Docker Installation

This page explains how to run a VoipNow container using Docker containers on Windows, Mac or Linux.

- On Linux
- Windows 10
- MacOS

Get Help
If something is unclear or you need further assistance, do not hesitate to open a ticket in the **4PSA Support Zone** or ask a question in our monitored **GetSatisfaction community**.

On Linux

In order to set up VoipNow under Docker on a Linux machine, follow the steps below:

1. Stop Postfix on your Docker host. This step is necessary because the container will use the Docker host network stack and will fail to bind on port 25 if Postfix is already running on the host. Please stop Nginx/Apache or any other service you might have running on port 80 or 443 of your host.

   ```
   # systemctl stop postfix
   ```

2. Download a VoipNow image from Docker Hub. In this example, we have downloaded the latest VoipNow image. You can find the list of VoipNow images on hub.docker.com. Note that the tag name of the image corresponds to a specific **VoipNow release**.

   ```
   # docker pull 4psa/voipnow
   ```

3. Create the data container which will retain important VoipNow files. In the future, you will be able to update your Docker image while retaining existing data.

   ```
   # docker create --name data_voipnow 4psa/voipnow
   ```

4. Create the VoipNow container.

   ```
   docker run -d --stop-timeout 180 --net=host --ulimit nofile=248576:248576 --sysctl net.core.somaxconn=4096
   --shm-size 4G --volumes-from data_voipnow --name voipnow  4psa/voipnow
   ```

   You're free to adjust the parameters according to your preferences:

Parameter	Observation
`--stop-timeout 180`	When you stop a container, by default the docker waits for 10 seconds for the container to exit. When time expires, the docker will send SIGKILL to the container. Increase the stop timeout to 180 seconds so that the services can shutdown properly.
`--net=host`	For testing purposes, we will start the VoipNow docker container attached to the Docker host network. This will simplify the initial setup of the container since you don't have to configure VoipNow in order to run with NAT, but will also limit your machine to a single VoipNow container. For more details, see **Docker container networks**.
`--volumes-from data_voipnow`	Use your data container. In this example, we used *data_voipnow*.
`--name voipnow`	This is the name of your docker container.
`--ulimit nofile=248576:248576`	Increase the defaults limits of your container.
`--sysctl net.core.somaxconn=4096`	Increase the number of incoming connections.
`--shm-size 4G`	Configure the shared memory.

5. At this point, the VoipNow container is being initialized (passwords are generated, the database is populated, files are set up, etc.). This step takes approximately 5 minutes. To view the logs during the container initialization, run the following command:

   # docker logs -f voipnow

```
*** Running /etc/rc3.d/S09dahdi...
*** Running /etc/rc3.d/S10generate_certificates...
No DAHDI modules on the system. Not starting
*** Running /etc/rc3.d/S12rsyslog...
Starting system logger: [ OK ]
*** Running /etc/rc3.d/S15voipnow_postinstall_scripts...


------- OUTPUT OMMITED --------



*** Running /etc/rc3.d/S77voipnow...
Running in container, will not start...
*** Running /etc/rc3.d/S80postfix...
Starting VoipNow Web Management Interface: [ OK ]
*** Running /etc/rc3.d/S85httpsa...
*** Running /etc/rc3.d/S90crond...
httpsa running...
Starting crond: [ OK ]
*** Finished initializing VoipNow container
```

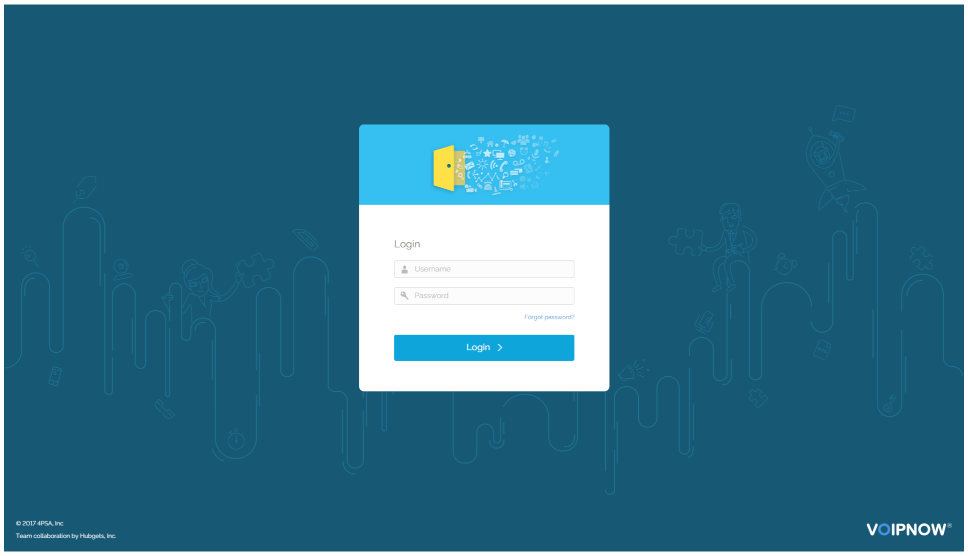
6. Set the `root` password for the VoipNow container.

   ```
   # docker exec -t -i voipnow /usr/bin/passwd
   ```

7. At this point, you should be able to access the VoipNow webinterface at `https://<YOUR_DOCKER_HOST_IP>`

   Log in to the interface with the following details:

   ```
   Username: admin
   Password: welcome
   ```

8. To access the VoipNow terminal, you can SSH to the container with the following command:

   ```
   # ssh -p 2222 root@<YOUR_DOCKER_HOST_IP>
   ```

Windows 10

In order to set up VoipNow under Docker on a Windows machine, follow the steps below:

1. Open a shell, for example cmd.exe.
2. Download a VoipNow image from Docker Hub. In this example, we have downloaded the latest VoipNow image. You can find the list of VoipNow images on **hub.docker.com**. Note that the tag name of the image corresponds to a specific **VoipNow release**.

   ```
   > docker pull 4psa/voipnow
   ```

3. Create the data container which will retain important VoipNow files. In the future, you will be able to update your Docker image while retaining existing data.

   ```
   > docker create --name data_voipnow 4psa/voipnow
   ```

4. Create the VoipNow container.

```
> docker run -d --stop-timeout 180 --net=host --ulimit nofile=248576:248576 --sysctl net.core.
somaxconn=4096 --shm-size 4G --volumes-from data_voipnow --name voipnow 4psa/voipnow
```

5. At this point, the VoipNow container is being initialized (passwords are generated, the database is populated, files are set up, etc.). This step takes approximately 5 minutes. To view the logs during the container initialization, run the following command:

```
> docker logs -f voipnow
*** Running /etc/rc3.d/S09dahdi...
*** Running /etc/rc3.d/S10generate_certificates...
No DAHDI modules on the system. Not starting
*** Running /etc/rc3.d/S12rsyslog...
Starting system logger: [ OK ]
*** Running /etc/rc3.d/S15voipnow_postinstall_scripts...

------- OUTPUT OMMITED --------

*** Running /etc/rc3.d/S77voipnow...
Running in container, will not start...
*** Running /etc/rc3.d/S80postfix...
Starting VoipNow Web Management Interface: [ OK ]
*** Running /etc/rc3.d/S85httpsa...
*** Running /etc/rc3.d/S90crond...
httpsa running...
Starting crond: [ OK ]
*** Finished initializing VoipNow container
```

6. Set the `root` password for the VoipNow container.

```
>  docker exec -t -i voipnow /usr/bin/passwd
```

7. Find the IP address of the container.

   > docker exec voipnow ip a sh dev hvint0

   4: hvint0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000

   link/ether 00:15:5d:03:e3:0a brd ff:ff:ff:ff:ff:ff

   inet 10.0.75.2/24 scope global hvint0

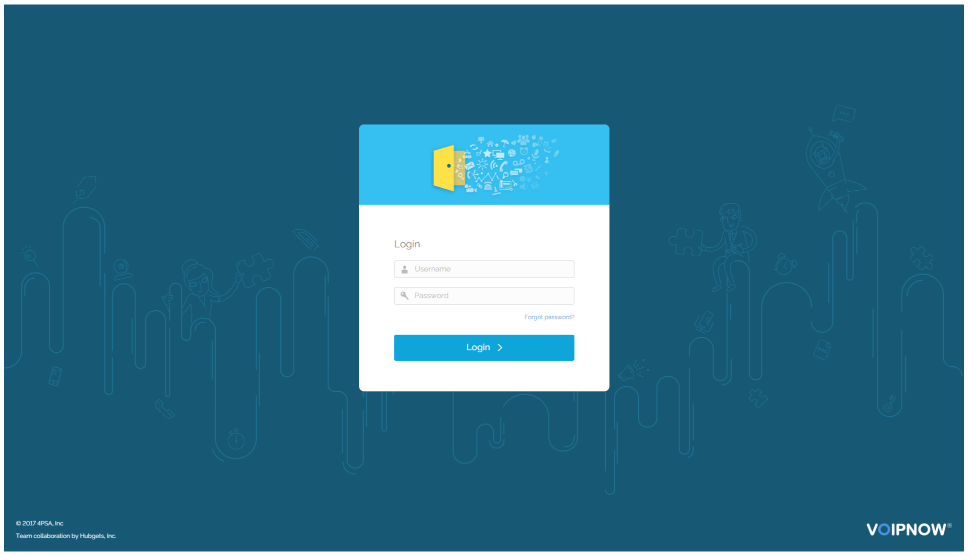
   valid_lft forever preferred_lft forever

   inet6 fe80::215:5dff:fe03:e30a/64 scope link

   valid_lft forever preferred_lft forever

8. Now you should be able to access the VoipNow webinterface at `https://<YOUR_DOCKER_HOST_IP>`
   Log in to the web interface with the following details:

```
Username: admin
Password: welcome
```



9. To access the VoipNow terminal, you can SSH to the container with the following command:

```
ssh -p 2222 root@<YOUR_CONTAINER_IP>
```

MacOS

In order to set up VoipNow under Docker on a macOS machine, please follow the steps below.

Due to **limitations** of **Docker for Mac**, we will show you how to start a VoipNow container on your MAC using **Docker Toolbox**.

1. **Install Docker Toolbox on macOS** and start the Docker Quickstart Terminal application. In this terminal, you can execute Docker commands.
2. Download a VoipNow image from Docker Hub. In this example, we have downloaded the latest VoipNow image. You can find the list of VoipNow images on **hub.docker.com**. Note that the tag name of the image corresponds to a specific **VoipNow release**.

   ```
   # docker pull 4psa/voipnow
   ```

3. Create the data container which will retain important VoipNow files. In the future, you will be able to update your Docker image while retaining existing data.

   ```
   # docker create --name data_voipnow 4psa/voipnow
   ```

4. Create the VoipNow container.

   ```
   docker run -d --stop-timeout 180 --net=host --ulimit nofile=248576:248576 --sysctl net.core.somaxconn=4096
   --shm-size 4G --volumes-from data_voipnow --name voipnow  4psa/voipnow
   ```

   You can adjust the parameters according to your preferencese:

Parameter	Observation
`--stop-timeout 180`	When stopping a container, by default, docker waits 10 seconds for container to exit. After this time expires, docker will send SIGKILL to container. Increase stop timeout to 180 seconds to allow services to shutdown properly.
`--net=host`	For testing purposes, we will start VoipNow docker container attached to Docker host network. This will simplify the initial setup of container as you don't have to configure VoipNow in order to run with NAT, but will also limit your machine to a single VoipNow container. For more details, see Docker container networks. Under the hood, VoipNow docker container will use network stack of VirtualBox virtual machine created by Docker Toolbox which runs Docker Engine Daemon.
`--volumes-from data_voipnow`	Use your data container, in this example *data_voipnow*.
`--name voipnow`	Name of your docker container.
`--ulimit nofile=248576:248576`	Increase defaults limits of your container.
`--sysctl net.core.somaxconn=4096`	Increase the number of incoming connections.
`--shm-size 4G`	Configure the shared memory.

5. At this point, the VoipNow container is being initialized (passwords are generated, the database is populated, files are set up, etc.). This step takes approximately 5 minutes. To view the logs during the container initialization, run the following command:

   ```
   # docker logs -f voipnow

   *** Running /etc/rc3.d/S09dahdi...
   *** Running /etc/rc3.d/S10generate_certificates...
   No DAHDI modules on the system. Not starting
   *** Running /etc/rc3.d/S12rsyslog...
   Starting system logger: [ OK ]
   *** Running /etc/rc3.d/S15voipnow_postinstall_scripts...


   ------- OUTPUT OMMITED --------
   ```

```
*** Running /etc/rc3.d/S77voipnow...
Running in container, will not start...
*** Running /etc/rc3.d/S80postfix...
Starting VoipNow Web Management Interface: [ OK ]
*** Running /etc/rc3.d/S85httpsa...
*** Running /etc/rc3.d/S90crond...
httpsa running...
Starting crond: [ OK ]
*** Finished initializing VoipNow container
```

6. Set the `root` password for VoipNow container.

   ```
   # docker exec -t -i voipnow /usr/bin/passwd
   ```

7. Find the IP address of the container.

   ```
   $ docker exec -t -i voipnow ip r s
   default via 10.0.2.2 dev eth0 metric 1
   10.0.2.0/24 dev eth0 proto kernel scope link src 10.0.2.15
   127.0.0.1 dev lo scope link
   172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1
   192.168.99.0/24 dev eth1 proto kernel scope link src 192.168.99.100
   ```
   In this example:

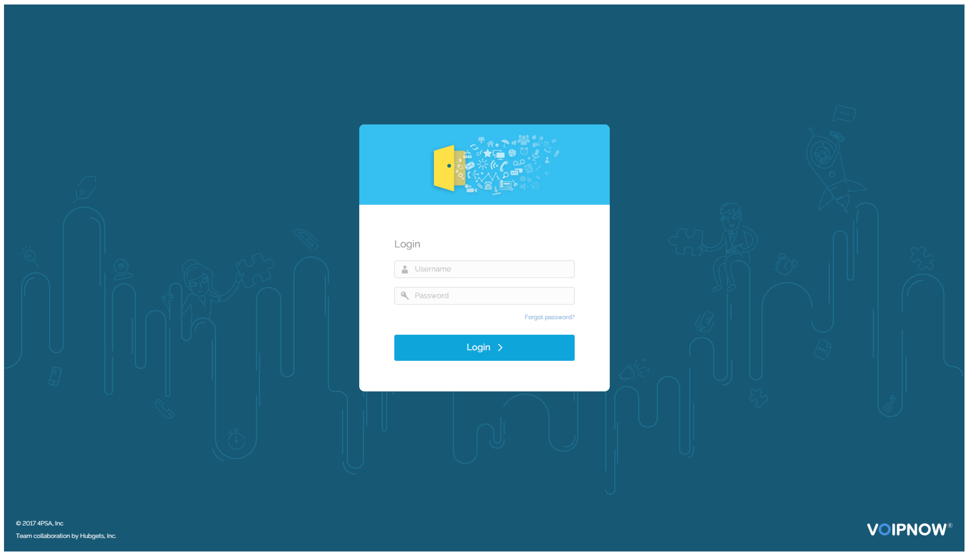
   a. eth0 is the VirtualBox NAT interface.
   b. eth1 is the VirtualBox Host-only Adapter.

   From your macOS, you can access VoipNow Docker Container using the IP address of the eth1 interface.

8. At this point, you should be able to access the VoipNow webinterface at `https://<YOUR_DOCKER_HOST_IP>`

   Log in to the web interface with the following details:

   ```
   Username: admin
   Password: welcome
   ```

9. To access the VoipNow terminal, you can SSH to the container with the following command:

   ```
   # ssh -p 2222 root@<YOUR_DOCKER_HOST_IP>
   ```