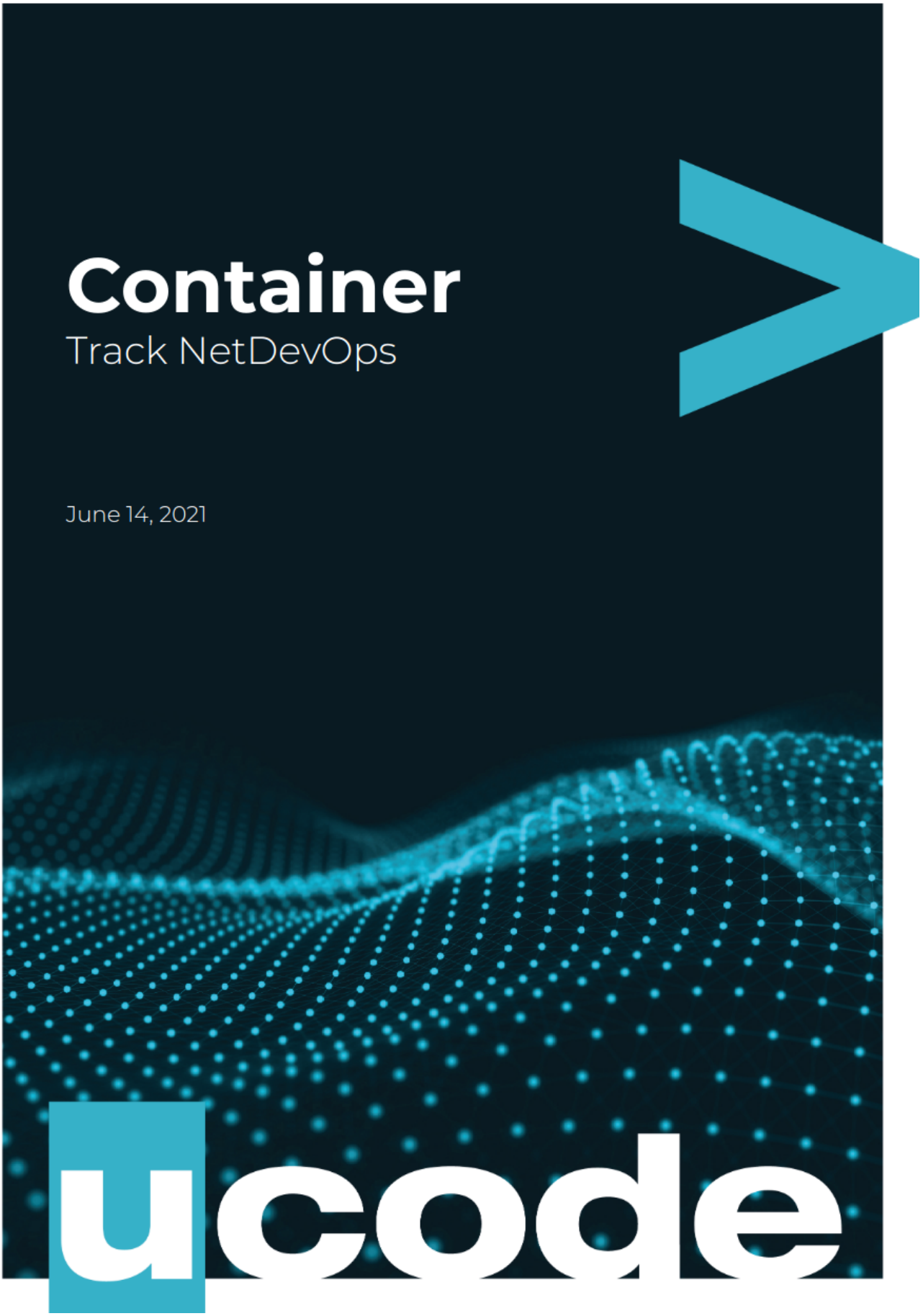
Container
Track NetDevOps
June 14, 2021
ucode

Contents

ucode

Engage

DESCRIPTION

Welcome to the partnership NetDevOps Track designed by Cisco and ucode IT academy!

As we see, today, programs have become much bigger than ever before.
And it is not always very convenient to pass complicated systems between devices.
Have you ever thought about why nearly all ucode materials contain information about software versions you need to use?

For example, you can write `print "box:", box` in Python2, but it won`t work in Python3.
Yes, it is an oversimplified example, but it shows that you always need to take care of the working environment. Because it won't be so easy with 10 dependencies, and error messages will definitely infuriate you.

Until 2013, there was no standard solution, which would help to transfer systems with their dependencies across devices. But then Docker appeared with the idea that we can put everything we want into containers, and transfer them as easily as we transfer goods on ships or planes. Docker can package an application and its dependencies in a virtual container that can run on any Linux, Windows, or macOS computer.

This challenge includes tasks for understanding how to deal with this cool technology.
Figure out why no modern DevOps engineer could live without it.

BIG IDEA

Containerization.

ESSENTIAL QUESTIONS

How to put your program into a container?

CHALLENGE

Create your first containers and learn how to work with them.

Investigate

GUIDING QUESTIONS

We invite you to find answers to the following questions. By researching and answering them, you will gain the knowledge necessary to complete the challenge. To find answers, ask the students around you and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- What is `Docker`?
- What are the methods of code deployment? What is their history?
- How do containers solve deployment problems?
- How to use Docker tools for containers?
- How to run a Docker container?
- What command can be used to audit the differences between your container and its base image?
- What is `Istio`?
- What is load balancing?
- What is HAProxy load balancer?

GUIDING ACTIVITIES

Complete the following activities. Don't forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

- Investigate how to build Docker container images and try to do it.
- Try to build and version your own Docker images.
- Build container images automatically, based on a Dockerfile template.
- Find how to create an IOx Application with Docker.
- Sign up for a free account at a public image registry and find something interesting there.
- Learn how to search for images in the registry.
- Read about automating OS migration pre-checks and post-checks with pyATS: getting a CLI output from a device.
- Find information on the importance of the service mesh and how it enables enhanced traffic control between microservices using Istio.
- Clone your git repository, issued on the challenge page in the LMS.
- Start developing your solution.

ANALYSIS

Analyze your findings. What conclusions have you made after completing guiding questions and activities? In addition to your thoughts and conclusions, here are some more analysis results.

- Be attentive to all statements of the story.

- Analyze all information you have collected during the preparation stages. Try to define the order of your actions.
- You can proceed to `Act: Creative` only after you have completed all requirements in `Act: Basic`. But before you begin to complete the challenge, pay attention to the program's architecture. Take into account the fact that many features indicated in the `Act: Creative` require special architecture. And in order not to rewrite all the code somewhen, we recommend you initially determine what exactly you will do in the future. Note that the `Act: Basic` part gives the minimum points to validate the challenge.
- Perform only those tasks that are given in the story.
- You should submit only the specified files in the required directory and nothing else. In case you are allowed to submit any files to complete the task you should submit only useful files. Garbage shall not pass.
- To shape an application, developers should follow the Twelve-Factor App guidelines. These important points are considered the gold standard for designing and delivering cloud-native applications.
- The solution will be checked and graded by students like you.
  Peer-to-Peer learning.
- Your work may also be graded by your mentor. So, be ready for that.
- If you have any questions or don't understand something, ask other students or just Google it.

Act: Task 00

NAME

Testapp

DIRECTORY

`t00/`

SUBMIT

`00.sh, 01.sh, 02.txt, 03.sh, 04.sh`

DESCRIPTION

Create an Ubuntu container with the name `testapp` and run BASH inside it. Store one line command in file with name `00.sh`
Inside the Ubuntu container, create a small application that just echoes `'It works!'`.
Store commands in file with name `01.sh`.

Run the command that helps to audit the difference between your container and its base image. Store the output in a `02.txt` file. Store the command you used in a `03.sh` file.

Save the container (`testapp`) into a local image with a new name - `myapp`. Write the corresponding command in a `04.sh` file.

Act: Task 01

NAME

Dockerfile

DIRECTORY

`t01/`

SUBMIT

`00.sh, Dockerfile`

DESCRIPTION

Delete the running container and confirm that it is not running anymore. Store the commands you used to delete the container and to check if deletion was successful in a `00.sh` file, each command in a new line.

Create a `Dockerfile` for a Ubuntu container. Use this Dockerfile to build an image.

Act: Task 02

NAME

Horse

DIRECTORY

`t02/`

SUBMIT

`Dockerfile`

DESCRIPTION

Create a Dockerfile that can show the following output:

Act: Task 03

NAME

Redis

DIRECTORY

`t03/`

SUBMIT

`redisSearch.sh, redis.sh, pullRedis.sh`

DESCRIPTION

Log in to Docker Hub.

Find the command that you need to run to search registry images by names.
Search for `redis` in the registry.

Store the search results in a `redisSearch.sh` file. Store the command you used in a `redis.sh` file.

Pull the `redis` image. Store the command you used in a `pullRedis.sh` file.

Act: Task 04

NAME

Server

DIRECTORY

`t04/`

SUBMIT

`server.py, SimpleHTTPServer.sh, inspect.sh`

DESCRIPTION

Create a python web server in a `server.py` file.
Your server must run on a `8000` port.

Build and run the new web server container image.
Store the commands you used in a `SimpleHTTPServer.sh` file.

Inspect Docker container using `curl` to check the http server. Store the commands you used in a `inspect.sh` file.

Act: Task 05

NAME

Explore images

DIRECTORY

`t05/`

SUBMIT

`dive.sh`

DESCRIPTION

Explore images using Dive tools.
Shrink the size of this Dockerfile.
Store the commands you used in a `dive.sh` file.

SEE ALSO

Best practices for writing Dockerfiles

Act: Task 06

NAME

Connection between containers

DIRECTORY

`t06/`

SUBMIT

`docker-compose.yml, checkCommands.sh`

DESCRIPTION

Choose two simple Dockerhub web images and start the two defined containers in detached mode. `docker-compose.yml` must contain rules for starting these containers.
Run two containers, and verify connectivity between them using commands:

```
docker exec -it someclient /bin/bash
ping myapp -c 2
exit
```

Find out at least two more commands for verifying connectivity between containers and store them in a `checkCommands.sh` file.

Act: Task 07

NAME

Load balance

DIRECTORY

`t07/`

SUBMIT

`docker-compose.yml, loadBalance.sh`

BEFORE YOU BEGIN

Get acquainted with HAProxy load balancer and container networking.

DESCRIPTION

Start two containers for `myapp` and one for `HAProxy` on different ports, both in detached mode.
Map your app port to `HAProxy` port.
Store docker content that you used in a `docker-compose.yml` file.
Store other docker CLI command in `loadBalance.sh`.

Act: Task 08

NAME

Docker blog

DIRECTORY

`t08/`

SUBMIT

`docker-compose.yaml`

DESCRIPTION

Create `docker-compose.yaml` that can set up your blog using Ghost (open source blogging platform written in JavaScript and distributed under the MIT License) + Nginx + MySQL.

After running your blog, confirm its working capacity by adding some content to it, for example publishing posts.

SEE ALSO

Ghost
Ghost on docker

Document

DOCUMENTATION

One of the attributes of Challenge Based Learning is documentation of the learning experience from challenge to solution. Throughout the challenge, you document your work using text and images, and reflect on the process. These artifacts are useful for ongoing reflection, informative assessment, evidence of learning, portfolios, and telling the story of challenge. The end of each phase (Engage, Investigate, Act) of the challenge offers an opportunity to document the process.

Much of the deepest learning takes place by considering the process, thinking about one's own learning, analyzing ongoing relationships with the content and between concepts, interacting with other people, and developing a solution. During learning, documentation of all processes will help you analyze your work, approaches, thoughts, implementation options, code, etc. In the future, this will help you understand your mistakes, improve your work, and read the code.

At the learning stage, it is important to understand and do this, as this is one of the skills that you will need in your future job. Naturally, the documentation should not be voluminous, it should be drawn up in an accessible, logical, and connected form.

So, what must be done?

- a nice-looking and helpful README file. In order for people to want to use your product, their first introduction must be through the README on the project's git page. Your README file must contain:
  - Short description. This means, that there must be some info about what your project actually is. For example, what your program does.
  - Screenshots of your solution. This point is about screenshots of your project "in use".
  - Requirements and dependencies. List of any stuff that must be installed on any machine to build your project.
  - How to run your solution. Describe the steps from cloning your repository to the first launch of your program.
- a full-fledged documentation in any forms convenient for you. By writing this, you will get some benefits:
  - you have an opportunity to think through implementation without the overhead of changing code every time you change your mind about how something should be organized. You will have very good documentation available for you to know what you need to implement
  - if you work with a development team and they have access to this information before you complete the project, they can confidently start working on another part of projects that will interact with your code
  - everyone can find how your project works
- your documentation must contain:
  - Description of progress after every competed CBL stage.
  - Description of the algotithm of your whole program.

Keep in mind that the implementation of this stage will be checked by peers at the assessment!

Also, there are several links that can help you:

- Make a README
- How to write a readme.md file?
- A Beginners Guide to writing a README
- Google Tools - a good way to journal your phases and processes:
  - Google Docs
  - Google Sheets
- Dropbox Paper - a tool for internally developing and creating documentation
- Git Wiki - a section for hosting documentation on Git-repository
- Haroopad - a markdown enabled document processor for creating web-friendly documents
- Canva - a good way to visualize your data
- QuickTime - an easy way to capture your screen, record video or audio (macOS)
- ScreenToGif - screen, webcam, and sketchboard recorder with an integrated editor (Windows)
- code commenting - source code clarification method. The syntax of comments is determined by the programming language
- and others to your taste

Share

PUBLISHING

Last but not least, the final stage of your work is to publish it. This allows you to share your challenges, solutions, and reflections with local and global audiences. During this stage, you will discover ways of getting external evaluation and feedback on your work. As a result, you will get the most out of the challenge, and get a better understanding of both your achievements and missteps.

To share your work, you can create:

- a text post, as a summary of your reflection
- charts, infographics or other ways to visualize your information
- a video, either of your work, or a reflection video
- an audio podcast. Record a story about your experience
- a photo report with a small post

Helpful tools:

- Canva - a good way to visualize your data
- QuickTime - an easy way to capture your screen, record video or audio (macOS)
- ScreenToGif - screen, webcam, and sketchboard recorder with an integrated editor (Windows)

Examples of ways to share your experience:

- Facebook - create and share a post that will inspire your friends
- YouTube - upload an exciting video
- GitHub - share and describe your solution
- Telegraph - create a post that you can easily share on Telegram
- Instagram - share photos and stories from ucode. Don't forget to tag us :)

Share what you've learned and accomplished with your local community and the world. Use #ucode and #CBLWorld on social media.