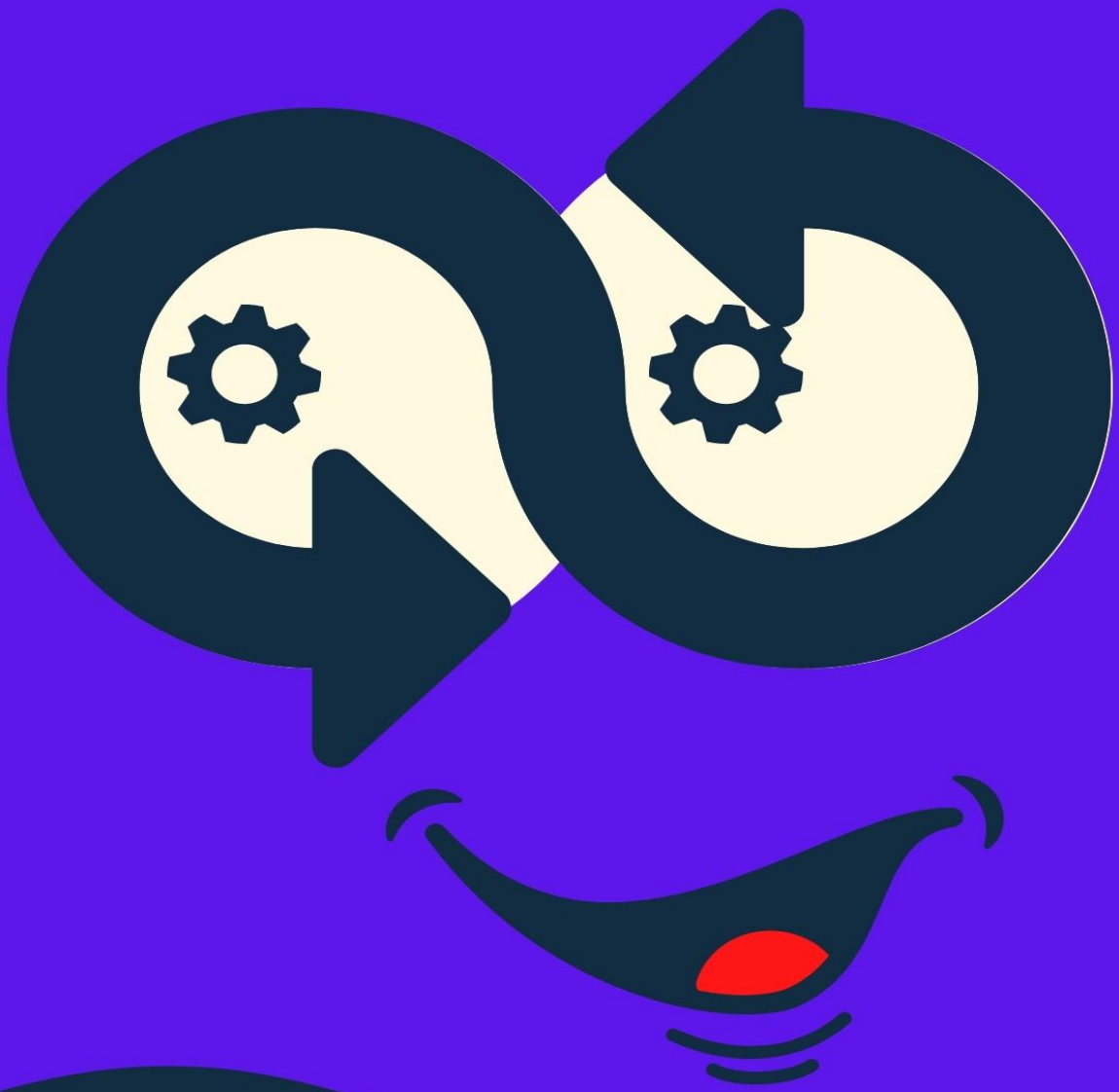
BY DENNIS (DENCTL)



THE DEVOPS HANDBOOK

Table of Contents

Book Layout

About The Book

The DevOps cookbook is a free open-source book that aims to provide general knowledge about DevOps, and a collection of recipes for common DevOps practices. The book is written in Markdown and is published using Ibis.

Book Sections

Chapters are organized in a way that you can read them in order, or you can jump to the chapter you need. The book is divided into 4 sections.

Section 1: Theory

Section 1.1: Must-Have Theory

- **I am a SysAdmin!**: This section is about the general knowledge a DevOps engineer should have from system administration perspective. It covers topics like System Administration basics, the Linux command line, and the basics of networking.
- **I am a Developer!**: This section is about the general knowledge a DevOps engineer should have from developement perspective. It covers topics like the basics of programming, the basics of databases, and the basics of web development.
- **I am *almost* a DevOps!**: This section is a checkpoint. Take your time and grab a cup of tea, you are now ready to become a DevOps engineer.

Section 1.2: DevOps Basics

- **DevOps 101**: This section covers topics like what is DevOps, the DevOps culture, and the DevOps tools.
- **DevOps 102**: This section covers topics like the DevOps lifecycle, the DevOps mindset, and the DevOps principles.

Section 1.3: DevOps Advanced

- **DevOps 201**: This section covers topics like continuous integration, continuous delivery, and continuous deployment.
- **DevOps 202**: This section covers topics like infrastructure as code, and configuration

management.
- **DevOps 203**: This section covers topics like monitoring, and logging.
- **DevOps 204**: This section covers topics like security, and disaster recovery.
- **DevOps 205**: This section covers topics like containers, orchestration, and microservices.
- **DevOps 206**: This section covers topics like serverless, and cloud computing.
- **DevOps 207**: This section covers topics like DevOps in the enterprise.

Section 2: Practice

Section 2.1: Cloud

- **Cloud 1: Create a DigitalOcean Droplet and configure LEMP stack on it**
- **Cloud 2: Deploy Laravel on a DigitalOcean droplet**
- **Cloud 3: Configure a DigitalOcean Managed Database for your Laravel App**
- **Cloud 4: Configure a DigitalOcean space for your Laravel static assets**
- **Cloud 5: Deploy a Laravel App to a DigitalOcean Kubernetes Cluster**

Section 2.2: Linux

- **Linux 1: Automate Laravel Deployment With Bash**
- **Linux 2: Load Balance Laravel With HAProxy**
- **Linux 3: Migrate a Laravel App from one server to another**
- **Linux 4: Write a Bash Script to Monitor System Resources (MEM, CPU, STORAGE) and send a Discord Notification**

Section 2.3: Databases

- **Database 1: Replicate a MySQL Database to a Slave Server**

Section 2.4: Containers

- **Containers 1: Deploy a Laravel App to a Docker Container**
- **Containers 2: Deploy a Laravel App with Docker Compose**
- **Containers 3: Deploy a Laravel App to a Kubernetes Cluster**

Section 2.5: Infrastructure as Code

- **IaC 1: Deploy a DigitalOcean Droplet, Kubernetes Cluster, and a managed Database with Terraform**
- **IaC 2: Deploy a Laravel Application on a DigitalOcean Droplet with a managed Database with Terraform and Ansible**

How to Contribute

The book is open-source and you can contribute to it by sending a pull request to the GitHub repository

Section 1: Theory

I am a SysAdmin

Since you want to become a DevOps engineer, you should have a good understanding of system administration. This section is about the general knowledge a DevOps engineer should have from system administration perspective. It covers topics like System Administration basics, the Linux command line, and the basics of networking.

What is System Administration?

System administration is the process of managing computer systems and networks. It includes maintaining the hardware, software, and configurations of computer systems. Most companies distribute the work of system administration among a multiple teams of administrators. These teams usually include an SRE(site realiability engineer), a network administrator, a database administrator, a storage administrator, and a security administrator.

System administrators are responsible for the security, availability, and performance of computer systems. They also make sure that the systems are up-to-date and running smoothly. System administrators are also responsible for:

- Installing and configuring new hardware and software.
- Monitoring the performance of the systems.
- Troubleshooting problems.
- Performing backups and disaster recovery.
- Managing user accounts and permissions.
- Managing the network.
- MANAGING EVERYTHING!

What knowledge do you need from the System Administration world?

You should have a good understanding of the following topics:

- Linux command line.
- Networking.
- Cloud.
- Databases.
- Security.
- Monitoring.
- Backups and disaster recovery.

Let's take a look at each of these topics.

Linux Command Line

What is the Linux command line?

The Linux command line is a text interface for your computer. It's a program that takes in commands, which it passes on to the computer's operating system to run. The command line is also known as the terminal, console, or shell.

The Linux command line is a powerful tool that allows you to do things that are difficult or impossible to do with a graphical user interface. It's also a great way to learn about the inner workings of your computer. It is a must-have skill for any DevOps and System engineer.

How to learn the Linux command line?

There are many resources available online to learn the Linux command line. Here are some of the best resources:

- Linux Command Line Basics

Networking

What is Networking?

Networking is the practice of connecting two or more computing devices together for the purpose of sharing data. In a data network, computing devices exchange data with each other using connections (data links) between nodes. These data links are established over cable media such as wires or optic cables, or wireless media such as WiFi.

How to learn Networking?

You should have a good understanding of the following topics:

- OSI Model.
- IP addresses.
- Subnets.
- Ports.
- Protocols.
- DNS.
- Load balancing.
- Firewalls.
- VPNs.

There are many resources available online to learn Networking. Here are some of the best resources:

- OSI Model
- IP addresses
- Subnets
- Ports
- Protocols
- DNS
- Load balancing
- Firewalls

Cloud

What is Cloud?

Cloud computing is the on-demand delivery of IT resources and applications via the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider like DigitalOcean, Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform.

How to learn Cloud?

You should have a good understanding of the following topics (actually there are a lot more but let's start with these absolutely mandatory ones before we jump into DevOps):

- Cloud computing.
- Virtualization.

There are many resources available online to learn Cloud. Here are some of the best resources:

- Cloud computing
- Virtualization

Databases

What is a Database?

A database is an organized collection of data. It is the place where you store all your data. Databases are used to store and retrieve related information. They are used to manage large amounts of data efficiently.

How to learn Databases?

You should have a good understanding of the following topics:

- Relational databases.
- Non-relational databases.
- SQL.
- NoSQL.

There are many resources available online to learn Databases. Here are some of the best resources:

- Relational databases
- Non-relational databases
- SQL
- NoSQL

Security

What is Security?

Security is the process of protecting information by preventing unauthorized access, use, disclosure, disruption, modification, inspection, recording, or destruction. It is a must-have skill for any DevOps and System engineer.

How to learn Security?

You should have a good understanding of the following topics:

- Encryption.
- Authentication.
- Authorization.
- Access control.
- Vulnerabilities.
- Security best practices.

There are many resources available online to learn Security. Here are some of the best resources:

- Encryption
- Authentication
- Authorization
- Access control
- Vulnerabilities
- Security best practices

Monitoring

What is Monitoring?

Monitoring is the process of observing the performance of a system over time. It is a must-have skill for any DevOps and System engineer.

How to learn Monitoring?

You should have a good understanding of the following topics:

- Monitoring tools.
- Monitoring best practices.

There are many resources available online to learn Monitoring. Here are some of the best resources:

- Monitoring Tools: Grafana
- Monitoring Tools: Prometheus
- Monitoring Tools: Nagios
- Monitoring Tools: Zabbix
- Monitoring Tools: PagerDuty

I am a Developer

Since you want to become a DevOps engineer, you should have a good understanding of software development. This section is about the general knowledge a DevOps engineer should have from software development perspective.

What is software development?

Need Information

What knowledge do you need from the Development world?

You should have a good understanding of the following topics:

- Topic 1
- Topic 2
- Topic 3

Best Programing Languages

- Python
- Javascript
    - Node / Typescript

Need Information

Programming Basics

Need Information

Progamming Advanced

Need Information

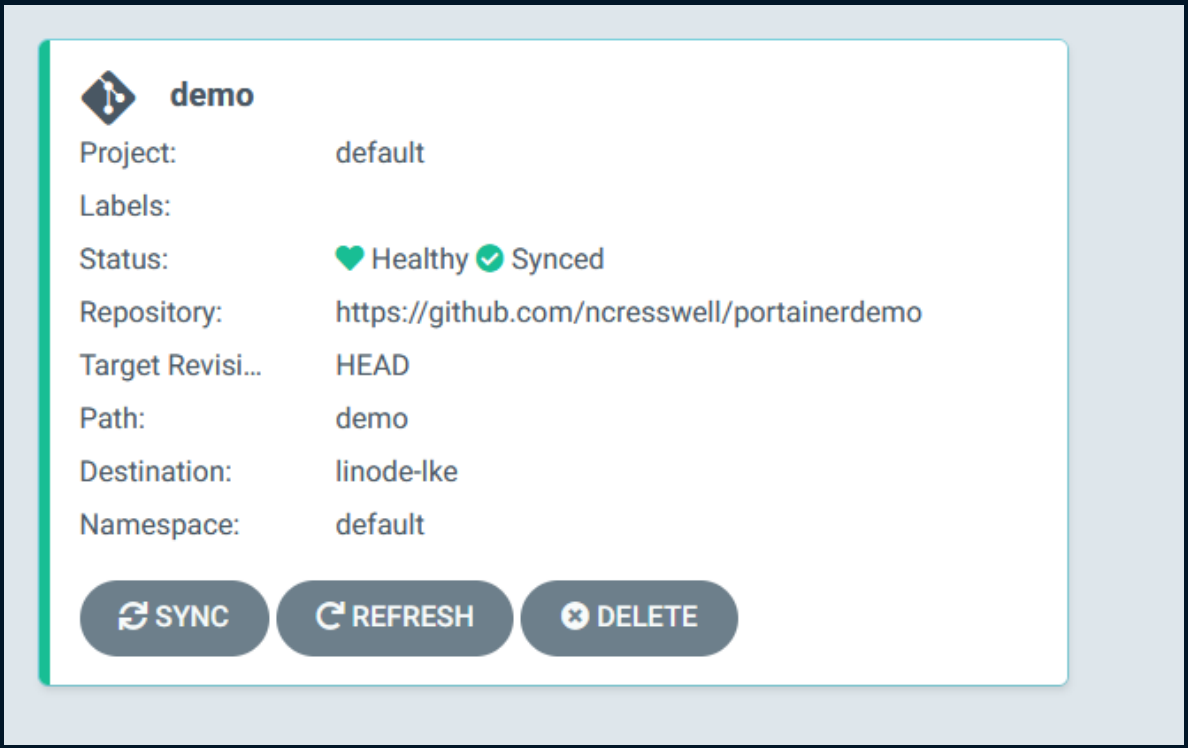
Programming in DevOps

Need Information

I am almost a DevOps

This is a checkpoint. Take your time and grab a cup of tea, you are now ready to become a DevOps engineer!

Here's an image of a healthy ArgoCD cluster for your future self to enjoy:



DevOps 101: What is DevOps

Text

What is DevOps

Text

DevOps Culture

Text

DevOps Tools

Text

DevOps 102: DevOps lifecycle, the DevOps mindset, and the DevOps principles

Text

DevOps Lifecycle?

Text

DevOps Mindset

Text

DevOps Principles

Text

DevOps 201: Continuous integration, Continuous Delivery, and Continuous Deployment

Continuous Integration

Text

Continuous Delivery

Text

Continuous Deployment

Text

DevOps 202: Infrastructure as Code, and Configuration Management

Infrastructure as Code

Text

Configuration Management

Text

DevOps 203: Monitoring, and Logging

Monitoring

Text

Logging

Text

DevOps 204: Security, and Disaster Recovery

Security

Text

Disaster Recovery

Text

DevOps 205: Containers, Orchestration, and Microservices

Containers

Text

Orchestration

Text

Microservices

Text

DevOps 205: Serverless, and Cloud Computing

Serverless

Text

Cloud Computing

Text

DevOps 205: DevOps in the Enterprise

DevOps in the Enterprise

Text

Section 2: Practice

Cloud 1: Deploy a DigitalOcean Droplet and Configure LEMP stack on it

Task Description

Some text

Guide

Some text

Step 1

Some text

Step 2

Some text

Step N

Some Text

Conclusion

Cloud 2: Deploy Laravel on a DigitalOcean droplet

Task Description

Some text

Guide

Some text

Step 1

Some text

Step 2

Some text

Step N

Some Text

Conclusion

Cloud 3: Configure a DigitalOcean Managed Database for your Laravel App

Task Description

Some text

Guide

Some text

Step 1

Some text

Step 2

Some text

Step N

Some Text

Conclusion

Cloud 4: Configure a DigitalOcean space for your Laravel static assets

Task Description

Some text

Guide

Some text

Step 1

Some text

Step 2

Some text

Step N

Some Text

Conclusion

Cloud 5: Deploy a Laravel App to a DigitalOcean Kubernetes Cluster

Task Description

Some text

Guide

Some text

Step 1

Some text

Step 2

Some text

Step N

Some Text

Conclusion

Linux 1: Automate Laravel Deployment With Bash

Task Description

Some text

Guide

Some text

Step 1

Some text

Step 2

Some text

Step N

Some Text

Conclusion

Linux 2: Load Balance Laravel With HAProxy

Task Description

Some text

Guide

Some text

Step 1

Some text

Step 2

Some text

Step N

Some Text

Conclusion

Linux 3: Migrate a Laravel App from one server to another

Task Description

Some text

Guide

Some text

Step 1

Some text

Step 2

Some text

Step N

Some Text

Conclusion

Linux 4: Write a Bash Script to Monitor System Resources (MEM, CPU, STORAGE) and send a Discord Notification

Task Description

Some text

Guide

Some text

Step 1

Some text

Step 2

Some text

Step N

Some Text

Conclusion

Database 1: Replicate a MySQL Database to a Slave Server

Task Description

Some text

Guide

Some text

Step 1

Some text

Step 2

Some text

Step N

Some Text

Conclusion

Containers 1: Deploy a Laravel App into a Docker Container

Get Started

Whenever you search on google you find tutorials on how to deploy Laravel App with docker-compose not just docker container.

But what is the difference between Docker and Docker Compose?

1. Docker run is entirely command line based, while docker-compose reads configuration data from a YAML file.
2. Docker run can only start one container at a time, while docker-compose will configure and run multiple.

When you are working with Laravel, you need multiple services to be configured while starting Laravel project, like: MySQL database, redis, mailhog and other services. All these can be configured together using docker-compose and they become connected to each other. But sometimes we just need to configure one container for our laravel project without reconfiguring all other services.

Here this tutorial come in handy, I will explain step by step how to install Laravel App project into a Docker container.

Prerequisite

Laravel project downloaded on your local environment.

Creating and writing into Dockerfile

Step 1: Create a Dockerfile

Let's start of by adding a docker file to the laravel project. So, in the root of the project add a new file called Dockerfile.

Step 2: Choose the right base image

After adding this Dockerfile, we need to write the first instruction which is FROM. FROM instruction is used to specify the base image. In our case the image will be operating system (alpine preferable) plus a run time environment which is PHP version that our project needs. Inorder to know which image to use we need to

1. Navigate to https://hub.docker.com/.
2. Write PHP in the search field on the top.
3. Click on PHP (Docker Official Image) - you will be navigated to php page
4. Click on tags
5. Write php version of the project in Filter Tags search input.
6. Copy the tag of this image
7. Paste the tag in the Dockerfile as follow:

```
FROM php:8.0.24-zts-alpine3.16
```

Step 3: Changing the working directory

For php projects we need to move all our files to /var/www/html folder. So let us add below FROM instruction:

```
WORKDIR /var/www/html
```

Step 3: Execluding files and directories

Before copying laravel app directories and files we need to execlude couple of folders that are not needed to be transfered to the image. We will add folder in the directory and name it: .dockerignore Inside it we will add the following: vendor/

So when building an image, docker will not copy a large directory to docker engine. This will reduce build context massively and unneeded.

Step 4: Installing all prerequisite packages

Before executing composer install, we have to install all the packages needed in our project. I will list here the common prerequests for all laravel projects, and you can add to them as much as needed.

```
RUN apk update
RUN curl -sS https://getcomposer.org/installer | php -- --
version=1.10.26 --install-dir=/usr/local/bin --filename=composer
```

Step 5: Copying files and directories

Now that we have a base image, and we are in the correct directory, next step is to copy the application files into the image. To do so, we need to add this instruction:

```
COPY . .
```

Step 6: Running commands

Next step we are going to install project dependencies using composer. We will use the RUN command. Add this line:

```
RUN composer install
```

Step 7: Defining entrypoints

Whenever we want to start a laravel application we have to execute this command: php artisan serve --host=0.0.0.0 Same in docker, to start the project we need this command in execute form to start our project, so we will add this line in the last line of Dockerfile

```
CMD ["php","artisan","serve","--host=0.0.0.0"]
```

Our final Dockerfile is as follow

```
FROM php:8.0.24-zts-alpine3.16
WORKDIR /var/www/html

RUN apk update
RUN curl -sS https://getcomposer.org/installer | php -- --version=2.4.3
--install-dir=/usr/local/bin --filename=composer

COPY . .
RUN composer install

CMD ["php","artisan","serve","--host=0.0.0.0"]
```

Build and run your container

Build

Whenever we want to start laravel project, you need to navigate to project files and build it using this command:

```
docker build -t laravel-app .
```

Run

When done build you will have to run laravel project using the following command:

```
docker run -d -p 81:8000 --name backend laravel-app
```

Open browser

Now in your browser write: localhost:81

Conclusion

With only couple of commands you can deploye your laravel project on docker, and leave all other configurations as is.

Happy Coding

Containers 2: Deploy a Laravel App with Docker Compose

Task Description

Some text

Guide

Some text

Step 1

Some text

Step 2

Some text

Step N

Some Text

Conclusion

Containers 3: Deploy a Laravel App to a Kubernetes Cluster

Task Description

Some text

Guide

Some text

Step 1

Some text

Step 2

Some text

Step N

Some Text

Conclusion

IaC 1: Deploy a DigitalOcean Droplet, Kubernetes Cluster, and a managed Database with Terraform

Task Description

Some text

Guide

Some text

Step 1

Some text

Step 2

Some text

Step N

Some Text

Conclusion

IaC 2: Deploy a Laravel Application on a DigitalOcean Droplet with a managed Database with Terraform and Ansible

Task Description

Some text

Guide

Some text

Step 1

Some text

Step 2

Some text

Step N

Some Text

Conclusion