Deploying Web Application with Docker

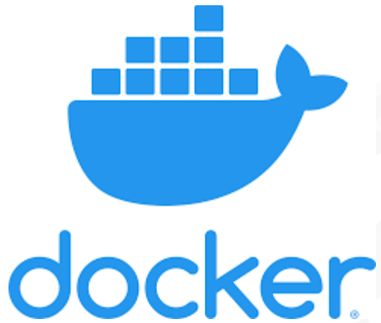
simplilearn

You Already Know

Course(s):

1. Docker Basics and Jenkins integration with Docker

2. Develop and Deploy Modern Web Applications with Docker



Recap

- Docker Basics and Jenkins integration with Docker
  - Introduction to Docker
  - Basic docker commands
  - Docker swarm

- Develop and Deploy Modern Web Applications with Docker
  - Docker containers
  - Configure and deploy pipelines
  - Deploying application on docker containers

Recap

- Explain Docker Compose, Docker Swarm, and Docker Images
  - What is Docker?
  - Docker Architecture
  - Containers vs. Virtual Machines
  - Docker Command Line
  - Docker Commands
  - Exporting a port
  - Docker run
  - Docker Images
  - Docker compose
  - Docker swarm

A Day in the Life of a MERN Stack Developer

Joe, a MERN Stack Developer in an IT company, has been working very hard to deliver all projects adhering to the expected quality.

Joe has been assigned a new task during sprint planning. He needs to configure a Jenkins job to perform Docker image build and configure Swarm service on Swarm master. He needs to finally deploy the Docker image to Docker Swarm.

In this lesson, we will learn how to solve this real-world scenario and help Joe effectively complete his task.

Learning Objectives

By the end of this lesson, you will be able to:

- Build an image and run it as a container
- Push an image to Docker Hub
- Deploy a Docker container on Docker Swarm
- Scale a Docker container on Docker Swarm
- Set up Jenkins pipeline
- Install and use Docker Community Edition

Docker Image

Docker Image

- Docker is based on images
- Images are a file that is composed of a file system and parameters

Syntax

```
docker images
```

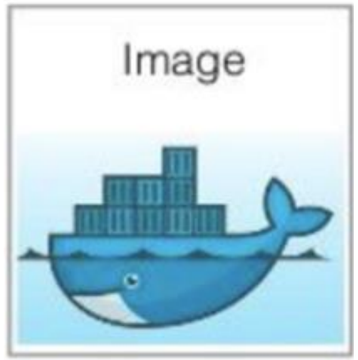
- Let's take an example of the following command:

Docker command

Docker program on the operating system

run command

Creates instance of an image, which is then called a container



Docker image

Build a Custom Docker Image

Duration: 15 min.

Problem Statement:

Demonstrate how to build a Docker image and run it as a container.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to build an image and execute:

1. Build a custom Docker image
2. Initialize a custom container on Docker host
3. Validate connectivity using the curl command
4. Push the code to GitHub repositories

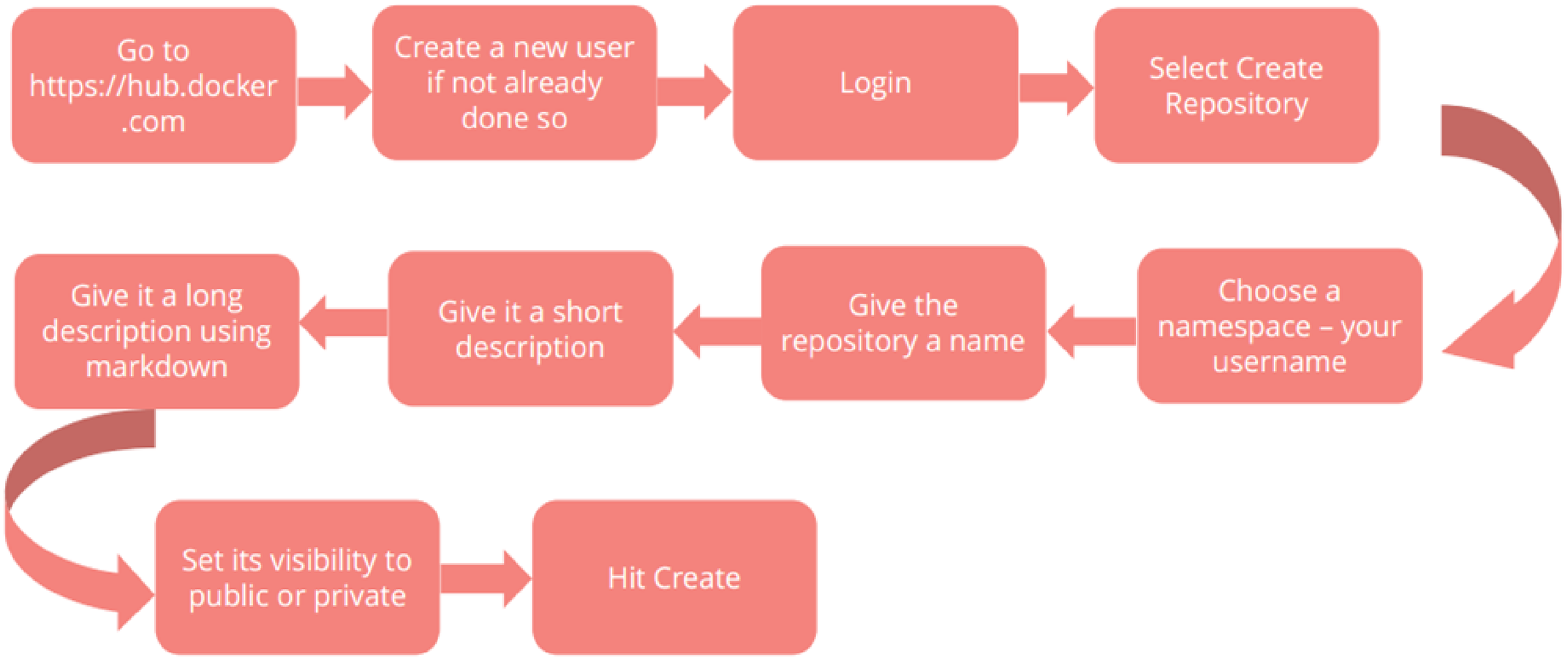
Introduction to Docker Hub

Docker Hub

Docker Hub is a hosted repository service provided by Docker to find and share container images with your team. Key features include:

- Private Repositories: Push and pull container images
- Automated Builds: Automatically build container images from GitHub and Bitbucket and push them to Docker Hub
- Teams & Organizations: Manage access to private repositories
- Official Images: Pull and use high-quality container images provided by Docker
- Publisher Images: Pull and use high-quality container images provided by external vendors. Certified images also include support and guarantee compatibility with Docker Enterprise
- Webhooks: Trigger actions after a successful push to a repository to integrate Docker Hub with other services

Docker Hub Configuration: Step-by-Step Procedure



Docker Hub Login

Docker login command is used to connect to Docker Hub from Docker client CLI. Before doing this, you must tag existing image with Docker image concatenated with Docker username.

Syntax

```
docker tag <image_name><tag> <docker_hub_username>/ <image_name><tag>
docker push <docker_hub_username>/ <image_name><tag>
```

```
root@docker:~# docker tag docker_apache anujsharma1990/docker_apache
root@docker:~# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: anujsharma1990
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@docker:~# docker push anujsharma1990/docker_apache
The push refers to repository [docker.io/anujsharma1990/docker_apache]
e7e6e790a930: Pushed
735b1c81b582: Pushed
76c033092e10: Mounted from library/ubuntu
2146d867acf3: Mounted from library/ubuntu
ae1f631f14b7: Mounted from library/ubuntu
102645f1cf72: Mounted from library/ubuntu
latest: digest: sha256:7d8cccf25ef3b9f1578f0efb2d26a0c67cacc7c168709c12310c0ad47ad821e9 size: 1776
root@docker:~#
```

Push an Image to Docker Hub

Duration: 20 min.

Problem Statement:

Demonstrate how to push a Docker image to Docker Hub.

ASSISTED PRACTICE

Assisted Practice:Guidelines

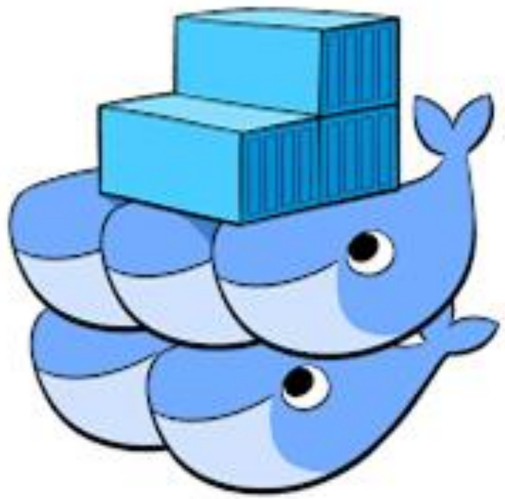
Steps to demonstrate push image:

1. Prepare a custom Docker image
2. Push the Docker image to Docker Hub
3. Push the code to GitHub repositories

Introduction to Docker Swarm

Docker Swarm

- A swarm is a group of machines that run Docker and join into a cluster.
- An IT administrator controls swarm through a swarm manager, which orchestrates and schedules containers.
- The swarm manager allows a user to create a primary manager instance and multiple replica instances in case the primary instance fails.
- A swarm is made up of multiple nodes, which can be either physical or virtual machines.



Docker swarm

Importance of Docker Swarm

- It allows to connect multiple hosts with Docker.
- Compared to Kubernetes, working with Docker is relatively easy.
- There are two node types in cluster, namely master and worker. One of the masters is the leader. If the current leader fails, another master will become the leader. If the worker host fails, all the containers will be rescheduled to other nodes.
- Swarms store configuration for containers. If the configuration is updated, containers are updated in batches. Therefore, service will be available all the time by default.
- If a port from a service is exposed, it will be available on any node in the cluster. This helps with external load balancing.

Container Deployment Using Docker Swarm

Duration: 20 min.

Problem Statement:

Demonstrate how to deploy a Docker container on Docker swarm.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to perform deployment:

1. Set up the Docker instance
2. Build a custom Docker image
3. Initialize the Docker swarm cluster and deploy a container to the cluster
4. Push the code to GitHub repositories

Container Scaling with Docker Swarm

Duration: 15 min.

Problem Statement:

Demonstrate how to scale a Docker container on Docker swarm.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to perform container scaling:

1. Create services for scaling
2. Push the code to GitHub repositories

Distribute Your App Across a Swarm Cluster

Duration: 15 min.

Problem Statement:

Demonstrate how to distribute your app across a swarm cluster.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to distribute the application across cluster:

1. Set up the Docker instance
2. Set up Docker swarm with multiple nodes
3. Deploy a custom Docker image to Docker swarm cluster
4. Push the code to GitHub repositories

Setting Up Jenkins Pipeline with Docker

Duration: 15 min.

Problem Statement:

Demonstrate how to set up Jenkins pipeline and connect it to docker cloud.

ASSISTED PRACTICE

Assisted Practice: Guidelines

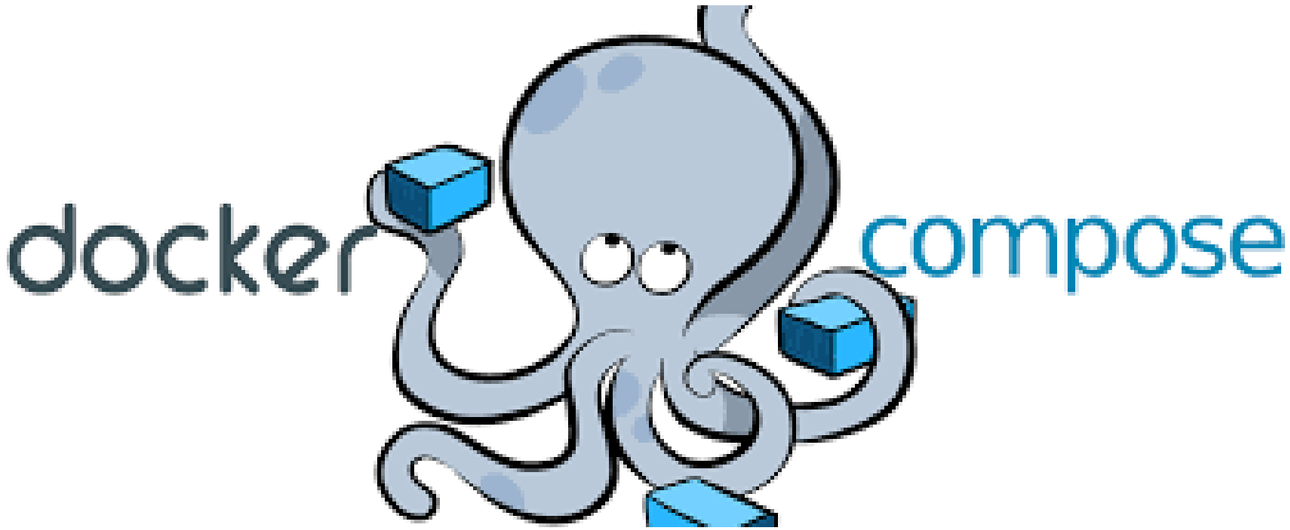
Steps to integrate Docker with Jenkins:

1. Install Docker plugin and configure Docker cloud
2. Configure Jenkins job
3. Push the code to GitHub repositories

Introduction to Docker Compose

Docker Compose

- Docker compose is a tool used to define and run multi-container Docker applications.
- The features of Docker compose, that make it effective, are:
  - Multiple isolated environments on a single host
  - Preserve volume data when containers are created
  - Only recreate containers that have changed
  - Variables and moving a composition between environments



Applications of Docker Compose

- **Development environments:**
  - The ability to run an application in an isolated environment and interact with it is crucial.
  - The Compose command-line tool can be used to create the environment and interact with it.

- **Automated testing environments:**
  - An important part of any continuous deployment or continuous integration process is the automated test suite.
  - Automated end-to-end testing requires an environment in which to run tests.
  - Compose provides a convenient way to create and destroy isolated testing environments for a test suite.

- **Single host deployments:**
  - Compose has traditionally been focused on development and testing workflows, but with each release progress is being made on more production-oriented features.
  - Docker Compose can be used to deploy to a remote Docker Engine.
  - The Docker Engine may be a single instance provisioned with Docker Machine or an entire Docker Swarm cluster.

Docker Compose

Duration: 20 min.

Problem Statement:

Demonstrate how to run an SSH Docker container on Docker compose.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate working of Docker Compose:

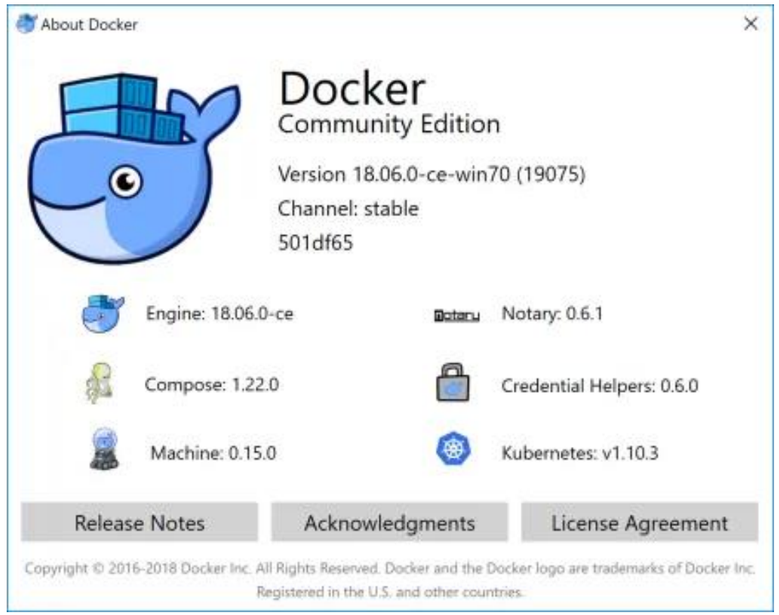
1. Performing Docker Compose operations
2. Push the code to GitHub repositories

Docker CE

Docker Community Edition

Docker Community Edition (CE) is ideal for developers and small teams looking to get started with Docker and experimenting with container-based apps. Docker CE has three types of update channels, which are:

- **Stable** gives you latest releases for general availability.
- **Test** gives prereleases that are ready for testing before general availability.
- **Nightly** gives you latest builds of work in progress for the next major release.



Docker CE

Duration: 20 min.

Problem Statement:

Demonstrate how to install Docker Community Edition.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to perform installation of Docker CE:

1. Install Docker Community Edition (CE) on your server
2. Add Docker's official GPG key to the server
3. Add a stable repository to download Docker CE package for installation
4. Install the latest version of Docker CE
5. Run sample Docker image to test Docker CE installation
6. Push the code to the GitHub repositories

Key Takeaways

- Docker is a tool designed to create, deploy, and run applications easily using containers.
- A container is a standard unit of software that packs up code and its dependencies.
- Docker Hub is a hosted repository service provided by Docker to find and share container images with your team.
- Swarm utilizes the same command line from Docker.

Set Up Jenkins Pipeline to Deploy Image to Docker Swarm

Duration: 40 min.

Problem Statement:

You are given a project to accomplish the following:

- Configure a Jenkins job to perform Docker image build
- Configure Swarm service on Swarm master for Docker image
- Deploy image to Docker Swarm

LESSON-END PROJECT

Before the Next Class

Course(s):

1. Learn ELK stack 6.0
2. Cloud basics and web hosting

You should be able to:

- Explain ELK stack
- Analyze data on Kibana dashboard
- Launch AWS EC2 instance and demonstrate AWS Cloudfront