Pour répondre au besoin de la PME **Smarttech**, voici un plan détaillé pour concevoir et déployer ces services à l'aide de **Docker**, **Dockerfile**, et **Docker Compose** :

**Étape 1 : Prérequis**

* **Installer Docker** : Installer Docker et Docker Compose sur une machine Linux (ex : Ubuntu Server).
* **Configurer le réseau** : Créer un réseau Docker bridge pour permettre la communication entre les conteneurs.
* **Structure de projet** :

smarttech-services/

├── ftp/

│ ├── Dockerfile

├── ssh/

│ ├── Dockerfile

├── toip/

│ ├── Dockerfile

├── dns/

│ ├── Dockerfile

├── mail/

│ ├── Dockerfile

├── vnc/

│ ├── Dockerfile

│

│

├── ftp-client/

│ ├── Dockerfile

├── ssh-client/

│ ├── Dockerfile

├── toip-client/

│ ├── Dockerfile

├── dns-client/

│ ├── Dockerfile

├── mail-client/

│ ├── Dockerfile

├── vnc-client/

│ ├── Dockerfile

└── docker-compose.yml

├── dns/

│ ├── Dockerfile

│ ├── db.smarttech

│ ├── named.conf.local

│ ├── named.conf.options

Dockerfile:

FROM ubuntu:latest

# Installer Bind9

RUN apt-get update && apt-get install -y bind9 bind9utils bind9-doc

# Copier les fichiers de configuration

COPY named.conf.local /etc/bind/

COPY named.conf.options /etc/bind/

COPY db.smarttech /etc/bind/

# Exposer le port DNS

EXPOSE 53/udp

EXPOSE 53/tcp

# Commande pour démarrer le service DNS

CMD ["named", "-g"]

db.smarttech:

$TTL 604800

@ IN SOA dns.smarttech.smarttech. admin.smarttech.smarttech. (

2024010101 ; Serial

604800 ; Refresh

86400 ; Retry

2419200 ; Expire

604800 ) ; Negative Cache TTL

; Nom de serveur DNS

@ IN NS dns.smarttech.smarttech.

; Enregistrements A

dns IN A 172.20.0.5

ftp IN A 172.20.0.2

ssh IN A 172.20.0.3

toip IN A 172.20.0.4

mail IN A 172.20.0.6

vnc IN A 172.20.0.7

named.conf.local:

zone "smarttech.local" {

type master;

file "/etc/bind/db.smarttech";

};

named.conf.options:

options {

directory "/var/cache/bind";

// Autoriser les requêtes du réseau Docker

allow-query { 172.20.0.0/24; };

// Adresses sur lesquelles écouter

listen-on { any; };

listen-on-v6 { none; };

recursion yes;

dnssec-validation no;

};

├── toip/

│ ├── Dockerfile

Dockerfile:

# Dockerfile générique (ftp, ssh, mail, dns, etc.)

FROM ubuntu:latest

# Définir la variable d'environnement USER (en fonction de votre service)

ENV USER=root

# Préparez l'environnement (ex: installation de packages, etc.)

# Par exemple, si vous avez besoin de packages spécifiques, installez-les ici

RUN apt-get update && apt-get install -y bash

# Exécution d'un processus qui maintient le conteneur en vie

CMD ["tail", "-f", "/dev/null"]

docker-compose.yml:

version: '3'

services:

  dns:

    build: ./dns

    container\_name: dns\_server

    ports:

      - "53:53/udp"

      - "53:53/tcp"

    networks:

      smarttech\_network:

        ipv4\_address: 172.20.0.5

networks:

  smarttech\_network:

    driver: bridge

    ipam:

      config:

        - subnet: 172.20.0.0/24

**Dans docker desktop**

docker-compose up --build -d

Acceder au server dns

docker exec -it dns\_server bash

apt update && apt install net-tools iputils-ping nano

apt install dnsutils -y

service bind9 start

service bind9 restart

service bind9 status

Editer par le fichier nano /etc/hosts et ajouter ceci:

172.20.0.2 ftp.smarttech.local ftp

172.20.0.3 ssh.smarttech.local ssh

172.20.0.4 toip.smarttech.local toip

172.20.0.5 dns.smarttech.local dns

172.20.0.6 mail.smarttech.local mail

172.20.0.7 vnc.smarttech.local vnc

Ensuite lancer le server avec:

named -g

Depuis le container ssh:

docker exec -it ssh\_server bash

apt update && apt install net-tools iputils-ping nano

apt install dnsutils -y

ping 172.20.0.5

nslookup ftp.smarttech.local 172.20.0.5

nslookup mail.smarttech.local 172.20.0.5

Pour que la resolution des adresses IP soit prit en compte niveau client.

Editer par le fichier nano /etc/resolv.conf et ajouter l’adresse du server dns devant nameserver à la place de existant :

172.20.0.5



nslookup ftp.smarttech.local

ping mail.smarttech.local

Docker-compose.yml:

version: '3'

services:

  dns:

    build: ./dns

    container\_name: dns\_server

    ports:

      - "53:53/udp"

      - "53:53/tcp"

    networks:

      smarttech\_network:

        ipv4\_address: 172.20.0.5

networks:

  smarttech\_network:

    driver: bridge

    ipam:

      config:

        - subnet: 172.20.0.0/24

**Installer Asterisk et configure le server :**

├── toip/

│ ├── Dockerfile

Dockerfile:

# Dockerfile générique (ftp, ssh, mail, dns, etc.)

FROM ubuntu:latest

# Définir la variable d'environnement USER (en fonction de votre service)

ENV USER=root

# Préparez l'environnement (ex: installation de packages, etc.)

# Par exemple, si vous avez besoin de packages spécifiques, installez-les ici

RUN apt-get update && apt-get install -y bash

# Exécution d'un processus qui maintient le conteneur en vie

CMD ["tail", "-f", "/dev/null"]

docker-compose.yml:

version: '3'

services:

  toip:

    build: ./toip

    container\_name: toip\_server

    networks:

      smarttech\_network:

        ipv4\_address: 172.20.0.4

  dns:

    build: ./dns

    container\_name: dns\_server

    ports:

      - "53:53/udp"

      - "53:53/tcp"

    networks:

      smarttech\_network:

        ipv4\_address: 172.20.0.5

networks:

  smarttech\_network:

    driver: bridge

    ipam:

      config:

        - subnet: 172.20.0.0/24

**Dans docker desktop**

docker-compose up --build -d

**Configurer le server toip acceder au server :**

docker exec -it toip\_server bash

apt update && apt install net-tools iputils-ping nano

apt install dnsutils -y

Installez les dépendances nécessaires pour Asterisk :

apt install -y wget build-essential subversion libncurses5-dev libssl-dev libxml2-dev libsqlite3-dev libjansson-dev uuid-dev libedit-dev

wget <https://downloads.asterisk.org/pub/telephony/asterisk/asterisk-20-current.tar.gz>

tar -xvzf asterisk-20-current.tar.gz

cd asterisk-20.11.0/

make clean

./configure

make

make install

asterisk -v

apt install asterisk

asterisk -cvvv

asterisk -r

nano /etc/asterisk/sip.conf:

[general]

context=default

[1000]

type=friend

secret=secretpassword

host=dynamic

context=toip

[1001]

type=friend

secret=secretpassword

host=dynamic

context=toip

nano /etc/asterisk/extensions.conf:

[toip]

exten => 1000,1,Dial(SIP/1000)

exten => 1001,1,Dial(SIP/1001)

**Configuration des client :**

docker exec -it ssh\_server bash

Installez les dépendances nécessaires pour Asterisk :

apt install -y wget build-essential subversion libncurses5-dev libssl-dev libxml2-dev libsqlite3-dev libjansson-dev uuid-dev libedit-dev

wget <https://downloads.asterisk.org/pub/telephony/asterisk/asterisk-20-current.tar.gz>

tar -xvzf asterisk-20-current.tar.gz

cd asterisk-20.11.0/

make clean

./configure

make

make install

asterisk -v

apt install asterisk

asterisk -cvvv

asterisk -r

nano /etc/asterisk/sip.conf:

[general]

context=default

allowguest=no

srvlookup=yes

[1000]

type=friend

secret=secretpassword

host=172.20.0.5 ; Adresse IP du serveur Asterisk (toip\_server)

context=toip

nano /etc/asterisk/extensions.conf:

default]

exten => [1000,1,Dial(SIP/1000@172.20.0.5)](mailto:1000,1,Dial(SIP/1000@172.20.0.5))

Configurattion du server ssh:

smarttech-services/

├── ssh/

│ ├── Dockerfile

Dockerfile:

# Dockerfile générique (ftp, ssh, mail, dns, etc.)

FROM ubuntu:latest

# Définir la variable d'environnement USER (en fonction de votre service)

ENV USER=root

# Préparez l'environnement (ex: installation de packages, etc.)

# Par exemple, si vous avez besoin de packages spécifiques, installez-les ici

RUN apt-get update && apt-get install -y bash

# Exécution d'un processus qui maintient le conteneur en vie

CMD ["tail", "-f", "/dev/null"]

Docker-compose.yml:

version: '3'

services:

  ssh:

    build: ./ssh

    container\_name: ssh\_server

    networks:

      smarttech\_network:

        ipv4\_address: 172.20.0.3

  toip:

    build: ./toip

    container\_name: toip\_server

    networks:

      smarttech\_network:

        ipv4\_address: 172.20.0.4

  dns:

    build: ./dns

    container\_name: dns\_server

    ports:

      - "53:53/udp"

      - "53:53/tcp"

    networks:

      smarttech\_network:

        ipv4\_address: 172.20.0.5

networks:

  smarttech\_network:

    driver: bridge

    ipam:

      config:

        - subnet: 172.20.0.0/24

**Dans docker desktop**

docker-compose up --build -d

docker exec -it ssh\_server bash

apt update && apt install net-tools iputils-ping nano

apt install dnsutils -y

apt install -y openssh-server

service ssh start

nano /etc/ssh/sshd\_config

· Assurez-vous que les lignes suivantes sont correctement configurées :

· Autoriser l'authentification par mot de passe :

PasswordAuthentication yes

Interdire la connexion en tant que root :

PermitRootLogin no

Activer l'utilisation des clés publiques/privées :

PubkeyAuthentication yes

· Redémarrer le service SSH pour appliquer les changements :

· · service ssh restart

adduser sshuser

usermod -aG ssh sshuser

ssh-keygen -t rsa -b 2048

ssh-copy-id -i ~/.ssh/id\_rsa.pub sshuser@172.20.0.3

ssh [sshuser@172.20.0.3](mailto:sshuser@172.20.0.3)

**Configuration ftp:**

├── ftp/

│ ├── Dockerfile

Dockerfile:

# Dockerfile générique (ftp, ssh, mail, dns, etc.)

FROM ubuntu:latest

# Définir la variable d'environnement USER (en fonction de votre service)

ENV USER=root

# Préparez l'environnement (ex: installation de packages, etc.)

# Par exemple, si vous avez besoin de packages spécifiques, installez-les ici

RUN apt-get update && apt-get install -y bash

# Exécution d'un processus qui maintient le conteneur en vie

CMD ["tail", "-f", "/dev/null"]

Docker-compose.yml:

version: '3'

services:

  ftp:

    build: ./ftp

    container\_name: ftp\_server

    networks:

      smarttech\_network:

        ipv4\_address: 172.20.0.2

  ssh:

    build: ./ssh

    container\_name: ssh\_server

    networks:

      smarttech\_network:

        ipv4\_address: 172.20.0.3

  toip:

    build: ./toip

    container\_name: toip\_server

    networks:

      smarttech\_network:

        ipv4\_address: 172.20.0.4

  dns:

    build: ./dns

    container\_name: dns\_server

    ports:

      - "53:53/udp"

      - "53:53/tcp"

    networks:

      smarttech\_network:

        ipv4\_address: 172.20.0.5

networks:

  smarttech\_network:

    driver: bridge

    ipam:

      config:

        - subnet: 172.20.0.0/24

**Dans docker desktop**

docker-compose up --build -d

**Configuration server ftp:**

docker exec -it ftp\_server bash

apt update

apt update && apt install net-tools iputils-ping nano

apt install dnsutils -y

apt install -y vsftpd

nano /etc/vsftpd.conf

mkdir -p /home/ftp\_shared

chmod 755 /home/ftp\_shared

adduser ftpuser

mkdir -p /home/ftpuser/ftp\_files

chown ftpuser:ftpuser /home/ftpuser/ftp\_files

chmod 750 /home/ftpuser/ftp\_files

touch /home/ftpuser/ftp\_files/ficher1

service vsftpd restart

service vsftpd status

apt install -y ftp

nano etc/resolv.conf

ftp 172.20.0.2

**Étape 1 : Accéder au conteneur client**

docker exec -it client\_container bash

apt install -y ftp

ftp 172.20.0.2

ls

get fichier\_a\_telecharger.txt

put fichier\_a\_envoyer.txt

Exit

**Configuration du server vnc:**

├── vnc/

│ ├── Dockerfile

└── docker-compose.yml

Dockerfile:

# Dockerfile générique (ftp, ssh, mail, dns, etc.)

FROM ubuntu:latest

# Définir la variable d'environnement USER (en fonction de votre service)

ENV USER=root

# Préparez l'environnement (ex: installation de packages, etc.)

# Par exemple, si vous avez besoin de packages spécifiques, installez-les ici

RUN apt-get update && apt-get install -y bash

# Exécution d'un processus qui maintient le conteneur en vie

CMD ["tail", "-f", "/dev/null"]

docker-compose.yml:

version: '3'

services:

  ftp:

    build: ./ftp

    container\_name: ftp\_server

    networks:

      smarttech\_network:

        ipv4\_address: 172.20.0.2

  ssh:

    build: ./ssh

    container\_name: ssh\_server

    networks:

      smarttech\_network:

        ipv4\_address: 172.20.0.3

  toip:

    build: ./toip

    container\_name: toip\_server

    networks:

      smarttech\_network:

        ipv4\_address: 172.20.0.4

  dns:

    build: ./dns

    container\_name: dns\_server

    ports:

      - "53:53/udp"

      - "53:53/tcp"

    networks:

      smarttech\_network:

        ipv4\_address: 172.20.0.5

  vnc:

    build: ./vnc

    container\_name: vnc\_server

    ports:

      - "5901:5901" # Port VNC

      - "6080:6080" # Port NoVNC

    networks:

      smarttech\_network:

        ipv4\_address: 172.20.0.7

networks:

  smarttech\_network:

    driver: bridge

    ipam:

      config:

        - subnet: 172.20.0.0/24

**Dans docker desktop**

docker-compose up --build -d

# Dans le conteneur, installez les paquets nécessaires :

apt-get update

apt-get install -y tigervnc-standalone-server tigervnc-common xterm

apt install net-tools iputils-ping nano

apt install dnsutils -y

useradd -m vncuser1

useradd -m vncuser2

su - vncuser1

vncpasswd

Répétez cette étape pour chaque utilisateur.

vncserver :1 -geometry 1280x1024 -depth 24

vncserver :2 -geometry 1280x1024 -depth 24

apt install git

git clone https://github.com/novnc/noVNC.git /opt/noVNC

cd /opt/noVNC

apt-get install -y websockify

websockify --web /opt/noVNC 6080 localhost:5901

**AU NIVEAU DU CLIENT**

vncserver -kill :1

vncserver :1 -geometry 1280x1024 -depth 24 -localhost no

netstat -tuln | grep 5901

**Tester VNC :**

telnet 172.20.0.7 5901

**Tester NoVNC :**

curl http://<nom\_conteneur\_vnc>:6080/vnc.html