**Registry server and docker compose**

# Step 1. Deploy a registry server

Before you can deploy a registry, you need to install Docker on the host. A registry is an instance of the registry image, and runs within Docker.

This Lab provides basic information about deploying and configuring a registry. If you have an air-gapped datacenter, refer for Considerations for air-gapped registries.

## Run a local registry

Use a command like the following to start the registry container:

$ docker run -d -p 5000:5000 --restart=always --name registry registry:2

The registry is now ready to use.

**Warning**: These first few examples show registry configurations that are only appropriate for testing. A production-ready registry must be protected by TLS and should ideally use an access-control mechanism.

## Copy an image from Docker Hub to your registry

You can pull an image from Docker Hub and push it to your registry. The following example pulls the ubuntu:16.04 image from Docker Hub and re-tags it as my-ubuntu, then pushes it to the local registry. Finally, the ubuntu:16.04 and my-ubuntu images are deleted locally and the my-ubuntu image is pulled from the local registry.

1. Pull the ubuntu:16.04 image from Docker Hub.

$ docker pull ubuntu:16.04

1. Tag the image as localhost:5000/my-ubuntu. This creates an additional tag for the existing image. When the first part of the tag is a hostname and port, Docker interprets this as the location of a registry, when pushing.

$ docker tag ubuntu:16.04 localhost:5000/my-ubuntu

1. Push the image to the local registry running at localhost:5000:

$ docker push localhost:5000/my-ubuntu

1. Remove the locally-cached ubuntu:16.04 and localhost:5000/my-ubuntu images, so that you can test pulling the image from your registry. This does not remove the localhost:5000/my-ubuntu image from your registry.

$ docker image remove ubuntu:16.04

$ docker image remove localhost:5000/my-ubuntu

1. Pull the localhost:5000/my-ubuntu image from your local registry.

$ docker pull localhost:5000/my-ubuntu

## Stop a local registry

To stop the registry, use the same docker container stop command as with any other container.

$ docker container stop registry

To remove the container, use docker container rm.

$ docker container stop registry && docker container rm -v registry

## Basic configuration

To configure the container, you can pass additional or modified options to the docker run command.

The following sections provide basic guidelines for configuring your registry. For more details, see the [registry configuration reference](https://docs.docker.com/registry/configuration/).

### Start the registry automatically

If you want to use the registry as part of your permanent infrastructure, you should set it to restart automatically when Docker restarts or if it exits. This example uses the --restart always flag to set a restart policy for the registry.

$ docker run -d \

-p 5000:5000 \

--restart=always \

--name registry \

registry:2

### Customize the published port

If you are already using port 5000, or you want to run multiple local registries to separate areas of concern, you can customize the registry’s port settings. This example runs the registry on port 5001 and also names it registry-test. Remember, the first part of the -p value is the host port and the second part is the port within the container. Within the container, the registry listens on port 5000 by default.

$ docker run -d \

-p 5001:5000 \

--name registry-test \

registry:2

If you want to change the port the registry listens on within the container, you can use the environment variable REGISTRY\_HTTP\_ADDR to change it. This command causes the registry to listen on port 5001 within the container:

$ docker run -d \

-e REGISTRY\_HTTP\_ADDR=0.0.0.0:5001 \

-p 5001:5001 \

--name registry-test \

registry:2

# Step 2 Installing Compose

## Install Docker Compose

You can run Compose on macOS, Windows, and 64-bit Linux.

## Prerequisites

Docker Compose relies on Docker Engine for any meaningful work, so make sure you have Docker Engine installed either locally or remote, depending on your setup.

* On desktop systems like Docker for Mac and Windows, Docker Compose is included as part of those desktop installs.
* On Linux systems, first install the [Docker](https://docs.docker.com/install/#server) for your OS as described on the Get Docker page, then come back here for instructions on installing Compose on Linux systems.
* To run Compose as a non-root user, see [Manage Docker as a non-root user](https://docs.docker.com/install/linux/linux-postinstall/).

## Install Compose

Follow the instructions below to install Compose on Mac, Windows, Windows Server 2016, or Linux systems, or find out about alternatives like using the pip Python package manager or installing Compose as a container.

* Mac
* Windows
* Linux
* Alternative Install Options

### Install Compose on Linux systems

On **Linux**, you can download the Docker Compose binary from the [Compose repository release page on GitHub](https://github.com/docker/compose/releases). These step by step instructions are also included below.

1. Run this command to download the latest version of Docker Compose:

$ sudo curl -L https://github.com/docker/compose/releases/download/1.19.0/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose

Use the latest Compose release number in the download command.

The above command is an example, and it may become out-of-date. To ensure you have the latest version, check the [Compose repository release page on GitHub](https://github.com/docker/compose/releases).

If you have problems installing with curl, see [Alternative Install Options](https://docs.docker.com/compose/install/#alternative-install-options) tab above.

1. Apply executable permissions to the binary:

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

1. Optionally, install [command completion](https://docs.docker.com/compose/completion/) for the bash and zsh shell.
2. Test the installation.

$ docker-compose --version

docker-compose version 1.19.0, build 1719ceb

## Upgrading

If you’re upgrading from Compose 1.2 or earlier, remove or migrate your existing containers after upgrading Compose. This is because, as of version 1.3, Compose uses Docker labels to keep track of containers, and your containers need to be recreated to add the labels.

If Compose detects containers that were created without labels, it refuses to run so that you don’t end up with two sets of them. If you want to keep using your existing containers (for example, because they have data volumes you want to preserve), you can use Compose 1.5.x to migrate them with the following command:

$docker-compose migrate-to-labels

Alternatively, if you’re not worried about keeping them, you can remove them. Compose just creates new ones.

$docker container rm -f -v myapp\_web\_1 myapp\_db\_1 ...

# Step 3 Deploying registry as compose file

## Deploy your registry using a Compose file

If your registry invocation is advanced, it may be easier to use a Docker compose file to deploy it, rather than relying on a specific docker run invocation. Use the following example docker-compose.yml as a template.

registry:

restart: always

image: registry:2

ports:

- 5000:5000

environment:

REGISTRY\_HTTP\_TLS\_CERTIFICATE: /certs/domain.crt

REGISTRY\_HTTP\_TLS\_KEY: /certs/domain.key

REGISTRY\_AUTH: htpasswd

REGISTRY\_AUTH\_HTPASSWD\_PATH: /auth/htpasswd

REGISTRY\_AUTH\_HTPASSWD\_REALM: Registry Realm

volumes:

- /path/data:/var/lib/registry

- /path/certs:/certs

- /path/auth:/auth

Replace /path with the directory which contains the certs/ and auth/ directories.

**Registry container & UI**

Here is the template of the docker-compose.yml file for a basic local docker registry:

version: '2'

services:

lb:

image: dockercloud/haproxy:1.6.2

links:

- registry

- registry-ui

ports:

- '80:80'

- '443:443'

- '5000:5000'

restart: always

volumes:

- /var/run/docker.sock:/var/run/docker.sock

registry:

build: ./registry

restart: always

expose:

- 5000

environment:

TCP\_PORTS: '5000'

VIRTUAL\_HOST: '\*:5000, [https://\*:5000](https://*:5000)'

FORCE\_SSL: 'true'

REGISTRY\_STORAGE\_DELETE\_ENABLED: 'true'

registry-ui:

image: konradkleine/docker-registry-frontend:v2

restart: always

environment:

VIRTUAL\_HOST: '\*, https://\*'

ENV\_DOCKER\_REGISTRY\_HOST: 'registry'

ENV\_DOCKER\_REGISTRY\_PORT: 5000

links:

- registry

expose:

- 80

## Services

**HAProxy**

This service is the load balancer. The only thing we had to do, is to bind port 5000 in order to redirect the traffic to the registry.

**Registry**

This service is the docker registry. A lot of configuration was required:

* Set the TCP\_PORTS and VIRTUAL\_HOST environment variable
  + This is required for HAProxy to redirect all traffic from port 5000 to this service
* Set registry service specific environment variables:
  + – REGISTRY\_STORAGE\_DELETE\_ENABLED=true: otherwise, the registry does not support deleting images

**Registry UI (Docker Registry Frontend)**

This service hosts a very simple docker UI name docker-registry-frontend by Konrad Kleine. In this service, not so much was required to be configured:

* Set the VIRTUAL\_HOST environment variable
  + This is required for HAProxy to redirect all traffic (not already taken care of ) to this service
* Set the registry-ui service specific environment variables:
  + ENV\_DOCKER\_REGISTRY\_HOST=registry : name of the service for which a link exists
  + ENV\_DOCKER\_REGISTRY\_PORT=5000 : the port on which the registry listens to

To start the registry locally, simply run this command:

$ docker-compose up -d

The registry is reachable at localhost:5000.

The registry UI is reachable <http://localhost:80>.

### Repeat steps 1-5 as described in section for [copying an image to your docker registry](#_Copy_an_image)

Consider the following statements about registry container, which is:

1. Running locally
2. Not using any authentication mechanism
3. Storing docker images in the container only.
4. If you want to persist it for some reason, add this volume to the registry service definition
5. Not using SSL