

Dominik Heinrich, Multi-Cloud DevOps & AI Solution Architect

Cloud-Kosten senken, ohne etwas kaputt zu machen

Die FinOps-Checkliste, die ich auf Azure-, AWS- und Kubernetes-Plattformen einsetze. Zehn Schritte, in der Reihenfolge mit dem meisten Effekt bei kleinstem Risiko.

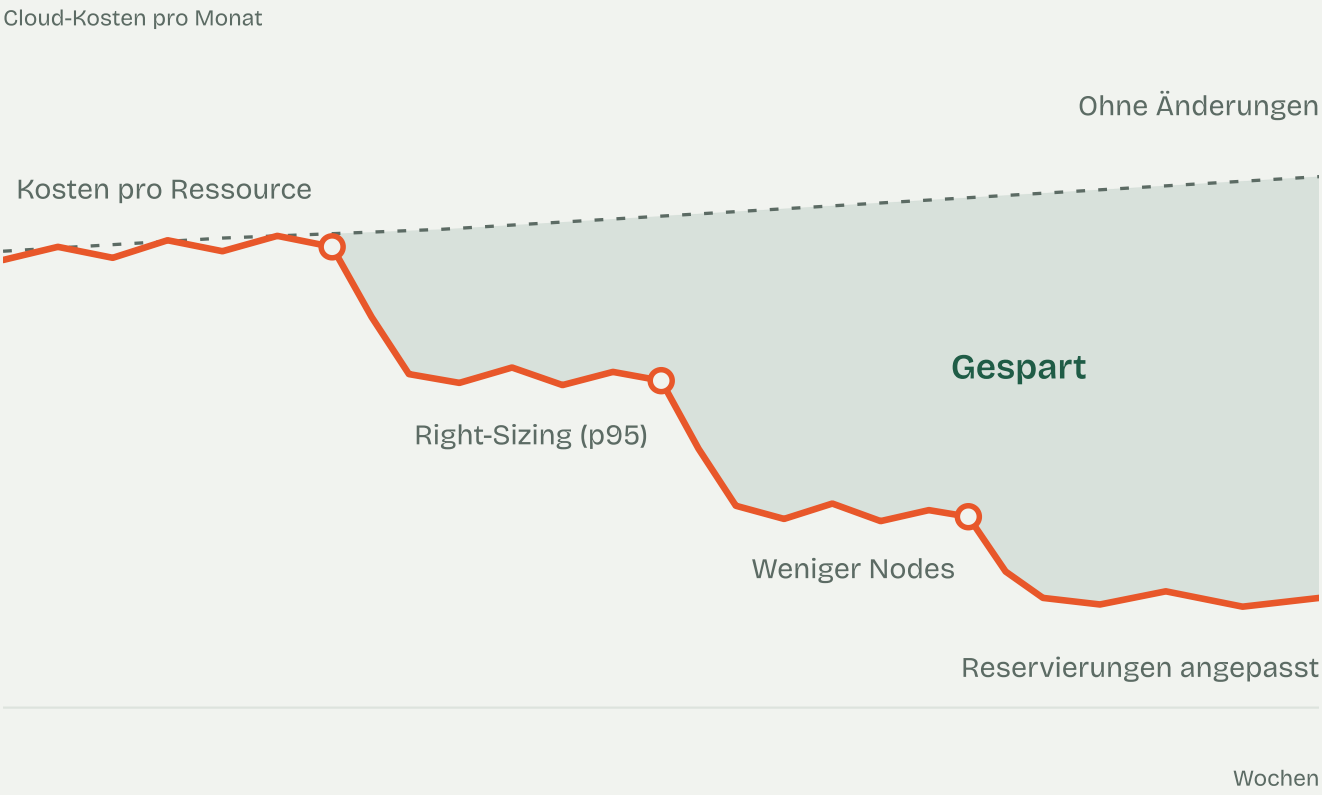


Illustration: wie eine Cloud-Rechnung in einem FinOps-Projekt typischerweise sinkt

Aus einem aktuellen FinOps-Projekt, anonymisiert

−48 %	**7 → 4**	**~40×**	**~35**
Azure-Kosten in fünf Wochen	Kubernetes-Nodes	weniger Cosmos-DB-Last	Änderungen in einem Monat, je ein Ticket

Die zehn Schritte

Der Reihe nach abarbeiten und abhaken. Jeder Schritt sagt, was vor einer Änderung zu prüfen ist.

01 Sehen, wohin das Geld geht

- [ ] Amortisierte Kosten pro Ressource und Meter aus der Billing-API ziehen, nicht nur die Monatsrechnung lesen.
- [ ] Eine Run-Rate-Prognose aufbauen, damit der Effekt einer Änderung nach Tagen sichtbar ist, nicht erst am Monatsende.
- [ ] Jede Ressource mit Produkt, Team und Stage taggen. Ungetaggte Kosten bekommen innerhalb einer Woche einen Owner.
- [ ] Einen kurzen täglichen Kostenbericht in den Team-Chat schicken. Was gesehen wird, wird behoben.

02 Prüfen, bevor man kürzt

- [ ] **Für jeden Kandidaten:** Was läuft darauf, wer nutzt es, und wie hoch ist die Spitzenlast (p95), nie der Durchschnitt.
- [ ] Alles, was ein Kunde bemerken könnte, ist tabu, bis es nachweislich sicher ist.
- [ ] Eine Änderung pro Ticket, jeweils mit Auswirkung, Risiko und Rückweg.

03 Compute richtig dimensionieren

- [ ] Kubernetes-Requests und -Limits nach echter p95-Nutzung setzen, dann folgt die Zahl der Nodes.
- [ ] Autoscaling (Cluster Autoscaler oder Karpenter) und Spot-Kapazität für unterbrechbare Jobs nutzen.
- [ ] **In einem Projekt:** ein Produktiv-Cluster von 7 auf 4 Nodes, ohne Auswirkung auf Kunden.

04 Verkleinern und löschen

- [ ] Ungenutzte VMs, verwaiste Disks und Volumes, freie IPs, alte Snapshots und vergessene Testressourcen.
- [ ] Datenbank-Standbys und Hochverfügbarkeit nur dort, wo das Geschäft sie wirklich braucht.

05 Eine Data-Retention-Policy schreiben

- [ ] **Logs:** Debug in Produktion aus, Aufbewahrung pro Log-Klasse, günstige Speicherklassen für Audit-Logs.
- [ ] **Backups:** Aufbewahrung nach Wiederherstellungszielen und gesetzlichen Pflichten, nicht nach Standardwerten.
- [ ] **Object Storage:** Lifecycle-Regeln, die alte Daten in kalte Speicherklassen verschieben und Unnötiges löschen.

06 Die teuren Queries reparieren

- [ ] Die Top-Queries nach Kosten finden, nicht nach Anzahl. Ein einziger Query-Fix senkte die Last auf einer Cosmos-DB-Datenbank um etwa das 40-Fache.
- [ ] Kleine Mandanten-Datenbanken in den Cluster holen (zum Beispiel mit CloudNativePG), wenn es sich lohnt, mit geprüften Zeilenzahlen und Prüfsummen.

07 Zuletzt binden, nicht zuerst

- [ ] Reservierungen oder Savings Plans erst nach dem Right-Sizing kaufen, damit man sich nicht an Maschinen bindet, die gleich wegfallen.
- [ ] Verpflichtungen sind schwer rückgängig zu machen: Azure Savings Plans sind nicht kündbar, AWS Savings Plans nur innerhalb von sieben Tagen rückgabefähig, und Rückerstattungen für Azure-Reservierungen sind gedeckelt. Ändert sich die Umgebung, die vorhandenen Umtauschmöglichkeiten nutzen.

08 Umgebungen abschalten

- [ ] Dev und Test standardmässig nachts und am Wochenende aus, morgens automatisch wieder aufgebaut.
- [ ] Preview-Umgebungen mit Ablaufzeit und nächtliche Aufräumläufe für verwaiste Ressourcen.

09 Das Netzwerk im Blick behalten

- [ ] Alerts auf Cross-Zone- und Egress-Traffic, und NAT-Gateways und Load Balancer ohne Owner prüfen.

10 Unten halten

- [ ] Budgets mit Prognose-Alerts pro Team und ein wöchentlicher One-Pager mit Zwei-Monats-Prognose für das Management.
- [ ] Ein benannter Owner pro Kostenstelle, und die Checkliste jedes Quartal neu durchgehen.

Die häufigsten Fehler

- Zuerst Reservierungen kaufen, dann verkleinern, und für Maschinen zahlen, die nicht mehr laufen.
- Nach Durchschnitt dimensionieren: Der Cluster wirkt leer, bis die Montagsspitze kommt.
- Löschen, bevor geklärt ist, wer es nutzt. Ein vergessener Nutzer kostet mehr als die Ersparnis.
- Logs abschalten, statt sie in günstigere Klassen zu verschieben, und damit den Audit-Trail verlieren.
- Ein grosses Aufräum-Wochenende statt eines Tickets pro Änderung, und niemand weiss, was was kaputt gemacht hat.

So arbeiten wir zusammen

Dieselbe Qualität für ein Startup mit zehn Leuten und einen globalen Konzern. Das Format passt sich Ihnen an.

Startups

Fractional DevOps- & KI-Lead

Ein paar Tage im Monat: Architektur, Release-Pipeline und KI-Funktionen, damit sich die Gründer auf Kunden konzentrieren können. Vom Prototyp in die Produktion, ohne Neubau.

KMU und Mittelstand

Ein Projekt mit klarem Ergebnis

Ein klares Ziel wie tiefere Cloud-Kosten, eine Migration oder die ersten KI-Agenten in Produktion, mit festem Umfang, wöchentlichem Fortschritt und einer Übergabe, die Ihr Team betreiben kann.

Konzerne

Ein Architekt in Ihren Teams

Eingebettet in Plattform-, Security- und Produktteams: Architekturentscheidungen, Governance für KI und regulierte Umgebungen. Für grössere Vorhaben bringe ich ein erfahrenes Senior-Team mit.

Wie ich arbeite

Entscheidungen schriftlich: Architecture Decision Records mit den Abwägungen, damit Ihr Team weiss, warum, nicht nur was.

Sicher ab Werk: OIDC und Managed Identities, Least Privilege, Secrets im Vault, ein Audit-Trail.

Betrieben, nicht nur geliefert: SLOs, Alerts auf Geschäftsergebnisse und eine Incident-Routine.

Gemessen: Kosten, Durchlaufzeit, Verfügbarkeit und gesparte Stunden, jede Woche berichtet.

Das Wissen bleibt bei Ihnen: Runbooks, Dokumentation und Workshops für Ihr Team.

Ein Ansprechpartner, der mit Engineering und Management spricht, auf Deutsch, Englisch und Spanisch.



Soll ich das auf Ihre Rechnung anwenden?

30-Minuten-Call buchen

E-mail mail@dominikheinrich.ch
Web dominikheinrich.ch

Weltweit verfügbar, remote. Vor Ort auf Anfrage, je nach Land.