# INFRA LABOS - 2023

**chmod 0400 infrakeys_shera.pem**

**ssh -i ./infrakeys_shera.pem shera@13.93.66.177**

**scp -i infrakeys_shera.pem ./syllabusHTML.zip shera@13.93.66.177:/home/shera**

**sudo tail -f /var/log/apache2/error.log**

**sudo docker compose down --rmi all --volumes**

**sudo docker compose ps -a**

## Fiche 2

### EX1

- sudo apt-get update
- sudo apt-get install lynx
- apt-get install apache2
- cd /etc/apache2/sites-available/
- nano syllabusHTML.conf :

```
<VirtualHost *:80>
        ServerName syllabusHTML

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/syllabusHTML
        ErrorLog ${APACHE_LOG_DIR}/monsite_error.log
        CustomLog ${APACHE_LOG_DIR}/monsite_access.log combined

        <Directory /var/www/syllabusHTML >
        Require all granted
        AllowOverride All
        </Directory>
</VirtualHost>
```

- a2ensite syllabusHTML.conf
- systemctl reload apache2
- importer le dossier syllabusHTML avec son contenu /var/www :
  sudo mv syllabusHTML / /var/www/
- sudo nano /etc/hosts 127.0.0.1  syllabusHTML
- Dans la VM > Paramètres réseau > ajouter une règle de port > d'entrée > PORT 80
- Pour rendre accessible depuis la machine physique (Windows) alors, ouvrir PowerShell en tant qu'admin :
  - Notepad C:\Windows\System32\drivers\etc\hosts
  - @adresse_publique_VM  syllabusHTML
  - Enregistrer
  - Accéder au site depuis votre navigateur préféré http://syllabushtml/

## EX2

- apt-get install php php-mysql libapache2-mod-php
- apt-get install default-mysql-server
- mettre le dossier bonnesNouvelles dans /var/www/
- cd /etc/apache2/sites-available/
- nano sitePHP.conf :

```
<VirtualHost *:80>
        ServerName sitePHP
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/bonnesNouvelles
        ErrorLog ${APACHE_LOG_DIR}/monsite_error.log
        CustomLog ${APACHE_LOG_DIR}/monsite_access.log combined
         <Directory /var/www/bonnesNouvelles >
                Require all granted
                AllowOverride All
          </Directory>
 </VirtualHost>
```

- a2ensite sitePHP.conf
- sudo systemctl reload apache2
- lynx http://sitePHP
- installer mariadb:
    - sudo apt-get install mariadb-server mariadb-client
    - redémarrer serveur : sudo systemctl restart mariadb
- mysql –u root -p
- CREATE USER 'ipl'@'localhost' identified by 'ipl';
- GRANT ALL PRIVILEGES ON bdbn.* TO 'ipl'@'localhost' ;
- use bdbn;
- SELECT * FROM livres
- Aller dans le fichier Db.php qui contient la connexion avec la db dans le dossier models et mettre le user ‘ipl’ et mdp ‘ipl’ .
- lynx http://sitePHP -> onglet livres

# Fiche 3 – Apache 2

## EX1

- sudo a2enmod ssl
- sudo systemctl restart apache2
- sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/monsite.key -out /etc/ssl/certs/syllabusHTML.crt
- sudo nano /etc/apache2/sites-available/syllabusHTMLssl.conf :

```
<VirtualHost *:443>
        ServerName syllabus

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/syllabus
        ErrorLog ${APACHE_LOG_DIR}/monsite_error.log
        CustomLog ${APACHE_LOG_DIR}/monsite_access.log combined

        SSLEngine on
        SSLCertificateFile /etc/apache2/ssl/apache.crt
        SSLCertificateKeyFile /etc/apache2/ssl/apache.key

        <Directory /var/www/syllabus>
        Require all granted
        AllowOverride All
        </Directory>
</VirtualHost>
```

- sudo a2ensite syllabusHTMLssl.conf
- sudo systemctl reload apache2
- lynx https://syllabusHTML
- s’assurer que le port 80 est ouvert sur la VM
- tester le même lien sur navigateur et accepter le risque du certificat

**EX2**

- sudo adduser jetty
- sudo chmod -R 755 /home/jettyv2
- sudo usermod -aG sudo jettyv2
- sudo su - jettyv2
- sudo apt-get update
- sudo apt-get install default-jre
- cd siteJetty
- java -jar NoDBRunTest.jar &
- lynx http://localhost:8080
- Si c'est ok en localhost alors ouvrir le port 8080 sur la VM
- tester sur le navigateur : @clé publique:8080

- sudo a2enmod proxy
- sudo a2enmod proxy_http
- sudo nano /etc/apache2/sites-available/siteJetty.conf

```
<VirtualHost *:80>
        ServerName siteJetty
        ServerAdmin webmaster@localhost

        # attention / final !!!
        ProxyPass /  http://localhost:8080/
        ProxyPassReverse / http://localhost:8080/

        ErrorLog ${APACHE_LOG_DIR}/siteReverse_error.log
        CustomLog ${APACHE_LOG_DIR}/siteReverse_access.log combined

        SSLEngine on
        SSLCertificateFile /etc/apache2/ssl/apache.crt
        SSLCertificateKeyFile /etc/apache2/ssl/apache.key
</VirtualHost>
```

- sudo nano /etc/hosts mettre : 127.0.0.1  siteJetty
- sudo a2ensite siteJetty.conf
- sudo systemctl reload apache2
- lynx http://siteJetty
- si c'est ok, ouvrir powershell en mode admin :
    - Notepad C:\Windows\System32\drivers\etc\hosts
    - @adresse_publique  sitejetty

# Fiche 4 - Docker

## EX1

- sudo apt-get update
- sudo apt-get install docker.io
- Créer un dossier parent et mettre le dossier syllabusHTML dedans
- Créer 2 fichiers Dockerfile et 000-default.conf
- Dans le Dockerfile :

```
FROM debian:latest

RUN  apt-get update && \

    apt-get install -y apache2 -q


COPY 000-default.conf /etc/apache2/sites-available/000-default.conf

COPY siteHTML /var/www/html

CMD ["apache2ctl", "-D", "FOREGROUND"]
```

- Dans 000-default.conf:

```
<VirtualHost *:80>
        # pas de servername -> site par défaut
        #ServerName siteHTML

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ErrorLog ${APACHE_LOG_DIR}/siteHTML_errors.log
        CustomLog ${APACHE_LOG_DIR}/siteHTML_access.log combined

        <Directory /var/www/siteHTML>
            Require all granted
            AllowOverride all
        </Directory>

</VirtualHost>
```

- sudo docker build -t my-apache2-image .
- sudo docker run -p 8090:80 my-apache2-image
- lynx http://localhost:8080
- si OK tester sur votre navigateur préféré.

**EX2**

- Télécharger le dossier WebApplication2 depuis Moodle et le mettre dans la VM
- nano Dockerfile :

```
# Utilisez l'image SDK pour construire
FROM mcr.microsoft.com/dotnet/core/sdk:3.1 AS build-env
WORKDIR /app

# Copiez le fichier csproj et restore en tant que couches distinctes
COPY *.csproj ./
RUN dotnet restore

# Copiez tout le reste et construisez
COPY . ./
RUN dotnet publish -c Release -o out

# Construire l'image finale
FROM mcr.microsoft.com/dotnet/core/aspnet:3.1
WORKDIR /app
COPY --from=build-env /app/out .

# Exposer le port 80 pour les connexions web
EXPOSE 80

ENTRYPOINT ["dotnet", "WebApplication2.dll"]s
```

- sudo docker build -t dotnet .
- sudo docker run -p 8080:80 dotnet

# Fiche 5 – Docker compose

## EX1

# Téléchargement de docker-compose

- sudo apt-get update
- sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

# Rendre le fichier exécutable
- sudo chmod +x /usr/local/bin/docker-compose

# Vérifier l'installation
- docker-compose –version
- créer un dossier parent et mettre une copie du dossier de l'exo1 fiche 4
- dans le dossier parent créer un fichier docker-compose.yml :

```yaml
version: "3"

services:
  web:
    build:
      context: ./siteHTMLDocker  #mettre le chemin vers son Dockerfile
      dockerfile: Dockerfile
    ports:
      - "8091:80"
```

- sudo docker-compose up -d
- lynx http://localhost:8091

## EX2

- Télécharger le dossier exoplanets_pgdb depuis moodle
- Mettre le Dockerfile dans le dossier exoplanets
- Nano docker-compose.yml :

Attention à l'indentation du yaml, passez par VSC !

```
version: "3.9"
services:
  app:
    build: ./exoplanets/.
    Ports:
      - 9000:3000
    depends_on:
      postgres_db:
        condition: service_healthy
  postgres_db:
    image: postgres:latest
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U postgres"]
      interval: 30s
      timeout: 30s
      retries: 3
    volumes:
      - ./initdb:/docker-entrypoint-initdb.d
      - postgres-data:/var/lib/postgresql/data
volumes :
  postgres-data :
```

- nano exoplanets/Dockerfile:

```
FROM node:14
WORKDIR /app
COPY package*.json ./
RUN npm i
COPY . .
CMD ["node","app.js"]
```

- sudo docker-compose up -d

  lynx http://localhost:9000

# Fiche 6 – load balancer & Ansible

## EX1

Version compliqué avec des dockerfile comme app

- Créer un dossier load-balancer dans le dossier Nodejs_ipshow, et créer un Dockerfile et nodejs_ipshow.conf
- Dans ce Dockerfile :

```
FROM debian:latest

LABEL author="Choquet Olivier"
LABEL description="load-balancer apache"

# install apache
RUN apt-get update && apt-get install apache2 -y
# install module for load-balancer
RUN a2enmod proxy
RUN a2enmod proxy_http
RUN a2enmod proxy_balancer
RUN a2enmod lbmethod_byrequests
# copy vhost load-balancer sur le site par défaut
COPY nodejs_ipshow.conf /etc/apache2/sites-available/000-default.conf
EXPOSE 80
# démarrer apache dans le conteneur
CMD apachectl -D FOREGROUND
```

- Dans nodejs_ipshow.conf:

```
<VirtualHost *:80>
        #ServerName "nodejs_ipshow"
        ServerAdmin webmaster@localhost

         <Proxy balancer://mycluster>
            #BalancerMember http://localhost:3001
            #BalancerMember http://localhost:3002
            #BalancerMember http://localhost:3003
            BalancerMember http://app1:3000
            BalancerMember http://app2:3000
            BalancerMember http://app3:3000
        </Proxy>
        ProxyPreserveHost On
        ProxyPass / balancer://mycluster/
        ProxyPassReverse / balancer://mycluster/

        ErrorLog ${APACHE_LOG_DIR}/exoplanets_error.log
        CustomLog ${APACHE_LOG_DIR}/exoplanets_access.log combined
</VirtualHost>
```

- On revient à la racine du dossier : cd ..

- Dans le docker-compose.yml :

```yaml
version: '3.8'
services:
  app1:
    build:
      context: .
    environment:
      - NODE_ENV="test_app1"
    #ports:
    #  - '3001:3000'
  app2:
    build:
      context: .
    environment:
      - NODE_ENV="test_app2"
    #ports:
    #  - '3002:3000'
  app3:
    build:
      context: .
    environment:
      - NODE_ENV="test_app3"
    #ports:
    #  - '3003:3000'
  load-balancer:
    build:
      context: load-balancer/.
    ports:
      - '9000:80'
```

- Dans le Dockerfile :

```dockerfile
FROM node:20-alpine3.17

WORKDIR /app

# Copy the package.json and package-lock.json (if available)
COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "start"]
```

- sudo docker-compose up -d
- nano /etc/hosts :
  127.0.0.1 app1
  127.0.0.1 app2
  127.0.0.1 app3
- Si problème, sudo service docker restart

- lynx http://app1:9000
- lynx http://app2:9000
- lynx http://app3:9000
- lynx http://localhost:9000 # le load balancer
- Il faut remarquer l'adresse IP du container qui change

## EX2

- apt-get install ansible
- créer un dossier et dedans mettre le fichier exoplanets-playbook.yml :

  dans dest : mettre le chemin du dossier fraichement créer/nouveau dossier pour stocker le repo git dedans

```yaml
---
- name: Exoplanets Site
  hosts: localhost
  become: yes

  tasks:
    - name: Install Node.js
      apt:
        name: nodejs
        state: present

    - name: Install npm
      apt:
        name: npm
        state: present

    - name: Clone Exoplanets repository
      git:
        repo: https://gitlab.vinci.be/olivier.choquet/exoplanets_infra.git
        dest: /home/shera/Revision/exoplanets-ansible/exoplanets_infra
       force: yes

    - name: Install npm dependencies
      command: npm install
      args:
        chdir: /home/shera/Revision/exoplanets-ansible/exoplanets_infra

    - name: Start the application
      command: npm start
      args:
        chdir: /home/shera/Revision/exoplanets-ansible/exoplanets_infra

    - name: Install PM2
      npm:
        name: pm2
        state: present
        global: yes

    - name: Start the application with PM2
      command: pm2 start npm -- start
      args:
        chdir: /home/shera/Revision/exoplanets-ansible/exoplanets_infra
```

- sudo ansible-playbook exoplanets-playbook.yml
- lynx http://localhost:3000 devrait fonctionner

# Fiche 7 – Ansible 2

## EX1

- créer un dossier parent et dedans exécuter :
  ansible-galaxy init deploy_nodejs_app_from_git
- un dossier avec une structure va être créée, il suffit maintenant juste d'étaler le yaml de l'exercice précédent en plusieurs yaml.
- Créer dans le dossier parent un yaml et mettre dedans :

```yaml
---
- name: Deploy Node.js App
  hosts: localhost
  become: yes
  roles:
    - deploy_nodejs_app_from_git
```

- cd  deploy_nodejs_app_from_git/tasks
- nano main.yml :

```yaml
---
- name: Install Node.js
  apt:
    name: nodejs
    state: present

- name: Install npm
  apt:
    name: npm
    state: present

- name: Clone Git repository
  git:
    repo: "{{ git_repo }}"
    dest: "{{ app_dir }}"
    force: yes
- name: Install npm dependencies
  npm:
    path: "{{ app_dir }}"
    state: present
- name: Install PM2
  npm:
    name: pm2
    state: present
    global: yes
- name: Start the application with PM2
  command: pm2 start npm -- start
  args:
    chdir: "{{ app_dir }}"
```

- Dans defaults/main.yml :

```yaml
---
git_repo: https://gitlab.vinci.be/olivier.choquet/exoplanets_infra.git
app_dir: /home/shera/Revision/exoplanets-ansible/exoplanets_infra
```

**EX2**

- Créer un yaml et mettre dedans :

```yaml
- name: Déploiement d'un conteneur avec le syllabusHTML
  hosts: localhost
  vars:
    docker_image_to_deploy: olivierchoquet/syllabushtml:alpine
    container_name: syllabushtml_alpine
  tasks:
    - name: Installer prérequis Docker
      apt:
        name:
          - apt-transport-https
          - ca-certificates
          - curl
          - gnupg-agent
          - software-properties-common

    - name: Ajouter la clé dépôt Docker
      apt_key:
        url: https://download.docker.com/linux/debian/gpg
        state: present

    - name: Ajouter dépôt Docker
      apt_repository:
        repo: deb https://download.docker.com/linux/debian buster stable
        state: present

    - name: Installer Docker
      apt:
        name:
          - docker-ce
        state: latest

    - name: Installer pip
      apt:
        name: python3-pip

    # fonctionne pas
    #    - name: Supprimer environnement virtuel managé
    #      command: rm /usr/lib/python3.11/EXTERNALLY-MANAGED

    - name: Installer docker module python
      pip:
        name: docker

    - name: Pull syllabushtml image
      docker_image:
        source: pull
        name: "{{ docker_image_to_deploy }}"

    - name: Run a container
      docker_container:
        container_default_behavior: no_defaults
        name: container{{ container_name }}
        image: "{{ docker_image_to_deploy }}"
        ports:
          - "8080:80"
```

- sudo ansible-playbook -vvv dockerhubSyllabusHTML.yml
- lynx http://localhost:8080

# Fiche 8 – Terraform

## EX1

Bien expliquer dans la fiche, juste choisir Node – 20 LTS, la 16 LTS a été retirée.

## EX2

- pour installer terraform sur la VM
    - wget https://releases.hashicorp.com/terraform/1.6.6/terraform_1.6.6_linux_amd64.zip
    - unzip terraform_1.6.6_linux_amd64.zip
    - sudo mv terraform /usr/local/bin/
- terraform init
- terraform plan
- terraform apply “tfplan”
- terraform refresh si ça fonctionne pas

  output "app_url" {

  value = azurerm_linux_web_app.webapp.default_hostname

  }
- pour avoir l’url :

  terraform output app_url