# Virtualsy

Cette plateforme vous permet de créer et d'utiliser une ou plusieurs machines virtuelles en fonction du type d'utilisateur que vous êtes.

## Installation

Pour installer ce projet en local sans docker, vous devez avoir python installé sur votre machine.

- Commencez par ouvrir un terminal à la racine de ce projet
- Creez un environnement virtuel de test python :
    - `python -m venv env` ( ou "python3" en fonction de votre installation )
    - `source env/bin/activate` (Linux shells (Bash, ZSH, etc.) or `.\env\Scripts\activate.bat` (pour Windows).
      Cette commande permettra de n'installer les dépendance que dans l'environnement de ce projet.

---

- Installez les dépendances avec `pip install -r requirements.txt` ou `pip3 install -r requirements.txt` si vous utilisez python3
- lancez la commande `python app.py` (ou `python3 app.py` si vous utilisez python3 )

un Serveur devrait s'être lancé sur le port :5000 de localhost, accedez au site web en copiant l'addresse sur le navigateur : `http://127.0.0.1:5000`

Et voila !! vous avez réussi à installer le projet.

## Utilisation

> Notons que les variables d'environnement sont définies dans le fichier .env et sont modifiables. Le fichier est automatiquement chargé lors du lancement du programme

Il y'a trois utilisateurs principaux dont les identifiants sont donnés ci dessous:

> vous pouvez vous déconnecter en appuyant sur le boutton `Disconnect` à droite, en dessous le l'avatar

```
1 - nocredituser {
    // qui a 0 crédit et ne peut donc utiliser aucune des machines
```

```
        email: nocredit@gmail.com
        password: password
    }

2 - onemachineuser {
    // qui ne peut utiliser qu'une machine préconfigurée
    email: onemachine@gmail.com
    password: password
}

3 - manymachinesuser {
    // qui peut utiliser plusieurs machines
    email: manymachines@gmail.com
    password: password
}
```

Une fois connecté, vous êtes redirigé vers la page d'accueil où vous pourrez choisir entre plusieurs machines en fonction de votre rôle.

Choisissez une machine et validez la selection. Vous serez redirigé vers une page vous indiquant l'etat de chargement de votre machine. Une fois le chargement terminé, vous aurez accès aux détails de votre machine.

> Veuillez noter que le chargement peut être plus ou moins long et que pour des soucis d'efficacité, nous n'avons pas accordé plus de temps à la validation des actions donc il se pourrait que vous ayez une erreur si vous quittez la page et que vous revenez pendant la création de la machine virtuelle. Nous vous invitons donc à surveiller la console si le chargement dépasse 6 minutes.

Une fois votre machine créée et les informations affichées, vous pourrez vous connecter avec les identifiants fournis via ssh ou rdp en fonction de l'os indiqué sur l'interface.

La machine ainsi créée s'autodétruira `10 minutes` plus tard, mais vous pouvez la détruire vous même en appuyant sur le bouton 'détruire'.

> l'opération de destruction peut également prendre 3 à 5 minutes en fonction du temps de réponse azure.

## Explication du code

### Choix de Technologie

Ce programme a été basé sur `Flask (Python)`, pour l'api et la gestion des ressources azure, et `Jquery` pour la gestion des animations de requêtes front end ainsi que `Jinja2`, qui est un langage de template pour Python.

## Fonctionnement et logique

Le programme principal `app.js` constitue l'api qui propose plusieurs routes à partir desquelles les requêtes sont écoutées.

A partir de l'interface, des listeners `Jquery` sont chargés d'interroger l'api pour créer, supprimer des VM ou encore rediriger vers des pages.

La route `/create-<vm>-vm` appelle en asynchrone la fonction de création de machine azure, définie dans le fichier `connect.py`. Elle prend en paramètre le type d'os récupéré depuis l'url.

Cette fonction appelée va alors définir les identifiants avec `get_credentials` puis appeler la fonction qui devra vérifier si une VM est déja existante, et la retourner si oui, sinon, créer une VM en fonction de l'OS demandé

> Si une VM existe déja, elle sera retournée, peu importe l'os demandé. Il faudra d'abord détruire cette machine avant de changer d'OS. (vous pourrez toujours verifier le type d'OS renvoyé sur l'interface)

## Suppression automatique

La suppression automatique a été gérée à l'aide des Threads python afin que le compteur ne gène pas l'execution des autres programmes.

```
    vm_deletion_thread =
threading.Thread(target=delete_ressources_after_delay)
    vm_deletion_thread.start()
```

Le Thread est lancé à la création de la VM.
(Pour tester, Il faudra prendre en compte le temps de supression des ressources qui peut prendre jusqu'à 5 minutes en fonction du temps de réponse azure)

> NB : Ce code est un prototype à améliorer (validation des actions, attribution de VM en fonction des comptes, navigation, etc...)