Contents at a Glance

PART I Introduction

1 Hello World 27
2 Installation 37
3 Basic Principles 51
4 Custom Docker Images (Dockerfiles) 87
5 docker compose 105
6 Tips, Tricks, and Internal Details 123
7 docker Command Reference 153

PART II Toolbox

8 Alpine Linux 177
9 Web Servers and Company 185
10 Database Systems 213
11 Programming Languages 233
12 Web Applications and Content Management Systems 255

PART III Exercises

13 A Modern Web Application 271
14 Grafana 301
15 Modernizing a Traditional Application 319
16 GitLab 343
17 Continuous Integration and Continuous Delivery 375
18 Security 397
19 Swarm and Amazon Elastic Container Service 411
20 Kubernetes 435

Contents

Preface 21

PART I Introduction

1 Hello World 27

1.1 Docker Quick Install 27
1.1.1 Windows 27
1.1.2 macOS 27
1.1.3 Linux 28
1.2 Apache with PHP 8 28
1.2.1 Dockerfile 30
1.3 Node.js 31
1.4 Python 33

2 Installation 37

2.1 Docker Variants 37
2.2 Version Numbers 38
2.3 Installation on Windows 38
2.4 Installation on macOS 40
2.5 Installation on Linux 41
2.5.1 Installation on Ubuntu 42
2.5.2 Installation on the Raspberry Pi Operating System 44
2.5.3 Installation on Red Hat Enterprise Linux 8 and Fedora 44
2.5.4 Other Installation Variants 45
2.5.5 Troubleshooting and Logging 45
2.5.6 Working without "sudo" 45
2.6 Rootless Docker 46
2.6.1 "newuidmap" and "newgidmap" 47
2.6.2 Using a System-Wide Rootless Docker Installation 47
2.6.3 Installing Rootless Docker 49

2.6.4 No "ping" ... 49
2.6.5 Local Docker Files ... 50
2.6.6 Problems with the Virtual File System Directory ... 50

3 Basic Principles 51

3.1 Basic Principles and Terminology ... 51
3.1.1 Images and Containers ... 51
3.1.2 Volumes ... 52
3.1.3 Services, Stacks, and Clusters ... 53
3.1.4 docker Command versus Docker Daemon (dockerd) ... 54
3.1.5 Linux versus Windows ... 54
3.1.6 Virtual Machines versus Containers ... 55

3.2 Running Containers ... 56
3.2.1 Status Information ... 57
3.2.2 Tidying Up ... 57

3.3 Using Containers Interactively ... 58
3.3.1 Network Connection ... 60
3.3.2 Containers versus Images Again ... 61
3.3.3 Running the Container Again via "docker start" ... 62
3.3.4 Custom Container Names ... 63
3.3.5 Running Other Processes in Parallel via docker exec ... 64
3.3.6 Tidying Up ... 64

3.4 Port Redirection ... 64
3.4.1 Connecting Container Ports to Local Ports ... 65
3.4.2 Problems with Port 80 ... 66
3.4.3 Tidying Up ... 67

3.5 Data Storage in Volumes ... 67
3.5.1 Storing the Database in an Automatically Created Volume ... 69
3.5.2 Analyzing Volumes ... 69
3.5.3 Calling the MariaDB Client ... 71
3.5.4 A Look Inside the Container ... 72
3.5.5 Logging ... 72
3.5.6 Tidying Up ... 73

3.6 Named Volumes ... 73
3.6.1 Container Update without Data Loss ... 74
3.6.2 Tidying Up ... 74

3.7 Volumes in Custom Directories ... 75
3.7.1 Problems with Security-Enhanced Linux ... 76

3.7.2 Problems on Windows Hosts ... 76
3.7.3 Tidying Up ... 76

3.8 Communication between Containers ... 77
3.8.1 Setting Up a Private Docker Network ... 77
3.8.2 MariaDB ... 77
3.8.3 phpMyAdmin ... 78
3.8.4 WordPress ... 79
3.8.5 Stopping and Restarting Containers ... 81
3.8.6 Updates ... 81
3.8.7 Tidying Up ... 82

3.9 Administration of Docker ... 83
3.9.1 Determining Space Requirements of Images and Containers ... 83
3.9.2 Deleting Containers and Images ... 84
3.9.3 Managing Volumes ... 85
3.9.4 General Overview ... 85
3.9.5 Releasing Unused Space ... 85

4 Custom Docker Images (Dockerfiles) 87

4.1 Dockerfiles ... 87
4.1.1 Dockerfile Syntax ... 88
4.1.2 Introductory Example ... 88
4.1.3 Specifying the Source Image ("FROM") ... 89
4.1.4 Adding Files ("ADD" versus "COPY") ... 89
4.1.5 Container "start" Command ("CMD" and "ENTRYPOINT") ... 89
4.1.6 Running Commands (RUN) ... 92
4.1.7 Volume Directories (VOLUME) ... 92
4.1.8 Creating and Testing an Image ... 93
4.1.9 Tidying Up ... 94

4.2 A Custom Web Server Image ... 95
4.2.1 Creating and Testing an Image ... 96

4.3 Uploading Images to Docker Hub ... 97
4.3.1 Login ... 97
4.3.2 "push" Command ... 99
4.3.3 Image Description ... 99
4.3.4 Image Tags ... 99
4.3.5 Deleting Uploaded Images ... 100

4.4 Setting Up a Pandoc and LaTeX Environment as an Image ... 100
4.4.1 Installation Work ... 102

4.4.2 Working Directory and Volume ... 102
4.4.3 Creating the Image ... 102
4.4.4 Running the Container ... 103
4.4.5 Troubleshooting ... 103

5 docker compose 105

5.1 docker-compose versus docker stack ... 105
5.2 Installing "docker-compose" ... 106
5.3 YAML Ain't Markup Language Syntax ... 107
5.4 Hello Compose! ... 109
5.4.1 "docker compose" ... 110
5.4.2 "docker stack deploy" ... 111
5.4.3 Debugging ... 113
5.4.4 Interactive Use ... 113
5.5 The docker-compose.yml File ... 114
5.5.1 Image or Dockerfile? ... 115
5.5.2 Networks ... 115
5.5.3 Network Ports ... 116
5.5.4 Volumes ... 116
5.5.5 Named Volumes ... 117
5.5.6 Environment Variables ... 118
5.5.7 Command Execution ... 118
5.5.8 Restart Behavior ... 119
5.5.9 Deploy Settings ... 119
5.6 Passwords and Other Secrets ... 120
5.6.1 Sharing MariaDB/MySQL Passwords between Two Containers ... 121
5.6.2 Secure Exchange of Secrets ... 122

6 Tips, Tricks, and Internal Details 123

6.1 Visual Studio Code ... 123
6.1.1 Docker Extension ... 124
6.1.2 Remote Containers ... 125
6.2 Portainer ... 126
6.2.1 Tidying Up ... 128

6.3 Pull Limit in Docker Hub ... 128
6.3.1 Lots of Activity in Docker Hub ... 128
6.3.2 No Unlimited Access to Images ... 129
6.3.3 Bypassing the Limit ... 130
6.3.4 Using Alternative Registries ... 132

6.4 Using Different CPU Architectures ... 133
6.4.1 Working on ARM Computers ... 135
6.4.2 Running Containers of a Different Architecture ... 136

6.5 Starting Containers Automatically ... 137
6.5.1 Restart Behavior with docker compose ... 138
6.5.2 Automatically Starting Docker Containers on a Linux Server with systemd ... 139
6.5.3 A Separate Service File ... 139
6.5.4 Starting and Ending the Service ... 141

6.6 A Look Behind the Scenes ... 142
6.6.1 docker, dockerd, and containerd ... 142
6.6.2 The Overlay File System ... 142
6.6.3 The /var/lib/docker Directory ... 143
6.6.4 Process Management ... 144
6.6.5 Resource Management through Control Groups ... 145
6.6.6 Container Isolation through Namespaces ... 145
6.6.7 Why the docker Command Usually Requires Root Privileges ... 146
6.6.8 More "sudo" Convenience ... 147
6.6.9 Network Management ... 147
6.6.10 Windows (WSL2) ... 148
6.6.11 Limiting RAM Utilization through Docker and WSL2 ... 149
6.6.12 macOS (Hypervisor and Virtualization Framework) ... 150

7 docker Command Reference ... 153

7.1 docker attach ... 155
7.2 docker build ... 155
7.3 docker commit ... 155
7.4 docker compose * ... 156
7.5 docker-compose config ... 156
7.6 docker-compose down ... 156
7.7 docker-compose events ... 156
7.8 docker-compose kill ... 156

7.9 docker-compose logs 157
7.10 docker-compose pause and docker compose unpause 157
7.11 docker-compose ps 157
7.12 docker-compose rm 157
7.13 docker-compose run 157
7.14 docker-compose start, docker-compose stop, and docker compose restart 158
7.15 docker-compose top 158
7.16 docker-compose up 158
7.17 docker container * 159
7.18 docker create 159
7.19 docker diff 159
7.20 docker events 160
7.21 docker exec 160
7.22 docker export 160
7.23 docker image * 160
7.24 docker images 161
7.25 docker import 161
7.26 docker info 161
7.27 docker inspect 162
7.28 docker kill 163
7.29 docker login and docker logout 163
7.30 docker logs 163
7.31 docker network * 164
7.32 docker node * 165
7.33 "docker pause" and "docker unpause" 165
7.34 "docker port" 165
7.35 docker ps 166
7.36 docker pull 166
7.37 docker push 167
7.38 docker rename 167
7.39 docker restart 167
7.40 docker rm and docker rmi 167

7.41 docker run ... 168
7.42 docker secret * ... 169
7.43 docker service * ... 170
7.44 docker stack * ... 170
7.45 docker start and docker stop ... 171
7.46 docker stats ... 171
7.47 docker swarm * ... 172
7.48 docker system * ... 172
7.49 docker tag ... 173
7.50 docker top ... 173
7.51 docker update ... 173
7.52 docker volume * ... 174
7.53 docker wait ... 174

PART II Toolbox

8 Alpine Linux 177

8.1 Characteristics ... 177
8.1.1 Trying Out Alpine Linux ... 178
8.1.2 Shell ... 178
8.1.3 BusyBox ... 179
8.1.4 Init System and Logging ... 179
8.1.5 The "musl" Library ... 179
8.1.6 Documentation ... 180
8.1.7 Missing "root" Password ... 180
8.2 Package Management with "apk" ... 181
8.2.1 Package Sources ... 182
8.2.2 Packages and Their Files ... 183

9 Web Servers and Company 185

9.1 Apache HTTP Server ... 185
9.1.1 Using an Official Docker Image ... 185

9.1.2 Using a Custom Dockerfile 186
9.1.3 Using the Official Docker Image 189

9.2 Nginx 191
9.2.1 Usage with a Custom Dockerfile 191

9.3 Nginx as Reverse Proxy with SSL Certificates from Let's Encrypt 193
9.3.1 Nginx Reverse Proxy 194
9.3.2 Creating the First Certificate 195
9.3.3 Application Programming Interface Server 198
9.3.4 Renewing Certificates 200

9.4 Node.js with Express 201
9.4.1 Express Framework 201
9.4.2 Docker Development Environment for Express 201
9.4.3 Docker Production Environment for Express 204
9.4.4 Debugging the Applications 204

9.5 HAProxy 205
9.5.1 Hello World! 205

9.6 Traefik Proxy 207
9.6.1 Installing and Configuring Traefik 207
9.6.2 Connecting Docker Services with Traefik 211

10 Database Systems 213

10.1 MySQL and MariaDB 213
10.1.1 Setting Up and Using MariaDB Containers Manually 214
10.1.2 Setting Up a MariaDB Service with "docker-compose" 215
10.1.3 Setting Up the Database during Initialization 217
10.1.4 Backups 217
10.1.5 Using a Custom Configuration File (my.cnf) 218

10.2 PostgreSQL 219
10.2.1 PostgreSQL and pgAdmin with "docker compose" 219
10.2.2 Backups with "docker compose" 222

10.3 MongoDB 223
10.3.1 Simple "docker compose" Setup with MongoDB 224
10.3.2 Database Access 227
10.3.3 MongoDB with Authentication 229

10.4 Redis 230
10.4.1 Redis Replication with "docker compose" 231

11 Programming Languages 233

11.1 JavaScript (Node.js) 233
11.1.1 Trying Out Node.js 234
11.1.2 "printheadlines": Packaging Node.js Script as a Docker Image 234

11.2 Java 236
11.2.1 "printheadlines" Example 237
11.2.2 Resource Management 239

11.3 PHP: Hypertext Preprocessor 240
11.3.1 Official PHP Docker Images 240
11.3.2 PHP as a Command-Line Program 241
11.3.3 PHP with Web Servers (EXIF Example) 242
11.3.4 Dockerfile for the EXIF Example 244
11.3.5 docker-compose.yml for the EXIF Example 246

11.4 Ruby 247
11.4.1 "printheadlines" Example 247

11.5 Python 248
11.5.1 "printheadlines" Example 248
11.5.2 Packaging an Existing Python Application as a Docker Image 249
11.5.3 Sample Code 250
11.5.4 Localization and Character Sets in Python Programs 252

12 Web Applications and Content Management Systems 255

12.1 WordPress 255
12.1.1 Custom Docker Images 255
12.1.2 The "wp-cli" Command 258
12.1.3 Dealing with Passwords 258
12.1.4 Update 260
12.1.5 Backup 261

12.2 Nextcloud 263
12.2.1 Installation via "docker compose" 263
12.2.2 Creating Backups 265

12.3 Joomla 266

PART III Exercises

13 A Modern Web Application 271

13.1 The Application 272
13.1.1 Technical Background 274

13.2 The Frontend: Vue.js 274
13.2.1 Vue.js Setup 275
13.2.2 Development and Production Image (Multistage Builds) 276
13.2.3 Implementation via docker-compose.override.yml 277
13.2.4 Bundling JavaScript Code with Webpack 278
13.2.5 Docker "ENTRYPOINT" Script for Node.js Modules 279
13.2.6 Source Code 280
13.2.7 The Main Vue.js Component 281

13.3 The API Server: Node.js Express 284
13.3.1 Stateless or Stateful? 286
13.3.2 Node.js Modules in the Development Environment 287
13.3.3 Accessing the Application Programming Interface 288
13.3.4 Source Code (server.js) 290
13.3.5 Source Code (routes.js) 292

13.4 The Mongo Database 294

13.5 The Session Storage: Redis 298

14 Grafana 301

14.1 Grafana-Docker Setup 301
14.1.1 Generating Data with Collectors (Telegraf) 302
14.1.2 Storing Data with InfluxDB 304
14.1.3 Visualizing Data with Grafana 305

14.2 Provisioning 310

14.3 A Customized Telegraf Image 313
14.3.1 Dashboard Customizations 317

15 Modernizing a Traditional Application 319

15.1 The Existing Application ... 320

15.2 Planning and Preparation ... 322

15.2.1 The Web Server ... 324
15.2.2 WordPress ... 326
15.2.3 WordPress Database Migration ... 328
15.2.4 Automatic Updates in WordPress ... 329
15.2.5 MariaDB ... 330
15.2.6 Node.js Server ... 331
15.2.7 The Map Application ... 332
15.2.8 Cron Jobs ... 334
15.2.9 Backups ... 334

15.3 The Development Environment ... 335

15.4 Production Environment and Migration ... 337

15.5 Updates ... 339

15.6 Tips for the Transition ... 340

15.7 Results ... 341

16 GitLab 343

16.1 GitLab Quick Start ... 344

16.2 GitLab Web Installation ... 346

16.3 HTTPS via a Reverse Proxy Setup ... 348

16.4 Email Dispatch ... 348

16.4.1 Sending Emails with Your Own Exim Mail Server ... 349
16.4.2 Integration into the GitLab Configuration ... 350

16.5 Secure Shell Access ... 352

16.6 Volumes and Backup ... 352

16.6.1 Backing Up GitLab Application Files ... 353
16.6.2 Restore ... 353

16.7 Custom Docker Registry for GitLab ... 355

16.8 The Complete docker-compose File ... 356

16.9 Using GitLab ... 358

16.9.1 User ... 359
16.9.2 Projects ... 360

16.9.3 Wiki ... 361
16.9.4 Tickets and Issues ... 361

16.10 GitLab Runner ... 362
16.10.1 Installation ... 362
16.10.2 The Shell Executor ... 364
16.10.3 The Docker Executor ... 364
16.10.4 The Kubernetes Executor ... 365

16.11 Mattermost ... 369
16.11.1 Connecting to GitLab ... 371

17 Continuous Integration and Continuous Delivery 375

17.1 The dockerbuch.info Website with gohugo.io ... 376
17.1.1 Running Hugo as a Docker Container ... 377
17.1.2 Hugo Quick Start ... 378

17.2 Docker Images for the CI/CD Pipeline ... 380
17.2.1 Web Server Docker Image ... 381
17.2.2 Link Checker ... 381
17.2.3 HTML Validator ... 382

17.3 The CI/CD Pipeline ... 384
17.3.1 CI/CD with the GitLab Shell Runner ... 384
17.3.2 Testing the Pipeline ... 388
17.3.3 CI with the GitLab Docker Runner ... 389
17.3.4 CI/CD with Kubernetes in the Google Cloud ... 393

18 Security 397

18.1 Software Installation ... 397

18.2 Origin of Docker Images ... 398
18.2.1 External Registries ... 399

18.3 "root" in Docker Images ... 401
18.3.1 Nginx Web Server without "root" Privileges ... 401

18.4 The Docker Daemon ... 403
18.4.1 Rootless Docker ... 404

18.5 User Namespaces 404
18.5.1 Functionality 405
18.6 Control Groups 406
18.7 Secure Computing Mode 408
18.8 AppArmor Security Profiles 408
18.8.1 AppArmor Profile for Docker 408
18.8.2 Custom AppArmor Rules 409

19 Swarm and Amazon Elastic Container Service 411

19.1 Swarm versus Kubernetes 411
19.2 Docker Swarm 412
19.2.1 Swarm Quick Start 413
19.2.2 "Hello World" in the Swarm 414
19.3 Docker Swarm in the Hetzner Cloud 417
19.3.1 CLI Installation and Configuration with "hcloud" 418
19.3.2 Installing Docker with "cloud-init" 419
19.3.3 Creating Cloud Instances 420
19.3.4 Launching the Diary App in Docker Swarm 422
19.3.5 Secure Sockets Layer Certificates 423
19.3.6 Simulating a Server Failure 426
19.3.7 Security 427
19.4 Amazon Elastic Container Service 428
19.4.1 Quick Start: Launching a Container in the AWS Cloud 428
19.4.2 Amazon Elastic Container Service with "docker compose" 431

20 Kubernetes 435

20.1 Minikube 436
20.1.1 Autocompletion for "kubectl" 438
20.1.2 The First Deployment in Minikube 438
20.1.3 Object Configuration in Files 441
20.1.4 Kubernetes as a Cloud Platform 443
20.1.5 Grafana Setup in Minikube 444

20.2 Amazon Elastic Kubernetes Service 447
20.2.1 Creating Clusters with "eksctl" 448
20.2.2 Deleting Data Clusters 452
20.3 Microsoft Azure Kubernetes Service 452
20.3.1 Creating Clusters 453
20.3.2 Installing an Application 455
20.3.3 Starting the Application 457
20.3.4 First Test 459
20.3.5 Tidying Up 460
20.4 Google Kubernetes Engine 461
20.4.1 Creating a Project and Cluster 461
20.4.2 The "gcloud" Command 462
20.4.3 The Diary Application in the Google Kubernetes Engine 464
20.4.4 Using Secure Sockets Layer Certificates from Let's Encrypt 466

Appendices 471

A Podman as a Docker Alternative 471
B The Authors 483

Index 485