Creating a Custom incus Image

Draft

1 Introduction

This document describes the process of creating a custom incus image for Debian system, which includes additional packages.

2 Requirements

The following are needed to create a custom image:

- incus installed
- Internet access
- Basic knowledge of package management in Debian

3 Creating a Base Container

The first step is to create a temporary Debian container that will serve as a base:

```
incus launch images:debian/12 temp-container

incus profile device remove default eth0
incus network attach incusbr0 temp-container eth0 eth0
```

4 firewall

... text ...

```
sudo iptables -t nat -A POSTROUTING -s 10.127.194.1/24 -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -s 10.127.194.1/24 -o eth0 -j ACCEPT
sudo iptables -A FORWARD -d 10.127.194.1/24 -m state --state \
   ESTABLISHED,RELATED -i eth0 -j ACCEPT
```

bash cmd script

```
f() { \
    # Sprawdź, czy parametr został podany \
    if [ -z "$1" ]; then \
        echo "Użycie: f <nazwa_interfejsu>"; \
        return 1; \
    fi; \
\
    # Nazwa interfejsu sieciowego przekazana jako parametr \
    INTERFACE="$1"; \
\
    # Odczytaj adres IP przypisany do incusbr0 \
    IP_ADDR=$(ip -4 addr show incusbr0 | grep -oP \
       '(?<=inet\s)\d+(\.\d+){3}'); \
\
    # Oblicz sieć na podstawie adresu IP \
    NETWORK=$(echo $IP_ADDR | sed 's/\.[0-9]*$/.0/24'); \
\
    # Dodaj reguły iptables \
    sudo iptables -t nat -A POSTROUTING -s $NETWORK -o $INTERFACE -j \
       MASQUERADE; \
    sudo iptables -A FORWARD -s $NETWORK -o $INTERFACE -j ACCEPT; \
    sudo iptables -A FORWARD -d $NETWORK -m state --state ESTABLISHED,RELATED \
       -i $INTERFACE -j ACCEPT; \
\
    echo "Reguły iptables zostały dodane dla sieci $NETWORK i interfejsu \
       $INTERFACE."; \
}; f eth0
```

5 Proxy

The given commands demonstrate how to configure proxy devices within Linux Containers (incus), specifically for forwarding network traffic from the host to containers. This can be useful for a variety of purposes, including exposing a service running inside a container to the outside network.

```
incus config device add incus-owrt0 owrt-proxy proxy listen=tcp:0.0.0.0:1234 \
   connect=tcp:0.0.0.0:80
incus config device add deb deb-proxy proxy listen=tcp:0.0.0.0:1234 \
   connect=tcp:0.0.0.0:1234
```

6 Installing Additional Packages

Next, use the `exec` command to launch a shell in the container and install the necessary packages:

```
incus exec temp-container -- apt update
incus exec temp-container -- apt install -y \
```

```
        netplan.io \
        sudo vim nano git tmux mc zip unzip curl wget htop lynx\
        iproute2 termshark bridge-utils \
        python3 python3-ipython python3-pyroute2 python3-scapy \
        docker.io docker-compose
```

7 Configuring Users and Permissions

After installing the additional packages, it's important to configure user access and permissions within the container.

7.1 Changing the Root Password

To change the root password to "passroot", execute the following command:

```
echo "root:passroot" | incus exec temp-container -- chpasswd
```

7.2 Adding a New User

To add a new user named "user" with the password "pass", and to add this user to the "sudo" and "docker" groups, follow these steps:

```
incus exec temp-container -- useradd -m -s /bin/bash user
echo "user:pass" | incus exec temp-container -- chpasswd
incus exec temp-container -- usermod -aG sudo user
incus exec temp-container -- usermod -aG docker user
```

This series of commands creates a new user with a home directory and bash shell, sets their password, and adds them to the necessary groups for system administration and Docker management.

8 Configuringing Vim and Tmux inside the Container

To install Vim-Plug, a plugin manager for Vim, inside the ‘deb‘ container, execute the following command from your host system:

```
incus exec temp-container -- bash -c "curl -fLo \
   /home/user/.vim/autoload/plug.vim --create-dirs \
    https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim"
```

Ensure that the specified user directory exists and the user has the necessary permissions to write to it.
To transfer a file from your host system to a container, use the ‘incus file push‘ command. For example, to push ‘_confs.zip‘ to the ‘deb‘ container:

```
incus file push _confs.zip temp-container/home/user/_confs.zip

#Folder _confs must to be zipped with -rj option
incus exec temp-container -- bash -c "unzip /home/user/_confs.zip -d \
   /home/user/ && sudo chown -R user:user /home/user/* /home/user/.*"
```

9 Cleaning the Container

Before creating the image, clean the system of unnecessary files:

```
incus exec temp-container -- apt clean
incus exec temp-container -- apt autoremove
```

10 Creating the Image

After installing all the necessary packages and cleaning the system, create an image from the container:

```
incus stop temp-container
incus publish temp-container --alias my-custom-image-deb
```

Replace `my-custom-image` with the chosen name for your image.

11 Cleanup

After creating the image, delete the temporary container:

```
incus delete temp-container
```

12 Using the Image

Now you can use your custom image to create new containers:

```
incus launch my-custom-image-deb deb

sudo incus network attach incusbr0 deb eth0 eth0

incus project set user-1000 restricted.containers.privilege allow

incus stop deb
incus config set deb security.nesting true
incus config set deb security.privileged true
incus start deb
```

13 Configuring Netplan Interface with DHCP

This section demonstrates the method to dynamically create and apply a Netplan configuration for setting up a network interface with DHCP on an incus/incus container. The procedure allows for the on-the-fly generation of the configuration file and its immediate application without manually creating or transferring the file.

13.1 Creating and Transferring the Configuration

To configure an interface named `v1a` with DHCP enabled, you can use shell commands combined with incus/incus functionalities. The following commands illustrate how to create the configuration dynamically and apply it directly:

13.1.1 Using the `echo` Command

... text ...

```
f() { \
sudo python3 veth.py \
    -ns1 $1 \
    -t1 incus \
    -n1 va-$3 \
    -b1 $2 \
    -ns2 $3 \
    -t2 incus \
    -n2 vb-$3 \
; }; f incus-owrt0 br-lan deb
```

13.1.2 Using the `echo` Command

This Bash function allows you to dynamically create a Netplan configuration file within an incus container. The function takes two parameters: the first parameter `$1` is the name of the container, and the second parameter `$2` is the name of the network interface. This name is also used to name the Netplan configuration file, resulting in a filename of the form '`$2.yaml`', where '`$2`' is replaced with the actual interface name provided as an argument. The configuration enables DHCP for the specified interface.

```
f() { incus exec "$1" -- bash -c "echo \"
network:
    version: 2
    ethernets:
        vb-$1:
            dhcp4: true\" > /etc/netplan/vb-$1.yaml"; }; f deb
```

To use this command, replace 'deb' with the actual name of your incus container, and 'v1b' with the name of the network interface you wish to configure. This command creates a Netplan configuration file in the '`/etc/netplan`' directory of the specified container, with the filename based on the interface name ('`v1b.yaml`' in this example).

13.1.3 Applying the Configuration with Netplan

After creating the Netplan configuration file, apply the changes to activate the network settings using the following command. This command uses the container name provided as the first argument `$1` to specify which incus container to apply the Netplan configuration in.

```
f() { incus exec "$1" -- sudo netplan apply; }; f deb
```

This command ensures that the newly created Netplan configuration is applied, enabling the network interface with DHCP as specified in the configuration file. Replace 'deb' with the actual name of your incus container when executing the command.

13.1.4 Checking Netplan Status Inside the Container

To verify the application of the Netplan configuration within an incus container, you could use a 'netplan status' command. This example uses the container name provided as the first argument `$1`.

```
f() { incus exec "$1" -- sudo netplan status; }; f deb
```

Replace 'deb' with the actual name of your incus container.

13.1.5 Cat Netplan conf file

... text ...

```
f() { incus exec $1 --  cat /etc/netplan/vb-$1.yaml;} ; f deb
```

14 Exporting and Importing the Image

To share the custom image with another incus host, you need to export and then import the image.

14.1 Exporting the Image

Export the image to a file:

```
    incus image export my-custom-image ./my-custom-image
```

14.2 Transferring the Image

Transfer the image file to the target incus host using scp or another file transfer method:

```
//scp ./my-custom-image.tar.gz user@target-host:/path/to/directory
incus file push ./my-custom-image.tar.gz deb/home.user \
   ./my-custom-image.tar.gz
```

14.3 Importing the Image on the Target Host

On the target incus host, import the image from the file:

```
incus image import /path/to/directory/my-custom-image.tar.gz --alias \
   my-custom-image
```

— ToDo —

15 Creating and Using an incus Image Repository

After customizing and publishing your incus image, you may want to share it across different hosts. An efficient way to do this is by creating an incus image repository and uploading your image there. This section outlines how to upload a custom image to a repository and then download it on a different host.

15.1 Setting Up an incus Image Repository

First, you need to set up an incus image repository. This could be a private server or a service that supports the incus image server protocol. For the sake of this example, we'll assume you have access to a server that can act as an incus image repository.

15.2 Uploading the Custom Image to the Repository

To upload your custom image to the repository, you will need to export it from your local incus, then upload it to your repository server. You can use SCP or any file transfer method preferred. Here's how to export and transfer the image:

```
incus image export my-custom-image ./my-custom-image.tar.gz
scp ./my-custom-image.tar.gz user@repository-server:/path/to/repository
```

Replace 'my-custom-image' with your image's alias, 'user' with your username on the repository server, and 'repository-server' with the server's address.

15.3 Importing the Image on the Repository Server

Log in to your repository server, and import the image into the incus image repository:

```
incus image import /path/to/repository/my-custom-image.tar.gz --alias my-custom-image
```

15.4 Accessing the Image from Another Host

On any other incus host that you want to use the custom image, you first need to add the repository server as a remote incus server:

```
incus remote add my-repo repository-server-url
```

Replace 'my-repo' with a name for the remote repository and 'repository-server-url' with the actual URL or IP address of your repository server.

Now, you can launch a new container using the custom image from the repository:

```
incus launch my-repo:my-custom-image my-new-container
```

15.5 Reconfiguring incus for Image Sharing

If you have already initialized incus on your server and need to adjust its configuration to share images, you may need to reconfigure certain settings. This can be necessary to enable image sharing or to adjust network settings for remote access. Here's how you can modify the incus configuration:

```
incus config set core.https_address [::]:8443
incus config set core.trust_password some-secret-password
```

The first command configures incus to listen for remote connections on all IPv4 and IPv6 addresses on port 8443, which is the default port used by incus for secure remote access. Replace 'some-secret-password' with a strong, unique password. This password will be used by remote incus clients to authenticate with your incus server.

Note: If your incus server is behind a firewall or NAT, you may need to set up port forwarding to ensure that remote hosts can connect to your incus server on the specified port (8443 by default).

After reconfiguring incus, you may need to restart the incus service for the changes to take effect. The command to restart incus depends on your system's init system. For systems using systemd, you can use:

```
sudo systemctl restart incus
```

Once incus is configured to accept remote connections and the service has been restarted, you can proceed with uploading images to the repository and accessing them from other hosts as described in the previous sections.

15.6 Conclusion

By setting up an incus image repository, you can easily share custom images across multiple hosts. This method simplifies the management of custom images and allows for efficient distribution of containerized environments.

15.7 Linkownia

http://mpabi.pl:1000/vlan/Screen%20recording%202024-03-01%2013.43.59.webm
http://mpabi.pl:1000/vlan/vlan-debs4.tar.gz