

Skalierbare Container-Infrastrukturen

Das Handbuch für Planung und Administration

DAS INHALTS- VERZEICHNIS

» Hier geht's
direkt
zum Buch

Inhalt

1	Catch-22	39
1.1	Vorbemerkungen	43
1.1.1	Verwendete Formatierungen	44
1.1.2	Weiterführende Hinweise	44
1.1.3	Klartext	44
1.2	Kernziele und rote Fäden	45
1.3	Was dieses Buch sein soll und was nicht	46
1.4	Wie dieses Buch zu lesen ist	46
1.4.1	Neue Gliederung und Day 0-1-2-3 Operations	47
1.4.2	Weniger Detailschritte, Listings und Outputs, mehr Automation	47
1.4.3	Kapitel und Zielgruppen im groben Überblick	47
1.5	Docker-Replacement-Tools	48
1.5.1	Podman	48
1.5.2	Buildah	48
1.5.3	Skopeo	49

TEIL I Strategische Vorbetrachtungen, Foundations und Preflights

2	Grundsätzliche strategische Fragen	53
2.1	Worum geht es?	53
2.2	Überblick: Container- und Infrastruktur-Konzepte	54
2.2.1	Die Layer	55
2.2.2	Plattformen für Cluster-Nodes mit schlankem OS	56
2.2.3	Container-Engines	57
2.2.4	Die große Frage – Welcher Orchestrierer: Vanilla Kubernetes? OpenShift? Rancher? Tanzu?	57

2.3	Generelle Infrastruktur-Fragen: Cloud vs. On-Prem, Managed Kubernetes, Managed Server, hybrider Mischbetrieb	58
2.3.1	Konzeptioneller Aufbau und Planungsstrategien	58
2.3.2	Managed Kubernetes vs. Self-driven	59
2.3.3	Implementierungs- und Kostenfaktoren in der Cloud	60
2.3.4	Multi-Cloud-Implementierungen – Vor- und Nachteile	61
2.3.5	Exkurs: Managed Server kleinerer SPs als günstigere Cloud-Alternative mit höherer Flexibilität?	62
2.3.6	Implementierungs- und Kostenfaktoren: Self-Hosted	63
2.3.7	Datensicherheit	64
2.3.8	Storage und Netzwerk	64
2.3.9	Hybrider Ansatz: On-Premises und Cloud (Pay-per-Use)	65
2.3.10	Alles cool? In der Cloud oft eher nicht: Temperatur-, Performance- und damit Kostenfragen	65
2.4	Maximale Vollautomation – IaC, Operatoren, GitOps	66
2.4.1	Maximale Automation – in allen Bereichen	66
2.4.2	Kriterien	67
2.4.3	Konzepte: IaaS und IaC	68
2.4.4	Terraform und das Komplexitätsproblem	69
2.4.5	Vollautomation (Infrastruktur), Teil 1: IaaS und IaC	71
2.4.6	Vollautomation (In-Cluster), Teil 2: Operatoren	72
2.4.7	Vollautomation (In-Cluster), Teil 3: GitOps-Pipelines	72
2.5	Registries	73
2.5.1	Container-Image-Registries und -Repositories	73
2.5.2	Auswahlkriterien	74
2.5.3	Kandidaten	75
2.6	Ganzheitliche Security – High-Level View	76
2.6.1	Insel-Lösungen sind keine Lösung	76
2.6.2	Öffentliche Registries? Nein.	77
2.6.3	Trusted Images und Registries	78
2.6.4	Red Hat Registry	79
2.6.5	Automatisches Image Signing und Verify	82
2.6.6	Red Hats Universal Base Image (UBI)	82
2.6.7	Seccomp-Profil und Kernel-Capabilities	84
2.6.8	BSI-Richtlinien für Container-Cluster	85
2.6.9	Fazit	86

TEIL II Kubernetes-Architektur, Core-Concepts, Workloads und Day 1 Operations

3	Kubernetes	89
3.1	Kubernetes im Überblick	89
3.1.1	Aufgaben	90
3.1.2	Container-Engines	90
3.1.3	Skalierbarkeit	91
3.2	Vanilla Kubernetes und das traurige Thema LTS	91
3.2.1	Geld verbrennen? Oder besser doch nicht?	91
3.2.2	Vanilla Kubernetes: Releases, Changes und kein Ende	92
3.2.3	Vanilla Kubernetes und Derivate: LTS-Mankos, benötigte 3rd-Party Tools, asynchrone Produktzyklen und Märchenstunden	93
3.2.4	Vanilla Kubernetes: Fazit	94
3.2.5	VMware, Tanzu und das Eckige, das durchs Runde soll	95
3.2.6	AKS, EKS, GKE & Co.	96
3.2.7	OpenShift	97
3.2.8	Rancher	98
3.2.9	Übersicht über Enterprise-relevante Kernfeatures einiger Kubernetes-Plattformen	99
3.2.10	Fazit	100
3.3	Kubernetes-Komponenten	100
3.3.1	High-Level-Architektur	100
3.3.2	Master- bzw. Controlplane-Nodes	101
3.3.3	Worker-Nodes und GPU-Worker-Nodes	102
3.3.4	Infrastruktur-Nodes	102
3.4	Dienste auf allen Node-Typen: Kubelet, Container-Engine, Overlay-Netze, Proxies	103
3.4.1	High-Level: Kubelet und Container-Engine, Proxies, Overlay-Netze	103
3.4.2	Kubelet	103
3.4.3	Kubelets: Housekeeping und no-swap	104
3.4.4	kube-proxy und Alternativen	105
3.5	Dienste auf den Kubernetes-Master-/Controlplane-Nodes	106
3.5.1	API-Server	106
3.5.2	Controller-Manager	107
3.5.3	Node-Controller (kube-controller-manager) und die Verfügbarkeit der Worker-Nodes	108
3.5.4	Scheduler	110

3.5.5	Cloud-Controller-Manager	111
3.5.6	etcd	112
3.6	Etcd als Key/Value-Store in Kubernetes-basierten Clustern	112
3.6.1	Foundations und Arbeitsweise	113
3.6.2	Backup- und Verfügbarkeitsstrategien	113
3.6.3	HA-Aspekte	114
3.6.4	Implementierungsdetails	115
3.6.5	Best Practice – Anzahl der etcd-Instanzen und Fault-Tolerance	117
3.6.6	RZ-Topologien	118
3.7	Networking in Kubernetes	118
3.7.1	Vorbetrachtungen	118
3.7.2	Arbeitsweise	119
3.7.3	Kubernetes und IPv6 bzw. IPv4/IPv6-Dual-Stacks	119
3.7.4	Network-Plugins	119
3.8	Windows-Nodes in Kubernetes-Clustern?	121
3.8.1	Generelle Aspekte	122
3.8.2	Limitationen: OS-Version, Worker oder Master	123
3.8.3	Limitationen: Container-Engine	123
3.8.4	Limitationen: Netzwerk	124
3.8.5	Limitationen: Storage Probleme	125
3.8.6	Fazit	125
3.9	Container-Engines für Kubernetes	126
3.9.1	CRI – das Container Runtime Interface	126
3.9.2	CRI-O vs. cri-containerd vs. Docker	127
3.9.3	runC	127
3.9.4	CRI-O	127
3.9.5	Cri-Containerd	129

4 Kubernetes-Setup-Varianten im kompakten Überblick 131

4.1	Optionen und Grad der Verwaltbarkeit	131
4.1.1	Lösungen nach Grad der Eigen-Verwaltbarkeit	131
4.1.2	Setup-Tools nach Plattform-Diversität	132
4.2	Setup-Ansätze (Auszüge)	133
4.2.1	Cloud-basiert	133
4.2.2	OpenShift	134

4.2.3	Rancher	134
4.2.4	Minikube	134
4.2.5	Terraform nativ	135
4.2.6	kubeadm	135
4.2.7	Kubermatic	135
4.2.8	Kubespray	136
4.3	Zeitsynchronisation	137
4.4	Instance Sizing	137
4.4.1	Mindestanforderungen	137
4.4.2	Wenige große oder viele kleine Nodes?	138
4.4.3	Cloud-Instance-Calculator	139
5	Kubernetes-Cluster-Setups (Cloud)	141
5.1	GKE	142
5.1.1	GKE – Google Kubernetes Engine	142
5.1.2	Regionen, Zonen und Verfügbarkeiten	143
5.1.3	GKE-Setup-Varianten	144
5.1.4	GKE Shielded Nodes	144
5.1.5	Verfügbare Maschinen bzw. Instanz-Typen in der GCP	145
5.1.6	Mindestangaben für das Setup	145
5.1.7	Googles Node-OS-Varianten	146
5.1.8	Auszüge sonstiger einstellbarer Features	146
5.1.9	gcloud – CLI-basierte Cluster-Installation	147
5.1.10	gcloud init auf dem Verwaltungs-Node	148
5.1.11	Installierte/verfügbare gcloud-Komponenten auflisten bzw. nachinstallieren	149
5.1.12	Container-API und Billing im Projekt aktivieren	149
5.1.13	GKE-Channels (Stable, Regular, Rapid)	150
5.1.14	Cluster-Installation mit Anpassungen	152
5.1.15	GKE-Cluster und NetworkPolicies	154
5.1.16	Manuelles Cluster-Sizing/-Scaling	154
5.1.17	Einen Node-Pool oder Cluster löschen	155
5.1.18	Auth-Entries, Kontexte fetchen/switchen, Projekt setzen	156
5.1.19	X509-Zertifikatsfehler bei der Ausführung von kubectl	156
5.1.20	Die Google-Registry nutzen	156
5.1.21	Zugriff auf GKE-Worker-Nodes	157
5.2	EKS	158
5.2.1	Region, Zonen und Verfügbarkeiten	158

5.2.2	Instanzen und Preise	159
5.2.3	eksctl-CLI und Setup	159
5.3	AKS	160
5.3.1	Region, Zonen und Verfügbarkeiten	161
5.3.2	Instanzen und Preise, SLAs	161
5.3.3	Azure-CLI und Setup	162
5.4	Vergleichstabelle für Managed-Kubernetes-Angebote	164

6 Kubernetes: Deployment-Tools und -Konzepte, API-Foundations, Manifest- und CLI-Handling 165

6.1	Überblick: Tools zum Deployment von Kubernetes-Ressourcen	165
6.1.1	Die Qual der Wahl? Nicht wirklich	165
6.1.2	kubect!	166
6.1.3	Helm	166
6.1.4	kustomize	166
6.1.5	Operatoren	167
6.2	Helm und Kustomize – the Big Short	167
6.2.1	kustomize	169
6.2.2	Helm	171
6.3	Editoren und Tools: VI(M), Visual Studio Code und K9s	172
6.3.1	VI(M)	172
6.3.2	Visual Studio Code	173
6.3.3	K9s	176
6.4	Grundlegende Verfahren zum Erstellen von Workloads	176
6.4.1	Manifeste erzeugen	176
6.4.2	Mehrere Ressourcen in einem YAML-File	177
6.4.3	YAML-Manifest-Versionierung via Git	177
6.5	Grundlagen zu kubect!	178
6.5.1	kubect!-Bash-Completion, kubect!-Alias und kubect!-Caching	178
6.5.2	kubect!-Client/Server-Versionen	179
6.5.3	Die Konfiguration von kubect!	180
6.5.4	Die KubeConfig wurde gelöscht oder ist nicht mehr verfügbar	180
6.5.5	Plugins für kubect!	181
6.5.6	Syntax-Prüfungen für kubect! und YAML (Editoren und Pipelines)	181
6.5.7	High-Level-View: api-versions und api-resources verstehen und abfragen	181
6.5.8	kubect! api-versions	183

6.5.9	kubectl api-resources	185
6.5.10	Migrationen, Never-ending Stories: apiVersion- und Kind-Changes	186
6.5.11	kubectl explain	187
6.5.12	kubectl-Debugging im Verbose Mode	188
6.5.13	kubectl: Kontext/namespace-Switching	188
6.6	kubectl-Operations	188
6.6.1	kubectl get	189
6.6.2	kubectl describe	189
6.6.3	Multiple Ressourcen per kubectl anlegen, modifizieren und löschen	190
6.6.4	kubectl create	190
6.6.5	kubectl attach	191
6.6.6	Laufenden Pod bzw. laufende Ressource editieren: kubectl edit	191
6.6.7	kubectl patch	191
6.6.8	kubectl replace	193
6.6.9	Imperativ vs. deklarativ: kubectl create/delete/replace vs. kubectl apply/patch	193
6.6.10	kubectl set	194
6.6.11	kubectl wait	194
6.6.12	kubectl (force) delete (und Finalizers)	195
6.6.13	Logs von Containern eines Pods abfragen	196
6.6.14	kubectl exec – Remote Kommandos im Pod/Container ausführen	197
6.6.15	Server Side Apply und kubectl apply	197
6.6.16	Weitere kubectl-Subkommandos	198
6.7	Debugging von Kubernetes-Ressourcen	198
6.7.1	Pod-Debugging mit kubectl debug	198
6.7.2	Node-Debugging mit kubectl debug	199
7	Kubernetes-Cluster: Day 1 Operations – Core-Workloads	201
7.1	Namespaces: Foundations	201
7.1.1	Vorbetrachtungen	202
7.1.2	Funktionalitäten und Regeln bzw. Vorschriften für Namespaces	203
7.1.3	Standard-Namespaces in einem Vanilla-Kubernetes-Cluster	204
7.1.4	Uniqueness pro Namespace	205
7.1.5	Objekte mit und ohne Namespace-Zuordnung	205
7.1.6	Namespaces erzeugen	205
7.1.7	Namespace löschen	206
7.1.8	Wenn die Namespace-Löschung hängt	206

7.2	Namespaces: Multi-Tenancy- und Security-Aspekte	207
7.2.1	Namespaces/Networking – kundenspezifische und System-Namespaces	207
7.2.2	Geteilte Core-Komponenten	208
7.2.3	Problematiken mit Scheduling, Eviction, Preemption	208
7.2.4	Node-Fixing als Lösung?	209
7.2.5	Node Security	209
7.2.6	Logging/Monitoring	209
7.2.7	Wechselwirkungen mit Cluster-Autoscalern	209
7.2.8	Security-Lösungen	210
7.2.9	Haftung	210
7.2.10	Fazit	210
7.3	Pods und Container	210
7.3.1	Foundations	210
7.3.2	Überblick: Pods, Startup-Orderings, Init-Container und die Nonsense- Altlasten von Docker, Inc.	213
7.3.3	Einfache Pod-Manifeste	213
7.3.4	ImagePullPolicies für Container	215
7.3.5	Serielle und parallele Image-Pulls	216
7.3.6	Pod/Container-Phasen	217
7.3.7	Pod-RestartPolicies und Startverzögerung	218
7.3.8	Auszüge einiger Beispiele für mögliche Zustände von Pods	218
7.4	Pod-Sidecar-Patterns und das Applikations-Design	220
7.4.1	Klassischer Sidecar	221
7.4.2	Ambassador	221
7.4.3	Adapter	221
7.4.4	Initializer	221
7.5	Pods und Init-Container	222
7.5.1	Funktionsweise	222
7.5.2	Readiness	223
7.5.3	Anwendungsmöglichkeiten für Init-Container	223
7.5.4	Phasen des Init-Containers, mehrstufiges Init-Beispiel	223
7.5.5	Init-Container und Compute-Ressourcen	225
7.6	Pod- und Container-Security	225
7.6.1	SecurityContext für Container: Kernel-Capabilities und mehr	226
7.6.2	Pod Security Admission Controls	228
7.7	Pod-/Container-Attribute über Umgebungsvariablen nutzen	231
7.7.1	Pod- oder Container-Variablen?	231
7.7.2	Pod-Attribute auslesen und Variablen zuordnen	231
7.7.3	Erzeuge Ordner im Mountpath	233

7.8 Überblick: ConfigMaps, ServiceAccounts und Secrets	233
7.9 ConfigMaps	234
7.9.1 Funktionsweise	234
7.9.2 Ein einfaches Beispiel	235
7.9.3 Verfahren zur Nutzung in einem Pod	236
7.9.4 ConfigMap in einem Pod nutzen (ENV-Variante)	236
7.9.5 ConfigMaps als Volumes	238
7.9.6 Semi-Auto-Updates bei eingemounteten ConfigMaps	240
7.9.7 Eingemountete ConfigMaps oder lieber per ENV?	240
7.9.8 Varianten zur Erstellung	241
7.9.9 Exkurs: YAML-Multiline-Attribute (in ConfigMaps)	242
7.9.10 Erzeugung von ConfigMaps aus Files und Binary Data in ConfigMaps	243
7.9.11 Re-Deploy einer Ressource bei Änderung der ConfigMap	243
7.9.12 ConfigMaps des Controlplanes	244
7.9.13 Immutable ConfigMaps	245
7.9.14 Fazit zu ConfigMaps	245
7.10 ServiceAccounts	245
7.10.1 Konzept und Funktionsweise	245
7.10.2 Hands-On	246
7.10.3 Opt-Out Credential Automount	247
7.10.4 ServiceAccounts im System	248
7.10.5 ServiceAccount direkt mit ImagePullSecrets ausstatten	249
7.11 Secrets	249
7.11.1 Secrets in Pods verwenden	251
7.11.2 Docker-kompatible Secrets für den Registry-Zugriff	253
7.11.3 Secrets mit Zertifikaten: (Secret-)Auto-Rotation/Update?	254
7.11.4 Really Secret Secrets? Nicht wirklich	254
7.11.5 Secret-Synchronisation zwischen Namespaces	255
7.12 Jobs	255
7.12.1 Failure-Verhalten	256
7.12.2 Job-Beispiel	257
7.12.3 CronJobs	258
7.13 Label, Selektoren und Annotations	260
7.13.1 Label und Selektoren	260
7.13.2 Label für Constraints	261
7.13.3 Aufbau	261
7.13.4 Ein paar einfache, praktische Beispiele	263
7.13.5 Annotations	264

7.14 Deployments	265
7.14.1 Deployment-Aufbau und -Features	265
7.14.2 matchLabels/matchExpressions ab Kubernetes 1.16	267
7.14.3 Deployment im Überblick	267
7.14.4 Rolling Updates	269
7.14.5 Revisionshistorie? Guter Witz	270
7.14.6 kubectl rollout	271
7.14.7 Generische Log-Abfrage von Pods in Deployments (und DaemonSets oder StatefulSets)	271
7.14.8 Umgebungsvariablen in Deployments nutzen (Pod-Name-basierte Log-Ordner)	272
7.15 DaemonSets	275
7.15.1 Scheduling der Pods eines DaemonSets	275
7.15.2 Der Node kann nicht mehr? Kein Problem, das DaemonSet schon	276
7.15.3 DaemonSet-Beispiel	277
7.16 StatefulSets	279
7.16.1 Die Komponenten des StatefulSets in der Praxis	281
7.16.2 volumeClaimTemplates für StatefulSets	282
7.16.3 Praxisbeispiele	282
7.17 Entscheidungshilfe: Wann Deployment, wann DaemonSet, wann StatefulSet?	282
7.17.1 Leichtgewichtige (Stateless-)Applikationen irgendwo im Cluster: Deployments	282
7.17.2 Node-bezogene Applikationsinstanzen (gegebenenfalls mit Persistenzen): DaemonSets	283
7.17.3 Orderings, Namens- und Daten-Persistenzen: StatefulSets	283
7.18 Update-Strategien für Pods im Überblick	284
7.18.1 Rolling Update	284
7.18.2 Canary (Sukzessiver Schwenk/Traffic Weighting)	286
7.18.3 Blue/Green (»Ganz oder gar nicht«-Schwenk)	288
7.18.4 Fortgeschrittene Betrachtungen	289
7.19 Kubernetes: Autorisierung/RBAC	289
7.19.1 Vorbetrachtungen und Scope	289
7.19.2 Kubernetes und ABAC/RBAC	289
7.19.3 RBAC: Konzepte und Objekte	290
7.19.4 Berechtigungen (Verbs)	291
7.19.5 RBAC-Beispiel	291
7.19.6 Berechtigungen verwalten	292
7.19.7 User- und Systemrollen	293

7.19.8	Admission-Controls und Admission-Controller	295
7.19.9	Verfügbare Admission-Controller und Kontrollphasen	295
7.20	Kubernetes-Volumes und dynamische Storage-Provisionierung	296
7.20.1	Generelles: RWO vs. RWX	297
7.20.2	RWOP – ReadWriteOncePod	299
7.20.3	Persistente und nicht persistente Volumes	299
7.20.4	PersistentVolumes und PersistentVolumeClaims	301
7.20.5	StorageClasses	302
7.20.6	StorageClasses und ReclaimPolicies	304
7.20.7	Eine bestimmte StorageClass als Default setzen/löschen	305
7.20.8	Multiple Default-StorageClasses?	305
7.20.9	Provisioner	306
7.20.10	CSI: Container Storage Interface für Kubernetes	308
7.20.11	Plugin-spezifische AccessModes (Auszüge)	310
7.20.12	PV/PVC-Limits und »echte« Quotas	312
7.20.13	Forcierte Löschung von hängenden/»stuck« PVCs	313
7.20.14	Geschichtliches: Storage-Protection	313
7.20.15	Hands-On: NFS-Provisioner	314
7.20.16	Wiederverwendung von Recycled/Retained PVs	318
7.20.17	SDS-Volumes und Ceph	321
7.20.18	Topology Aware Dynamic Provisioning für PVs und VolumeBindingMode	321
7.20.19	Mount-Propagation	323
7.21	Storage für cloudbasiertes Kubernetes: GKE, EKS und AKS	323
7.21.1	GKE	323
7.21.2	EKS	326
7.21.3	AKS	327
7.22	Services	327
7.22.1	High-Level-Überblick	327
7.22.2	Die Service-Ressource	328
7.22.3	Technische Arbeitsweise der Service-Ressource	329
7.22.4	Service-Endpoints	331
7.22.5	Endpoint-Slices	332
7.22.6	Service für ein Blue/Green-Deployment	333
7.22.7	Service-Typen	335
7.22.8	Services ohne Selektoren	337
7.22.9	ServiceType ClusterIP ohne interne IP	337
7.22.10	Der Default-kubernetes-Service	338
7.22.11	MetalLB: Virtueller LoadBalancer für den ServiceType »LoadBalancer« in On-Premises-Clustern	339
7.22.12	Kubernetes-Services und DNS: interne Namesauflösung	343

7.22.13	Pod-DNS-Policies	344
7.22.14	NodeLocal DNSCache	346
7.22.15	Kubernetes-Services und Proxy-Modes (iptables, ipvs)	346
7.22.16	Runtime-Change des ipvs-Balancer-Modes	349
7.22.17	Session-Persistenzen für Services	350
7.22.18	LoadBalancerClass für Services angeben (stable ab 1.24)	351
7.22.19	Headless Services: Weiche Migration von Legacy-Systemen	352
7.22.20	Egress- und Firewall-Betrachtungen (Headless vs. URL via ConfigMap vs. Egress-IP) 355	
7.22.21	ExternalName-Services	356
7.22.22	Services und der Rest der Welt	357
7.23	Ingress	358
7.23.1	Grundlagen	358
7.23.2	Verfügbare Ingress-Controller, Vergleichstabelle	359
7.23.3	Protokolle und involvierte Komponenten	360
7.23.4	Ingress-Ressource (exemplarisch)	360
7.23.5	Ingress-Controller vs. API-Gateways vs. Service-Meshes	363

TEIL III Skalierbare Container-Cluster mit Kubernetes: Day 2 Operations

8 Day 2 Operations: In-Cluster-Vollautomation mit Operatoren – Foundations 367

8.1	Vorbetrachtungen: Zwei Operator-spezifische Hauptkapitel	367
8.2	CustomResourceDefinitions	368
8.2.1	Funktionaler Überblick	369
8.2.2	CRDs abfragen	370
8.2.3	CRDs, Operatoren und Controller	370
8.2.4	Imperativ oder deklarativ?	371
8.2.5	CRDs und Structural Schemas	372
8.2.6	CRDs vs. API-Server-Aggregation	373
8.2.7	Eigene CRDs erstellen	374
8.2.8	Descriptions für kubect! explain <crd>.<spec status>.<subattribute>.<...> hinterlegen	376
8.2.9	Eigener Sample-Controller mit CRD	378
8.2.10	Multiple CRD-Versionen	380
8.2.11	CRD-Lifecycle	381

8.2.12	Verwalten mehrerer CRD-Versionen	381
8.2.13	Hängende CRDs löschen (Finalizer)	381
8.3	Operatoren unter Kubernetes	382
8.3.1	Full-Lifecycle-Automation	383
8.3.2	Was ist ein Operator?	383
8.3.3	Controller-Loops	386
8.3.4	Operatorhub.io und OpenShift-Operatoren	386
8.4	Operator-Typen und Maturitäts-Level: Helm vs. Ansible vs. Go	387
8.4.1	Operator-Maturitäts-Level: 1 bis 5	387
8.4.2	Detaillierte Operator-Level-Einstufung	388
8.4.3	Helm vs. GitOps und Operatoren	390
8.5	Operator-Typen im funktionalen Vergleich: Ansible vs. Go	391
8.5.1	Unterschiede von Go- und Ansible-basierten Operatoren	391
8.6	Operator-Preflights: OLM – wer überwacht die Wächter?	392
8.6.1	OLM – Operator Lifecycle Manager: CRDs	393
8.6.2	OLM – Operator Lifecycle Manager: Operatoren und Registry	394
8.6.3	Installation des OLM (Operator Lifecycle Manager) – nur Vanilla Kubernetes!	395
8.7	Operator-Management	396
8.7.1	Operator-Management per CLI	396
8.7.2	OLM-Uninstall	398
8.7.3	Spezifische Version eines Operators installieren und über Upgrades behalten	398
8.7.4	Operator-Updates per Graph	399
8.7.5	Operator-Updates unter OpenShift	399
8.8	Hands on: PostgreSQL-Operator (Level 5)	401
8.8.1	Postgres	401
8.8.2	Der Postgres-Operator	401
8.8.3	Verfügbare Versionen	403
8.8.4	Hochverfügbarkeit und Datenreplikation	403
8.8.5	Setup	403
8.8.6	Der Zustand nach dem Rollout	407
8.8.7	Crash-Simulation	408
8.8.8	Skalierung	408
8.8.9	Upgrade	408
8.8.10	Autoscaling	410
8.8.11	War es das zu (L5-)Operatoren?	410

9	Kubernetes-Cluster: Day 2 Operations – Pod-Lifecycle, De-Scheduling, Tenancy und Limits	411
9.1	Pod-Lifecycle und Health-Checks	411
9.1.1	Die Notwendigkeit der Probes	411
9.1.2	Ready? Live? Startup? – Welche Probe wofür?	412
9.1.3	Auswirkungen der Liveness-Probes: Container Recreate	413
9.1.4	Liveness- und Startup-Probe-Nonsense	413
9.1.5	Auswirkung der Readiness-Probes: Remove Endpoint from Service List	414
9.1.6	Probe-Verfahren: exec, TcpSocket, HttpGet, gRPC	416
9.1.7	Monitoring-Intervalle, Timeouts und Thresholds	418
9.1.8	Failure- und Success-Thresholds	418
9.1.9	Timeouts, Initial Delays und Delays von Init-Containern mit einkalkulieren	419
9.1.10	Beispiele für eine generische Probe	419
9.1.11	Update-Fehler und minReadySeconds (Deployments und DaemonSets, ab Version 1.22 auch für StatefulSets)	421
9.1.12	Startup-Probe	422
9.1.13	Design-Flaws	424
9.1.14	Pod Readiness Gate	424
9.1.15	Pod-Lifecycle-Management und postStart-/preStop-Hooks	425
9.1.16	Pod Network Readiness und Pod Scheduling Readiness	427
9.2	(De-)Scheduling: Überblick	429
9.2.1	Scheduler: Default-Verteilungsstrategien	429
9.2.2	Scheduler-Algorithmen: Predicates und Priorities	430
9.2.3	Scheduling-Policies	431
9.2.4	Scheduling-Prozess: Node-Selection im Scheduler	433
9.2.5	KubeSchedulerConfiguration	436
9.2.6	Scheduler-Deep-Dive	437
9.2.7	Nominated Nodes	437
9.3	(De-)Scheduling: Constraints – Node-Selektoren, Pod Topology Spread Constraints	438
9.3.1	Node-Selektoren	438
9.3.2	Pod Topology Spread Constraints	440
9.4	(De-)Scheduling: (Anti-)Affinity, Taints und Tolerations	443
9.4.1	Überblick: Taints, Tolerations und Node/Pod(Anti-)Affinity	443
9.4.2	Mehrere Namespaces für PodAffinity: NamespaceSelector	445
9.4.3	Welchen Effekt haben Taints und Tolerations?	446
9.4.4	Taints/Tolerations-Beispiel	447

9.4.5	Taints abfragen, löschen	448
9.4.6	Typische Anwendungsfälle für Affinities, Taints und Tolerations: GPU-Nodes, Node-Problems	449
9.5	(De-)Scheduling: QoS-Classes, Compute Resource Requests und Limits	450
9.5.1	CPU-Requests und -Limits	450
9.5.2	Memory-Requests und -Limits	452
9.5.3	GPU-Limits	453
9.5.4	QoS – Quality-of-Service-Klassen im Überblick	454
9.5.5	QoS-Klassen im Detail	455
9.5.6	QoS-Klassen und Eviction	458
9.5.7	QoS-Klassen und systemd-Slices	458
9.5.8	Exklusives CPU-Pinning	459
9.5.9	Default-Requests und -Limits	460
9.5.10	QoS für Memory-Ressourcen (ab v1.22)	460
9.5.11	In-Place Compute-Resource Change (ResizePolicy) – in der Theorie	461
9.5.12	PID-Limits (seit v1.14)	463
9.5.13	(Requests und) Limits für Ephemeral Storage	463
9.5.14	Pod-Overhead (Compute-Ressourcen) per RuntimeClass	464
9.5.15	OOM-Scores	467
9.5.16	DRA – Dynamic Resource Allocation	469
9.6	(De-)Scheduling: Pod-Priorities	469
9.6.1	Pod-Prioritäten und Preemption	469
9.6.2	Wie verhalten sich QoS-Klassen und Priorities zueinander?	473
9.7	(De-)Scheduling: PodDisruptionBudgets	474
9.7.1	PDBs im Detail	475
9.7.2	PDB bei Node-Drainings	475
9.7.3	PDBs bei Rolling Upgrades	476
9.7.4	Technische Arbeitsweise des PDB	476
9.7.5	Best Practices	477
9.7.6	Praxisbeispiel mit separatem Node-Pool (GKE)	478
9.7.7	PDB und VPA	483
9.8	(De-)Scheduling: Node-Kapazitäten	483
9.8.1	Analyse der Node-Kapazität	483
9.8.2	Korrespondierende Kubelet-Direktiven bzw. Kubelet-Config-File	484
9.9	De-Scheduling und HA-Abstinenz: Descheduler und Re-Balancing	485
9.9.1	Der Teufel in den immer komplexeren Details	485
9.9.2	Ursachen, involvierte Komponenten und die Auswirkungen	486
9.9.3	Simpler Node-Crash mit Stateless oder Stateful Pods (mit PV)	487
9.9.4	Betrachtungen im Detail und ernüchternde Hintergründe	487
9.9.5	Der Descheduler als Lösung?	488

9.9.6	Non-Graceful Node-Shutdown-Recovery	489
9.10	Namespaces und (Compute-)Resource-Limits	490
9.10.1	LimitRanges vs. ResourceQuota	490
9.10.2	LimitRanges für Namespaces in der Praxis	491
9.10.3	Node- und Pod-spezifische Compute-Ressourcen anzeigen (pro Namespace)	495
9.10.4	ResourceQuotas für Namespaces	496
9.10.5	Object Count Quota	498
9.10.6	ResourceQuota-Scopes und PriorityClasses	498
9.10.7	Das typische PriorityClass-Problem und eine Lösung per ResourceQuota	500
9.10.8	LimitRanges und ResourceQuotas – Fazit und the Big Picture	501
9.11	Namespaces und NetworkPolicies	501
9.11.1	Vorbetrachtungen	502
9.11.2	Pod-Isolation	502
9.11.3	Achtung: namespaceSelector, podSelector und Arrays mit - from oder - to	504
9.11.4	ipBlocks und NATting	505
9.11.5	Performance-Impact	506
9.11.6	Achtung: GKE-Cluster und NetworkPolicies	506
9.11.7	Test mit NetworkPolicy	507
9.11.8	Calico und GlobalNetworkPolicies	510
9.11.9	Fazit	511
10	Kubernetes-Cluster: Day 2 Operations – DNS, Certificates, API-Gateways	513
10.1	ExternalDNS für externe Hostnamenauflösung	513
10.1.1	Funktionsweise	514
10.1.2	Unterstützte DNS-Systeme	515
10.1.3	GKE-Preflight: Cluster-Setup mit scopes	516
10.1.4	GKE-Preflight: Cloud-DNS	517
10.1.5	Setup des ExternalDNS	518
10.2	Automatisierte Zertifikatserzeugung (alle Plattformen): Cert-Manager	519
10.2.1	Cert-Manager-Releases und Kubernetes-Versionen	520
10.2.2	Cert-Manager-CRDs	521
10.2.3	Die HTTP-01- und DNS-01-Challenges	521
10.2.4	Cert-Manager-Workflow und Certificate Lifecycle	523
10.2.5	Setup-Preflights für den Cert-Manager (GKE)	523

10.2.6	Setup des Cert-Managers für Google Cloud DNS	524
10.2.7	Beispiel-Setup: Ingress mit ExternalDNS und per Cert-Manager ausgestelltem Zertifikat	525
10.2.8	Wildcard Certificates	529
10.2.9	Cert-Manager debuggen	531
10.2.10	Cert-Manager unter OpenShift	531
10.2.11	Exkurs: Automatisierte Zertifikatserzeugung (GKE-spezifisch) – ManagedCertificates	534
10.3	Gateway-API	536
10.3.1	Gateway-API: nur sechs Jahre zu spät	537
10.3.2	Funktionales	538
10.3.3	Gateway-API-Konzepte	539
10.3.4	Routing-Beispiele, -Guides und weitere Details	540
10.3.5	GKE-Implementierung der Gateway-API	540
10.4	API-Gateway: Foundations	541
10.4.1	Funktionales	541
10.4.2	API-Gateway-Features	542
10.5	API-Gateway: Beispiel-Setup (GKE)	543
10.5.1	GKE-Beispiel-Setup (Single Cluster)	543
10.5.2	Vorbereitendes GCP-Setup und Rollout des Gateways	545
10.5.3	Demo-Anwendung	547
10.5.4	Zugriff auf die Backends	549
10.5.5	Zugriff über externes Gateway	550
10.5.6	Löschung	551
10.6	API-Gateway: Beispiel-Setup mit Kong (alle Plattformen)	551
10.6.1	Setup-Vorbetrachtungen	551
10.6.2	Kong-Setup per Helm	553
10.6.3	Setup einer einfachen Test-Applikation mit HTTPRoute	555
10.6.4	HTTPRoute mit multiplen Backends und Weights	558
10.6.5	ExternalDNS und Gateway-Objekte	559
10.6.6	Plugins (pro Route) hinzufügen	560
10.6.7	Kong auf OpenShift	561
11	Kubernetes-Cluster: Day 2 Operations – Metrics, Monitoring, Logging, APM/Observability, Autoscaler	563
11.1	Kubernetes-Standard-Metriken: Metrics Server und kube-metrics	563
11.1.1	Metrics	563

11.1.2	Metriken im API-Server per kubectl get direkt abfragen	565
11.2	Log-Erfassung und mehr unter Kubernetes: Elastic	566
11.2.1	Elasticsearch vs. OpenSearch	567
11.2.2	Der Elastic Stack und Elasticsearch	567
11.2.3	Bereitstellungsverfahren	568
11.2.4	Elastic Agent	568
11.2.5	Fleet (Server)	569
11.2.6	Kibana	570
11.2.7	Elastic Enterprise Search	570
11.2.8	Setup des Elastic Operators	570
11.2.9	Setup des Elastic-Clusters und zugehöriger Komponenten	572
11.2.10	Rollout des Elastic-Clusters	578
11.2.11	Node-Affinity und Zone-Awareness der Elastic-Instanzen	579
11.2.12	Upgrades des ES-Stacks per Operator, Built-in PDBs	581
11.2.13	HA-Überlegungen zum ES-Upgrade	581
11.2.14	Kibana, APM und andere Module	582
11.3	Log-Erfassung und mehr unter Kubernetes: Loki – Grafana-Logging	584
11.3.1	Loki	584
11.3.2	Funktionsweise und Architektur	585
11.3.3	Loki-Setup unter Vanilla Kubernetes	586
11.3.4	Loki unter Red Hat OpenShift (OCP): Preflights	587
11.3.5	Loki unter Red Hat OpenShift (OCP): Setup	587
11.4	Cluster-Monitoring mit Prometheus	589
11.4.1	Aufbau und Funktionsweise	590
11.4.2	Messwerte: Gauge, Counter, Histogramm und Summary	591
11.4.3	Prometheus-Komponenten im Überblick	592
11.4.4	Prometheus-Operator und kube-prometheus	594
11.4.5	Prometheus-Installation und Betrieb – Vorbetrachtungen	595
11.4.6	Setup per kube-prometheus	596
11.4.7	Post-Rollout	600
11.4.8	Externer Zugriff auf die UIs	601
11.4.9	Setup einer Example-App	601
11.4.10	PodInfo mit ServiceMonitor	604
11.4.11	Service-Monitore für infrastrukturelevante Stacks einrichten: Ingress	604
11.4.12	Debugging des Service-Monitors	607
11.4.13	Alertmanager und Alert-Receiver	608
11.4.14	Eigene Alerting-Rules hinzufügen	612
11.4.15	Prometheus-HA, Scaling und Sharding	613
11.5	Federated Prometheus mit Thanos	615
11.5.1	Thanos	615

11.5.2	Die Komponenten	616
11.5.3	Setup-Vorbetrachtungen: Sidecar vs. Receiver	617
11.5.4	Vergleich der beiden Ansätze und Empfehlungen	618
11.5.5	Thanos-Setup (Sidecar)	619
11.5.6	Prometheus-Operator: Sidecars und Storage	620
11.5.7	Abschließende Betrachtungen	624
11.6	Tracing mit Jaeger	625
11.6.1	Die Mankos unter der Haube	625
11.6.2	Setup-Varianten	626
11.7	Full-Stack-Monitoring: APM und Observability	627
11.7.1	Monitoring vs. echte Observability	628
11.7.2	Dynatrace	630
11.7.3	Instana	636
11.7.4	New Relic und Datadog	638
11.7.5	High-Level-Vergleich zwischen Instana, Dynatrace, Datadog und New Relic	639
11.8	HPA – Horizontaler Pod-Autoscaler	640
11.8.1	Auslastungskontrolle	641
11.8.2	Thresholds, Resource-Requests und Metrik-Auswertung	642
11.8.3	Autoscaler-Metrics (CPU) im Detail	642
11.8.4	Parameter zur selektiven Steuerung des Ansprechverhaltens im HPA-Objekt	644
11.8.5	Lastgeneratoren: ab, nghttp2, Gatling & Co.	644
11.8.6	HPA-Scaling über CPU-Load mit Scale-Up/-Down-Behavior	644
11.8.7	Scaling nach einzelnen Containern pro Pod: ContainerResource	647
11.8.8	Horizontales Autoscaling über Custom-Metrics	648
11.8.9	Kubernetes-Addon: Cluster Proportional Autoscaling (Scaling Infra-Deployments)	648
11.9	Custom-Metrics-Autoscaling mit KEDA und HPA	650
11.9.1	Vorbetrachtungen	651
11.9.2	Event-Driven Scaling	653
11.9.3	Komponenten	653
11.9.4	ScaledObjects	654
11.9.5	Setup	656
11.9.6	KEDA-Autoscaling (Queue Length) für RabbitMQ	657
11.9.7	KEDA: HTTP-Requests-Trigger über den Prometheus-Scaler	661
11.9.8	PromQL	664
11.10	Vertical Pod Autoscaler	665
11.10.1	Vorbetrachtungen	665
11.10.2	Vertical Pod Autoscaler im Detail	666

11.10.3	Limitierungen und Nachteile	667
11.10.4	Datenerfassung, VPA-Checkpoints vs. Prometheus-Backend, Polling-Intervalle	667
11.10.5	VPA-Komponenten und Modi	668
11.10.6	Manuelles Setup	669
11.10.7	Setup: GKE (Built-in)	670
11.10.8	Beispiel-Setup	670
11.10.9	VPA-Recommendations und LimitRanges	674
11.10.10	Single-Pod-Deployments	675
11.10.11	Prometheus als VPA-Verlaufs-Backend	675
11.10.12	Uninstall	676
11.10.13	VPA unter OpenShift 4.12+	676
11.11	Multidimensionales Pod-AutoScaling (GKE)	678
11.12	Cluster-Autoscaling	678

12 Kubernetes-Cluster: Day 2 Operations – Meshes, Authentication, Debugging, Backup/Recovery 681

12.1	Service-Meshes	681
12.1.1	Überblick	682
12.1.2	Service-Mesh: Konzepte und Funktionsweise	684
12.1.3	Mesh-Benefits	685
12.1.4	Evaluierung	686
12.1.5	Service-Meshes verschiedener Anbieter im High-Level-Vergleich	688
12.1.6	Feature und Performance vergleichen	690
12.1.7	Meshes – ein Fazit	691
12.2	Kubernetes: Authentifizierung und Autorisierung (Keycloak-basiert)	692
12.2.1	Authentifizierung und Benutzer in Vanilla Kubernetes? Nicht wirklich	692
12.2.2	Keycloak	693
12.2.3	Keycloak, OIDC, OAuth2 (Proxy) und Kubernetes	695
12.2.4	Authentifizierungs- und Autorisierungs-Workflow mit und ohne OAuth Proxy	697
12.2.5	Keycloak-Setup	698
12.2.6	Externe Datenbank im Postgres-Cluster	700
12.2.7	Keycloak-UI-Login und Administration	701
12.2.8	Keycloak als Single-Sign-On-Lösung für OpenShift einrichten	703
12.2.9	Externe IDP anbinden	707
12.2.10	Applikation mit RHSSO/Keycloak authentifizieren	707

12.3 Debugging und Troubleshooting	709
12.3.1 Link-Aufstellung	709
12.3.2 kubectl cluster-info dump	710
12.4 Backup und Disaster-Recovery	710
12.4.1 High-Level-Betrachtungen	710
12.4.2 Resilienz durch Multi-Cluster- bzw. Multi-Cloud-Strategien	711
12.4.3 Backup/DR vs. GitOps/IaC-Recovery	711
12.4.4 Backup-Software für Kubernetes-basierte Systeme	713
12.4.5 Backup/DR per Kasten	715
12.4.6 Backups für GKE	719

TEIL IV Vollautomation und Resilienz mit eigenen Operatoren

13 Day 3 Operations: In-Cluster-Vollautomation mit Operatoren – Advanced Concepts 723

13.1 Operator-SDK, OLM und weitere Konzepte	723
13.1.1 Red Hats Operator-Framework	723
13.1.2 Vorbetrachtungen zum Operator-SDK	724
13.1.3 Operator-Build-Automation durch Pipelines	725
13.1.4 Operator-Bundle	725
13.1.5 Customisierbare Functional-Tests	726
13.1.6 Vorbetrachtungen und Preflights zum Operator-Build	726
13.2 Ansible oder Go?	727
13.2.1 Die Qual der Wahl? Nicht in jedem Fall	727
13.2.2 Go-basierter Operator: Überblick	728
13.2.3 Ansible-basierter Operator: Überblick	729
13.2.4 Operator-SDK-Setup	731
13.2.5 Vorbereitungen: Weitere Tasks für ALLE Operator-Typen (Go, Ansible)	732
13.3 Operator-Build-Demo: Level-5-Operator in Go	733
13.3.1 Preflights	734
13.3.2 Build-Foundation: Scaffold-Erzeugung	735
13.3.3 Build-Foundation: Scaffold-Erweiterung – API- und Controller-Templates ...	737
13.3.4 Types- und Controller-Bibliothek	739
13.3.5 Make (generate, manifests)	739
13.3.6 Optional: Quickrun/Testlauf per make install run	740
13.3.7 L5-Demo-Operator-Image erzeugen und hochladen	741
13.3.8 Build & Push	741
13.3.9 Test-Rollout des Operators per make deploy	742

13.3.10 Undeploy	745
13.4 Operator-Bundle für den L5-Operator erzeugen	745
13.4.1 Operator-Bundles	746
13.4.2 Hands-On: Bundle	747
13.4.3 Run bundle	748
13.5 Index/Catalog (für L5-Operator und andere) erzeugen	749
13.5.1 Vorbetrachtungen	749
13.5.2 OPM (Operator Package Manager)	751
13.5.3 File-Based Catalog für Operator-Bundles erzeugen	751
13.5.4 Exkurs: SQLite	753
13.5.5 CatalogSource erzeugen	753
13.5.6 Subscription	754
13.6 Hands-On: Memcached-Operator mit Ansible	756
13.6.1 Hands-On: Operator-Image	756
13.6.2 Hands-On: Bundle	757
13.6.3 Hands-On: Index-Image und CatalogSource um weitere Operator-Packages ergänzen	759
13.7 Diverses	760
13.7.1 Operatoren in Disconnected oder Air-Gapped Environments	760
13.7.2 Weitere Informationen zu Operatoren	761

TEIL V High-Level-Setup- und Orchestrierungs-Tools für Kubernetes-basierte Container-Infrastrukturen

14 Red Hat OpenShift 765

14.1 Vorbetrachtungen und Historisches	765
14.1.1 OpenShift	765
14.2 Lizenzierung und Lifecycle	767
14.2.1 Das »kostenlose« Vanilla Kubernetes und »billige« Cloud-Lösungen	767
14.2.2 Self-Managed OpenShift: SLAs nach Knoten-Typen und OpenShift-Ausstattung	768
14.2.3 Self-Hosted/Self-Managed OpenShift – leider ohne flexible Lizenzierung	769
14.2.4 Managed-OpenShift-(Cloud-)Angebote und On-Demand-Abrechnung über Provider	769
14.2.5 OpenShift-Lifecycles, CRI-O- und Kubernetes-Releases im Unterbau	770

14.3	OpenShift, das Enterprise-Kubernetes in »ready to use«	773
14.3.1	Unterschiede und Ergänzungen zu Kubernetes (Auszüge)	773
15	OpenShift-Setup	775
15.1	Generelle Vorbetrachtungen und Vorbereitungen	775
15.1.1	Genereller Tool-Hinweis zu allen folgenden Setups (AWS, vSphere, GCP & Co.)	775
15.1.2	OpenShift auf Bare Metal oder VMs installieren: Assisted Bare Metal Installer	776
15.1.3	Benötigte Internet-Zugriffe	776
15.1.4	OpenShift-Version (IPI) und oc	777
15.1.5	Der OpenShift-Installer: Terraform included	777
15.1.6	Überblick: Bootstrapping- und Installationsprozesse im Cluster	779
15.1.7	Exkurs: Konzept – RHCOS und Ignition	779
15.1.8	RHEL oder CoreOS (RHCOS)	781
15.1.9	Setup-Varianten: interaktiv oder per install-config.yaml	782
15.1.10	Präferierte Variante: customisiertes Setup über install-config.yaml	782
15.1.11	Anpassungsmöglichkeiten der install-config.yaml	783
15.1.12	Cluster-Capabilities	785
15.1.13	Beispielkonfiguration: OpenShift 4.12 auf AWS	785
15.1.14	Test-Cluster ohne Compute-/Worker-Nodes	787
15.1.15	Master-Nodes manuell als schedulable kennzeichnen	787
15.1.16	Fallstricke bei der Installation	787
15.1.17	MachineSets und Cluster-(Auto-)Scaler	788
15.1.18	(Fehlgeschlagene) Installation komplett löschen	788
15.1.19	Löschung eines per IPI ausgerollten OpenShift-Clusters ohne Terraform-Files	789
15.1.20	Backup der Credentials, die beim Rollout erzeugt werden	789
15.2	Setup von OpenShift 4.12 (IPI) auf AWS	789
15.2.1	DNS-Setup	790
15.2.2	AWS-User und Berechtigungen, IAM Access Analyzer	790
15.2.3	Account-Limits und -Requests, VPCs und IP-Range	791
15.2.4	Instanz-Typen, Leistungs- und Kostenfaktoren, Accelerated Computing	792
15.2.5	Credentials, Pull-Secrets, install-config.yaml, Multi-Cluster-Setup	793
15.2.6	Rollout	793
15.3	Setup von OpenShift 4.12 (IPI) auf GCP	795
15.3.1	Kontingente gegebenenfalls erhöhen	795
15.3.2	Domain, DNS und APIs	795

15.3.3	Service-Account zur OpenShift-Cluster-Erzeugung	796
15.3.4	Anpassungen in der install-config.yaml	797
15.4	Setup von OpenShift 4.13 (IPI) auf vSphere	798
15.4.1	Preflights für das folgende Setup	798
15.4.2	DNS und DHCP	799
15.4.3	vSphere-HA und openshift-install-Fehler	800
15.4.4	Zones und FailureDomains unter VMware (ab 4.13 GA)	800
15.4.5	install-config.yaml für vSphere-IPI-Installation (Zone-Aware)	801
15.4.6	Rollout	804
15.4.7	Änderungen der vSphere-Credentials und Konfiguration nach dem Rollout	806
15.4.8	Zusätzlicher vSphere-Datastore über zusätzliche SC	807
15.5	Post-install Tasks und Day 2 Operations für OpenShift	808
15.5.1	Built-in Registry und Registry-Operator	808
15.5.2	Zertifikatsrotation einmalig nach 25 Stunden und periodisch nach 30 Tagen	809
15.5.3	MachineSet Scaling / RHCOS Template Firstboot Error "Ignition: no config provided by user"	809
15.5.4	Node-/Hostname von Workern wird nicht gesetzt (Fallback auf localhost) ...	810
15.5.5	Zugriff auf die RHCOS-Nodes	810
15.5.6	SSH-Keys der RHCOS-Nodes nach der Installation ändern oder neu hinzufügen	811
15.5.7	KubeConfig nicht mehr verfügbar	811
15.6	Disconnected/Air-Gapped-Installation und der Betrieb	813
15.6.1	Registries	813
15.6.2	Spiegelung	815

16 OpenShift-Administration 819

16.1	CLI-Tools	819
16.1.1	Login als kubeadmin oder system:admin	819
16.1.2	oc – die OpenShift-CLI	820
16.1.3	Aktivierung der Bash-Completion für die oc-*-Tools	821
16.1.4	Die wichtigsten oc-Kommandos im Überblick	821
16.2	Administration per GUI	823
16.2.1	Administrator-View	823
16.2.2	Developer-View	824

16.3	OpenShifts Cluster-Operatoren	824
16.3.1	Funktion	825
16.3.2	Verfügbare COs	825
16.3.3	Konfiguration der Core-Stacks, die von Cluster-Operatoren betreut werden	826
16.4	OpenShift-Networking im Überblick	826
16.4.1	OpenShift SDN – Network-Plugins	826
16.4.2	Network-Management per Operator und Namespace-Isolation	828
16.5	Authentifizierung und Autorisierung unter OpenShift	830
16.5.1	Authentifizierung unter OpenShift	830
16.5.2	Der integrierte Oauth-Server	830
16.5.3	Externe Identity-Provider (LDAP/AD)	831
16.5.4	Automatisch LDAP-Gruppen aus einem Verzeichnisdienst syncen	833
16.6	Authentifizierung und Autorisierung: Security Context Constraints	834
16.6.1	Funktionsweise	835
16.6.2	Strategien und Auswirkungen	836
16.6.3	Aufschlüsselung der SCC Zugriffsregelwerke (Auszüge)	837
16.6.4	SCC einsetzen	838
16.6.5	Das Label openShift.io/run-level im Namespace und SCC-Deaktivierung/-Bypassing	840
16.7	Imagestreams	841
16.7.1	Vorbetrachtungen	841
16.7.2	Imagestreams im Detail	842
16.7.3	Der Re-Deployment-Trigger	844
16.8	OpenShift-Router	845
16.8.1	Funktionsweise	846
16.8.2	Ingress-Controller: Routen, Ingress und Host-Ports	847
16.9	OpenShift-Router: Ingress-Operator und Ingress-Controller	847
16.9.1	Konfiguration des Routers/Ingress-Controllers per Ingress-Operator	848
16.9.2	Routen-Typen	850
16.9.3	Routen: Alternate Backends und Weightings	851
16.9.4	Routen: Ingress-Zertifikate und Zertifikats-Workflow des Ingress-Controllers und -Operators	853
16.9.5	Eigene Zertifikate für OpenShift-Router	856
16.9.6	Routen-spezifische Annotations: LB-Modes, Stickiness und mehr	857
16.9.7	Sticky Sessions und LB-Modes	858
16.9.8	Managed Routes automatisch über Ingress-Objekte erzeugen	859
16.9.9	Router-Sharding	861

16.10 Egress-Limitierung und Priorisierung	864
16.10.1 Namespace-spezifische Egress-IPs	864
16.10.2 Egress-Firewall	867
16.10.3 Egress-Firewall-Policy konfigurieren	868
16.10.4 Egress-Traffic-Priorisierung mit QoS-DSCP und EgressQoS	869
16.11 DNS-Customizing	870
16.11.1 Umsetzung	870
16.11.2 Management-State	870
16.12 MachineConfigs, Machines, MachineSets und Scaling	871
16.12.1 MachineConfigs	872
16.12.2 MachineConfig-Operator	873
16.12.3 Komponenten des MCO	873
16.12.4 MachineConfigPool	874
16.12.5 Machines, MachineSets: manuelle Skalierung	875
16.12.6 MachineConfigs nach dem Deployment anpassen	877
16.12.7 Defekte MachineConfigs entfernen und debuggen	879
16.13 Cluster-Autoscaler und Machine-Autoscaler	879
16.13.1 High-level Betrachtung	880
16.13.2 Machine-Autoscaler	880
16.13.3 Cluster-Autoscaler	881
16.13.4 Thresholds	882
16.13.5 Zu beachtende Punkte	883
16.14 Customisierte MachineSets für spezielle Instanztypen – (z. B. GPU- oder Storage-Nodes) erzeugen	884
16.14.1 MachineConfigPool für GPU-Nodes erstellen	885
16.14.2 Erzeugung eines GPU-MachineSets	886
16.14.3 GPU-MachineSets unter vSphere mit angepasstem RHCOS-VM-Template	887
16.14.4 Angepasste AWS/GCP-MachineSets mit multiplen Disks (z. B. für Storage-Cluster mit Rook-Ceph)	889
16.15 Infrastructure-Nodes in OpenShift	890
16.15.1 Die wichtige Rolle der Infra-Nodes	890
16.15.2 Umsetzung	891
16.15.3 Einen Custom-Pool für die Infra-Node-Templates erzeugen	891
16.15.4 Umzug	893
16.16 HA für das OpenShift-Controlplane mit ControlPlaneMachineSets	894
16.16.1 Neue Foundation und alte Mankos	894
16.16.2 CPMS Hands-On	896
16.16.3 Fazit aus der Praxis	896

16.17 OpenShift-Upgrades: Foundations	897
16.17.1 Channels und Update-Kanäle	897
16.17.2 Koordinierte Cluster-Updates über OpenShift-Cluster-Management und RHACM	898
16.17.3 Grafische Auflösung von möglichen Update-Pfaden	899
16.18 OpenShift-Upgrades: EUS Upgrades	899
16.18.1 OpenShift 4.10 mit Kubernetes 1.23 auf OpenShift 4.12 mit Kubernetes 1.25	900
16.18.2 EUS-Channel	900
16.18.3 Preflights und Durchführung des Upgrades	900
16.19 Interaktive OpenShift-Workshops	903

TEIL VI Day 3 Operations: Cluster-Federation, Security, CI/CD-GitOps-Systeme, SDS und mehr

17 Day 3 Operations: Multi-Cluster-Management und Federated Cluster 907

17.1 Historisches	907
17.1.1 Setup und andere Kopfschmerzen, Friedhofsglocken und Fazit	907
17.1.2 Multi-Cluster = Cluster Federation?	908
17.2 Multi-Cluster-Management mit Red Hat Advanced Cluster Management	909
17.2.1 Hub Cluster	910
17.2.2 Managed Cluster	910
17.2.3 Cluster-Lifecycle	910
17.2.4 MultiCluster-Networking mit Submariner	912
17.3 Setup und grundlegende Cluster-Verwaltung per RHACM	914
17.3.1 Preflights	914
17.3.2 Den Hub Cluster (die Management-Instanz) per RHACM-Operator erzeugen	914
17.3.3 Setup des MultiClusterHubs (MCH)	916
17.3.4 Grafische Oberfläche für das MultiClusterHub-Management	918
17.3.5 ClusterSets, ClusterPools und -Claims	918
17.3.6 Cluster erzeugen	919
17.3.7 Import bestehender Cluster	920
17.3.8 Multi-Cluster-Networking aktivieren, ApplicationSets	921
17.3.9 MultiClusterHub-HA, Backup und Disaster Recovery	924

17.3.10	Konzeptionelle Arbeitsschritte zur Einrichtung eines Backups (nicht Hub-spezifisch)	925
17.4	Services, Ingress und Gateways in Multi-Cluster-Umgebungen	927
17.4.1	GKE	928
17.4.2	AKS und EKS	930
17.4.3	3rd-Party Multi-Cluster-Ingress: Netscaler, F5	930
17.4.4	Red Hat Service Interconnect	930

TEIL VII Virtualisierung, Security und GitOps

18 Day 3 Operations: VMs in Kubernetes/ OpenShift-Cluster einbinden 935

18.1	KubeVirt – VMs als Container	936
18.1.1	Funktionsweise	936
18.1.2	Nested oder nicht?	937
18.1.3	Stateful oder stateless?	938
18.1.4	Netzwerk	938
18.1.5	Preflights für das Setup	939
18.1.6	HyperConverged-CR ausrollen	940
18.1.7	OpenShift-Virtualization-Management	942
18.1.8	Connect mit externen Netzen	945
18.1.9	Import bestehender VMs	946

19 Day 3 Operations: Container-Security – Full-Featured Security-Stacks 947

19.1	Vorbetrachtungen zu Security-Lösungen	948
19.1.1	Grundsätzliche Aspekte	948
19.1.2	NeuVector und StackRox	949
19.2	NeuVector	950
19.2.1	Überblick über die wichtigsten Features	950
19.2.2	Architektur	951
19.2.3	CVE-Sources und CVE-Whitelisting	951
19.2.4	Setup	952

19.2.5	Login und Verwaltung	952
19.2.6	Federation-Management-/Multi-Cluster	955
19.3	RHACS – Red Hat Advanced Cluster Security für OpenShift	956
19.3.1	Setup	956
19.3.2	Cluster hinzufügen	957
19.3.3	Management	958
20	Day 3 Operations: Container-Security – Advanced Secret Management	961
20.1	EncryptionConfiguration für Secrets und andere Objekte	962
20.2	Secret Encryption unter GKE und EKS	963
20.3	HashiCorp Vault	964
20.3.1	Funktionsweise im Überblick	964
20.3.2	Alles durchgängig schön? Schön wär's.	965
20.3.3	Authentifizierungs- und Autorisierungs-Workflow	966
20.3.4	Kernkonzepte und Komponenten (Auszüge)	967
20.3.5	Setup-Fragen: Vault extern als VM oder (Kubernetes-)Cluster-intern?	971
20.3.6	Vault-Referenzarchitektur, HA-Aspekte und Best Practices	971
20.4	Setup des Vault Clusters	973
20.4.1	Preflights	973
20.4.2	Setup	974
20.4.3	Manueller Vault-Init und Unseal	976
20.4.4	Die Vault-UI	978
20.4.5	Die Ansätze	979
20.4.6	Ein simples App-Setup	980
20.4.7	Vault Agent Sidecar Injector	982
20.4.8	Vault Secrets Operator	986
20.4.9	External Secrets Operator	994
20.4.10	Argo CD Vault Plugin	994
20.5	Vault PKI Secrets Engine	995
20.5.1	Hands-On	995
20.6	Sealed Secrets (Bitnami)	998
20.6.1	Setup	999
20.6.2	Die Downsides	1002

TEIL VIII Vollautomatisierte CI/CD-GitOps-Pipelines

21 Day 3 Operations: CI/CD-Pipelines und GitOps	1005
21.1 GitOps	1005
21.1.1 Modelle für GitOps-Pipelines	1006
21.1.2 Multiple Stages und Applikationen	1008
21.1.3 Weitere Tools, um GitOps zu implementieren	1009
21.2 GitOps mit Tekton (CI-Fokus)	1009
21.2.1 Tekton	1009
21.2.2 Tekton und GitLab	1010
21.2.3 Tekton: Aufbau und Funktion	1011
21.2.4 Ablösung von ClusterTasks durch Resolver	1013
21.2.5 Tekton Hub und Tekton Catalog Repository	1013
21.2.6 Webhooks	1014
21.2.7 CI-Pipeline-Trigger mit Tekton	1014
21.2.8 Trigger: Die CRD-Komponenten	1015
21.2.9 CD-Automation	1017
21.3 Tekton-Setup	1017
21.3.1 Preflights	1017
21.3.2 Aufbau und Struktur der folgenden Beispiele	1018
21.3.3 Setup per Tekton-Operator (GKE)	1018
21.3.4 Troubleshooting bei Upgrades	1021
21.4 Beispiele für Tekton Pipeline (Pi-Calculator, Build, Push & Deploy)	1022
21.4.1 Vorbetrachtungen	1022
21.4.2 Achtung: Verwendete Registry und davon abhängige Secret-Konfiguration	1024
21.4.3 Start des PipelineRuns	1025
21.5 Tekton Pipelines unter OpenShift (OpenShift Pipelines)	1026
21.5.1 Die Tekton-CLI	1026
21.5.2 Installation der Tekton-Pipeline via Operator	1027
21.5.3 Beispiel für das Pipeline-Setup: Vote-API/-UI	1027
21.5.4 Start der PipelineRuns	1029
21.5.5 Tekton-Trigger und EventListener	1031
21.6 GitOps mit Argo CD (CD-Fokus)	1033
21.6.1 Vorbetrachtungen	1033
21.6.2 Setup unter Vanilla Kubernetes und GKE	1034
21.6.3 GitOps/Argo CD unter OpenShift	1035
21.6.4 Einzelschritte des Setups (Operator-basiert)	1035

21.6.5	CRDs	1036
21.6.6	GUI-Login (nur Vanilla Kubernetes)	1037
21.6.7	Login auf dem Argo-CD-Server	1038
21.6.8	Repos und Apps hinzufügen	1038
21.6.9	Akuity	1041
21.7	Argo Rollouts	1042
21.7.1	Motivation	1043
21.7.2	Funktionsweise	1044
21.7.3	Integration von Rollouts in Argo CD und das Argo-Rollouts-Dashboard	1045
21.7.4	Das technische Konzept	1045
21.7.5	Setup	1046
21.7.6	Aufbau einer Rollout-CR (Canary-Upgrades)	1048
21.7.7	Setup und Management eines Canary-Rollouts	1052
21.7.8	Analysis	1055
21.7.9	Metrik-Erfassung des Argo-Rollout-Controllers via Prometheus und ServiceMonitor	1057

TEIL IX Software-Defined Storage für verteilte Container-Infrastrukturen

22	Day 3 Operations: Software-Defined Storage für Container-Cluster	1061
22.1	SDS-Funktionsprinzipien	1061
22.1.1	Multi-Purpose-SDS für jeden Anwendungsfall: Block, File, Object	1062
22.1.2	ObjectStore für Container im Überblick	1063
22.1.3	ObjectStorage und Anwendungsfälle	1063
22.2	Ceph	1064
22.2.1	Ceph und RADOS	1065
22.2.2	Librados und Crushmaps	1066
22.2.3	Die Ceph-Daemons im Kurzüberblick: MON, OSD, MDS, MGR	1066
22.3	Ceph: Storage-Bereitstellungsverfahren für Container-Cluster	1067
22.3.1	RBD	1067
22.3.2	CephFS	1067
22.3.3	ObjectStore (RGW)	1068
22.4	Containerized SDS – Ceph per Rook	1068
22.4.1	Arbeitsweise und Konfiguration	1069

22.4.2	Beispielhafte Abläufe für die Bereitstellung der verschiedenen Storage-Varianten	1071
22.4.3	Rook, Ceph, NooBaa – und OpenShift?	1071
22.5	Setup von Rook	1072
22.5.1	Setup-Preflights: Admission-Controller, Raw-Disks vs. PVC	1073
22.5.2	Setup-Preflights: Common Ressourcen (CRDs, RBAC etc.)	1075
22.5.3	Rook-Operator: Konfiguration	1076
22.5.4	Setup-Preflights: Node-Label und Discovery	1077
22.5.5	Rollout des Operators	1079
22.5.6	Achtung – Preflights GKE: Erhöhung der ResourceQuota für Pods der PriorityClass system-*-critical	1080
22.5.7	Rollout des Ceph-Clusters	1080
22.6	Rook-Administration	1084
22.6.1	Dashboard	1084
22.6.2	Rook-Toolbox für Ceph-Cluster	1086
22.6.3	Dashboard-Settings und Orchestrator	1087
22.6.4	Neue Pools anlegen	1089
22.6.5	CephFS (MDS) einrichten	1089
22.6.6	StorageClasses für RBD und CephFS einrichten	1092
22.6.7	Ceph-ObjectStore als Storage-Backend	1093
22.6.8	Ceph-ObjectStore: Setup und Betrieb	1095
22.6.9	ObjectStore-Consumer erzeugen	1098
22.6.10	User für den direkten Zugriff auf Buckets anlegen	1100
22.6.11	Prometheus-Monitoring für Rook	1102
22.6.12	Zugriff von innen und außen	1103
22.6.13	Nachträgliche Erweiterung, Anpassungen und Upgrades	1103
22.6.14	Crushmaps über Node-Label konfigurieren	1103
22.6.15	Rook-Ceph-Cluster deinstallieren	1103

23 Day 3 Operations: Kostenkontrolle in Kubernetes/ OpenShift-Clustern (FinOps) 1105

23.1	FinOps	1106
23.1.1	Cost Management	1106
23.1.2	Kostenmanagement in OpenShift (Built-in)	1107
23.1.3	Kubecost	1108

24 Day 3 Operations: GPU-beschleunigte KI/ML-Container-Infrastrukturen	1113
<hr/>	
24.1 GPUs und autoskalierbare KI/ML-Stacks	1113
24.1.1 Das große Ganze	1113
24.1.2 Die unerschöpflichen GPU-Ressourcen in der Cloud – oder eher nicht?	1114
24.2 Konkrete Einsatzszenarien und Kosten	1115
24.2.1 vGPU	1116
24.2.2 MIG – Multi Instance GPU	1117
24.2.3 GPU-Sharing	1118
24.3 NVIDIAs GPU-Operator	1118
24.3.1 Die Einzelkomponenten des GPU-Operators im Überblick	1118
24.3.2 NFD-Operator	1120
24.4 GKE-Cluster mit NVIDIA-A100-Instanzen und MIG-Partitionierung	1121
24.4.1 GCP-Preflights: Passende Instanzen/Machine-Types für GPUs, Quotas	1121
24.4.2 Setup des GPU-Node-Pools	1122
24.4.3 Rollout des GPU-Operators	1123
24.4.4 Management der A100-GPU und Anwendung der MIG-Schemata	1125
24.5 OpenShift-Cluster mit NVIDIA-A100-GPUs in der GCP	1128
24.5.1 Preflights für GCP und OpenShift	1128
24.5.2 GPU-Node skalieren und partitionieren	1129
24.6 AKS- und EKS-Cluster mit NVIDIA-GPUs	1131
25 The Road ahead	1133
<hr/>	
Index	1135