

# Skalierbare KI/ML-Infrastrukturen

Evaluieren, Automatisieren, Praxis

# DAS INHALTS- VERZEICHNIS

» Hier geht's  
direkt  
zum Buch

# Inhalt

<b>1</b>	<b>Vorwort</b>	19
<b>1.1</b>	<b>Vorbemerkungen</b>	28
1.1.1	Verwendete Formatierungen	29
1.1.2	Breites Buch-/Seitenformat	29
1.1.3	Klartext	29
1.1.4	KI/ML-Begrifflichkeiten	30
1.1.5	Weiterführende Hinweise	30
1.1.6	Verwendete Testsysteme	30
1.1.7	Im Buch verwendete Grafiken	31
<b>1.2</b>	<b>Was dieses Buch sein bzw. nicht sein soll</b>	31
1.2.1	Was es sein soll	31
1.2.2	Was es nicht sein soll – und nicht ist	32
1.2.3	Scope und Fokus des Buches	32
1.2.4	Wissensaufbau	33
<b>1.3</b>	<b>Wie dieses Buch zu lesen ist</b>	34
<b>1.4</b>	<b>Thematischer Überblick – was wird in welchen Kapiteln behandelt</b>	34

## TEIL I Technische Foundations zu skalierbaren KI/ML-Infrastrukturen

<b>2</b>	<b>Am Anfang war die Dunkelheit</b>	41
<b>2.1</b>	<b>Eine kurze Einführung: KI/ML-Systeme – und alles wird gut. Oder eher nicht?</b>	42
2.1.1	Historisches – kurz und kompakt	42
2.1.2	KI as a panacea?	43
2.1.3	Eine kurze Einordnung: KI, Machine Learning, neuronale Netze und Deep Learning	44
<b>2.2</b>	<b>Use Cases für KI/ ML-Anwendungen – Auszüge</b>	45
2.2.1	Wer profitiert vom Einsatz von KI/ML-Systemen? Mögliche Use Cases im Überblick	45
2.2.2	Exemplarische Use Cases	47

<b>2.3 Fehlerfreie KI? Sicher nicht.</b>	50
2.3.1 Regeln und Transparenz	50
2.3.2 Lösungsansätze	50
2.3.3 Vorbereitung	51
<b>2.4 Einige Grundbegrifflichkeiten im KI/ML-Kontext</b>	52
2.4.1 Machine Learning: Training und Inference	52
2.4.2 CNN (Convolutional Neural Networks)	53
2.4.3 Alles fließend: FP/TF/BF (Floating Point)	54
2.4.4 GPUs und Parallel-Computing	55
2.4.5 CPUs mit ML-Erweiterungen	56
2.4.6 CUDA (Cores)	57
2.4.7 Tensor (Cores)	58
2.4.8 Präzision, Performance und Kosten	60
2.4.9 Tensor-Core-Effizienz und Mixed/Reduced Precision	60
2.4.10 CUDA-Cores vs. Tensor-Cores	62
2.4.11 Und noch einmal Performance: NVIDIA Hopper, TMA, Transformer-Engine und FP8	62

### **3 High-Level-Vorbetrachtungen zur Implementierung von skalierbaren KI/ML-Infrastrukturen** 65

---

<b>3.1 Bare-Metal, Virtualisierung, Containerisierung</b>	65
3.1.1 Bare-Metal vs. Virtualisierung	65
3.1.2 Containerisierung	66
3.1.3 Die Kernkomponenten/-Layer des (auto-)skalierbaren KI/ML-Infra-Stacks	68
<b>3.2 Generelle Infrastruktur-Fragen: Cloud vs. On-Prem, Managed Server, hybrider Mischbetrieb, dedizierte KI-Plattformen (NVIDIA DGX)</b>	69
3.2.1 Implementierungs- und Kostenfaktoren in der Cloud	69
3.2.2 Exkurs: Managed Server kleinerer SPs als günstigere Cloud-Alternative mit höherer Flexibilität?	72
3.2.3 Implementierungs- und Kostenfaktoren: Self-Hosted	72
3.2.4 Datensicherheit	75
3.2.5 Storage	75
3.2.6 Netzwerk	76
3.2.7 Hybrider Ansatz: On-Prem und Cloud (Pay-per-Use)	76
3.2.8 Alles cool? In der Cloud oft eher nicht. Temperatur-, Performance- und damit Kostenfragen.	77
3.2.9 Generelle Funktions- und Lizenzkosten-Betrachtungen: vGPU vs. MIG	79

3.2.10	Für größere Budgets: Out-of-the-Box-, Ready-to-use-ML-Server (NVIDIA DGX) .....	81
3.2.11	DGX: Technische Eckdaten und Blick unter die Haube .....	82
3.2.12	HPE ML – und wieder NVIDIA .....	84
3.2.13	Miete von RZ-tauglicher Hardware und Bereitstellung im eigenen RZ .....	84
<b>3.3</b>	<b>Entscheidungshilfe: Reguläre GPU-Server, KI/ML-Boliden wie DGX oder alles in die Cloud?</b> .....	<b>85</b>
3.3.1	KMU .....	85
3.3.2	Größere Unternehmen und Konzerne .....	86
<b>3.4</b>	<b>Generelle GPU-Hardware-Fragen: NVIDIA vs. AMD vs. Intel vs. Googles TPU</b> .....	<b>86</b>
3.4.1	Vorbetrachtungen: Was darf es denn sein? .....	86
3.4.2	GPU vs. TPU ... oder doch gemeinsam? .....	88
3.4.3	NVIDIA .....	88
3.4.4	AMD .....	90
3.4.5	Intel .....	91
3.4.6	Fazit: GPU-Provider .....	92

## **4 NVIDIA-Datacenter-GPUs und mehr – technischer Background** 93

---

<b>4.1</b>	<b>NVIDIA und ML-Cluster</b> .....	<b>93</b>
<b>4.2</b>	<b>Partitionierte GPUs mit NVIDIAs vGPU und MIG</b> .....	<b>95</b>
4.2.1	Vorbetrachtungen und Scope .....	95
4.2.2	Vorbetrachtungen: Partitionierte GPUs mit vGPU und MIG .....	96
4.2.3	NVIDIAs vGPU und (leider noch kein) Cloud-Einsatz .....	97
<b>4.3</b>	<b>vGPU – Virtual GPU</b> .....	<b>97</b>
4.3.1	Generelle vGPU-Architektur .....	97
4.3.2	Details zur Funktionsweise .....	98
4.3.3	Time-Sliced vGPU .....	100
4.3.4	Passthrough GPU vs. vGPU im Hypervisor .....	101
4.3.5	vGPU-Produkte .....	102
4.3.6	vGPU-Versionen .....	102
4.3.7	vGPU-Features je nach Produkt .....	103
4.3.8	vGPU-Arbeitsweise (konzeptionell) .....	105
4.3.9	Scheduling-Policies von vGPU .....	105
4.3.10	vGPU-Profile und Zuordnung (exemplarisch: NVIDIA A100 40 GB) .....	108
4.3.11	Konkrete vGPU-Profile und Details .....	109
4.3.12	Exemplarische vGPU-Partitionslayouts .....	111

4.3.13	Erforderliche vGPU-Lizenzen und Entitlements je nach Modell und Typ .....	112
4.3.14	Übersicht der vGPU-Modi nach GPUs/Karten .....	112
<b>4.4</b>	<b>MIG – Multi-Instance GPU</b> .....	<b>113</b>
4.4.1	MIG-fähige GPUs .....	114
4.4.2	MIG-Konzepte, Terminologien und technische Details .....	115
4.4.3	Allgemeine MIG-Tech-Specs .....	118
4.4.4	MIG-Instanzen und -Partitionen .....	120
4.4.5	Compute-Instanzen/Compute-Sub-Partitionierung .....	121
4.4.6	MIG-Profilübersichten (A100 und A30) .....	122
4.4.7	Der A100-MIG-Black-Hole-Effekt, oder: die verschwundenen Partitionen .....	124
4.4.8	MIG-Strategien .....	126
4.4.9	MIG-Exposition: gesamte GPU via Passthrough oder MIG-Partitionen per VM-Template? .....	127
<b>4.5</b>	<b>MIG: Multi-Tenancy revisited</b> .....	<b>128</b>
4.5.1	Modelle und Konzepte, Vor- und Nachteile, plattformspezifische Limitierungen .....	128
4.5.2	MIG, echte Tenancy – und Next-Gen-MIG mit Confidential Computing .....	129
4.5.3	Nicht nur MIG: Das Problem, Passthrough GPUs (auto-)skalierbar anzubieten .....	133
<b>4.6</b>	<b>Technische Daten und Preise ausgewählter NVIDIA Datacenter-GPUs</b> .....	<b>134</b>
4.6.1	Supported NVIDIA GPUs Optimized for Compute (AI/ML) Workloads .....	135
4.6.2	Supported NVIDIA GPUs Optimized for Mixed Workloads .....	136
<b>4.7</b>	<b>GPU-Time-Slicing und GPU-Overcommitment</b> .....	<b>137</b>
4.7.1	Theoretische Vorbetrachtungen .....	137
4.7.2	Konzepte zur Umsetzung .....	139
<b>4.8</b>	<b>NVLink und NVSwitch: GPU Big Blocks – Bündelung multipler GPUs</b> .....	<b>139</b>
4.8.1	NVLink .....	139
<b>4.9</b>	<b>GPUDirect (RDMA)</b> .....	<b>142</b>
4.9.1	GPUDirect Storage .....	142
4.9.2	GPUDirect RDMA .....	143
<b>4.10</b>	<b>GPU-Performance in ML-Trainings – Bare-Metal vs. vGPU/MIG</b> .....	<b>145</b>
4.10.1	Vorbetrachtungen .....	145
4.10.2	Konkretes Setup und Messwerte .....	146
<b>4.11</b>	<b>NVIDIA-Datacenter-Produkte: The Road Ahead</b> .....	<b>149</b>

## TEIL II Implementierung von skalierbaren KI/ML-Infrastrukturen

<b>5</b>	<b>Implementierung: vSphere als Hypervisor für skalierbare ML-Infrastrukturen</b>	153
<b>5.1</b>	<b>Hardware-Voraussetzungen und Vorbetrachtungen (vSphere/On-Prem)</b>	153
<b>5.2</b>	<b>Preflights</b>	154
5.2.1	BIOS/UEFI-Settings, SR-IOV, vSphere Edition, DRS	154
5.2.2	vSphere 7 und ESXi-Patchlevel	155
5.2.3	Update 7U3 und VMClasses	156
5.2.4	Host-Updates für vSphere/ESXi	156
5.2.5	vGPU für RTX 6000/8000 und RTX A5000/A6000 aktivieren	156
5.2.6	ECC-Memory	157
5.2.7	Preflight-Checks: Tools und Tests	157
5.2.8	Virtualization-Mode (Achtung: wichtig!)	159
<b>5.3</b>	<b>Setup des GPU-Managers/vGPU-Host-Drivers (ESXi/vSphere 7)</b>	160
5.3.1	NVD-AIE oder NVD-VGPU, NVIDIA vGPU Certified Server	160
5.3.2	Setup des NVIDIA-VIB (vGPU-Manager) auf den ESXis	161
5.3.3	PoC: Einfaches Passthrough	162
<b>5.4</b>	<b>VM-Templates mit GPUs erstellen</b>	164
5.4.1	Erstellung und Konfiguration eines VM-Templates (vGPU-Variante, OpenShift)	165
5.4.2	Erstellung und Konfiguration eines VM-Templates (MIG-backed vGPU-Variante, OpenShift)	167
5.4.3	Erstellung eines skalierbaren PCI-Passthrough-VM-Templates (identische GPUs per Dynamic DirectPath IO und Hardware-Bezeichner)	169
5.4.4	Die GPU-Power-Modi (P0–P8) und (Idle-)Leistungsaufnahme	170
5.4.5	Checkliste für mögliche Fehler beim vGPU-Betrieb	170
<b>5.5</b>	<b>MIG-Mode auf dem Hypervisor aktivieren</b>	171
5.5.1	Setup – Vorbereitungen	171
5.5.2	GPU (ohne Reboot des ESXi-Hosts) auf MIG-Mode umstellen	173
5.5.3	Manuelle Partitionierung anlegen (nur zur Veranschaulichung)	176

---

<b>6</b>	<b>Der NVIDIA AI Enterprise (NVAIE)-Stack – infrastrukturelevante Betrachtungen</b>	181
<hr/>		
<b>6.1</b>	<b>Vorbetrachtungen</b>	181
<b>6.2</b>	<b>Motivation</b>	182
<b>6.3</b>	<b>Plattformen für NVAIE</b>	183
<b>6.4</b>	<b>NVAIE vs. vGPU vs. Free GPU Operator</b>	185
6.4.1	VIBs	185
6.4.2	GPU-Operator	185
6.4.3	NVAIE-Features	186
<b>6.5</b>	<b>NVAIE in der Public Cloud</b>	186
<b>6.6</b>	<b>NVAIE ist Pflicht für skalierbare ML-Cluster?</b>	187
<b>6.7</b>	<b>NVAIE als AI-End-to-End-Plattform</b>	187
<b>7</b>	<b>vGPU-/NVAIE-Preflights: Lizenzierung</b>	189
<hr/>		
<b>7.1</b>	<b>Grundsätzliches: vGPU- vs. NVAIE-Lizenzen und DLS vs. CLS</b>	189
7.1.1	Preise und SLAs: vGPU	190
7.1.2	Preise und SLAs: NVAIE	191
7.1.3	NVIDIA-Entitlement beantragen	193
<b>7.2</b>	<b>NVIDIA Licensing System (NLS)</b>	194
7.2.1	Vorbetrachtungen	194
7.2.2	License Server: Self-Hosted License Server und Alternativen	195
<b>7.3</b>	<b>License Server: DLS vs. CLS</b>	196
<b>7.4</b>	<b>Self-Hosted License Server: DLS und Legacy License Server</b>	197
7.4.1	License Server – Legacy-Variante (self-hosted)	197
7.4.2	License Server – DLS-Variante (DLS Virtual Appliance, On-Prem)	200
7.4.3	Troubleshooting – Token Debugging	204
<b>7.5</b>	<b>Cloud-Hosted License Server: CLS</b>	205
7.5.1	License Server – CLS-Variante (cloudbasiert)	205

---

<b>8</b>	<b>Kubernetes-basierte Plattformen für skalierbare, GPU-Accelerated KI/ML-Cluster</b>	209
<b>8.1</b>	<b>The Road so far</b>	209
<b>8.2</b>	<b>Generelle Plattform-Fragen: (Vanilla-)Kubernetes-Derivate und OpenShift im Überblick</b>	211
<b>8.3</b>	<b>Vanilla Kubernetes</b>	213
8.3.1	Test and Play	213
8.3.2	Benötigte 3rd Party Tools und asynchrone Produktzyklen	214
8.3.3	Vanilla Kubernetes und das traurige Thema LTS: Geld verbrennen? Oder besser doch nicht?	214
8.3.4	Releases, Changes und kein Ende	215
8.3.5	Vanilla Kubernetes und TTM-Märchenstunden	216
8.3.6	AKS, EKS, GKE & Co.	217
<b>8.4</b>	<b>VMwares Tanzu und das Eckige, das durchs Runde soll</b>	217
8.4.1	Historisches	217
8.4.2	Tanzu	218
<b>8.5</b>	<b>OpenShift</b>	219
<b>8.6</b>	<b>Abschließende LTS-Betrachtungen</b>	220
<b>8.7</b>	<b>Kubernetes-Basics – Aufbau des Systems</b>	222
8.7.1	Kernkomponenten und Konzepte	222
8.7.2	Kubernetes-spezifische Dienste auf den Master-Nodes (Controlplane)	224
8.7.3	Kubernetes-spezifische Dienste auf allen Nodes	224
<b>8.8</b>	<b>Kubernetes-Basics – Ressourcen/Workloads</b>	225
8.8.1	Kubernetes/OpenShift: API-Version und API-Ressourcen	225
8.8.2	Namespaces	226
8.8.3	Pods	227
8.8.4	Pod-Metahüllen: Deployments, StatefulSets, DaemonSets	228
8.8.5	ConfigMaps	232
8.8.6	Node-Objekte und Node-Label	232
8.8.7	Services	233
8.8.8	Ingress und Routen	234
<b>8.9</b>	<b>Sonstige im Folgenden verwendete, Kubernetes-spezifische Tools</b>	236
8.9.1	kubectl- und oc-Bash-Completion und kubectl-Alias	236
8.9.2	Kustomize und Helm	238



---

<b>9</b>	<b>Preflights für GPU-Accelerated Container-Cluster: Operatoren</b>	241
<b>9.1</b>	<b>Generelle Vorbetrachtungen zum Thema Operatoren</b>	241
9.1.1	Einführung	242
9.1.2	Was ist ein Operator?	242
9.1.3	Horizontal? Vertikal? Beides?	245
9.1.4	Controller-Loops	245
9.1.5	Operator-Kategorien	246
9.1.6	Red Hats Operator Framework und Operator-SDK	246
<b>9.2</b>	<b>Operator-Typen und Maturitäts-Level: Helm vs. Ansible vs. Go</b>	247
9.2.1	Operator-Maturitäts-Level und -Kategorien	247
9.2.2	Operator-Build	248
9.2.3	Operatorhub.io und OpenShift-Operatoren	249
<b>9.3</b>	<b>Die wichtige Rolle von Operatoren im auto-skalierbaren KI/ML-Stack</b>	250
9.3.1	Team-Play	250
9.3.2	Der GPU-Operator – Vergangenheit und Zukunft	250
<b>9.4</b>	<b>NVIDIAs GPU-Operator – die Architektur</b>	251
9.4.1	Das Gesamtkonstrukt	251
9.4.2	GPU-Operator: Unterstützte Hypervisoren und GPUs	252
9.4.3	All-in-One	253
9.4.4	GPU-Operator und DGX	253
9.4.5	Die Einzelkomponenten des GPU-Operators im High-Level-Überblick	253
9.4.6	Preflight: der NFD-Operator	256
9.4.7	Die Einzelkomponenten des GPU-Operators im Detail	257
<b>9.5</b>	<b>Automatische Provisionierung eines Nodes durch den GPU-Operator</b>	258
9.5.1	K8s-Device-Plugin	259
9.5.2	GPU Feature Discovery	259
9.5.3	Driver	261
9.5.4	Container-Toolkit	261
9.5.5	DCGM/DCGM-Exporter	261
9.5.6	Der MIG-Manager	261
9.5.7	MIG-Manager und assoziierte ConfigMap	264
9.5.8	MIG-Strategies: »mixed« vs. »single« in der Praxis	266
9.5.9	Custom-MIG-ConfigMap	267
<b>9.6</b>	<b>NVIDIAs Network-Operator – die Architektur</b>	268
9.6.1	Vorbetrachtungen und Übersicht	268
9.6.2	Arbeitsweise (High-Level)	269

<b>9.7</b>	<b>Komponenten des Network-Operators im Überblick</b> .....	270
9.7.1	Mofed (NVIDIA_MLNX_OFED) Driver .....	270
9.7.2	Kubernetes RDMA Shared Device Plugin .....	271
9.7.3	NVIDIA Peer Memory Driver .....	271
9.7.4	Sonstige wichtige Komponenten .....	271
<b>10</b>	<b>OpenShift (GPU-Accelerated) – Multiplattform (Cloud und On-Premises)</b> .....	273
<hr/>		
<b>10.1</b>	<b>Theoretische Vorbetrachtungen</b> .....	273
10.1.1	Preflights: NVIDIA-Entitlements/-Lizenzen, Lizenzserver .....	274
10.1.2	Funktionsweise – High-Level-Überblick .....	274
<b>10.2</b>	<b>Konzeptionelle Vorbetrachtungen zum Setup (On-Prem mit vSphere)</b> .....	275
10.2.1	Überblick .....	275
10.2.2	Setup-Prozeduren GPU-Accelerated OpenShift IPI on vSphere – schematisch .....	276
<b>10.3</b>	<b>On-Premises: OpenShift 4.10-Setup – Installer Provisioned Infrastructure (IPI) auf vSphere</b> .....	277
10.3.1	Preflights: Infrastruktur und OpenShift-Cluster .....	277
10.3.2	Generelle Tool-Hinweise zu allen OpenShift-Setups (AWS, GCP, vSphere & Co.) .....	277
10.3.3	Der OpenShift-Installer: Terraform in schön .....	278
10.3.4	Vorbetrachtungen: Cluster Sizing .....	279
10.3.5	Zusammenfassung der technischen Preflights für das vSphere-Setup .....	279
10.3.6	Achtung, wichtig: DNS-Settings .....	280
10.3.7	DNS-Reverse-Zonen .....	281
10.3.8	vSphere-HA und OpenShift-Installer (OVA Upload fails in Single Datastore) .....	281
10.3.9	install-config.yaml für vSphere-IPI-Installation (Auszüge) .....	282
10.3.10	Rollout .....	283
10.3.11	Der Post-Rollout-Zustand .....	286
<b>10.4</b>	<b>Preflights für skalierbare GPU-Nodes unter OpenShift: MachineSets, MachineConfigs und Machine-/Cluster-Autoscaler</b> .....	286
10.4.1	Vorbetrachtungen .....	287
10.4.2	Cluster-Operatoren und Machine*-Ressourcen .....	287
10.4.3	MachineConfigs .....	289
10.4.4	MachineConfig-Operator .....	290
10.4.5	Komponenten des MCO .....	290

10.4.6	MachineConfigPool .....	291
10.4.7	Machines und MachineSets, Skalierung .....	292
<b>10.5</b>	<b>Cluster-Autoscaler/Machine-Autoscaler .....</b>	<b>294</b>
10.5.1	High-Level-Betrachtung .....	294
10.5.2	Machine-Autoscaler .....	295
10.5.3	Cluster-Autoscaler .....	295
10.5.4	Thresholds .....	296
10.5.5	Zu beachtende Punkte .....	297
10.5.6	GPU-VM-Template (vSphere) in MachineSet einbinden .....	298
10.5.7	GPU-MachineConfigPool und customisiertes MachineSet für skalierbare GPU-Nodes erzeugen .....	299
10.5.8	Skalierung des neuen GPU-MachineSets .....	303
10.5.9	Exemplarische Erzeugung eines GPU-MachineSets unter AWS .....	304
10.5.10	Fazit .....	306
<b>10.6</b>	<b>vGPU-/MIG-spezifisches Setup des OpenShift-Clusters: NFD- und GPU-Operator .....</b>	<b>306</b>
10.6.1	Historisches – NVIDIA-Driver-Build mit Red Hat Entitlements .....	306
10.6.2	Kernel für Driver-DaemonSet zu neu? Achtung bei OpenShift-Release-Updates .....	307
10.6.3	Installationsverfahren, generelle Operator-Settings .....	307
10.6.4	GPU-Manager-managed MIG-Mode und vGPU .....	308
10.6.5	NFD-Operator-Installation und -Konfiguration .....	308
10.6.6	GPU-Operator-Installation und -Konfiguration .....	311
10.6.7	License-ConfigMap .....	313
10.6.8	ImagePullSecret für Driver-Images aus der NGC-Registry .....	315
10.6.9	Die ClusterPolicy-CR (GPU-Operator) .....	316
<b>10.7</b>	<b>Automatisches vGPU-Node-Setup per Operator – OpenShift-MachineSet mit Tesla T4 .....</b>	<b>320</b>
10.7.1	Rollout der ClusterPolicy-CR .....	320
10.7.2	Status auf den ESXi-Hosts .....	322
10.7.3	Analyse des ausgerollten (v)GPU-Stacks .....	324
<b>10.8</b>	<b>Automatisches MIG-Slice-Setup per Operator – A30 on-premises .....</b>	<b>327</b>
10.8.1	MIG im PCI Passthrough (A30 on-premises), Partitionierung durch den MIG-Manager .....	328
10.8.2	OpenShift-MachineSet und Default-MIG-Settings .....	328
10.8.3	Skalierung des MachineSets .....	330
10.8.4	Teilen? Oder lieber doch nicht? .....	332

<b>10.9 Cloud: GPU-MachineSets in OpenShift 4.10 unter GCP mit A100-Instanzen (MIG-Partitionen via Operator)</b>	333
10.9.1 Vorbetrachtungen	333
10.9.2 Verfügbare VM-Instanzen (GCP) mit GPU	334
10.9.3 Setup-Prozeduren – schematisch	334
10.9.4 Preflights – GCP-Kontingente gegebenenfalls erhöhen	335
10.9.5 Preflights – Domain, DNS und APIs	335
10.9.6 Service-Account zur OpenShift-Cluster-Erzeugung	337
10.9.7 Anpassungen der install-config.yaml, Rollout des Clusters	338
10.9.8 Setup der GPU-Nodes	341
10.9.9 Extraktion, Anpassung und Re-Import MachineSet und MCP	342
10.9.10 Skalierung des neuen GPU-MachineSets	345
10.9.11 Check der provisionierten GPU-Nodes	346
10.9.12 NFD- und GPU-Operator	346
10.9.13 MIG-Mode aktivieren, MIG-Partition-Size für A100 einstellen	348
10.9.14 Debugging und Troubleshooting	352
<b>10.10 GPU-Sharing/-Overcommitment</b>	353
10.10.1 Konzept-Recap und praktische Umsetzung	353
10.10.2 Setup (OpenShift)	355
10.10.3 Shared Workload testen	356
10.10.4 GPU-Sharing-Konfiguration per Node zur Laufzeit ändern	358
10.10.5 GPU Sharing mit vGPU	359
10.10.6 GPU-Sharing mit MIG-Slices	360
10.10.7 GPU-Sharing in der GCP-Cloud als kurzes PoC	366
<b>10.11 Setup des Network-Operators (OpenShift on vSphere [IPI]) für GPUDirect RDMA</b>	371
10.11.1 Preflights	371
10.11.2 High-Level-Workflow für den Network-Operator	378
10.11.3 Network-Operator und NFD-CR	379
10.11.4 Tests nach erfolgreichem Rollout	380
10.11.5 GPUDirect-RDMA-Test mit MacVLAN	382
10.11.6 Connect-Tests	383
10.11.7 Ein (nicht wirklich rundes) Fazit	385
<b>10.12 KI/ML-System-Performance-Test (OpenShift on DGX)</b>	386
<b>10.13 GPU-Dashboard für OpenShift</b>	387

---

## 11 GKE – Google Kubernetes Engine Cluster (GPU-Accelerated) 389

---

<b>11.1 Überblick</b> .....	389
11.1.1 Generelle Preflights: GPU-Verfügbarkeit nach Regionen/Zonen, geeignete Instanztypen .....	389
<b>11.2 Setup-Variante 1: GKE-Cluster mit separatem Node-Pool für GPU-Nodes</b> .....	390
11.2.1 Setup .....	390
11.2.2 Rollout des GPU-Operators .....	392
<b>11.3 Setup-Variante 2: GPU-Cluster auf GKE direkt ausrollen</b> .....	395

## TEIL III ML-Stacks für skalierbare KI/ML-Infrastrukturen

## 12 CI/CD-Pipelines, GitOps und MLOps 399

---

<b>12.1 Von der (ML-)Insel zur Pipeline</b> .....	399
<b>12.2 CI/CD und GitOps</b> .....	400
12.2.1 CI/CD .....	400
12.2.2 GitOps .....	401
<b>12.3 GitOps-Pipeline-Modelle</b> .....	401
12.3.1 Pull- vs. Push-based .....	401
12.3.2 Push-based .....	402
12.3.3 Pull-based .....	403
12.3.4 Multiple Stages/Applications .....	404
<b>12.4 MLOps, LTS und Portierbarkeit</b> .....	404
12.4.1 MLOps und CRISP-DM .....	406
12.4.2 MLOps und ML-Pipelines – technische Foundation/schematisch .....	407

---

<b>13 ML-Pipeline- und AI-End-to-End-Implementierungen mit Kubeflow/Vertex AI, Open Data Hub und NVIDIA AI Enterprise</b>	411
<b>13.1 ML-Pipeline-Implementierungen in Kubernetes-basierten Clustern</b>	411
13.1.1 Der (KI/ML-)Pipeline-Ansatz	411
13.1.2 End-to-End-AI-Plattformen und Workflows	412
13.1.3 Das generelle Findungsproblem	413
13.1.4 Containerisierte ML-Pipelines und Segen und Fluch der Modularität	414
13.1.5 Kubernetes/Kubeflow to the Rescue? Genau betrachtet eher (noch) nicht.	415
13.1.6 Eine Lösung ...	416
<b>13.2 Kubeflow</b>	417
13.2.1 Kubeflow-Komponenten im Überblick	417
13.2.2 Entwicklung und Module (Auszüge)	418
13.2.3 Die Kernkomponenten	419
13.2.4 All together?	420
13.2.5 Istio	421
13.2.6 Kubeflow war gestern – es lebe Vertex AI. Na, zumindest ganz sicher bis ... sagen wir mal: morgen Mittag.	421
<b>13.3 Hands-on: Kubeflow unter GKE in der Praxis</b>	422
13.3.1 Preflights	422
13.3.2 Setup	423
13.3.3 Grafische Oberflächen	428
<b>13.4 Open Data Hub</b>	430
13.4.1 Die Unterschiede zu Kubeflow – ein High-Level-Überblick	430
13.4.2 Open Data Hub (ODH) – Architektur und Arbeitsweise	430
13.4.3 Die ODH-Module	432
<b>13.5 Hands-on: Open-Data-Hub-Setup unter OpenShift</b>	433
13.5.1 Preflights	433
13.5.2 Setup	434
13.5.3 Post Rollout	439

<b>13.6 NVIDIA AI Enterprise (AI-End-to-End-relevante Betrachtungen)</b> .....	442
13.6.1 NVIDIAs AI-End-to-End-Stack – reloaded .....	443
13.6.2 Die Module im Detail .....	444
13.6.3 NVIDIAs AI-End-to-End-Patterns .....	445
<b>13.7 Hands-on: NVIDIA AI Enterprise (AI End-to-End) unter OpenShift</b> .....	447
13.7.1 NVIDIA Morpheus AI Engine .....	447
13.7.2 Triton Inference Server .....	448
13.7.3 Morpheus MLflow Triton Plugin .....	449
13.7.4 Vorbetrachtungen: AI End-to-End mit Morpheus AI Engine .....	449
13.7.5 Preflights .....	450
13.7.6 Hands-on .....	450
13.7.7 Cybersecurity mit Morpheus AI (Red Hat Developer) .....	458
13.7.8 NVIDIA Launchpad .....	458
 <b>14 The Road Ahead</b> .....	 459
 Index .....	 463

---