A Ilica 242, HR-10000 Zagreb
T (01) 2222 182
E student@algebra.hr
www.algebra.hr

ALGEBRA

DEPARTMENT OF OPERATING SYSTEMS

# Intro to DevOps

## Lab 03 – Building container images

REV 3.3

A Ilica 242, HR-10000 Zagreb
T (01) 2222 182
E student@algebra.hr
www.algebra.hr

ALGEBRA

# Contents

A Ilica 242, HR-10000 Zagreb
T (01) 2222 182
E student@algebra.hr
www.algebra.hr

ALGEBRA

# Introduction

In this lab we will learn how to build container images and different Dockerfile instructions. Also we will learn how to configure the container to automatically start with the VM and automatically pull the new image when it is available.

A Dockerfile is a text file that contains a set of instructions used to build a Docker image. These instructions define the environment and configuration needed for running a specific application or service within a container. Dockerfiles follow a straightforward syntax and allow developers to automate the creation of container images, ensuring consistency across different environments.

As you continue working with containers, there will be a need to automate the start of containers with the host operating system. When using docker this is achieved with restart policies, and when using podman this is achieved with systemd services or quadlets.

# Tasks

1. Based on the source code available at https://github.com/docker/getting-started-app containerise the application. There is a guide available at https://docs.docker.com/get-started/02_our_app/. For solving of this task use podman instead of docker.
2. Build this image again, but this time tag it with 0.0.1.
3. Inspect the image contents by using **podman image inspect**.
4. Review all available Dockerfile instructions at https://docs.docker.com/reference/dockerfile/.
5. Change the application so a different message appears when you have no to do items to complete. There is a guide available at https://docs.docker.com/get-started/03_updating_app/.
6. Build this image again, but this time tag it with 0.0.2.
7. See the difference between these two images by using **podman diff** command.
8. Create another repository on Docker Hub and push the images there. There is a guide available at https://docs.docker.com/get-started/04_sharing_app/.
9. Create another Containerfile which will use rhel7 image as a base image. Update all installed packages and install httpd. Configure the image in a way that httpd is started as container is created. Name the image myweb and tag it 0.0.1. Note that the image is in a private repository to which you can login with **podman login**.
10. Create a quadlet file to configure the system so a container named web from image myweb:0.0.1 is started with the system. There is a guide available at https://www.redhat.com/sysadmin/quadlet-podman.
11. Clone git repository available at https://github.com/jstanesic/example-go-app. Build a container image by using Dockerfile Dockerfile1 and tag it example:Dockerfile1. Build another image by using Dockerfile Dockerfile2 and tag it example:Dockerfile2. Check sizes of the images. Discover what benefits multi-stage builds has in use cases when application needs to be compiled.

A Ilica 242, HR-10000 Zagreb
T (01) 2222 182
E student@algebra.hr
www.algebra.hr

ALGEBRA

## Additional reading

- https://docs.docker.com/reference/dockerfile/
- https://linuxhandbook.com/autostart-podman-containers/
- https://systemd.io/
- https://catalog.redhat.com/software/containers/rhel7/57ea8cee9c624c035f96f3af?architecture=amd64&image=65e70f46befb5000328d4d90
- https://github.com/nigelpoulton/psweb
- https://developers.redhat.com/articles/2023/05/17/build-lean-nodejs-container-images-ubi-and-podman
- https://earthly.dev/blog/docker-multistage/
- https://docs.docker.com/build/building/multi-stage/
- https://docs.bitnami.com/tutorials/optimize-docker-images-multistage-builds/
- https://earthly.dev/blog/docker-multistage/