Autom

Livraison et déploiement continu, livre de jeux

C’est quoi ?

- C’est le livre de jeux associé au support de formation « Livraison et déploiement continu ».
- Le support de formation ne contient pas de guide « pas à pas », et laisse quelques zones de présentation ouvertes, fonction de l’auditoire (les quelques encarts « On joue »). Le présent support contient des guides « pas à pas » et des éléments de jeux associés aux encarts.

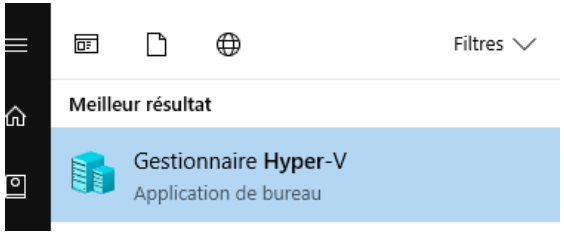
Infrastructure virtuelle

Slide 30 ou approchant et suivants

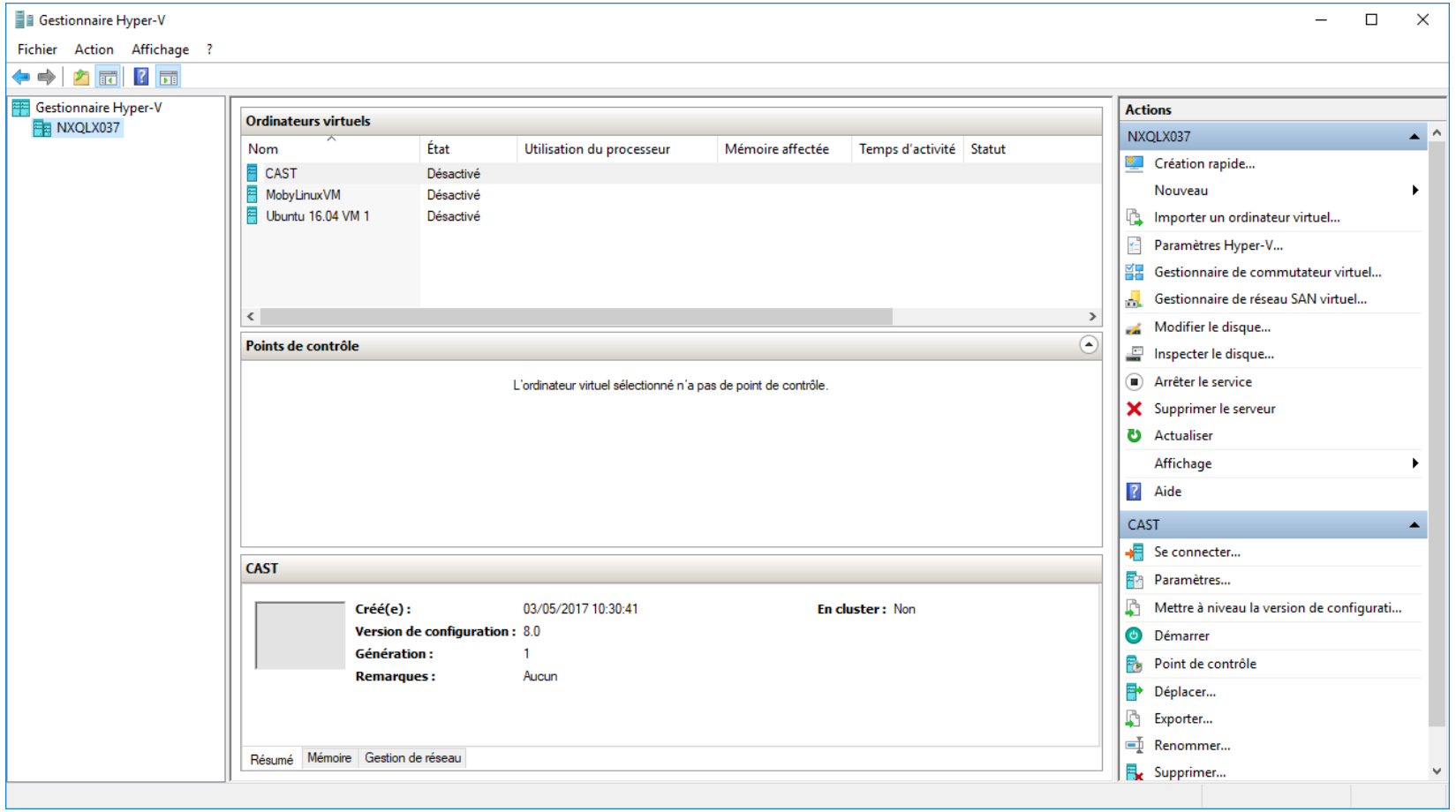
Installation d’Hyper-V

- https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/enable-hyper-v
- https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/index

Le gestionnaire Hyper-V

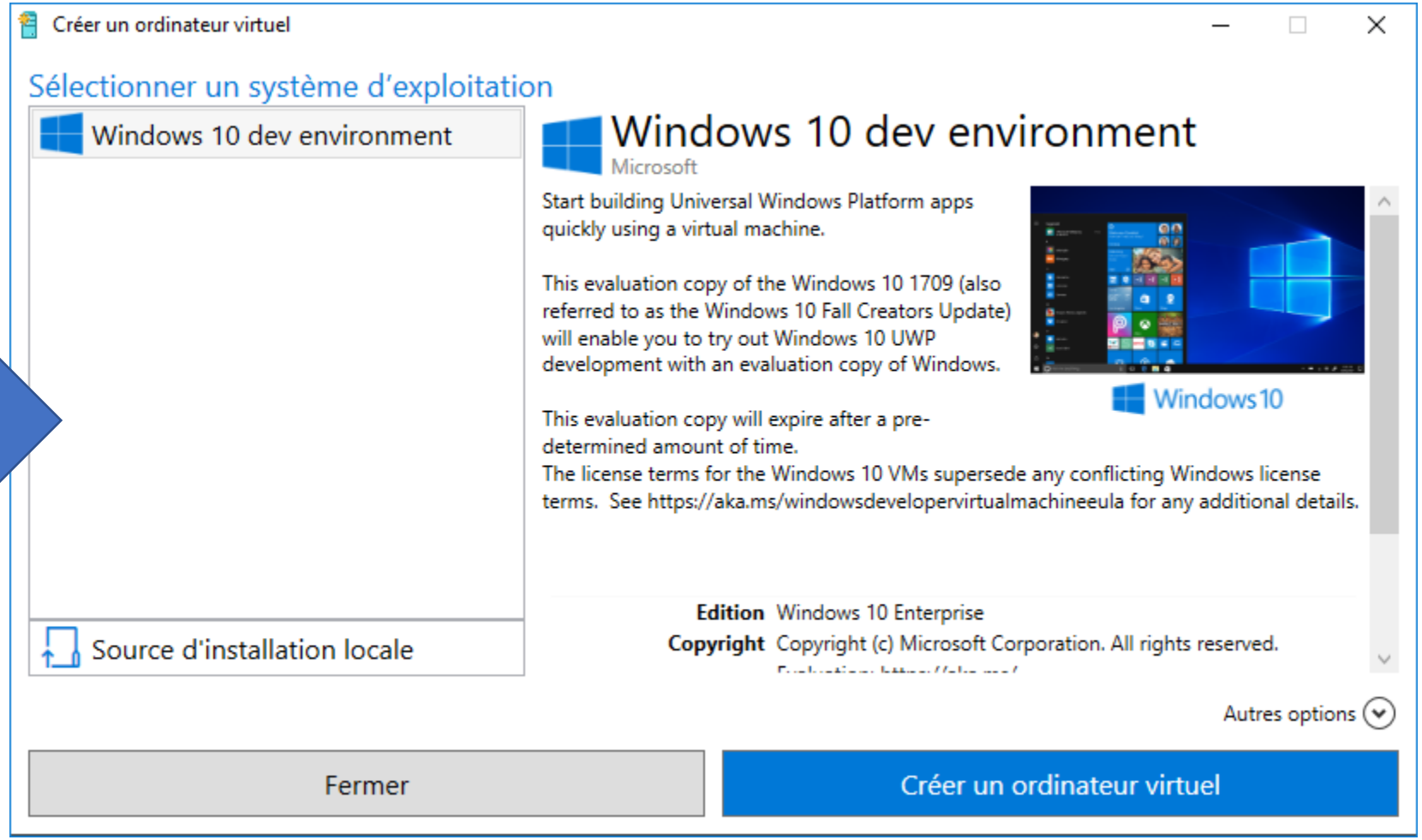


En mode administrateur (si vous ne l'êtes pas déjà, clic droit et choisir « Exécuter en tant qu'administrateur »)



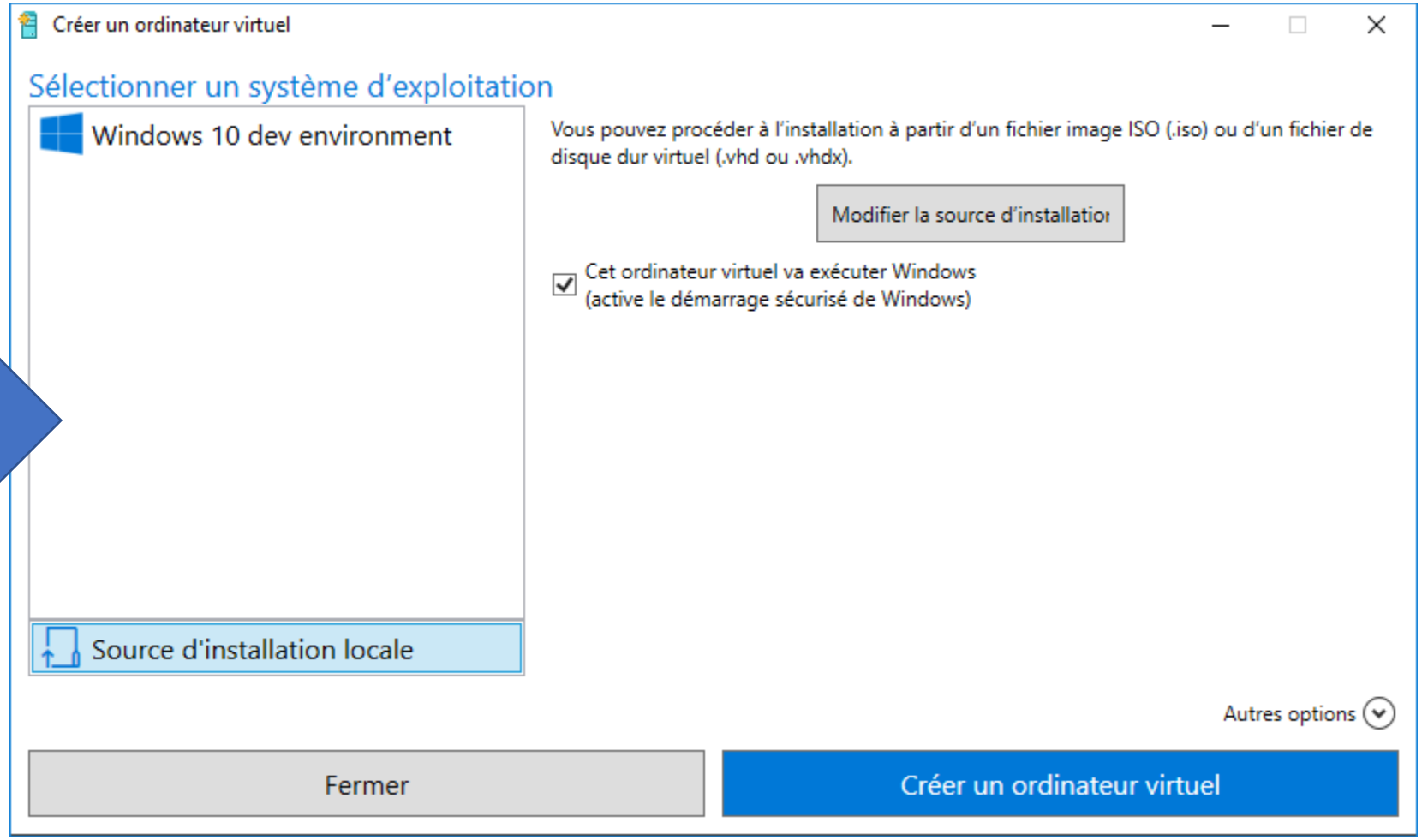
Créer une nouvelle machine virtuelle

Choisir « Création rapide… » dans le menu (ou dans le bandeau à droite)



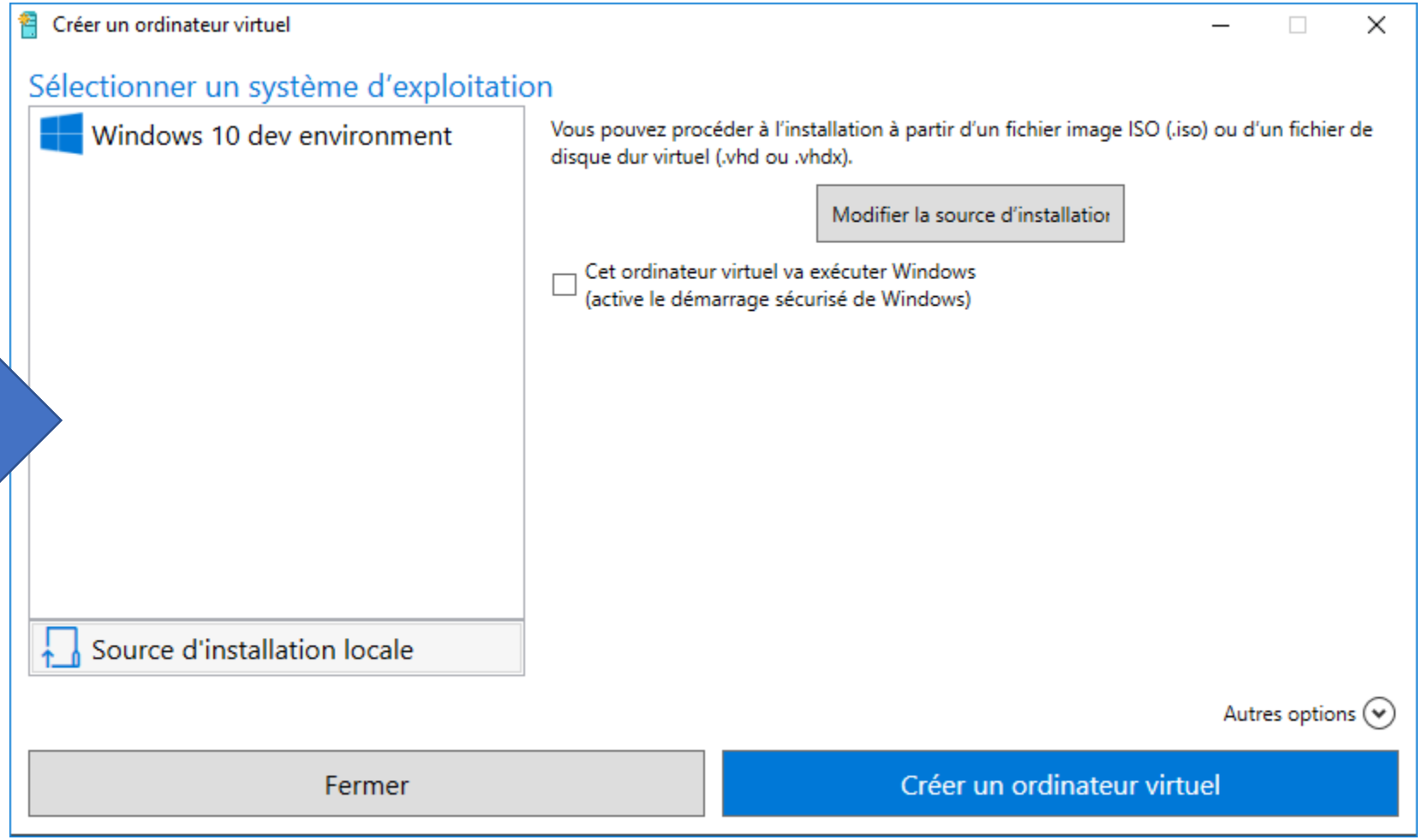
Créer une nouvelle machine virtuelle

Puis choisir « Source d'installation locale »



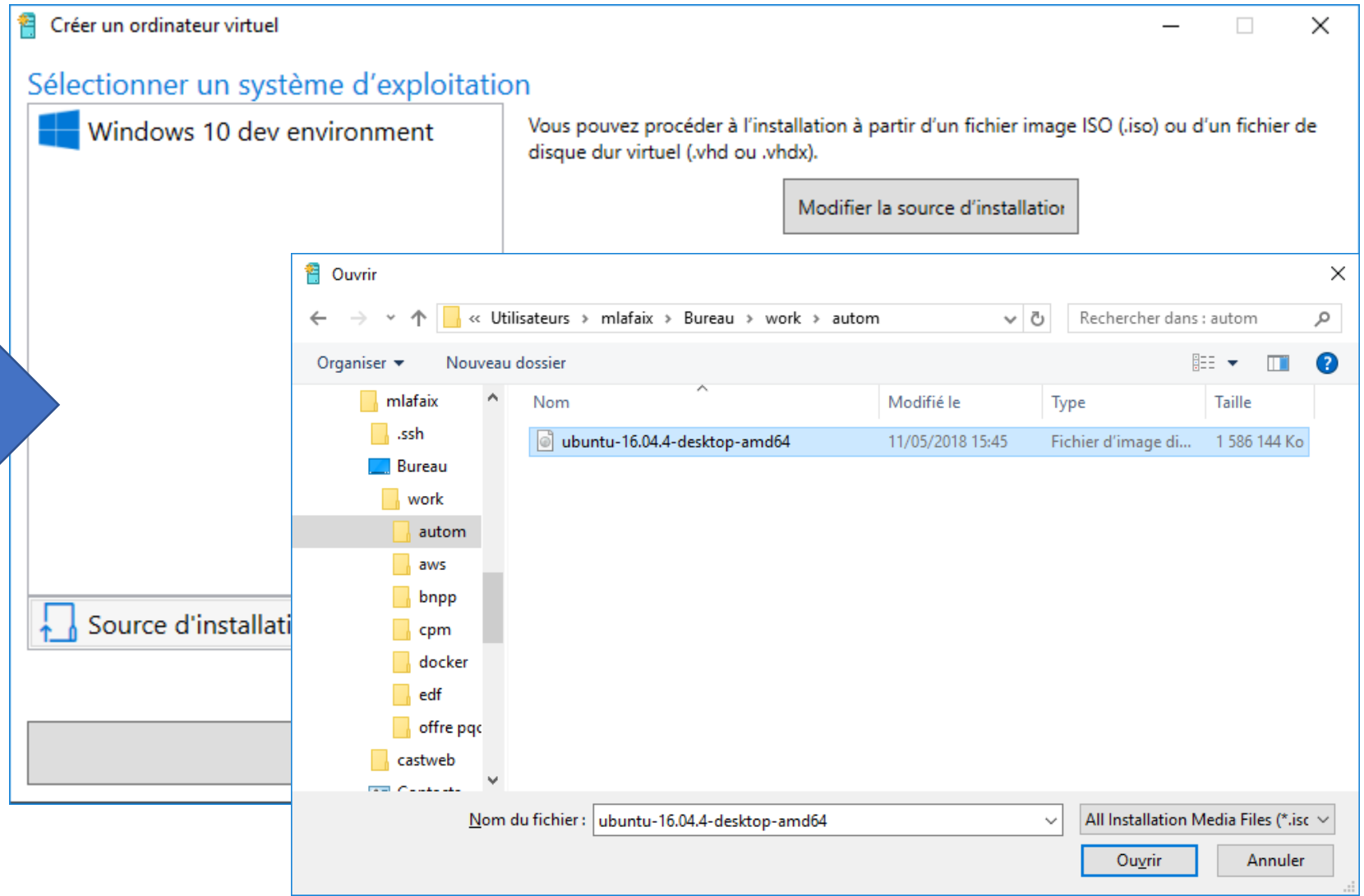
Créer une nouvelle machine virtuelle

Décocher « Cet ordinateur virtuel va exécuter Windows (…) »



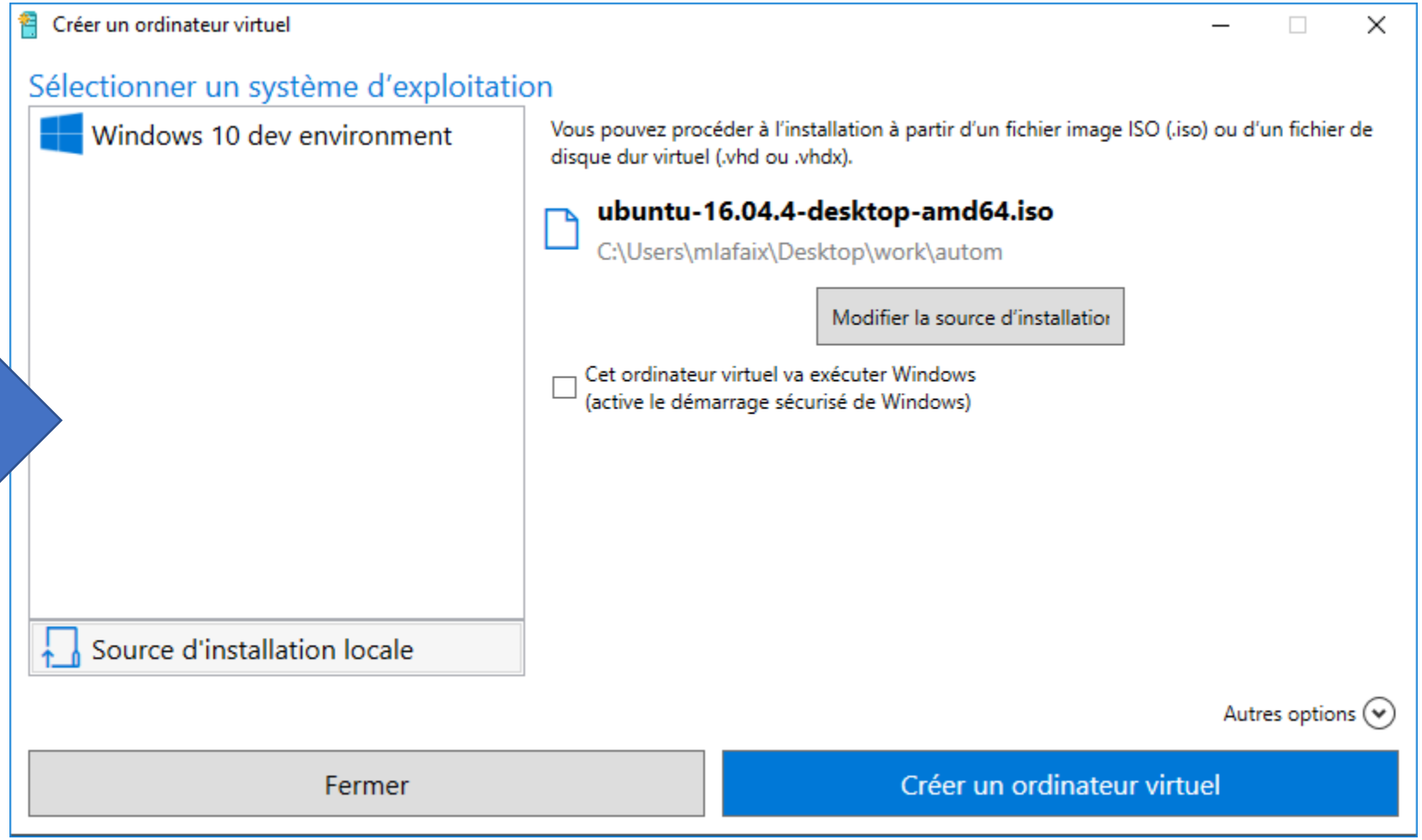
Créer une nouvelle machine virtuelle

Modifier la source d'installation et trouvez l'ISO qui vous intéresse



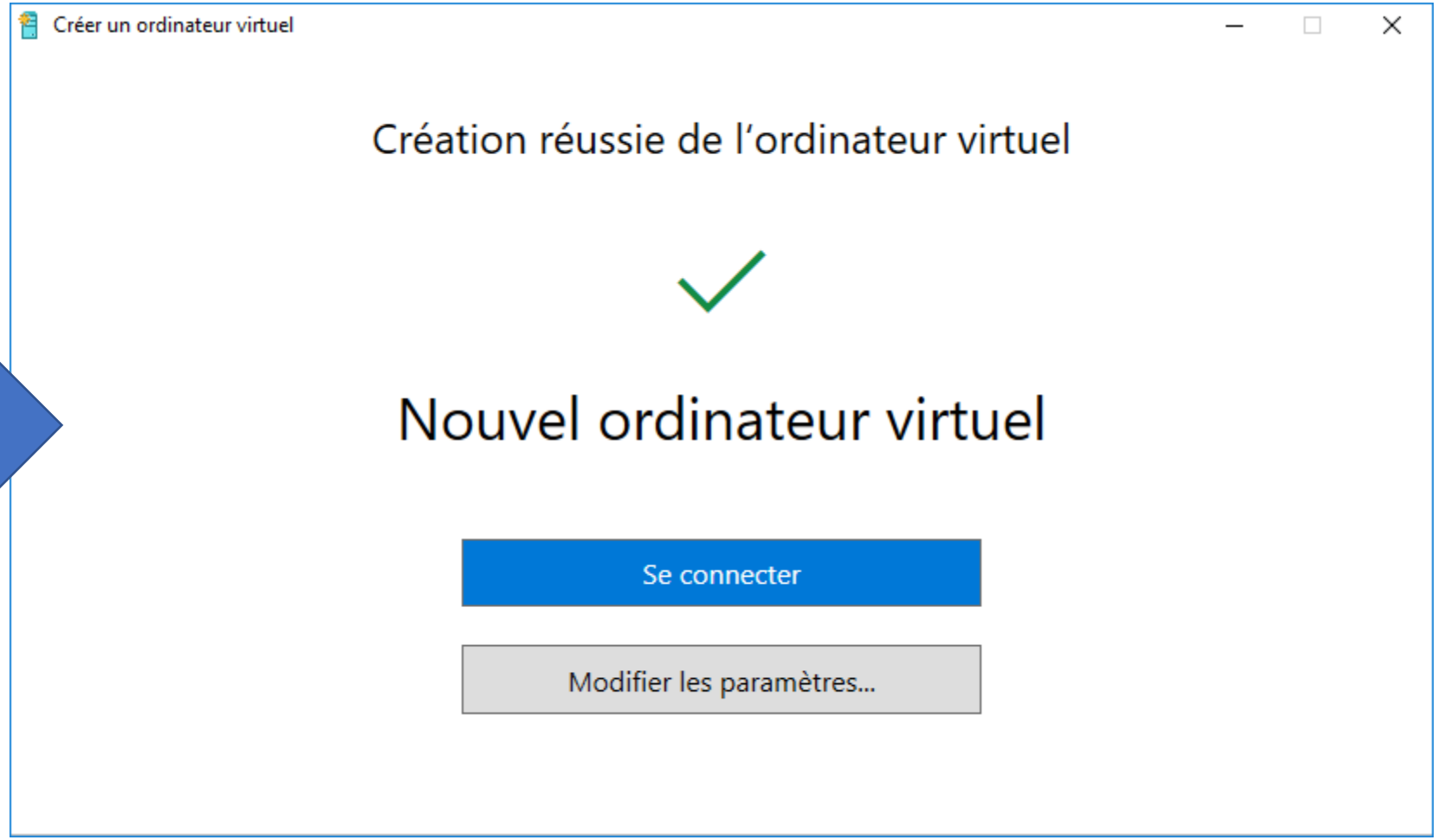
Créer une nouvelle machine virtuelle

Et créer l'ordinateur virtuel.



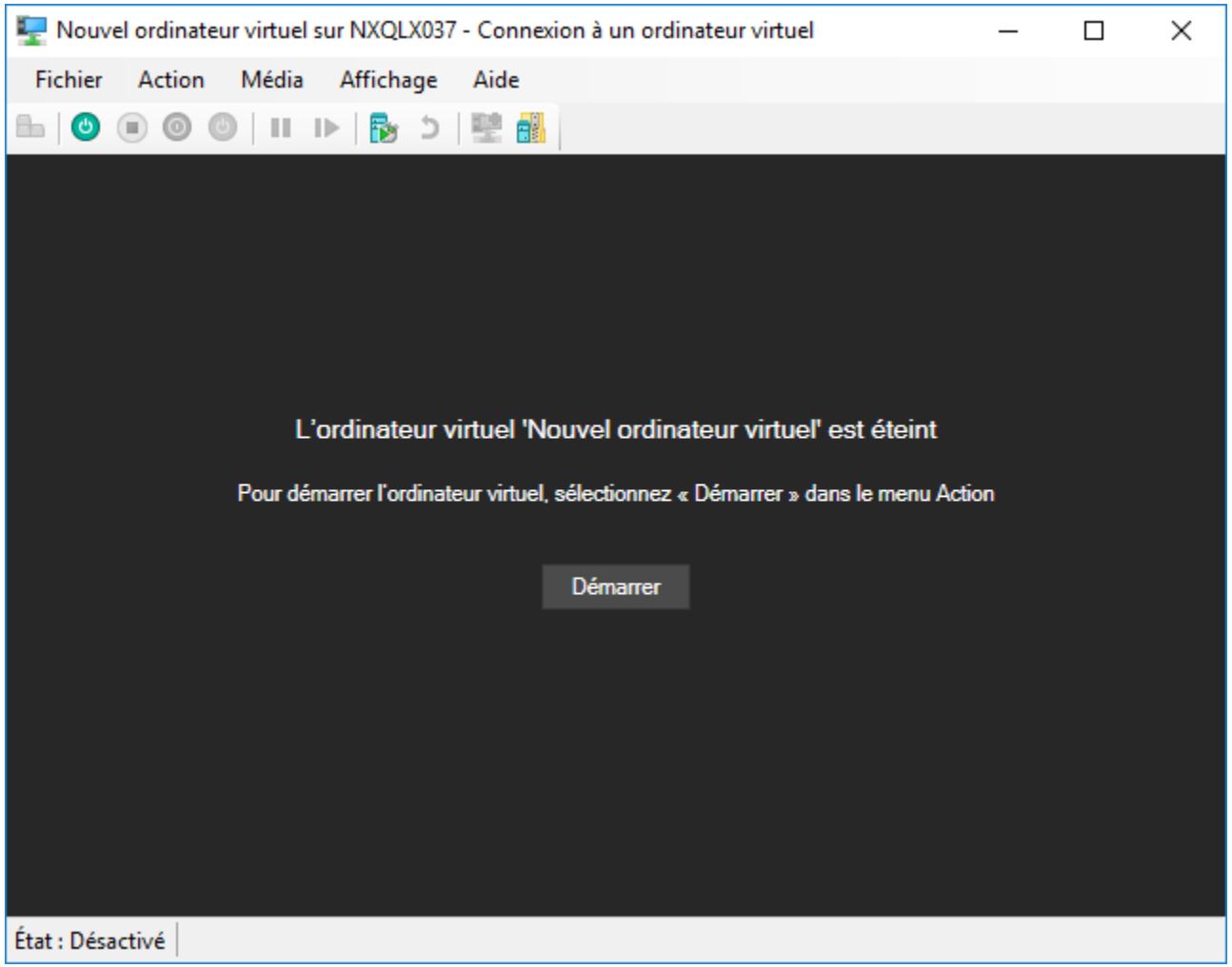
Créer une nouvelle machine virtuelle

Une fois l'ordinateur créé, s'y connecter.



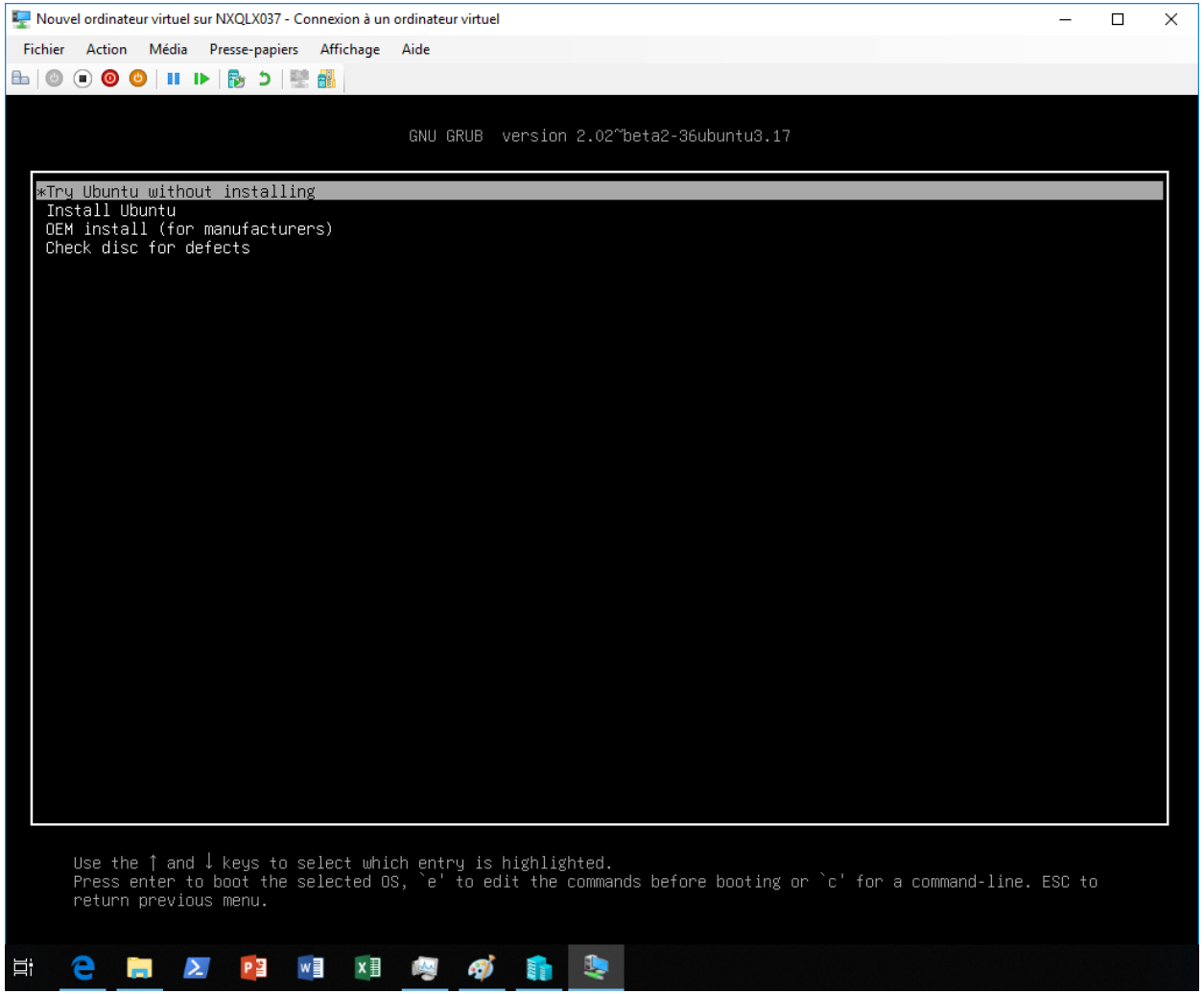
Créer une nouvelle machine virtuelle

Le démarrer.



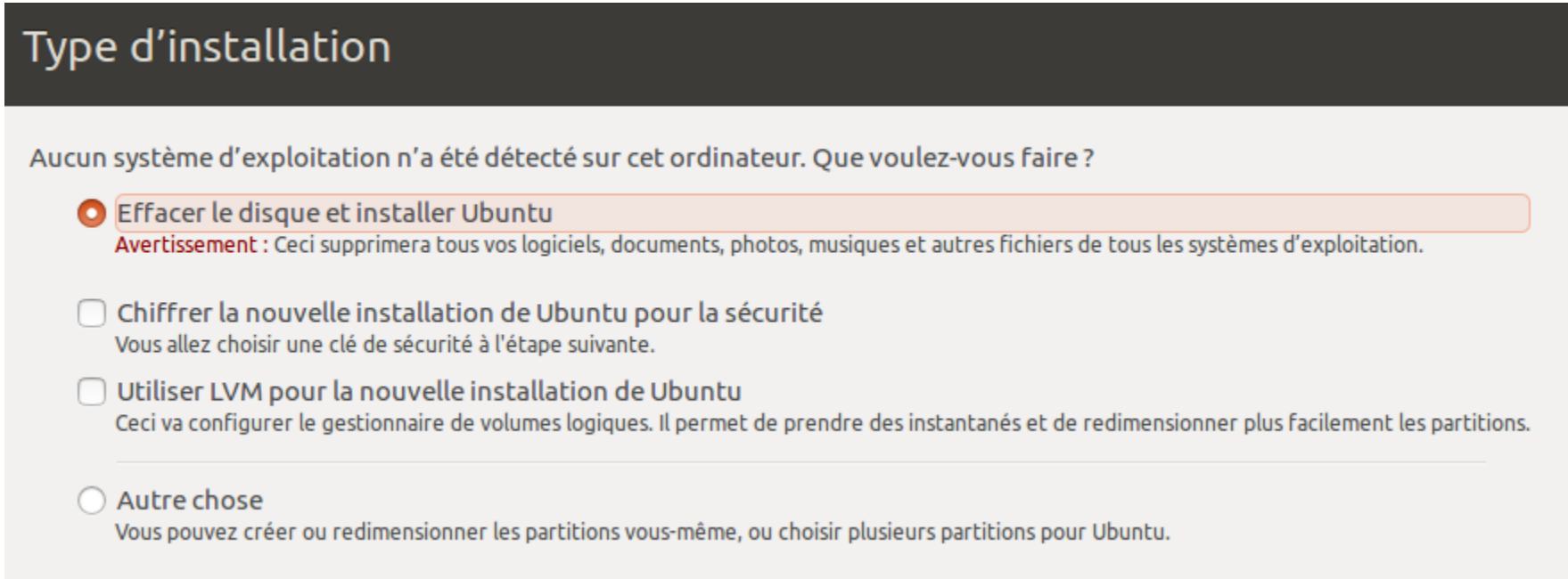
Créer une nouvelle machine virtuelle

Installer Ubuntu.



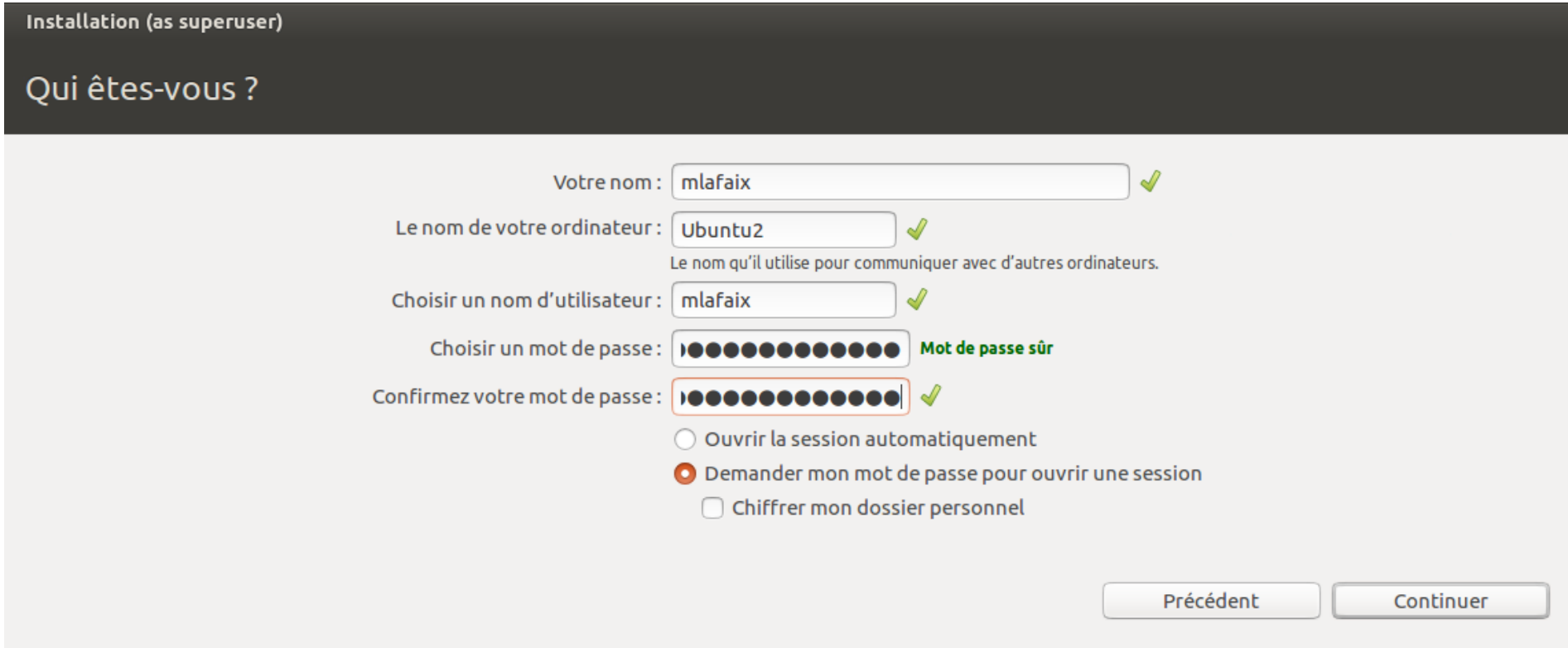
Installer Ubuntu

- Choisir la langue que vous voulez (et que vous comprenez ☺)
  (à la souris ou au clavier)
- Demander le téléchargement des mises à jour (deuxième écran)
- Type d'installation tel quel, et accepter la suite



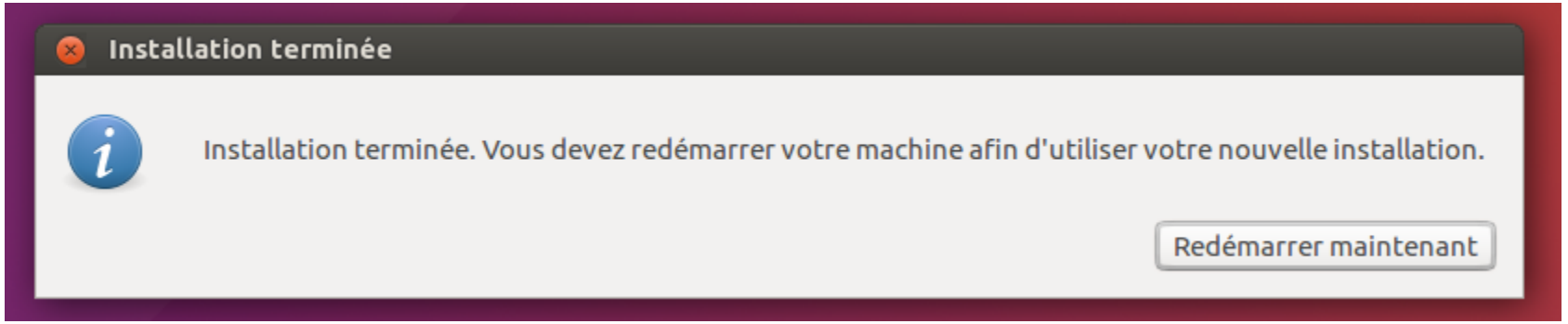
Installer Ubuntu

- Donnez les identifiants du compte que vous allez créer (dont vous allez vous souvenir ☺)

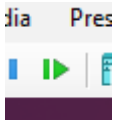


Installer Ubuntu

- L’installation s’effectue ensuite sans intervention. Compter 20 bonnes minutes. Il faut redémarrer à la fin (une fenêtre vous y invite)



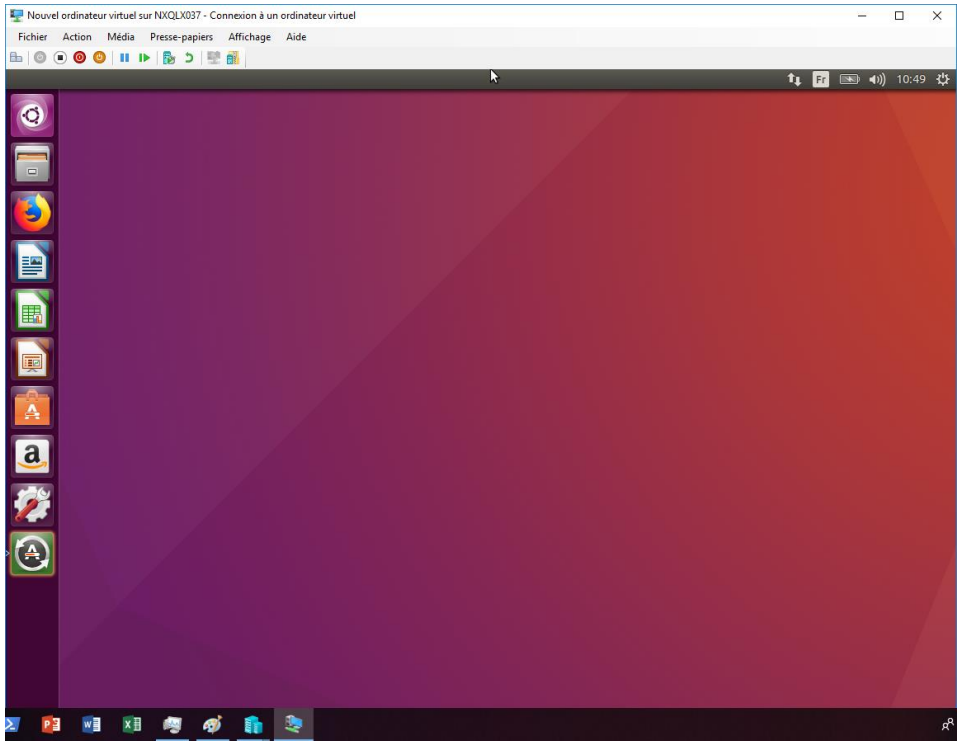
- Si le système vous demande d’enlever le support, réinitialisez la VM



Installer Ubuntu

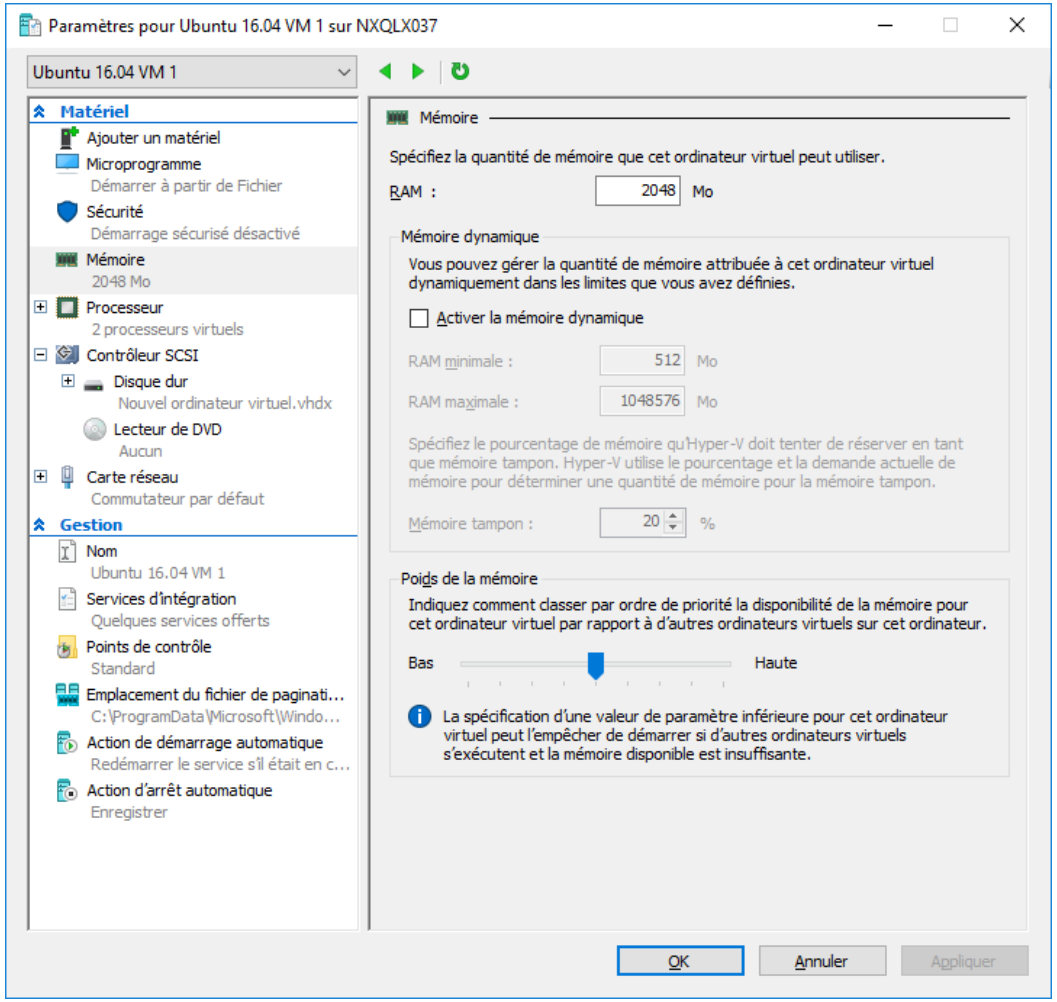
- Entrez votre mot de passe définit plus haut.
- La première connexion prendra un peu plus de temps que les suivants (de l'ordre d'une à deux minutes, vs. quelques secondes)
- Voilà, c'est fini, la VM est prête.

[Pour l'éteindre, menu en haut à droite]



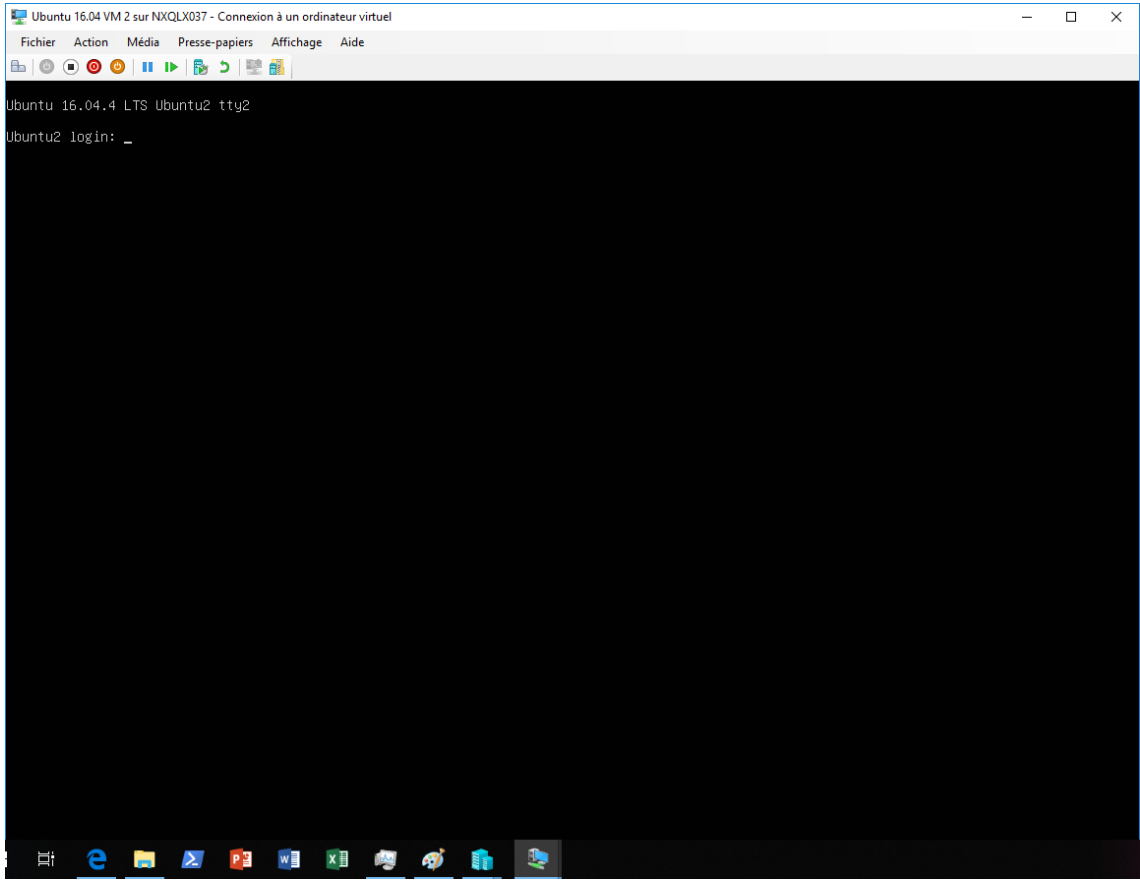
Limiter la consommation mémoire

- Par défaut, la mémoire allouée à chaque machine virtuelle est allouée dynamiquement.
- Comme nos postes ont une quantité limitée de mémoire, vous allons limiter la taille de la mémoire de chacune de nos VM à 2 Go.
- Machine virtuelle arrêtée, ouvrir les paramètres, section « Mémoire ».



Bypasser l’interface graphique

- L’interface graphique peut être … lente.
- Ce sont des machines virtuelles. Donc de « vrais » ordinateurs, donc ce qu’on peut faire sur un vrai ordinateur on peut le faire sur la VM. Donc on peut faire Ctrl+Alt+F1 ☺ (ou F2 ou …, avec F7 pour revenir à la session graphique).



SDN

Réseau publique, réseau privé

Gnome Boxes

- Gnome Boxes est une interface simple. Très simple.
- Parfois, on a besoin de plus.

- Heureusement, nous ne sommes pas forcément obligés d'utiliser une ligne de commande ou la modification de fichiers abscons.

- Il existe d'autres interfaces. Moins jolies, mais qui permettent plus de chose. Le Gestionnaire de machine virtuelle, par exemple (virt-manager).

Bridges

- https://mike42.me/blog/2019-08-how-to-use-the-qemu-bridge-helper-on-debian-10

(Avec quelques erreurs, en particulier sur les droits du fichier /etc/qemu/bridges.conf, qui doivent être en 644, pas 640)

NAT-Network

Commutateur virtuel (Virtual Switch)

Déploiement sur des conteneurs

Slide 62 ou approchant

Déploiement sur des conteneurs

Slide 84 ou approchant

Orchestration

Slide 100 ou approchant

Des fichiers de configuration

- On peut vouloir ajouter des fichiers de configuration à un (ou plusieurs) conteneur(s)

- Ces fichiers pourront être communs à plusieurs services

- (Une autre solution est de faire un mapping vers un répertoire local qui contient lesdits fichiers de configuration)

[Version 3.3 et ultérieure uniquement, et uniquement avec swarm]

Des ensembles de variables d’environnement

- On peut (re)définir explicitement la valeur d’une variable d’environnement déclarée dans un Dockerfile via le fichier docker-compose.yml (section ‘environment’).
- Mais on peut aussi le faire via des fichiers (section ‘env_file’).

- Plus pratique lorsque les valeurs sont partagées entre plusieurs services, ou si elles sont nombreuses, ou si elles dépendent de l’environnement.

Des liens vers des conteneurs extérieurs

- La section ‘links’ (et la section ‘depends_on’) ne peuvent faire référence qu’à des services décrits dans le docker-compose.yml.

- La section ‘external_links’ permet de se lier à des conteneurs démarrés hors du présent docker-compose.yml.

- On peut définir des alias, comme avec ‘links’ (CONTENEUR:ALIAS).

Vérifier la bonne santé des services

- La section ‘healthcheck’ permet de définir quand un service est malade.
  - Comment tester l’état d’un conteneur (section ‘test’)
  - À quel intervalle (‘interval’), après quelle attente (‘timeout’), après combien de tentatives (‘retries’), après une période de grâce (‘start_period’).

[Version 2.1 et ultérieures, 3.4 pour la section ‘start_period’.]

Relancer (ou pas) un service

- Si le service échoue, on peut le redémarrer automatiquement.
- Section ‘restart’.
- Valeurs possibles :
  - « no »
  - always
  - on-failure
  - unless-stopped

Oui mais mon service est juste ‘malade’…

- Pas géré à ce jour (discussions en cours).

- Mais un palliatif (☺) :

```
version: '2’
services:
 autoheal:
  restart: always
  image: willfarrell/autoheal
  environment:
   - AUTOHEAL_CONTAINER_LABEL=all
  volumes:
   - /var/run/docker.sock:/var/run/docker.sock
```

Logs, logs, logs

- Je peux laisser les logs sur la console, mais c'est vite illisible (et encore, nous n'avons que 2 services dans nos stacks).

- Je peux utiliser 'docker logs <conteneur>', mais c'est de l'interrogation *a posteriori*.

- Section 'logging'. Quelques modes (« json-file », « syslog », « none »), et la possibilité de définir des options.

Substitutions

- On peut utiliser des variables d’environnement dans le fichier docker-compose.yml.

- Par exemple:

```
db:
 image: "mysql:${MYSQL_VERSION}"
```

L’abus de YAML peut être fatal

- Des ancres, des jointures, aïe !

```
version: '2.1'
x-volumes:
  &default-volume
  driver: foobar-storage

services:
  web:
    image: myapp/web:latest
    volumes: ["vol1", "vol2", "vol3"]
volumes:
  vol1: *default-volume
  vol2:
    << : *default-volume
    name: volume02
  vol3:
    << : *default-volume
    driver: default
    name: volume-local
```

Et si je veux lancer plusieurs fois ma stack ?

- -p -p -p

Concours

De la plus petite…

- … image qui fait ce qu'on veut (initialisation si besoin, et lancement de squash, et qui fonctionne ☺)

- [SANS toucher au tar.gz décompressé de Squash TM ☺]

Sous-réseaux

Sous-réseaux

- En production, souvent, les diverses couches applicatives sont dans des sous-réseaux différents (afin de les isoler).

- La couche IHM peut accéder à la couche métier, mais pas à la couche BDD.

Sous-réseaux

- Par défaut, l'ensemble d'une stack est déployée dans son propre sous-réseau (qui a pour nom <project>_default).

- Si une stack est déployée plusieurs fois (avec l'option –p, par exemple), chaque déploiement sera dans un sous-réseau distinct.

- C'est parfait dans une chaîne CI/CD (dans l'essentiel des cas). Mais, si on souhaite utiliser le même mécanisme de déploiement en production, ce n'est pas idéal.

Sous-réseaux

- On peut, heureusement, définir plusieurs réseaux, et attacher les services aux seuls réseaux pertinents.

- La couche métier, dans l'exemple, voit les services des deux sous-réseaux (et c'est heureux, car elle doit être visible de la partie IHM et pouvoir accéder à la BDD).

```
version: "3"

services:
  proxy:
    build: ./proxy
    networks:
      - frontend
  app:
    build: ./app
    networks:
      - frontend
      - backend
  db:
    image: postgres
    networks:
      - backend

networks:
  frontend:
    # Use a custom driver
    driver: custom-driver-1
  backend:
```

Sous-réseaux

- Un sous-réseau peut être « privé » à la stack (i.e., seule cette stack y aura accès, et, si on déploie plusieurs instances de cette stack, des sous-réseaux séparés seront créés).

- Mais on peut aussi spécifier un (ou plusieurs) sous-réseaux partagés.

- Dans ce cas, ils seront définis hors de la stack, et, dans celle(s)-ci, ils seront déclarés comme « external »

```
networks:
  default:
    external:
      name: my-pre-existing-network
```

Mises à jour d’une stack

- Lorsqu’on modifie une stack (i.e., on modifie le fichier docker-compose.yml), il n’est pas nécessaire de détruire la stack et de la redéployer.

- Rejouer la commande « docker-compose up » va supprimer les conteneurs modifiés et les relancer (et se faisant le service redémarré va très probablement changer d’adresse IP, ce que certains services utilisateurs n’aiment pas, d’où le comportement par défaut de la commande docker-compose up qui va aussi redémarrer les services dépendants, même s’ils n’ont pas été modifiés).

Mise à jour d’une stack

- D’où des options intéressantes pour docker-compose up :
  - --no-deps : ne (re)démarre pas les services liés
  - --no-recreate : ne recréé pas les services s’ils existent déjà (i.e., sont déployés)
- Et on peut aussi spécifier explicitement les services à (re)démarrer :
  - docker-compose up -d mysql squashtm2
  - docker-compose up -d --no-deps squashtm1

Mini projet

SquashTM en SaaS

- Par équipe, sur les VM1 à 4 :
    - Une (ou plusieurs) stack qui permet d'ajouter (ou de supprimer) des instances de SquashTM ;
    - Ces instances sont accessibles au monde (via le port 808x de votre VM) ;
    - L'ajout/suppression d'une instance Squash peut se faire via ligne de commande (via une IHM si on a le temps, c'est mieux ☺) ;
    - On peut choisir la version de SquashTM des instances (i.e., les instances peuvent être des versions différentes de SquashTM(*)).

(*) (Bon, pas besoin de gérer toutes les versions, 2 suffisent, du style une 1.16 et notre 1.7.4 adorée.)

SquashTM en SaaS

- Non, ne partez pas (enfin, pas tout de suite ☺).
- On va faire ensemble…
- … mais, en vrai, vous avez tout ce qu'il faut (bon, nous n'avons pas joué ensemble avec un reverse proxy, mais c'est pas compliqué ☺)
- (Au moins pour la version 'prototype', dans passer par l'API, mais, on peut jouer avec l'API aussi, c'est bien ☺)

Les contrôler tous

Ou comment passer des commandes Docker à partir d’un conteneur

Rompre l’isolation

- Par défaut les conteneurs sont isolés (ils ne peuvent voir que les conteneurs qui partagent son ou ses sous-réseaux).

- Mais, parfois, il peut être nécessaire de rompre cette isolation

  - Reverse proxy
  - Redémarrage d’un service malade
  - Un jenkins dans un conteneur qui veut déployer une stack
  - …

Rompre l’isolation

- Donner accès au /var/run/docker.sock
  - En lecture seule si on veut juste regarder (voir l’état, pas le modifier)
  - En lecture/écriture si on veut pouvoir modifier (déployer, …)
- Installer Docker (ou le SDK) sur le conteneur
- Et donner les bon droits au user du conteneur si ce n’est pas root

Donner les bons droits

- Le plus simple :
    - Ajouter le user dans les sudoers
    - … sans demande du mot de passe (sinon, via script, c’est plus compliqué)
- Le plus propre :
    - Ajouter le user dans le groupe docker