Ansible Exp 7,8

Install and Configure OpenSSH

sudo apt install openssh-server

1. على كل الأجهزة (Ubuntu Server أو Workstation):

افتح الطرفية وأدخل الأمر التالي لتثبيت خدمة OpenSSH:

Edit Copy bash

```bash
sudo apt install openssh-server
```

- هذا الأمر يقوم بتنزيل وتثبيت حزمة `openssh-server` التي تسمح باستقبال اتصالات SSH على الجهاز.
- عند نجاح التثبيت، سيكون السيرفر جاهزًا لاستقبال اتصالات SSH من أجهزة أخرى.

ssh student@[SRVR_IP]

🌐 ثانيًا: الاتصال الأولي من أجهزة Workstation إلى السيرفرات

✅ الهدف:

عند الاتصال لأول مرة عبر SSH، الجهاز سيطلب منك تأكيد البصمة الرقمية (fingerprint) الخاصة بالسيرفر، وذلك لأسباب أمنية.

✅ الخطوات:

. على **Workstation1** و **Workstation2**:

اتصل على كل سيرفر من السيرفرات الثلاثة (SRVR01 و SRVR02 و SRVR03):

Edit Copy bash

```bash
ssh student@[SRVR_IP]
```

مثال:

Edit Copy bash

```bash
ssh student@192.168.1.10
```

ولًا افحص هل عندك مجلد مفاتيح

~$ ls -la .ssh

لو المجلد لا يحتوي على مفاتيح، ننتقل لإنشائها:.

~$ ssh-keygen -t ed25519 -C "ansible"

عند السؤال:

Enter file in which to save the key (/home/student/.ssh/id_ed25519):

❗ اكتب:

/home/student/.ssh/ansible

🔍 شرح الخيارات:

- `-t ed25519` : نوع المفتاح (ed25519 أكثر أمانًا من rsa).
- `-C "ansible"` : تعليق يوضح أن هذا المفتاح مخصص لـ ansible.

ls -la ~/.ssh

ستجد ملفين جديدين:

- `ansible` : المفتاح **الخاص** (لا تعطيه لأي أحد)
- `ansible.pub` : المفتاح **العام** (هو ما ترسله للسيرفرات)

يمكنك رؤية محتواهم:

Edit Copy bash

```bash
cat ~/.ssh/ansible
cat ~/.ssh/ansible.pub
```

~/.ssh$ ssh-copy-id -i ~/.ssh/ansible.pub [SRVR_IP]

📤 خامسًا: إرسال المفتاح العام إلى كل سيرفر

✅ الهدف:

تُرسل المفتاح العام للسيرفر حتى يسمح لنا بالاتصال بدون كلمة مرور.

✅ الخطوات:

على **Workstation1**:

Edit Copy bash

```bash
ssh-copy-id -i ~/.ssh/ansible.pub student@[SRVR_IP]
```

كرر هذا الأمر مع كل سيرفر: SRVR01, SRVR02, SRVR03

📍 بعد الإرسال، يتم إضافة المفتاح في الملف:

Edit Copy bash

```bash
~/.ssh/authorized_keys
```

على السيرفر.

Workstation1$ scp ~/.ssh/ansible student@[workstation2_IP]:/home/student/.ssh/ansible
Workstation1$ scp ~/.ssh/ansible.pub student@[workstation2_IP]:/home/student/.ssh/ansible.pub

🔁 سادسًا: نقل المفاتيح إلى Workstation2

✅ الهدف:

بما أن Ansible هو المستخدم نفسه على كل الأجهزة، نحتاج نفس المفاتيح على الجهاز الثاني.

✅ الخطوات:

من **Workstation1**:

Edit Copy bash

```bash
scp ~/.ssh/ansible student@[Workstation2_IP]:/home/student/.ssh/ansible
scp ~/.ssh/ansible.pub student@[Workstation2_IP]:/home/student/.ssh/ansible.pub
```

🔒 تأكد أنك في مجلد `ssh.` على Workstation2 وأن الملفات وصلت.

Workstation1$ ssh [SRVR_IP]
Workstation2$ ssh [SRVR_IP]

✅ سابعًا: اختبار الاتصال بدون كلمة مرور

✅ الهدف:

نتأكد أن الاتصال صار يعمل بدون إدخال كلمة السر.

✅ الخطوات:

من **Workstation1** و**Workstation2**، جرب الاتصال بأي سيرفر:

Edit Copy bash

```bash
ssh student@[SRVR_IP]
```

✅ إذا فتحت الجلسة بدون ما يطلب كلمة مرور → مبروك! إعداد المفاتيح نجح 🎉

Version Control

على
Workstation1 و **Workstation2**:

sudo apt update
sudo apt install git

بعدين إنشاء حساب GitHub

Then Create a new repository and name it nislab.

✅ Enable the "Initialize this repository with a README" option.

Click Create Repository.

cat ~/.ssh/ansible.pub

🔐 إضافة مفتاح SSH إلى GitHub

حتى نقدر نربط جهازنا بـ GitHub بطريقة آمنة دون الحاجة لإدخال اسم المستخدم وكلمة السر كل مرة.

✅ على Workstation1:

1. في الطرفية:

Edit Copy bash

```bash
cat ~/.ssh/ansible.pub
```

انسخ كل المحتوى.

2. ارجع لموقع GitHub:

- اضغط على صورتك الشخصية > Settings
- اختر: **SSH and GPG Keys**
- اضغط: **New SSH Key**
- العنوان: `ansible`
- الصق المفتاح المنسوخ في الحقل
- اضغط: **Add SSH Key**

3. كرر نفس العملية لاحقًا على Workstation2.

```
nano ~/.ssh/config

Host github.com
  Hostname ssh.github.com
  Port 443
```

⚙️ إعداد SSH لاستخدام المنفذ 443 (مفيد لو في حجب على port 22)

✅ على Workstation1 و Workstation2:

1. افتح ملف جديد داخل مجلد `ssh.`:

Edit Copy bash

```bash
nano ~/.ssh/config
```

2. الصق التالي:

Edit Copy nginx

```nginx
Host github.com
  Hostname ssh.github.com
  Port 443
```

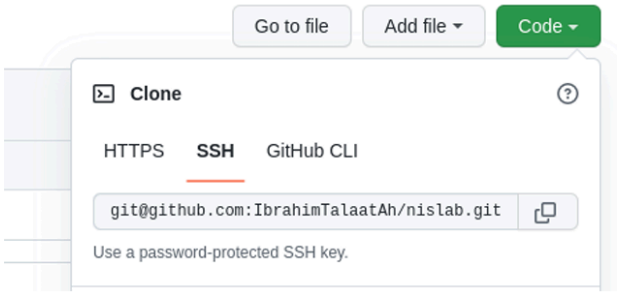
⚠️ **انتباه:** المسافات مهمة جدًا، السطرين الثاني والثالث يجب أن يبدأوا بمسافتين.

Clone Repository:

git clone [url copied]

example: git clone git@github.com:YourUsername/nislab.git

Clone the Repository
Now you should have GitHub.com open in a browser on both workstations. Navigate to the repository and click the green code button and copy the ssh link.



🧲 نسخ المستودع (Clone Repository)

✅ على Workstation1 و Workstation2:

1. ارجع لـ GitHub → المستودع `nislab`
2. اضغط على زر **Code** → اختر SSH → انسخ الرابط مثل:

Edit Copy scss

```scss
git@github.com:YourUsername/nislab.git
```

3. في الطرفية:

Edit Copy bash

```bash
git clone git@github.com:YourUsername/nislab.git
```

4. وافق على البصمة: اكتب `yes`
5. افحص:

Edit Copy bash

```bash
ls
cd nislab
ls
cat README.md
```

Configure user ID on both workstations:
We need to tell git who we are on both machines:

Workstation1$ git config -- global user.name "Admin1"
Workstation1$ git config -- global user.email "Admin1@nis.lab"
Workstation1$ cat ~/.gitconfig

Workstation2$ git config -- global user.name "Admin2"
Workstation2$ git config -- global user.email "Admin2@nis.lab"
Workstation2$ cat ~/.gitconfig

Testing version control

nano README.md

add this line: This line is written by admin1 on workstation1

then:

```
git status        # يعرض الملفات المعدّلة
git diff          # يعرض التعديلات بالتفصيل
git add README.md  # تجهيز الملف للإضافة
git commit -m "Admin1 edited readme file"
git push origin main  # إرسال التغييرات إلى GitHub
```

Workstation1~/nislab$ git status

Workstation1~/nislab$ git diff

Workstation1~/nislab$ git add README.MD
Workstation1~/nislab$ git status
Workstation1~/nislab$ git commit -m “Admin1 edited readme file”
Workstation1~/nislab$ git push origin main

In Workstation2:

شوف الملف الحالي:

Workstation2~/nislab$ cat README.MD
Workstation2~/nislab$ git status

رح تلاحظ إن التعديلات ما وصلت.

ف بسحب التغيرات عن طريق:

Workstation2~/nislab$ git pull
Workstation2~/nislab$ cat README.MD

🧪 الآن جرّب العكس (على Workstation2):

1. عدّل الملف وأضف:

Edit Copy nginx

```
Admin2 says hi
```

2. ثم:

Edit Copy bash

```bash
git add README.md
git commit -m "Admin2 added his line"
git push origin main
```

✅ على Workstation1:

Edit Copy bash

```bash
git pull
cat README.md
```

رح تلاحظ التعديلات الجديدة ظهرت 👍

Ansible [ad-hoc commands]

On both workstations:

sudo apt update
sudo apt install ansible :
هذا يثبّت أداة `ansible` اللي بنستخدمها لإرسال أوامر لكل السيرفرات من خلال SSH.

on workstation1:

إنشاء ملف Inventory : هذا الملف يحتوي على عناوين الايبي للسيرفرات التي نريد التحكم بها.

```
cd nislab
nano inventory
```

أضف عناوين IP لسيرفرات:
192.168.1.101
192.168.1.102

📌 كل IP في سطر منفصل.

```
Workstation1~/nislab$ git add inventory
Workstation1~/nislab$ git commit -m “Admin1 created inventory file and added SRVR01 and SRVR02”
Workstation1~/nislab$ git push origin main
```

on the second workstation2:

```
Workstation2~/nislab$ git pull
```

on workstation2 edit the inventory file and add the IP address of SRVR03 save and push to GitHub
with a comment “Admin2 modified the inventory and added SRVR03” then pull it on workstation1.

```
nano inventory
```

add this: 192.168.1.103
then:

```
git add inventory
git commit -m "Admin2 modified the inventory and added SRVR03"
git push origin main
```

In Workstation1:

git pull

To run a command on all servers:

Workstation1~/nislab$ ansible all --key-file ~/.ssh/ansible -i inventory -m ping

all: command will run on all servers
~/.ssh/ansible: key used to connect to the servers

-i inventory: server IP list

-m: module to use in this case ping

“The ping here is NOT ICMP ping it’s an ansible module that tests for a successful SSH connection to
each of the servers in the inventory list we created.”

🧪 رابعًا: أول تجربة لأمر Ansible

Edit Copy bash

```bash
ansible all --key-file ~/.ssh/ansible -i inventory -m ping
```

شرح الأمر:

- `all` : يعني شغّل الأمر على كل IPs في inventory
- `--key-file` : تحدد المفتاح الخاص SSH
- `-i inventory` : تحدد الملف الذي يحتوي على عناوين السيرفرات
- `-m ping` : تستخدم وحدة "ping" (ليست ping عادية، بل أنسبل يتأكد من اتصال SSH)

✅ إذا كانت الإعدادات صحيحة، ستحصل على رد من كل سيرفر شبيه بـ:

Edit Copy bash

```bash
192.168.1.101 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

This command is too long let's shorten it. To make it shorter we will store the inventory file name and
key to be used in a configuration file:
Workstation1
~/nislab$ nano ansible.cfg

input the following lines:

```
[defaults]

inventory = inventory
private_key_file = ~/.ssh/ansible
```

Now you can execute Ansible commands without retyping the options every time:

```
ansible all -m ping
```

Sync settings to sent it to Workstation2

```
git add ansible.cfg
git commit -m "Add ansible.cfg with inventory and SSH key config"
git push origin main
```

🔎 لمعرفة السيرفرات الموجودة في inventory:

```
ansible all --list-hosts
```

🔍 لجمع معلومات شاملة (facts) عن السيرفرات:

```
ansible all -m gather_facts
```

💥 الناتج ضخم جدًا. لتقييده لسيرفر واحد:

```
ansible all -m gather_facts --limit 192.168.1.101
```

🔍 لمعرفة نوع النظام:

```
ansible all -m gather_facts --limit 192.168.1.101 | grep ansible_distribution
```

سابقًا: تحديث الريبو (apt update)

❌ هذا الأمر لن ينجح:

```
ansible all -m apt -a update_cache=true
```

لأنه يحتاج صلاحيات `sudo`.

```
ansible all -m apt -a update_cache=true --become --ask-become-pass
```

- `-become`: مثل `sudo`
- `-ask-become-pass`: يطلب منك كلمة مرور sudo

```
ansible all -m apt -a "name=apache2" --become --ask-become-pass
ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass
```

🌐 ثامنًا: تثبيت apache2 على كل السيرفرات

Edit Copy bash

```bash
ansible all -m apt -a "name=apache2" --become --ask-become-pass
```

✅ افتح المتصفح:

Edit Copy cpp

```cpp
http://192.168.1.101
http://192.168.1.102
http://192.168.1.103
```

📌 سترى صفحة الترحيب بـ Apache.

✅ لتحديث الحزمة دائمًا (لو كانت موجودة أو لا):

Edit Copy bash

```bash
ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass
```

✅ لتحديث كل الحزم في النظام (يعادل apt dist-upgrade):

ansible all -m apt -a "upgrade=dist" --become --ask-become-pass

Package Management Using Ansible Playbooks

A playbook contains one or more tasks we want to execute. playbooks are written using the yaml language which is a human readable data-serialization language.

Yaml is usually used for configuration files.

✅ أولًا: ما هو Playbook؟

- الـ **playbook** هو ملف مكتوب بلغة YAML.
- يحتوي على **مجموعة من المهام (tasks)** نريد تنفيذها على أجهزة معينة.
- كل مهمة تسمى **"play"**.
- نكتب فيه "الحالة النهائية" التي نريد أن يكون عليها السيرفر (مثلاً: Apache يكون مثبت).

Create the first Playbook to install apache2

```
cd ~/nislab
nano install_apache.yml
```

```
---
- hosts: all
  become: true
  tasks:
    - name: install apache2 package
      apt:
        name: apache2
```

In the nislab directory on workstation1 create a file called install_apache.yml with the following contents:

```
---

- hosts: all
  become: true
  tasks:

  - name: install apache2 package
    apt:
      name: apache2
```

شرح كل سطر:

السطر	الشرح
`---`	بداية ملف YAML
`hosts: all -`	نفّذ المهام على جميع السيرفرات
`become: true`	فعّل صلاحيات sudo
`:tasks`	المهام التي ستُنفذ
`... name: install -`	وصف المهمة
`:apt`	استخدم وحدة APT لتثبيت الحزم
`name: apache2`	اسم الحزمة المطلوب تثبيتها

تشغيل الـ Playbook

ansible-playbook --ask-become-pass install_apache.yml

```
ansible-playbook --ask-become-pass install_apache.yml
```

The output will have a few important info when running the play on each host:

ok: this lists the number of plays that ran without problems on the host.

changed: the number of plays that made changes when ran on the host.

unreachable: if the host is offline.

failed: number of failed plays on this host.

skipped: number of plays that were skipped because the host did not meet the conditions for running this play.

rescued: number of plays that ran as a rescue because other plays failed to run.

ignored: number of ignored plays.

Update the repo first, then install (apt update + apt install)

عدّل الملف إلى التالي:

```
---
- hosts: all
  become: true
  tasks:
    - name: update repository index
      apt:
        update_cache: yes

    - name: install apache2 package
      apt:
        name: apache2
```

✅ ما هو `update_cache: yes` ؟

هو خيار داخل وحدة `apt` في Ansible، ويعني:

"اعمل `apt update` قبل تنفيذ هذه المهمة."

أي: حدث قائمة الروابط (URLs) والحزم المتوفرة من الإنترنت.

then: ansible-playbook --ask-become-pass install_apache.yml

Installing more than one package (Apache2 + PHP)
Update the file to read as follows:

```
CopyEdit
---
- hosts: all
  become: true
  tasks:
    - name: install Apache2 and PHP packages
      apt:
        name:
          - apache2
          - libapache2-mod-php
        update_cache: yes
```

then: ansible-playbook --ask-become-pass install_apache.yml

✅ هذا سيثبت الحزمتين، لكن لن يحدثهما إن كان هناك نسخة جديدة.

Ensure the latest version is installed. Add the line state: latest:

```
---
- hosts: all
  become: true
  tasks:
    - name: install Apache2 and PHP packages
      apt:
        name:
          - apache2
          - libapache2-mod-php
        update_cache: yes
        state: latest
```

- `state: latest`: يضمن تحديث الحزمة إن كانت قديمة.

بعد التشغيل، افتح متصفحك:

```
http://192.168.1.101
```

وشاهد صفحة Apache.

Create a Playbook to remove packages

nano remove_apache.yml

```
---
- hosts: all
  become: true
  tasks:
    - name: remove Apache2 and PHP packages
      apt:
```

```
    name:
      - apache2
      - libapache2-mod-php
    state: absent
```

- `state: absent`: تعني حذف الحزمة إن وُجدت.

ansible-playbook --ask-become-pass remove_apache.yml

👍 ثم جرّب فتح السيرفر من المتصفح → لن تجد صفحة Apache
أعد تشغيل install_apache.yml لترجع الصفحة.

---

git status
git add .
git commit -m "install/remove apache and php playbooks created by admin 1"
git push origin main

git pull

🔄 ثامنًا: مزامنة الـ Playbooks مع GitHub

✅ على Workstation1:

Edit Copy bash

```bash
git status
git add .
git commit -m "install/remove apache and php playbooks created by admin 1"
git push origin main
```

✅ على Workstation2:

Edit Copy bash

```bash
git pull
```

ستظهر الملفات `install_apache.yml` و `remove_apache.yml` في الجهاز الثاني.

The 'when' Conditional

🧠 لماذا نحتاج `when` في Ansible؟

في المثال السابق كنا نستخدم:

Edit Copy yaml

```yaml
apt:
  name: apache2
```

لكن:

- بعض السيرفرات مثل **Ubuntu** تعتمد على **APT** (Debian-based).
- بينما **AlmaLinux** تعتمد على **DNF** (RHEL-based).

⚠️ النتيجة:

إذا شغّلت نفس المهمة على AlmaLinux، رح تفشل لأنه ما في شيء اسمه `apt` عندها.

✅ الحل؟ استخدم شرط `when`

أنسبل توفر شرط اسمه `when:` تقدر من خلاله تقول:

"نفذ المهمة **فقط** إذا تحقق شرط معين".

🛠️ الخطوة 1: أضف IP سيرفر AlmaLinux إلى ملف `inventory`

على Workstation2:

Edit Copy bash

```bash
nano inventory
```

أضف IP الجديد في سطر منفصل، مثل:

Edit Copy

```
192.168.1.104
```

ثم:

Edit Copy bash

```bash
git add inventory
git commit -m "Added AlmaLinux server to inventory"
git push origin main
```

واعمل `git pull` على Workstation1.

🧪 الخطوة 2: تحقق من نوع النظام لكل سيرفر

```
ansible all -m gather_facts | grep ansible_distribution
```

هذا يعطينا معلومة نستخدمها داخل `when`.

الخطوة 3: تعديل الـ Playbook ليعمل فقط على الأنظمة المتوافقة

افتح ملف الـ Playbook وعدّل ليكون:

```
---
- hosts: all
  become: true
  tasks:
```

```yaml
- name: install Apache2 and PHP packages on Ubuntu
  apt:
    name:
      - apache2
      - libapache2-mod-php
    update_cache: yes
    state: latest
  when: ansible_distribution == "Ubuntu"
```

✅ **النتيجة:**

- أنسبل سيشغّل المهمة **فقط** على الأنظمة اللي فيها:

Edit Copy yaml

```yaml
ansible_distribution == "Ubuntu"
```

- وبالنسبة لـ AlmaLinux: المهمة راح تنكتب في الخرج كـ "skipped" ✅

🧩 الخطوة 4: أضف مهمة مخصصة لـ AlmaLinux

لتثبيت Apache و PHP على AlmaLinux، نستخدم `dnf` بدل `apt`.

الملف النهائي يصير:

```yaml
---
- hosts: all
  become: true
  tasks:

  - name: install Apache2 and PHP packages on Ubuntu
```

```
    apt:
      name:
        - apache2
        - libapache2-mod-php
      update_cache: yes
      state: latest
    when: ansible_distribution == "Ubuntu"

  - name: install httpd and PHP packages on AlmaLinux
    dnf:
      name:
        - httpd
        - php
      update_cache: yes
      state: latest
    when: ansible_distribution == "AlmaLinux"
```

ansible-playbook --ask-become-pass install_apache.yml

✨ النتيجة المتوقعة:

- Ubuntu: يتم تثبيت `apache2` و `libapache2-mod-php`
- AlmaLinux: يتم تثبيت `httpd` و `php`
- السيرفرات غير المتوافقة مع الشرط → يتم **تخطي المهمة** دون فشل.

✅ هل كل شيء تمام؟

- `when` يسمح لك بكتابة Playbook ذكي يتصرف حسب نوع النظام.
- استخدم `gather_facts` للحصول على المتغيرات التي يمكنك استخدامها في `when`.