

# GitOps

Das Praxisbuch für DevOps-Teams und  
Systemarchitekten

# DAS INHALTS- VERZEICHNIS

» Hier geht's  
direkt  
zum Buch

---

# Auf einen Blick

1	CI/CD und GitOps unter Kubernetes/OpenShift in Enterprise-Umgebungen .....	23
2	Grundsätzliche strategische Überlegungen .....	31
3	Preflights: Operatoren .....	39
4	CI/CD-Pipelines und GitOps – Überblick .....	61
5	GitOps mit Tekton/OpenShift Pipelines (CI-Fokus) .....	89
6	GitOps mit Argo CD/OpenShift GitOps (CD-Fokus) .....	343
7	Security für CI/CD- und GitOps-Systeme mit Policies as Code .....	451
8	Secret-Management für CI/CD und GitOps – Überblick .....	473
9	Secret-Management für GitOps-Szenarien – Hands-on .....	481
10	Argo Rollouts .....	547
11	GitOps – IaC: Kubernetes-Workload-Cluster mit der Cluster API ausrollen .....	595
12	GitOps – IaC: Externe Infrastrukturen über Kubernetes mit Crossplane managen .....	637

# Inhalt

Vorwort .....	21
<b>1 CI/CD und GitOps unter Kubernetes/ OpenShift in Enterprise-Umgebungen</b> .....	<b>23</b>
<b>1.1 Vorbemerkungen</b> .....	<b>27</b>
1.1.1 Verwendete Formatierungen .....	28
1.1.2 Weiterführende Hinweise .....	28
<b>1.2 Kernziele und die rote Fäden</b> .....	<b>28</b>
<b>1.3 Zielgruppen und verwendete Systeme</b> .....	<b>29</b>
1.3.1 Verwendete Systeme .....	29
1.3.2 Erforderliche Vorkenntnisstände .....	30
1.3.3 Gescrriptete Setups .....	30
1.3.4 Betrachtete Stände .....	30
<b>2 Grundsätzliche strategische Überlegungen</b> .....	<b>31</b>
<b>2.1 Entscheidungsfindung</b> .....	<b>31</b>
<b>2.2 Changes</b> .....	<b>32</b>
<b>2.3 CI/CD und GitOps</b> .....	<b>32</b>
2.3.1 Die Kernprinzipien von GitOps .....	33
<b>2.4 Plattformgebundene vs. generische GitOps-Tools</b> .....	<b>34</b>
<b>2.5 Ein Überblick über die Tools</b> .....	<b>35</b>
2.5.1 Tekton (OpenShift Pipelines) .....	35
2.5.2 Argo CD (OpenShift GitOps) .....	36
2.5.3 Argo Rollouts .....	36
2.5.4 Cluster API .....	36
2.5.5 Crossplane .....	37
2.5.6 Security und Compliance .....	37
2.5.7 Das strategische Gesamtbild .....	37
2.5.8 Fazit .....	38

### 3 Preflights: Operatoren 39

- 3.1 Operator Foundations** ..... 40
  - 3.1.1 CI/CD und GitOps – und Operatoren ..... 41
  - 3.1.2 Operator-Typen und Entwicklungsansätze ..... 42
  - 3.1.3 Operator: Technische Betrachtung ..... 44
  - 3.1.4 Controller-Loops ..... 46
  - 3.1.5 Integrierte oder externe Controller ..... 47
  - 3.1.6 Operatorhub.io und OpenShift-Operatoren ..... 47
- 3.2 Operator-Typen und Maturitätslevel** ..... 48
  - 3.2.1 Helm vs. GitOps und Operatoren ..... 50
- 3.3 Operator-Typen im funktionalen Vergleich: Ansible vs. Go** ..... 51
  - 3.3.1 Unterschiede zwischen Go- und Ansible-basierten Operatoren ..... 51
- 3.4 OLM – wer überwacht die Wächter?** ..... 53
  - 3.4.1 CRDs ..... 54
  - 3.4.2 Operatoren und Registry ..... 55
  - 3.4.3 Installation von OLM (Operator Lifecycle Manager) –  
nur Vanilla Kubernetes ..... 56
- 3.5 Operator-Management** ..... 58
  - 3.5.1 Operator-Management per CLI ..... 58
  - 3.5.2 OLM-Uninstall (nur Vanilla Kubernetes) ..... 59

### 4 CI/CD-Pipelines und GitOps – Überblick 61

- 4.1 GitOps** ..... 62
  - 4.1.1 Modelle für CI/CD- und GitOps-Pipelines ..... 62
  - 4.1.2 Multiple Stages und Applikationen ..... 64
- 4.2 Git-Systeme und CI/CD- und GitOps-Tools im Vergleich** ..... 64
  - 4.2.1 Git-Anbieter und Tools, um CI/CD- und GitOps-Systeme zu  
implementieren (Auszüge) ..... 65
  - 4.2.2 GitHub ..... 66
  - 4.2.3 GitLab ..... 67
  - 4.2.4 Bitbucket ..... 69
  - 4.2.5 Gitea ..... 70
  - 4.2.6 AWS CodeCommit ..... 71
  - 4.2.7 Azure DevOps ..... 72
  - 4.2.8 Kubernetes-native? ..... 73

4.2.9	Tabellarischer Überblick .....	74
4.2.10	Fazit .....	76
4.2.11	Strategische Vorbetrachtungen (CI-Fokus): Standard-Tasks von CI-Pipelines in Enterprise-Umgebungen .....	77
4.2.12	Grundlagen zur Struktur von (Git-)Repos .....	79
<b>4.3</b>	<b>Git-Foundations</b> .....	<b>82</b>
4.3.1	Was ist Git? .....	82
4.3.2	Begriffe und Kommandos .....	83
4.3.3	Zustände .....	85
4.3.4	Git-Operationen als Auslöser für CI-Pipelines .....	85
4.3.5	GitHub: CI-relevante Webhook-Events im Überblick (Auszüge) .....	86
4.3.6	GitLab: CI-relevante Webhook-Events im Überblick (Auszüge) .....	88

## 5 GitOps mit Tekton/OpenShift Pipelines (CI-Fokus) 89

<b>5.1</b>	<b>Tekton</b> .....	<b>89</b>
<b>5.2</b>	<b>Tekton: Aufbau, Funktion, Konzepte und Custom Resources</b> .....	<b>91</b>
5.2.1	High-Level-Konzepte und Komponenten sowie CRs .....	91
<b>5.3</b>	<b>PipelineRun, Pipeline, Tasks und mehr</b> .....	<b>92</b>
5.3.1	Pipeline .....	92
5.3.2	PipelineRun .....	94
5.3.3	Task .....	94
5.3.4	TaskRun .....	95
5.3.5	Step .....	95
5.3.6	Parameter-Übergabe zwischen PipelineRun, Pipeline und Tasks .....	96
5.3.7	Results .....	96
5.3.8	ClusterTasks .....	97
5.3.9	(Remote-)Resolver .....	97
5.3.10	PipelineRuns ohne Pipeline (Embedded Pipelines) .....	101
5.3.11	StepActions .....	102
5.3.12	Tekton Results .....	104
5.3.13	Die Custom Resource Tekton Results .....	109
5.3.14	Pruner .....	113
5.3.15	Event-basierter Tekton-Pruner (Tech-Preview) .....	115
<b>5.4</b>	<b>Webhooks, Trigger*, EventListener und Interceptor unter Tekton</b> .....	<b>118</b>
5.4.1	Webhooks .....	118
5.4.2	CI-Pipeline-Trigger mit Tekton .....	120
5.4.3	Trigger: Die CRD-Komponenten .....	120

5.4.4	Vereinfachtes Zusammenspiel der Trigger-Komponenten .....	123
5.4.5	Tekton-Interceptors .....	124
5.4.6	Mehrere Interceptors zur Vermeidung des Ping-Pong-Problems .....	128
5.4.7	Trigger-Modifikationen mit CEL-Interceptor und Overlays .....	130
5.4.8	TriggerGroups (Nested Triggers) .....	130
<b>5.5</b>	<b>Tekton-Setup .....</b>	<b>131</b>
5.5.1	Preflights .....	131
5.5.2	Migration von gcr.io auf ghcr.io .....	132
5.5.3	Tekton Operator (Vanilla Kubernetes) .....	132
5.5.4	Funktionsweise des Tekton Operators .....	132
5.5.5	Sub-CRDs: TektonPipeline, TektonDashboard und mehr .....	135
5.5.6	Setup per Tekton Operator, Setup der CRDs und des Pipeline-Stacks .....	137
5.5.7	Tekton CLIs (TKN/OPC) für Vanilla Kubernetes und OpenShift .....	138
5.5.8	Post-Rollout-Details (Vanilla Kubernetes) .....	138
5.5.9	Tekton-Dashboard (Vanilla Kubernetes) – Auth per GitHub .....	145
5.5.10	Troubleshooting bei Tekton-Upgrades .....	152
5.5.11	Setup-Details (OpenShift) .....	153
<b>5.6</b>	<b>Privater Tekton/Artifact Hub .....</b>	<b>156</b>
5.6.1	Privater Tekton Hub: Funktionsweise und Komponenten .....	157
5.6.2	Anpassungsmöglichkeiten: Categories .....	158
5.6.3	Anpassungsmöglichkeiten: Custom Catalogs mit eigenen Tasks/Pipelines .....	159
5.6.4	Privater Artifact Hub .....	161
<b>5.7</b>	<b>Hands-on: Simple Pipeline (Pi-Calculator) – Build, Push &amp; Deploy .....</b>	<b>163</b>
5.7.1	Vorbetrachtungen / Pipeline-Details .....	163
5.7.2	Pipeline-Security, ServiceAccounts, SCCs, RBAC .....	165
5.7.3	Abarbeitung der Tasks, Task-Serialisierung und Parallelisierung .....	168
5.7.4	Verwendete Registry und davon abhängige Secret-Konfiguration .....	169
5.7.5	Installation der Pipeline und Tasks .....	170
5.7.6	PipelineRun für die Picalc-Pipeline .....	171
5.7.7	Start des PipelineRuns .....	173
5.7.8	PipelineRun-Status .....	174
5.7.9	Add-On: Dev/Committer über PipelineRun benachrichtigen .....	175
5.7.10	Unterstützte Variablen-Substitution in Tasks und Pipelines .....	181
5.7.11	Add-on: Git-Push-YAML Manifest mit Image-Digest .....	183
5.7.12	Pod Affinity Assistant .....	183
5.7.13	Trigger und EventListener .....	185

---

<b>5.8</b>	<b>Pipeline mit Checks – Vorbetrachtungen</b> .....	185
5.8.1	Exkurs: Trügerische Sicherheit? Scanner-Rauschen, CVEs, SBOM, VEX, DAST, SCA und mehr .....	185
5.8.2	Check-Tasks: Trivy (Überblick) .....	189
5.8.3	Check-Tasks: Code-Quality-Checks mit SonarQube .....	191
5.8.4	Staging-Task: Skopeo .....	193
5.8.5	Policy Engines .....	193
<b>5.9</b>	<b>Pipelines mit Checks – Überblick</b> .....	194
5.9.1	Vote-App Pipeline .....	194
5.9.2	Pipeline mit Checks und Staging Task .....	194
5.9.3	Pipeline mit Checks (SonarQube) .....	194
<b>5.10</b>	<b>Die Pipeline »Vote-App« mit Checks und interner Registry</b> .....	194
5.10.1	Beispiel-Pipeline: Vote-API/-UI .....	195
5.10.2	Der Task golang-test .....	198
5.10.3	Konditionale Ausführung des Tasks golang-test .....	199
5.10.4	Der doppelte Trivy-Scanner-Task .....	201
5.10.5	Start der beiden PipelineRuns .....	203
5.10.6	CVE-Scan-Tasks (wie Trivy) zur Anzeige der CVEs in der OpenShift-Pipelines-UI erweitern .....	205
5.10.7	SBOM-Task .....	208
5.10.8	Manual Approval Tasks (TP) .....	216
5.10.9	Tekton Trigger und EventListener für die Vote App Pipeline .....	219
<b>5.11</b>	<b>Pipeline mit Checks (GitLeaks, Trivy, Golang) und Staging-Task (Skopeo)</b> .....	221
5.11.1	Details und zu beachtende Punkte .....	223
5.11.2	ServiceAccount-Fragen .....	223
5.11.3	Gitleaks-Task: Überblick und technische Implementierung .....	224
5.11.4	Skopeo als Staging-Task: Technische Implementierung .....	226
<b>5.12</b>	<b>Pipeline mit Checks (SonarQube)</b> .....	232
5.12.1	Beispiel-Pipeline: pipeline-golang-sonarqube-trivy .....	232
5.12.2	Pipeline-Tasks und Parameter .....	233
5.12.3	Das SonarQube-Setup .....	234
5.12.4	SonarQube-Management über die UI .....	235
5.12.5	Result-Übergabe zwischen SonarQube- und Quality-Gate-Check-Task .....	237
5.12.6	Parameter der SonarQube-Tasks .....	240
5.12.7	Setup der Pipeline und Tasks .....	241
5.12.8	Details und zu beachtende Punkte .....	241

<b>5.13 Überblick: Supply Chain Security (CI)</b> .....	241
5.13.1 SLSA .....	242
5.13.2 SLSA-Level und Implementierungen .....	242
5.13.3 Detaillierte Beschreibungen der SLSA-Level .....	243
5.13.4 Einschränkungen .....	244
<b>5.14 Tekton Chains (SLSA 2)</b> .....	246
5.14.1 SLSA unter Tekton .....	246
5.14.2 Tekton Chains .....	246
5.14.3 Signature Server und Sigstore .....	248
5.14.4 Überblick zu Sigstore .....	249
5.14.5 Architektur von Sigstore .....	250
5.14.6 Sigstore Workflow .....	252
5.14.7 Sicherheit und Integrität .....	254
5.14.8 Funktionsweise von Sigstore .....	254
5.14.9 Integration mit Tekton Chains .....	255
5.14.10 Technischer Ablauf (Rekor im Zusammenspiel mit Tekton Chains) ...	256
5.14.11 The Big Picture: Tekton-Chains-Signing-Workflow mit Cosign .....	259
5.14.12 Vault als Chains Backend .....	260
<b>5.15 Tekton Chains (SLSA 2) unter Vanilla Kubernetes</b> .....	261
5.15.1 Setup .....	261
5.15.2 PipelineRun .....	263
5.15.3 Image-Signaturen verifizieren .....	265
5.15.4 Auto-Verify mit Kyverno (unsigned Images dürfen nicht ausgerollt werden) .....	266
5.15.5 Sigstore-Bundles .....	268
<b>5.16 Tekton Chains unter OpenShift</b> .....	268
5.16.1 Einrichtung .....	268
5.16.2 Signing-Secret erzeugen .....	269
5.16.3 Transparency aktivieren .....	270
5.16.4 Signed Run und Verify .....	271
<b>5.17 Tekton-Bundles</b> .....	275
5.17.1 Überblick .....	275
5.17.2 Verwendung und Nutzen von Tekton-Bundles .....	275
5.17.3 Aufbau und Erstellung eines Tekton-Bundles .....	276
5.17.4 Nutzung von Tekton-Bundles in der Pipeline .....	277
5.17.5 Bundle mit Pipelines und Tasks .....	278
<b>5.18 Tekton Pipeline für einen Operator-Build</b> .....	280
5.18.1 Schematischer Ablauf eines manuellen Operator-Build-Prozesses ...	281

5.18.2	Funktionsbeschreibung, erforderliche Tasks und Anpassungen gegenüber dem manuellen Modell .....	282
5.18.3	Operator-Build-Pipeline – Hands-on .....	284
5.18.4	Ausbau des Modells .....	285
<b>5.19</b>	<b>Debugging von Tekton-Pipelines .....</b>	<b>286</b>
5.19.1	Workspaces und Parameter .....	286
5.19.2	Results .....	287
5.19.3	Umgebungsvariablen und Secrets .....	288
5.19.4	Einsatz von Breakpoints für ein Echtzeit-Debugging .....	289
5.19.5	Implementierung von Monitoring und Alerting .....	289
<b>5.20</b>	<b>Pipelines as Code .....</b>	<b>290</b>
5.20.1	Features .....	290
5.20.2	Argo CD vs. Pipelines as Code .....	292
5.20.3	Der vereinfachte PaC-Workflow .....	293
5.20.4	Pipelines as Code – mögliche Objektreferenzen .....	294
5.20.5	Pipelines für verschiedene Branches .....	297
5.20.6	Workflow für ein PaC-Setup .....	297
5.20.7	PaC-Setup .....	298
5.20.8	Die PaC-CRs .....	300
5.20.9	PAC-Zugriff: GitHub-App vs. Personal Access Token (PAT) .....	300
5.20.10	Setup einer GitHub-App für PaC .....	301
5.20.11	Repository, Secret und Webhook per <code>opc/tnk pac create repo</code> erzeugen (PAT) .....	305
5.20.12	PaC und die Public Tekton Hub Deprecation .....	311
5.20.13	Triggern des PipelineRuns .....	311
5.20.14	Mehrere PaC-Controller für mehrere GitHub-Apps .....	316
<b>5.21</b>	<b>Automatische Erzeugung von KubeVirt/OpenShift-Virtualization-VM-Images mit OpenShift Pipelines .....</b>	<b>317</b>
5.21.1	Automatisierte Erzeugung von Bootable Volumes für VMs über Pipelines .....	318
5.21.2	Storage-Begriffserklärung im Kontext von OpenShift Virtualization/KubeVirt .....	320
5.21.3	High-Level-Arbeitsweise der Pipeline und Tasks .....	321
5.21.4	Die Tasks im Detail .....	323
5.21.5	PipelineRun .....	326
5.21.6	Erzeugte Objekte .....	328
5.21.7	Argo CD/OpenShift GitOps im Kontext von KubeVirt/OpenShift Virtualization .....	329

<b>5.22</b>	<b>Argo Workflows (CI)</b> .....	330
5.22.1	Argo Workflows vs. Tekton .....	330
5.22.2	Vergleich der Ansätze .....	331
5.22.3	Was fehlt ... ..	332
5.22.4	Argo Workflows – Hands-on .....	332
5.22.5	Argo-Workflow-CRDs .....	332
5.22.6	UI-Login .....	334
5.22.7	Beispiele .....	335
5.22.8	Argo Workflows im OpenShift-AI-Stack .....	339

## **6 GitOps mit Argo CD/OpenShift GitOps (CD-Fokus)** 343

---

<b>6.1</b>	<b>Vorbetrachtungen</b> .....	343
6.1.1	Argo CD Enterprise: Akuity vs. Open Source .....	344
6.1.2	Deployment-Optionen (Überblick) .....	345
6.1.3	Versionen .....	346
6.1.4	Breaking Changes zwischen Version 2.x und 3.x .....	347
<b>6.2</b>	<b>Argo-Setup unter Vanilla Kubernetes/GKE</b> .....	348
6.2.1	Setup ohne Operator .....	348
6.2.2	Setup unter Vanilla Kubernetes/GKE per Argo-CD-Operator .....	349
6.2.3	Die Argo-CD-CLI unter Vanilla Kubernetes .....	349
6.2.4	Login im Argo-Dashboard (nur Vanilla Kubernetes) .....	350
6.2.5	Login auf dem Argo-CD-Server (CLI) .....	350
<b>6.3</b>	<b>OpenShift GitOps-/Argo-CD-Setup unter OpenShift</b> .....	351
6.3.1	Installation des Operators .....	351
6.3.2	Einzelschritte des Setups (Operator-basiert unter OpenShift) .....	351
<b>6.4</b>	<b>Argo-Konfiguration und -Management</b> .....	353
6.4.1	Die Argo-CRDs .....	353
6.4.2	Namespaced- vs. Cluster-Scope, Label, SourceNamespaces und mehr .....	358
6.4.3	Webhooks für sofortigen Sync .....	361
6.4.4	Argo und HPA .....	362
6.4.5	Argo und ConfigMap-Sync .....	363
6.4.6	Argo-CD-Monitoring/Alerting per Prometheus .....	365
6.4.7	Config Management Plugins .....	368
6.4.8	Export und Disaster Recovery (Import) .....	369
6.4.9	Der Argo-CD-Source-Hydrator .....	370

---

<b>6.5</b>	<b>Argo CD: Applications</b> .....	371
6.5.1	Repos und Apps hinzufügen, Argo-CD-Namespace-Label .....	371
6.5.2	Argo-Applications: Sync-Optionen konfigurieren .....	373
6.5.3	(Globale) SyncOptions .....	374
6.5.4	Diff-Strategien verstehen .....	376
6.5.5	Tool-Detection und Directory-Settings .....	379
6.5.6	Bestimmte Ressourcen/Attribute vom Sync ausschließen/ skippen .....	380
6.5.7	Hands-on: Vote-API und die Vote-UI-App unter OpenShift .....	381
6.5.8	Hands-on: App mit Ingress-/Routen-Kustomizations für multiple Cluster/FQHNs .....	385
6.5.9	Argo-Sync-Waves und -Phasen verstehen und einsetzen .....	391
6.5.10	Zugriff auf private Repos .....	400
6.5.11	Ressourcen automatisch löschen, wenn die Application gelöscht wird .....	401
6.5.12	Multi-Source-Apps .....	402
6.5.13	Das Pattern »App-of-Apps« .....	403
<b>6.6</b>	<b>App-Projects</b> .....	405
6.6.1	Custom-App-Projects definieren und nutzen .....	405
6.6.2	Argo-Rollen verstehen und einsetzen (Scope: AppProjects) .....	407
<b>6.7</b>	<b>Argo Notifications</b> .....	412
6.7.1	Notifications aktivieren .....	413
6.7.2	Notification-Trigger und die NotificationsConfiguration-CR .....	414
6.7.3	Konfiguration von NotificationsConfigurations .....	415
6.7.4	Eine einfache Notification per Mail .....	416
6.7.5	Notifications an Mattermost senden .....	420
<b>6.8</b>	<b>Argo ApplicationSets</b> .....	423
6.8.1	ApplicationSets .....	423
6.8.2	Generators .....	425
6.8.3	Pull- und Push-Modell .....	427
<b>6.9</b>	<b>Argo ApplicationSets (Argo-CD-nativ)</b> .....	429
6.9.1	Preflights .....	429
6.9.2	Hands-on .....	430
6.9.3	Die Argo CD-UI und die Abstinenz von ApplicationSets .....	432
<b>6.10</b>	<b>Argo-ApplicationSets unter RHACM</b> .....	434
6.10.1	Preflights .....	434
6.10.2	Hands-on .....	436
6.10.3	Placements anpassen .....	439

<b>6.11</b>	<b>Argo Resource Health</b> .....	442
6.11.1	Überblick .....	442
6.11.2	Bedeutung von Argo Resource Health für GitOps-Workflows und Automatisierung .....	443
6.11.3	Technische Funktionsweise mit Kubernetes-Ressourcen .....	444
6.11.4	Health Assessments, benutzerdefinierte Health Checks und das Statusmodell .....	445
6.11.5	Konzeptionelle Beispiele für Argo-Health-Checks .....	446
6.11.6	Erweiterungs- und Anpassungsmöglichkeiten / Custom-Health-Checks .....	446
6.11.7	Weitere Features und Methoden .....	447
6.11.8	Beispiel: Argo CD Resource Health für Secret Store CSI Driver mit Vault-Provider .....	448

## **7 Security für CI/CD- und GitOps-Systeme mit Policies as Code** 451

---

<b>7.1</b>	<b>Tool-Überblick: Kyverno, OPA Gatekeeper, Kubewarden</b> .....	452
<b>7.2</b>	<b>Kyverno-Policies</b> .....	453
7.2.1	Überblick .....	453
7.2.2	Kyverno-CRDs .....	454
7.2.3	Kyverno-Setup .....	456
7.2.4	Beispiel-Policies .....	458
7.2.5	Weitere Template-Policies sowie eigene Policies .....	459
7.2.6	Anwendungsbeispiele .....	460
7.2.7	Kyverno Playground .....	462
<b>7.3</b>	<b>OPA Gatekeeper</b> .....	463
7.3.1	OPA und Gatekeeper .....	463
7.3.2	Rego vs. YAML .....	464
7.3.3	Policy-Management mit Templates und Constraints .....	465
7.3.4	Setup und CRDs .....	465
7.3.5	Beispiele .....	467
<b>7.4</b>	<b>OPA Gatekeeper vs. Kyverno</b> .....	470

<b>8</b>	<b>Secret-Management für CI/CD und GitOps – Überblick</b>	473
<b>8.1</b>	<b>Die Problematik</b>	473
<b>8.2</b>	<b>Tools und Konzepte im Überblick</b>	474
8.2.1	EncryptionConfiguration nativ (etcd)	474
8.2.2	Secret-Encryption unter GKE und EKS	475
8.2.3	Vault	476
8.2.4	Sealed Secrets	477
8.2.5	(K)SOPS	477
8.2.6	External Secrets Operator	478
8.2.7	Argo CD Vault Plugin	479
8.2.8	Secret Store CSI Driver	479
<b>9</b>	<b>Secret-Management für GitOps-Szenarien – Hands-on</b>	481
<b>9.1</b>	<b>HashiCorp Vault – Überblick</b>	481
9.1.1	Funktionsweise im Überblick	481
9.1.2	Kernkonzepte und Komponenten	484
9.1.3	Setup-Fragen: Vault extern als VM oder (Kubernetes-)Cluster-intern?	487
9.1.4	Vault-Referenzarchitektur, HA-Aspekte und Best Practices	488
<b>9.2</b>	<b>Vault-Cluster – Setup</b>	490
9.2.1	Preflights	490
9.2.2	Setup	491
9.2.3	Manueller Vault-Init und Unseal	493
9.2.4	Die Vault-UI	496
9.2.5	Die Vault-spezifischen Secret-Ansätze für Kubernetes	496
9.2.6	Kern-Features der Ansätze	497
9.2.7	Vault-Config-Operator	499
<b>9.3</b>	<b>Vault Agent Sidecar Injector</b>	499
9.3.1	Secrets über Annotations	500
9.3.2	Automatische Secret-Updates und Secret-Sync	500
9.3.3	Sidecar Injector Workflow	501
9.3.4	Preflights für das folgende Setup	502
9.3.5	Hands-on	502

<b>9.4</b>	<b>Vault Secrets Operator</b> .....	506
9.4.1	Foundation .....	506
9.4.2	Hands-on .....	507
9.4.3	Achtung: Vorbereitung des Rollouts des Vault Secrets Operator .....	508
9.4.4	Test-Setup .....	509
9.4.5	Dynamic Secrets .....	512
9.4.6	Fazit .....	514
<b>9.5</b>	<b>Argo CD Vault Plugin</b> .....	514
9.5.1	Konfiguration und Setup .....	515
9.5.2	Credentials in Vault erzeugen, Consumer-Setup .....	519
9.5.3	Secret-Refresh .....	521
<b>9.6</b>	<b>External Secrets Operator (Google Secret Manager)</b> .....	521
9.6.1	Technische Funktionsweise .....	522
9.6.2	Hands-on: Setup (Google Secret Manager) .....	522
9.6.3	Preflights .....	523
9.6.4	Setup von SecretStore und ExternalSecret .....	525
9.6.5	Versionsverwaltung, multiple Versionen und Auto-Update .....	527
9.6.6	Tenancy .....	528
<b>9.7</b>	<b>External Secrets Operator unter OpenShift mit Vault-Backend</b> .....	529
9.7.1	Setup .....	529
9.7.2	SecretStore und Secret erzeugen .....	530
<b>9.8</b>	<b>Vault CSI Provider / Secrets Store CSI Driver</b> .....	533
9.8.1	Vorbetrachtungen .....	533
9.8.2	CSI Inline Ephemeral Volumes .....	534
9.8.3	Hands-on .....	535
9.8.4	Secret Store / Vault CSI Driver und Secrets per ENV .....	539
<b>9.9</b>	<b>Sealed Secrets (Bitnami)</b> .....	539
9.9.1	Implementierung .....	539
9.9.2	Detaillierter Ablauf der Ver- und Entschlüsselung .....	540
9.9.3	Setup .....	541
9.9.4	Hands-on .....	542
<b>10</b>	<b>Argo Rollouts</b> .....	547
<b>10.1</b>	<b>Update-Strategien für Pods im Überblick</b> .....	547
10.1.1	Rolling Update .....	548
10.1.2	Recreate .....	549

---

10.1.3	Canary (Sukzessiver Schwenk/Traffic Weighting)	550
10.1.4	Blue/Green (»Ganz oder gar nicht«-Schwenk)	551
<b>10.2</b>	<b>Fortgeschrittene Update-Verfahren mit Argo Rollouts</b>	<b>552</b>
10.2.1	Funktionsweise	553
10.2.2	Das technische Konzept	554
10.2.3	Traffic-Steuerung (Canary)	555
10.2.4	Zusammenspiel: Traffic Weighting und Analysis	557
10.2.5	Integration von Rollouts in Argo CD und das Argo-Rollouts-Dashboard	558
<b>10.3</b>	<b>Die Rollout-Ressourcen im Detail: Canary und Blue/Green</b>	<b>560</b>
10.3.1	Aufbau einer Canary-Rollout-CR	560
10.3.2	Aufbau einer Blue/Green-Rollout-CR	564
<b>10.4</b>	<b>Argo Rollouts: Hands-on</b>	<b>567</b>
10.4.1	Setup von Argo Rollouts (Vanilla Kubernetes)	567
10.4.2	Setup und Management eines einfachen Canary-Rollouts	569
10.4.3	AnalysisTemplates	571
10.4.4	Argo Rollouts mit Prometheus-basiertem AnalysisTemplate und Traffic Weighting	573
10.4.5	Weighted Canary-Ingress mit Nginx	575
10.4.6	Rollout-Verwaltung	577
10.4.7	Metrik-Erfassung des Argo-Rollout-Controllers via Prometheus und ServiceMonitor	578
<b>10.5</b>	<b>Argo Rollouts unter OpenShift</b>	<b>579</b>
10.5.1	Setup	580
10.5.2	Canary-Rollout mit AnalysisTemplates und User-Workload-Monitoring	580
10.5.3	Token und Secret erzeugen	580
10.5.4	AnalysisTemplate	581
10.5.5	Kombinierte Queries	584
10.5.6	Die Sample-App	585
10.5.7	Sample-App aus dem Git und das Route-Sync-Problem umschiffen	586
10.5.8	Post Rollout, Metrics/Targets, App-Upgrade mit AnalysisRun starten	587
10.5.9	Blue-Green-Rollout mit Pre-/Post-PromotionAnalysisTemplates	590
<b>10.6</b>	<b>Argo-Experiments</b>	<b>593</b>
10.6.1	Anwendungsfall für Experiment	593

<b>11</b>	<b>GitOps – IaC: Kubernetes-Workload-Cluster mit der Cluster API ausrollen</b>	595
<b>11.1</b>	<b>Die Cluster API</b>	595
11.1.1	High-Level-Überblick: Kubernetes-Cluster provisionieren mit der Cluster API und GitOps (Argo CD)	595
11.1.2	Vereinheitlichung	596
11.1.3	Komponenten und Funktionsweise im Überblick	597
11.1.4	Provider-Landschaft für On-Prem und Self-Managed Clouds	598
11.1.5	CRDs/CRs	600
11.1.6	GitOps-Integration mit Argo CD	601
<b>11.2</b>	<b>CAPI und kubeadm, Alternativen</b>	602
11.2.1	Permanenter Vorbereitungsaufwand beim Kubeadm-Provider	603
11.2.2	Produktive Risiken und Kosten	603
11.2.3	MikroK8s oder Talos als Alternativen zu Kubeadm?	604
11.2.4	Fazit	605
11.2.5	Vanilla CAPI statt RHACM?	606
<b>11.3</b>	<b>Hands-on: Setup eines Workload-Clusters per CAPI unter GKE</b>	608
11.3.1	Benötigte Preflights	608
11.3.2	CAPI-Operator und Namespaces	608
11.3.3	Namespaces	610
11.3.4	Der Google-ServiceAccount	611
11.3.5	Erzeugen der Provider	613
11.3.6	Workload-Cluster – Aufbau	615
11.3.7	Installation des Workload-Clusters	620
11.3.8	Zugriff auf den neuen Workload-Cluster	621
11.3.9	Löschung der Workload-Cluster	622
11.3.10	Alternative Cluster-Rollouts mit ClusterClasses	622
<b>11.4</b>	<b>IaC mit CAPI und Argo</b>	623
11.4.1	Aufbau	623
11.4.2	IaC – Workload-Cluster-Rollout per Argo	624
11.4.3	Automatischer Application-Rollout auf selektive Workload-Cluster per ApplicationSets	625
11.4.4	ClusterResourceSets für fehlende Core-Services	628
11.4.5	Core-Service-Rollout mit App-of-Apps	631
11.4.6	Löschen der Ressourcen	635
<b>11.5</b>	<b>Zusammenfassung der wichtigsten Schritte für Vollautomation</b>	635

<b>12</b>	<b>GitOps – IaC: Externe Infrastrukturen über Kubernetes mit Crossplane managen</b>	637
<b>12.1</b>	<b>Von Cluster API zu Crossplane: Kubernetes als Universal-Controlplane ...</b>	638
12.1.1	Universal-Controlplane .....	638
12.1.2	Cluster API vs. Crossplane .....	639
12.1.3	Upbound und Crossplane .....	639
12.1.4	GitOps für Infrastruktur: Crossplane in Kombination mit Argo CD ....	640
12.1.5	Privilegien bzw. Credentials für die Provider .....	641
12.1.6	Eigene abstrahierte APIs mit Compositions und XRDs .....	641
12.1.7	Provider .....	644
12.1.8	Die Crossplane-UI .....	644
12.1.9	Herausforderungen beim Einsatz von Crossplane .....	645
<b>12.2</b>	<b>Crossplane einrichten</b> .....	647
12.2.1	Crossplane 2.0 .....	647
12.2.2	Installation des Crossplane-Stacks (Management-Cluster) .....	649
12.2.3	Erzeugte Crossplane/Upbound-CRDs .....	651
12.2.4	Erläuterung der wichtigsten CRDs .....	652
<b>12.3</b>	<b>Crossplane-Demo</b> .....	655
12.3.1	Preflights (AWS) .....	655
12.3.2	Provider installieren und managen .....	656
12.3.3	ProviderRevisions .....	657
12.3.4	Managed Resource Activation .....	659
12.3.5	MRAP-Settings .....	661
12.3.6	ProviderConfig .....	664
12.3.7	Einen AWS S3-Bucket erzeugen .....	664
12.3.8	Logs über nicht funktionierende Status-Updates in MRAP-Szenarien .....	665
12.3.9	Eine AWS-RDS-Instanz erzeugen .....	666
12.3.10	Integration in Argo .....	669
12.3.11	Resümee .....	669
	Schlusswort .....	671
	Index .....	673