Tests Theory

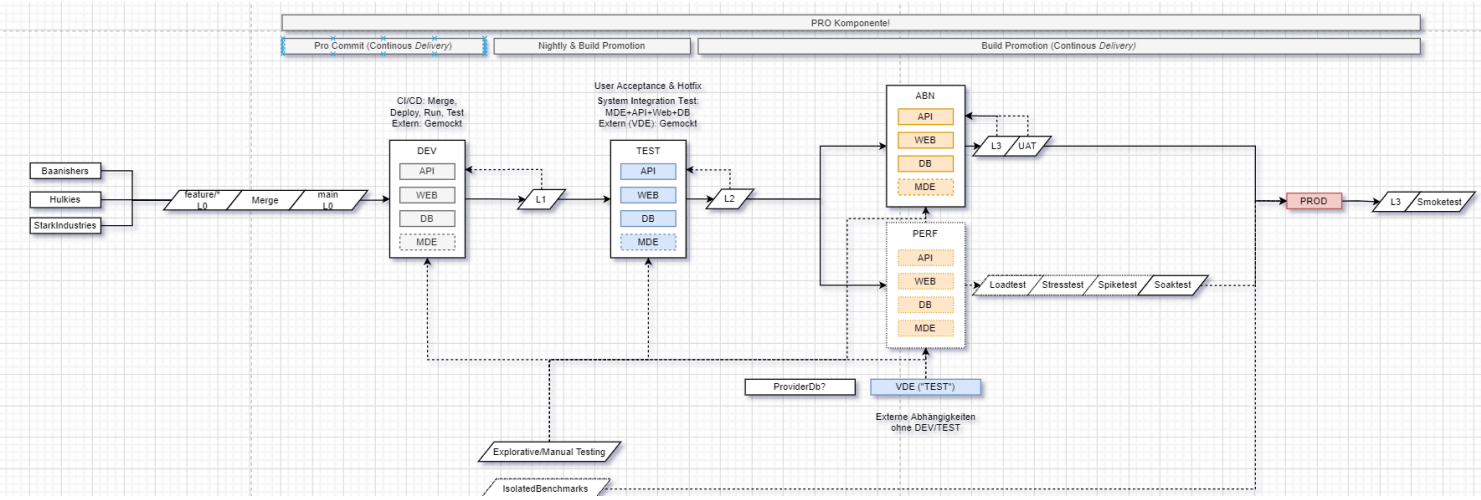
Last updated by | SITKOVICH Stefan | Jan 23, 2025 at 3:01 PM GMT+1

Contents

Tests

Tests sind wie folgt einzuteilen (lose basierend auf MS DevOps sowie k6.io Testtypen)

Zielvision



Das Ziel ist es alle Arten von Tests, wo nur möglich automatisiert, entsprechend ihrem Aufwand und Wert in den Entwicklungszyklus zu integrieren um Fehler möglichst früh zu finden und mit hohem Vertrauen produktiv deployen zu können. Durch kurze Feedbackschleifen soll es möglich Software schnell und fehlerfrei deployen zu können.

Level 0 "UnitTests":

- Schnelle, in-memory Tests
- Benötigt ausschließlich die Assemblies und hat keine anderen Abhängigkeiten
- Durchschnittliche Zeit pro Test <=60ms
- Maximale Zeit pro Test <=2s
- Maximale Zeit für alle Tests <=5 Min
- Automatische Ausführung im Zuge der Build Pipeline & Pull Request Validation
- Müssen zu 100% grün sein um Code zu mergen und damit automatisch nach DEV zu deployen

Level 1 "IntegrationTests":

- Schnelle Tests mit Abhängigkeiten zu DB oder Filesystem aber nicht zu anderen Services
- Benötigen potentiell eine Datenbank oder das Filesystem
- Durchschnittliche Zeit pro Test <= 400 ms
- Maximale Zeit pro Test <=2s
- Durchschnittliche Zeit für alle Tests <= 5 Min
- Automatische Ausführung nach einem DEV Deployment
- Müssen zu 100% grün sein für ein TEST Deployment

Level 2 "FunctionalTests"

- Funktionale Tests die ein Service Deployment erfordern, bei dem aber einige Services potentiell gemocked sind.
- Benötigen potentiell eine Datenbank oder das Filesystem
- Durchschnittliche Zeit pro Test <= 2s ms
- Maximale Zeit pro Test <= 1 Min
- Durchschnittliche Zeit für alle Tests <= 5 Min
- Automatische Ausführung nach einem TEST Deployment
- Müssen zu 100% grün sein für ein ABN Deployment

Level 3 "EndToEndTests"

- Ausgewählte Tests die gegen Produktion ausgeführt werden
- Benötigen ein vollständiges Deployment
- Inkludieren Smoketests um Funktion des Systems bei minimaler Last zu zeigen
- Durchschnittliche Zeit pro Test <= 2s ms
- Maximale Zeit pro Test <= 1 Min
- Durchschnittliche Zeit für alle Tests <= 5 Min
- Automatische Ausführung nach einem ABN oder PROD Deployment
- Müssen zu 100% grün sein für ein PROD Deployment oder einen Rollback verursachen

Isolierte Benchmarks "IsolatedBenchmark"

- Benchmarks die nach jedem Build eines Moduls ausführt werden um Performanceveränderungen zu messen
- Besitzen keine externen Abhängigkeiten
- Automatische Durchführung nach jedem Build
- Klassifizierung der Resultate: Besitzen keinen allgemeinen "grün" Status, sondern sind Performance Indikatoren um Bottlenecks die optimiert werden müssen zu identifizieren.

Lasttests

- Lasttests die nach jedem Build ausgeführt werden um SLI zu erfassen
- Benötigen ein Service Deployment für alle betroffenen Services, nicht getestete Services dürfen aber gemocked werden.
- Das Ziel Environment muss "produktionsnahe" sein um Aussagen über die Werte treffen zu können (UAT, PERF)
- Die Last soll 100% der erwarteten täglichen Last entsprechen.
- Schwellwerte basierend auf dem 99% Percentil sind einzuhalten
- Lasttests sollen automatisiert jeden Tag nach der Regelarbeit gestartet werden um täglicher Regression entgegen zu wirken

Stresstests (~unsere aktuellen Lasttests)

- Lasttests die nach jedem Build ausgeführt werden und das System unter Extrem Situationen testen (maximale Last im gesamten Jahr, z.B. Weihnachten)
- Das Ziel Environment muss "produktionsnahe" sein um Aussagen über die Werte treffen zu können (UAT, PERF)
- Schwellwerte basierend auf dem 99% Percentil sind einzuhalten (aber eine deutlich höhere Fehlerrate als bei einem Lasttest wird akzeptiert)

Spiketests

- Kurze Last mit einem vielfachen der Maximallast (z.B. x2-x10)
- Spiketests sollen vor jedem Lasttests durchgeführt werden (fail-fast).
- Das Ziel Environment muss "produktionsnahe" sein um Aussagen über die Werte treffen zu können (UAT, PERF)
- Klassifizierung des Testresultates:
  - Exzellent: 99% Percentil Schwellwerte eines Lasttest werden auch bei einem vielfachen der Last eingehalten.
  - Gut: Antwortzeiten liegen über den 99% Percentil, aber die Fehlerrate steigt nicht an und alle Requests werden bearbeitet.
  - Schlecht: Antwortzeiten und Fehlerrate liegen höher, aber das System erholt sich zu normalen Werten in der Cooldown Phase.
  - Kritisch: Das System stürzt ab und/oder erholt sich nicht mehr davon obwohl der Spike vorbei ist.

Soaktest (~unsere aktuellen Lasttests)

- Mehrstündiger Test mit 80% der Kapazität um die Resillienz des Systems zu messen

- Aufzeigen von zeitabhängigen Degradierungen und Fehlern
- Kosten der Soaktests sind vor Durchführung einzuplanen

Explorative / Manuelle Tests

- Werden von den Entwicklern (incl. Tester*Innen) im Zuge der Entwicklung auf den Entwicklungsstages und UAT Stage durchgeführt (`DEV` , `TEST` , `ABN`)
- Entsprechen primär einer kurzen stichprobenartigen Überprüfung ob ihr lokale System mit der Änderung wie erwartet funktioniert

Akzeptanztests

- Werden von Usern im Zuge der User Acceptance Tests in der UAT-Stage `ABN` durchgeführt.
- Entscheiden in erster Linie darüber ob eine US korrekt umgesetzt wurde und in den Status "Accepted" wechseln darf.

Aktuelle Lasttests in IRIS:

- Sind eine Mischung aus Soak (Stundenlange Tests mit hoher Last) und Stresstests (Last entspricht dem Jahreshöchstand, auf schwächerer Hardware). Sie eignen sich aber nicht als Soaktests, da die Last mit >=100% der Last zu hoch ist um ein echtes Perfomance Profil abzubilden. Sie eignen sich auch nicht als Stresstests, da die Dauer mit >=8h deutlich zu lange um schnell genug ein Feedback zu erhalten.
- Sind aktuell zu volatil um überhaupt Aussagen zu treffen. Oft kann von 5 Tests nur einer gewertet werden (manuelle Fehler, Einfrieren der Lasttests Agents aufgrund deutlich zu hoher Last am Testclient, Messwerte durch einfrieren von Clients teilweise derart verfälscht, ...)