Load-Testing Containerized Microservices

Develop and load-test using JMeter a simple containerized application-server, and write a comprehensive report discussing results of the stress testing.

REQ-1	GET /factorial/<number>	Calculates the factorial value of the given number and returns the value .If the number is negative, reject it and return a message to use a smaller number; Otherwise store the successful calculation output in the connected Redis object-storage service
REQ-2	GET /calculationsStored	Returns a list with all the calculated values stored in the connected Redis service

Using Apache JMeter conduct a number of stress tests on the application. Run JMeter from the same machine, firing requests at localhost:500001

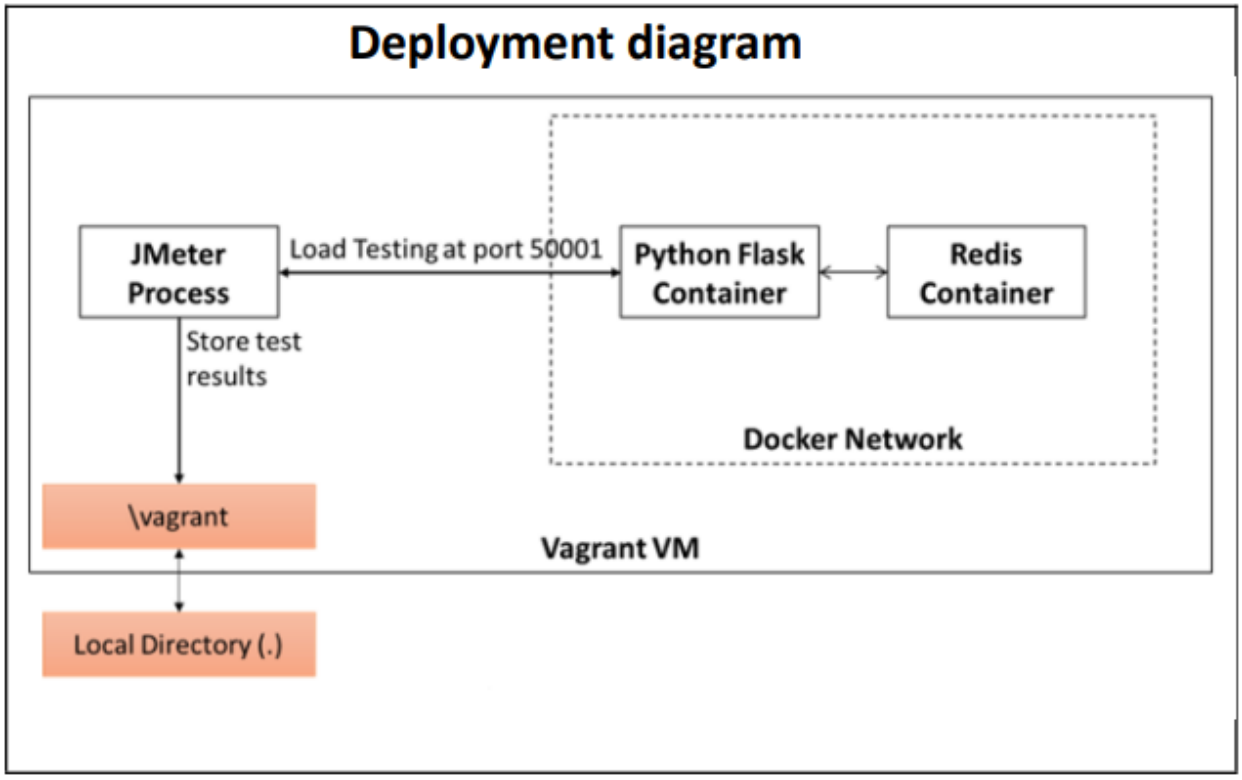
a. Write two scenarios that use 50 threads:
   i. Scenario 1 repeatedly calculates the large factorial 2022 by invoking the app's /factorial URI

   ii. Scenario 2 first invokes the factorial API for all integer numbers between 10,000 and 11,000 inclusive; then, it repeatedly invokes the calculationsStored URI of the app

   iii. The repeating part of both scenarios should last 60s

b. Run a number of experiments and collect Response-Time and Throughput data
   i. Try at three different CPU limits for the web service
   ii. Try at three different time delays in JMeter
   iii. Repeat each experiment 3 times
   iv. Thus, this means 2 Scenarios times 3 CPU limits times 3 timer delays times 3 iterations = 54
   v. Ensure that the response code of the requests is 200 OK.

c. Can automate via scripts


Deployment diagram
JMeter Process
Load Testing at port 50001
Python Flask Container
Redis Container
Store test results
Docker Network
\vagrant
Vagrant VM
Local Directory (.)