

Kubernetes Best Practices

Praktische Anleitungen und Vorlagen zu
Grundlagen und fortgeschrittenen Themen

DAS INHALTS- VERZEICHNIS

» Hier geht's
direkt
zum Buch

Inhalt

	Einleitung	xvii
1	Einen einfachen Service einrichten	1
1.1	Die Anwendung im Überblick	1
1.2	Konfigurationsdateien verwalten	2
1.3	Mit Deployments einen replizierten Service erstellen	4
1.3.1	Best Practices für die Verwaltung von Images	4
1.3.2	Replizierte Anwendung erstellen	5
1.4	Externen Ingress für HTTP-Verkehr einrichten	7
1.5	Anwendung mit ConfigMaps konfigurieren	9
1.6	Authentifizierung mit Secrets verwalten	10
1.7	Einfache zustandsbehaftete Datenbank bereitstellen	13
1.8	TCP-Load-Balancer mithilfe von Services erstellen	17
1.9	Ingress zur Weiterleitung des Datenverkehrs an einen statischen Dateiserver verwenden	18
1.10	Ihre Anwendung mit Helm parametrisieren	20
1.11	Best Practices für die Bereitstellung von Services	22
1.12	Zusammenfassung	22
2	Workflows für Entwickler	23
2.1	Ziele	23
2.2	Aufbau eines Entwicklungclusters	24
2.3	Einen gemeinsam nutzbaren Cluster für mehrere Entwickler einrichten	26
2.3.1	Onboarding von Benutzern	26
2.3.2	Namespace erstellen und absichern	29
2.3.3	Namespaces verwalten	30
2.3.4	Services auf Clusterebene	31

2.4	Entwickler-Workflows ermöglichen	32
2.4.1	Anfängliches Setup	32
2.4.2	Aktive Entwicklung ermöglichen	33
2.4.3	Testen und Debuggen ermöglichen	34
2.5	Best Practices für das Einrichten einer Entwicklungsumgebung	35
2.6	Zusammenfassung	36
3	Monitoring und Protokollierung in Kubernetes	37
3.1	Metriken vs. Protokolle	37
3.2	Monitoringtechniken	37
3.3	Monitoringmuster	38
3.4	Kubernetes-Metriken im Überblick	39
3.4.1	cAdvisor	40
3.4.2	Metrics-Server	40
3.4.3	kube-state-metrics	41
3.5	Welche Metriken muss ich monitoren?	41
3.6	Monitoringtools	42
3.7	Kubernetes mit Prometheus überwachen	44
3.8	Protokollierung im Überblick	48
3.9	Tools für die Protokollierung	50
3.10	Protokollierung mit einem Loki-Stack	50
3.11	Warnmeldungen	53
3.12	Best Practices für Monitoring, Protokollierung und Alarmierung	54
3.12.1	Monitoring	54
3.12.2	Protokollierung	55
3.12.3	Warnmeldungen	55
3.13	Zusammenfassung	55
4	Konfiguration, Secrets und RBAC	57
4.1	Konfiguration durch ConfigMaps und Secrets	57
4.1.1	ConfigMaps	57
4.1.2	Secrets	58
4.2	Gemeinsame Best Practices für die APIs ConfigMap und Secrets	59
4.3	Best Practices speziell für Secrets	64
4.4	RBAC	65
4.4.1	RBAC-Einmaleins	66
4.5	Best Practices für RBAC	68
4.6	Zusammenfassung	70

5	Continuous Integration, Testen und Bereitstellung	71
5.1	Versionsverwaltung	72
5.2	Continuous Integration	72
5.3	Testen	72
5.4	Container-Builds	73
5.5	Tagging von Container-Images	74
5.6	Continuous Deployment	75
5.7	Bereitstellungsstrategien	75
5.8	Tests in der Produktivumgebung	80
5.9	Einrichten einer Pipeline und Durchführen eines Chaos-Experiments	82
5.9.1	CI einrichten	82
5.9.2	CD einrichten	85
5.9.3	Rolling Upgrade durchführen	85
5.9.4	Ein einfaches Chaos-Experiment	86
5.10	Best Practices für CI/CD	86
5.11	Zusammenfassung	87
6	Versionierung, Releases und Rollouts	89
6.1	Versionierung	89
6.2	Releases	90
6.3	Rollouts	91
6.4	Alles zusammenfügen	92
6.5	Best Practices für Versionierung, Releases und Rollouts	95
6.6	Zusammenfassung	96
7	Weltweite Distribution und Staging von Anwendungen	97
7.1	Ihr Image distribuieren	98
7.2	Parametrisierung Ihres Deployments	99
7.3	Lastausgleich für weltweiten Datenverkehr	100
7.4	Zuverlässiger weltweiter Rollout von Software	101
7.4.1	Validierung vor dem Rollout	102
7.4.2	Canary-Region	105
7.4.3	Typen von Regionen identifizieren	106
7.4.4	Globales Rollout planen	106
7.5	Wenn etwas schiefgeht	107
7.6	Best Practices für ein globales Rollout	109
7.7	Zusammenfassung	109

8	Ressourcenverwaltung	111
8.1	Kubernetes-Scheduler	111
8.1.1	Predicates	111
8.1.2	Prioritäten	112
8.2	Fortgeschrittene Scheduling-Techniken	113
8.2.1	Pod-Affinity und Anti-Affinity	113
8.2.2	nodeSelector	114
8.2.3	Taints und Tolerations	115
8.3	Pod-Ressourcenverwaltung	116
8.3.1	Ressourcenanforderung	117
8.3.2	Ressourcenobergrenzen und die Quality of Service von Pods	118
8.3.3	PodDisruptionBudgets	119
8.3.4	Ressourcen mit Namespaces verwalten	121
8.3.5	ResourceQuota	122
8.3.6	LimitRange	124
8.3.7	Skalierung von Clustern	125
8.3.8	Anwendungsskalierung	126
8.3.9	Skalierung mit Horizontal Pod Autoscaler (HPA)	127
8.3.10	HPA mit benutzerdefinierten Metriken	128
8.3.11	Vertical Pod-Autoscaler (VPA)	128
8.4	Best Practices der Ressourcenverwaltung	129
8.5	Zusammenfassung	130
9	Vernetzung, Netzwerksicherheit und Service Meshes	131
9.1	Grundsätze des Kubernetes-Netzwerks	131
9.2	Netzwerk-Plug-ins	134
9.2.1	Kubenet	135
9.2.2	Best Practices für Kubenet	135
9.2.3	Das CNI-Plug-in	135
9.2.4	Best Practices für CNI	136
9.3	Services in Kubernetes	136
9.3.1	Service-Typ ClusterIP	137
9.3.2	Service-Typ NodePort	139
9.3.3	Service-Typ ExternalName	140
9.3.4	Service-Typ LoadBalancer	140
9.3.5	Ingress und Ingress-Controller	142
9.3.6	Gateway-API	143
9.3.7	Best Practices für Services und Ingress-Controller	145

9.4	Netzwerksicherheitsrichtlinien	146
9.5	Best Practices für Netzwerksicherheitsrichtlinien	148
9.6	Service Meshes	150
9.7	Best Practices für Service Meshes	151
9.8	Zusammenfassung	152
10	Pod- und Container-Sicherheit	153
10.1	Pod Security Admission Controller	153
10.1.1	Pod Security Admission Controller aktivieren	154
10.1.2	Pod-Sicherheitsstandards	154
10.1.3	Pod-Sicherheit mit Namespace-Labels aktivieren	155
10.2	Workload-Isolierung und RuntimeClass	156
10.2.1	RuntimeClass verwenden	157
10.2.2	Laufzeit-Implementierungen	158
10.2.3	Best Practices für Workload-Isolierung und RuntimeClass	158
10.3	Weitere Überlegungen zur Pod- und Container-Sicherheit	159
10.3.1	Admission Controller	159
10.3.2	Werkzeuge zur Erkennung von Angriffen und Anomalien	159
10.4	Zusammenfassung	160
11	Policy und Governance für Ihren Cluster	161
11.1	Warum Policy und Governance wichtig sind	161
11.2	Was ist an dieser Policy anders?	161
11.2.1	Cloud Native Policy Engine	162
11.3	Einführung in Gatekeeper	162
11.3.1	Beispielrichtlinien	163
11.3.2	Gatekeeper-Terminologie	163
11.3.3	Einschränkungsvorlagen definieren	164
11.3.4	Constraints definieren	166
11.3.5	Datenreplikation	167
11.3.6	UX	167
11.4	Durchsetzungsmaßnahmen und Audits verwenden	168
11.4.1	Mutation	170
11.4.2	Richtlinien testen	170
11.4.3	Sich mit Gatekeeper vertraut machen	170
11.5	Best Practices für Policy und Governance	170
11.6	Zusammenfassung	171

12	Verwaltung mehrerer Cluster	173
12.1	Warum mehrere Cluster?	173
12.2	Herausforderungen beim Multi-Cluster-Design	175
12.3	Multi-Cluster-Bereitstellungen verwalten	177
12.4	Deployment- und Managementmuster	178
12.5	Der GitOps-Ansatz zur Verwaltung von Clustern	179
12.6	Tools für das Multi-Cluster-Management	181
12.7	Kubernetes Federation	182
12.8	Best Practices für die Verwaltung mehrerer Cluster	183
12.9	Zusammenfassung	184
13	Externe Services in Kubernetes integrieren	185
13.1	Services in Kubernetes importieren	185
13.1.1	Services ohne Selektor für stabile IP-Adressen	186
13.1.2	CNAME-basierte Dienste für stabile DNS-Namen	187
13.1.3	Aktive Controller-basierte Ansätze	188
13.2	Services aus Kubernetes exportieren	189
13.2.1	Dienste mit internen Load Balancern exportieren	190
13.2.2	Services auf NodePorts exportieren	190
13.2.3	Integration von externen Maschinen und Kubernetes	192
13.3	Gemeinsame Nutzung von Services zwischen Kubernetes-Clustern	193
13.4	Tools von Drittanbietern	194
13.5	Best Practices für die Verbindung von Cluster und externen Services	194
13.6	Zusammenfassung	195
14	Maschinelles Lernen in Kubernetes ausführen	197
14.1	Warum eignet sich Kubernetes hervorragend für maschinelles Lernen?	197
14.2	Workflow für maschinelles Lernen	198
14.3	Maschinelles Lernen für Kubernetes-Cluster-Administratoren	199
14.3.1	Modell auf Kubernetes trainieren	200
14.3.2	Verteiltes Training auf Kubernetes	203
14.3.3	Ressourcenbeschränkungen	204
14.3.4	Spezialisierte Hardware	204
14.3.5	Bibliotheken, Treiber und Kernel-Module	205
14.3.6	Storage	205
14.3.7	Vernetzung	206
14.3.8	Spezialisierte Protokolle	207

14.4	Tools für Datenwissenschaftler	207
14.5	Best Practices für maschinelles Lernen auf Kubernetes	208
14.6	Zusammenfassung	209
15	Auf Basis von Kubernetes übergeordnete Anwendungs-Patterns erstellen	211
15.1	Ansätze zur Entwicklung von Abstraktionen auf höherer Ebene . . .	211
15.2	Kubernetes erweitern	212
15.2.1	Kubernetes-Cluster erweitern	213
15.2.2	Kubernetes-User Experience erweitern	214
15.2.3	Containerisierte Entwicklung einfacher machen	215
15.2.4	Eine »Push-to-Deploy«-Erfahrung entwickeln	215
15.3	Designüberlegungen beim Erstellen von Plattformen	216
15.3.1	Unterstützung für den Export in ein Container-Image . . .	216
15.3.2	Bestehende Mechanismen für Service und Service Discovery unterstützen	217
15.4	Best Practices für den Aufbau von Anwendungsplattformen	217
15.5	Zusammenfassung	218
16	Status und zustandsbehaftete Anwendungen verwalten	219
16.1	Volumes und Volume Mounts	220
16.2	Best Practices für Volumes	221
16.3	Kubernetes-Datenspeicher	221
16.3.1	PersistentVolume	221
16.3.2	PersistentVolumeClaims	222
16.3.3	StorageClass	223
16.3.4	Best Practices für Kubernetes-Datenspeicher	224
16.4	Zustandsbehaftete Anwendungen	225
16.4.1	StatefulSets	226
16.4.2	Operatoren	228
16.4.3	Best Practices für StatefulSets und Operatoren	229
16.5	Zusammenfassung	230
17	Zugangskontrolle und Autorisierung	231
17.1	Zugangskontrolle	232
17.1.1	Was sind Admission Controller?	232
17.1.2	Warum sind Admission Controller wichtig?	232
17.1.3	Arten von Zugangscontrollern	233
17.1.4	Konfigurieren von Zugangs-Webhooks	233
17.1.5	Best Practices für die Zugangskontrolle	236

17.2	Autorisierung	239
17.2.1	Autorisierungsmodule	239
17.2.2	ABAC	240
17.2.3	RBAC	241
17.2.4	Webhook	242
17.2.5	Best Practices bei der Autorisierung	242
17.3	Zusammenfassung	242
18	GitOps und Bereitstellung	243
18.1	Was ist GitOps?	244
18.2	Warum GitOps?	245
18.3	GitOps Repo-Struktur	247
18.4	Secrets verwalten	248
18.5	Flux einrichten	250
18.6	GitOps-Tools	252
18.7	Best Practices für GitOps	253
18.8	Zusammenfassung	253
19	Sicherheit	255
19.1	Clustersicherheit	256
19.1.1	Zugriff auf etcd	256
19.1.2	Authentifizierung	256
19.1.3	Autorisierung	256
19.1.4	TLS	257
19.1.5	Kubelet und Zugriff auf Cloud-Metadaten	257
19.1.6	Secrets	257
19.1.7	Protokollierung und Überwachung	258
19.1.8	Tools für Cloud Security Posture Management (CSPM)	258
19.2	Best Practices für die Clustersicherheit	258
19.3	Container-Sicherheit auf der Workload-Ebene	259
19.3.1	Pod-Sicherheit	259
19.3.2	Seccomp, AppArmor und SELinux	259
19.3.3	Admission Controller	260
19.3.4	Operatoren	260
19.3.5	Netzwerk Policy	260
19.3.6	Sicherheit der Laufzeitumgebung	261
19.3.7	Best Practices für die Sicherheit von Workload- Containern	261

19.4	Codesicherheit	262
19.4.1	Non-Root und Distroless-Container	262
19.4.2	Container auf Schwachstellen scannen	262
19.4.3	Sicherheit des Code-Repositorys	263
19.4.4	Best Practices für die Codesicherheit	263
19.5	Zusammenfassung	264
20	Chaos Engineering, Lasttests und Experimente	265
20.1	Chaos Engineering	265
20.1.1	Ziele für Chaos Engineering	266
20.1.2	Voraussetzungen für Chaos Engineering	266
20.1.3	Chaosexperiment für die Kommunikation Ihrer Anwendung	267
20.1.4	Chaosexperiment für den Betrieb Ihrer Anwendung	268
20.1.5	Fuzz-Testing Ihrer Anwendung für Sicherheit und Ausfallsicherheit	269
20.1.6	Zusammenfassung	269
20.2	Lasttests	269
20.2.1	Ziele für Lasttests	270
20.2.2	Voraussetzungen für Lasttests	271
20.2.3	Realistischen Traffic generieren	271
20.2.4	Lasttest Ihrer Anwendung	272
20.2.5	Optimieren Sie mit Lasttests Ihre Anwendung	273
20.2.6	Zusammenfassung	274
20.3	Experimente	274
20.3.1	Ziele für Experimente	274
20.3.2	Voraussetzungen für ein Experiment	275
20.3.3	Aufbau eines Experiments	275
20.3.4	Zusammenfassung	277
20.4	Zusammenfassung	277
21	Einen Operator implementieren	279
21.1	Schlüsselkomponenten von Operatoren	280
21.2	Custom Resource Definitions	280
21.3	Unsere API erstellen	282
21.4	Reconciliation: Ist- und Soll-Zustand abgleichen	289
21.5	Ressourcen-Validierung	291
21.6	Controller-Implementierung	291

21.7	Lebenszyklus des Operators	296
21.7.1	Versionsupgrades	297
21.7.2	Best Practices für Operatoren	298
21.8	Zusammenfassung	300
22	Schlussfolgerung	301
	Index	303