

Inhalt

TEIL I Brave New World?

1	Die neue alte Welt der Virtualisierung	37
1.1	Evolution, Beschleunigung und Standbilder	39
1.2	Vorbemerkungen	40
1.2.1	Verwendete Formatierungen	40
1.2.2	Weiterführende Hinweise	41
1.2.3	Beispieldateien	41
1.3	Was dieses Buch sein/nicht sein soll	41
1.3.1	Was es sein soll	41
1.3.2	Was es nicht sein soll und nicht ist	41
1.3.3	Wie dieses Buch zu lesen ist	42
1.4	Verwendete Plattformen und Versions-Spezifikationen	43
1.4.1	Vorbetrachtungen	43
1.4.2	Container-OS und die Zukunft	44
2	Container	45
2.1	Warum Container?	45
2.2	Microservices	46
2.2.1	Wie erkläre ich es meinem CEO?	46
2.2.2	Die neue Welt der Microservices: Admins/DevOps-Teams	47
2.2.3	Die neue Welt der Microservices: aus der Perspektive der CEOs/Entscheider	49
2.3	Continuous Delivery/Continuous Integration und DevOps	50
2.3.1	Semi- oder vollautomatisch: Continuous Integration/ Continuous Delivery	50
2.3.2	CD/CI und das Big Bang Release-Problem	51
2.4	Continuous Delivery	52
2.4.1	Was verstehen wir darunter?	52
2.4.2	Continuous Delivery Pipelines	54
2.4.3	Commit-Stage	55

2.4.4	Acceptance-Test-Stage	56
2.4.5	Exkurs: Acceptance-Tests und Dreieinigkeit	56
2.4.6	Load-/Capacity-, Security- und Exploration-Tests	57
2.4.7	Rollout/Go-Live	57
2.4.8	Die Gates	57
2.4.9	Fazit: Wo kann CD nutzbringend eingesetzt werden?	57
2.5	DevOps: Gewaltenteilung oder Kooperation?	59
2.5.1	Grundsätzliche Betrachtungen	59
2.5.2	Das DevOps-Prinzip	59
2.5.3	Kommunikationsblackouts im DevOps-Team	60
2.5.4	Das konkrete DevOps-Problem im klassischen Umfeld	60
2.5.5	»Works for me« und anderer Nonsense im DevOps-Business – und ein Ausweg?	61
2.5.6	BizDevOps – und noch eine Silbe	63

TEIL II Single Node Container-Systeme

3 Container-Plattformen, Basics und Konzepte 67

3.1	World of Tiers – Teil 1	67
3.2	Container – Basics	68
3.2.1	Namespaces und Container-Konzepte	68
3.2.2	Enter Namespace – nsenter	70
3.2.3	Namespaces und Sicherheit?	72
3.3	VMs – obsolet durch Container?	74
3.3.1	Container vs. VM	75
3.3.2	Packungsdichte und Ressourcen	78
3.4	Zwischenfazit	80
3.5	Container-Lösungen im Überblick	81
3.5.1	Von LXC zu Docker	81
3.5.2	Docker	81
3.5.3	LXD	89
3.5.4	CoreOS /Container Linux und Rocket	91
3.5.5	Im Rampen-»Licht«? – VMware Photon	93
3.5.6	Überblick der Container-Formate: Docker vs. CoreOS/Rkt vs. LXD vs. Photon ... und OCF	94
3.5.7	Fazit	95

3.6	Container: eine funktionale Übersicht	96
3.6.1	Aufbau eines Container-Hosts	96
3.6.2	Docker Images	96
3.6.3	Anzahl der Layer	97

4 Docker 99

4.1	Docker-Versionen	99
4.1.1	Docker-Versionen, wichtige Meilensteine und Inkompatibilitäten	99
4.1.2	Docker-LTS-Versionen?	100
4.1.3	Betrachtete Plattformen und Docker-Versionen	101
4.1.4	Docker-Versionsnummern und Bedeutung	101
4.1.5	Funktionaler Überblick: Docker CLI, dockerd, Registry	102
4.2	Docker-Installation	103
4.2.1	Paketnamen und Dependencies	103
4.2.2	Docker Bash-Completion	104
4.2.3	Docker-Installation unter Ubuntu 16.04 LTS	104
4.2.4	Docker-Installation unter RHEL/CentOS 7.3	105
4.2.5	Virt-7-Repo für CentOS	107
4.2.6	Installation per yum-config	107
4.2.7	Default-Setup unter SUSE/SLES	109
4.2.8	SUSE/SLES 12 und BTRFS-/Docker-Problematiken	109
4.2.9	SUSE-/SLES-spezifische Docker-Konfigurationsdateien	110
4.2.10	SUSE-spezifische Docker-Repos	110
4.2.11	Installation der Commercially Supported Docker Engine	111
4.2.12	Distributionsunabhängige Installation von Docker 1.13	114
4.3	Deinstallation/Upgrade/Umstellung auf andere Storage Backends	116
4.3.1	Deinstallation	116
4.3.2	Upgrade	116
4.3.3	Umstellung des Storage Backends	116
4.4	Docker und systemd-Integration	116
4.4.1	Vorbetrachtungen	116
4.4.2	systemd-Service-Units für Docker	117
4.5	Docker im Betrieb	118
4.5.1	Vorbetrachtungen	118
4.5.2	Permanente Dienststeinbindung	119
4.5.3	Lokale HA	120
4.5.4	Verbose Mode	120

4.5.5	Status Überprüfung/Features	120
4.5.6	Docker-Systeminformationen	122
4.5.7	Docker Daemon-Konfigurationsmöglichkeiten	122
4.5.8	Mögliche Startoptionen/Schalter des Docker Daemons	123
4.5.9	Konfiguration per /etc/docker/daemon.json	125
4.5.10	Alternatives Docker-Verzeichnis als Konfigurationsbeispiel	127
4.5.11	Plugins	128
4.6	Docker Image-Management – Basics	128
4.6.1	Vorbetrachtungen	128
4.6.2	Auszug der Docker CLI-Subkommandos	129
4.6.3	Neue CLI-Strukturen in 1.13	130
4.6.4	Docker 1.13 sowie neue container- und image-Subkommandos	131
4.6.5	Einfaches Image-Management	131
4.6.6	Docker-Namensräume und das Default Registry-Problem	132
4.6.7	Docker Images (unter docker.io) suchen	134
4.6.8	Image-Schema-Versionen	135
4.6.9	Offizielles CentOS-Image von docker.io pullen	136
4.6.10	Lokal verfügbare Docker Images listen und filtern	137
4.6.11	Meta-Informationen von lokalen Images abfragen	141
4.6.12	Images löschen	144
4.6.13	docker save & load Images	145
4.6.14	Dangling Images: The good and the bad <none>:<none>	146
4.6.15	Build-History eines Images inspizieren	149
4.7	Trusted Images	150
4.7.1	Vorbetrachtungen	150
4.7.2	Trusted Docker Images unter SUSE/SLES mit sle2docker	150
4.7.3	Eigenes, generisches Trusted Basis-Image erzeugen	153
4.7.4	Gescriptete Image-Erzeugung (YUM Based)	153
4.7.5	Mikro-Image »from scratch«	154
4.7.6	Images: grundlegende Security-relevante Betrachtungen	155
4.8	Betrieb und Management von Docker-Containern	156
4.8.1	Vorbetrachtungen	156
4.8.2	Kurzübersicht der relevanten Docker CLI-Kommandos	157
4.8.3	Neues docker container-Subkommando	158
4.8.4	docker history	158
4.8.5	Container starten – docker [container] run	159
4.8.6	(Random) Container-Names und automatische Löschung (run --rm)	160
4.8.7	Container-HA: automatische Restarts	161
4.8.8	docker [container] run --readonly	162

4.8.9	Detached Container im Hintergrund starten	162
4.8.10	Auflisten von Container-Instanzen – docker ps	164
4.8.11	Starten und Stoppen existierender Container	167
4.8.12	docker [container] rename	169
4.8.13	Container-Instanzen löschen: docker [container] rm/prune	170
4.8.14	docker [container] attach-Optionen	170
4.8.15	Befehle im laufenden Container ausführen:	
	docker [container] exec	172
4.8.16	docker [container] create	173
4.8.17	Container-Instanzen exportieren und als Images importieren	174
4.8.18	Kopieren von Daten: Container <-> Host	175
4.8.19	docker checkpoint	176
4.9	Prozessverwaltung im Container	178
4.9.1	docker top	178
4.9.2	Prozesse im Container beenden	180
4.9.3	docker wait und Return-/Exit-Codes	180
4.9.4	Container und alle Prozesse darin temporär pausieren	181
4.9.5	Live-Events mit docker events	181
4.9.6	Exkurs: Container-Capabilities/-Privilegien	183
4.9.7	Prüfung der Capabilities	183
4.10	Docker Logging	186
4.10.1	Log Driver	186
4.10.2	Zentralisierte Logs für Container-Instanzen	187
4.10.3	Container-Logs mit docker logs	188
4.11	Einfache Applikationen im Container	189
4.11.1	Vorbetrachtungen	189
4.11.2	Installation von Applikationen im gestarteten Container	189
4.12	Image-Modifikationen commiten und taggen	193
4.12.1	Vorbetrachtungen	193
4.12.2	Commit – Beispiel	194
4.12.3	Nachträgliches Taggen von Images	196
4.13	Layer-Strukturen	199
4.13.1	Vorbetrachtungen	199
4.13.2	Verzeichnisstrukturen auf dem lokalen Docker Host	199
4.13.3	Was ist beim letzten Commit passiert, wo liegt der neue Layer (Struktur)?	200
4.13.4	IDs der RW Layer von gestarteten Containern und Querbezüge	200
4.13.5	Layer-Analyse und Flattening (Zusammenfassung)	202

4.14	Limitierte Container-Instanzen	204
4.14.1	Vorbetrachtungen	204
4.14.2	docker [container] stats	205
4.14.3	Mögliche Limitierungen	205
4.14.4	Beispiele aus der Praxis für limitierte Container-Instanzen	207
4.14.5	Nachträgliche Limitierung	207
4.15	Dedizierte Docker Image-Stände bauen (docker build) und verwalten	208
4.15.1	Vorbetrachtungen	208
4.15.2	Best-Practice/File-Hierarchie	210
4.15.3	docker build	210
4.15.4	Dockerfile-Direktiven/-Instruktionen	211
4.15.5	Build-Anwendungsbeispiel: Apache-Container	223
4.15.6	Weiteres Beispiel: ssh-Container	226
4.15.7	Docker Images mit systemd	227
4.16	Best Build Practices	231
4.16.1	Container sind kurzlebig und jederzeit reproduzierbar	231
4.16.2	Wer hat's gemacht?	231
4.16.3	Wie ist es bezeichnet?	232
4.16.4	Verwenden eines eigenen Build-Ordnern pro Template	232
4.16.5	Verwendung eines .dockerignore-Files	232
4.16.6	Schlanke Images	233
4.16.7	Nur ein Prozess pro Container	233
4.16.8	Anzahl der Layer minimieren/niedrig halten	234
4.16.9	Multi-Line-Argumente in Befehlen (alphanumerisch) sortieren	234
4.16.10	Build-Cache	234
4.17	Docker Networking	235
4.17.1	Vorbetrachtungen	235
4.17.2	Packungsdichten und die Realität	236
4.17.3	Rework	236
4.17.4	docker network-Hilfesystem	237
4.17.5	Basics: Netzwerkverbindung zum Container	237
4.17.6	Docker und iptables	239
4.17.7	IP eines gestarteten Docker-Containers auslesen	242
4.17.8	IP-Zuweisung und die /etc/hosts im Container	243
4.17.9	Komplette Netzwerk-Info eines Containers auslesen	244
4.17.10	Docker networks: bridge, host, none und mehr	245
4.17.11	--net(work)= in der Praxis	246
4.17.12	Kommunikation: Welt zu Container, Docker Portmapping	249
4.17.13	Portmapping explizit setzen	252
4.17.14	Docker-Netzwerke ab Version 1.9 einrichten und modifizieren	254

4.17.15	Welche Möglichkeiten stellt das neue docker network zur Verfügung?	256
4.17.16	Die neuen Netzwerk-Driver im Detail	257
4.17.17	Network Connect ändern	258
4.17.18	docker network create – userdefinierte Netzwerke hinzufügen	259
4.17.19	Erzeugen eines neuen Bridged Networks	260
4.18	Container miteinander verknüpfen	262
4.18.1	Vorbetrachtungen	262
4.18.2	Legacy Links	263
4.18.3	Beispiel-Setup Legacy Link	263
4.18.4	Beispiel-Setup: Legacy Link Apache/OpenLDAP-Container	266
4.18.5	Verlinkung über userdefinierte Netzwerke	272
4.18.6	docker network prune	281
4.19	Docker-Compose	282
4.19.1	Vorbetrachtungen	282
4.19.2	Bearbeitung von Yaml-Konfigurationsdateien	283
4.19.3	Was passiert beim Rollout?	283
4.19.4	Installation	284
4.19.5	Compose – Praxisbeispiel	285
4.19.6	Apache und OpenLDAP als Compose-Rollout	286
4.19.7	Build and Run	287
4.19.8	Handling der Services per docker-compose	289
4.19.9	Auszüge der gängigsten docker-compose-Sub-Befehle	290
4.19.10	Docker-Compose Startup - Dependencies	293
4.19.11	Docker-Compose und Application Bundles	294
4.20	Docker Storage Driver und Volumes	296
4.20.1	Vorbetrachtungen	296
4.20.2	Storage Driver (local)	297
4.20.3	Übersicht der Storage Driver – Vor- und Nachteile	297
4.20.4	Storage Driver und Filesystem-Kombinationen	298
4.20.5	So what? – Entscheidungsfragen	299
4.21	Deep Dive in die Beziehung zwischen Images/Container-Instanzen und dem Storage Driver	300
4.21.1	Grundsätzliches: Images, Layer und der Storage Driver	300
4.21.2	Graphdriver?	301
4.21.3	Aufgaben des Storage Drivers	301
4.21.4	Exkurs: Secure Content Hashes und Content addressable Storage ab Docker 1.10	301
4.21.5	Image Layering und Sharing gemeinsamer Layer	302

4.21.6	Read/Write-Container, Readonly Image, Storage Driver und Datenspeicherung	303
4.22	Storage Driver im Detail	307
4.22.1	AUFS	307
4.22.2	OverlayFS	309
4.22.3	Neuerungen in Overlay(FS)2	312
4.22.4	BTRFS	313
4.22.5	Devicemapper Storage Driver	319
4.22.6	Umbau loop-lvm auf direct-lvm (RHEL/CentOS 7.x)	327
4.22.7	ZFS	335
4.22.8	Schlussbemerkung	341
4.23	Data Sharing mit Docker Volumes	342
4.23.1	Vorbetrachtungen	342
4.23.2	Docker Data Volumes	342
4.23.3	Einfache Data Volumes	343
4.23.4	Host mounted Data Volumes	345
4.23.5	Data Volume Mounts vom Host für OpenLDAP-Container	349
4.23.6	Volumes From	352
4.23.7	Maximale Portabilität über reine Docker Data Volumes	354
4.23.8	Readonly Data Volumes	356
4.23.9	Backup, Restore und Migration von reinen Docker Data Volumes	357
4.23.10	(Anonyme) Volumes entfernen	358
4.23.11	Zusammenfassung	359
4.24	Spezielle Volume Plugins	359
4.24.1	Vorbetrachtungen	359
4.24.2	Überblick	360
4.24.3	NFS-Share als Docker Volume	361
4.24.4	Netshare-Plugin (NFS/CIFS)	363
4.24.5	vSphere Volume Plugin	364
4.24.6	Eigene Volume Plugins	369
4.24.7	Docker Storage Volumes – grundsätzliche Schlussbetrachtung	369

5 Docker Security 371

5.1	Docker Security: TLS/SSL	371
5.1.1	Vorbetrachtungen	371
5.1.2	Grundlagen	372
5.1.3	SSL-Standard und veraltete Protokolle	372

5.1.4	Vorbetrachtungen zur Zertifikatserzeugung	373
5.1.5	Anpassungen der openssl.cnf	374
5.1.6	Erzeugung der Zertifikate mit angepasster OpenSSL-Konfiguration	376
5.1.7	Die CA erzeugen	376
5.1.8	Zertifikats-Request erzeugen	378
5.1.9	Zertifikats-Request signieren	379
5.1.10	Bearbeiten des Keys	380
5.1.11	Erzeugte Dateien und weitere Tasks	380
5.1.12	Weitere Docker Hosts/Zertifikate	381
5.1.13	Daemon-Startparameter	381
5.1.14	Test des Client-Zugriffs per TLS	382
5.1.15	Übersicht der zur Verfügung stehenden TLS-Flags für Server und Client	385
5.2	Docker Security: Content Trust/Notary	385
5.2.1	Vorbetrachtungen	385
5.2.2	DCT (Docker Content Trust) und TUF	386
5.2.3	Notary	388
5.2.4	Manuelles Notary-Test-Setup	389
5.2.5	Build and Run	391
5.2.6	Notary-Client-CLI	392
5.3	Docker Security: Vulnerability Scanner	394
5.3.1	Vorbetrachtungen	394
5.3.2	Clair	394
5.3.3	Clair – Beispiel-Setup	396
5.3.4	Clair und PostgreSQL-Rollout via Compose	397
5.3.5	Überprüfung eines vorhandenen Image per analyze-local-images	399
5.3.6	Docker Bench	402
5.3.7	Testlauf Docker Bench	402

6 Die eigene Trusted Docker Registry 405

6.1	Die Registry im Detail	405
6.1.1	Vorbetrachtungen	405
6.1.2	Up and running?	406
6.1.3	Registry-Architektur	407
6.1.4	Registry Upgrades V1 zu V2	408

6.2	Vorbereitungen zum Setup	408
6.2.1	Insecure Registry	408
6.2.2	Das Docker Default Registry-Image	409
6.2.3	Konfigurations-Override	411
6.3	Registry Setup-Möglichkeiten	413
6.3.1	Die Container-basierte Docker Registry	413
6.3.2	RHEL/CentOS	414
6.3.3	SLES 12/SUSE Leap	415
6.4	Registry-Setup: Vorbereitungen und Betrieb	416
6.4.1	Vorbereitungen für den Upload in eine Insecure-Registry	416
6.4.2	Vorbereitungen: Regeln zum Image Tagging für Registry-Uploads verstehen	417
6.4.3	Upload/Push eines Images in die Registry	417
6.4.4	Löschen von Images in einer privaten Registry	418
6.5	Lokaler Registry-Mirror	421
6.5.1	Vorbetrachtungen	421
6.5.2	Beispiel	421
6.6	Docker Registry mit TLS	424
6.6.1	Vorbetrachtungen	424
6.6.2	Zertifikatslokationen	424
6.6.3	CA, Zertifikat und Key für den Registry-Host	425
6.6.4	Start und Betrieb der Registry mit TLS	426
6.6.5	Push in die Registry mit TLS	428
6.6.6	Verbindung zur TLS gesicherten Registry mit Client-Zertifikat	429
6.6.7	Distributionsspezifische Registry mit TLS (SUSE Leap)	430
6.6.8	Troubleshooting-Tipps	432
6.7	Registry-Authentifizierung	432
6.7.1	Vorbetrachtungen	432
6.7.2	Simple/Basic Authentication via htpasswd	433
6.7.3	Distributionsspezifische Registry mit Simple/Basic Auth	436
6.7.4	Silly-Authentifizierung	436
6.7.5	Token-Authentifizierung	437
6.8	Zentrale Registry-Authentifizierung via LDAP/TLS	438
6.8.1	Vorbetrachtungen	438
6.8.2	Entfernen der Simple/Basic Auth-Konfiguration	439
6.8.3	Setup des LDAP-Images mit TLS	440
6.8.4	Setup des HTTP-Proxy	443
6.8.5	Build & Start Registry/LDAP/http-Proxy per Compose	447

6.8.6	Test des Setups	448
6.8.7	Zugriffskontrollen	450
6.9	Registry mit AD-Authentifizierung	452
6.9.1	Konzepte und Verfahren	452
6.9.2	Simple Bind	452
6.9.3	SSO/Kerberos-Anbindung	452
6.10	Docker Management – Uls	453

7 Weitere Container-Plattformen 455

7.1	Atomic Host (RHEL/CentOS)	456
7.1.1	Grundsätzliches	456
7.1.2	Die Funktionsweise von Atomic Host/Aufbau des Dateisystems ...	456
7.1.3	Upgrades und Rollbacks	456
7.1.4	Installation und Betrieb	459
7.1.5	Atomic Scan (CVE-Scanner)	460
7.1.6	Skopeo	461
7.2	Docker unter Windows	463
7.2.1	Vorbetrachtungen	463
7.2.2	Unterschiede zu Linux	464
7.2.3	Windows Server-Container vs. Hyper-V-Container	464
7.2.4	Linux Container unter Windows Server mit Hyper-V	465
7.2.5	Weiterführende Informationen/Setup-Anleitungen	466
7.3	Rocket Science? – CoreOS/Container Linux	466
7.3.1	CoreOS/Container Linux und Rocket	467
7.3.2	CoreOS: nicht vorhandene Pakete und Rolling Upgrades	467
7.3.3	Upgrades	467
7.3.4	rkt und das Appc-Containerformat	470
7.3.5	Die Unterschiede zwischen Docker- und Rocket-Containern	471
7.3.6	Relativer, echter Prozessbaum und Paradigmen	471
7.3.7	Setup einer CoreOS-Instanz als VM	472
7.3.8	Erste Einstellungen und ssh-Connect	473
7.3.9	CoreOS-Anpassungen	474
7.3.10	Nach dem Start	475
7.3.11	Rocket Launch-Container starten und verwalten	475
7.4	SLE MicroOS	479
7.5	RancherOS	479

8	Fazit – Single Node Container-Plattformen	481
8.1	Der Wandel	481
8.2	»Die« Container-Plattform?	481

TEIL III Skalierbare Container Cluster und Container-Orchestrierung

9	Container Cluster – von Planern und Orchestern	485
9.1	Worum es geht – the Big Picture	485
9.2	World of Tiers – Teil 2	486
9.2.1	Scheduling vs. Orchestration?	486
9.2.2	Die Layer/Tiers und ihr Zusammenwirken	486
9.2.3	Container Cluster	487
9.2.4	Unser Ziel	488
9.3	Vorbereitungen	488
9.3.1	Distributionsfragen	488
9.4	Pre-Flight Requirements: Zeitsynchronisation	489
9.4.1	NTP und die Relativität	489
9.4.2	NTP-Basics	490
9.4.3	NTP-Setup	491
9.4.4	NTP-Setup mit zusätzlichen Peers	494
9.4.5	Chrony	495
9.5	Pre-Flight Requirements: pssh	495
9.5.1	Grundsätzliches	495
9.5.2	Setup pssh auf allen Nodes	496
9.5.3	pssh – korrespondierende Dateien/Einstellungen	497

10 Schlüsselmeister im Container Cluster: Key/Value Stores und Service Registry/Discovery 499

10.1	Key/Value Stores	500
10.1.1	Vorbetrachtungen	500
10.1.2	Key/Value Stores im Detail	500

10.1.3	Backup- und Verfügbarkeitsstrategien	501
10.2	Service Discovery/Registry	502
10.2.1	Sichten – und nicht vernichten	502
10.2.2	Service Discovery im Detail	503
10.2.3	Statisch vs. dynamisch	505
10.2.4	Konfigurationsreplikation	505
10.3	Verfügbare Key Value/Stores im Kurzüberblick	506
10.3.1	Teile und herrsche	506
10.3.2	Entscheidungsfindung – Raft vs. Paxos	506
10.3.3	etcd	507
10.3.4	Consul	508
10.3.5	Zookeeper	509
10.4	Key/Value Store Cluster für Container am Beispiel von Consul	510
10.4.1	Die Consul-Komponenten in der Übersicht	510
10.4.2	Zusammenfassung der Consul-Kernfunktionen und Features	512
10.4.3	Zusammenfassung der Consul-Basisarchitektur und Funktionsbeschreibung	513
10.4.4	Consul Service Monitoring im Detail	514
10.4.5	Setup eines Consul Clusters: Vorbetrachtungen	515
10.4.6	Einfache Consul-Grundkonfiguration	516
10.4.7	Consul als generischer Key/Value Store für Container Cluster	520
10.4.8	Endpunkte/Endpoints	522
10.4.9	Cluster Leader (Re-)Election	524

11 Schwarm...-Intelligenz? Docker Swarm Mode 525

11.1	Docker Swarm Mode, Swarmkit, old/native Swarm ... oder noch einen Schwarm vergessen?	525
11.1.1	Docker (native) Swarm	526
11.1.2	Swarmkit	526
11.1.3	Docker Swarm Mode	527
11.1.4	Grundsätzliches zum Swarm Mode	527
11.1.5	Docker Swarm Mode mit integriertem KV Store und TLS	527
11.1.6	Full Rework: Docker Swarm und TLS ab 1.12	528
11.2	Swarm – Pre-Flight-Betrachtungen	529
11.2.1	Vorbetrachtungen	529
11.2.2	Best Practice: Anzahl Swarm Manager Nodes/Fault Tolerance	530
11.2.3	RZ-Topologien	531

11.2.4	Grundsätzliches zum (Swarm-)Netz	532
11.2.5	Swarm und IPVS	532
11.2.6	Container-HA und Scale-Out	533
11.2.7	Overlay-Netzwerke im Swarm Mode	533
11.2.8	Multiple Swarm-Netze und Verschlüsselung	534
11.2.9	Swarm Services – Key Concepts	535
11.3	Swarm-Setup	536
11.3.1	Vorbetrachtungen	536
11.3.2	Überblick über die Swarm-Kommandos	536
11.3.3	Swarm Mode-HA	539
11.3.4	Anlegen eines neuen Swarm-Netztes (Typ Overlay)	539
11.4	Swarm-Administration	541
11.4.1	Weitere Swarm Nodes hinzufügen oder entfernen	541
11.4.2	Node Status inspizieren	542
11.4.3	Weitere Schwarm-Informationen	544
11.4.4	Beförderung und Degradierung	544
11.5	Swarm Services	546
11.5.1	Vorbetrachtung	546
11.5.2	Swarm Service, interner DNS und Service Discovery	546
11.5.3	Swarm und internes Loadbalancing	548
11.5.4	Swarm Mode Routing Mesh	548
11.5.5	Externes Loadbalancing/HTTP Routing Mesh (HRM)	549
11.5.6	Einfaches Swarm Service-Beispiel	550
11.5.7	Storage	556
11.5.8	Swarm Container IP(s)	557
11.5.9	Swarm Service-IPs: dnsrr vs. vip	558
11.5.10	Service Discovery-Beispiele: vip vs. dnsrr	559
11.5.11	Global vs. Replicated Services	561
11.5.12	Scale-Out	561
11.5.13	Service-Verfügbarkeit /Healthchecks	562
11.6	Rolling Updates im Schwarm	565
11.6.1	Vorbetrachtungen	565
11.6.2	Beispiel	565
11.6.3	Rollbacks	567
11.6.4	Leichen im Keller	567
11.6.5	Kontrolle des Rolling Updates	568
11.6.6	Interner Ablauf des Rolling Upgrades	569
11.7	Swarm Stacks	569
11.7.1	Vorbetrachtungen	569
11.7.2	Beispiel	570

11.8	Label- und Constraint-based Placement	573
11.8.1	Vorbetrachtungen	573
11.8.2	Warum Label?	574
11.8.3	Constraints	574
11.8.4	Swarm Node Labels	578
11.8.5	Platzierung über Swarm-Label (nicht dockerd)	579
11.8.6	Service (Placement) updaten	580
11.8.7	Der lange leidige Weg der (guten) alten Constraints	581
11.9	Swarm: weitere zu beachtende Punkte	585
11.9.1	Backup Swarm State/Disaster Recovery	585
11.9.2	Backup und Disaster Recovery	585
11.9.3	Rolling Upgrade der Swarm Nodes im Produktivumfeld	585
11.9.4	Zusammenfassung: die wichtigsten Swarm Mode Best Practices	586
11.9.5	Wohin geht der Weg für Docker Swarm?	586

12 Docker Datacenter, Docker Trusted Registry und CaaS 587

12.1	DDC, UCP und DTR im Überblick	587
12.1.1	Vorbetrachtungen	587
12.1.2	Microservice-Architektur im DCC	588
12.2	UCP – Universal Control Plane	589
12.2.1	Generelle Vorbetrachtungen	589
12.3	UCP-Installation, Setup und weiterer DDC Nodes	590
12.3.1	Vorbetrachtungen zum Setup	590
12.3.2	Installations-/Konfigurationsoptionen des UCP	591
12.3.3	UCP-Setup	593
12.3.4	UCP Login	595
12.3.5	Die UCP-Einzelkomponenten im Detail	596
12.3.6	Beim Setup erzeugte/verwendete Volumes	597
12.3.7	Weitere UCP Nodes zum DDC hinzufügen	598
12.3.8	UCP Dashboard im Kurzüberblick	600
12.3.9	UCP HA	602
12.3.10	UCP Backup und Restore	602
12.3.11	UCP LDAP-Authentifizierung	603
12.4	DTR – Docker Trusted Registry	605
12.4.1	Vorbetrachtungen	605
12.4.2	DTR HA Image Storage	605

12.5	DTR-Installation	606
12.5.1	Vorbetrachtungen	606
12.5.2	DTR-Hilfeoptionen	607
12.5.3	Die DTR-Container	608
12.5.4	DTR-Netzwerke	608
12.5.5	Vom DTR erzeugte und verwendete Volumes	608
12.5.6	Hilfeoptionen zur Installation	609
12.5.7	DTR-Applikation im DDC	611
12.5.8	DTR Management	612
12.5.9	DTR Backups und Restore	613
12.5.10	TLS-Setup DTR und Client	614
12.5.11	Push/Pull in die/aus der DTR	615
12.6	Notary/TUF im DDC	616
12.6.1	Vorbereitungen	616
12.6.2	Test des Notary Clients	617
12.6.3	Target Keys und Snapshot Keys	619
12.6.4	Repo-Test	619
12.7	Image Security in der DTR	621
12.8	DDC – Sonstiges	622
12.8.1	DDC: Preise und Service-Modelle	622
12.8.2	Fazit	622
13	Kubernetes (K8s)	625
13.1	Kubernetes (K8s) im Überblick	625
13.1.1	Vom Borg zum Steuermann	625
13.1.2	dockerd, rkt, CRI-O? – K8s-Container-Unterbau, KISS und die Zukunft	627
13.1.3	K8s, Key/Value Backends und SPoFs	629
13.2	K8s-Komponenten	630
13.2.1	Komponenten des K8s Clusters	630
13.2.2	K8s-Dienste auf den Master Nodes	631
13.2.3	K8s-Dienste auf den Workern	632
13.3	Networking in Kubernetes	634
13.3.1	Vorbetrachtungen	634
13.3.2	Unterschiede zu Docker	635
13.3.3	CNI-Plugins	636
13.3.4	Netzwerkkommunikation im K8s Cluster	636

13.4	etcd: Key/Value Store für Kubernetes im Detail	637
13.4.1	Grundsätzliches	637
13.4.2	Arbeitsweise/Funktionsprinzip von etcd im Zusammenspiel mit K8s	637
13.4.3	K8s und Redundanzen	638
13.4.4	etcd-Datenstrukturen im Detail	639
13.4.5	etcd: Hot-Backup	641
13.5	etcd-Cluster – Installation und Setup	642
13.5.1	Vorbetrachtungen	642
13.5.2	Minikube? Nö.	643
13.5.3	Vorbetrachtungen und Vorbereitungen	644
13.5.4	etcd-Konfiguration	645
13.5.5	Manueller etcd-Cluster-Start	649
13.5.6	etcd-Integration in systemd/lokale HA	651
13.5.7	etcd-Start via systemd-Unit	652
13.5.8	Failover-Test	654
13.6	Flannel CNI für K8s Cluster	655
13.6.1	Vorbetrachtungen	655
13.6.2	K8s Cluster und Overlay-Networking	655
13.6.3	Network Extensions mit Flannel	656
13.6.4	Flannel Dependencies	657
13.6.5	Flannel-Setup auf den drei Nodes	658
13.6.6	Exkurs: multiple Flannel Overlay-Netzwerke	664
13.7	K8s Single Master-Setup	664
13.7.1	Vorbetrachtungen: Apiserver, Controller Manager und Scheduler	664
13.7.2	Setup kube-apiserver	665
13.7.3	Setup kube-controller-manager	670
13.7.4	Setup kube-scheduler	671
13.7.5	Scheduler-Algorithmen: Predicates und Priorities	671
13.8	K8s-Worker-Konfigurationen für Single Master	673
13.8.1	Vorbetrachtungen	673
13.8.2	Der Node Controller (kube-controller-manager) und die Verfügbarkeit der Worker Nodes	674
13.8.3	kubelet-Konfiguration	675
13.8.4	kube-proxy-Konfiguration	676
13.8.5	Sonstiges: zentrale Konfigurationsparameter	677
13.8.6	kubectrl	677
13.8.7	kubectrl-Bash-Completion	677
13.8.8	Start der Master- und aller Worker-Dienste	678

13.9 Redundante und hochverfügbare K8s Master:	
Konzepte und Möglichkeiten	680
13.9.1 Vorbetrachtungen: Ist-/Soll-Stand	681
13.9.2 Vorbetrachtungen: mögliche Vorgehensweisen	681
13.9.3 Pre-Flight Requirements und konkrete Setup- Vorbetrachtungen	682
13.9.4 Das Multi-Active-Problem	684
13.9.5 Zertifikatsanpassungen/Cluster-Aliase	685
13.9.6 Failover-Varianten	685
13.10 Vorbereitendes Setup der K8s Master für Hot-Failover	686
13.10.1 Hot-Failover K8s Master	686
13.10.2 Vorbereitende Konfiguration	687
13.10.3 Kubectrl-Zertifikatsfehlermeldungen	689
13.11 Pacemaker-Integration der drei K8s Master	689
13.11.1 Vorbetrachtungen – HA der K8s-Kernkomponenten mit Pacemaker	689
13.11.2 Pacemaker und Corosync	690
13.11.3 Pacemaker: Agenten, Ressourcen und Constraints im Kurzüberblick	692
13.11.4 Pakete und vorbereitende Tasks	693
13.11.5 Corosync-Setup	694
13.11.6 Benötigte Grundeinstellungen	697
13.11.7 Erzeugung der benötigten Pacemaker-Ressourcen für den K8s Cluster	699
13.11.8 Erzeugen der weiteren K8s-Master-Ressourcen	700
13.11.9 Gruppieren der kube*-Ressourcen	701
13.11.10 Klonen der Gruppe	701
13.11.11 IP-Colocation	702
13.11.12 etcd und flanneld	702
13.11.13 Inbetriebnahme des Pacemaker-gesteuerten K8s Master Clusters	704
13.11.14 Failover-Test	705
13.11.15 Grafisches Pacemaker Cluster-Management per HAWK2	706
13.11.16 Grafisches Pacemaker Cluster-Management per PCS-GUI	708
13.12 Stonith	709
13.12.1 Ein kurzer Überblick	709
13.12.2 vSphere/vCenter	710
13.12.3 SBD	710
13.12.4 Andere Stonith Varianten	710
13.13 Grundlegende Management-Tasks und Regeln für Pacemaker Cluster	711

14 Verstehen und Verwalten von Ressourcen im K8s Cluster

713

14.1	Vorbereitungen	713
14.2	kubectl	713
14.2.1	kubectl-Bash-Completion	713
14.2.2	Das kubectl-Kommando	714
14.2.3	kubectl-Konfiguration	714
14.2.4	Die wichtigsten kubectl-Subkommandos in der Übersicht	714
14.2.5	Die kubectl-Manpages	716
14.2.6	Welche Ressourcen/Aktionen können im K8s Cluster via kubectl-CLI verwaltet werden?	716
14.2.7	API-Versionierungsdickicht	717
14.2.8	Grundsätzliche Hilfe zu den K8s-Ressourcen/Aktionen	719
14.2.9	K8s-Ressourcen direkt mit kubectl run deployen	720
14.2.10	Der kubectl-run --restart=-Schalter und seine Auswirkung auf erzeugte Objekte	720
14.2.11	Einfache kubectl-Beispiele im Hinblick auf erzeugte API-Objekte	721
14.2.12	Grundlegende K8s-Cluster-Informationen abfragen	723
14.3	Kompose – Docker Compose-Konverter für K8s	724
14.3.1	Vorbetrachtung	724
14.3.2	Beispiele	724
14.4	Kleine K8s Cluster und Taint Nodes	725
14.4.1	Vorbetrachtung	725
14.4.2	Beispiele	725
14.5	(Worker-)Node-Kapazitäten	726
14.5.1	Vorbetrachtungen	726
14.5.2	Analyse	726
14.5.3	Weitere Node-Informationen abfragen	728
14.6	Ressourcen im K8s Cluster ausrollen	730
14.6.1	Was passiert beim Ausrollen einer Ressource?	730
14.6.2	Default-Verteilungsstrategien des Scheduler für die Worker Nodes	731
14.6.3	Unterschiedliche Ressourcen in einem Yaml-File	732
14.7	Pods	732
14.7.1	Vorbetrachtungen	732
14.7.2	Die Images hinter den Pods	735

14.7.3	K8s Image Pull und (lokale) Trusted Registries	735
14.7.4	K8s-Besonderheit: das »pause«-Image und Reservierungen	736
14.7.5	Erstellen eines Pods	737
14.7.6	Laufenden Pod/laufende Resource editieren	739
14.7.7	Kubernetes Pod-Phasen und -Zustände (Status)	741
14.7.8	Ein paar konkrete Beispiele für Zustände der Pods	742
14.7.9	Pods nach Namespaces	743
14.7.10	Debugging mit kubectl describe	744
14.7.11	K8s Pods aus der Docker-Sicht	745
14.7.12	(Force) Removal eines Pods	746
14.7.13	Pod mit unterschiedlichen Containern/Images	746
14.7.14	Setzen von Kommandos in der Pod-Spezifikation per command/args	747
14.7.15	Logs	748
14.7.16	Detailliertes Auslesen von Pods	749
14.7.17	Restart Policies	750
14.7.18	Pods und Node Bindings	750
14.7.19	Kommandos im Pod/Container ausführen	750
14.7.20	Pod Health Checks	751
14.7.21	Pod Lifecycle-Management/Hooks	756
14.7.22	Automatische Bereinigung alter Pods	758
14.7.23	Pod-Ressourcen und Limitierungen	759
14.7.24	PodDisruptionBudget	761
14.8	Pods und Volumes	761
14.8.1	Vorbetrachtungen	761
14.8.2	K8s Volumes im Unterschied zu Docker	762
14.8.3	Persistente und nicht persistente Volumes	763
14.8.4	emptyDir	765
14.8.5	hostPath	766
14.8.6	nfs	767
14.8.7	iscsi-Volume	769
14.8.8	Ceph	770
14.8.9	PersistentVolumes/SAN	770
14.8.10	PersistentVolumes in der Praxis	771
14.9	Pods und ConfigMaps	780
14.9.1	Vorbetrachtungen	780
14.9.2	ConfigMaps in der Praxis	780
14.10	Scale-Out: von Replication Controllern zu ReplicaSets	783
14.10.1	Vorbetrachtungen	783
14.10.2	Beispiel für einen Replication Controller (rc)	785

14.10.3	Scale-Out	788
14.10.4	Löschung alter Pods in rc	789
14.10.5	Rolling Updates von rc	789
14.10.6	Multiple Release Tracks für rc	790
14.10.7	ReplicaSets und Set-based Label Selectors	791
14.11	Deployments	794
14.11.1	Verkapselung	795
14.11.2	Erstellen eines Deployments	795
14.11.3	Deployment per kubectl run erzeugen	797
14.11.4	Deployment mit Service erzeugen	798
14.11.5	Rolling Update/Rollout/Revisionssicherheit	800
14.11.6	Update-/Revisionshistorie	800
14.11.7	Update-Strategien	802
14.11.8	Resume/Pause	802
14.11.9	Absichtlich erzeugter Fehler beim Ausrollen/Auto-Stop des Rollouts	802
14.11.10	Rollback-Verfahren	803
14.12	Namespaces	804
14.12.1	Vorbetrachtungen	804
14.12.2	Namens- und Designfragen	805
14.12.3	Praktischer Einsatz	806
14.12.4	Ressourcen im Namespace erzeugen	808
14.12.5	Limitierte Pods und Namespaces	810
14.12.6	Limits für Namespaces in der Praxis	810
14.12.7	Node- und Pod-spezifische Statistiken	813
14.12.8	Namespaces und Ressource-Quotas	813
14.13	Services	817
14.13.1	Vorbetrachtungen	817
14.13.2	K8s-DNS-Integration	820
14.13.3	Vorbereitungen: Cluster-IP, NodePort, LoadBalancer, --cluster-cidr und mehr	820
14.13.4	Services und kube-proxy	820
14.13.5	kubelet und Services	823
14.13.6	ServiceType	823
14.13.7	Beispiel: Service-Bereitstellung via NodePort	824
14.13.8	Headless Service	829
14.13.9	Ingress und Alternativen	831
14.13.10	Multi-Port-Services mit NodePort und externer IP	833
14.13.11	Service/Deployment-Beispiel: K8s-Dashboard	834

14.13.12	Service/Deployment-Beispiel: kube-dns als Cluster-DNS-Service einrichten	841
14.14	PetSets/StatefulSets/Stateful Pods	849
14.14.1	Vorbetrachtungen	849
14.14.2	Im Detail	849
14.15	Jobs	851
14.15.1	Vorbetrachtungen	851
14.15.2	Jobs-Beispiel	852
14.16	DaemonSets	855
14.16.1	Vorbetrachtungen	855
14.16.2	Mögliche Anwendungsfälle	856
14.16.3	DaemonSet Scheduler und Node-Zuordnung	856
14.16.4	Beispiele	857
14.17	HPA – Horizontaler Pod Autoscaler	860
14.17.1	Vorbetrachtungen	860
14.17.2	Hintergrund	860
14.17.3	Auslastungskontrolle	861
14.17.4	HPA-Setup	861
14.17.5	Last-Test	865
14.17.6	Löschen des hpa-Objekts:	866
14.18	Weitere K8s-Objekte und -Ressourcen	866
14.18.1	Events	866
14.18.2	ThirdPartyResource	867
14.19	K8s-Labels und -Constraints	868
14.19.1	Vorbetrachtungen	868
14.19.2	Label-Meeting	869
14.19.3	Aufbau	869
14.19.4	Ein paar praktische Beispiele	870
14.19.5	Labels und Node-/Pod-Affinity	872
14.19.6	Affinity und Anti-Affinity	874
14.20	K8s-Authentifizierung und -Autorisierung	875
14.20.1	Vorbetrachtungen	875
14.20.2	Accounts	876
14.20.3	Umsetzung unter K8s	877
14.20.4	Secrets	878
14.20.5	User Account in einem Kontext anlegen	881
14.20.6	Authentifizierung an weiteren Clustern	883
14.20.7	Dex	883

15 The Road Ahead: Kubernetes ab Version 1.5/1.6 885

15.1 K8s als rein Pod-basiertes System	885
15.1.1 Vorbetrachtungen	885
15.2 Setup	886
15.2.1 Setup-Voraussetzungen	886
15.2.2 Repositories und Installation	887
15.2.3 Erzeugte Files	889
15.2.4 Verfügbare kubeadm init-Direktiven	890
15.2.5 K8s-Initialisierung mit kubeadm init	890
15.2.6 etcd-Sidecar und K8s	892
15.2.7 Under the Hood – was ist beim Setup passiert?	892
15.2.8 Weave	896
15.2.9 Flannel-CNI als Pod/Deployment	897
15.2.10 Join weiterer K8s Nodes	898
15.2.11 Verfügbarkeit/Ausfallsicherheit der K8s-Komponenten im Pod-Modell	900
15.2.12 Redundanter, realer etcd-Cluster für K8s im Container-Setup	901
15.3 etcd-Pod-Cluster out-of-the-box?	902
15.3.1 Vorbetrachtungen	902
15.3.2 Funktionalitäten	903
15.3.3 Praxisbeispiel	904
15.3.4 Henne-Ei-Problem?	905
15.3.5 Prometheus	906
15.4 K8s Multimaster und native HA	906
15.4.1 Vorbetrachtungen	906
15.4.2 Die Tasks im Detail	907

16 K8s GUIs/Monitoring: Cockpit, Dashboard und mehr 909

16.1 Cockpit	910
16.1.1 Vorbetrachtungen	910
16.1.2 Setup	910
16.1.3 Kurzübersicht der verfügbaren K8s-Module	911
16.2 K8s-Dashboard	912
16.2.1 Vorbetrachtungen	912
16.2.2 Administration mit dem K8s-Dashboard	913

16.3	fabric8	914
16.3.1	Vorbetrachtungen	914
16.3.2	Setup und Betrieb	915
16.4	Weave Scope	916
16.4.1	Vorbetrachtungen	916
16.4.2	Setup	916

17 Federated/Geografisch verteilte K8s Cluster 919

17.1	Vorbetrachtungen	919
17.1.1	Wie funktioniert es?	919
17.1.2	Federated Services	921
17.1.3	Federated Controller Manager und Scheduling	921
17.1.4	RZ-Failures	922
17.1.5	Setup und andere Kopfschmerzen	922

18 K8s: Debugging, Rolling Upgrades, Fazit 925

18.1	Debugging/Troubleshooting	925
18.1.1	Grundsätzliches	925
18.1.2	Projektspezifische Debugging-Ressourcen	925
18.2	Rolling Upgrades des K8s Clusters	926
18.2.1	Vorbetrachtungen	926
18.2.2	Best Practices: Master Nodes	926
18.2.3	Best Practices: alle Nodes und kubelets	927
18.2.4	kube-proxy	928

TEIL IV Übergreifende Orchestrierungstools für verteilte Container-Infrastrukturen

19 Rancher 931

19.1	Design und Aufgabenbereiche	932
19.1.1	Infrastruktur-Orchestrierung	932
19.1.2	Container-Orchestrierung und -Scheduling	932

19.1.3	Anwendungen	933
19.1.4	Authentifizierung	933
19.1.5	Alles easy? Teilweise	933
19.2	Setup	933
19.2.1	Setup Vorbetrachtungen	933
19.2.2	Rancher-Server/Web-UI	934
19.2.3	Authentifizierungseinstellungen	935
19.2.4	Hosts hinzufügen	937
19.2.5	Rancher CLI	940
19.2.6	Natives Container-Management	940
19.2.7	Rancher und Swarmkit Cluster	942
19.3	Rancher und K8s	943
19.3.1	Vorbetrachtung	943
19.3.2	Setup	943
19.4	Fazit	944

20 Ab in die Mesosphäre: DC/OS 945

20.1	DC/OS	945
20.1.1	Apache Mesos und DC/OS	945
20.1.2	Mesospheres' DC/OS	946
20.1.3	Architektur	947
20.1.4	Generelle HA-Anforderungen	948
20.2	Dienste auf den Mesos Nodes	949
20.2.1	Dienste auf den Master Nodes	949
20.2.2	Marathon	950
20.2.3	Mesos-DNS	950
20.2.4	Zookeeper	951
20.2.5	Dienste auf den Agent Nodes	953
20.2.6	User Space-Prozesse	953
20.2.7	DC/OS-Zonenkonzept	953
20.2.8	Loadbalancing	954
20.2.9	Applikationsanforderung im DC/OS Cluster	955
20.2.10	DC/OS Containerizer/Container-Runtime	956
20.2.11	Netzwerke unter DC/OS	958
20.2.12	Zusätzliche Overlay-Netze	959
20.3	Setup eines DC/OS Clusters	960
20.3.1	Setup-Vorbetrachtungen	960

20.3.2	Nodes	961
20.3.3	System-Requirements und Setup-Schritte	962
20.3.4	Weitere Pakete, IPs, Updates und Docker Storage Backend	963
20.3.5	Zusätzliche Overlay-Netze	965
20.3.6	Disk-Ressourcen auf den Agent Nodes	966
20.4	Setup-Vorbereitungen	966
20.4.1	DC/OS Installer	967
20.4.2	Benötigte Files für CLI-basierte Installation	967
20.5	Setup	969
20.5.1	Check der (Pre-)Requirements	970
20.5.2	GUI-Setup	971
20.5.3	Telemetriedaten	973
20.5.4	CLI-Setup	973
20.5.5	Pre-Flight Checks	976
20.5.6	Deployment	977
20.5.7	Erzeugte/verwendete Ordner und Verzeichnisse	978
20.5.8	Post-Flight Checks	979
20.5.9	Login und DC/OS Dashboard	979
20.5.10	Post-Install-Troubleshooting	981
20.5.11	Hinzufügen weiterer Nodes	983
20.6	DCOS EE – Authentifizierung und Security	984
20.6.1	ID-Provider	985
20.6.2	External Directory (LDAP)	985
20.6.3	Ablauf einer Session	986
20.7	DC/OS-CLI	986
20.7.1	DC/OS-CLI (offene Version)	986
20.7.2	DC/OS-CLI (EE)	988
20.7.3	Nodes und Komponenten per CLI anzeigen lassen	989
20.7.4	dcos via SSH	989
20.8	DC/OS-Ressourcen-Terminologien und -Konzepte	990
20.8.1	Application/Service	990
20.8.2	Ressourcen-Requests	994
20.8.3	Jobs	996
20.8.4	Pod	997
20.8.5	Group	998
20.8.6	Stateful Services und DC/OS Volumes	999
20.8.7	MySQL-(und Wordpress-)Gruppe mit persistenter Datenablage	1000
20.8.8	Deployment vs. Application	1004

20.8.9	Dependencies	1005
20.8.10	Labels	1006
20.8.11	Service Upgrades/Rolling Upgrades	1006
20.8.12	Constraints	1007
20.8.13	Healthchecks	1008
20.8.14	Autoscaling	1009
20.8.15	Universe-Repository und Ausrollen von DC/OS-Applikationen	1010
20.8.16	Applikationsgruppen und Dependencies	1010
20.8.17	Dependencies abbilden	1012
20.9	Fazit	1013

21 Fazit Container-Orchestrierung 1015

21.1	Die Kandidaten im Überblick	1015
21.2	Docker Swarm/DDC/EE	1015
21.3	Rancher	1016
21.4	Kubernetes (K8s)	1016
21.5	DC/OS	1017

TEIL V Software Defined Storage für verteilte Container-Infrastrukturen

22 Ab in den Untergrund 1021

22.1	SDS und Container	1021
22.1.1	Vorbetrachtungen	1021
22.1.2	Cluster Storage, Skalierbarkeit und der SPoF	1022
22.2	SDS-Funktionsprinzipien	1023
22.2.1	Software Defined Storage – SDS	1023
22.2.2	Hoch? Oder lieber breit? Scale-Out anstelle von Scale-Up	1023
22.2.3	Traditionelle Storage Cluster(-FS) vs. SDS	1024
22.2.4	Die Abstraktion	1025
22.2.5	SDS und Storage Tiers	1025
22.2.6	Multi-Purpose-SDS	1027
22.2.7	Die roten Hüte, Ceph, Gluster und die Zukunft	1027

22.3 Ceph	1028
22.3.1 Vorbetrachtungen	1028
22.4 Ceph und RADOS	1029
22.4.1 Vorbetrachtungen	1029
22.4.2 RADOS und Ceph SDS-Bereitstellung	1030
22.4.3 librados	1030
22.4.4 Crush(maps)	1031
22.5 Die Ceph-Daemons im Kurzüberblick: MON, OSD, MDS	1032
22.5.1 Vorbetrachtung	1032
22.5.2 OSD	1032
22.5.3 MON	1033
22.5.4 MDS	1034
22.6 Ceph-Bereitstellungsverfahren für Container Cluster	1035
22.6.1 RADOS Block Device	1035
22.6.2 CephFS	1035
22.7 Setup des Ceph Clusters für Container-Storage-Bereitstellung	1036
22.7.1 Vorbetrachtungen	1036
22.7.2 Vorbereitungen zum Setup	1036
22.7.3 Erzeugen der neuen Ceph-Cluster-Konfiguration	1037
22.7.4 Deployment der Monitore	1038
22.7.5 Deployment der OSDs	1039
22.7.6 Kapazitätsbetrachtungen	1040
22.7.7 Deployment der MDS, PG-Kalkulation	1040
22.8 Ceph als SDS für K8s (RBD)	1041
22.8.1 Vorbetrachtungen	1041
22.8.2 Persistent Volume (Claims) mit Ceph-RBD	1042
22.8.3 Client-Integration (K8s Worker Node)	1043
22.8.4 Die wichtigsten Schritte in der Übersicht	1047
22.9 Ceph als SDS für K8s (CephFS)	1048
22.9.1 Vorbetrachtungen	1048
22.9.2 Vorbereitung (Image)	1048
22.9.3 Vorbereitung (CephFS)	1048
22.9.4 Vorbereitung K8s und Setup	1049
22.10 Ceph als SDS für Docker	1050
22.11 Ceph als SDS für DC/OS	1051
22.12 Grundlegende Ceph Cluster-Designregeln für Produktivumgebungen	1051
22.12.1 Netzwerkredundanzen	1052
22.12.2 Backup der Cluster-Konfiguration	1052

22.12.3	Full Data Backup?	1052
22.12.4	Rolling Upgrades	1053
23	Was war, was ist, was sein wird	1055
23.1	Neue Meta-Ebenen und Verkomplizierung	1055
23.1.1	Container Cluster und Microservices als Allheilmittel?	1055
23.1.2	The Road Ahead	1056
Index		1057