**Test Engagement - Performance Testing**

**Performance Tech Stack.**

Apache JMeter: java

Gatling: **Java, Kotlin and Scala**

LoadRunner: java, c, JavaScript, and .net

k6: javascript

NeoLoad: JavaScript

BlazeMeter: jmx,json,yml,

Locust: Python

Artillery: Go, Python, Zig, C++

**Integration with other components**

A blue arrow pointing to a logo

Description automatically generated

**High level usage of components**

ADO – is the Test management tool where test cases are available and linked to the JMeter.

JMeter: is integrated with ADO to link the Test scenarios and all the performance Scripts.

GitHub is the CI/CD tool to run the scripts whenever there is a change in the repo, scheduler triggers the scripts in the JMeter to run, and reports back the results to the ADO.

**When to use?**

1. Load Testing
   * **What:** Is to apply increasing amounts of load on the system and reach a point where degradation starts.
   * **Usage:** 
     + To baseline the performance parameters like response time, throughput, of the system under normal production working conditions.
     + To identify areas of performance bottlenecks
     + To proactively engage a team when there is performance degradation build over build.
2. Stress Testing
   * **What:** Is to keep increasing the load to break the system.
   * **Usage:** 
     + Is to measure the system’s capability to recover after a failure.
     + Is to document the negative impact and risks to system (data corruption, unable to startup etc.) post a system crash.
     + Is to document the steps to get the system online post system crash.
3. Spike Testing
   * **What:** Is to subject the system to sudden bursts of peak loads which a production system might encounter under special conditions like a black Friday sale opening
   * **Usage:** 
     + The goal is to see if all aspects of the system, including server and database, can handle sudden bursts in demand.
4. Endurance Testing
   * **What:** Is to keep the system under test on predetermined sustained loads for extended periods of test
   * **Usage:** 
     + The goal is to check for performance degradation due to memory leaks, log file fills up, server errors, GC health etc.
5. Scalability Testing
   * **What:** Is to increase or decrease certain parameters of the system E.g. Increasing the thread pool count in DB connection, Decreasing the number of server cluster, increasing cluster server and run load tests
   * **Usage:** 
     + The goal is to check if the system scales and continues optimal performance when resources are added or removed.
6. DB Volume Testing
   * **What:** Is to increase the size of data in DB and run load tests
   * **Usage:** 
     + The goal is to check the performance of the system and end user experience as the back-end systems grow heavy.
     + To check if DB CRUD operations take more time as DB size increases.

**High Level Integration:**

**A diagram of a software process

Description automatically generated**

**Cost: Yes/No**

Postman: Yes (when you want to integrate with CI/CD)

Swagger: No

Apache JMeter: No

Gatling: No

LoadRunner: No

k6: No

NeoLoad: Yes(cost need to get in touch with sales team from neo load)

BlazeMeter: Yes(cost need to get in touch with sales team)

Locust: No

Artillery: No

**Ease Of Integration: Yes/No**

Postman: Yes

Swagger: Yes

Apache JMeter: Yes

Gatling: Yes

LoadRunner: Yes

k6: Yes

NeoLoad: Yes

BlazeMeter: Yes

Locust: Yes

Artillery: Yes

**Open source: - Yes/No**

Postman: No - Partial (when you want to integrate with CI/CD)

Swagger: No

Apache JMeter: Yes

Gatling: Yes

LoadRunner: Yes

k6: Yes

NeoLoad: No

BlazeMeter: No

Locust: Yes

Artillery: Yes