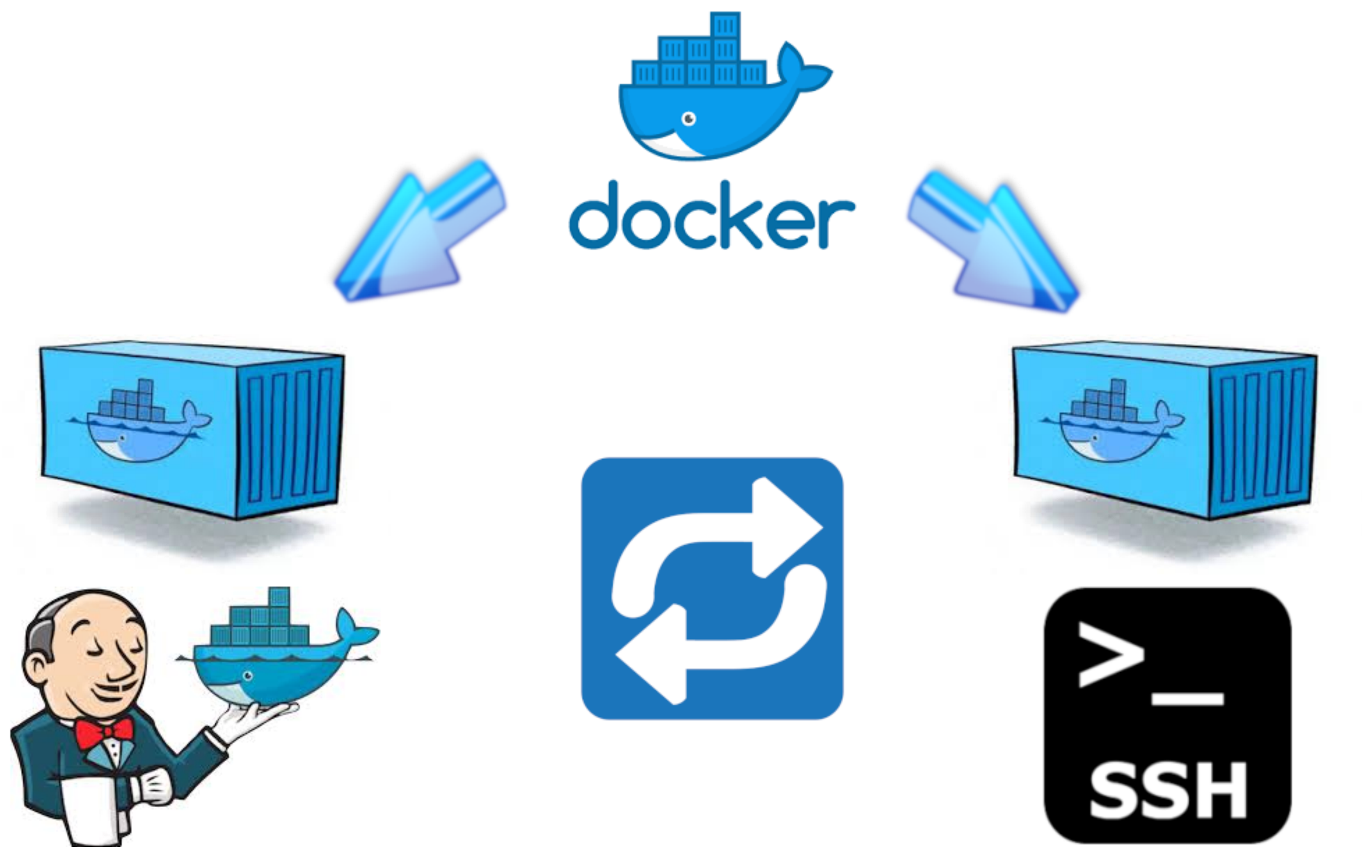
📌 Setup-SSH-For-Containers-Jenkins 📌

🚩 Abstract and Architecture:

We have two containers, one is Jenkins and the other is a slave has applications & tools. We want to connect the slave to the Jenkins container. So Jenkins can run and excute commands on the other container. We will use SSH to connect the two containers based on the Docker-compose Network.

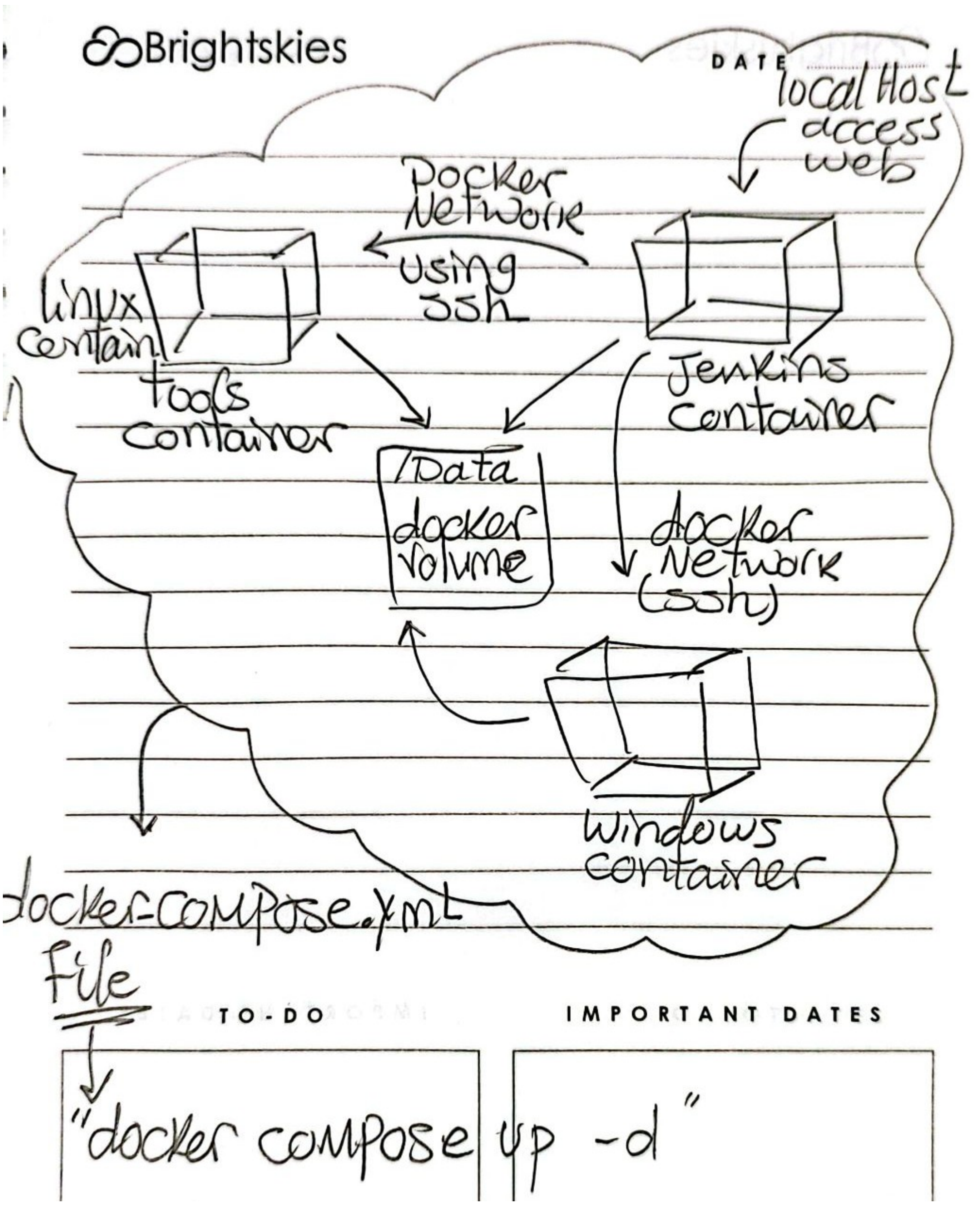


⚙ Prerequisites:

- Jenkins Image with SSH-Client installed.
- Tools Image with the tools required & SSH-Server installed.
- Docker-compose installed.

👨‍💻 Imagine the scene :

- We are going to Use Jenkins as an interface for the other containers to run pipelines using the other containers which contain tools and applications.
- After configuring the SSH connection between the Jenkins container and the other containers, we can run pipelines on the other containers using Jenkins.
- This approach provide us with varity of containers as we can add more containers to the Jenkins container to run different pipelines. (In this example we have only one container)



Steps:

1. └┈➤ Create a Dockerfile for Jenkins container:

```
# You will find the files in this repo
FROM jenkins/jenkins:lts
```

```
# Set environment variables
ENV JENKINS_HOME /var/jenkins_home

# Switch to the root user to install additional packages
USER root

# Update the package list and install additional tools
RUN apt-get update && apt-get install -y \
    nano \
    openssh-client \
    vim \
    curl \
    && rm -rf /var/lib/apt/lists/*



# Expose Jenkins web and agent ports
EXPOSE 8080 50000
```

2. └┈➤ Create a Dockerfile for the tools container:

```
FROM ubuntu:latest
#Updating the package list
RUN apt-get update

#Installing the essential Tools
RUN apt-get install -y \
    software-properties-common \
    nano \
    openssh-server \
    wget \
    unzip \
    git \
    gnupg2 \
    && rm -rf /var/lib/apt/lists/*

#Installing Python3 and pip
RUN apt-get update && apt-get install -y python3
RUN apt-get update && apt-get install -y python3-pip

#installing Pytest
RUN apt update && apt install -y python3-pytest


# Install MinGW64
RUN apt-get update && apt-get install -y \
    mingw-w64 \
    && rm -rf /var/lib/apt/lists/*

#install CppCheck
RUN apt-get update && apt-get install -y \
    cppcheck \
    && rm -rf /var/lib/apt/lists/*


# Install CMake
RUN apt-get update && apt-get install -y \
    cmake \
    && rm -rf /var/lib/apt/lists/*


# Install Google Test
RUN apt-get update && apt-get install -y \
    libgtest-dev \
    && rm -rf /var/lib/apt/lists/*


# Install JDK-17
RUN apt-get update && apt-get install -y \
    openjdk-17-jdk \
    && rm -rf /var/lib/apt/lists/*


# Install Allure 2.30
RUN wget https://github.com/allure-
framework/allure2/releases/download/2.30.0/allure-2.30.0.zip && \
    unzip allure-2.30.0.zip -d /opt && \
    ln -s /opt/allure-2.30.0/bin/allure /usr/bin/allure && \
    rm allure-2.30.0.zip


# Set environment variables
ENV JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64
ENV PATH=$PATH:/opt/w64devkit/bin:/usr/lib/gcc/x86_64-w64-mingw32

# Expose SSH port
EXPOSE 22


# Default command
CMD ["tail", "-f", "/dev/null"]
```

3. └┈➤ Build the Jenkins image:

```
#Make sure to go to the directory where the Dockerfile is located before
running the command
docker build -t jenkinsssh:2 .
```

4. └┈➤ Build the tools image:

```
```

```
#Make sure to go to the directory where the Dockerfile is located before
running the command
docker build -t tools:1.2 .
```

5. └┈➤ If you use the following command, you will see the images you have created:

   -
     ```
     docker images
     ```

6. └┈➤ Then create a Docker-compose file:

```
version: '3.8'

services:
  jenkins: # Jenkins container
    image: jenkinsssh:2
    container_name: jenkinsssh
    ports:
      - "8085:8080"
    volumes: # Mount the volume to the container
      - jenkins_home:/var/jenkins_home
    networks: # Connect the container to the network
      - jenkins_network

  tools: # Tools container
    image: tools:1.2
    container_name: tools
    networks:
      - jenkins_network
    volumes: # Mount the volume to the container
      - jenkins_home:/var/jenkins_home

volumes:
  jenkins_home:

networks: # Create a network
  jenkins_network: # Connect all the containers to the network
    driver: bridge
```

7. └┈➤ Run the Docker-compose file:

```
docker-compose up -d
```

8. └┈➤ Check the containers You will see the two containers running:

```
docker ps
```

9. └┈➤ Now we are going to Test the SSH connection between the Jenkins container and the tools container:
   - Go to the Tools container:

   ```
   docker exec -it "container name (or ID)" bash
   ```

   - go to the ssh directory to change the sshd_config file to allow the root user to connect:

   ```
   nano /etc/ssh/sshd_config
   ```

   - Scroll down to the PermitRootLogin uncomment it if conmmented and change it to yes:
   - Now check the status of the ssh-server:

   ```
   service --status-all
   ```

   - Almost you will find it stopped like this: [-], so start it:

   ```
   service ssh start
   ```

   - We need to know the root password to connect to the container from Jenkins, so change the root password:

   ```
   passwd root
   ```

   - Now exit the container:

   ```
   exit
   ```

   - Now we need to know the IP address of the tools container to connect to it from Jenkins:

   ```
   docker inspect tools | grep "IPAddress"
   # Don't forget to change `tools` to the name of your container
   ```

- Now we are going to connect to the tools container from Jenkins:
- Go to the Jenkins container:

```
docker exec -it jenkinsssh bash
# Don't forget to change `jenkinsssh` to the name of your container
```

- Now we are going to connect to the tools container:

```
ssh root@tools_IP
```

- You will be asked to enter the password, enter the password you have set before.
- If you have connected successfully, you will see the tools container name in the terminal.
- Now exit the tools container:

```
exit
```

- Now exit the Jenkins container:

```
exit
```

- Now we have successfully 🎉 connected the Jenkins container to the tools container using SSH .