TU Dublin, Tallaght Campus, Department of Computing

MSc in DevOps

Enterprise Architecture Design CA1

Value: **50%** of final mark

Due: **04/05/2020**

Submission: Git repository link, pasted in @ **CA1** on Moodle

Viva: **05/05/2020** and **06/05/2020** (each student will be given a slot for

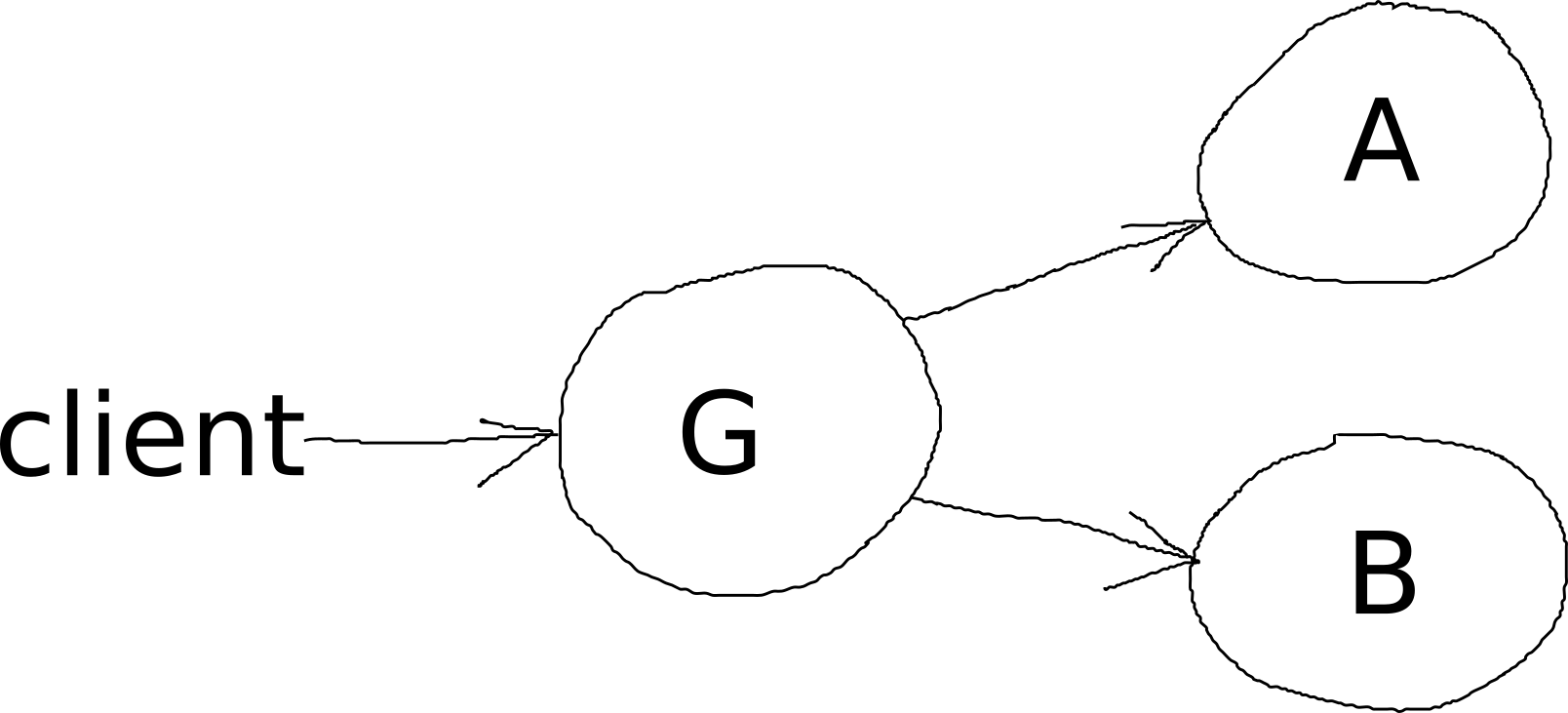
a 15-minute demo with Q&A)

► Sources must be properly referenced

► TU Dublin plagiarism rules apply

**Assignment**

***Part 1 [40%]***

Build and deploy a microservices application consisting of three services, A, B and G that interact like those shown in the picture, where arrows are directed from client to server for each line of communication. The development and system requirements are as follows:

* you may implement the services yourself or use service implementations from other sources, including EADes labs, properly referencing any external sources
* design and implement two versions of the system:
  + **sync**, where communication G - B and G - A is synchronous
  + **async**, where communication G - B and G - A is asynchronous
* the two versions should provide the exact same service from the client’s point of view
* the **async** versions of the services should be configurable with respect to the frequency of raised events (A and B) and frequency of polling (G)
* the system should be deployed on Kubernetes, either locally with **minikube** or on the **Google Kubernetes Engine (GKE)**

***Part 2 [60%]***

Design and implement a way of measuring the average response time of the system, then

1. measure and compare the values obtained for the **sync** and **async** versions, presenting the results in a barchart
2. plot the **async** version response as a function of the frequency of events emanating from services A and B and as a function of polling frequency in G
3. plot a barchart of average recovery times for 6 cases: service G, A or B failing in the **sync** version and service G, A or B failing in the **async** version

In carrying out this task, you are required to

* run the system that you are testing on Google Kubernetes Engine
* derive the response time as the average of that for a large number of individual requests
* devise and apply a way of ‘killing’ the services in order to carry out task 3 from the list above
* define and call some Google Cloud Function instances to create the graphs, defining the functions in any way you choose
* take one of the following two options for running the tests:
  + automate the process of test measurement, result reporting and graph creation with the use of a script, where the produced report and graphs can be in any form (e.g. HTML or command line text for the report and HTML-embedded image or image saved on disk for the graph)
  + use tools such as Graphana for measurements, automate only parts of the process and fill in the workflow gaps manually; write a concise report