14th Int. Conf. on IT Security Incident Management & IT Forensics (IMF 2025)
Practical Malware Analysis and Memory Forensics for Incident Response
Albstadt-Sigmaringen University (Germany)

Setting Up a Docker Environment for Volatility and Volatility 3

This document explains how to install and configure a development environment for Volatility 2 and Volatility 3 within a Docker container, providing a reproducible and isolated setup for forensic analysis. Docker allows you to create a controlled environment without having to install Volatility 2, Volatility 3, and their dependencies directly on the host system. Throughout this guide, we'll detail the steps required to configure the container and ensure an efficient, flexible, and portable environment for memory analysis.

1 Setting Up a Docker-Based Environment

Before you begin, make sure Docker is installed on your computer. If you haven't already, you can download it from the official Docker website: `https://www.docker.com/get-started/`. For this guide, we recommend installing `Docker Desktop` (`https://docs.docker.com/desktop/`), as it provides an intuitive interface for managing containers and simplifies the setup process on Windows, macOS, and Linux.

Once installed, launch the application. You'll see a welcome window similar to Figure 1. You don't need to create an account; simply skip this step by selecting the Skip option in the top right corner.

2 Creating a Docker Container

There are several ways to work with Docker. One of the simplest methods is to use a ≪`Dockerfile`≫, which specifies the configuration and software required for the desired environment. In this course, we will work with this approach.

The ≪`Dockerfile`≫ automates the creation of a Docker image, ensuring that the environment is reproducible on different systems. The script defines how to set up a development environment, install the necessary libraries, and configure essential services.

Listing 1 is an example ≪`Dockerfile`≫ that sets up a Python-based environment within a Docker container. Create this ≪`Dockerfile`≫ in your working directory (or simply download the file from the workshop website[1]).

[1] See `https://webdiis.unizar.es/~ricardo/imf-25-workshop`.

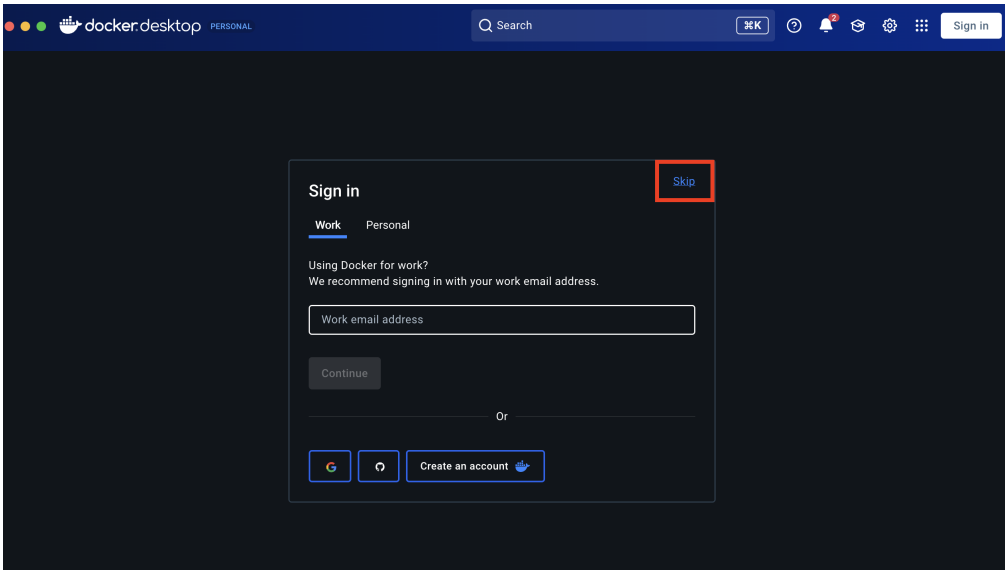

Figure 1: `Docker Desktop` welcome screen.

Listing 1: «`Dockerfile`» that sets up a Python-based environment for Volatility 2 and Volatility 3.

```
# Use the official Python 3.8 image as the base
FROM python:3.8

# Set environment variables to avoid interactive prompts
ENV DEBIAN_FRONTEND=noninteractive
# Set the container hostname
ENV HOSTNAME=volatility-box

# Install OS dependencies
RUN apt-get update && apt-get install -y apt-utils
RUN apt-get update && apt-get install -y \
    git \
    openssh-server \
    sudo \
    vim \
    zsh \
    man \
    curl \
    wget \
    unzip zip \
    exiftool \
    graphviz \
    && rm -rf /var/lib/apt/lists/*
```



```
# Create a user with sudo privileges
RUN useradd -m -s /bin/bash forensic && echo "forensic:forensic" |
↪     chpasswd && adduser forensic sudo

# Add forensic user to sudoers file with NOPASSWD option
RUN echo "forensic ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers

# Define build argument for architecture detection
ARG TARGETPLATFORM
RUN echo "Building for platform: $TARGETPLATFORM"

# Set up SSH
RUN mkdir /var/run/sshd
RUN echo 'PermitRootLogin no' >> /etc/ssh/sshd_config
RUN echo 'PasswordAuthentication yes' >> /etc/ssh/sshd_config

# Generate SSH host keys
RUN ssh-keygen -A

# Install Volatility 2
RUN git clone https://github.com/volatilityfoundation/volatility.
↪    git /opt/volatility2 && \
     ln -s /usr/bin/python2 /usr/bin/python && \
     chmod +x /opt/volatility2/vol.py

# Enable contrib and non-free repositories (for Debian image base)
#RUN sed -i 's/main/main contrib non-free/g' /etc/apt/sources.list
↪     && \
#     apt-get update

# Install Volatility required dependencies including development
↪    tools
RUN apt-get update && apt-get install -y \
     python3-dev \
     build-essential \
     python3-yara \
     libssl-dev\
     yara \
     libyara-dev \
     libntirpc-dev

# Install Volatility 3 from git
RUN git clone https://github.com/volatilityfoundation/volatility3.
↪    git /opt/volatility3 && \
     chmod +x /opt/volatility3/vol.py

# Create a shared folder for memory dumps
RUN mkdir /shared && chmod 777 /shared
```



```
# Change the default shell for the forensic user to zsh
RUN chsh -s /bin/zsh forensic

# Install Oh My Zsh for the forensic user
USER forensic
RUN sh -c "$(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/
    ohmyzsh/master/tools/install.sh)" \
    && rm -rf /home/forensic/.oh-my-zsh/custom/themes/robbyrussell
        .zsh-theme

# On macOS (arm, tested)
RUN pip3 install -U git+https://github.com/VirusTotal/yara-python
RUN curl -fsSL https://sh.rustup.rs -o /tmp/rustup.sh
RUN sh /tmp/rustup.sh -y
RUN rm /tmp/rustup.sh
RUN sh -c "PATH=$PATH:$HOME/.cargo/bin"

# Install Python3 dependencies
RUN pip3 install --upgrade setuptools pip
RUN pip3 install distorm3 yara-python pefile pycrypto tqdm

# Python2 installation (from source code)
# Download source tarball into a subfolder named src, and untar:
USER root
WORKDIR /tmp
# Get Pyhton2.7.14 and build it from source
RUN wget https://www.python.org/ftp/python/2.7.14/Python-2.7.14.
    tgz
RUN tar xvf Python-2.7.14.tgz
WORKDIR /tmp/Python-2.7.14
# Apply this patch to bypass SSL error
RUN wget https://gist.githubusercontent.com/rkitover/2
    d9e5baff1f1cc4f2618dee53083bd35/raw/7
    f33fcf5470a9f1013ac6ae7bb168368a98fe5a0/python-2.7.14-custom-
    static-openssl.patch
RUN git apply python-2.7.14-custom-static-openssl.patch
RUN mkdir /opt/python-2.7.14
RUN ./configure --prefix=/opt/python-2.7.14
```



```
RUN for f in \
    rpc/rpc.h \
    rpc/types.h \
    rpc/xdr.h \
    rpc/tirpc_compat.h \
    rpc/auth.h \
    rpc/rpc_error.h \
    rpc/rpc_err.h \
    rpc/clnt_stat.h \
    rpc/auth_stat.h \
    rpc/clnt.h \
    rpc/svc.h \
    rpc/rpc_msg.h \
    rpc/work_pool.h \
    rpc/pool_queue.h \
    rpc/auth_unix.h \
    rpc/rpcb_clnt.h \
    rpc/rpcb_prot.h \
    rpc/rpcent.h \
    ; do ln -sf /usr/include/ntirpc/$f /usr/include/rpc/; done &&
↪       \
    for f in \
    misc/stdio.h \
    misc/rbtree.h \
    misc/opr.h \
    misc/wait_queue.h \
    misc/queue.h \
    misc/portable.h \
    misc/timespec.h \
    misc/os_epoll.h \
    ; do ln -sf /usr/include/ntirpc/$f /usr/include/misc/; done &&
↪       \
    for f in \
    netconfig.h \
    intrinsic.h \
    reentrant.h \
    ; do ln -sf /usr/include/ntirpc/$f /usr/include/; done

RUN make
RUN make install
RUN sh -c "ln -sf /opt/python-2.7.14/bin/* /usr/bin/."

# Make sure the latest setuptools and pip are installed:
RUN /opt/python-2.7.14/bin/python2 -m ensurepip
# Install pip for Python 2 manually
RUN curl -fsSL https://bootstrap.pypa.io/pip/2.7/get-pip.py -o get
```

```
    -pip.py && \
     /opt/python-2.7.14/bin/python2 get-pip.py && \
     rm get-pip.py
# Install Vol2 dependendices
RUN /opt/python-2.7.14/bin/pip2 install distorm3 pycrypto pefile
    yara-python

# Set up a default .zshrc configuration for the forensic user
RUN echo "PATH=$PATH:/opt/python-2.7.14/bin:/home/forensic/.local/
    bin" >> /home/forensic/.zshrc
RUN echo "export LD_LIBRARY_PATH=/opt/python-2.7.10/lib:
    $LD_LIBRARY_PATH" >> /home/forensic/.zshrc
RUN sh -c "export LD_LIBRARY_PATH=/opt/python-2.7.10/lib:
    $LD_LIBRARY_PATH"
RUN echo "LANGUAGE=en_US.UTF-8" >> /home/forensic/.zshrc
RUN echo "LC_CTYPE=en_US.UTF-8" >> /home/forensic/.zshrc
RUN echo "LC_ALL=en_US.UTF-8" >> /home/forensic/.zshrc

RUN chown forensic:forensic -R /opt

RUN echo "alias volatility2='/usr/bin/python2 /opt/volatility2/vol
    .py'" >> /home/forensic/.zshrc && \
     echo "alias volatility3='/usr/bin/python3 /opt/volatility3/vol
        .py'" >> /home/forensic/.zshrc && \
     echo "alias vol2='volatility2'" >> /home/forensic/.zshrc && \
     echo "alias vol3='volatility3'" >> /home/forensic/.zshrc && \
     echo "alias grep='grep --color=auto'" >> /home/forensic/.zshrc

# Install r2
WORKDIR /opt
RUN git clone https://github.com/radareorg/radare2.git
RUN /opt/radare2/sys/install.sh
RUN pip3 install r2pipe

# Install readpe
WORKDIR /opt
RUN git clone https://github.com/mentebinaria/readpe.git
WORKDIR /opt/readpe
RUN make
RUN make install
RUN echo "/usr/local/lib" >> /etc/ld.so.conf.d/libpe.conf
RUN ldconfig

# Expose SSH port
EXPOSE 22

# Start SSH service
CMD ["/usr/sbin/sshd", "-D"]
```


3 Building the Docker Image

To prepare the environment, you need to build the Docker image using the ≪`Dockerfile`≫ you created in the previous section. Open a terminal in the directory containing the configuration file (≪`Dockerfile`≫) and simply run the following command:

```
docker build -t volatility-container .
```

This command creates a Docker image named `volatility-container`, which includes all the dependencies needed for the course. The process may take a few minutes as it downloads a Debian GNU/Linux base image and installs all the necessary libraries. Once complete, the image is ready to use.

A Docker image is a packaged environment that contains everything defined in a ≪`Dockerfile`≫, including the base operating system, dependencies, configurations, scripts, and any additional software needed to run an application. It serves as a *blueprint* for containers, defining how an application should be configured and run in a consistent environment.

Docker images are read-only. When launching a container, we create an instance of a Docker image. This allows users to deploy applications quickly, reproducibly, and in isolation across different systems. Each container runs in its own isolated environment, ensuring consistency of dependencies and configurations regardless of the host system[2]. Because containers share the host operating system kernel but have their own file system, processes, and network, they offer a lightweight and efficient alternative to traditional virtual machines.

4 Running the Docker Container

After building the image, open `Docker Desktop` and navigate to the Images section. As shown in Figure 2, you will see the newly created image named `volatility-container` and labeled "latest."

There are two ways to start the container:

Option 1.- **Use `Docker Desktop`** You can start the container directly from `Docker Desktop` by clicking the Run button (see Figure 3). A configuration window will open, as shown in Figure 4, where you will need to specify:

- *Container name*: You can leave it as `volatility-container` or change it to a custom one.
- *Ports*: Make sure the host port is set to an available port. Since we will be connecting to the container via SSH, you will need to map a port on your host machine to port 22 on the container (see Figure 4).
- *Volumes*: Make sure the *host path* exists in your host. This will be the folder you can use to share files between the host and the container. Since we share large files (such as memory dumps) between the host and the container, it is important to properly configure the volume to ensure efficient access and avoid unnecessary data duplication, allowing for seamless forensic analysis without excessive disk space usage.

[2]Docker can be thought of as a "`chroot` on steroids", providing not only file system isolation like a `chroot`, but also process, network, and resource isolation through containerization, while adding features such as portability, version control, and easy deployment to different environments.

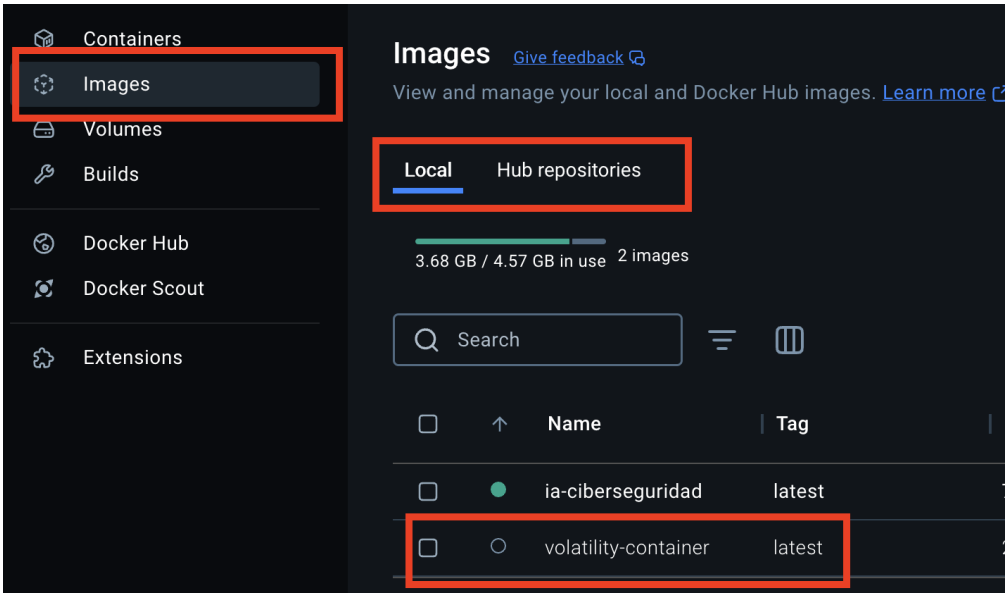

Figure 2: `Docker Desktop` showing the local images.

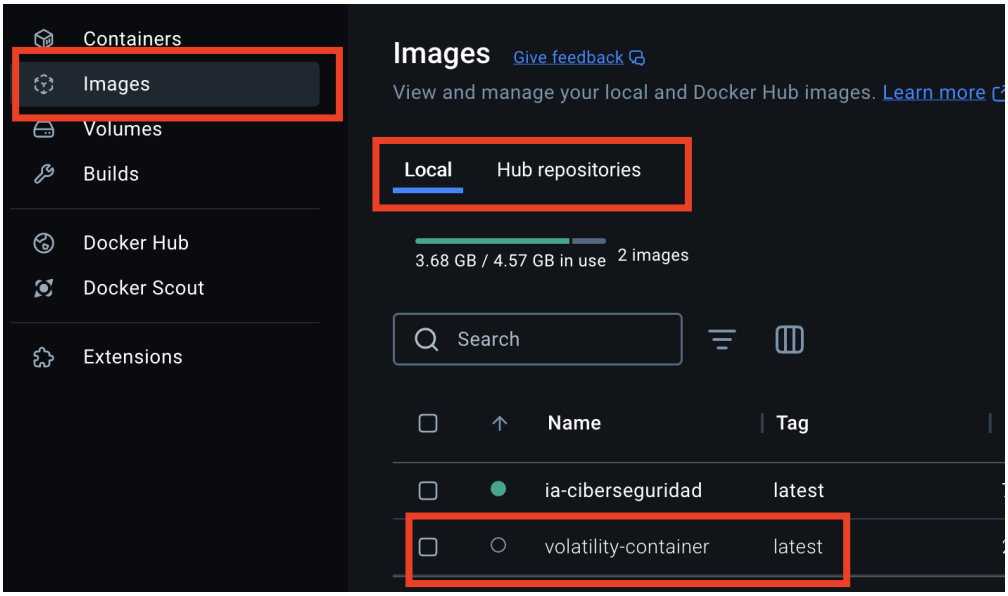

Figure 3: `Docker Desktop` showing the `volatility-container` image. Press Run to launch a new container of the `volatility-container` image.

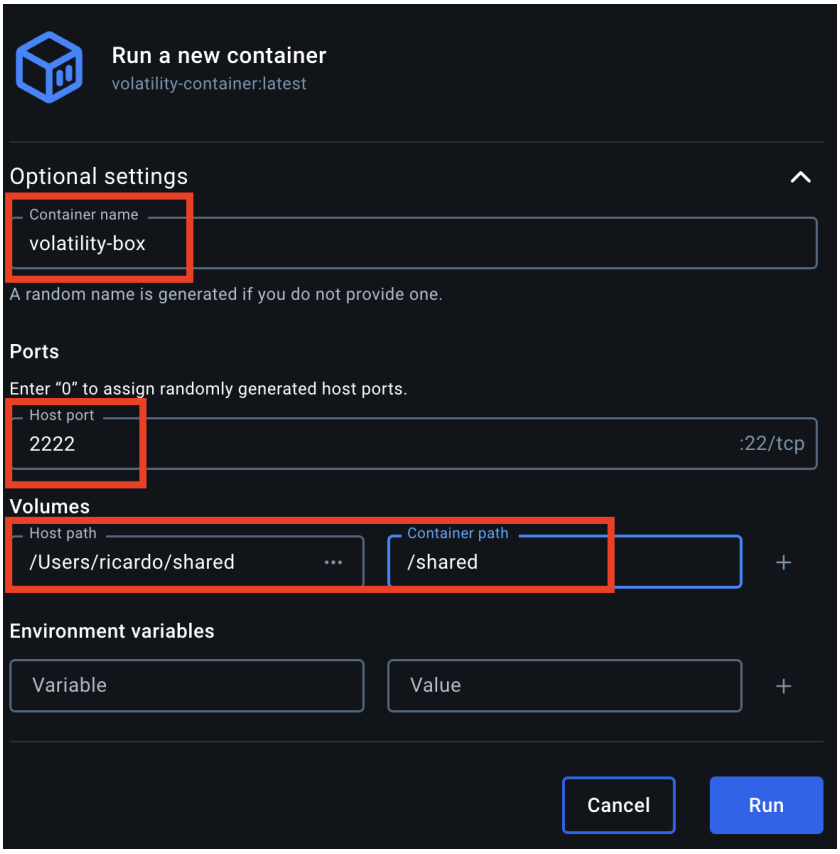

Figure 4: Container configuration in `Docker Desktop`.

After starting the container, you will see a log window showing the startup process, as shown in Figure 5. If you see something similar to this, the setup is complete and you're ready to go. You can now connect via SSH, access the shared folder inside the container, and use the configured environment for your Volatility and Volatility 3 analysis.

Option 2.- **Use the terminal**. To start the container from the terminal, open a new terminal window and run:

```
docker run -dit --name volatility-box -p 2222:22 \
       -v $(pwd)/shared:/shared volatility-container
```

The parameters used are:

- `-dit`: Run the container in interactive mode and in background, printing container ID at the end.
- `--name development-env`: Assigns a name to the container.
- `-v $(pwd)/shared:/shared`: Mounts the current host directory (left side of the `:` character, `$(pwd)/shared` in this example) inside the container (right side, path `/shared`) to share files between the host system and the container.

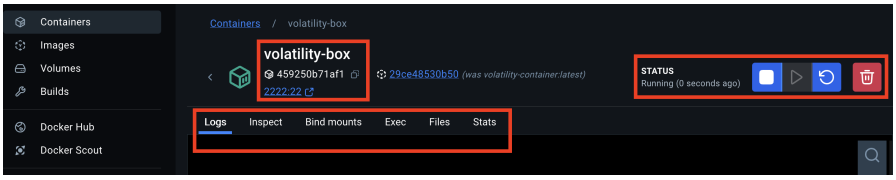

Figure 5: Execution log and control (stop, pause, reset, and destroy) of the running container in `Docker Desktop`.

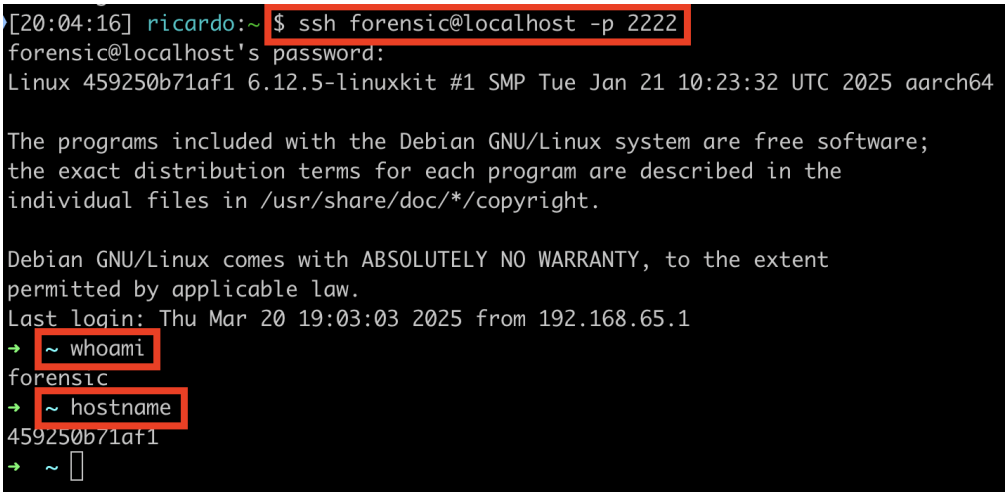

Figure 6: Test connection from the host machine to the container via SSH.

Finally, connect to the container via SSH. Open a terminal on your host machine and type:

```
ssh forensic@localhost -p 2222
```

You will be prompted for a password. Enter `forensic` and press ↵. Once logged into the container, you can test the container by running the commands `whoami` and `hostname`. The `whoami` command will return the current user inside the container, which should be `forensic`. The `hostname` command will display the name assigned to the container, which by default is a randomly generated identifier assigned by Docker (unless a specific hostname was set during container creation).

You should see something similar to Figure 6, confirming a successful SSH connection from the host to the Docker container – well done!.

This completes the setup of your Docker-based development environment for Volatility 2 and Volatility 3 analysis. You can now work inside the container, ensuring a controlled and reproducible configuration for your projects – *enjoy*! 🙂.

