Training: Micro Focus
DevOps340 - Docker Containerization

<table>
<thead>
<tr>
<th>FORM OF TRAINING</th>
<th>MATERIALS</th>
<th>PRICE</th>
<th>DURATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional</td>
<td>Digital materials</td>
<td>2160 USD</td>
<td>3 days</td>
</tr>
<tr>
<td>Traditional</td>
<td>CTAB Tablet</td>
<td>2290 USD</td>
<td>3 days</td>
</tr>
<tr>
<td>Distance learning</td>
<td>Digital materials</td>
<td>2160 USD</td>
<td>3 days</td>
</tr>
<tr>
<td>Distance learning</td>
<td>CTAB Tablet</td>
<td>2160 USD</td>
<td>3 days</td>
</tr>
</tbody>
</table>

LOCATIONS

Krakow - 5 Tatarska Street, II floor, hours: 9:00 am - 4:00 pm
Warsaw - 17 Bielska Street, hours: 9:00 am - 4:00 pm

TRAINING GOALS:

Learn the skills to radically simplify application deployment, IT workflow, version control, production support and much more with the power of Docker containerization. This Docker Containerization Training class is a continuous 3-day hands-on workshop. In recent years, few IT tools have been as disruptive or innovative as Docker. Lightweight and fast, the open-source Docker engine provides an environment to run your code as well as an efficient workflow to get code from developer machines to the test environment, and rapidly into production. Docker streamlines deployment of your applications into isolated containers, allowing versatile new solutions across the entire IT value chain.

Audience/Job Roles:

This Docker Containerization training workshop is ideal for developers and operations staff who want to containerize and improve reliability throughout the entire software development life cycle. To get the most out of the course, you should be familiar with some Linux basics, including package management, basic networking and simple shell scripting.

Course Objectives:

Learning to use Docker immediately gives you skills to:

- Rapidly deploy applications - By completely avoiding full machine and/or OS virtualization, containers vastly reduce application size and delivery time.
- Develop for nearly universal portability - By bundling all application dependencies into one container.
- Version control & reuse - Recursive reuse makes applications much more lightweight. Version control and rollback are easy.
- Enjoy extremely lightweight, minimal overhead - Docker images are small. Delivery and
deployment of new application containers are as fast as it gets.

- Vastly simplify maintenance - Much less overhead and fewer dependencies mean far less maintenance.

Learn how to use Docker to leverage containerization across your own application life cycle. This 3-day Docker Containerization training class is a continuous hands-on workshop which teaches you how to use Docker and the workloads for which it is best suited. From your clean new Docker install on the first day, you will work through continuous real-world use cases to learn pragmatic, immediately useful skills. Our real-world format gets you up and running as quickly as possible with the technology, and focuses on making sure you understand how to best integrate Docker into your workflow for maximum productivity as soon as you return to work.

Workshop includes a copy of the O'Reilly text Docker: Up & Running by Karl Matthias and Sean Kane.

This 3-day Docker Containerization Training workshop is conducted as a continuous hands-on lab. From initial install to multi-container application stack, you will learn Docker through real-world practice.

Audience/Job Roles

- DevOps

CONSPECT:

- Introduction
  - What can you use Docker for?
  - A logical segregation of duties
  - The relationship between Docker and SOA
  - How Docker fits into the development lifecycle
  - How Docker ensures consistency from development through UAT and staging, and on to production
  - Example use cases of Docker in the real world

- The components of Docker
  - Underlying technology
  - Docker client and server
  - Filesystem images
  - Registries
  - Containers
  - Networking

- Getting set up to start using Docker
  - Getting set up on Windows
  - Getting set up on the Mac
- Trying out our first container
- Getting set up for production on Linux
- Tweaking your production environment for best performance
- User interfaces for Docker management

- Container management
  - Container naming
  - Starting and stopping containers
  - Attaching to a container
  - Seeing what is happening in a container
  - Running a process inside a container
  - Daemonizing a container
  - Automatic container restarts
  - Deleting containers when we are finished with them

- Docker images and repositories
  - Docker images explained
  - How Docker images work
  - Getting a list of images
  - Searching for images on a repository
  - Pulling an image
  - Creating our own image
  - Specify an image in a Dockerfile
  - Building Dockerfile images
  - Using the build cache for templating
  - Viewing the image we have created
  - Launching a container using our new image

- Registries
  - What is the Docker hub?
  - Pushing images to the Docker hub
  - Running your own internal Docker registry
  - Testing the internal registry

- A simple use case
  - A single container static website
  - Setting up a container running Nginx
  - Launching our static site
  - Updating our static site from git or bitbucket

- Continuous integration with Docker
- How Docker enables and supports CI
- Getting set up for Jenkins and Docker
- A basic Jenkins job
- Multi configuration jobs
- Drone
- Shippable

- A more complex use case: Multi container application stacks
  - A container for our NodeJS application
  - A base image for our Redis containers
  - Creating our Redis back-end cluster
  - Capturing logs
  - Managing containers

- Docker orchestration and service discovery
  - Getting set up with Fig
  - Configuring the fig.yml file
  - How to use Fig
  - Console
  - Running a Console cluster

- Integrating with configuration management
  - Managing your Docker hosts with Chef / Puppet / Ansible
  - Building containers using configuration management tools
  - Managing running containers with configuration management

- Docker and DevOps
  - Enabling collaboration with Docker
  - Using Docker to streamline workflow
  - Using Docker's version control capabilities to enable experimentation and learning
  - Docker's role in the overall IT value chain
  - Creating value and quality with Docker
  - Enabling smoother flow of work

- Course conclusion, open discussion and Q&A
  - Going back to work with a plan
  - What was not covered in this class
  - Q&A with the instructor
  - Goodbyes
REQUIREMENTS:

To get the most out of the course, you should be familiar with some Linux basics, including package management, basic networking and simple shell scripting.

Difficulty level

CERTIFICATE:

The participants will obtain certificates signed by Micro Focus (course completion).

TRAINER:

Authorized Micro Focus Trainer