Red Hat

Automating Container Management with Ansible

Building Container Images

© 2020 Red Hat, Inc., licensed to Pluralsight, LLC. All trademarks, service marks, and logos used herein are the property of their respective owners.

Objectives

This module demonstrates how to:

- Create container images with a Dockerfile.
- Automate creation of container images with Docker modules in Ansible.

© 2020 Red Hat, Inc., licensed to Pluralsight, LLC. All trademarks, service marks, and logos used herein are the property of their respective owners.

Writing Dockerfiles

© 2020 Red Hat, Inc., licensed to Pluralsight, LLC. All trademarks, service marks, and logos used herein are the property of their respective owners.

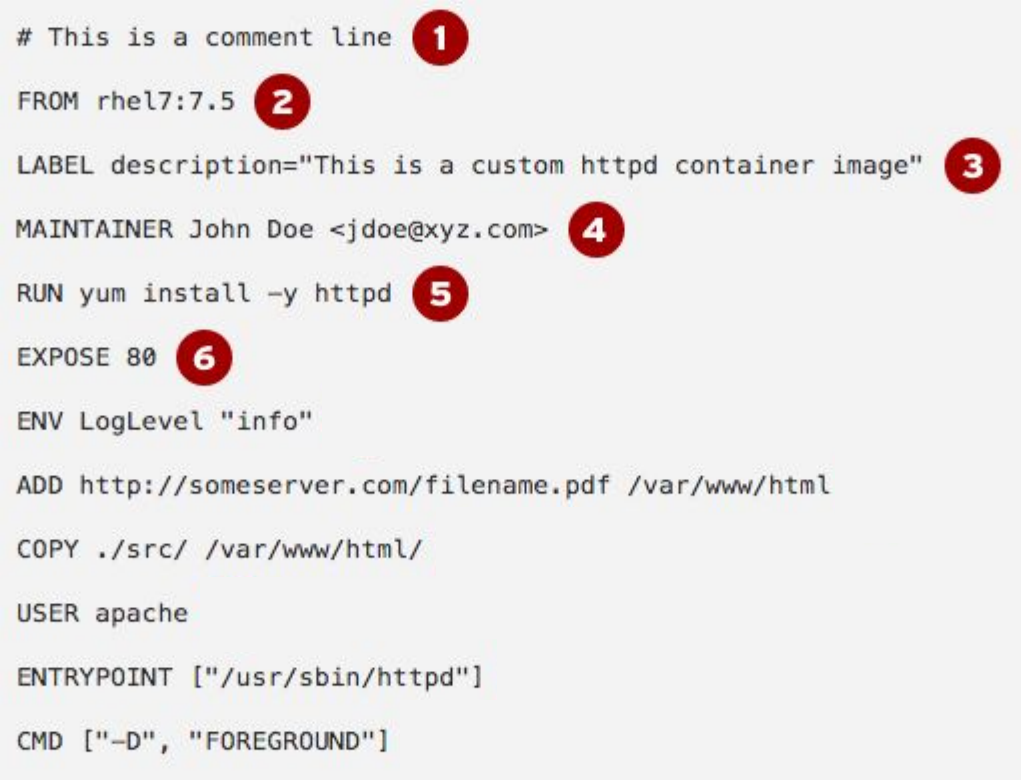
Dockerfile

- A Dockerfile is a text file used by Docker to automate the building of container images.
- Specifies a base image with which to start.
- Uses a set of steps called instructions to modify the base image to create a final image.
- Each instruction creates a new layer on the image that is used to build the final container image. Instructions in a Dockerfile are executed in the order they appear.
- Using a Dockerfile is the suggested approach to building Docker images because it controls which files are added to each layer.

© 2020 Red Hat, Inc., licensed to Pluralsight, LLC. All trademarks, service marks, and logos used herein are the property of their respective owners.

Dockerfile Example

1. Lines that begin with a pound sign (**#**) are comments.
2. The new container image will be constructed using **rhel7:7.5** container base image. Dockerfiles must begin with the **FROM** instruction.
3. **LABEL** is responsible for adding generic metadata to an image. A **LABEL** is a simple key/value pair.
4. **MAINTAINER** is responsible for setting the Author field of the generated container image.
5. **RUN** executes commands in a new layer on top of the current image, then commits the results.
6. **EXPOSE** indicates that the container listens on the specified network port at runtime. The **EXPOSE** instruction defines metadata only; it does not make ports accessible from the host.

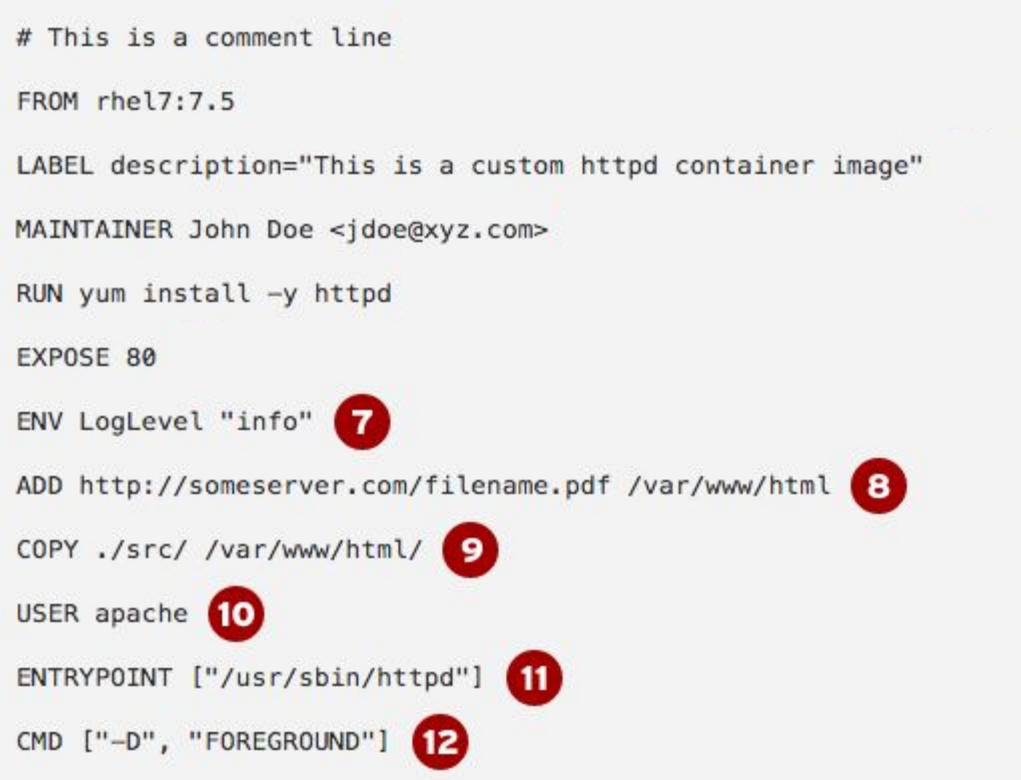


```
# This is a comment line 1
FROM rhel7:7.5 2
LABEL description="This is a custom httpd container image" 3
MAINTAINER John Doe <jdoe@xyz.com> 4
RUN yum install -y httpd 5
EXPOSE 80 6
ENV LogLevel "info"
ADD http://someserver.com/filename.pdf /var/www/html
COPY ./src/ /var/www/html/
USER apache
ENTRYPOINT ["/usr/sbin/httpd"]
CMD ["-D", "FOREGROUND"]
```

© 2020 Red Hat, Inc., licensed to Pluralsight, LLC. All trademarks, service marks, and logos used herein are the property of their respective owners.

Dockerfile Example, Continued

7. **ENV** is responsible for defining environment variables that will be available to the container.
8. **ADD** copies files from local or remote source and adds them to the container's file system.
9. **COPY** only copies files from local source and adds them to the container's file system.
10. **USER** specifies the username or the UID to use when running the container image for the **RUN**, **CMD**, and **ENTRYPOINT** instructions.
11. **ENTRYPOINT** specifies the default command to execute when the container is created.
12. **CMD** provides the default arguments for the **ENTRYPOINT** instruction.



```
# This is a comment line
FROM rhel7:7.5
LABEL description="This is a custom httpd container image"
MAINTAINER John Doe <jdoe@xyz.com>
RUN yum install -y httpd
EXPOSE 80
ENV LogLevel "info"
ADD http://someserver.com/filename.pdf /var/www/html
COPY ./src/ /var/www/html/
USER apache
ENTRYPOINT ["/usr/sbin/httpd"]
CMD ["-D", "FOREGROUND"]
```

© 2020 Red Hat, Inc., licensed to Pluralsight, LLC. All trademarks, service marks, and logos used herein are the property of their respective owners.

Docker Build

- The **`docker build`** command processes the `Dockerfile` and builds a new image based on the instructions it contains.

- The syntax for this command is as follows:

```
$ docker build -t NAME:TAG DIR
```

  - *`DIR`* is the path to the working directory. It can be the current directory (.).
  - *`NAME:TAG`* is a name with a tag that is assigned to the new image. It is specified with the **`-t`** option. If the *`TAG`* is not specified, then the image is automatically tagged as **`latest`**.

  - Note that the build is run by the Docker daemon, not by the **`docker`** CLI command.

© 2020 Red Hat, Inc., licensed to Pluralsight, LLC. All trademarks, service marks, and logos used herein are the property of their respective owners.

Creating Container Images with Ansible

© 2020 Red Hat, Inc., licensed to Pluralsight, LLC. All trademarks, service marks, and logos used herein are the property of their respective owners.

Docker Modules

The Ansible modules for Docker require the Docker Engine API Python library. To install run the following inside a virtualenv:

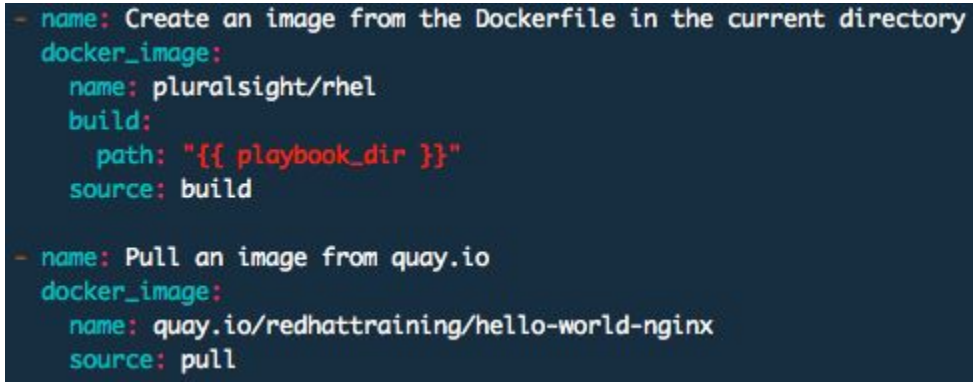
```
    # pip install docker
```

© 2020 Red Hat, Inc., licensed to Pluralsight, LLC. All trademarks, service marks, and logos used herein are the property of their respective owners.

Docker Image Module

The `docker_image` Ansible module can be used to automate image creation from a Dockerfile.

It can also be used to retrieve an existing image from a container registry (such as docker.io or quay.io), tag an image into a repository, and archive an image to a tar file.



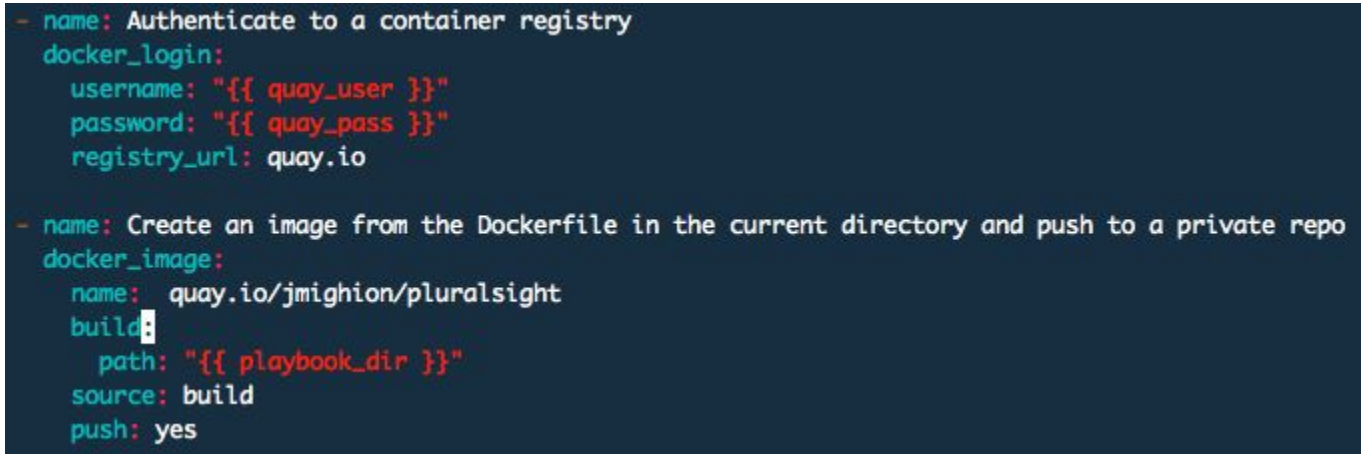
```yaml
- name: Create an image from the Dockerfile in the current directory
  docker_image:
    name: pluralsight/rhel
    build:
      path: "{{ playbook_dir }}"
    source: build

- name: Pull an image from quay.io
  docker_image:
    name: quay.io/redhattraining/hello-world-nginx
    source: pull
```

© 2020 Red Hat, Inc., licensed to Pluralsight, LLC. All trademarks, service marks, and logos used herein are the property of their respective owners.

Docker Login Module

The `docker_login` Ansible module can be used to authenticate to a container registry and add credentials to your local configuration file. The module provides functionality similar to the **`docker login`** command.



```
- name: Authenticate to a container registry
  docker_login:
    username: "{{ quay_user }}"
    password: "{{ quay_pass }}"
    registry_url: quay.io

- name: Create an image from the Dockerfile in the current directory and push to a private repo
  docker_image:
    name:  quay.io/jmighion/pluralsight
    build:
      path: "{{ playbook_dir }}"
    source: build
    push: yes
```

© 2020 Red Hat, Inc., licensed to Pluralsight, LLC. All trademarks, service marks, and logos used herein are the property of their respective owners.