FIT3140 Advanced Programming

Assignment 3
Spiking and Analysis of Alternatives

Group assignment - worth 10% of your final mark

Due Sunday, 22 April 2018 @ 11:55 PM

Objectives

- Design and Implement Serverless applications
- Microservices Architecture
- Cloud vs on-premise apps
- Continuous delivery (CD)

In this assignment, you'll be going through project inception: doing some initial investigation into what will be needed and deciding on an approach that's right for your team.

The scenario:

You are a member of a development team and you have been asked to develop an application that requires data from IoT sensors. You are not sure whether to develop an on-premise server or cloud based serverless application.

Your task is to investigate both techniques and decide which one is better for your application.

To investigate, you need to conduct spikes (at least two), where each one represents one technique.

What is spike?

A spike is a tiny program – the smallest amount of code that demonstrates the feature or approach you're looking at. You can think of it as "throwaway" code, because it's usually not integrated into your final project, but it shouldn't be thrown away completely. That's because, when it's finished, it is a working example of your new toolkit that you can use for reference. The purpose of a spike is to gain the knowledge necessary to reduce the risk of a technical approach, better understand a requirement, or increase the reliability of a Story estimate [1,2].

What is Serverless application? [3]

A serverless architecture is a way to build and run applications and services without having to manage infrastructure. Serverless computing allows you to build and run applications and services without thinking about servers. Serverless applications don't require you to provision, scale, and manage any servers [5].

- Serverless doesn't mean running code without servers.
- Traditionally the architecture looks something like this Business logic is implemented on the server side connected to a Database, with an HTML and JavaScript component as the client.
- With a Serverless architecture this will look more like this a rich client with business logic uses API Gateway with Authorization and Payment services, external Analytics and Search engine, with Cloud storage and Database. All components could be independent third-party services.

Major Serverless Computing Vendors

1. Google Cloud
2. Microsoft Azure
3. Amazon AWS
4. IBM Cloud Function

System Components

The system consists of the following components:

1. A node.js application that reads data from the motion sensor and pushes it to Firebase DB. The data is a JSON object that has three attributes (Same App for all spikes):
   - 1.1. Timestamp
   - 1.2. Motion start time
   - 1.3. Motion end time

   Ref: Week 3 Lab sheet & Assignment 2

2. Another application that:
   - 2.1. Listens to the Firebase DB
   - 2.2. For each new object, you have to perform two functions:
     - 2.2.1. Function 1: Sends email to a pre-defined address if the length of the motion l is in range $t1<l<t2$, where $t1$ and $t2$ are pre-defined thresholds.
     - 2.2.2. Function 2: Sends another email to a pre-defined address shows the summary of the last new five motions.
     - 2.2.3. Function 3: Clears the database if a motion occurs with length $l>t2$.

Note: No client side is required (i.e. no HTML for clients)

Task 1. Spikes (10 + 10 = 20 marks)

Spike 1: In this spike, you will implement both components as on-premise applications. This means, both components (apps) will run on your server (i.e. your machine).

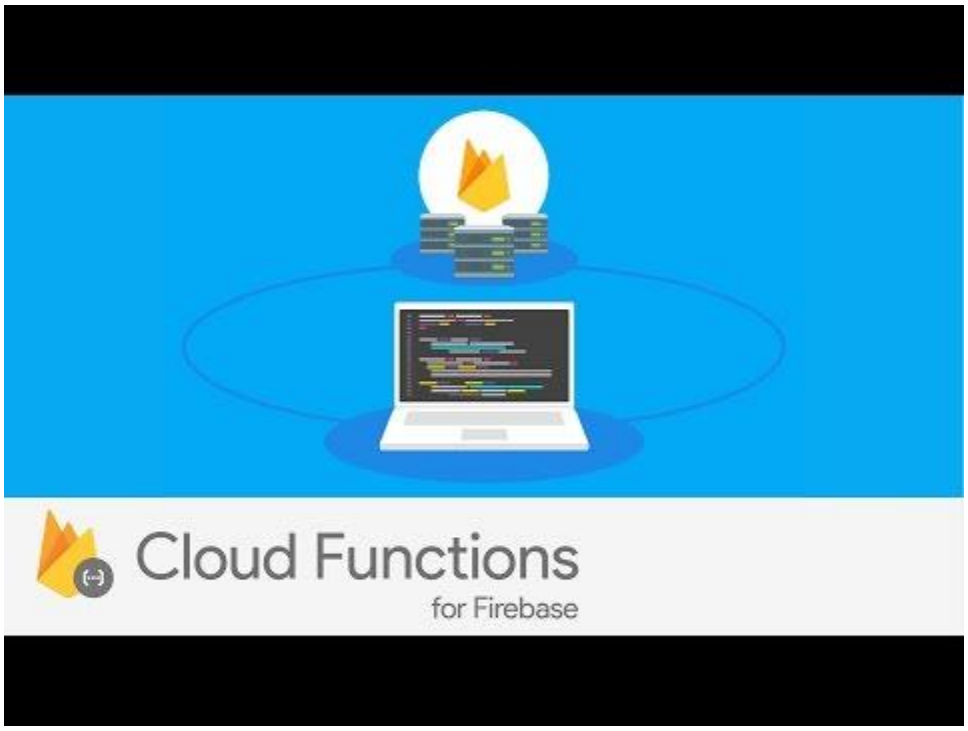
Spike 2: In this spike, you will implement serverless application using Firebase functions. You have to do the following:

- First component is similar to Spike 1, which is responsible for pushing IoT data to Firebase.
- Develop three Firebase functions represents points 2.2.1, 2.2.2, and 2.2.3 and deploy them to Firebase.

Ref: Week 6 Lab Sheet

Firebase Functions:

Cloud Functions for Firebase lets you automatically run backend code in response to events triggered by Firebase features and HTTPS requests. Your code is stored in Google's cloud and runs in a managed environment. There's no need to manage and scale your own servers [4].



NOTE: Examples on how to build Firebase functions will be provided in Week 6 lab sheet

Before you can decide on a toolchain, you'll need to investigate the options available to you. There is a certain amount of technical risk associated with this decision: you need to select an option that you either know or can learn to use quickly, is not too hard to install and maintain, and can support all the required functionality.

Note: Spike code is not usually integrated into the final version of a project, so your code does not have to be fully robust or systematically documented in the same way production code would. You should comment it, though – you and your partner will be referring to it when you implement similar features in your project, so good inline comments on the mechanics of the code will save you a lot of time and effort later on.

You need to plan your spikes. You can use the template provided at the end of this document (on Moodle as well). Normally, you would also write an outcome report, but in this case the outcome reporting will be done in the analysis of alternatives.

<u>Task 2. Analysis of alternatives (10 marks)</u>

Once you've done some hands-on exploration, you need to write the whole thing up systematically in a report. Your report should cover:

- technical and nontechnical specifications of each spike
- for each objective mentioned above,
    - provide some background that might include figures
    - how did you practice it in each spike? (If applicable)
- Compare the spikes in terms of:
    - Scalability
    - Hosting
    - Ease of database access
    - Team work
- for each spike, how well it meets each of your criteria - including your evidence from conducting the spikes.
- which option you recommend

This is the kind of report you would usually write when your coworkers or managers have asked for your input on a technical decision. As such, you are writing for a technically-literate audience that values clear and concise writing. You may refer to external documents, websites, etc. in your report, provided you include them in your bibliography.

The exact criteria you choose to examine are up to you, but they should encompass things that you will need to efficiently develop and potentially support and maintain the app. This can include your existing expertise with specific technologies, but that is not a sufficient basis to make a decision.

The length of the report will depend on how many alternatives you consider and which criteria you choose to address.

What to submit

Your group should submit:

- spike source code
- spike plan
- an analysis of alternatives detailing the toolchains you've investigated and clearly stating your preferred option

All required files are to be archived in a ZIP (not RAR, TGZ, or any other archiving format) and submitted to Moodle on by the due date. Don't forget to submit your spike documentation and analysis of alternatives in PDF format.

Marking criteria

When marking your submission, your demonstrator will take into account:

- the quality of your writing
- the clarity of your analysis report
- the thoroughness of your analysis and the feasibility of your recommendation
- the readability and presentation quality of your documents
- the functionality and readability of your spike code

Marking Scheme

- 20% Interview
- 10% GitHub activities (minimum 5 non-trivial commits)
- 40% for the spikes
- 30% reports

Penalties

Late submissions attract a late penalty of 10% per working day, unless an extension has been arranged beforehand with the lecturer.

References

[1]. http://www.scaledagileframework.com/spikes/

[2]. http://agiledictionary.com/209/spike/

[3]. https://www.youtube.com/watch?v=5AIS_y6CJ6E

[4]. https://firebase.google.com/docs/functions/

[5]. https://aws.amazon.com/lambda/serverless-architectures-learn-more/