Contents

[Introduction to Mender 2](#_Toc48833365)

[Installation 2](#_Toc48833366)

[Download 2](#_Toc48833367)

[Installing Docker 2](#_Toc48833368)

[Set Up the Repository 2](#_Toc48833369)

[Install Docker Engine 3](#_Toc48833370)

[Installing Mender 3](#_Toc48833371)

[Downloading / Copying Pi OS 4](#_Toc48833372)

[Convert the Raspberry Pi Disk Image to Support Mender 4](#_Toc48833373)

[Created Output Images 6](#_Toc48833374)

[An Improved Workflow to Generate Artifacts 7](#_Toc48833375)

[Issues 7](#_Toc48833376)

[Artifact Name/Release 7](#_Toc48833377)

[Artifact Size and Initial Size 8](#_Toc48833378)

[Deployment Size Difference 9](#_Toc48833379)

[Mender-Hub 9](#_Toc48833380)

[Artifact Failure 9](#_Toc48833381)

[Extend Virtual Linux Space 10](#_Toc48833382)

[Clean Disk Space of Linux 10](#_Toc48833383)

[Some Other Methods 10](#_Toc48833384)

[Installing Mender in An Existing Linux System / Client 10](#_Toc48833385)

[Artifact Creation Tool 10](#_Toc48833386)

[Want to Study More About Mender Artifacts? 10](#_Toc48833387)

[Other Ways to Convert the Disk Image to Support Mender (mender-convert) 11](#_Toc48833388)

[Demo Server 11](#_Toc48833389)

[Installing Pre-Requisites 11](#_Toc48833390)

[Docker Permissions 11](#_Toc48833391)

[Docker Compose 1.6 12](#_Toc48833392)

[Install JQ 13](#_Toc48833393)

[Installing Demo Server of Mender 13](#_Toc48833394)

[Open the Mender UI 14](#_Toc48833395)

[Downloads 14](#_Toc48833396)

# Introduction to Mender

Mender is an open source over-the-air (OTA) software updater for embedded Linux devices. Mender comprises a client running at the embedded device, as well as a server that manages deployments across many devices.

This IOT platform can be used with multiple embedded Linux devices. I used this with Raspberry Pi 3 model B+.

Mender has two parts, one is the “client or device” that is remotely installed and needs frequent firmware upgrades. The second one is the “server”, that is used to deploy the firmware updates onto the client.

We can use Mender Professional I-e: Mender’s official server portal hosted on (<https://hosted.mender.io/ui/>) that has a web portal on which you can view devices, upload releases & create deployments on your device. Or Mender server webpage can be installed on-premise, inside your system.

Now, I’m using Mender Professional server portal, that will act as server. I’ll create initial image from it then burn it into Pi’s SD-card. Afterwards, create an artifact and upload on server, then deploy on the device.

# Installation

## Download

Install a Linux based OS on your system, I installed Ubuntu and it worked fine. The link for downloading Ubuntu is mentioned below:

<https://ubuntu.com/#download>

After installing the latest version of Ubuntu, ‘docker engine’ has to be installed. Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. Install docker from the link given below.

<https://docs.docker.com/engine/install/ubuntu/>

Install using the repository method, follow the steps given in the link. Then set up the repository. Afterwards, install it.

## Installing Docker

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

### Set Up the Repository

1. Update the apt package index and install packages to allow apt to use a repository over HTTPS:

$ sudo apt-get update

$ sudo apt-get install \

apt-transport-https \

ca-certificates \

curl \

gnupg-agent \

software-properties-common

1. Add Docker’s official GPG key:

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

Verify that you now have the key with the fingerprint 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88, by searching for the last 8 characters of the fingerprint.

$ sudo apt-key fingerprint 0EBFCD88

You will get an output like the one given below:

pub rsa4096 2017-02-22 [SCEA]

9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88

uid [ unknown] Docker Release (CE deb) <docker@docker.com>

sub rsa4096 2017-02-22 [S]

1. Use the following command to set up the stable repository. To add the nightly or test repository, add the word nightly or test (or both) after the word stable in the commands below.

$ sudo add-apt-repository \

"deb [arch=amd64] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) \

stable"

### Install Docker Engine

1. Update the apt package index, and install the latest version of Docker Engine and containerd.

$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli containerd.io

1. Verify that Docker Engine is installed correctly by running the hello-world image.

$ sudo docker run hello-world

This command downloads a test image and runs it in a container. When the container runs, it prints an informational message and exits. Now, Docker Engine is installed and running.

## Installing Mender

Build Docker image for mender-convert. Open a terminal in Ubuntu and clone the mender-convert repository into your system.

git clone -b 2.1.0b1 https://github.com/mendersoftware/mender-convert.git

Now, enter your mender-convert environment by changing the directory.

cd mender-convert

There is a utility script which can be used to generate the appropriate docker image to run mender-convert:

Use sudo with the command below:

./docker-build

Sit back, this command might take 2,3 mins. This will create a container image you can use to run mender-convert.

## Downloading / Copying Pi OS

Either download the raw Raspberry Pi disk image from raspberrypi.org or copy your own custom image file which is to be integrated with mender.

Make sure you copy the raw image into a subdirectory “input” inside the recently created mender-convert directory.

mkdir -p input

cd input

Now, if you want to download fresh Raspbian version then type this in the terminal or copy your own custom image.

wget http://downloads.raspberrypi.org/raspbian\_lite/images/raspbian\_lite-2020-02-07/2020-02-05-raspbian-buster-lite.zip

Extract the raw Raspberry Pi disk image:

unzip 2020-02-05-raspbian-buster-lite.zip && cd ..

## Convert the Raspberry Pi Disk Image to Support Mender

With the raw disk image and the container configured above, we can convert the image.

You can get your Mender Professional tenant token at the My organization page in Mender Professional portal.

./scripts/bootstrap-rootfs-overlay-hosted-server.sh \

--output-dir ${PWD}/rootfs\_overlay\_demo \

--tenant-token "Paste token from Mender Professional"

Now, remove the quotation marks and the text within and post your tenant token in that location. Like this:

./scripts/bootstrap-rootfs-overlay-hosted-server.sh --output-dir ${PWD}/rootfs\_overlay\_demo --tenant-token eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZW5kZXIudGVuYW50IjoiNWVmYzY3OGM2YTJmOTRiZTkzNmEyMDNkIiwiaXNzIjoiTWVuZGVyIiwic3ViIjoiNWVmYzY3OGM2YTJmOTRiZTkzNmEyMDNkIn0.iJUY8SHD0-vez3mcXk23TgXviUcIPR-lhxB26HFp8i8BQ47CQ7FjiXK7wNNQSh4uBe1ZK-9kNDAQO3PDUVBOXLzBgwarhETFEBBvtTJqWZmhOSY8qeJNbcOxPj07axpBOERVWkMF4KmpT3QckNkVEjew74HyUdvypR5jphYseE4kMbssPk\_fSXGP6BkkfTW\_7Mbs08izD8NYGX2htzCCZP4HxqTkWfmrD0oAolCKYk524zbDJpbMFzJQD8yj7qzlYIWGiNF47AT\_N9tubetsUYTFZdArhdXhfpLYrW5BwBKTisQBUNN73g1v9zyk2HfKppffeXC8iymPmo2l4abBbXuNT4l2nImgveYFE8sPWIm6F3GCN8oDrFV5pjZFsQgOXsaZCbt94EH0EodX23kT2dwlauLeUQuEoO0jwXGH6rXvNgPzLd6sTm5URJTjqXehVp-mxoMOUrFKFEWbPBwJYHFvP0BYIqb2UOcxEBsMvb4T\_Pgb9j-CR\_fERrphVRM

There are additional scripts in the scripts/ directory to configure with a local demo server, or production server.

Run mender-convert inside the container by running:

MENDER\_ARTIFACT\_NAME=release-1 ./docker-mender-convert \

--disk-image input/2020-02-05-raspbian-buster-lite.img \

--config configs/raspberrypi3\_config \

--overlay ./rootfs\_overlay\_demo

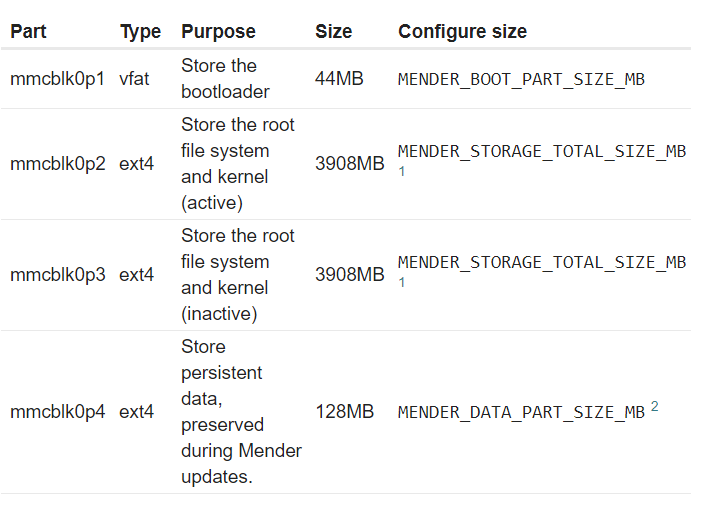
This “2020-02-05-raspbian-buster-lite.img” is the name of image file which is to be converted. If raw image is downloaded like in the above section then no need to change. Otherwise, for custom image; copy your custom image in “input” sub-directory of “mender-convert”, and copy its name and place it in the designated place of command.

Conversion will take 10-15 minutes, depending on your storage and resources available.

**NOTE:**

Change the MENDER\_ARTIFACT\_NAME when uploading new releases, for example; change “release-1” to “release2”. Otherwise, when you upload a new artifact but with the same name, it will say that artifact is same.

You will need an SD card with a capacity of at least 8GB when using the default arguments. The SD card layout will be the following using the default arguments:



1. The rootfs partition size is calculated using the following formula:

rootfs partition size = (storage-total-size-mb - data-part-size-mb - boot-part-size) / 2

2. You rarely need to adjust the data part size using MENDER\_DATA\_PART\_SIZE\_MB as this will expanded on first boot to occupy the remaining free blocks of the SD card.

You can read more about the partition layout required for Mender here (<https://docs.mender.io/hosted/devices/general-system-requirements#partition-layout>).

## Created Output Images

After a successful conversion, the images and artifacts are automatically placed in the “deploy” sub-directory of “mender-convert” directory:

* deploy/2020-02-05-raspbian-buster-lite-raspberrypi3-mender.img.gz
* deploy/2020-02-05-raspbian-buster-lite-raspberrypi3-mender.ext4
* deploy/2020-02-05-raspbian-buster-lite-raspberrypi3-mender.cfg
* deploy/2020-02-05-raspbian-buster-lite-raspberrypi3-mender.mender

If the name of your custom image is “My\_custom\_Image” then the output image files will be like:

* deploy/My\_custom\_Image-raspberrypi3-mender.img.gz
* deploy/My\_custom\_Image-raspberrypi3-mender.ext4
* deploy/My\_custom\_Image-raspberrypi3-mender.cfg
* deploy/My\_custom\_Image-raspberrypi3-mender.mender

For burning the image file on SD-card either use etcher (which I recommend) or use the dd command. Provide the path of SD-card attached.

zcat deploy/2020-02-05-raspbian-buster-lite-raspberrypi3-mender.img.gz | sudo dd of=<path to device> bs=4M && sudo sync

You can check the SD-card path by typing the command below, it will display all the devices attached:

lsblk -f

It will display like this:

sdb

└─sdb1 vfat 0065-5B05 /media/usman/0065-5B05

So, provide path like /dev/sd[drive letter], I did like this /dev/sdb1

If having difficulty in finding path, look into this:

<https://forum.manjaro.org/t/solved-error-when-trying-to-burn-iso-to-usb-using-linux-terminal/9736>

# An Improved Workflow to Generate Artifacts

The workflow of using an emulated device works for testing purposes, but it might have some limitations as we are emulating and not logged in to a real device or user.

When working with real deployments the recommended workflow is to have one **golden device**, that has not been converted to support Mender. On this device you carry out all the modifications you need, and then use the resulting SD card to create Mender Artifact files, in summary:

* flash your Rasbian image to the SD card
* boot the SD card, log in and make any modifications needed
* copy the SD card into an image on your workstation (e.g. using dd or Win32DiskImager)
* run mender-convert with the --disk-image option to generate a Mender Artifact (like above)
* upload the Artifact to your Mender server
* deploy it to your devices

Note that your golden device or SD card is not running Mender and is not modified during deployments. It is simply the “source” for generating the Artifacts that you deploy to the devices in the field.

# Issues

The issues arise while implementing over the air update for Raspberry Pi 3b+ will be discussed in this section. First, the errors / issues will be discussed in detail. Afterwards, their solutions will be laid down.

## Artifact Name/Release

When the Artifact was created that had the updated firmware, an error was displayed while deploying that artifact on the Mender Professional portal. Error displayed the message that Artifact already exists. After further investigating the error log of deployment it was revealed that the already installed firmware and the new one both had the same Artifact Name (some release name). Although it different inside but the names were same.

To fix this mistake, the name of the Artifact was changed and then again Mender-Convert command was executed but this time it had different name. Initially, by default Mender Artifact name was “release-1”, so the text in the quotation marks was replaced with “release-2”.

MENDER\_ARTIFACT\_NAME=release-2 ./docker-mender-convert \

--disk-image input/2020-02-05-raspbian-buster-lite.img \

--config configs/raspberrypi3\_config \

--overlay ./rootfs\_overlay\_demo

After doing this fix. The (.mender) file was uploaded in the releases section. When deployment was applied, it successfully completed.

## Artifact Size and Initial Size

Size of my custom image was almost 10,500 MB or 10.5 GB. By default, mender allocates 8192 MB so that it can fit in in 8GB. But this default allocation was not enough for the custom image, so rootfs partition size calculation formula was used. That gave a result of 10663MB (also error was caused when size was not increased initially, error asked that size required is 10663 MB. While allocated is 3908 MB).

Storage sizes are set in /mender-convert/configs/mender\_convert\_config file. Open it, make the changes then save it. Once, it saved now run the mender-convert artifact command for conversion according to your self-given sizes.

rootfs partition size = (storage-total-size-mb - data-part-size-mb - boot part size (40MB)) / 2

So, set the “MENDER\_STORAGE\_TOTAL\_SIZE\_MB” to be 22,000 MB by giving a margin of few hundred MBs. Remember that we have to allocate double the space of our custom image, as Mender creates two rootfs boot partitions, one is inactive, the other is active. In the total storage the boot part is also included. In this same fashion, default calculation was done, 8192 MB was allocated for two partitions, in which 3908 MB was for each rootfs partition.

Change was made in “mender\_convert\_config” named file, that is located in “configs” sub-directory of directory “mender\_convert”. After opening the file, line # 79 was edited. Initially it was like this:

MENDER\_STORAGE\_TOTAL\_SIZE\_MB="8192"

After the edit, it looked like:

MENDER\_STORAGE\_TOTAL\_SIZE\_MB="22000"

Apart from this boot partition size also needs to be adjusted for larger image files. When it gave the error earlier, the error asked to set the minimum boot size to be 250 MB. Default was 40 MB, then it was replaced with 256 MB so that it accumulates the custom image boot part size.

For editing the boot part size, same file is to be edited. Boot part size is defined at line # 83.

MENDER\_BOOT\_PART\_SIZE\_MB="40"

The 40, that indicated 40 MB boot partition size was replaced with “256”.

MENDER\_BOOT\_PART\_SIZE\_MB="256"

## Deployment Size Difference

Once a sdimg (image) file is flashed into Mender client package. Its rootfs partition sizes are fixed, and if the Artifact of a large custom file is created, uploaded in releases tab at Mender Professional, then it gives error on deployment. For example: the initial Mender client was installed with (.img) file of rootfs partition size 3908 MB, I-e: 8 GB in total for two rootfs plus other files. Than it won’t let the Mender Professional to deploy (.mender) file of large image (an image whose rootfs partition size is 11GB), and it covers a total of 23 GB. Hence, small to too big update cannot be deployed in Mender Convert because of fixed partition sizes set initially.

Make sure you are keeping 1-2 GB extra while creating the mender integrated image, this will help when creating deployments onto that image later on. The minimum rootfs partition size required is the amount of data you have on your custom image, so you have to allocate double plus the space for boot partition etc, along with that some extra must be assigned for future deployments.

## Mender-Hub

Mender Hub is a community for Mender users to discuss, contribute and maintain integrations for Mender on different boards and operating systems. Our goal is to enable OTA updates with Mender on every board and OS.

While performing the Raspberry Pi tutorial for interfacing Mender several issues were faced. This discussion forum helped in solving these issues. Most of the questions and their answers already existed. I was faced with a new kind of issue

## Artifact Failure

Initially when I performed testing with raw disk Raspbian image there was in issue when deploying updates. I searched the issue on Mender Hub but couldn’t get any answer. Then I made an account on Mender Hub and posted my question there.

One of their software developers replied to my query and asked for more details. I told him the steps performed from making raw image into Mender integrated to deploying Mender Artifacts. For more details please look into the link given below.

<https://hub.mender.io/t/artifact-deployment-failed-in-rpi/2185?u=usman>

Basically, the problem was with the (sd.img) image file. When Mender-Convert was used to create Mender integrated client using the (sd.img) image file an issue occurred while deploying the Artifact in the device that was converted into Mender client by using (sd.img) file. There was a bug in their software so they provided a solution and issued two pull requests on their Github repository. The solution provided had two parts, one was to edit “mender\_convert\_modify” file, while the other modified the “configs/raspberrypi\_config” file. In the first file, “sudo” was added before running /work/boot/dtb. In the “raspberrypi\_config” file, Drew Mosely added an “if” statement that prohibited converting an already converted image. Actually, when (sd.img) file was being used, the software was overwriting files and converting already converted image when Artifact was uploaded. After doing these two fixes, my Artifact was uploaded successfully.

Drew Mosely asked his colleges to review the pull request, after review from other developers the pull requests were comitted into the “mender-convert” Github repository.

## Extend Virtual Linux Space

I experienced issue of less allocated disk space, this came in when large image files were to be converted using Mender-Convert. First, they had to be copied in /mender-convert/input folder. Then after running mender command, the outputs are automatically placed /mender-convert/deploy folder. These outputs, plus the image in the input folder takes significant space if image files are larger.

So, you do not want to reinstall the Ubuntu/Linux distribution and also increase the size of disk space. Then follow the procedure laid down below:

Go to “Virtual Machine Settings”, Click the “Hard Disk” from the device options. Now, the right side of the window will display the capacity, disk information and disk utilities options. In the “disk utilities” there is an option called “Expand disk capacity”, click the button next to it; the “Expand..” button. Here, increase the size of your virtual machine disk space depending upon requirement.

## Clean Disk Space of Linux

When making images using Mender-Convert, disk storage space might run out if it is limited. Actually, the when old Artifact files or images are deleted, they don’t actually get removed from disk space. They remain in cache. So, they must be cleaned from the cache also. Go to disk usage analyzer in Ubuntu, just below the home directory there is a directory named “cache”, delete the contents of this directory and the accumulated disk space will be freed.

# Some Other Methods

## Installing Mender in An Existing Linux System / Client

Either upload the image file created from above method into your Client Linux system.

The link below describes how to install the Mender client in an existing Linux system. Installing this way does not offer a full Mender integration. However, it is possible to use Update Modules and update parts of the system.

<https://docs.mender.io/2.2/client-configuration/installing>

You can also download Mender client and install in device; the download link is in the downloads section below.

## Artifact Creation Tool

Either you can use the above section (Convert the Raspberry Pi disk image to support Mender) to create artifacts. The (.mender) file is the artifact, that is deployed in the device which is already running on Mender and needs a firmware update. Or you can use standalone artifact creation tool called mender-artifact utility. In the downloads section below, you’ll get a link to download it.

## Want to Study More About Mender Artifacts?

Go through this link if you want to explore and study about working of Mender Artifacts.

<https://docs.mender.io/2.2/architecture/mender-artifacts>

## Other Ways to Convert the Disk Image to Support Mender (mender-convert)

We used the Mender Professional method in this wiki.

./scripts/bootstrap-rootfs-overlay-hosted-server.sh \

--output-dir ${PWD}/rootfs\_overlay\_demo \

--tenant-token "Paste token from Mender Professional"

Well, there are some other options as well. Like using the Mender demo server.

./scripts/bootstrap-rootfs-overlay-demo-server.sh \

--output-dir ${PWD}/rootfs\_overlay\_demo \

--server-ip 192.168.1.1

Or using the Mender production server.

./scripts/bootstrap-rootfs-overlay-production-server.sh \

--output-dir ${PWD}/rootfs\_overlay\_demo \

--server-url https://foobar.mender.io \

[ --server-cert ~/server.crt ]

For further details on mender-convert visit the link given below:

<https://github.com/mendersoftware/mender-convert/tree/master>

## Demo Server

Instead of Mender Professional this method provides local server setup option with portal also running on local host. That is an open source solution for achieving OTA firmware upgrade in your client nodes.

### Installing Pre-Requisites

Install docker engine 17.03

<https://docs.mender.io/2.2/getting-started/on-premise-installation/requirements>

### Docker Permissions

Invoking the docker commands may fail when the local user has insufficient permissions to connect to the docker daemon. In Ubuntu 18.04, the user must be a member of the docker group to be able to access it. Please check the documentation for your host OS if you encounter connection issues with docker.

<https://docs.docker.com/engine/install/linux-postinstall/>

Manage Docker as a non-root user.

The Docker daemon binds to a Unix socket instead of a TCP port. By default, that Unix socket is owned by the user root and other users can only access it using sudo. The Docker daemon always runs as the root user.

If you don’t want to preface the docker command with sudo, create a Unix group called docker and add users to it. When the Docker daemon starts, it creates a Unix socket accessible by members of the docker group.

To create the docker group and add your user:

Create the docker group.

$ sudo groupadd docker

Add your user to the docker group.

$ sudo usermod -aG docker $USER

Log out and log back in so that your group membership is re-evaluated.

If testing on a virtual machine, it may be necessary to restart the virtual machine for changes to take effect. On a desktop Linux environment such as X Windows, log out of your session completely and then log back in.

On Linux, you can also run the following command to activate the changes to groups:

$ newgrp docker

Verify that you can run docker commands without sudo.

$ docker run hello-world

This command downloads a test image and runs it in a container. When the container runs, it prints an informational message and exits.

If you initially ran Docker CLI commands using sudo before adding your user to the docker group, you may see the following error, which indicates that your ~/.docker/ directory was created with incorrect permissions due to the sudo commands.

WARNING: Error loading config file: /home/user/.docker/config.json -

stat /home/user/.docker/config.json: permission denied

To fix this problem, either remove the ~/.docker/ directory (it is recreated automatically, but any custom settings are lost), or change its ownership and permissions using the following commands:

$ sudo chown "$USER":"$USER" /home/"$USER"/.docker -R

$ sudo chmod g+rwx "$HOME/.docker" -R

### Docker Compose 1.6

<https://docs.docker.com/engine/install/linux-postinstall/>

Install compose for Linux.

On Linux, you can download the Docker Compose binary from the Compose repository release page on GitHub. Follow the instructions from the link, which involve running the curl command in your terminal to download the binaries. These step-by-step instructions are also included below.

Run this command to download the current stable release of Docker Compose:

sudo curl -L "https://github.com/docker/compose/releases/download/1.26.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

To install a different version of Compose, substitute 1.26.2 with the version of Compose you want to use. If you have problems installing with curl, see Alternative Install Options tab above.

Apply executable permissions to the binary:

sudo chmod +x /usr/local/bin/docker-compose

Note: If the command docker-compose fails after installation, check your path. You can also create a symbolic link to /usr/bin or any other directory in your path.

For example:

sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

Optionally, install command completion for the bash and zsh shell.

Test the installation.

$ docker-compose --version

docker-compose version 1.26.2, build 1110ad01

### Install JQ

My Linux did not have “jq” installed, and command was giving issue. Also, it was a pre-requisite, so installed it using the link below.

<https://stedolan.github.io/jq/download/>

Or use this command to install “jq” in your Linux distribution.

sudo apt-get install jq

### Installing Demo Server of Mender

Go through the steps to run demo server of mender on your own system. For detailed steps check the link given below.

<https://docs.mender.io/2.2/getting-started/on-premise-installation/create-a-test-environment>

Bring up the environment with Docker Compose. In a working directory, download the Mender integration environment:

git clone -b 2.2.2 https://github.com/mendersoftware/integration.git integration-2.2.2

cd integration-2.2.2

Mender comes with a wrapper script that brings up the environment with Docker Compose. Running this script will pull down the images and start them:

./demo up

Note: SAVE LOGIN CREDENTIALS

After the Docker images have been downloaded, the demo up script starts the Mender services, adds a demo user with the username mender-demo@example.com, and assigns a random password. Note that this password is not stored anywhere in the Mender demo environment. Make sure to copy this password for logins to this instance of the demo environment. You can change it after you log in to the Mender UI (below).

### Open the Mender UI

Open the local host and perform your OTA firmware update. It is just like Mender Professional, but remember to set your IP’s accordingly before running local host Mender UI.

# Downloads

For downloading Mender Artifact tool & Mender client go to this link:

<https://docs.mender.io/2.2/downloads>