

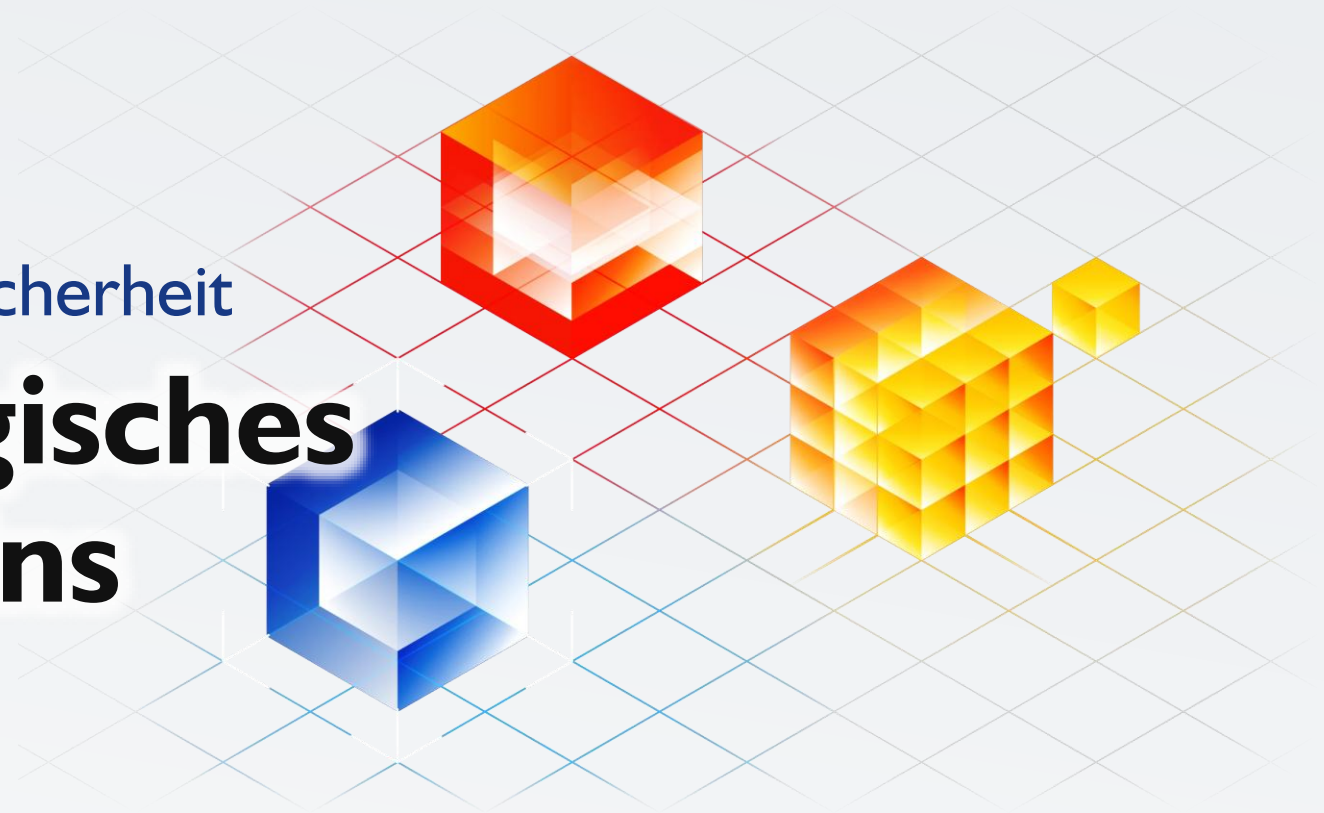
SLAC 2026

Bottlenecks, Belastbarkeit, Betriebssicherheit

Lasttests als strategisches Werkzeug für Admins

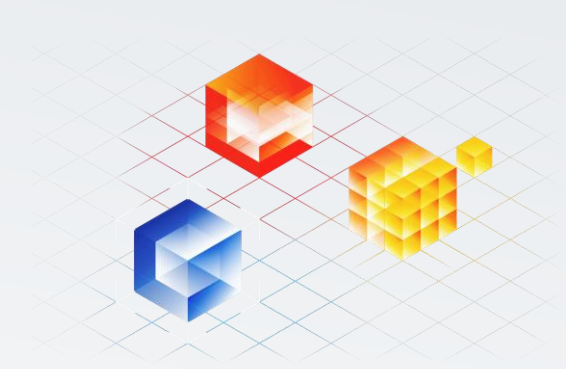
David Brückmann
Geschäftsführer MyControl GmbH

www.mycontrol.de



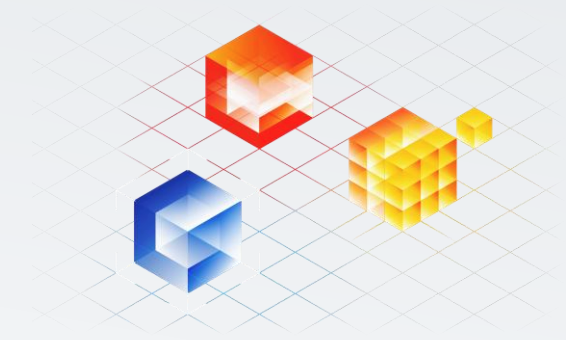
MYCONTROL

„Wir glauben, es hält“ ist heute kein Betriebsmodell mehr.

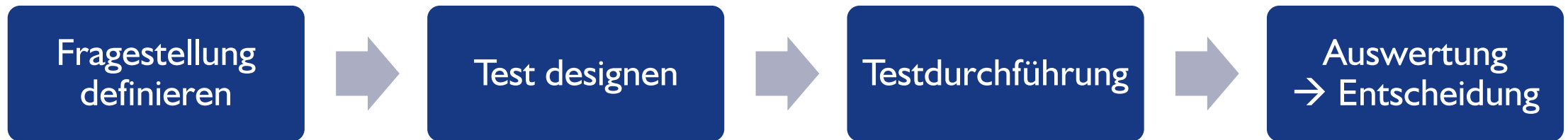


- Systeme müssen unter Last funktionieren, nicht nur im Leerlauf.
- Last ist heute Normalzustand.
- Die Komplexität aktueller Systeme ist so hoch, dass man das Verhalten unter Last nicht zuverlässig vorhersagen kann.
- Anforderungen wie NIS-2 machen Verfügbarkeit und Resilienz regulatorisch relevant.
- Proaktiv statt reaktiv: Lasttests sorgen für entspannte Admins.
- Lasttests sind die beste Grundlage, um Performanceoptimierung durchzuführen.

Lasttests scheitern nicht am Tool, sondern an fehlender Systematik.

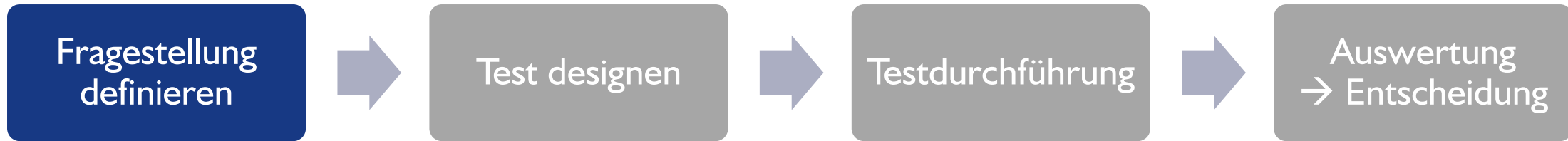
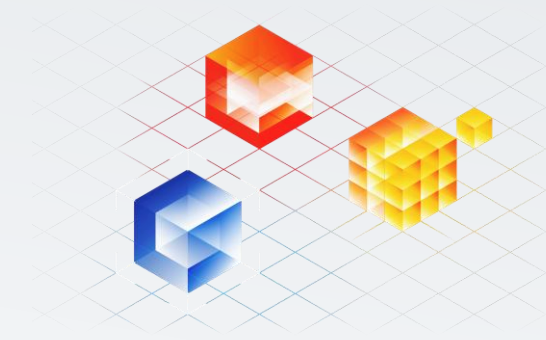


- Größter Fehler: Keine Lasttests.
- Tools gibt es viele geeignete: **Apache JMeter**, **Gatling**, **Locust**, **Grafana k6**,...
- Entscheidend ist nicht das Tool sondern das richtige Vorgehen:



- Performanceoptimierung sollte in einen Lasttest-Prozess eingebettet sein.

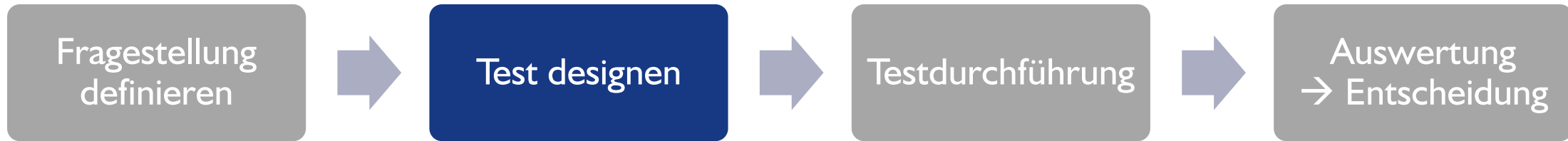
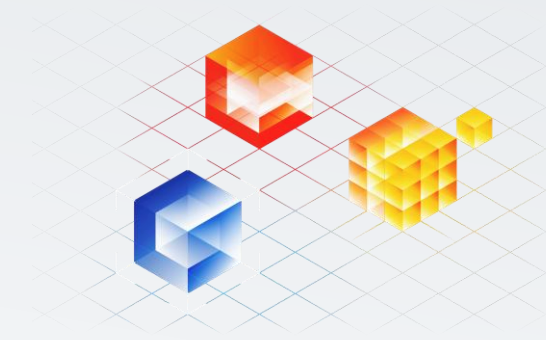
Wenn die Frage falsch ist, ist der gesamte Lasttest wertlos.



Typische Fragestellungen:

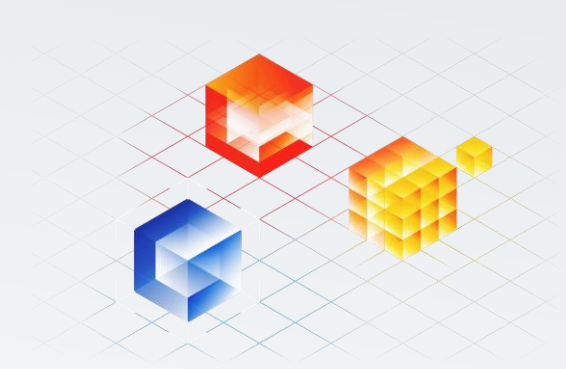
- Kapazität: „Wieviel schaffen wir?“
- Zielerfüllung: „Schaffen wir 20 Requests/sec?“ „Schaffen wir 5000 Benutzer?“
- Performance-Optimierung: „Wo liegt das Bottleneck?“
- Verhalten bei Überlast: „Was passiert, wenn es schiefeht?“
- Performance: „Wie schnell bekommen Benutzer Antworten?“

Ein unrealistischer Test liefert präzise aber irrelevante Ergebnisse.



- User-Story definieren
 - Auf Grundlage von Erfahrungen: statistisch korrekte Durchschnittsstory erarbeiten.
 - Ohne Erfahrung: Annahmen treffen
- Möglichst realistisch und breit: Alles, was nicht getestet wird, ist ein potenzielles Risiko.
- Performance-Optimierung: Kleinen, hypothesenbasierten Testplan auf Grundlage der Systemarchitektur definieren.

Häufige Fehler im Testdesign



- Keine realistischen Testdaten
 - Befüllen Sie alle Datenbanken mit Daten in realistischer Menge und Verteilung
- Caches nicht berücksichtigt
 - Verwenden Sie unterschiedliche Logins, unterschiedliche Suchbegriffe,...
- Assertions nicht zielgenau
 - Wir wollen nicht messen, wie schnell das Testobjekt Fehler produziert.
 - Beispiel für eine sinnvolle Assertion: HTTP-Statuscode 200 und „enthält Text ...“
- Akzeptable Fehlerrate und Timeouts definieren
 - Lasttests produzieren oft Fehlerartefakte. Deshalb ist eine gewisse Fehlerrate akzeptabel.
 - Am Sättigungspunkt steigt die Fehlerrate stark an, diesen Punkt wollen wir ermitteln.
- Thinking Times bei großen Lasttests verwenden.
 - Thinking Times verbrauchen Ressourcen auf dem Lasttreiber (Threads)
 - Man kann Durchsatz auch ohne Thinking-Time ermitteln.

User-Story in Apache JMeter

Ramp-Up

Realistische Testdaten

Request

Assertion auf Inhalt

Abbildung realer Verteilung

- Shop
 - View Results Tree
 - Report
 - Config: Server
 - Config: Login
 - ThreadGroup
 - Testdaten: Debitoren
 - HTTP Cookie Manager
 - TC: Startseite
 - TC: Login
 - AJAX: Login
 - TC: Startseite
 - Startseite
 - ... enthält Willkommen
 - AJAX: Bühnen
 - ... RETVAL OK
 - Miniwarenkorb
 - Loop 8x
 - Suche
 - Artikelgruppe
 - Testdaten: Artikelgruppen
 - TC: Artikelgruppe
 - TC: Artikelgruppe filtern
 - TC: Kataloggruppe
 - Loop 3x: Artikeldetails
 - 5% Meine Artikel (Favoriten & häufig bestellte Artikel)
 - TC: Meine Artikel ansehen
 - 50%: Bestellen und Logout
 - TC: Bestellassistent Initial
 - Loop: BA Seitenwechsel
 - TC: Bestellen
 - TC: Logout
 - TC: Startseite
 - Modules
 - Simple Data Writer

jp@gc - Ultimate Thread Group

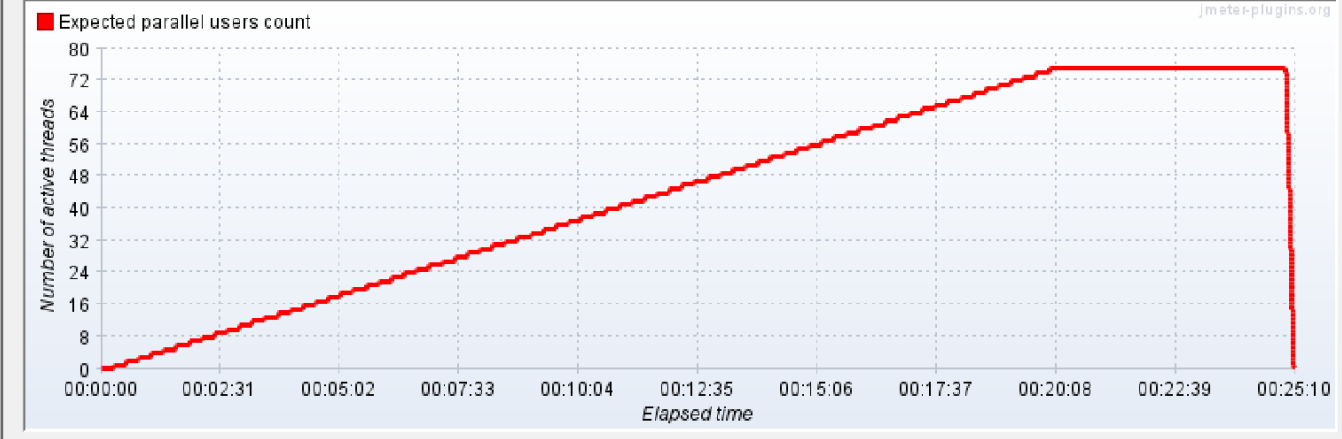
Name: ThreadGroup
 Comments:

Action to be taken after a Sampler error
 Continue Start Next Thread Loop Stop Thread Stop Test Stop Test Now

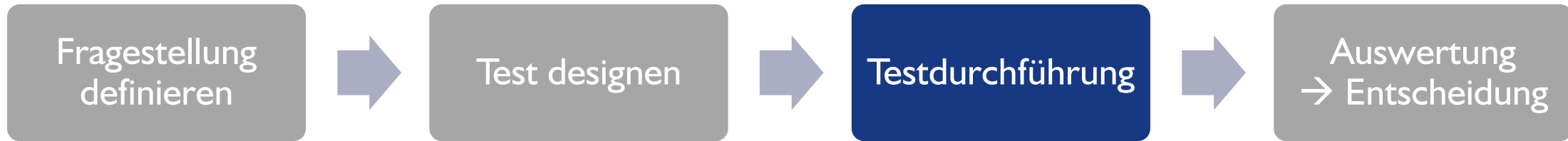
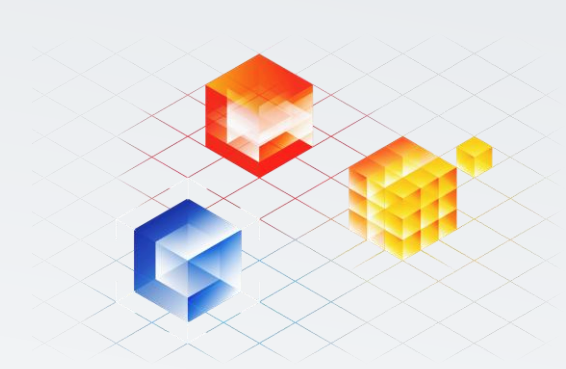
Threads Schedule

Start Threads Count	Initial Delay, sec	Startup Time, sec	Hold Load For, sec	Shutdown Time
75	0	1200	300	010

Add Row Copy Row Delete Row

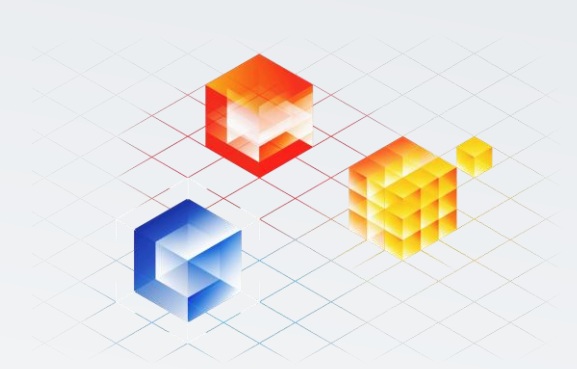


Ein guter Lasttest ist kein Skriptaufruf - sondern ein operatives Event.



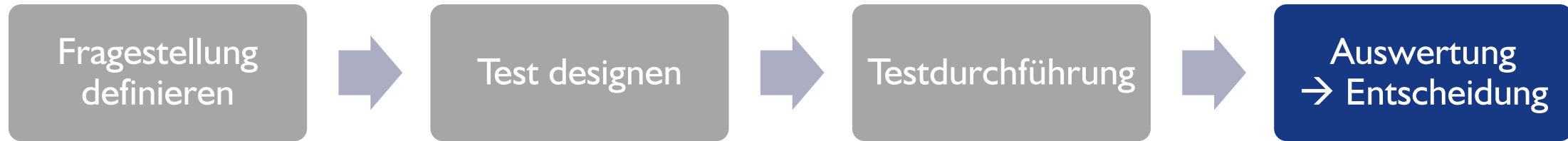
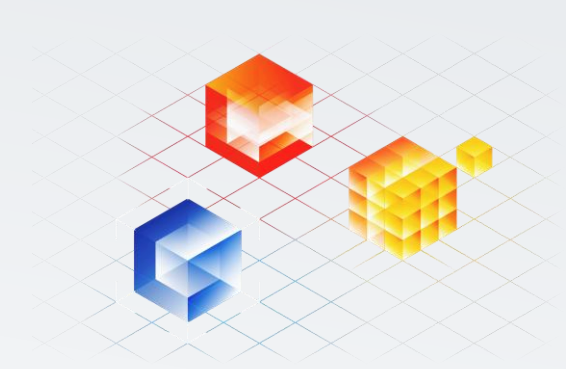
- Es läuft immer etwas schief. Vortests sind unverzichtbar.
- Jeder Lasttest sollte ein langsames Ramp-Up haben.
- Übliches Abbruchkriterium: Fehlerrate zu hoch oder Zeitablauf
- Best Practice: Administratoren überwachen die Systemmetriken in Echtzeit.
- Lasttests laufen unvorhersehbar, sie müssen schnell reagieren können. Machen Sie ein Event daraus.

Häufige Fallstricke bei der Testdurchführung



- Testtreiber als Bottleneck
 - Netzwerk, Load & RAM der Testtreiber überwachen
- Firewall / WAF blockiert Traffic
 - Testtreiber auf Whitelist setzen
- Testumgebung ist nicht mit Produktivumgebung vergleichbar.
 - Komplexe Systeme skalieren in der Regel nicht linear.
- Systemlast wird nicht live überwacht und Ramp-Up zu schnell
 - Sie verschenken die Chance Ihre Bottlenecks zu finden.

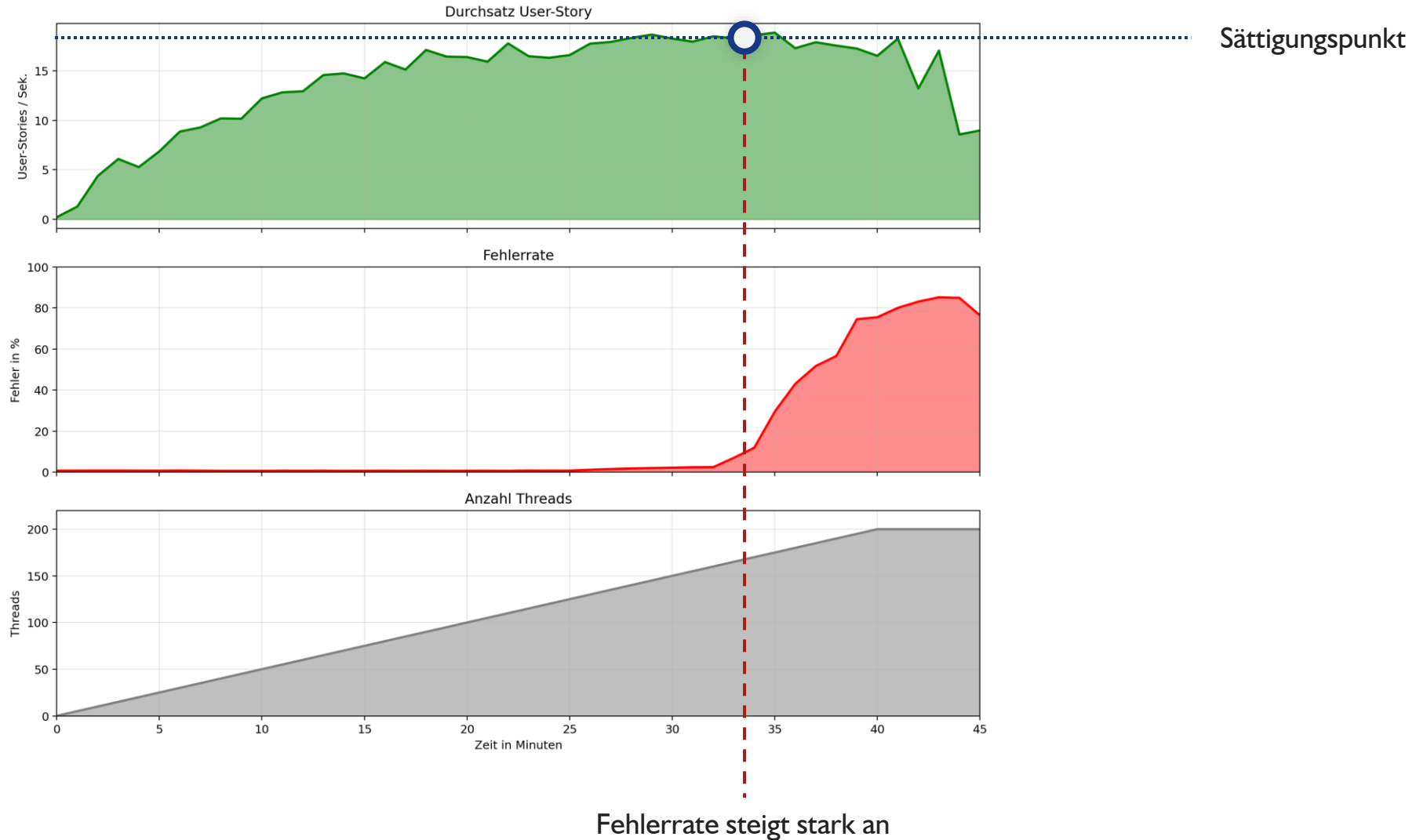
Wir brauchen keine schönen Diagramme, wir suchen die entscheidenden Signale.



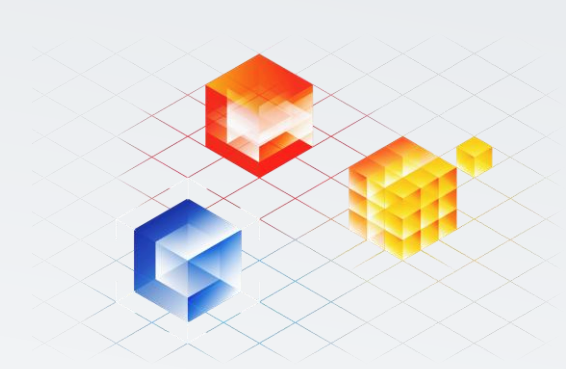
Die wichtigsten Metriken:

- Durchsatz User-Story → Sättigungspunkt.
- Fehlerrate: Üblicherweise primär ein Testendekriterium.
- Antwortzeiten: Arbeiten Sie mit Perzentilen, nicht mit average/min/max.
- Durchsatz und Latency im Test sind voneinander abhängig.

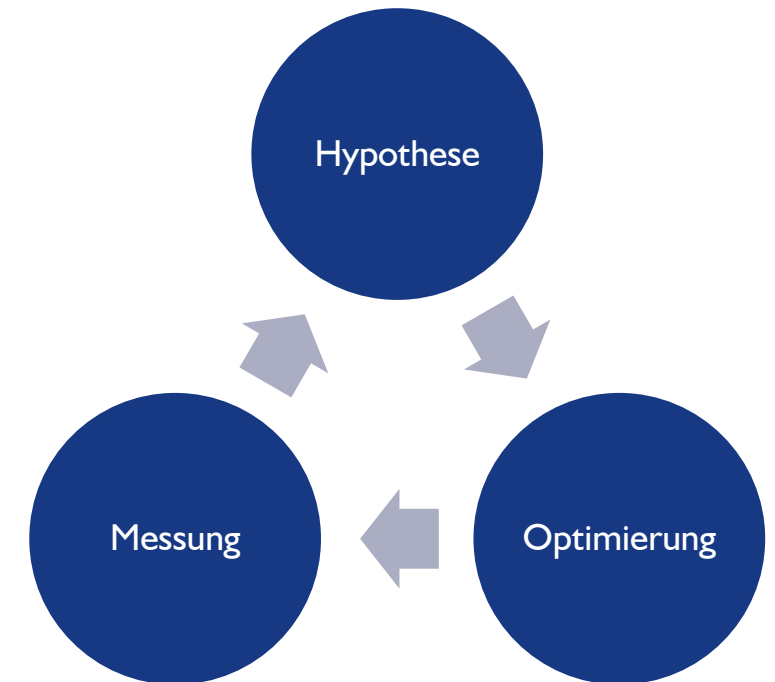
Sättigungspunkt: Die wichtigste Metrik



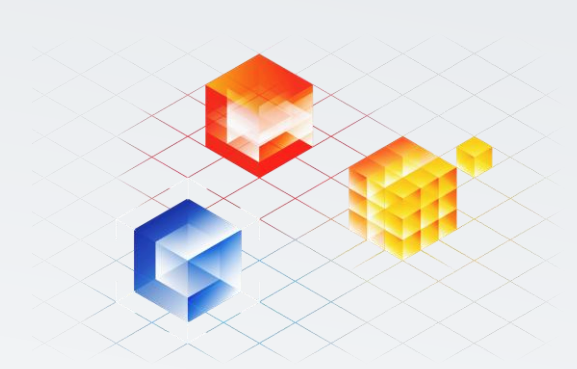
Optimieren ohne zu messen bedeutet, dass Sie raten.



- Erst den Lasttest, dann die Performanceoptimierung. Sonst optimieren Sie an der falschen Stelle.
- Performance-Optimierung basiert auf einer klaren Hypothese.
- Entwicklung eines kleinen angepassten Lasttests der nur das Bottleneck adressiert.
- Danach folgt der Optimierungszyklus: Hypothese → Optimierung → Messung
- Am Ende: Gesamten Lasttest wiederholen.



Lasttests sind kein Projekt. Sie sind Teil des Betriebs.



- Lasttests sollten Teil der operativen Routine sein.
- Lasttests sind am Anfang aufwändiger, später umso wertvoller.
- Lasttests sind eine Grundlage für einen entspannten Betrieb.
- Methodisch sauber arbeiten:
 - Fragestellung definieren
 - Testdesign realistisch designen
 - Durchführung gut vorbereiten
 - Auswertung so durchführen, dass Entscheidungen möglich sind

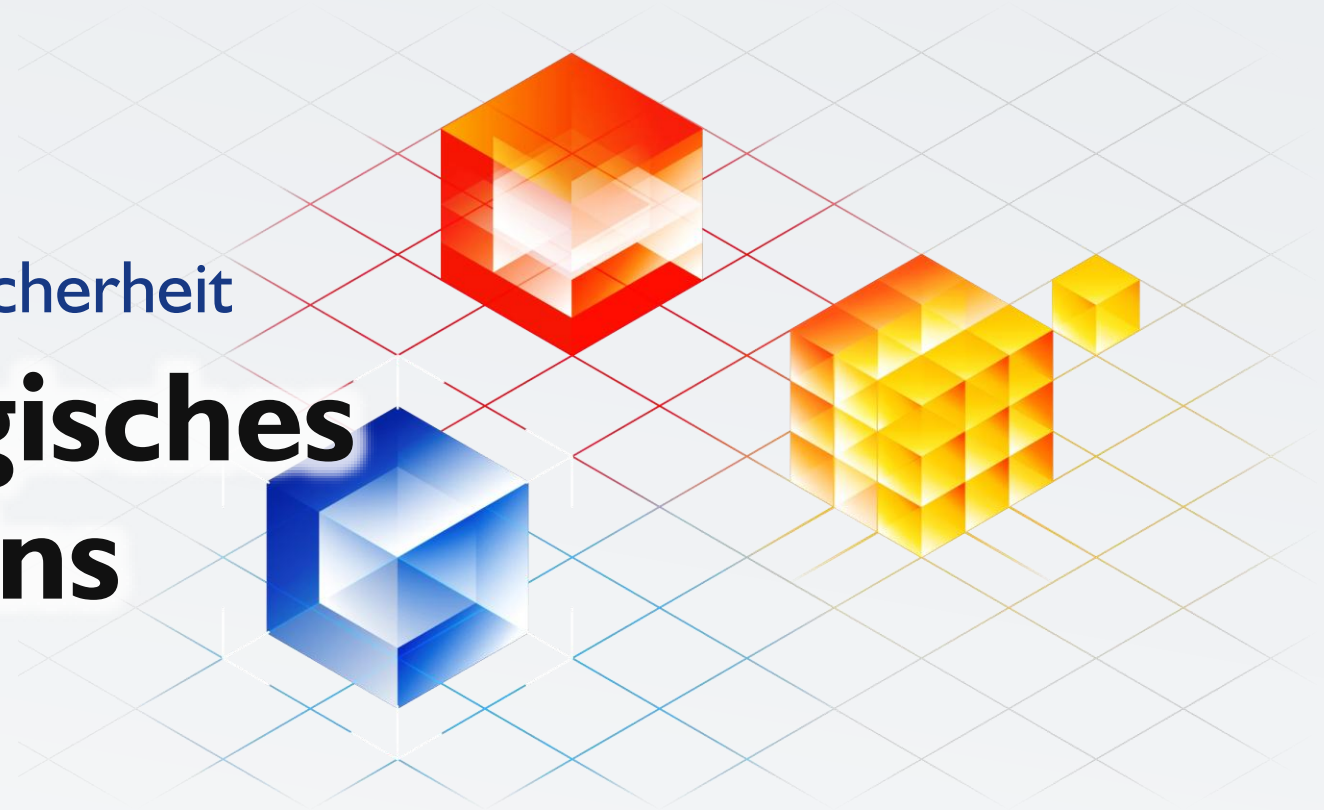
SLAC 2026

Bottlenecks, Belastbarkeit, Betriebssicherheit

Lasttests als strategisches Werkzeug für Admins

David Brückmann
Geschäftsführer MyControl GmbH

www.mycontrol.de



MYCONTROL