MB
hello-world         latest     74cc54e27dc4  5 months ago   10.1kB

```

עכשיו, צריך להתחבר לדוקר האב, ניתן לעשות זאת עם הפקודה הבאה :

docker login

ואחרי שהאימות בוצע, יש לדחוף את התמונה לדוקר האב :

docker push noa10203040/my-ui-app-python:v1.0

בוצע!

```
vagrant@manager:~$ docker push noa10203040/my-ui-app-python:v1.0
The push refers to repository [docker.io/noa10203040/my-ui-app-python]
b34b462c3ca7: Pushed
444118bfad9d: Pushed
7a81cd335151: Pushed
0e4eb191021d: Pushed
be318362d0da: Mounted from library/python
f0698886f24f: Mounted from library/python
8e7524389116: Mounted from library/python
0bd71a837902: Mounted from library/python
13cb14c2acd3: Mounted from library/python
v1.0: digest: sha256:0084bd9182a9c838284e01c0cebe6d2f22cb6c53bd53b8f3cd67318bc5c99527 size: 2205
```

לפני הצעדים הבאים נבין :

מה זה ingress controller?

Ingress בפני עצמו הוא דף חוקים שמפרט- איזה דומיינים או נתיבים זמינים מבחוץ ולאילו שירותים פנימיים בקלאסטר כל כתובת צריכה להגיע.

Ingress controller הוא כבר היישום בפועל של אותם חוקים- הוא יודע לקלוט את הבקשות מהאינטרנט ולשלוח אותן ליעד הנכון.

כאשר משתמש מקליד את כתובת האתר שלי בדפדפן שלו : הדפדפן שולח בקשת HTTP/HTTPS לדומיין, הבקשה מגיעה ל-ingress controller ובודק את חוקי ingress ומנתב את הבקשה על פי החוקים הקיימים ליעד הנכון בקלאסטר.

הפקודות הבאות הן חלק מההתקנה של ingress controller מסוג ingress-nginx באשכול הקוברנטיס :

הוספת repository של ingress nginx של helm :

helm repo add ingress-nginx <https://kubernetes.github.io/ingress-nginx>

עדכון הריפוזיטורי :

helm repo update

```
vagrant@manager:~/my-ui-app$ helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
"ingress-nginx" has been added to your repositories
vagrant@manager:~/my-ui-app$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "ingress-nginx" chart repository
Update Complete. ✨Happy Helming!✨
```

לאחר מכן, יש להתקין התקנה של Nginx ingress controller על ידי הפקודה הבאה :

helm install ingress-nginx ingress-nginx/ingress-nginx --namespace tbd --create-namespace

יש לאמת התקנה ושה ingress controller מוכן :

```
vagrant@manager:~$ kubectl get pods --namespace tbd -l app.kubernetes.io/name=ingress-nginx
NAME                                READY   STATUS    RESTARTS   AGE
ingress-nginx-controller-578c564c54-8dk57  1/1     Running   0           2m10s
```

שלב הבא : יצירת קובץ manifests.yaml לאפליקציה :
קובץ זה נועד להגדרת האפליקציה והחשיפה שלה לקוברנטיס

```
apiVersion: apps/v1
```

שורה זאת מציינת את גרסת API של הקוברנטיס שבה אנו משתמשים כדי להגדיר את המשאב.
App/v1 זו בדרך כלל הגרסה שמשתמשים בה עבור deployments.

```
kind: Deployment
```

הגדרת סוג המשאב שאנו יוצרים.

```
metadata:
  name: my-ui-app-deployment
```

השם של deployment חייב להיות יחודי בתוך namespace.

```
namespace: tbd
```

ה namespace שבו deployment הזה יפרוס את pods.

```
labels:
  app: my-ui-app-python
```

תווית ספציפית שמזהה את deployment כחלק מהאפליקציה "my-ui-app-python"

```
spec:
  replicas: 2
  selector:
    matchLabels:
      app: my-ui-app-python
```

פה בעצם יש הגדרה של כמות pods שאנחנו רוצים להריץ , ה selector מגדיר כיצד deployment מוצא את pods שאנחנו אחראים עליהם ויחפש את pods עם התווית my-ui-app-python.

```
template:
  metadata:
    labels:
      app: my-ui-app-python
```

זה בעצם התווית עבור pods ש deployment ייצור. כל pod שיווצר יסתמך על ההגדרות הבאות :

```
spec:
  containers:
  - name: my-ui-app-container
    image: noa10203040/my-ui-app-python:v1.0
    ports:
    - containerPort: 5000
```

כל pod יריץ קונטיינר אחד, שם הקונטיינר: my-ui-app-container.
נציין את השם של התמונה שדחפנו לדוקר האב שתשמש ליצירת הקונטיינר, נציין גם את הפורט שהאפליקציה מאזינה לו בתוך הקונטיינר. זהו פורט פנימי לקונטיינר בלבד.

```
apiVersion: v1
```

v1 זו בדרך כלל הגרסה שמשתמשים בה עבור deployments.

```
kind: Service
```

הגדרת סוג המשאב שאנו יוצרים.

```
metadata:
  name: my-ui-app-service
  namespace: tbd
  labels:
    app: my-ui-app-python
```

השם הייחודי של servicen בתוך namespace שיצרנו (של tbd), הגדרת servicen כחלק מהאפליקציה.

```
spec:
  selector:
    app: my-ui-app-python
  ports:
  - protocol: TCP
    port: 80
    targetPort: 5000
  type: ClusterIP
```

servicen ינתב תעבורה לpods שיש להם תווית של "my-ui-app-python", תווית זו חייבת להתאים לתווית pods שdeployment יצר.

פרוטוקול ופורטים שservice חושף: פרוטוקול TCP (עבור http/https), פורט 80- הפורט שיתקשרו אל ה service מתוך האשכול, פורט 5000- הפורט שמוגדר באפליקציה, targetport זה הפורט שהאפליקציה מאזינה לו בתוך הקונטיינר.

Type: clusterIP אומר שהservicen יהיה נגיש רק מתוך האשכול של קוברנטיס עצמו.

להלן קובץ manifests.yaml המלא:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-ui-app-deployment
  namespace: tbd
  labels:
    app: my-ui-app-python
spec:
  replicas: 2
  selector:
    matchLabels:
      app: my-ui-app-python
  template:
    metadata:
      labels:
        app: my-ui-app-python
    spec:
      containers:
      - name: my-ui-app-container
        image: noa10203040/my-ui-app-python:v1.0
        ports:
        - containerPort: 5000
---
apiVersion: v1
kind: Service
metadata:
  name: my-ui-app-service
  namespace: tbd
  labels:
    app: my-ui-app-python
spec:
  selector:
    app: my-ui-app-python
  ports:
  - protocol: TCP
    port: 80
    targetPort: 5000
  type: ClusterIP
```

באופן כללי, קודם כל אנו מגדירים את deploymentn שתפקידו העיקרי הוא ליצור ולנהל את הפודים באפליקציה, מציינים כמה עותקים (כמה פודים) אנחנו רוצים שירוצו, deploymentn מצמיד לכל אחד מהפודים שהוא יותר תווית ייחודית, תווית זו משמשת כדי לזהות את הפודים ששייכים לתהליך שלו ולתהליך של servicen. בכל אחד מהפודים ירוץ קונטיינר אחד, הוא נבנה מתמונת דוקר שדחפנו לדוקר האב.

לאחר מכן, רשמנו פעולה של "service" שתפקידה לנהל את התעבורה אל הפודים ולחשוף אותם לפורטים ופרוטוקולים בתוך האשכול. הקונטיינר מוגדר לרוץ על TCP ובפורט 5000, servicen מקבל בקשות בפורט 80 מתוך האשכול ומנתב אותם אל פורט 5000 בתוך הפודים.

כעת, אחרי שהגדרנו את הקובץ, יש להפעיל אותו על ידי הפקודה הבאה:

```
kubectl apply -f manifests.yaml
```

לוודא שכל המשאבים נוצרו:

פקודה שבודקת אם deployment נוצר:

```
kubectl get deployment my-ui-app-deployment -n tbd
```

פקודה הבודקת אם הפודים רצים:

```
kubectl get pods -n tbd -l app=my-ui-app-python
```

פקודה הבודקת אם servicen נוצר:

```
kubectl get service my-ui-app-service -n tbd
```

לשלב אחד לפני האחרון, אנחנו צריכים לעדכן את קובץ hostn במחשב הפיזי:

הנה השלבים:

1. כתיבת קובץ **my_app_ingress.yaml** - my-app-שמשגדיר Ingress-Nginx Controller שכמגיעה תעבורה עבור שם דומיין ספציפי (למשל my-ui-app.local), היא צריכה להיות מנותבת אל ה service של האפליקציה (my-ui-app-service) בתוך האשכול.

2. **הוספת שורה בקובץ hostn** של המחשב הפיזי שלי, דרך ה ip של ה manager שבו הרצתי את Ingress-Nginx Controller.

3. שינוי ה service בתוך ingress-Nginx מ **loadbalancer** ל **nodeport**, דבר זה מאפשר ל Ingress Controller להיות נגיש מחוץ ל VM דרך פורט גבוה (למשל 31162) על כתובת ה ip של ה VM, ואז פשוט לחפש בדפדפן:

<http://my-ui-app.local:31162>

קובץ my_app_ingress.yaml:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ui-app-ingress
  namespace: tbd
spec:
  ingressClassName: nginx
  rules:
  - host: my-ui-app.local
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: my-ui-app-service
            port:
              number: 80
  tls:
  - hosts:
    - my-ui-app.local
    secretName: my-ui-app-tls
```

הסבר על הקובץ:

apiVersion: networking.k8s.io/v1 - הגרסה הסטנדרטית עבור משאבי ingress.

kind: Ingress - סוג המשאב שאנחנו יוצרים.

name: my-ui-app-ingress - שם ייחודי בתוך namespace שיצרנו (tbd).

ingressClassName: nginx - מציין לאיזה Ingress Controller ספציפי באשכול אתה רוצה שיטפל בכללי הניתוב שהגדרתי בקובץ ingress הזה.

host: my-ui-app.local - זהו שם הדומיין של ingress controller יאזין לו, כל בקשה שמגיעה עם הכותרת my-ui-app.local תטופל על ידי הכללים שכתובים בקובץ.

http: כללים אלה חלים על תעבורת HTTP\HTTPS.

path: / - כל בקשה שמגיעה לדומיין (my-ui-app.local), לא משנה מה הניתוב שלה, תשלח לservice.

name: my-ui-app-service - השם של service הפנימי בתוך האשכול שאליו תנותב התעבורה. זה חייב להיות אחד לאחד השם שהוגדר בmanifests.yaml.

```

tls:
- hosts:
  - my-ui-app.local
secretName: my-ui-app-tls

```

הקטע הנל נועד בשביל להגיד ל-Ingress Controller : כאשר מישור מנסה לגשת ל-my-ui-app.local באמצעות HTTPS השתמש בתעודה ובמפתח שנמצאים בsecret שנקרא my-ui-app-tls בשביל לאבטח את החיבור.

number: 80 : Ingress Controller שולח את הבקשה שמגיעה מהדפדפן לservice בתוך האשכול ופונה אליו ספציפית בפורט 80 שלו.

יש ליצור תעודת SSL/TLS חתומה עצמית דרך הפקודה הבאה:

```

openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out tls.crt -subj
"/CN=my-ui-app.local/O=my-ui-app

```

יצירת secret עבור התעודה:

פקודה זו לוקחת את קבצי התעודה והמפתח שיצרנו בשלב הקודם, ומאחסנת אותם בsecret בתוך קוברנטיס. ה-Ingress Controller ישתמש בו כדי לאבטח את החיבור לדפדפן.

```
kubectl create secret tls my-ui-app-tls --cert=tls.crt --key=tls.key -n tbd
```

איד נעשה את השינוי בקובץ host :

נכנסים תחילה לקובץ הזה דרך פאנל של:

```
notepad C:\Windows\System32\drivers\etc\hosts
```

לאחר מכן, יוצאים ונכנסים שוב ל**notepad** כמנהל (בשביל ההרשאות לשמור את השינויים)

לבסוף, מוסיפים את השורה 192.168.56.12 my-ui-app.local

Ip של מכונת worker2 (איפה שרץ Ingress-Nginx Controller) ואת השם של הדומיין המקומי של האפליקציה.

ופשוט פותחים דפדפן בכתובת האקספלורר ומחפשים את כתובת:

<https://my-ui-app.local:30187>

כשניגשים לאפליקציה דרך HTTPS בסביבת vagrant, צריך לציין את פורט nodeport בכתובת URL במקרה שלנו זה (30187), מכיוון ש-ingress controller נחשף דרך פורט זה ל-VM ולא ישירות על פורט 443.

הנה הפלט שיוצא: המשימה הושלמה!

