Introduction to Jenkins, CI/CD, and DevOps 2024 (course notes)

Course notes

Course created by: Valentin Despa

Version: July 2024

Introduction

1. Course overview

Thank you for taking this course. 🙏

This course covers:

- Jenkins installation
- Introduction to DevOps
- Building a website using Node.js, testing and deploying it.
- Explanation of CI/CD
- Using Docker containers with Jenkins
- Installing CLI tools
- Basic Linux commands

Your notes

2. What is Jenkins?

- Jenkins is a free and open-source automation server mainly used to automatically build and test software.
- Jenkins is a Continuous Integration (CI) and Continuous Deployment (CD) tool.

Your notes

3. Jenkins installation

- **IMPORTANT!**
  I recommend installing Jenkins using Docker as shown in the course so that you can follow along.
- Download and install Docker Desktop
- Download the configuration needed to start Jenkins from the GitHub repository
- Follow the instructions from the GitHub repository
- Configure Jenkins
- Write down your Jenkins username & password

📁Resources

- **Download Docker Desktop**
  https://www.docker.com/products/docker-desktop/

- **Install Jenkins using Docker Compose**
  https://github.com/vdespa/install-jenkins-docker

Your notes

4. Your first Jenkins job

- A job is a set of commands we want Jenkins to execute
- To create a job click on “New item”
- *echo* is a command that outputs the text
- *whoami* is a command that outputs the current system username
- A pipeline is a set of stages, each stage running one or more commands
- In this course, we will use Pipeline jobs
-

Your notes

5. What is DevOps?

- DevOps is not a standard or a specification.
- DevOps is not a tool or particular software.
- DevOps represents a cultural change and a shift in mindset.
- Traditional development involves separate roles with little collaboration.
- Lack of collaboration leads to blame and finger-pointing when issues arise.
- Developers focus on building software, while IT operations ensure smooth infrastructure.
- DevOps addresses the lack of mutual understanding between development and operations.
- DevOps involves collaboration among all stakeholders in the software lifecycle.
- DevOps is connected to the agile movement, promoting experimentation and learning.
- Collaboration replaces working in silos and finger-pointing and everyone takes responsibility for the final product in a DevOps culture.
- Automation of tasks is crucial in DevOps to improve productivity.
- DevOps aligns with Agile frameworks like Scrum.
- "The Phoenix Project" audiobook is recommended for insights into DevOps adoption.
- DevOps practices require a cultural shift and automation tools.
- Using DevOps tools alone does not constitute practicing DevOps.

Your notes

Continuous Integration (CI) with Jenkins

6. Introduction to CI

- We will start working on a simple project to automate manual steps in integrating changes from multiple developers.
- The goal is to create a pipeline that will build and test the software.
- This process is known as Continuous Integration (CI), a fundamental DevOps practice.
- CI allows multiple developers to add and integrate changes multiple times per day, resulting in a new software version.
- Every time we change the code, it is tested and integrated with others' work.
- CI means work is integrated continuously as changes happen.
- Delaying integration increases the chances of encountering issues.
- Version control systems like Git enable CI by tracking all changes.
- For those new to Git, resources for a quick introduction are available below
- We will use Git and Jenkins to verify and integrate changes into the project.

📁Resources

- **Git course for beginners** (use password LEARN_GIT to access the course) https://www.udemy.com/course/learn-git-for-github-projects/?couponCode=EB6D9313C67E3F226620

Your notes

7. Creating a GitHub account

- to follow the pipeline as code principle, we need to store the pipeline code in a code versioning system like Git
- To work with Git and to use a Git repository, we will use GitHub.
- We will use GitHub to store our project code repository as well as the Jenkins pipeline file.

📁Resources

- **Create a GitHub account**
  https://github.com/signup

Your notes

8. Project overview

- Fork the “Learn Jenkins App” repository
- Open an IDE in your browser by creating a codespace
- the command *npm install* will download all project dependencies based on the *package.json* file
- The command *npm start* will start the application locally

📁Resources

- **Learn Jenkins App on GitHub** (fork this project)
  https://github.com/vdespa/learn-jenkins-app

Your notes

9. Using Docker as a build environment

IMPORTANT!

I recommend installing Jenkins using Docker as shown in Lecture 3 so that you can follow along with this and the upcoming lectures.

- By default, Jenkins does not contain the tools we need to build or run the projects
- There are two approaches:
    - Installing tools directly on the Jenkins agent
    - Using Docker
- The modern approach is to use Docker
- Install Docker Pipeline plugin
- Check if Node.js and npm is installed with the following commands:

  node --version

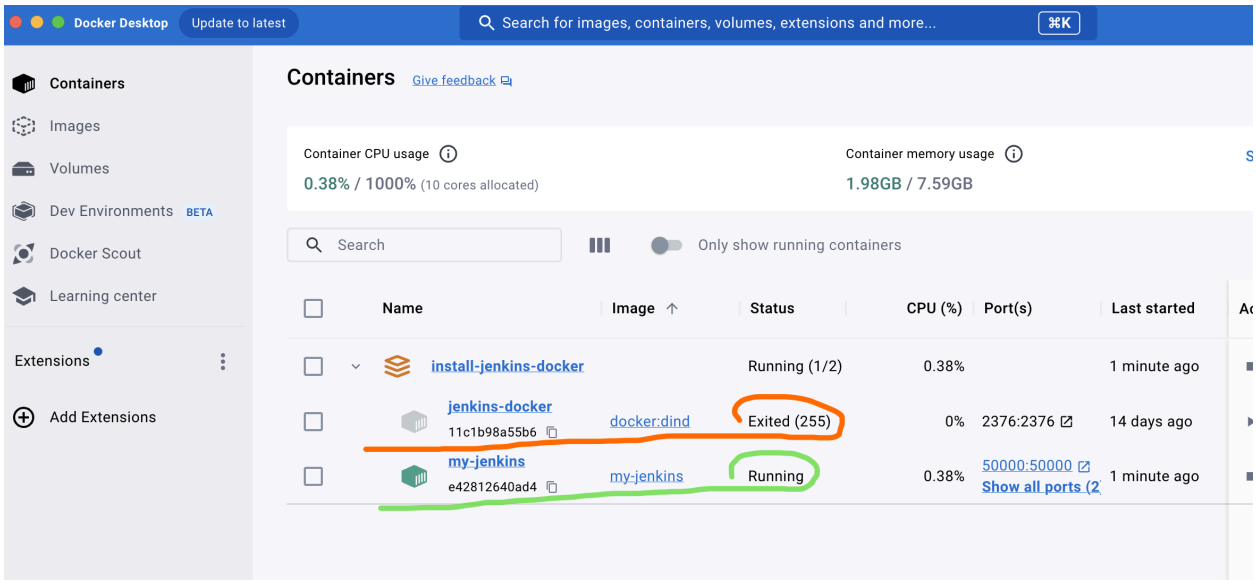
  npm --version

Your notes

Troubleshooting

> **IMPORTANT: Read the following text regardless of whether you are experiencing an issue right now or not.**

If you are having difficulties running the pipeline when you want to use a Docker container, open the Docker Desktop app and inspect the containers currently running.

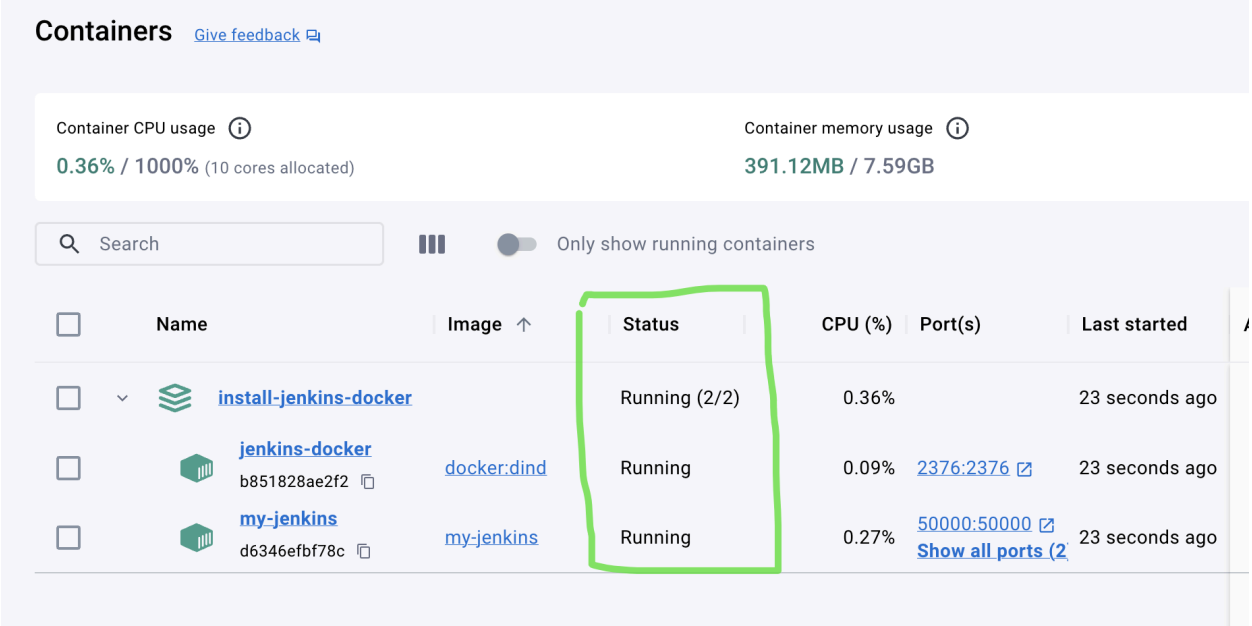


In the screenshot above, you can notice that one container is running (status *Running*) while the other has the status *Exited (255)*. In this case, Jenkins will not be able to run a stage in a Docker container.

To fix the issue, run the following commands while the terminal window is in the `install-jenkins-docker` folder.

```
docker compose down
docker compose up -d
```

After this, Docker Desktop should look as follows:



10. Workspace synchronization

- To reuse the Jenkins workspace files between stages, we need to configure the Docker agent with the flag *reuseNode***.**

```
agent {
    docker {
        image node:18-alpine
        reuseNode true
    }
}
```

Your notes

11. Using a Git repository

- In the Advanced Project Options, select “Pipeline script from SCM”
- Configure the repository URL
- Change the branch specifier from *master* to *main*.

Your notes

12. Building the project

- every software project has a build step
- For this project, we need to run the command *npm run build*
- Project dependencies are NOT stored in Git.
- The build output is NOT stored in Git.
- Inside Jenkins, the recommended command to install project dependencies is *npm ci*
- *ls* command lists all the files in the current directory
- *npm WARN* entries in the logs are okay, they are NOT errors.

Your notes

13. Running tests

- the *test -f <PATH>* command verifies if a file exists
- To run the tests, we use the command *npm test*
- Running the tests does not need a build artifact, so the tests can be executed at any stage in the pipeline

Your notes

14. JUnit test report

- A JUnit report is generated by the JUnit testing framework in Java projects in XML format.
- The report provides detailed information on test results, including pass/fail status, execution times, error messages, and summary statistics.
- JUnit reports are crucial for CI/CD workflows.
- Our testing framework can generate a report in the JUnit format even if our project does not use Java or the JUnit framework.
- The JUnit report will be stored in the project reports directory as *junit.xml*.
- We want to publish the JUnit report regardless of execution success or failure.

Your notes

Continuous Deployment (CD) with Jenkins

15. Introduction to CD

In this section:

- we will deploy a website project to the cloud using Netlify
- explore DevOps practices such as Continuous Deployment
- create a CI/CD pipeline to build, test, and deploy the website to the cloud.

- Sign-up for a free Netlify account, which takes only two minutes.
- Use a GitHub account or email and password to sign-up, and verify your email.
- Skip deploying the first project initially and proceed to the next page.

📁Resources

- **Sign-up for a Netlify account**
  https://app.netlify.com/signup

Your notes

16. Manual deployment

- Before we automate, we need to understand the manual steps so we will begin with a manual deployment.
- Unzip the provided asset to get a build directory containing one index.html file.
- Manually upload the file using the netlify.com UI.
- Open the URL provided by Netlify.
- Wait a few seconds for the deployment to complete.
- Use the provided URL to open the website in your browser.
- You should see the Test message displayed.
- In the next lecture, the same steps will be done using Jenkins.

Your notes

17. Installing CLI tools

- Jenkins is not ideal for automating manual steps through a UI
- Jenkins works best with CLI tools
- To automatically deploy our project to Netlify, we need a CLI tool.
- Install using: *npm install netlify-cli -g*
- Check the version using: *netlify --version*

📁Resources

- https://docs.netlify.com/cli/get-started/

Your notes

18. Environment variables

- environment variables enable dynamic configuration of build parameters and settings. This makes it easy to adapt the Jenkinsfile to different environments without changing the code
- environment variables help simplify the Jenkinsfile, making it more modular and easier to maintain.
- configurations can be defined once as environment variables and reused across multiple stages, reducing redundancy and potential errors.

Your notes

19. Managing secrets

- go to your netlify.com profile and generate a new Personal Access Token.
- securely store this token in Jenkins.

Your notes

20. Using credentials in the pipeline

- Hardcoding secrets in a Jenkinsfile is bad practice.
- Hardcoding secrets exposes them in plaintext, visible to anyone with access to the file.
- Accidental commits to public repositories can expose secrets, creating security vulnerabilities.
- Using Jenkins' credentials feature is recommended for security, maintenance, and compliance.
- Jenkins credentials are stored encrypted, reducing exposure risk.
- Access to Jenkins credentials can be controlled and audited.

Your notes

21. Deploying to production

- The Netlify CLI deploy command requires specifying the directory to deploy.
- After the Build stage, the website is in the build directory.
- Use the `--prod` flag to trigger a production deployment.
- Add the following step to the Deploy to prod stage:

sh 'netlify deploy --dir=build --prod'

- Execute the pipeline and inspect the logs.
- Manually verify the deployment by opening the website.

📁Resources

- https://docs.netlify.com/site-deploys/create-deploys/#netlify-cli

Your notes

22. Complete Jenkins course

Jenkins: Jobs, Pipelines, CI/CD, and DevOps for Beginners

What you will learn:

- ✅ Master Jenkins for CI/CD workflows
- ✅ Linux commands used in CI/CD
- ✅ DevOps & CI/CD fundamentals
- ✅ Automate builds and tests
- ✅ AWS deployments (S3, ECS)
- ✅Build Docker containers
- ✅Optimize pipeline speed
- ✅Troubleshooting

Learn more about the course

Get in touch!

- Feel free to reach out anytime you have questions. I am still there to help you, even after completing the course.

    - Connect on LinkedIn (please introduce yourself in the note): https://www.linkedin.com/in/vdespa/

    - Subscribe on YouTube: http://www.youtube.com/channel/UCUUl_HXJjU--iYjUkIgEcTw?sub_confirmation=1

    - Follow me on X: https://x.com/vdespa

Take care and bye-bye!