LU10a - Setup: VirtualBox & Vagrant installieren

Ziel: Du kannst die nötigen Tools (**VirtualBox**, **Vagrant**) korrekt installieren, die **Hardware-Virtualisierung** prüfen/aktivieren und ein erstes **Vagrant-Testprojekt** starten, um das Setup zu verifizieren.

Übersicht

Bevor wir mit **Multi-Node Labs** (LU10b) und **Provisioning mit Ansible** (LU10c) starten, muss dein Rechner als „Host“ bereit sein:

- Virtualisierung ist aktiv (CPU/BIOS/UEFI)
- VirtualBox läuft
- Vagrant ist installiert und im Terminal verfügbar
- Ein Test-VM-Start funktioniert

Was muss installiert sein?

Bestandteil	Pflicht?	Wozu?
VirtualBox	Ja	Virtualisierung (Provider) für unsere VMs
VirtualBox Extension Pack	Optional	Zusätze wie USB/weitere Features (nicht zwingend für Basics)
Vagrant	Ja	VMs als Code (Vagrantfile), Multi-VM-Labs, Lifecycle-Commands
Git	Optional	Projekte/Unterlagen klonen, Versionsverwaltung
Editor (VS Code o.ä.)	Optional	Vagrantfile/Playbooks bearbeiten

Wichtig: Für LU10b verwenden wir (empfohlen) **ansible_local**. Dann muss Ansible **nicht** auf deinem Host installiert sein, sondern wird in der VM genutzt.

Schritt 0: Hardware-Virtualisierung prüfen (sehr häufige Fehlerquelle)

Windows (10/11)

- Task-Manager → **Leistung** → **CPU** → „**Virtualisierung: Aktiviert**“
- Wenn dort „Deaktiviert“ steht: im BIOS/UEFI einschalten (Intel VT-x / AMD-V)

macOS

- Intel-Macs: Virtualisierung ist in der Regel ok.
- Apple Silicon (M1/M2/M3…): VirtualBox Support ist erst in neueren Versionen da – achte darauf, dass du eine passende VirtualBox-Version verwendest.

Linux

- Quick-Check im Terminal:

```
egrep -c '(vmx|svm)' /proc/cpuinfo
```

- Ergebnis > 0 bedeutet: CPU unterstützt Virtualisierung.

Wenn Virtualisierung im BIOS/UEFI aus ist, funktionieren VMs oft gar nicht oder extrem langsam. Das ist der #1-Showstopper.

Schritt 1: VirtualBox installieren

Download (offiziell)

- VirtualBox von der offiziellen Download-Seite beziehen.
- Nach Installation: VirtualBox einmal starten.

Windows

```
1) Installer ausführen
2) Standard-Optionen ok
3) Nach der Installation VirtualBox starten → es soll ohne Fehlermeldung
öffnen
```

macOS

- Intel & Apple Silicon: achte darauf, dass du das richtige Paket für deine Plattform installierst.
- Falls macOS Sicherheitsdialoge zeigt: System-Einstellungen → Datenschutz/Sicherheit → Systemerweiterung erlauben (falls verlangt).

Linux

- Je nach Distribution gibt es Pakete/Repos.
- Wichtig: Kernel-Module müssen korrekt gebaut/geladen sein (Fehler zeigen sich beim Start der

VM).

Schritt 2 (optional): VirtualBox Extension Pack

Das Extension Pack kann nützlich sein, ist für unseren Standard-Workflow aber meist nicht nötig.

Für Unterricht/Labs gilt: **Installiere das Extension Pack nur, wenn du es wirklich brauchst** (z.B. USB-Passthrough).

Schritt 3: Vagrant installieren

Installation (offiziell)

- Installiere Vagrant mit dem offiziellen Installer für dein OS.
- Danach Terminal neu öffnen (damit PATH/Umgebungsvariablen sauber geladen sind).

Verifikation

```
vagrant --version
```

Wenn du eine Versionsausgabe bekommst, ist Vagrant korrekt installiert.

Wenn `vagrant` „nicht gefunden“ wird: Terminal neu starten oder ab-/anmelden. Unter Windows ist das der häufigste Stolperstein.

Schritt 4: Testprojekt erstellen (Smoke Test)

Wir erstellen ein Minimalprojekt und starten eine VM.

1) Projektordner anlegen

```
mkdir vagrant-test
cd vagrant-test
```

2) Vagrantfile initialisieren

Variante A (Standard-Box, wenn verfügbar):

```
vagrant init ubuntu/jammy64
```

3) VM starten

```
vagrant up
```

4) Einloggen

```
vagrant ssh
```

5) In der VM prüfen

```
uname -a
ip a
ls -la /vagrant
```

6) Beenden

```
exit
vagrant halt
```

Erwartung:

- `vagrant up` lädt beim ersten Mal eine Box (kann dauern)
- `vagrant ssh` funktioniert
- `/vagrant` enthält deinen Projektordner

Schritt 5: Häufige Fehler & Fixes

Problem: „VT-x/AMD-V not available“ oder VM startet nicht

Ursache: Virtualisierung im BIOS/UEFI aus (oder Hypervisor-Konflikt). Fix:

- BIOS/UEFI: Intel VT-x / AMD-V aktivieren
- Danach neu starten

Problem (Windows): Hyper-V / WSL / Windows Hypervisor blockiert VirtualBox

Symptome:

- VM startet extrem langsam oder gar nicht
- Fehlermeldungen rund um Hypervisor

Fix-Ideen (je nach Schulgerät/Policy):

- Windows-Features: Hyper-V / Virtual Machine Platform / Windows Hypervisor Platform prüfen
- Falls nötig: deaktivieren (und neu starten)

In Schulumgebungen kann das durch Policies eingeschränkt sein. Wenn du keine Adminrechte hast: früh melden, damit wir eine Lösung finden (Alternativ-Setup / Schulgeräte / Remote-VM).

Problem: „Port already in use“

Ursache: Port-Forwarding kollidiert mit einem Dienst auf deinem Host. Fix:

- anderen Host-Port wählen
- oder den belegenden Dienst stoppen

Problem: Download/Box-Fehler

Fix:

- Internet/Proxy prüfen
- `vagrant box list` ansehen
- Box neu laden:

```
vagrant box remove ubuntu/jammy64
vagrant up
```

Apple Silicon Hinweis (M1/M2/M3/…)

Auf Apple Silicon ist Virtualisierung grundsätzlich möglich – aber:

- Du brauchst eine VirtualBox-Version, die Apple Silicon unterstützt.
- Deine Vagrant-Box muss zur Architektur passen (ARM64 vs x86_64).

Wenn du auf Apple Silicon Probleme hast, melde dich früh: Dann entscheiden wir gemeinsam, ob du

- mit kompatiblen ARM-Boxen arbeitest,
- oder einen alternativen Provider/Remote-Setup nutzt.

Abgabe/Checkliste (Selbstkontrolle)

Check	OK?
Virtualisierung aktiv (Windows Task-Manager / Linux vmx/svm)	☐
VirtualBox startet ohne Fehler	☐
`vagrant –version` funktioniert	☐
`vagrant up` im Testprojekt klappt	☐
`vagrant ssh` klappt	☐
`/vagrant` ist in der VM vorhanden	☐

M321-LU10