

# Inhalt

## TEIL I Brave New World?

<b>1 Die neue alte Welt der Virtualisierung</b>	<b>41</b>
<b>1.1 V2 .....</b>	41
<b>1.2 Vorbemerkungen .....</b>	46
1.2.1 Verwendete Formatierungen .....	46
1.2.2 Weiterführende Hinweise .....	47
1.2.3 Beispieldateien .....	47
<b>1.3 Was dieses Buch sein/nicht sein soll .....</b>	47
1.3.1 Was es sein soll .....	47
1.3.2 Was es nicht sein soll und nicht ist .....	47
<b>1.4 Wie dieses Buch zu lesen ist .....</b>	48
1.4.1 Grundsätzliches .....	48
1.4.2 Kapitel/Teile und Zielgruppen im groben Überblick .....	48
<b>1.5 Welche Teile des Buchs sind neu, welche wurden stark überarbeitet? .....</b>	48
<b>1.6 In welchen Kapiteln finde ich welche Themen? .....</b>	50
1.6.1 Themen nach Abschnitten/Schlagworten mit grundlegenden Erklärungen .....	50
<b>1.7 Verwendete Plattformen und Versionsspezifikationen .....</b>	53
1.7.1 Vor betrachtungen .....	53
1.7.2 Container-OS und die Zukunft .....	54
<b>2 Container</b>	<b>55</b>
<b>2.1 Warum Container? .....</b>	55
<b>2.2 Microservices, Container und der pawlowsche Hund .....</b>	57
2.2.1 Wie erkläre ich es meinem CEO? .....	57
2.2.2 Die neue Welt der Microservices: Admins, DevOps- und Container-Teams ....	59
2.2.3 Die neue Welt der Microservices: aus der Perspektive der CEOs/Entscheider .....	61

<b>2.3</b>	<b>Continuous Delivery/Continuous Integration und DevOps .....</b>	62
2.3.1	Semi- oder vollautomatisch: Continuous Integration/Continuous Delivery .....	62
2.3.2	CD/CI und das Big-Bang-Release-Problem .....	63
<b>2.4</b>	<b>Continuous Delivery .....</b>	64
2.4.1	Was verstehen wir darunter? .....	64
2.4.2	Continuous Delivery Pipelines .....	66
2.4.3	Commit-Stage .....	67
2.4.4	Acceptance-Test-Stage .....	67
2.4.5	Exkurs: Acceptance-Tests und Dreieinigkeit .....	68
2.4.6	Load/Capacity-, Security- und Exploration-Tests .....	68
2.4.7	Rollout/Go-Live .....	69
2.4.8	Die Gates .....	69
2.4.9	Fazit: Wo kann CD nutzbringend eingesetzt werden? .....	69
2.4.10	Jenkins-Integration in Kubernetes .....	70
<b>2.5</b>	<b>DevOps: Gewaltenteilung oder Kooperation? .....</b>	71
2.5.1	Grundsätzliche Betrachtungen .....	71
2.5.2	Vom DevOps-Paradigma/Hype zum Unwort .....	71
2.5.3	Kommunikationsblackouts im DevOps-Team .....	72
2.5.4	Das konkrete DevOps-Problem im klassischen Umfeld .....	72
2.5.5	»Works for me« und anderer Nonsense im DevOps-Business – und ein Ausweg? .....	73
2.5.6	BizDevOps – und noch eine Silbe .....	75

## TEIL II Single-Node Container-Systeme

<b>3</b>	<b>Container-Engines und -Plattformen, Basics und Konzepte .....</b>	79
<b>3.1</b>	<b>World of Tiers – Teil 1 .....</b>	79
<b>3.2</b>	<b>Container – Basics .....</b>	80
3.2.1	Namespaces, Security und Container-Konzepte .....	80
3.2.2	Enter Namespace – nsenter .....	82
3.2.3	Namespaces und Sicherheit? .....	84
<b>3.3</b>	<b>VMs – obsolet durch Container? .....</b>	88
3.3.1	Container vs. VM .....	89
3.3.2	Packungsdichte und Ressourcen .....	91

<b>3.4 Wann sind Container sinnvoll? .....</b>	94
<b>3.5 Container-Engines/Container-Runtime und Komponenten im Überblick .....</b>	94
3.5.1 Von LXC zu Docker .....	95
3.5.2 Docker .....	95
3.5.3 libcontainer, containerd und runC .....	96
3.5.4 runC, containerd und shim im Docker-Kontext .....	97
3.5.5 runC als eigenständiges Container-Tool .....	100
3.5.6 (cri-)containerd – Wirrwarr im Kubernetes-Kontext .....	103
3.5.7 LXD .....	105
3.5.8 CoreOS bzw. Container-Linux und Rocket/rkt .....	107
3.5.9 CRI-O .....	108
<b>3.6 Überblick der Container-Formate .....</b>	108
3.6.1 OCI – ... Standardisierung? .....	109
3.6.2 Runtime Specs .....	109
3.6.3 Image-Format .....	110
3.6.4 Das BSI und die lieben Container .....	112
3.6.5 Fazit .....	113
<b>3.7 Container: eine funktionale Übersicht .....</b>	113
3.7.1 Aufbau eines Container-Hosts .....	114
3.7.2 Docker-Images .....	114
3.7.3 Anzahl der Layer .....	115

## 4 Docker 117

---

<b>4.1 Docker-Versionen .....</b>	117
4.1.1 Docker-Versionen, wichtige Meilensteine und Inkompatibilitäten .....	117
4.1.2 Docker-LTS-Versionen und die Enterprise Edition .....	118
4.1.3 DDC, CE, EEB, EES, EEA – und der beständige Wechsel im Docker-Land .....	119
4.1.4 Betrachtete Plattformen und Docker-Versionen .....	122
4.1.5 Funktionaler Überblick: Docker CLI, dockerd, Registry .....	122
<b>4.2 Docker-Installation .....</b>	123
4.2.1 Paketnamen und Dependencies .....	123
4.2.2 Docker-Installation unter Ubuntu 18.04 LTS .....	124
4.2.3 Docker-Installation unter RHEL/CentOS ≥ 7.4 .....	124
4.2.4 Docker-Version unter CaaSP .....	126
4.2.5 CaaSP- und BTRFS/Docker-Problematiken .....	126
4.2.6 CaaSP-spezifische Docker-Konfigurationsdateien .....	127
4.2.7 Installation der Commercially Supported Docker-Engine .....	127

4.2.8	Docker CE, EE .....	127
4.2.9	Storage-Driver-Nonsense unter Docker CE und EE .....	130
4.2.10	Distributionsunabhängige Installation von Docker .....	131
<b>4.3</b>	<b>Deinstallation, Upgrade oder Umstellung auf andere Storage-Backends .....</b>	<b>132</b>
4.3.1	Deinstallation .....	132
4.3.2	Upgrade .....	132
4.3.3	Umstellung des Storage-Backends .....	132
<b>4.4</b>	<b>Docker und systemd-Integration .....</b>	<b>133</b>
4.4.1	systemd-Service-Units für Docker .....	133
<b>4.5</b>	<b>Docker und Proxies .....</b>	<b>134</b>
4.5.1	Docker-Daemon hinter einem Proxy betreiben .....	134
4.5.2	Docker-Client-Settings für Proxies (Docker $\geq$ 17.07) .....	135
4.5.3	Docker-Client- bzw. Image-Settings für Proxies (Docker $\leq$ 17.06) .....	135
<b>4.6</b>	<b>Docker im Betrieb .....</b>	<b>135</b>
4.6.1	Permanente Diensteinbindung .....	136
4.6.2	Lokale HA .....	136
4.6.3	Verbose Mode .....	137
4.6.4	Status-Überprüfung/Features .....	137
4.6.5	Docker-Systeminformationen .....	139
4.6.6	Docker-Daemon-Konfigurationsmöglichkeiten .....	139
4.6.7	Mögliche Startoptionen/Schalter des Docker-Daemons .....	140
4.6.8	Konfiguration per /etc/docker/daemon.json .....	141
4.6.9	Alternatives Docker-Verzeichnis als Konfigurationsbeispiel .....	143
4.6.10	Docker-CLI-Konfiguration .....	144
4.6.11	Docker-Plugins .....	144
<b>4.7</b>	<b>Docker-Image-Management – Basics .....</b>	<b>144</b>
4.7.1	Auszug der Docker-CLI-Subkommandos .....	145
4.7.2	CLI-Strukturen seit Docker 1.13 .....	146
4.7.3	docker container- und image-Subkommandos .....	147
4.7.4	Einfaches Image-Management .....	147
4.7.5	Docker-Namensräume und das Default-Registry-Problem .....	149
4.7.6	Docker-Images (unter docker.io) suchen .....	149
4.7.7	Image-Schema-Versionen .....	151
4.7.8	Offizielles CentOS-Image von docker.io pullen .....	151
4.7.9	Lokal verfügbare Docker-Images listen und filtern .....	153
4.7.10	Meta-Informationen von lokalen Images abfragen .....	157
4.7.11	Images löschen .....	159
4.7.12	docker save & load Images .....	161
4.7.13	Dangling Images: The good and the bad <none>:<none> .....	162
4.7.14	Build-History eines Images inspizieren .....	164

<b>4.8 Trusted Images .....</b>	165
4.8.1 Ein eigenes, generisches Trusted Basis-Image erzeugen .....	166
4.8.2 Gescripte Image-Erzeugung (YUM Based) .....	166
4.8.3 Mikro-Image »from scratch« mit go .....	167
4.8.4 Red Hats Container Health Index .....	168
4.8.5 Mikro-Image einer Legacy-Applikation .....	171
4.8.6 Transformation von Legacy-Apps in Images .....	171
4.8.7 Images: grundlegende Security-relevante Betrachtungen .....	171
<b>4.9 Betrieb und Management von Docker-Containern .....</b>	173
4.9.1 Kurzübersicht der relevanten Docker-CLI-Kommandos .....	174
4.9.2 Neues »docker container«-Subkommando .....	175
4.9.3 docker history .....	175
4.9.4 Container starten – docker [container] run .....	176
4.9.5 Docker-Registry, Image, Container run, lokaler Datastore – the Big Picture ...	177
4.9.6 (Random-)Container-Names und automatische Löschung (run --rm) .....	178
4.9.7 Container-HA: automatische Restarts .....	179
4.9.8 docker [container] run --readonly .....	180
4.9.9 Detached Container im Hintergrund starten .....	180
4.9.10 Auflisten von Container-Instanzen – docker ps .....	183
4.9.11 Starten und Stoppen existierender Container .....	184
4.9.12 docker [container] rename .....	187
4.9.13 Container-Instanzen löschen: docker [container] rm/prune .....	187
4.9.14 docker [container] attach-Optionen .....	188
4.9.15 Befehle im laufenden Container ausführen: docker [container] exec .....	190
4.9.16 docker [container] create .....	190
4.9.17 Container-Instanzen exportieren und als Images importieren .....	191
4.9.18 Kopieren von Daten: Container zwischen Host .....	193
4.9.19 docker checkpoint .....	194
<b>4.10 Prozessverwaltung im Container .....</b>	195
4.10.1 docker top .....	196
4.10.2 Prozesse im Container beenden .....	197
4.10.3 docker wait und Return/Exit-Codes .....	197
4.10.4 Den Container und alle in ihm laufenden Prozesse temporär pausieren .....	198
4.10.5 Live-Events mit docker events .....	198
<b>4.11 Container-Capabilities/Privilegien .....</b>	199
4.11.1 Prüfung und Auslesen der Capabilities .....	200
<b>4.12 Docker Logging .....</b>	202
4.12.1 Log-Driver .....	203
4.12.2 Zentralisierte Logs für Container-Instanzen .....	205
4.12.3 Container-Logs mit docker logs .....	205

<b>4.13 Einfache Applikationen im Container .....</b>	206
4.13.1 Vorberichtigungen .....	206
4.13.2 Installation von Applikationen im gestarteten Container .....	207
<b>4.14 Image-Modifikationen committen und taggen .....</b>	211
4.14.1 Commit – Beispiel .....	212
4.14.2 Nachträgliches Taggen von Images .....	214
4.14.3 Exkurs – die :latest-Problematik .....	217
<b>4.15 Layer-Strukturen .....</b>	217
4.15.1 Verzeichnisstrukturen auf dem lokalen Docker-Host .....	217
4.15.2 Was ist beim letzten Commit passiert, wo liegt der neue Layer? .....	218
4.15.3 IDs der RW-Layer von gestarteten Containern und Querbezüge .....	218
4.15.4 Layer-Analyse und Flattening (Zusammenfassung) .....	219
<b>4.16 Limitierte Container-Instanzen .....</b>	221
4.16.1 docker [container] stats .....	222
4.16.2 Mögliche Limitierungen .....	222
4.16.3 Beispiele aus der Praxis für limitierte Container-Instanzen .....	224
4.16.4 Nachträgliche Limitierung .....	225
<b>4.17 Docker-Images erstellen (docker build) und verwalten .....</b>	226
4.17.1 Best-Practice/File-Hierarchie .....	227
4.17.2 docker [image] build .....	227
4.17.3 Dockerfile-Direktiven/Instruktionen .....	228
4.17.4 Build-Anwendungsbeispiel: Apache-Container .....	240
4.17.5 Build-Exkurs: Single-Layer-»Squash«-Images .....	243
4.17.6 Multi-Stage Builds ab Docker 17.06 .....	246
4.17.7 Docker-Images mit systemd .....	247
<b>4.18 Best Build Practices .....</b>	250
4.18.1 Wegwerf-Produkte: Container sind kurzlebig und jederzeit reproduzierbar .....	250
4.18.2 Wer hat's gemacht? .....	251
4.18.3 Wie ist es bezeichnet? .....	251
4.18.4 Verwenden eines eigenen Build-Ordners pro Template .....	251
4.18.5 Verwendung eines .dockergignore-Files .....	251
4.18.6 Schlanke Images .....	252
4.18.7 Nur ein Prozess pro Container .....	252
4.18.8 Anzahl der Layer minimieren/niedrig halten .....	253
4.18.9 Multi-Line-Argumente in Befehlen (alphanumerisch) sortieren .....	253
4.18.10 Build-Cache .....	253
4.18.11 Image-Build und Container-Test-run ohne Docker? .....	254
4.18.12 Buildkit .....	254

<b>4.19 Docker-Networking .....</b>	254
4.19.1 Packungsdichten und die Realität .....	255
4.19.2 Der Docker-Netzwerkstack und Kubernetes .....	256
4.19.3 docker network-Hilfesystem .....	256
4.19.4 Basics: Netzwerkverbindung zum Container .....	257
4.19.5 Docker und iptables .....	258
4.19.6 IP eines gestarteten Docker-Containers auslesen .....	261
4.19.7 IP-Zuweisung und die /etc/hosts im Container .....	262
4.19.8 Komplette Netzwerk-Info eines Containers auslesen .....	263
4.19.9 Docker networks: bridge, host, none und mehr .....	264
4.19.10 Kommunikation: Welt zu Container, Docker-Portmapping .....	264
4.19.11 Portmapping explizit setzen .....	267
4.19.12 Docker-Netzwerke einrichten und modifizieren .....	270
4.19.13 Docker-Netzwerk-Driver .....	272
<b>4.20 Container per Docker-Netzwerk miteinander verknüpfen .....</b>	273
4.20.1 Beispiel-Setup: Apache/OpenLDAP-Container vernetzt .....	273
4.20.2 Verknüpfung der Container über userdefinierte Netzwerke .....	277
4.20.3 docker network prune .....	280
<b>4.21 Docker-Compose .....</b>	280
4.21.1 Portierbarkeit des Designs? Ja.....	281
4.21.2 Herkunft und Anwendungsbereiche .....	281
4.21.3 Bearbeitung von Yaml-Konfigurationsdateien .....	282
4.21.4 Was passiert beim Rollout? .....	282
4.21.5 Installation .....	283
4.21.6 Compose – Praxisbeispiel .....	283
4.21.7 Apache und OpenLDAP als Compose-Rollout .....	284
4.21.8 Build and Run .....	285
4.21.9 Handling der Services per docker-compose .....	287
4.21.10 Auszüge der gängigsten docker-compose-Sub-Befehle .....	288
4.21.11 Docker-Compose-Startup – Dependencies .....	291
<b>4.22 Docker-Storage-Driver .....</b>	293
4.22.1 Storage-Driver (local) .....	293
4.22.2 Übersicht der Storage-Driver .....	294
4.22.3 Storage-Driver- und Filesystem-Kombinationen .....	295
4.22.4 So what? – Storage-Driver Entscheidungsfragen .....	296
4.22.5 Shared Storage-Systeme und der Storage-Driver .....	296
4.22.6 Hot Replacement von Worker/Container-Nodes .....	297
<b>4.23 Deep Dive in die Beziehung zwischen Images bzw. Container-Instanzen und dem Storage-Driver .....</b>	297
4.23.1 Grundsätzliches: Images, Layer und der Storage-Driver .....	297

4.23.2	Graphdriver? .....	298
4.23.3	Aufgaben des Storage-Drivers .....	298
4.23.4	Exkurs: Secure Content Hashes und Content addressable Storage ab Docker 1.10 .....	299
4.23.5	Image-Layering und Sharing gemeinsamer Layer .....	300
4.23.6	Read/Write-Container, Readonly Image, Storage-Driver und Datenspeicherung .....	300
4.23.7	Alter Hund und neue Tricks? CoW für XFS .....	303
<b>4.24</b>	<b>Storage-Driver im Detail .....</b>	<b>304</b>
4.24.1	AUFS .....	304
4.24.2	OverlayFS .....	304
4.24.3	Neuerungen in Overlay(FS)2 .....	308
4.24.4	BTRFS .....	310
4.24.5	Devicemapper-Storage-Driver .....	316
4.24.6	Umbau des Storage-Drivers auf direct-lvm (RHEL/CentOS 7.x) .....	321
4.24.7	ZFS .....	329
4.24.8	Storage-Driver, Image-Layer und Performance .....	333
4.24.9	Schlussbemerkung .....	334
<b>4.25</b>	<b>Data-Sharing mit Docker-Volumes .....</b>	<b>335</b>
4.25.1	Vorab: Der Blick auf das große Ganze – Docker-Volumes im »echten« Cluster-Kontext .....	335
4.25.2	Docker-Data-Volumes .....	336
4.25.3	Host-mounted Data-Volumes .....	337
4.25.4	Data-Volume-Mounts vom Host für OpenLDAP-Container .....	340
4.25.5	»Docker-managed« Data-Volumes .....	343
4.25.6	Readonly Data-Volumes .....	345
4.25.7	(Anonyme) Volumes entfernen .....	345
4.25.8	Zusammenfassung .....	346
<b>4.26</b>	<b>SSSD to the Rescue – konsistente ID-Mappings für Daten in Containern und Container-Clustern .....</b>	<b>346</b>
4.26.1	ID-Divergenzen .....	347
4.26.2	LDAP/AD und SSSD .....	347
4.26.3	Ausgangsbasis: Der Verzeichnisdienst .....	348
4.26.4	Pre Flight Requirements .....	350
4.26.5	Vereinfachtes Test-Setup .....	351
4.26.6	Keine Wildcard-IDS, sondern feste Ranges und RIDs .....	356
4.26.7	Image Build and Run mit AD-IDs .....	356
4.26.8	Statische User- und Gruppen-IDs in den Dockerfiles/-Images? .....	357

---

<b>5 Container Security</b>	359
<b>5.1 Docker mit TLS/SSL .....</b>	359
5.1.1 Grundlagen .....	360
5.1.2 SSL-Standard und veraltete Protokolle .....	361
5.1.3 Vorbetrachtungen zur Zertifikatserzeugung .....	361
5.1.4 Anpassungen der openssl.cnf .....	362
5.1.5 Erzeugung der Zertifikate mit angepasster OpenSSL-Konfiguration .....	364
5.1.6 CA erzeugen .....	365
5.1.7 Zertifikats-Request erzeugen .....	366
5.1.8 Zertifikats-Request signieren .....	367
5.1.9 Key .....	369
5.1.10 Erzeugte Dateien und weitere Tasks .....	369
5.1.11 Weitere Docker-Hosts und Zertifikate .....	370
5.1.12 Daemon-Startparameter .....	370
5.1.13 Test des Client-Zugriffs per TLS .....	371
5.1.14 Übersicht der zur Verfügung stehenden TLS-Flags für Server und Client .....	373
<b>5.2 Images signieren, verifizieren und verwalten mit Skopeo und Atomic .....</b>	374
5.2.1 Funktionaler Überblick .....	375
5.2.2 Setup .....	376
5.2.3 Remote Inspect .....	377
5.2.4 Skopeo Transports .....	379
5.2.5 Image Signing .....	379
5.2.6 Digests, Manifeste und Signaturen: Mögliche Fehlerquellen minimieren .....	381
5.2.7 Hands-On: Images mit Skopeo und Atomic signieren .....	381
5.2.8 Verify .....	385
5.2.9 atomic diff .....	385
<b>5.3 »Manuelle« Vulnerability-Scans .....</b>	387
5.3.1 Atomic Scan (CVE-Scanner) .....	387
5.3.2 CVE-Scanner-Alternativen zu Atomic Scan bzw. OpenSCAP .....	390
5.3.3 Image-Scanning mit Anchore .....	390
5.3.4 Manueller Anchore-Scan auf dem Container- bzw. Build-Host .....	391
5.3.5 Ein paar kleine Anwendungsbeispiele .....	395
5.3.6 CVE-Analyse .....	396
5.3.7 Bench .....	398
5.3.8 Testlauf mit Docker Bench .....	399
5.3.9 Docker Online Image Scanning .....	402
<b>5.4 Fazit .....</b>	402

---

<b>6 Die private Trusted (Docker-)Registry</b>	403
<b>6.1 Die Registry im Detail</b>	403
6.1.1 Multi-Purpose Registry: Auswahlkriterien	403
6.1.2 Docker-Registry: up and running?	404
6.1.3 Registry-Architektur	405
<b>6.2 Vorbereitungen zum Setup</b>	407
6.2.1 Insecure Registry	407
6.2.2 Das Docker-Default-Registry-Image	407
6.2.3 Konfigurations-Override	409
<b>6.3 Registry-Setup-Möglichkeiten</b>	411
6.3.1 Die Container-basierte Docker-Registry	412
<b>6.4 Registry-Setup: Vorbereitungen und Betrieb</b>	412
6.4.1 Vorbereitungen für den Upload in eine Insecure-Registry	412
6.4.2 Vorbereitungen: Regeln zum Image-Tagging für Registry-Upserts verstehen	413
6.4.3 Upload/Push eines Images in die Registry	413
6.4.4 Löschen von Images in einer privaten, einfachen Docker-Registry?	414
6.4.5 Docker-Filesystem-Layer, Manifeste und die Garbage-Collection	415
<b>6.5 Docker-Registry mit TLS</b>	419
6.5.1 Zertifikatslokationen	419
6.5.2 CA, Zertifikat und Key für den Registry-Host	420
6.5.3 Start und Betrieb der Registry mit TLS	421
6.5.4 Push in die Registry mit TLS	422
6.5.5 Verbindung zur TLS-gesicherten Registry mit Client-Zertifikat	424
6.5.6 Troubleshooting-Tipps	424
<b>6.6 Zentrale Registry-Authentifizierung via LDAP/TLS</b>	424
6.6.1 Setup des LDAP-Images mit TLS	426
6.6.2 Setup des http-Proxys	429
6.6.3 Build & Start des Registry-LDAP-http-Proxys per Compose	433
6.6.4 Test des Setups	435
6.6.5 Zugriffskontrollen	438
<b>6.7 (Docker-)Registry mit AD-Authentifizierung</b>	440
6.7.1 Konzepte und Verfahren	440
6.7.2 Simple Bind	440
6.7.3 SSO/Kerberos-Anbindung	441
<b>6.8 Registry-Alternativen: Artifactory</b>	441
6.8.1 Aufbau und Funktionsweise	442
6.8.2 Installation via Docker-Compose	442

6.8.3	Artifactory-Administration .....	444
6.8.4	Repo-Setup .....	445
6.8.5	AD-Anbindung .....	446
6.8.6	Image-Upload .....	447
6.8.7	SSL .....	449
6.8.8	Xray .....	449
6.8.9	Einrichtung und Scanning .....	450
6.8.10	Watches und Build-Integration .....	452
6.8.11	Artifactory und Xray: Preise .....	453
<b>6.9</b>	<b>Registry-Alternativen: Sonatype Nexus .....</b>	<b>453</b>
6.9.1	Features und Funktionen .....	453
6.9.2	Installationsvor betrachtungen .....	454
6.9.3	Nexus bzw. Nexus Repository Manager (NRM) 3 als Bare-Metal-Installation .....	454
6.9.4	Überblick über die Datenstrukturen des NRM3 .....	456
6.9.5	Administration des NRM .....	457
6.9.6	Docker-Repo einrichten .....	457
6.9.7	Nexus Repository Manager: SSL mit integriertem »Jetty«-HTTP-Server .....	460
6.9.8	Externe Authentifizierungsquellen .....	464
6.9.9	RBAC .....	466
6.9.10	Rollen und Privilegien über das AD .....	466
6.9.11	HA für NRM .....	466
6.9.12	Preise für NRM Pro .....	467
<b>6.10</b>	<b>Registry-Alternativen: VMware Harbor .....</b>	<b>468</b>
6.10.1	Harbor-Architektur .....	468
6.10.2	Die Harbor-Container (ohne Notary und Clair) .....	469
6.10.3	Installation und Vorbereitungen .....	470
6.10.4	Security-Integration: Vulnerability-Scans und Content-Signing .....	470
6.10.5	Harbor-Setup mit SSL/TLS, Notary und Clair .....	472
6.10.6	Login und Einstellungen .....	474
6.10.7	Repo einrichten, Tag und Push .....	475
6.10.8	Image-Scanning .....	475
6.10.9	Images beim Push signieren .....	476
<b>6.11</b>	<b>Fazit zu den Container-Registries .....</b>	<b>478</b>

---

## 7 Weitere Container-Host-Plattformen 481

<b>7.1</b>	<b>Vorbetrachtungen .....</b>	<b>481</b>
7.1.1	Who's on? .....	481

7.1.2	Im Detail .....	482
7.1.3	Kandidaten? .....	482
<b>7.2</b>	<b>Atomic Host (RHEL/CentOS) .....</b>	<b>483</b>
7.2.1	Die Funktionsweise von Atomic Host und der Aufbau des Dateisystems .....	484
7.2.2	Upgrades und Rollbacks .....	484
7.2.3	Paketinstallation und Betrieb/Management des OS .....	487
7.2.4	Kubernetes nachinstallieren .....	487
7.2.5	Sonstige Managementfunktionen .....	488
<b>7.3</b>	<b>Rocket Science? – CoreOS .....</b>	<b>489</b>
7.3.1	CoreOS: Nicht vorhandene Pakete und Rolling Upgrades .....	490
7.3.2	Upgrades .....	490
7.3.3	CoreOS-Anpassungen .....	493
7.3.4	Nach dem Start .....	495
<b>7.4</b>	<b>SUSE CaaSP .....</b>	<b>495</b>
7.4.1	Wie funktioniert es? .....	495
7.4.2	Unter der Haube .....	497
7.4.3	Login und Überblick .....	498
<b>7.5</b>	<b>SLAs und Preise .....</b>	<b>501</b>
<b>7.6</b>	<b>Container-Node-Plattformen in der Cloud .....</b>	<b>501</b>
<b>8</b>	<b>Fazit – (Single-Node-)Container-Plattformen .....</b>	<b>503</b>
<b>8.1</b>	<b>Der Wandel .....</b>	<b>503</b>
<b>8.2</b>	<b>»Die« Container-Plattform? .....</b>	<b>503</b>
<b>TEIL III Skalierbare Container-Cluster und Container-Orchestrierung</b>		
<b>9</b>	<b>Container-Cluster – von Planern und Orchestern .....</b>	<b>507</b>
<b>9.1</b>	<b>Worum es geht – the Big Picture .....</b>	<b>507</b>
<b>9.2</b>	<b>World of Tiers – Teil 2 .....</b>	<b>508</b>
9.2.1	World of Tiers – die Welt der Schichten .....	508
9.2.2	Scheduling vs. Orchestration? .....	508
9.2.3	Die Layer/Tiers und ihr Zusammenwirken .....	508

9.2.4	Container-Cluster .....	509
9.2.5	Unser Ziel .....	510
<b>9.3</b>	<b>Vorbereitungen .....</b>	<b>510</b>
9.3.1	Distributions- und Plattform-Fragen .....	510
9.3.2	CaaSP, Kubernetes und OpenShift? .....	512
<b>9.4</b>	<b>Pre-Flight-Requirements: Zeitsynchronisation .....</b>	<b>513</b>
9.4.1	NTP und die Relativität .....	513
9.4.2	NTP-Basics .....	513
9.4.3	NTP-Setup .....	514
9.4.4	NTP-Setup mit zusätzlichen Peers .....	518
9.4.5	Chrony .....	518
<b>9.5</b>	<b>Pre-Flight-Requirements: pssh .....</b>	<b>519</b>
9.5.1	Grundsätzliches .....	519
9.5.2	Setup von pssh auf allen Nodes .....	519
9.5.3	pssh – korrespondierende Dateien/Einstellungen .....	520

## **10 Schlüsselmeister im Container-Cluster: Key/Value Stores und Service Registry/Discovery**

---

<b>10.1</b>	<b>Key/Value Stores .....</b>	<b>524</b>
10.1.1	Vorbetrachtungen .....	524
10.1.2	Key/Value Stores im Detail .....	524
10.1.3	Backup- und Verfügbarkeitsstrategien .....	525
<b>10.2</b>	<b>Service Discovery/Registry .....</b>	<b>526</b>
10.2.1	Sichten – und nicht vernichten .....	526
10.2.2	Service Discovery im Detail .....	527
10.2.3	Statisch vs. dynamisch .....	528
10.2.4	Konfigurationsreplikation .....	529
<b>10.3</b>	<b>Key/Value Stores im Kurzüberblick .....</b>	<b>530</b>
10.3.1	Teile und herrsche .....	530
10.3.2	Entscheidungsfindung – Raft vs. Paxos .....	531
10.3.3	Raft-Demo .....	532
10.3.4	etcd .....	532
10.3.5	Consul .....	535
10.3.6	Zookeeper .....	536
<b>10.4</b>	<b>Key/Value-Store-Cluster am Beispiel von Consul .....</b>	<b>536</b>
10.4.1	Die Consul-Komponenten in der Übersicht .....	537

10.4.2	Zusammenfassung der Consul-Kernfunktionen und -Features .....	538
10.4.3	Zusammenfassung der Consul-Basisarchitektur und Funktionsbeschreibung .....	539
10.4.4	Consul-Service-Monitoring im Detail .....	540
10.4.5	Setup eines Consul-Clusters: Vorberichtigungen .....	541
10.4.6	Einfache Consul-Grundkonfiguration .....	542
10.4.7	Consul als generischer Key/Value Store für Container-Cluster .....	549
10.4.8	Endpunkte/Endpoints .....	551
10.4.9	Cluster-Leader-(Re-)Election .....	552

## 11 Kubernetes (K8s) 555

---

<b>11.1</b>	<b>Kubernetes im Überblick .....</b>	555
11.1.1	Vom Borg zum Steuermann .....	555
11.1.2	Kubernetes und CRI-O .....	556
11.1.3	Releases, Changes und kein Ende .....	556
<b>11.2</b>	<b>Dockerd, cri-containerD, CRI-O? – K8s-Container-Engines, KISS, redundante KV Stores und die Zukunft .....</b>	558
11.2.1	Docker, Docker, Docker ...? Nö. CRI-O. Oder cri-containerd .....	559
11.2.2	K8s, Key/Value Backends, nice-to-have HA und SPoFs .....	561
<b>11.3</b>	<b>Kubernetes-Komponenten .....</b>	562
11.3.1	Komponenten eines Single-Master-K8s-Clusters .....	562
11.3.2	Dienste auf den Kubernetes-Master-Nodes .....	564
11.3.3	Dienste auf den Kubernetes-Workern aus technischer Sicht .....	567
<b>11.4</b>	<b>Networking in Kubernetes .....</b>	569
11.4.1	Unterschiede zu Docker .....	570
11.4.2	kubenet- bzw. CNI-Plugins .....	571
11.4.3	Netzwerkkommunikation im K8s-Cluster .....	572
11.4.4	Überblick über einige Kubernetes-Netzwerk-Plugins .....	573
<b>11.5</b>	<b>etcd: Key/Value Store für Kubernetes im Detail .....</b>	574
11.5.1	Grundsätzliches .....	574
11.5.2	Arbeitsweise/Funktionsprinzip von etcd im Zusammenspiel mit K8s .....	575
11.5.3	K8s und Redundanzen .....	575
11.5.4	Start eines Single-etcd-Service .....	578
11.5.5	etcd-Datenstrukturen im Detail .....	579
11.5.6	etcd: Hot-Backup .....	580
<b>11.6</b>	<b>Redundanter etcd-Cluster – Installation und Setup .....</b>	581
11.6.1	etcd-Konfiguration .....	583

11.6.2	Manueller etcd-Cluster-Start .....	588
11.6.3	etcd-Integration in systemd und lokale HA .....	590
11.6.4	etcd-Start via systemd-Unit .....	591
11.6.5	Failover-Test .....	593
<b>11.7</b>	<b>Flannel CNI für Kubernetes-Cluster .....</b>	<b>593</b>
11.7.1	Kubernetes-Cluster und Overlay-Networking .....	593
11.7.2	Overlay Network mit Flannel .....	594
11.7.3	Flannel-Dependencies .....	596
11.7.4	Flannel-Setup auf den drei Nodes .....	596
11.7.5	Backends .....	598
11.7.6	Exkurs: Multiple Flannel-Overlay-Netzwerke .....	602
<b>11.8</b>	<b>Kubernetes-Setup-Varianten .....</b>	<b>603</b>
11.8.1	Flavours? .....	603
11.8.2	Minikube? Nö. ....	604
11.8.3	Hyperkube – Aktuell: wer will. Zukünftig: vielleicht ein Muss .....	605
11.8.4	kubeadm now? .....	607
<b>11.9</b>	<b>Einfaches Kubernetes-Single-Master-Setup (3 etcd-Nodes) .....</b>	<b>608</b>
11.9.1	Vorbetrachtungen: API-Server, Controller Manager und Scheduler .....	608
11.9.2	Preflight-Checks und Tasks .....	608
11.9.3	Setup von kube-apiserver .....	608
11.9.4	Setup des kube-controller-manager .....	614
11.9.5	Setup des kube-scheduler .....	615
11.9.6	Scheduler-Algorithmen: Predicates und Priorities .....	616
<b>11.10</b>	<b>Kubernetes-Worker-Konfigurationen für Single Master .....</b>	<b>618</b>
11.10.1	Der Node Controller (kube-controller-manager) und die Verfügbarkeit der Worker-Nodes .....	618
11.10.2	Kubelet-Konfiguration .....	620
11.10.3	kube-proxy-Konfiguration .....	621
11.10.4	Sonstiges: zentrale Konfigurationsparameter .....	622
11.10.5	kubectl .....	622
11.10.6	kubectl-Bash-Completion .....	623
11.10.7	Start der Master- und aller Worker-Dienste .....	624
11.10.8	DNS (kube-dns/CoreDNS) .....	626
<b>11.11</b>	<b>Redundante und hochverfügbare Kubernetes-Master:</b>	
	<b>Konzepte und Möglichkeiten .....</b>	<b>627</b>
11.11.1	Vorbetrachtungen: Ist/Soll-Stand .....	628
11.11.2	Vorbetrachtungen: mögliche Vorgehensweisen .....	628
11.11.3	Pre-Flight-Requirements und konkrete Setup-Vor betrachtungen .....	629
11.11.4	Das Multi-Active-Problem .....	630

11.11.5	Zertifikatsanpassungen und Cluster-Aliase .....	630
11.11.6	Failover-Varianten .....	631
<b>11.12</b>	<b>Vorbereitendes Setup der Kubernetes-Master für Hot-Failover .....</b>	<b>632</b>
11.12.1	Hot-Failover Kubernetes-Master .....	632
11.12.2	Vorbereitende Konfiguration .....	632
11.12.3	Kubectl-Zertifikatsfehlermeldungen und kubectl-Debugging .....	635
<b>11.13</b>	<b>Pacemaker-Integration der drei Kubernetes-Master .....</b>	<b>636</b>
11.13.1	Vorbetrachtungen – HA der K8s-Kernkomponenten mit Pacemaker .....	636
11.13.2	Pacemaker und Corosync .....	636
11.13.3	Pacemaker: Agenten, Ressourcen und Constraints im Kurzüberblick .....	638
11.13.4	Pakete und vorbereitende Tasks .....	640
11.13.5	Corosync-Setup .....	641
11.13.6	Benötigte Grundeinstellungen .....	644
11.13.7	Erzeugung der benötigten Pacemaker-Ressourcen für den K8s-Cluster .....	645
11.13.8	Erzeugen der weiteren Kubernetes-Master-Ressourcen .....	647
11.13.9	Gruppieren der kube*-Ressourcen .....	647
11.13.10	Klonen der Gruppe .....	647
11.13.11	IP-Colocation .....	648
11.13.12	etcd und flanneld .....	648
11.13.13	Inbetriebnahme des Pacemaker-gesteuerten K8s-Master-Clusters .....	649
11.13.14	Failover-Test .....	651
11.13.15	Grafisches Pacemaker-Cluster-Management mit HAWK2 .....	651
11.13.16	Grafisches Pacemaker-Cluster-Management mit der PCS-GUI .....	654
11.13.17	Stonith – ein kurzer Überblick .....	655
11.13.18	Grundlegende Management-Tasks und -Regeln für Pacemaker-Cluster .....	657
<b>11.14</b>	<b>Setup eines K8s-Multi-Node-Clusters mit SUSE CaaSP .....</b>	<b>657</b>
11.14.1	Setup und Rollenauswahl .....	657
11.14.2	Rollout und Provisionierung .....	660
11.14.3	Login und Überblick .....	661

---

## **12 Kubernetes-Control-Plane als Microservice-Architektur**

---

<b>12.1</b>	<b>Vorbetrachtungen zum Setup einer Pod-basierten Kubernetes-Control-Plane .....</b>	<b>665</b>
<b>12.2</b>	<b>Setup .....</b>	<b>668</b>
12.2.1	Setup-Voraussetzungen und Vorbetrachtungen .....	668
12.2.2	Achtung Kubelet $\geq$ 1.11 und statische (Pod-)Manifests .....	668
12.2.3	Repositories und Installation .....	669

12.2.4	Erforderliche Pakete .....	669
12.2.5	cgroupfs-Fehler im kubelet .....	670
12.2.6	Von der Installation der Pakete erzeugte Files .....	670
12.2.7	kubelet-Service-Unit .....	671
12.2.8	Swap on/off .....	672
12.2.9	Dynamische Kubelet-Konfiguration ab K8s $\geq$ 1.10 .....	673
12.2.10	Stats-Fehler im Log des Kubelet-Containers .....	675
12.2.11	Verfügbare kubeadm-init-Direktiven .....	675
<b>12.3</b>	<b>Kubernetes-Initialisierung mit kubeadm init .....</b>	<b>677</b>
12.3.1	Pre-Flight-Requirements .....	677
12.3.2	Rollout des Masters .....	678
12.3.3	Achtung: Token-Fragen .....	681
12.3.4	Post-Initialisierungs-Tasks (kubectl) .....	682
12.3.5	kubectl auf weiteren Nodes gangbar machen .....	682
12.3.6	Weitere wichtige Initialisierungsdateien .....	683
12.3.7	Join weiterer K8s-Nodes .....	683
12.3.8	etcd-Sidecar und Kubernetes .....	685
12.3.9	Under the Hood – was ist beim Setup passiert? .....	686
<b>12.4</b>	<b>Einzelschritte des kubeadm init .....</b>	<b>690</b>
12.4.1	Einzel-Tasks des kubeadm init .....	691
12.4.2	Schrittweise Cluster-Generierung per kubeadm [alpha] phase .....	691
<b>12.5</b>	<b>Pod-basiertes Overlay-Netz .....</b>	<b>696</b>
12.5.1	Weave .....	696
12.5.2	Das war's? .....	700
<b>12.6</b>	<b>Arbeiten mit Pod-basiertem etcd .....</b>	<b>700</b>
<b>12.7</b>	<b>Verfügbarkeit und Ausfallsicherheit der K8s-Komponenten im Pod-Modell .....</b>	<b>702</b>
12.7.1	Redundanter, realer etcd-Cluster für K8s im Container-Setup .....	703
12.7.2	Das MasterConfiguration-Objekt .....	704
12.7.3	kubeadm init per config-File .....	706
12.7.4	Erzeugen der weiteren Master .....	706
12.7.5	»Easy«-HA in Kubernetes 1.11? Leider nein .....	709

## **13 Kubernetes-Cluster ohne Docker**

---

<b>13.1</b>	<b>Kubernetes-Cluster mit CRI-O als Container-Engine .....</b>	<b>711</b>
13.1.1	Das Big Picture und der prozedurale Ablauf .....	711
13.1.2	Stable? Yes .....	713
13.1.3	Setup-Voraussetzungen .....	713

## Inhalt

13.1.4	Erforderliche Pakete installieren .....	714
13.1.5	Haben und nicht haben: crictl .....	714
13.1.6	Setup .....	714
13.1.7	Netzwerk .....	715
13.1.8	kubeadm-init-Setup mit CRI-O .....	718
13.1.9	Low-Level Management mit crictl .....	719
<b>13.2</b>	<b>Buildah, Podman und Skopeo: (Docker-)Images ohne Docker erstellen und verwalten .....</b>	<b>722</b>
<b>13.3</b>	<b>Buildah: »Look Mom, no Docker ...« .....</b>	<b>723</b>
13.3.1	Buildah: Funktionsweise und Installation .....	724
13.3.2	Tests und Buildah-Beispiele .....	724
13.3.3	Involvierte Konfigurationsdateien .....	725
13.3.4	Image-Import .....	728
13.3.5	Buildah: CLI-Handling und Image-Verwaltung .....	729
13.3.6	Image-Konfiguration mit Buildah modifizieren .....	736
13.3.7	Anpassung und Commit des Containers .....	736
13.3.8	Push nach docker.io .....	739
<b>13.4</b>	<b>Image Build mit Kaniko .....</b>	<b>740</b>
13.4.1	Image-Build .....	740
<b>13.5</b>	<b>Podman .....</b>	<b>741</b>
13.5.1	Lib-Sharing .....	743
13.5.2	Installation .....	744
13.5.3	Podman: Praktische Beispiele .....	744
13.5.4	Create und run (simple Container) .....	745
13.5.5	Create und Run im Pod .....	746
<b>13.6</b>	<b>Podman vs crictl .....</b>	<b>748</b>
13.6.1	crictl und Podman im Vergleich .....	749
13.6.2	Funktionsunterschiede und Gemeinsamkeiten .....	750
<b>13.7</b>	<b>Kompose – der Docker-Compose-Konverter für K8s .....</b>	<b>750</b>

## **14 Kubernetes-Cluster: Ressourcen verstehen und verwalten**

---

753

<b>14.1</b>	<b>kubectl .....</b>	<b>753</b>
14.1.1	kubectl-Bash-Completion .....	753
14.1.2	Client/Server-Versionen .....	754
14.1.3	Das kubectl-Kommando .....	754

14.1.4	kubectl-Konfiguration .....	754
14.1.5	Die wichtigsten kubectl-Subkommandos in der Übersicht .....	755
14.1.6	Die kubectl-Manpages .....	756
14.1.7	Welche Ressourcen bzw. Workloads können im K8s-Cluster via kubectl-CLI verwaltet werden? .....	757
14.1.8	kubectl api-resources .....	759
14.1.9	API-Versionierungsdicticht .....	760
14.1.10	kubectl explain <Ressource> .....	763
14.1.11	K8s-Ressourcen direkt mit kubectl run/create deployen .....	764
14.1.12	Verbose Mode und kubectl-Debugging .....	765
14.1.13	Der kubectl-run --restart=--Schalter und seine Auswirkung auf erzeugte Objekte .....	765
14.1.14	Einfache kubectl-Beispiele im Hinblick auf erzeugte API-Objekte .....	765
14.1.15	kubectl-Generatoren .....	768
14.1.16	Grundlegende K8s-Cluster-Informationen abfragen .....	768
<b>14.2</b>	<b>Kleine Kubernetes-Cluster und »Taint Nodes«</b> .....	770
14.2.1	Was sind Taints und Tolerations? .....	770
14.2.2	Taint-Abfrage unseres Masters .....	772
14.2.3	Beispiele für das (Un-)Tainten von Nodes .....	772
<b>14.3</b>	<b>(Worker-)Node-Kapazitäten</b> .....	773
14.3.1	Analyse .....	773
14.3.2	Weitere Node-Informationen abfragen .....	777
14.3.3	Under Pressure .....	778
<b>14.4</b>	<b>Ressourcen im Kubernetes-Cluster ausrollen</b> .....	780
14.4.1	Was passiert eigentlich beim Ausrollen einer Ressource? .....	780
14.4.2	Vereinfachter Ablauf .....	784
14.4.3	ImagePullPolicies .....	785
14.4.4	Default-Verteilungsstrategien des Schedulers für die Worker-Nodes .....	786
14.4.5	Unterschiedliche bzw. mehrere Ressourcen in einem YAML-File .....	787
14.4.6	Manifest-Versionierung .....	788
<b>14.5</b>	<b>Pods</b> .....	788
14.5.1	Technische Details .....	788
14.5.2	Pods und Startup-Orderings der Container? .....	792
14.5.3	K8s-Besonderheit: Das »pause«-Image, der Pod und Namespace-Reservierungen .....	792
14.5.4	K8s-Image-Pull und (lokale) Trusted Registries .....	793
14.5.5	Erstellen eines Pods .....	793
14.5.6	Status- und Laufzeitattribute – kubectl get mit YAML-Output .....	796
14.5.7	kubectl attach .....	797
14.5.8	Laufenden Pod bzw. laufende Ressource editieren .....	797

14.5.9	Multiple Ressourcen per Manifest anlegen oder löschen .....	802
14.5.10	Exkurs: YAML vs. JSON und die Konvertierung auf eine neuere API-Version ....	802
14.5.11	Kubernetes-Pod-Phasen und -Zustände (Status) .....	803
14.5.12	Auszüge einiger Beispiele für mögliche Zustände von Pods .....	804
14.5.13	Pods/Ressourcen nach Namespaces anzeigen lassen .....	805
14.5.14	Debugging mit kubectl describe .....	806
14.5.15	K8s-Pods aus der Docker-Sicht .....	807
14.5.16	(Force) Removal eines Pods oder anderer Ressourcen .....	808
14.5.17	Pod mit unterschiedlichen Containern/Images .....	808
14.5.18	Setzen von Kommandos in der Pod-Spezifikation per command und args ....	809
14.5.19	Logs .....	810
14.5.20	Detailliertes Auslesen von Pods .....	811
14.5.21	RestartPolicies und Startverzögerung .....	811
14.5.22	Einfache Pods (ohne Deployments) und Node Bindings .....	812
14.5.23	Kommandos im Pod/Container ausführen .....	812
14.5.24	Automatische Bereinigung alter Pods auf den Kubelets .....	813
<b>14.6</b>	<b>Pod- und Container-Ressourcen, -Requests und -Limitierungen sowie QoS und Capabilities .....</b>	<b>814</b>
14.6.1	CPU-Requests und -Limits .....	815
14.6.2	Memory-Requests und Limits .....	816
14.6.3	QoS – Quality of Service im Bezug auf Limits und Request .....	817
14.6.4	Exklusives CPU-Pinning .....	818
14.6.5	Default-Requests und -Limits .....	820
14.6.6	OOM Scores .....	821
14.6.7	Capabilitäts für Container in Pods setzen .....	821
14.6.8	PodDisruptionBudget .....	823
14.6.9	Pod-Prioritäten .....	823
14.6.10	PodSecurityPolicies .....	826
<b>14.7</b>	<b>Umgebungsvariablen von Pods und Containern auslesen, setzen und nutzen ....</b>	<b>827</b>
14.7.1	Pod oder Container? .....	828
14.7.2	Print-Out ENVs – Pod Variablen setzen und ausgeben .....	828
14.7.3	Limits? .....	829
14.7.4	Erzeuge Ordner im Mountpath .....	831
<b>14.8</b>	<b>Pods und ConfigMaps .....</b>	<b>833</b>
14.8.1	Wie funktioniert es? .....	833
14.8.2	ConfigMaps in der Praxis .....	834
14.8.3	Varianten zur Erstellung von ConfigMaps .....	838
14.8.4	Unterschiede: --from-file und --from-env-file .....	838
14.8.5	Binary Data in ConfigMaps .....	839
14.8.6	Von der Control Plane verwendete ConfigMaps .....	840

<b>14.9 Pods und Init-Container .....</b>	841
14.9.1 Wie funktioniert es? .....	841
14.9.2 Anwendungsmöglichkeiten für Init-Container .....	842
14.9.3 Ein einfaches Beispiel .....	842
14.9.4 Phasen des Init-Containers .....	844
14.9.5 Mehrstufiges Init .....	845
<b>14.10 Health-Checks .....</b>	847
14.10.1 Ready? Live? .....	847
14.10.2 Unterschiedliche Auswirkungen der Probes .....	848
14.10.3 Probe-Verfahren .....	849
14.10.4 Timeouts, Initial Delays und Delays von Init-Containern mit einkalkulieren .....	851
14.10.5 Readiness Probe .....	851
14.10.6 Liveness Probe .....	853
14.10.7 Pod Readiness Gate .....	857
14.10.8 Pod-Lifecycle-Management und Hooks .....	857
<b>14.11 Scale-Out: von Replication Controllern zu ReplicaSets .....</b>	859
14.11.1 Beispiel für einen ReplicationController (rc) .....	862
14.11.2 Scale-Out .....	865
14.11.3 Pods in ReplicationControllern löschen .....	866
14.11.4 Rolling Updates von ReplicationControllern .....	866
14.11.5 ReplicaSets und Set-based Label Selectors .....	867
14.11.6 Löschen des ReplicaSets .....	871
<b>14.12 Jobs .....</b>	872
14.12.1 Failure-Verhalten .....	873
14.12.2 Jobs-Beispiel .....	873
14.12.3 CronJobs .....	876
<b>14.13 Deployments .....</b>	878
14.13.1 Verkapselung .....	878
14.13.2 Pod-Freigabeerkennung? Ja!n. .....	880
14.13.3 Erstellen eines Deployments .....	881
14.13.4 Umgebungsvariablen in Deployments nutzen (Pod-Name-basierte Log-Ordner) .....	883
14.13.5 Deployment per kubectl run erzeugen .....	886
14.13.6 Deployment mit Service erzeugen .....	888
14.13.7 Revisionshistorie .....	889
14.13.8 Rolling Update, Rollout und Revisionssicherheit .....	890
14.13.9 Update- und Revisionshistorie .....	891
14.13.10 Update-Strategien .....	893
14.13.11 Resume und Pause .....	894

## Inhalt

14.13.12	Absichtlich erzeugter Fehler beim Ausrollen bzw. Auto-Stop des Rollouts ...	894
14.13.13	Rollback-Verfahren .....	894
14.13.14	Multiple-Release-Tracks- und Deployment-Strategien .....	895
<b>14.14</b>	<b>DaemonSets .....</b>	<b>898</b>
14.14.1	Vorbetrachtungen .....	898
14.14.2	Kommunikation mit DaemonSets .....	900
14.14.3	Mögliche Anwendungsfälle .....	900
14.14.4	DaemonSet-Scheduler und Node-Zuordnung .....	901
14.14.5	Beispiel .....	901
<b>14.15</b>	<b>Namespaces: Limits, Quotas und echte Multi-Tenancy? .....</b>	<b>904</b>
14.15.1	Vorbetrachtungen .....	904
14.15.2	Objekte mit und ohne Namespace-Zuordnung .....	907
14.15.3	Namens- und Designfragen .....	908
14.15.4	Praktischer Einsatz .....	909
14.15.5	Ressourcen im Namespace erzeugen .....	910
14.15.6	Limitierte Pods und Namespaces .....	911
14.15.7	Limit(Range)s vs. Quota .....	911
14.15.8	Limits für Namespaces in der Praxis .....	912
14.15.9	Node- und Pod-spezifische Statistiken .....	915
14.15.10	Namespaces und Ressource Quotas .....	915
<b>14.16</b>	<b>Services .....</b>	<b>919</b>
14.16.1	Wie funktioniert es? .....	919
14.16.2	Service Endpoints .....	922
14.16.3	Die Service-Ressource aus funktionaler Sicht .....	923
14.16.4	Der Default-kubernetes-Service .....	924
14.16.5	K8s-Services und DNS .....	925
14.16.6	kube-dns bzw. CoreDNS als Pod-basierter Cluster-DNS für Service-Ressourcen .....	926
14.16.7	CoreDNS statt kube-dns – warum? .....	927
14.16.8	Pod-DNS-Policies .....	928
14.16.9	Kubelets, DNS-Erbschaft und resolv.conf .....	929
14.16.10	Implementierung des kube-dns-Deployments .....	929
14.16.11	Implementierung des CoreDNS-Deployments .....	930
14.16.12	Konkretes Beispiel: DNS und Service-Namensauflösung im Cluster .....	930
14.16.13	Service-Namenskonventionen .....	932
14.16.14	Ein einfaches Service-Manifest .....	933
14.16.15	Service-Infos in den Pods .....	934
14.16.16	Kubernetes-Services debuggen .....	934
14.16.17	Proxy-Modes .....	934
14.16.18	Service-Typen .....	943

14.16.19 Beispiel: Service-Bereitstellung via NodePort .....	944
14.16.20 Service-Bereitstellung in der GKE-Cloud mit dem Typ Loadbalancer .....	948
14.16.21 Deployment-Beispiele .....	949
14.16.22 Weiche Migration von Legacy-Systemen mit Headless Services .....	949
14.16.23 DNS-Redirects mit dem Service-Typ externalName .....	952
14.16.24 Service/Deployment-Beispiel: K8s-Dashboard .....	953
14.16.25 K8s-Services, Loadbalancer und der Rest der Welt .....	963
<b>14.17 Ingress .....</b>	<b>964</b>
14.17.1 Was ist ein Ingress? .....	964
14.17.2 Ingress-Funktionalität und -Komponenten .....	965
14.17.3 Aktuelle Ingress-Limitationen: »Wir brauchen doch nur HTTP(S), oder?« .....	967
14.17.4 Ingress-Implementierung .....	968
14.17.5 Ingress im Detail .....	968
14.17.6 Beispiel-Setup von Ingress mit RBAC .....	971
14.17.7 Erweiterung des bestehenden Ingress um einen zweiten Service .....	977
<b>14.18 Services, Service-Proxies, NetworkPolicies und mehr:</b>	
<b>Kube-Router to the rescue?</b> .....	<b>979</b>
14.18.1 Was macht das kube-router-Projekt anders? .....	980
14.18.2 Aufbau .....	981
14.18.3 Pre-Flight-Checkpunkte .....	982
14.18.4 Hairpin Mode .....	983
14.18.5 kubeadm-basiertes Setup .....	983
14.18.6 LB-Scheduler einstellen .....	986
14.18.7 Ingress? Nö, aber DSR .....	986
<b>14.19 Service Meshes</b> .....	<b>989</b>
14.19.1 Kube-proxy und K8s-Services vs. Service-Mesh .....	990
14.19.2 Service Mesh: Funktionsweise .....	992
14.19.3 Linkerd .....	995
14.19.4 Conduit bzw. Linkerd2 .....	998
14.19.5 Istio .....	1006
14.19.6 Fazit .....	1007
<b>14.20 Kubernetes-Volumes</b> .....	<b>1007</b>
14.20.1 K8s-Volumes im Unterschied zu Docker .....	1008
14.20.2 Persistente und nicht persistente Volumes .....	1009
14.20.3 emptyDir .....	1011
14.20.4 hostPath .....	1013
14.20.5 NFS .....	1014
14.20.6 iscsi-Volume .....	1016
14.20.7 Ceph .....	1017
14.20.8 PersistentVolumes und SAN .....	1017

14.20.9	PersistentVolumes in der Praxis .....	1019
14.20.10	Abstraktion und Access-Methoden .....	1027
14.20.11	Bindungsfragen zwischen PV und PVC .....	1028
14.20.12	Sharing eines NFS-PV über zwei Pods .....	1032
14.20.13	Storage Protection ab Kubernetes 1.11 .....	1035
14.20.14	K8s und Dynamic Storage Classes .....	1036
14.20.15	Beispiel für einen External NFS-Provisioner .....	1040
<b>14.21</b>	<b>PetSets, StatefulSets und Stateful Pods .....</b>	<b>1046</b>
14.21.1	Vorbetrachtungen .....	1046
14.21.2	Im Detail .....	1046
<b>14.22</b>	<b>HPA – Horizontaler Pod-Autoscaler .....</b>	<b>1048</b>
14.22.1	Hintergrund .....	1049
14.22.2	Auslastungskontrolle .....	1049
14.22.3	HPA-Setup .....	1050
14.22.4	Achtung: Änderungen in der Metrics-API in Kubernetes 1.9 und das Wiederherstellen der Funktionalität .....	1051
14.22.5	Exkurs: kube-state-metrics (metrics-server) vs. Heapster .....	1052
14.22.6	Test des HPA .....	1053
14.22.7	Last-Test .....	1056
14.22.8	Löschen des HPA-Objekts .....	1058
14.22.9	Kubernetes Addon: Cluster Proportional Autoscaling .....	1058
<b>14.23</b>	<b>Weitere K8s-Objekte und -Ressourcen .....</b>	<b>1060</b>
14.23.1	Events .....	1060
14.23.2	ThirdPartyResources bzw. CustomResourceDefinitions .....	1061
<b>14.24</b>	<b>K8s-Labels und -Constraints .....</b>	<b>1062</b>
14.24.1	Warum Label? .....	1063
14.24.2	Constraints .....	1063
14.24.3	Label-Planung .....	1064
14.24.4	Aufbau .....	1064
14.24.5	Ein paar praktische Beispiele .....	1065
14.24.6	Labels und Node/Pod-Affinity .....	1067
14.24.7	Affinity und Anti-Affinity .....	1069
14.24.8	Annotations .....	1070
<b>14.25</b>	<b>K8s-NetworkPolicies .....</b>	<b>1071</b>
14.25.1	Pod-Isolation .....	1072
14.25.2	Anwendungsfälle .....	1073
14.25.3	Performance-Impact? .....	1073
14.25.4	Test mit NetworkPolicy für httpd-Deployment .....	1074
14.25.5	Arten der ACLs .....	1076

<b>14.26 K8s-Authentifizierung und -Autorisierung .....</b>	1080
14.26.1 Kubernetes $\geq 1.6$ und ABAC/RBAC .....	1081
14.26.2 RBAC: Konzepte und Objekte .....	1082
14.26.3 ABAC: Berechtigungen (Verbs) .....	1084
14.26.4 Verbs-Varianten .....	1084
14.26.5 Admission Controls und der Admission Controller .....	1086
14.26.6 User- und Systemrollen .....	1089
14.26.7 Generelles zu Accounts .....	1091
14.26.8 Endlich externe Authentifizierungs-Provider? .....	1092
14.26.9 ServiceAccounts .....	1093
14.26.10 Secrets .....	1096
14.26.11 Mountable Secrets .....	1098
14.26.12 Ein einfaches RBAC-Beispiel mit einem Kubernetes-»User« .....	1101
14.26.13 User Account, Cluster, Namespaces und Contexts .....	1107
14.26.14 Authentifizierung an weiteren Clustern .....	1110
14.26.15 Selektives Löschen von Cluster und Kontexten .....	1110
14.26.16 Bestehenden kubernetes-admin-User für den neuen Namespace bzw. Context einrichten .....	1110
14.26.17 K8s-Authentifizierung gegen Azure AD .....	1112
14.26.18 Fazit .....	1112
<b>14.27 Die Operator-Ressource: Skalierung von Apps in Replikations-Setups out of the box? .....</b>	1113
14.27.1 Vorbetrachtungen zur Operator-Ressource .....	1113
14.27.2 Was ist ein Operator? .....	1113
14.27.3 Red Hats Operator Framework .....	1114
14.27.4 Scale-Out und das Pawlowsche-Hund-Syndrom auf Entscheider-Ebene ....	1115
14.27.5 Der etcd-Operator .....	1116
14.27.6 Funktionalitäten .....	1116
14.27.7 (Nicht-)Persistenz .....	1117
14.27.8 etcd-Operator in K8s $\geq 1.8$ .....	1117
14.27.9 Error-Handling und Scale-Up .....	1125
14.27.10 Scale-Up .....	1125
14.27.11 Upgrade bzw. Downgrade .....	1126
14.27.12 PVs und Backup/Recovery im etcd-Cluster .....	1127
14.27.13 Die Intelligenz im Operator .....	1128
14.27.14 Fazit .....	1129
<b>14.28 Helm-Deployments .....</b>	1129
14.28.1 Das Verfahren .....	1130
14.28.2 Helm-Architektur .....	1130
14.28.3 Ein Beispiel-Setup .....	1131
14.28.4 Charts finden, inspizieren und managen .....	1136

<b>14.29 Prometheus .....</b>	1138
14.29.1 Prometheus-Label .....	1140
14.29.2 In oder Out? .....	1141
14.29.3 Prometheus-Setup per Helm in einem SUSE-CaaSP-Cluster mit 6 Nodes .....	1141
14.29.4 Kube-State-Metrics .....	1144
14.29.5 Zugriff von außen .....	1145
<b>14.30 Zentrales Logging mit ElasticSearch, Fluentd und Kibana .....</b>	1148
14.30.1 Der EFK-Stack .....	1149
14.30.2 Pre-Flight-Requirements und Setup .....	1151
14.30.3 ElasticSearch-Deployment .....	1152
<b>14.31 Metrik-Erfassung im EFK-Stack .....</b>	1160
14.31.1 Implementierung des kube-metrics-server .....	1160
14.31.2 Kubernetes Metricbeat .....	1160

---

## **15 Federated- und geografisch verteilte K8s-Cluster**

---

<b>15.1 Vor betrachtungen .....</b>	1163
15.1.1 Wie funktioniert es? .....	1163
15.1.2 Federated Services .....	1165
15.1.3 On-Premise-Setup und andere Kopfschmerzen, Fazit .....	1166

---

## **16 K8s: Debugging, Rolling Upgrades, Fazit**

---

<b>16.1 Debugging/Troubleshooting .....</b>	1169
16.1.1 Grundsätzliches .....	1169
16.1.2 kubectl cluster-info dump .....	1170
<b>16.2 Rolling Upgrades des K8s-Clusters .....</b>	1170
16.2.1 Best Practices: Master-Nodes (VMs) .....	1171
16.2.2 Best Practices: Alle Nodes und kubelets (VMs) .....	1172
16.2.3 kube-proxy .....	1172
<b>16.3 Upgrades kubeadm-basierter Setups .....</b>	1172
16.3.1 Ab kubeadm bzw. K8s Version 1.6 .....	1172
16.3.2 Ab kubeadm bzw. K8s Version 1.8 .....	1173
16.3.3 kubeadm upgrade plan .....	1173
16.3.4 Upgrade Guidance .....	1176

## TEIL IV High-Level-Orchestrierungstools für Container-Infrastrukturen (on Premise und in der Cloud)

### 17 OpenShift

1179

<b>17.1 Vorberichtigungen und Historisches</b> .....	1179
17.1.1 All-in-one? .....	1179
17.1.2 OpenShift .....	1180
17.1.3 Werdegang .....	1180
17.1.4 Die Layer von OpenShift .....	1181
17.1.5 OpenShift-Vorteile für Entwickler und Admins .....	1182
17.1.6 OpenShift und OKD .....	1182
<b>17.2 OpenShift-Flavors</b> .....	1182
17.2.1 OpenShift Origin/OKD .....	1183
17.2.2 OpenShift Online (NextGen OpenShift) .....	1183
17.2.3 OpenShift Dedicated (Cloud Services) .....	1183
17.2.4 OpenShift Container Platform (OCP)/OpenShift Enterprise .....	1184
17.2.5 OpenShift unter Azure .....	1184
17.2.6 Docker-, CRI-O- und Kubernetes-Releases unter der Haube .....	1184
<b>17.3 Unterschiede und Ergänzungen zu Kubernetes</b> .....	1185
17.3.1 Vorbetrachtungen und Core Concepts .....	1185
17.3.2 Konfigurationsdaten .....	1186
17.3.3 Binaries .....	1186
17.3.4 Where's the Proxy? .....	1188
17.3.5 OpenShift Project's .....	1188
17.3.6 OpenShift Authentication und IDM .....	1189
<b>17.4 OpenShift Service Broker</b> .....	1190
17.4.1 Service Catalog to the rescue? .....	1191
17.4.2 Brokerage .....	1192
<b>17.5 OpenShift-Networking</b> .....	1192
17.5.1 ovs-subnet .....	1193
17.5.2 ovs-multitenant .....	1193
17.5.3 ovs-networkpolicy .....	1193
17.5.4 Third-Party Plugins .....	1194
17.5.5 Arbeitsweise der OpenShift-SDN-Plugins .....	1194
<b>17.6 OpenShift Router: Ingress made easy</b> .....	1196
17.6.1 Vorbetrachtungen .....	1196

17.6.2	OpenShift-Router und route – die Funktionsweise im Detail .....	1197
17.6.3	Router und Host-Ports .....	1198
17.6.4	HAProxy Template Router und eigene Implementierungen .....	1199
17.6.5	F5-Router-Plugin-Implementierung für OpenShift .....	1200

## 18 OpenShift-Setup

1201

---

<b>18.1</b>	<b>OpenShift Origin 3.9 unter CentOS 7.4 – RPM- sowie Ansible-basierte Installation .....</b>	1201
18.1.1	Pre-Flight-Requirements – oder: Die Prinzessin auf der Erbse .....	1201
18.1.2	VMs/Hardware: Memory- und Speicherplatz-Anforderungen .....	1205
<b>18.2</b>	<b>Einfache und geführte Single-Master-Installation .....</b>	1206
<b>18.3</b>	<b>OpenShift-Multimaster-Setup (3 Master, die alle ebenfalls Nodes sind) .....</b>	1208
18.3.1	Weitere Multimaster-spezifische Pre-Flight-Betrachtungen .....	1208
18.3.2	OpenShift-Multimaster-Setup (3 Master, die alle ebenfalls Nodes sind) – Rollout .....	1214
18.3.3	Auszüge möglicher Bugs bei der Installation und deren Behebung .....	1217
18.3.4	Rollout per /etc/ansible/hosts und ansible-playbook .....	1218
18.3.5	OpenShift-Deinstallation und Rollback .....	1221
18.3.6	Die Konfigurationsdateien von OpenShift Origin (Post-Rollout) .....	1221
18.3.7	Gestartete OpenShift-bezogene Services .....	1222
18.3.8	Ressourcen nach dem OpenShift-Deployment .....	1222
18.3.9	iptables-Rulesets .....	1223
<b>18.4</b>	<b>(Nachträgliche) Metrics-Konfiguration .....</b>	1224
18.4.1	Hawkular und Grafana .....	1224
18.4.2	Grundlegendes Metrics-Troubleshooting .....	1225
<b>18.5</b>	<b>Prometheus .....</b>	1225
18.5.1	Frische OpenShift-Cluster-Installation mit Prometheus .....	1225
<b>18.6</b>	<b>Setup eines Logging-Stacks (EFK) .....</b>	1226
18.6.1	Setup des EFK-Stacks .....	1227
<b>18.7</b>	<b>Späterer Join zusätzlicher (Worker-)Nodes .....</b>	1230
18.7.1	Vorbereitungen .....	1230
18.7.2	Tasks .....	1231
18.7.3	OpenShift mit Gluster Storage .....	1232

<b>19 OpenShift-Administration</b>	<b>1233</b>
<hr/>	
<b>19.1 CLI-Tools .....</b>	<b>1233</b>
19.1.1 oc – die OpenShift-CLI .....	1233
19.1.2 oc und die kubectl-CLI im Vergleich .....	1234
19.1.3 Ausrollen einer Applikation .....	1236
19.1.4 oc adm/oadm – Cluster-Management (Administrator-CLI) .....	1239
19.1.5 Der Befehl openshift* .....	1241
19.1.6 OpenShift-GUI .....	1241
<b>19.2 User-, Token- und Role-Management in OpenShift .....</b>	<b>1241</b>
19.2.1 User-Objekte unter OpenShift .....	1242
19.2.2 User-Privilegien und Rollenbindungen .....	1245
19.2.3 Cluster-Admin erzeugen .....	1247
19.2.4 LDAPPasswordIdentityProvider .....	1247
19.2.5 Active Directory-Anbindung .....	1252
19.2.6 Roles für (LDAP-)User .....	1254
19.2.7 Roles unter der Lupe .....	1254
<b>19.3 OpenShift-spezifische Ressourcen im Cluster .....</b>	<b>1256</b>
19.3.1 Login als Admin, Systemüberblick, Namespaces/Projects .....	1256
19.3.2 Objekte im Default-Namespace bzw. Project nach der Installation .....	1258
19.3.3 imagestream (is) im Namespace default .....	1259
19.3.4 deploymentconfig (dc) .....	1261
<b>19.4 OpenShift-Router in der Praxis .....</b>	<b>1263</b>
19.4.1 Der Router .....	1264
19.4.2 Backup und Restore des Default-Routers .....	1265
19.4.3 Wiederherstellung des Routers und der Default-Routen ohne Backup .....	1266
19.4.4 Multiple Router in einem Namespace bzw. auf einem Host? .....	1267
19.4.5 Weiteren Router per Node-Selector deployen .....	1268
19.4.6 HAProxy-Router: Strict SNI .....	1269
19.4.7 OpenShift-Routentypen .....	1269
19.4.8 Path based Routing – OpenShift Ingress .....	1269
19.4.9 Wildcard-Routing (Domain specific routing) .....	1270
19.4.10 IP-Whitelists für Routen .....	1272
19.4.11 Router-Sharding .....	1273
19.4.12 Weightings in Routen .....	1275
19.4.13 Route-Beispiel mit MySQL .....	1275
19.4.14 Single Master .....	1276
19.4.15 Multimaster mit keepalived und Subdomain-Setting .....	1277
19.4.16 Alternate (Route-)Backends und Weightings .....	1279
19.4.17 Setzen der LB-Modi .....	1281

19.4.18 Sticky Sessions .....	1281
19.4.19 Egress .....	1283
<b>19.5 Accounts, Berechtigungskonzepte und Constraints .....</b>	<b>1284</b>
19.5.1 SCC – OpenShift Security Contexts .....	1284
19.5.2 Arbeitsweise und Implementierung .....	1284
19.5.3 SCCs-Details abfragen .....	1285
19.5.4 Eigene SCCs definieren und Usern/Gruppen zuordnen .....	1286
19.5.5 Fehlende Berechtigungen bzw. den System-ServiceAccount zum anyuid-SCC setzen .....	1287
19.5.6 Benutzung des Default-ServiceAccounts in weiteren Namespaces .....	1289
<b>19.6 DeploymentConfig erzeugen, verwalten und versionieren .....</b>	<b>1289</b>
19.6.1 Beispiel-Setup .....	1290
19.6.2 Update und Rollback einer DeploymentConfig .....	1290
19.6.3 Nachträgliches Setzen von Liveness und Readiness Probes .....	1291
19.6.4 Route für eine DeploymentConfig setzen .....	1293
<b>19.7 OpenShift Internal Registry .....</b>	<b>1294</b>
19.7.1 Gesetztes Default-Routing für die Registry .....	1295
19.7.2 Persistent Storage .....	1295
19.7.3 dockerd-Push-Client-Settings für Registry-Nodes .....	1296
19.7.4 Login .....	1297
19.7.5 Push über Route, Privilegien und Tagging .....	1297
19.7.6 Gepushte Images via Imagestream zeigen lassen .....	1298
19.7.7 Die GUI der OpenShift-internen Atomic-Registry .....	1298
<b>19.8 OpenShift-Build-Prozesse .....</b>	<b>1299</b>
19.8.1 Docker-Build .....	1299
19.8.2 S2I (Source to Image) .....	1302
<b>19.9 Steuerung eines kompletten Container-Lifecycles:</b>	
<b>Build plus Rollout per OpenShift-GUI .....</b>	<b>1303</b>
19.9.1 Erzeugung der Applikation per Service Catalog .....	1303
19.9.2 Registry-Push und Rollout .....	1304
<b>19.10 Neuerungen in OpenShift 3.10 .....</b>	<b>1306</b>

---

## 20 Full-Featured Container-Security am Beispiel von OpenShift und NeuVector

1307

<b>20.1 Vor betrachtungen .....</b>	<b>1307</b>
<b>20.2 Architektur der NeuVector-Security-Lösung .....</b>	<b>1309</b>
20.2.1 Setup .....	1309

20.2.2 Setup von OpenShift .....	1309
20.2.3 Login und Verwaltung, Lizenzierung .....	1311

---

## 21 Cloud-Hosted Kubernetes 1315

<b>21.1 Vorberachtungen .....</b>	1315
21.1.1 Kubernetes, Google und der Rest der Welt .....	1315
21.1.2 Pressure .....	1315
21.1.3 Security .....	1316
21.1.4 Bedarf und Cloud-Angebote .....	1316
<b>21.2 GKE – Google Kubernetes Engine .....</b>	1317
21.2.1 Was ist im Angebot? .....	1317
21.2.2 Setup .....	1318
21.2.3 Installation nur per Klicki-Bunti? Jein .....	1321
21.2.4 Cloud-Shell ausführen .....	1321
<b>21.3 EKS – Amazons Elastic Kubernetes Service .....</b>	1322
21.3.1 Setup .....	1322

---

## 22 Fazit zur Container-Orchestrierung 1325

<b>22.1 Vanilla-Kubernetes .....</b>	1325
<b>22.2 OpenShift .....</b>	1326
<b>22.3 Cloud-Hosted Kubernetes: GKE .....</b>	1326
<b>22.4 Cloud-Hosted Kubernetes: EKS .....</b>	1326

# TEIL V Software-Defined Storage für verteilte Container-Infrastrukturen

---

## 23 Ab in den Untergrund 1329

<b>23.1 Software-Defined Storage (SDS) und Container .....</b>	1329
23.1.1 Vorberachtungen .....	1330
23.1.2 Cluster-Storage, Skalierbarkeit und der SPoF .....	1330

<b>23.2 SDS-Funktionsprinzipien .....</b>	1331
23.2.1 Software-Defined Storage – SDS .....	1331
23.2.2 Hoch? Oder lieber breit? Scale-Out anstelle von Scale-Up .....	1331
23.2.3 Traditionelle Storage-Cluster(-FS) vs. SDS .....	1332
23.2.4 Die Abstraktion .....	1332
23.2.5 SDS und Storage-Tiers .....	1333
23.2.6 Multi-Purpose-SDS .....	1333
23.2.7 Die roten Hüte, Ceph, Gluster und die Zukunft .....	1334
<b>23.3 Ceph .....</b>	1334
23.3.1 Ceph und RADOS .....	1335
23.3.2 librados .....	1336
23.3.3 Die Ceph-Daemons im Kurzüberblick: MON, OSD, MDS .....	1338
23.3.4 OSD .....	1338
23.3.5 MON .....	1339
23.3.6 MDS .....	1340
23.3.7 Ceph-Bereitstellungsverfahren für Container-Cluster .....	1340
23.3.8 Setup des Ceph-Clusters für die Container-Storage-Bereitstellung .....	1341
23.3.9 Ceph als SDS für K8s (RBD) .....	1346
23.3.10 Ceph als SDS für K8s (CephFS) .....	1354
23.3.11 Containerized Ceph .....	1356
<b>24 Einige Checkpunkte für Planung, Aufbau und Betrieb von Container-Clustern .....</b>	1357
<b>24.1 Evaluierung im Vorfeld .....</b>	1357
<b>24.2 Plattformauswahl .....</b>	1358
<b>24.3 Plattformen für den Testbetrieb und die spätere Produktion (On-Premise) .....</b>	1358
<b>24.4 Namespace-Design .....</b>	1358
<b>24.5 Build .....</b>	1359
<b>24.6 Registries .....</b>	1359
<b>24.7 Build und Rollout .....</b>	1360
<b>24.8 Welcher Orchestrierer? .....</b>	1360
<b>24.9 Betrieb und Administration .....</b>	1360
<b>24.10 Security .....</b>	1361

---

<b>25 Was war, was ist, was sein wird: Neue Meta-Ebenen und Verkomplizierung</b>	1363
<b>25.1 Container-Cluster und Microservices als Allheilmittel?</b> .....	1363
<b>25.2 The Road Ahead</b> .....	1364
 Index .....	 1367