UNIVERSITY of LIMERICK

OLLSCOIL LUIMNIGH

Autumn Interim Report

Project: Docker containers deployed using Bluemix

Name: Thomas Flynn

ID: 16117743

Supervisor: Dr. Sean McGrath

Course: Information & Network Security MEng

Year: 2016

Department: Electronic & Computer Engineering

Table of Contents

Table of Figures

1 Introduction and report outline

1.1 Project description

This project aims to provide a bridge between the university's wireless sensor network (WSN) mobile application and the data mining opportunities of the sensor information collected. This will comprise of a highly scalable database and a simple web application developed using Docker containers deployed on the cloud.

A Docker container consists of an entire runtime environment: an application, plus all its dependencies, libraries, and configuration files needed to run it, bundled into one package. By containerizing the application platform and its dependencies, differences in OS distributions and underlying infrastructure are abstracted away. Docker containers are highly scalable and will be used to handle the load balancing aspects of the system.

The Global Position System coordinates transmitted from the simulated WSN will be stored in a database for data mining purposes. The database will need to be a non relational database, allowing it to scale more efficiently.

The mobile sensor IoT application at this point is still in early development by another student. The application will track the Global Positioning System coordinates of the user when they have stepped onto the university's campus. This will help provide for potential data mining capabilities when combined with other data.

The web application will comprise of a simple user interface that will provide the ability to query the sensor data stored in the scalable database. The data will be used to display the current status of the sensors to the user . Graph databases will allow for additional database filtering features to be implemented on the website in the later stages of the development life cycle.

The primary focus of this project will reside in the area of container orchestration. This defines not only the initial deployment of the containers, but also the management of the multi-container as a single entity, such as availability, scaling and networking of containers.

In summary this project aims to provide a bridge between WSN data and data mining. The learning outcomes of this project will provide me with a foundational knowledge in the areas of DevOps, front-end/back-end web development, and container orchestration in relation to designing highly scalable and secure cloud applications.

1.2 Project aims and objectives

Primary objectives

- Implement continuous integration and continuous delivery pipeline
- Create an instance that will transmit pseudo sensor data
- Create an instance that will read sensor data
- Develop a database that can store the pseudo data in an organized manner
- Develop a web application that will display information from the database
- Develop a server-less architecture to handle web application database queries

Secondary objectives

- Scale up number of instances transmitting pseudo sensor data
- Test load balancing aspects of back-end under increased scale
- End to end security testing of front-end to back-end
- End to end security testing of IoT simulation to back-end
- End to end testing of whole system

Learning outcomes

- Interfacing nodejs with IoT simulation
- IBM DevOps services
- Server-less cloud architecture with microservices
- Container orchestration
- Cloud container and application security

1.3 Report outline

Literature survey

Summary of the various aspects of the project such as Docker, Cloud container services vendors, database, and application security research.

Theory

Expands upon in detail the technologies chosen for the project.

High level design

Diagram and discussion of how the multiple project components interact with each other from a high level of abstraction.

Project

Describes the project management strategy and the critical path.

Progress to date

Summarizes the progress made in each week of the semester.

Requirements facilities

Discussion of tools and financial requirements.

Bibliography

Listing of references used.

2 Literature survey

2.1 Section outline

The New Stack podcast proved to be a vital source of information for this thesis. The podcast provides analysis and explanations about application development and management at scale.**[1]** I first heard about Docker containers through this podcast. It continues to keep me updated on new emerging cloud technologies.

Figure 1

My supervisor suggested that I use containers to develop a back-end for an IoT wireless sensor network. This idea provided me with enough motivation and enthusiasm to fully engage with the literature survey process.

With so many cutting edge cloud technologies emerging, it was difficult to do significant research into any one technology. Due to this, certain technologies that were researched had to be cut out of the report in order to keep it relevant to the actual chosen technologies that will be used in the project.

Section Layout

- Docker survey
- Docker Swarm survey
- Cloud container service vendors survey
- Container deployment and management survey
- Cloud organizations and platforms survey
- NoSQL database survey
- Graph database survey
- IBM Cloudant database survey
- Graph visualization survey
- Programming languages survey

2.2 Docker

Docker containers

Docker containers wrap a piece of software in a complete file system that contains everything needed to run: code, runtime, system tools, and system libraries.[2]

Figure 2

Lightweight

Containers running on a single machine share the same operating system kernel, they start instantly and use less RAM. Images are constructed from layered filesystems and share common files, making disk usage and image downloads much more efficient.[2]

Open

Docker containers are based on open standards, enabling containers to run on all major Linux distributions and on Microsoft Windows -- and on top of any infrastructure.[2]

Secure by default

Containers isolate applications from one another and the underlying infrastructure, while providing an added layer of protection for the application.[2]

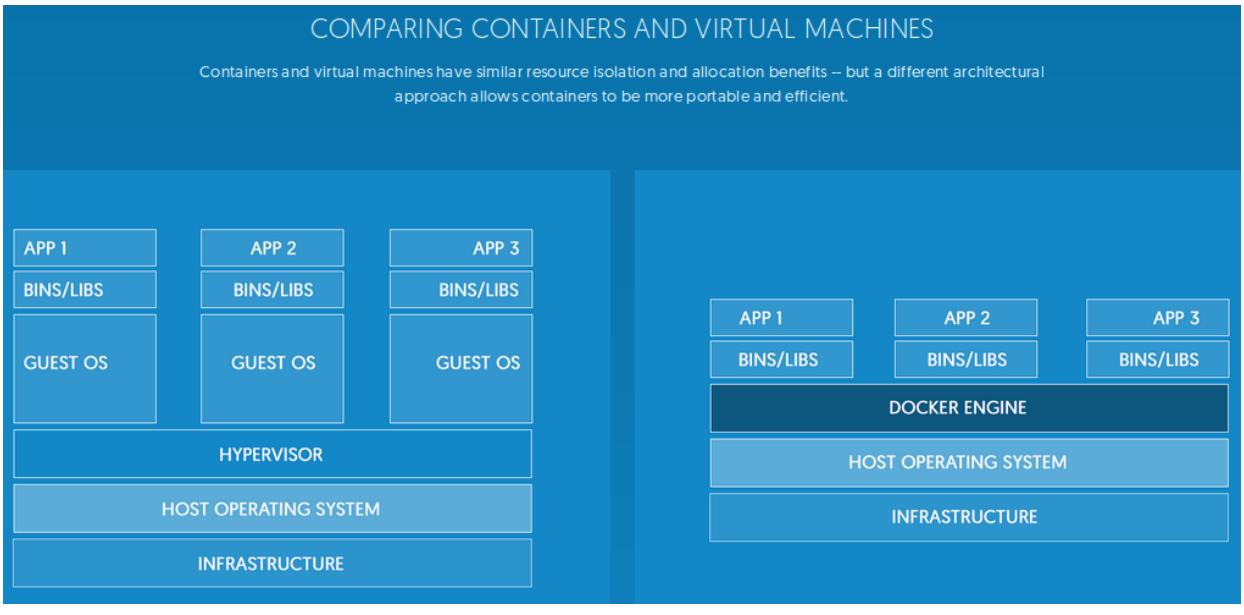


Figure 3

2.3 Docker Swarm

Docker Swarm provides native clustering capabilities to turn a group of Docker engines into a single, virtual Docker Engine. With these pooled resources, you can scale out your application as if it were running on a single, huge computer.**[3]**

Figure 4

Compatible with Docker Tools

Docker Swarm serves the standard Docker API, so any tool which already communicates with a Docker daemon can use Docker Swarm to transparently scale to multiple hosts: Dokku, Docker Compose, Krane, Flynn, Deis, DockerUI, Shipyard, Drone, Jenkins and, of course, the Docker client itself. Includes ability to pull from private repos within DTR or Hub as well.**[3]**

High Scalability and Performance

Swarm is production ready and tested to scale up to one thousand (1,000) nodes and fifty thousand (50,000) containers with no performance degradation in spinning up incremental containers onto the node cluster.**[3]**

Integrated Networking and Volumes

As a Docker native solution, you can use Docker Networking, Volumes and plugins through their respective Docker commands via Swarm.**[3]**

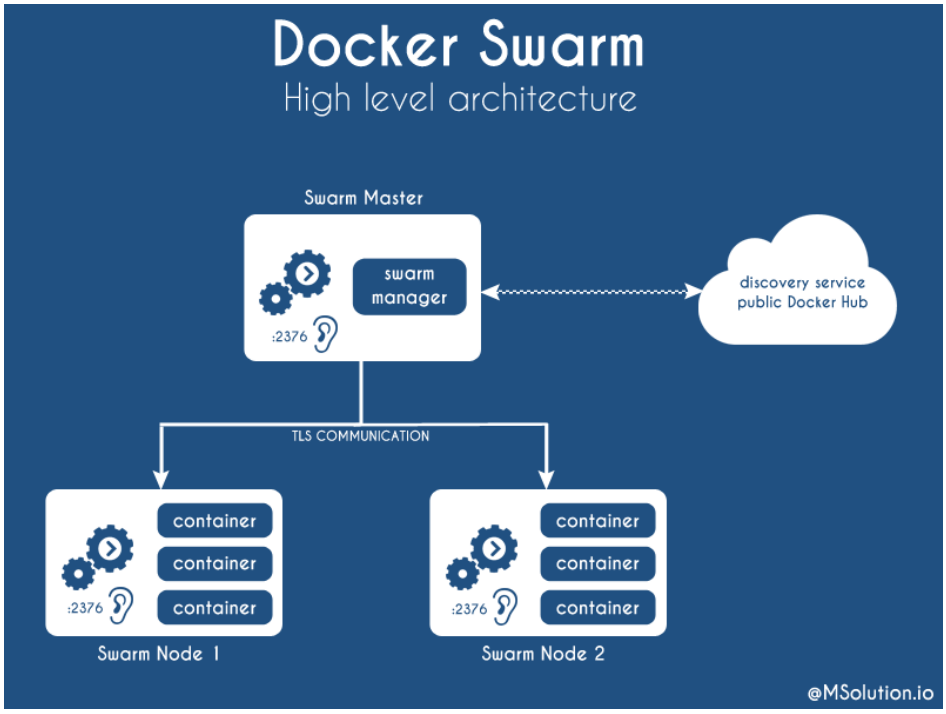


Figure 5

2.4 Container service vendors

Docker for IBM Bluemix

IBM and Docker offer integrated container solutions that can meet the diverse needs of enterprises. Supporting the creation and deployment of multi-platform, multi-container workloads across hybrid infrastructures, IBM and Docker accelerate application delivery and enable application lifecycle management for Dockerized containers.**[4]**

Figure 6

Docker for Azure

An integrated, easy-to-deploy environment for building, assembling, and shipping applications on Microsoft Azure, Docker for Azure is a native Azure application optimized to take optimal advantage of the underlying Azure IaaS services while giving you a modern Docker platform that you can use to deploy portable apps. Docker for Azure installs a Swarm of Docker Engines secured end to end with TLS by default, and is integrated with Azure VM Scale Sets for autoscaling, Azure Load Balancer and Azure Storage.**[5]**

Figure 7

Docker For AWS

An integrated, easy-to-deploy environment for building, assembling, and shipping applications on AWS, Docker for AWS is a native AWS application optimized to take optimal advantage of the underlying AWS IaaS services while giving you a modern Docker platform that you can use to deploy portable apps. Docker for AWS does not require any software installed.**[6]**

Figure 8

Docker for Google Cloud

Google Container Engine is a powerful cluster manager and orchestration system for running your Docker containers. Container Engine schedules your containers into the cluster and manages them automatically based on requirements you define (such as CPU and memory). It's built on the open source Kubernetes system, giving you the flexibility to take advantage of on-premises, hybrid, or public cloud infrastructure.**[7]**

Figure 9

2.5 Container deployment and management vendors

Kubernetes

An open-source platform for automating deployment, scaling, and operations of application containers across clusters of hosts, providing container-centric infrastructure. **[8]**

Figure 10

Kubernetes Features

- Deploy your applications quickly and predictably.
- Scale your applications on the fly.
- Seamlessly roll out new features.
- Optimize use of your hardware by using only the resources you need.

Apache Mesos

Mesos is built using the same principles as the Linux kernel, only at a different level of abstraction. The Mesos kernel runs on every machine and provides applications (e.g., Hadoop, Spark, Kafka, Elasticsearch) with API's for resource management and scheduling across entire datacenter and cloud environments.**[9]**

Figure 11

Mantl

Mantl is an end-to-end solution for deploying and managing a microservices infrastructure. Mantl currently uses Apache Mesos as a cluster manager for microservices.**[10]**

Figure 12

Mantl Features

- No vendor lock-in. Mantl runs equally well on any provider, saving you time and energy.
- Integrated tools like Cassandra, Spark & Hadoop.
- Service discovery, secret storage, load balancing, logging and more available right out-of-the-box.
- Run your services efficiently with multi-data center configuration and virtual networking tools.

2.6 Cloud organizations and platforms

Cloud Foundry

Cloud Foundry is an open source cloud platform as a service (PaaS) on which developers can build, deploy, run and scale applications on public and private cloud models. VMware originally created Cloud Foundry and it is now part of Pivotal Software. **[11]**

Figure 13

Cloud Foundry is licensed under Apache 2.0 and supports Java, Node.js, Go, PHP, Python and Ruby. The open source PaaS is highly customizable, allowing developers to code in multiple languages and frameworks. This eliminates the potential for vendor lock-in, which is a common concern with PaaS. **[11]**

Open Container Initiative

The Open Container Initiative (OCI) is a lightweight, open governance structure (project), formed under the auspices of the Linux Foundation, for the express purpose of creating open industry standards around container formats and runtime. The OCI was launched on June 22nd 2015. **[12]**

Figure 14

Cloud Native Computing Foundation

The Cloud Native Computing Foundation (CNCF) is a nonprofit organization committed to advancing the development of cloud native technology and services by creating a new set of common container technologies informed by technical merit and end user value, and inspired by Internet-scale computing. As a shared industry effort, CNCF members represent container and cloud technologies, online services, IT services and end user organizations focused on promoting and advancing the state of cloud native computing for the enterprise. **[13]**

Figure 15

OpenStack

OpenStack software controls large pools of compute, storage, and networking resources throughout a datacenter, managed through a dashboard or via the OpenStack API. OpenStack works with popular enterprise and open source technologies making it ideal for heterogeneous infrastructure. **[14]**

Figure 16

2.7 NoSQL database

A NoSQL database provides a mechanism for storage and retrieval of data which is modeled in means other than the tabular relations used in relational databases.**[15]**

NoSQL features

1. The ability to horizontally scale “simple operation” throughput over many servers.

2. The ability to replicate and to distribute (partition) data over many servers.

3. Simple call level interface or protocol (in contrast to SQL binding).

4. Efficient use of distributed indexes and RAM for data storage.

5. The ability to dynamically add new attributes to data records.

NoSQL database types

Document store

Documents are addressed in the database via a unique *key* that represents that document. One of the other defining characteristics of a document-oriented database is that in addition to the key lookup performed by a key-value store, the database offers an API or query language that retrieves documents based on their contents.**[15]**

Key value store

Data is represented as a collection of key-value pairs, such that each possible key appears at most once in the collection. The key-value model is one of the simplest non-trivial data models, and richer data models are often implemented as an extension of it. The key-value model can be extended to a discretely ordered model that maintains keys in lexicographic order.**[15]**

Graph

This kind of database is designed for data whose relations are well represented as a graph consisting of elements interconnected with a finite number of relations between them. The type of data could be social relations, public transport links, road maps or network topologies.**[15]**

2.8 Graph databases

Graphs are the most efficient and intuitive way of working with data, mimicking the interconnectedness of ideas in the human mind. Neo4j is built from the ground up to harness the power of graphs for real-time, bottom-line insights.**[16]**

Figure 17

Graph databases

Graph databases are based on graph theory. Graph databases employ nodes, edges and properties.

- **Nodes** represent entities such as people, businesses, accounts, or any other item you might want to keep track of. They are roughly the equivalent of the *record*, *relation* or *row* in a relational database.**[17]**
- **Edges**, also known as *graphs* or *relationships*, are the lines that connect nodes to other nodes; they represent the relationship between them.**[17]**
- **Properties** are pertinent information that relate to nodes.**[17]**

Graph example figure

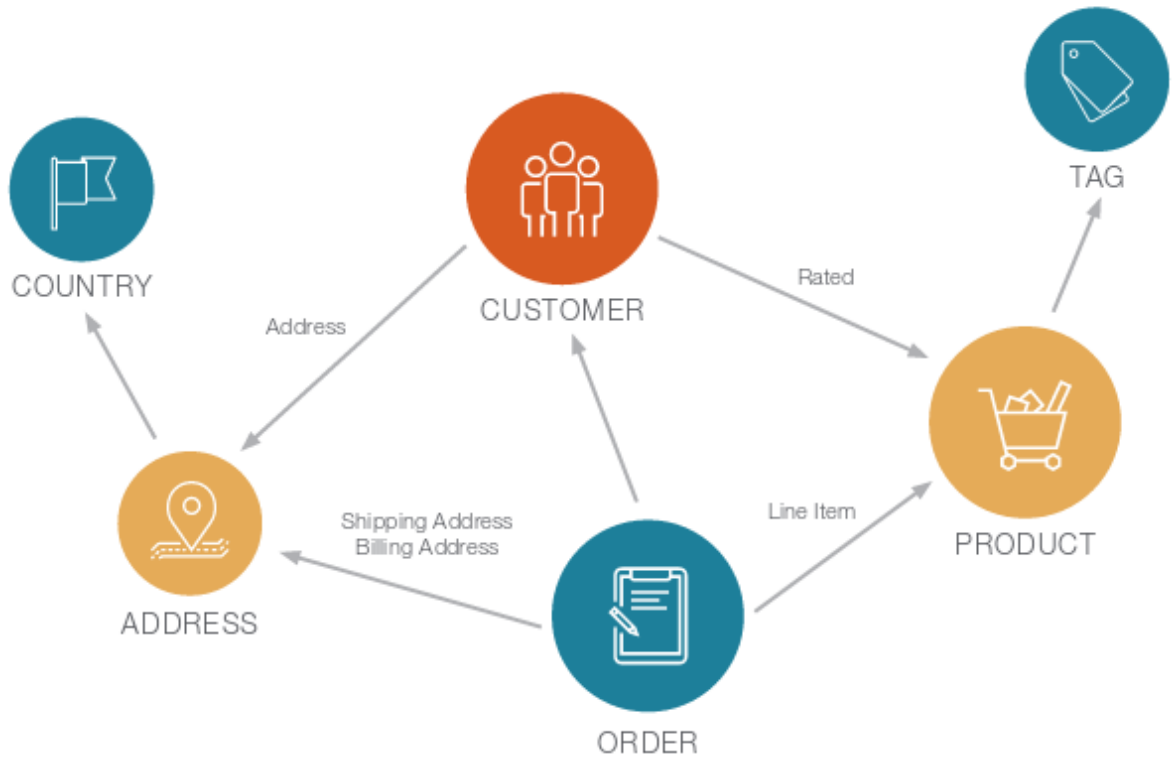


Figure 18

2.9 Graph visualization

Linkurious Graph Visualization

Linkurious uses Neo4j's graph database technology to offer you an easy solution to store, search and visualize graphs. You can start navigating inside your graph database. Search for properties, inspect nodes, and explore their relationships visually in your web browser. find any node in your database easily thanks to our built-in search engine. You can simply modify, add and remove nodes or relationships. You can also customize what is displayed, how the data is indexed.**[18]**

Figure 19

Keylines Graph visualization

KeyLines is an out-of-the-box JavaScript solution for visualizing networks. KeyLines does the job of rendering data and responding to user interactions like clicking, touching, moving nodes, and more. These events are binded to customize what happens, and, most importantly, the data stays under control at all times: KeyLines is self-contained and needs no external connections.**[19]**

Figure 20

Keylines example figure

Figure 21

2.10 IBM Cloudant

IBM Cloudant is a managed NoSQL JSON database service built to ensure that the flow of data between an application and its database remains uninterrupted. Developers are then free to build more, grow more.**[20]**

Figure 22

Global availability: Improve your app's performance

Cloudant's horizontal scaling architecture can handle millions of users and terabytes of data to grow seamlessly alongside your business.**[20]**

Users are connected to the closest copy of the data, which reduces data access latency caused by cloud network overhead.**[20]**

Data flexibility: JSON data store

Cloudant's RESTful API makes every document in your database accessible as JSON. It is also compatible with Apache CouchDB, enabling you to access an abundance of language libraries and tools.**[20]**

Schema flexibility makes Cloudant an excellent fit for multi-structured data, unstructured data and fast-changing data models.**[20]**

Advanced APIs: Integrated geospatial operations and search

Enhance web and mobile apps with geospatial operations that go beyond simple bounding boxes.

Seamlessly integrate web and mobile apps with fast, scalable, full-text Lucene indexing and search. **[20]**

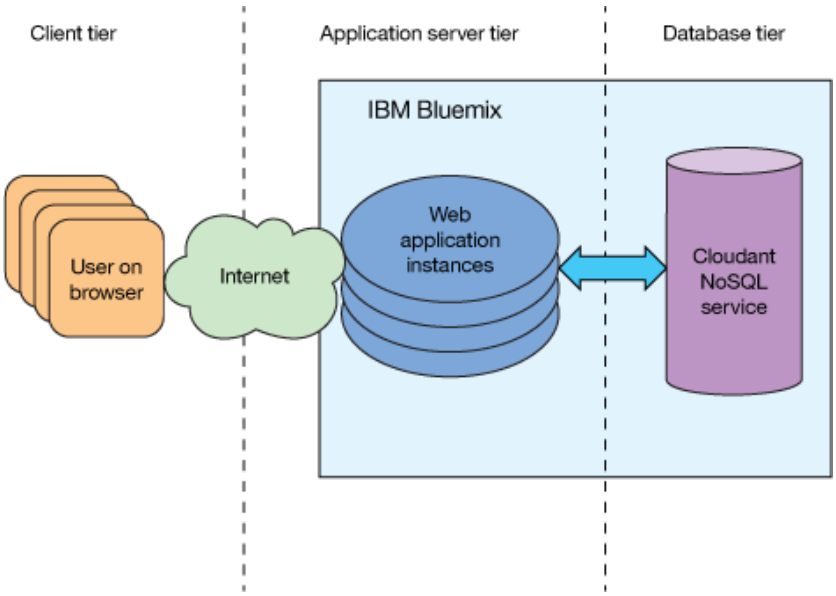


Figure 23

2.11 Programming languages

NodeJS

Figure 24

As an asynchronous event driven JavaScript runtime, Node is designed to build scalable network applications. Upon each connection the callback is fired, but if there is no work to be done Node is sleeping. This is in contrast to today's more common concurrency model where OS threads are employed. Thread-based networking is relatively inefficient and very difficult to use. Furthermore, users of Node are free from worries of dead-locking the process, since there are no locks. Almost no function in Node directly performs I/O, so the process never blocks. Because nothing blocks, scalable systems are very reasonable to develop in Node.**[21]**

Angular JS

Figure 25

AngularJS lets you write client-side web applications as if you had a smarter browser. It lets you use good old HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. It automatically synchronizes data from your UI (view) with your JavaScript objects (model) through 2-way data binding. To help you structure your application better and make it easy to test, AngularJS teaches the browser how to do dependency injection and inversion of control.**[22]**

Java

Figure 26

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible.**[23]**

Perl

Figure 27

Perl 5 is a highly capable, feature-rich programming language with over 29 years of development. Perl 5 runs on over 100 platforms from portables to mainframes and is suitable for both rapid prototyping and large scale development projects.**[24]**

PHP

Figure 28

What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was. You can even configure your web server to process all your HTML files with PHP, and then there's really no way that users can tell what you have up your sleeve.**[25]**

3 Theory

3.1 Section outline

IBM was chosen as the primary vendor as their catalog meets all the project requirements.

- IBM's Watson platform uses Node Red which is a visual tool for wiring the Internet of Things. This platform will be used for IoT simulation.
- Research into IoT protocols comprised of looking at IBM's WebSphere service. This service uses MQ Telemetry Transport which is a lightweight messaging protocol for IoT.
- IBM's OpenWhisk platform will provide the necessary tools for building the server-less architecture for the back-end that will receive pseudo data from the Watson platform.
- The REST protocol will act as the API gateway for the server-less architecture, abstracting away features of the back-end from the front-end website application.
- To meet the necessary requirements for monitoring GPS coordinates from the mobile application, IBM's Graph database service and IBM's Geo-spatial Analytics service were chosen.
- To incorporate continuous integration and delivery into the development life-cycle, IBM's DevOps service was chosen. This will help develop a custom development pipeline as well as automate repetitive tasks.
- The IBM Application Security Testing for Bluemix plug-in enables you to run security scans on your web applications that are hosted on Bluemix.
- Bluemix has both an auto-scaling and load-impact service, these services will be used in the later stages of the development life-cycle when testing scalability.

3.2 IBM IoT platform

IBM Watson

IBM Watson is a technology platform that uses natural language processing and machine learning to reveal insights from large amounts of unstructured data.**[26]**

Figure 29

Node Red

Node-RED is a tool for wiring together the Internet of Things in new and interesting ways, including hardware devices, APIs, and online services. It is built on top of Node.js and takes advantage of the huge node module ecosystem to provide a tool that is capable of integrating many different systems. Its lightweight nature makes it ideal to run at the edge of the network.**[27]**

Figure 30

Node-RED contains the following Watson IoT nodes that helps you to connect your devices, gateways and applications to Watson IoT Platform and create IoT solutions quickly.

- Watson IoT Node – A pair of nodes for connecting your device or gateway to the IBM Watson Internet of Things Platform. A device or gateway can use these nodes to send events and receive commands from the application.**[27]**
- IBM IoT App Node – A pair of nodes for connecting your application to Watson IoT Platform. An application can use these nodes to receive device events and send commands back to the device.**[27]**

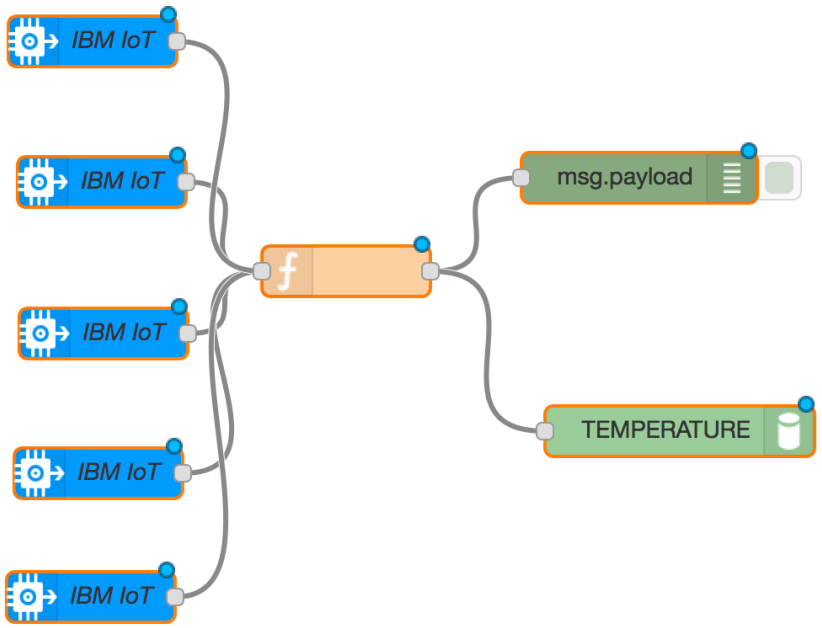


Figure 31

3.3 IBM Websphere

IBM WebSphere MQ Telemetry provides real-time access to a range of mobile devices, remote sensors, actuators and other telemetry devices leveraging WebSphere MQ. It uses the MQ Telemetry Transport (MQTT) protocol to transport robust messages to even the smallest devices for near instantaneous data exchange.**[28]**

Figure 32

MQTT Protocol

MQTT stands for MQ Telemetry Transport. It is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimise network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery. These principles also turn out to make the protocol ideal of the emerging “machine-to-machine” (M2M) or “Internet of Things” world of connected devices, and for mobile applications where bandwidth and battery power are at a premium. **[29]**

Figure 33

- TCP/IP port 1883 is reserved with IANA for use with MQTT.
- TCP/IP port 8883 is registered for using MQTT over SSL.
- Can pass a user name and password with an MQTT packet in V3.1 of the protocol. Encryption across the network can be handled with SSL, independently of the MQTT protocol itself

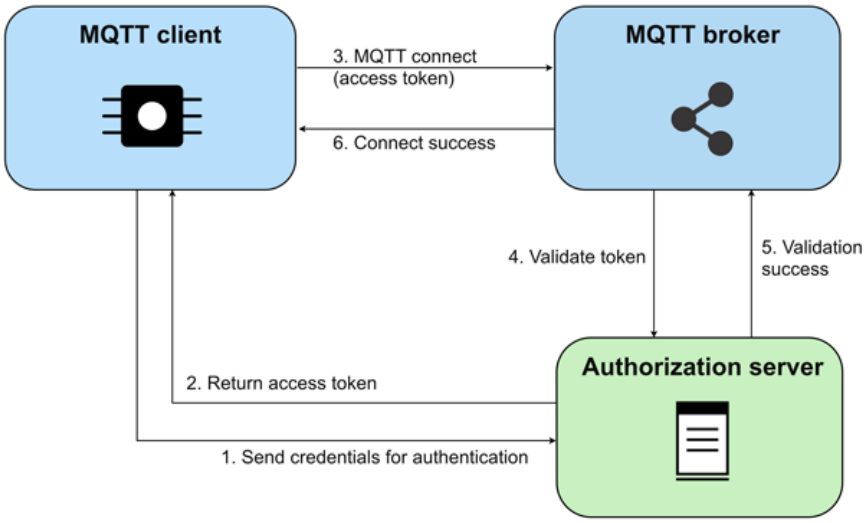


Figure 34

3.4 Serverless architecture

Serverless architectures refer to applications that significantly depend on third-party services (known as Backend as a Service or "BaaS") or on custom code that's run in ephemeral containers (Function as a Service or "FaaS").**[30]**

Server model

Figure 35

Serverless model

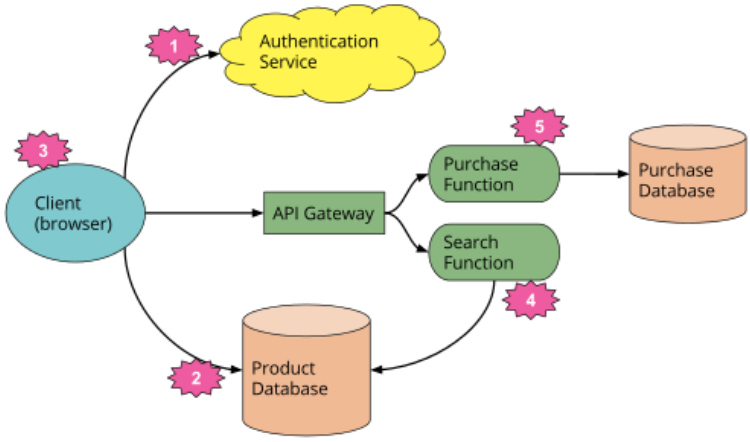


Figure 36

- The authentication logic in the server model has been replaced with a third party BaaS service.
- Allowed the client direct access to a subset of the database (for product listings).
- Some logic that was in the Pet Store server is now within the client, e.g. keeping track of a user session, understanding the UX structure of the application (e.g. page navigation), reading from a database and translating that into a usable view.
- Some UX related functionality will need to be kept in the server, e.g. if it's compute intensive or requires access to significant amounts of data, then a FaaS function that responds to http requests via an API Gateway can be implemented.
- Can now replace the 'purchase' functionality with another FaaS function, choosing to keep it on the server-side for security reasons, rather than re-implement it in the client.

3.5 IBM OpenWhisk

The OpenWhisk serverless architecture accelerates development as a set of small, distinct, and independent actions. By abstracting away infrastructure, OpenWhisk frees members of small teams to rapidly work on different pieces of code simultaneously, keeping the overall focus on creating user experiences customers want.**[31]**

Figure 37

key architectural concepts:

- **Triggers:** A class of events emitted by event sources.

- **Actions:** Encapsulate the actual code to be executed which support multiple language bindings including NodeJS, Swift and arbitrary binary programs encapsulated in Docker Containers. Actions invoke any part of an open ecosystem including existing Bluemix services for analytics, data, cognitive, or any other 3rd party service.

- **Rules:** An association between a trigger and an action

  .

- **Packages:** Describe external services in a uniform manner.

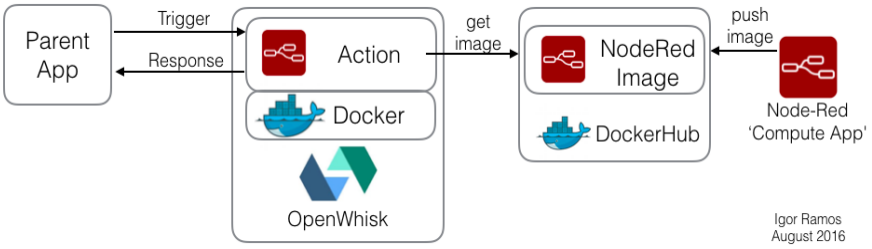


Figure 38

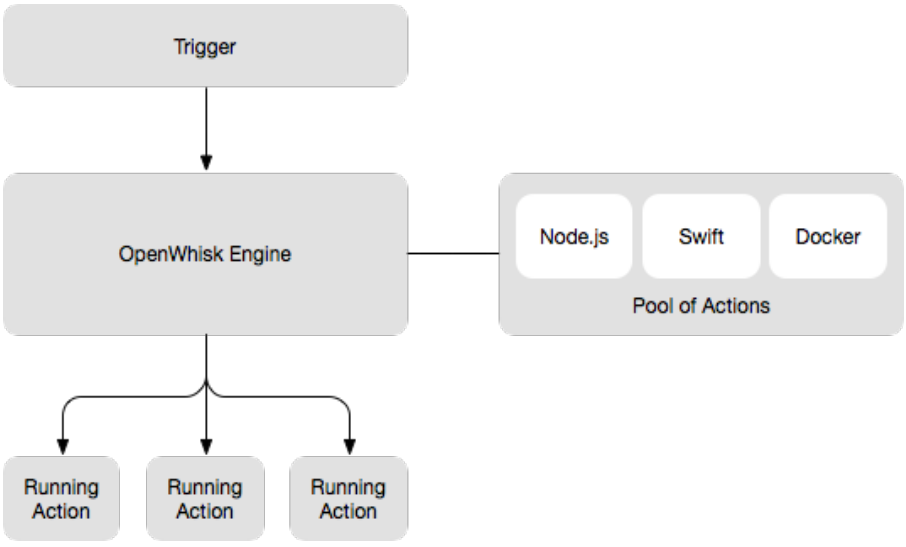


Figure 39

3.6 REST API

REST (REpresentational State Transfer) is an architectural style, and an approach to communications that is often used in the development of Web services. The use of REST is often preferred over the more heavyweight SOAP (Simple Object Access Protocol) style because REST does not leverage as much bandwidth, which makes it a better fit for use over the Internet. The SOAP approach requires writing or using a provided server program (to serve data) and a client program (to request data).**[32]**

{ REST }

Figure 40

REST'S decoupled architecture, and lighter weight communications between producer and consumer, make REST a popular building style for cloud-based APIs.**[32]**

REST architecture involves reading a designated Web page that contains an XML file. The XML file describes and includes the desired content. Once dynamically defined, consumers may access the interface.**[32]**

REST Architecture example diagram

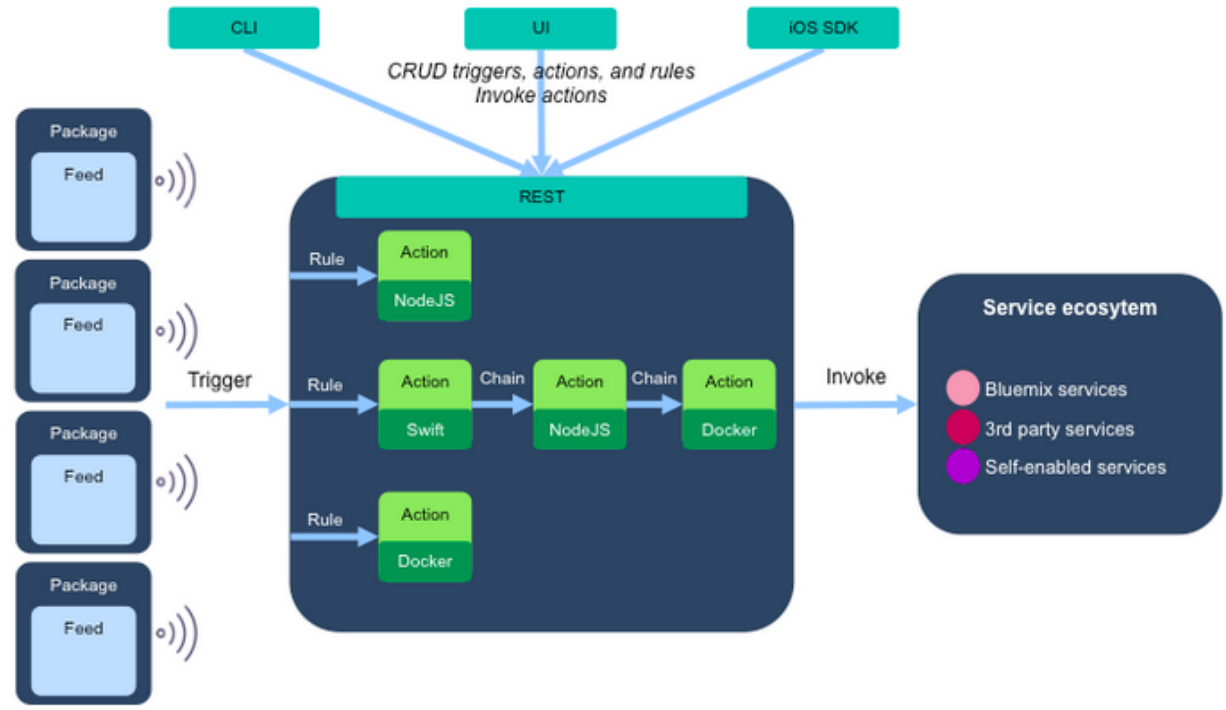


Figure 41

3.7 Graph database

3.2.1 IBM Graph platform

IBM Graph is an easy-to-use, fully-managed graph database service for storing and querying data points, their connections, and properties. IBM Graph offers an Apache TinkerPop3 compatible API and plugs into your Bluemix application seamlessly.**[33]**

Figure 42

Features

Highly Available

Architecture ensures the service is always up and your data is always accessible.

Scale Seamlessly

Start small and scale on-demand as data grows.

Managed 24x7

Stack is monitored, managed, and optimized by experts 365 days a year.

IBM Graph Standard Plan

Plan	Features	Pricing
Standard	• 500 MB of free data storage • 25,000 API calls free per month	• $15.00 USD/GB • $0.20 USD/1000 API Calls

Geospatial Analytics service

Expand the boundaries of your application. Leverage real-time geospatial analytics to track when devices enter, leave or hang out in defined regions. Powered by IBM Streaming Analytics on Bluemix.**[34]**

Figure 43

Features

- Monitor device locations in real-time.
- Connect to data sources that support the MQTT protocol and monitor devices as they move into geographic regions of interest.
- Control region monitoring using the geospatial API.
- Define geographic regions and control monitoring of regions using the geospatial application.

3.8 IBM Bluemix DevOps Services

Connect with your GitHub repository

It's easy to link work items with GitHub code changes: use your tools to code, and manage your project with IBM Bluemix DevOps Services. You can reference a work item in GitHub comments before you push code changes, and the work item is updated with the new commit information.**[35]**

Figure 44

Your work in one place

At a glance, see the projects that you own, belong to, and like. You can also see requests to join your project and invitations for you to join other projects. Just another way to help you stay organized.**[35]**

Built-in source code management

Easily deliver code and seamless track changes from all developers. Each project gets a Git repository or a Jazz source code management repository and workspace where project members check in changes, associate code changes with work items, and view a history of recent updates.**[35]**

Integrated source code editor

The Web IDE provides features to support these tasks:

- Import your source code.
- Copy a file by dragging it to a new directory.
- Edit code quickly. The editor includes several features:
  - Content assist for CSS, HTML, and JavaScript
  - Syntax highlighting for over 20 popular programming languages
  - Syntax highlighting, validation, and content assist for Cloud Foundry manifest files
  - Code validation
  - Keyboard shortcuts
  - Bracket and block-comment auto-completion
  - Key-bindings for vi and emacs users

3.9 IBM Bluemix Security

IBM Bluemix security features

The Bluemix platform secures data-in-transit by securing the end-user access to the application by using SSL through the network until the data reaches IBM DataPower Gateway at the boundary of the Bluemix internal network. IBM DataPower Gateway acts as a reverse proxy and provides SSL termination. From there to the application, IPSEC is used to secure the data as it travels from the IBM DataPower Gateway to the application.**[37]**

Security for both data-in-use and data-at-rest is your responsibility as you develop your application. You can take advantage of several data-related services available in the Bluemix Catalog to help with these concerns.**[37]**

You can use security capabilities that are provided by several Bluemix services to secure your applications. All Bluemix services that are produced by IBM follow IBM secure engineering development practices.**[37]**

IBM UrbanCode plug-in for application security testing

The IBM Application Security Testing for Bluemix plug-in enables you to run security scans on your web or Android apps that are hosted on Bluemix. This plug-in is developed and supported by the IBM UrbanCode Deploy Community on the IBM Bluemix DevOps Services platform.**[36]**

Figure 45

Bluemix Vulnerability advisor

Discover vulnerabilities and compliance policy problems in Docker images and learn new ways to improve images to meet best practices and upgrades to known industry fixes, regardless of the image source.**[37]**

Bluemix Integrated container monitoring

Log visibility and performance insight to the CPU, memory, and network utilization per container.**[37]**

3.10 Load balancing and auto-scaling services

Load Impact service

Worlds #1 load testing tool - trusted by over 120,000 developers and testers. Unlimited testing, on-demand from multiple geographic locations. Create sophisticated tests using the simple GUI or connect directly to the platform via the API.**[38]**

Figure 46

Load Impact service Standard Plan

- Max 100 concurrent users (VUs) per test
- 5 tests per month
- Max 5 minutes test duration
- Max 1 server metric agents

Auto-Scaling service

The Auto-Scaling for Bluemix service enables you to automatically increase or decrease the compute capacity of your application. The number of application instances are adjusted dynamically based on the Auto-Scaling policy you define.**[39]**

Features

- Automatically add or remove resources to match the current workload.
- Define policy on metrics of interest.
- Visualize the current and historical values of performance metrics.
- Query the scaling activities based on status, time and type.

3.11 IBM Softlayer

SoftLayer, an IBM Company, provides cloud infrastructure as a service from a growing number of data centers and network points of presence around the world. Their customers range from Web startups to global enterprises.

SOFTLAYER®
an IBM Company

Figure 47

Products and services include bare metal and virtual servers, networking, turnkey big data solutions, private cloud solutions, and more. Their unique advantages include the industry's first Network-Within-a-Network topology for true out-of-band access, and an easy-to-use customer portal and robust API for full remote-access of all product and service management options.**[40]**

4 High level design

4.1 High level design

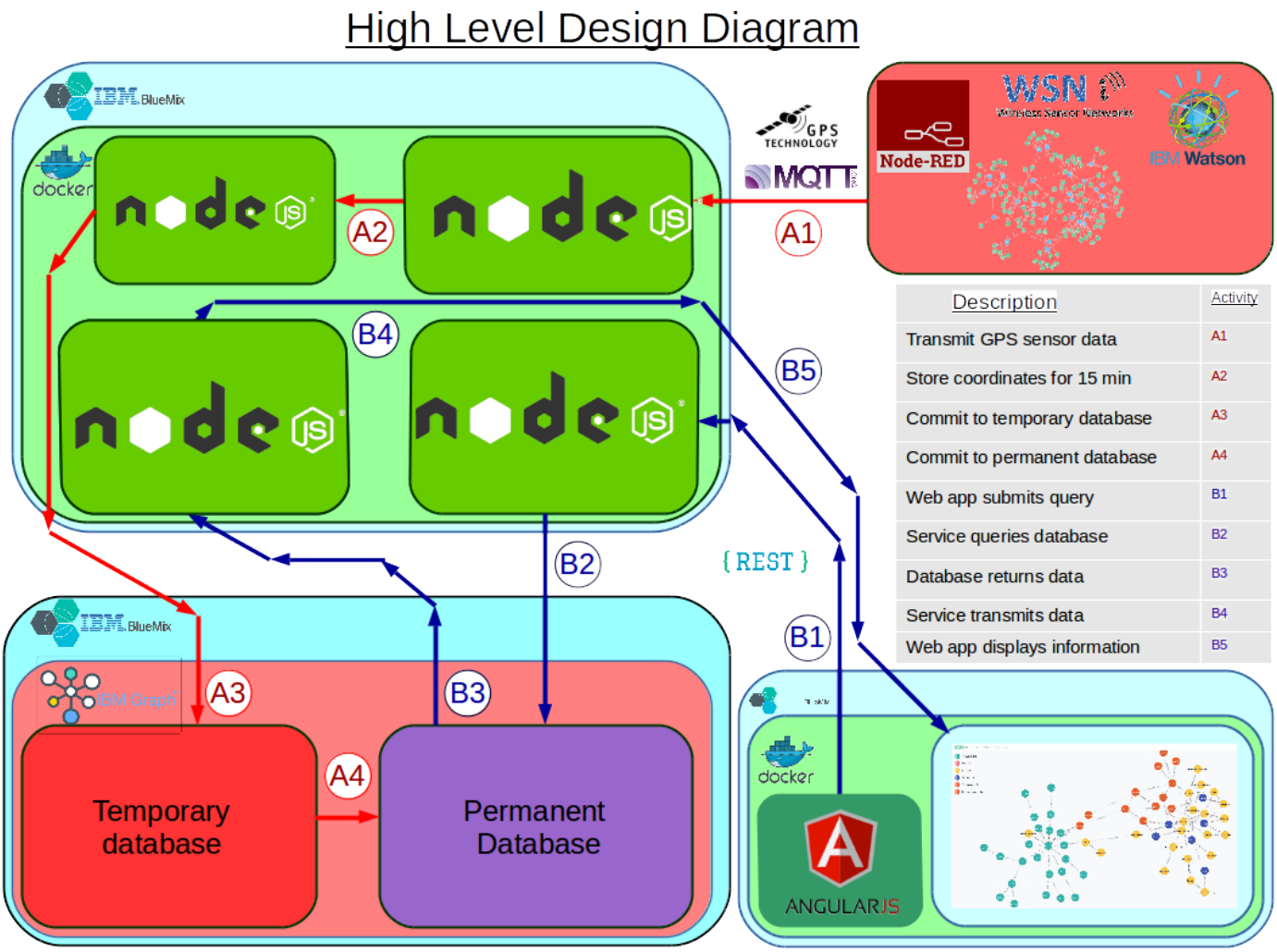


Figure 48

4.2 Discussion

This diagram is taken from a high level of abstraction. This will allow for flexibility in the development process as requirements, architecture and task priorities change.

Figure 48 displays 2 different use cases.

- **Red:** The path the GPS data takes from sensor to database.
- **Blue:** Interaction of the web application user.

5 Project

5.1 Section outline

This section describes the tools used for both managing, planning and documenting the project.

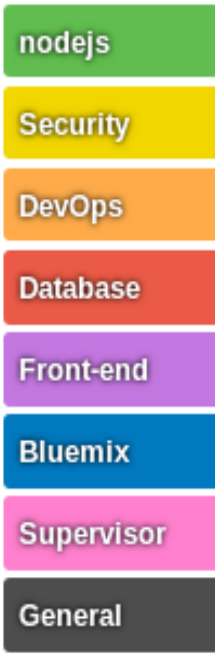
- Trello was chosen as the project management tool for the thesis. I chose this tool out of convenience as I use it in everyday life.
- Toggl was chosen as the tool to log the time that goes into the project. Toggl provides graphs that can be customized to your needs.
- Github was chosen as the remote repository for the thesis. I chose to use Git version control because I'm more familiar with it than other version control systems.
- The project tasks were first written down and organized using a project plan map, from this I was able to determine the critical path of project.
- Using the project plan, I estimated how long each task would take and then constructed a Gantt chart from it.

5.2 Trello

Trello uses the kanban paradigm for managing projects, originally popularized by Toyota in the 1980s for supply chain management. Projects are represented by *boards*, which contain *lists* (corresponding to task lists). Lists contain *cards* (corresponding to tasks). Cards are supposed to progress from one list to the next (via drag-and-drop), for instance mirroring the flow of a feature from idea to implementation.**[41]**

Figure 49

Project Board Labels



- Backend/nodejs related tasks
- Security related tasks
- Devops related tasks
- Database related tasks
- Website related tasks
- Bluemix/Docker related tasks
- Supervisor related tasks
- Related to general tasks such as documentation.

Figure 50

Project lists

Collect: Whenever a new task arises, it gets put in here.

This week: Tasks from “collect” list are moved to this list at the start of each week.

Today: Tasks from “this week” list are moved to this list everyday.

Next action: Tasks from “today” list are moved to this list one at a time.

Done: Tasks from the “today” list are moved to this list when completed.

5.3 Toggl

Toggl is time tracking software that offers online time tracking and reporting services through their website along with mobile and desktop applications. Toggl tracks time based on tasks and projects, either through an interactive task timer or through manual entry.**[42]**

Figure 51

5.4 Github

GitHub is a web-based Git repository hosting service. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.**[43]**

Figure 52

GitHub offers both plans for private repositories, and free accounts which are commonly used to host open-source software projects. As of April 2016, GitHub reports having more than 14 million users and more than 35 million repositories, making it the largest host of source code in the world.**[40]**

GitHub Account

Username: 16117743

Repositories

INS-Thesis-Documentation

This repository contains commits of my weekly logs and semester reports.**[50]**

INS-Thesis

This repository was setup before the project had a defined scope. This project will require a minimum of two separate repositories for the development of the front and back-end.

5.5 Spring action plan

Table of tasks for Spring semester

Task Description	Task Identity	Duration (days)	Preceding Task
Research similar microservices	JA1	5	
Research NoSQL query languages	JA2	5	
Research IBM graph examples	JA3	5	JA2
Deploy and run MS1 that creates and transmits pseudo sensor data	JA4	5	JA1
Deploy and run MS2 that handles incoming data	FE1	5	JA4
Deploy a basic graph database on IBM graph with suitable pseudo data	FE2	5	JA3
Research Bluemix website deployment examples	FE3	3	
Deploy a basic website on Bluemix	FE4	3	FE3
Create permanent graph DB	FE5	5	FE4
Create temporary	FE6	5	FE5
Develop code for MS2 to store incoming data for a short period of time	MAR1	5	FE1
MS2 writes to temporary database every 15 minutes	MAR2	5	MAR1 & FE6
MS3 reads data from temporary database	MAR3	5	
MS3 writes to permanent database	MAR4	5	MAR3
Develop map user interface for website	APR1	5	FE4
Identify important UL GPS coordinates for map user interface	APR2	5	APR1
Develop code for website to submit a simple database request	APR3	5	APR2
MS4 handles a simple website client database request	APR4	5	APR3 & MAR4

Spring Critical Path Map

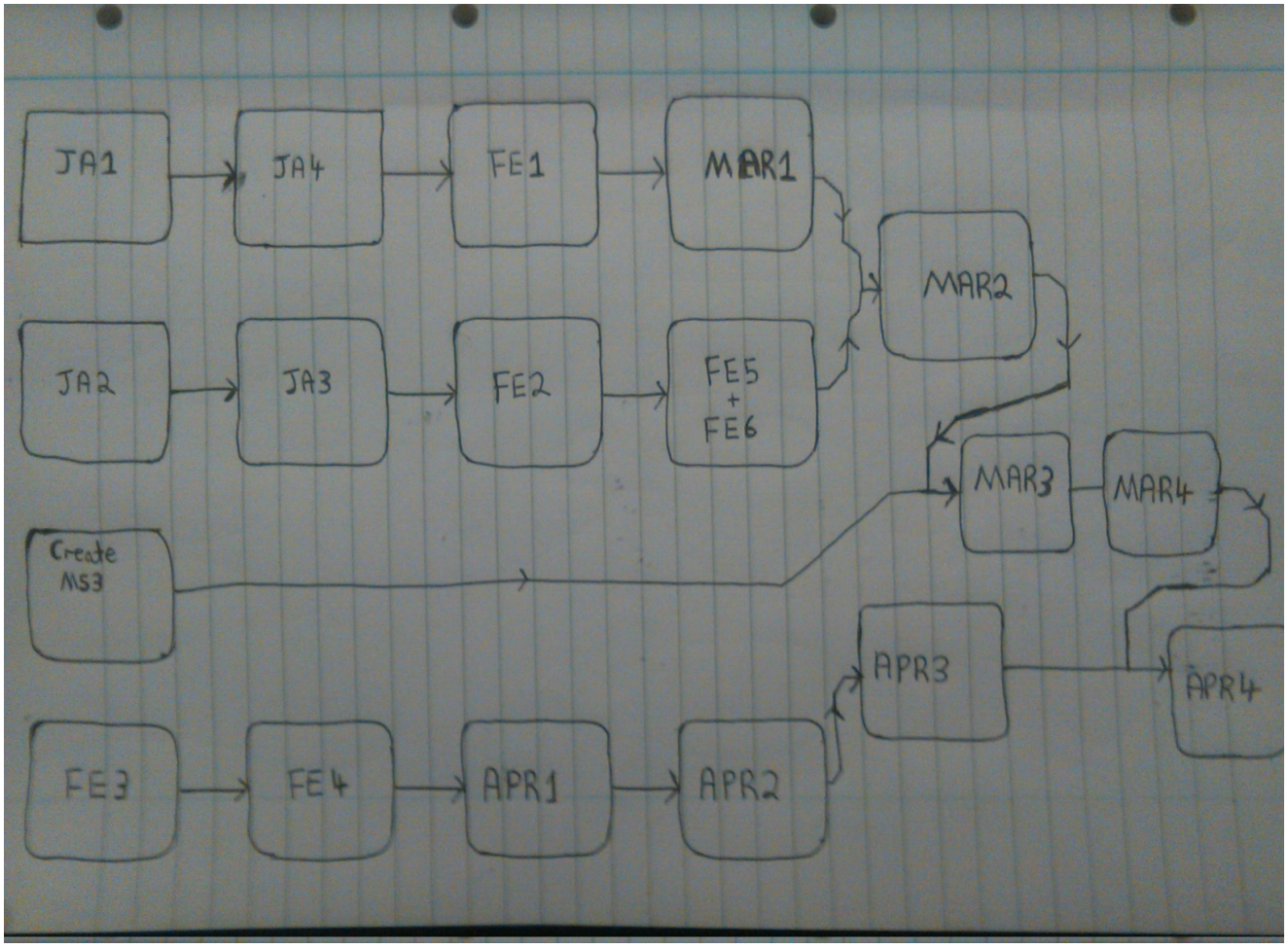


Figure 53

Spring Gantt Chart

Task Identity	Start	Days	End
JA1	02/01/17	5	08/01/17
JA2	09/01/17	5	15/01/17
JA3	16/01/17	5	22/01/17
JA4	23/01/17	5	29/01/17
FE1	30/01/17	5	05/02/17
FE2	06/02/17	5	12/02/17
FE3	13/02/17	3	15/02/17
FE4	16/02/17	3	20/02/17
FE5	20/02/17	5	26/02/17
FE6	27/02/17	5	05/03/17
MAR1	06/03/17	5	12/03/17
MAR2	13/03/17	5	19/03/17
MAR3	20/03/17	5	26/03/17
MAR4	27/03/17	5	02/04/17
APR1	03/04/17	5	09/04/17
APR2	10/04/17	5	16/04/17
APR3	17/04/17	5	23/04/17
APR4	24/04/17	5	30/04/17

Figure 54

5.6 Summer action Plan

Table of tasks for summer semester

Task Description	Task Identity	Duration	Preceding Task
Website displays DB readings on website map user interface	MAY1	5	APR4
Load balancing and scalability research	MAY2	5	
Increase funding	MAY3	5	
Increase MS1 instances	MAY4	5	MAY3
MS2 Automatically handles increase/decrease of incoming sensor data from MS1	MAY5	5	MAY2 & MAY 4
Test scalability of MS2 writing to temporary database	JUN1	9	MAY1 & MAY 5
Test scalability of MS3 reading from temporary database and writing to permanent database	JUN2	8	JUN1
End to end testing of website and database under increased scale	JUN3	3	JUN2
End to end testing of the whole system	JLY1	4	JUN3
Collect results	JLY2	2	JLY1
Analyze results	JLY3	3	JLY2
Discuss results with supervisor	JLY4	3	JLY3
Write up initial report and submit to supervisor for review	JLY5	3	JLY4
Edit initial report	JLY6	3	JLY5

Summer critical path map

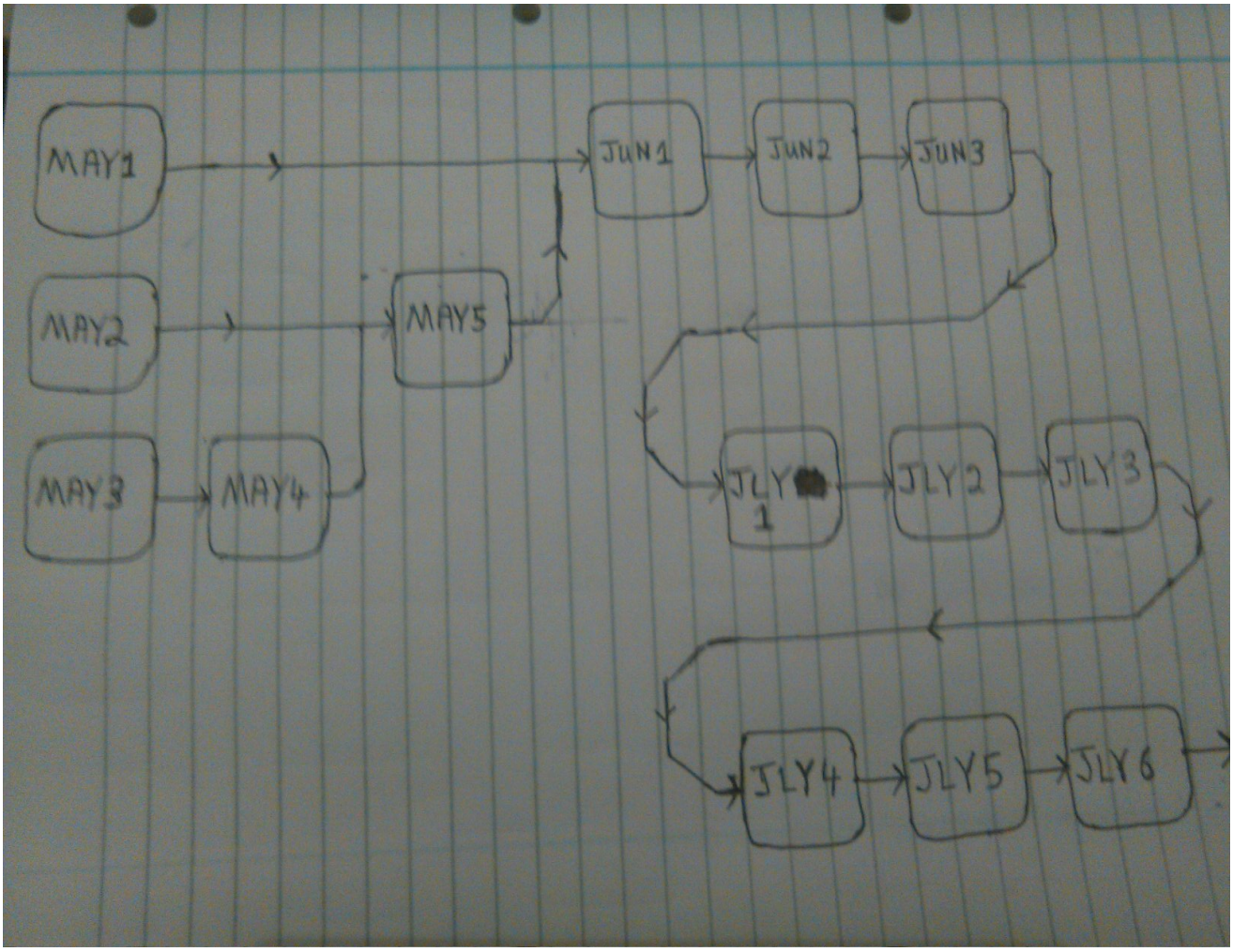


Figure 55

Summer Gantt chart

Task Identity	Start	Days	End
MAY1	01/05/17	5	07/05/17
MAY2	08/05/17	5	14/05/17
MAY3	15/05/17	5	21/05/17
MAY4	23/05/17	5	29/05/17
MAY5	29/05/17	5	04/06/17
JUN1	05/06/17	9	15/06/17
JUN2	16/06/17	8	27/06/17
JUN3	28/06/17	3	02/07/17
JLY1	03/07/17	4	06/07/17
JLY2	07/07/17	2	10/07/17
JLY3	11/07/17	3	13/07/17
JLY4	14/07/17	3	18/07/17
JLY5	19/07/17	3	23/07/17
JLY6	24/07/17	3	26/07/17

Figure 56

6 Progress to date

6.1 Section outline

This section gives a summary of the work done for each week of the Autumn semester. Each log can be found on GitHub in the INS-Thesis-Documentation repository.**[44]**

6.2 Week 5 summary

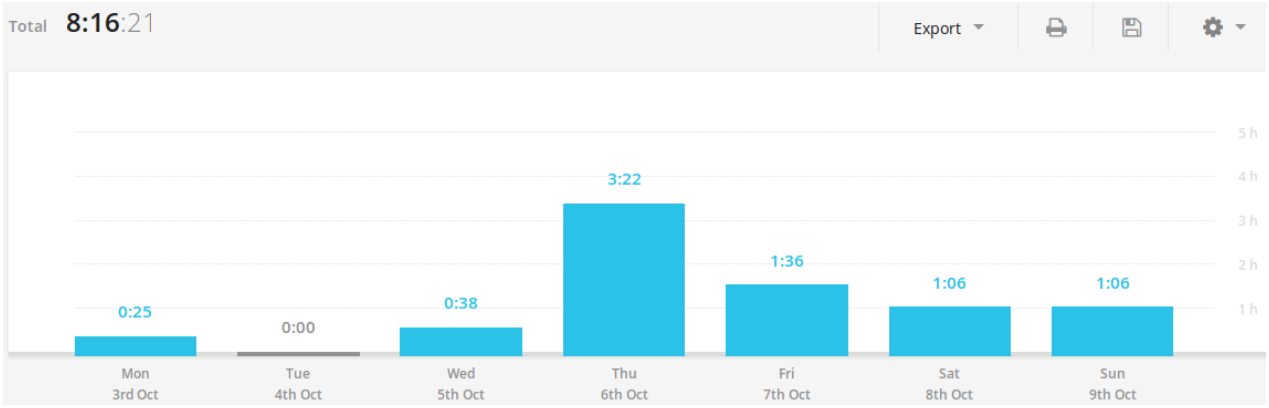


Figure 57

Tasks completed

- Create Toggl Project Labels
- Create Project folders
- Create Log template
- Arrange meeting with supervisor
- Backup files for installation of Ubuntu
- Initial Research
- Research Bluemix
- Meet with supervisor
- Reflect and organize
- Download Ubuntu ISO
- Install Ubuntu

6.3 Week 6 summary

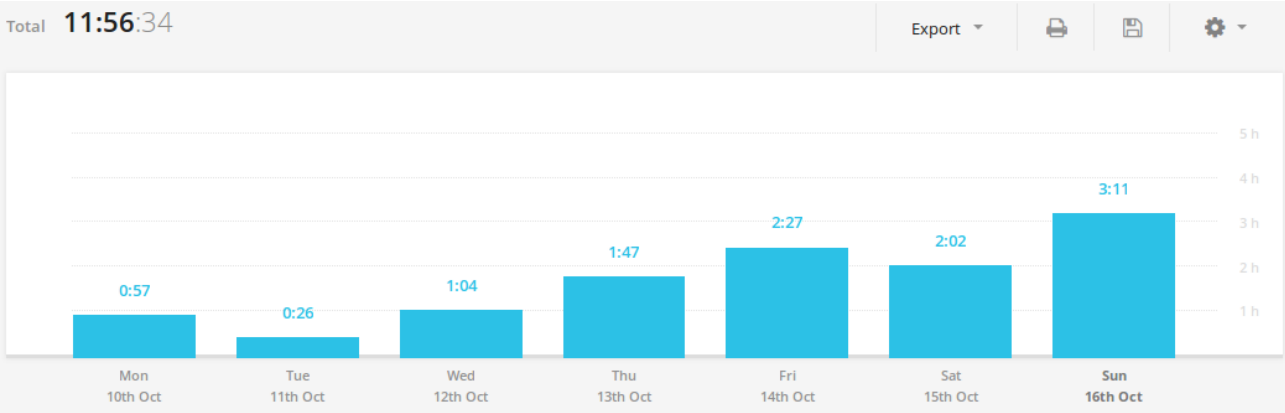


Figure 58

Tasks completed

- Collected Bluemix competitor bookmarks
- Read Bluemix competitor bookmarks
- Create Github repositories
- Take photos of hand written notes
- Finish week 5 log
- Installed Docker
- Amazon container service research
- Microsoft Azure container service research
- IBM Bluemix research
- Docker research
- Initial database research
- NoSQL database research
- Neo4j graph database research
- Layout Autumn report

6.4 Week 7 summary

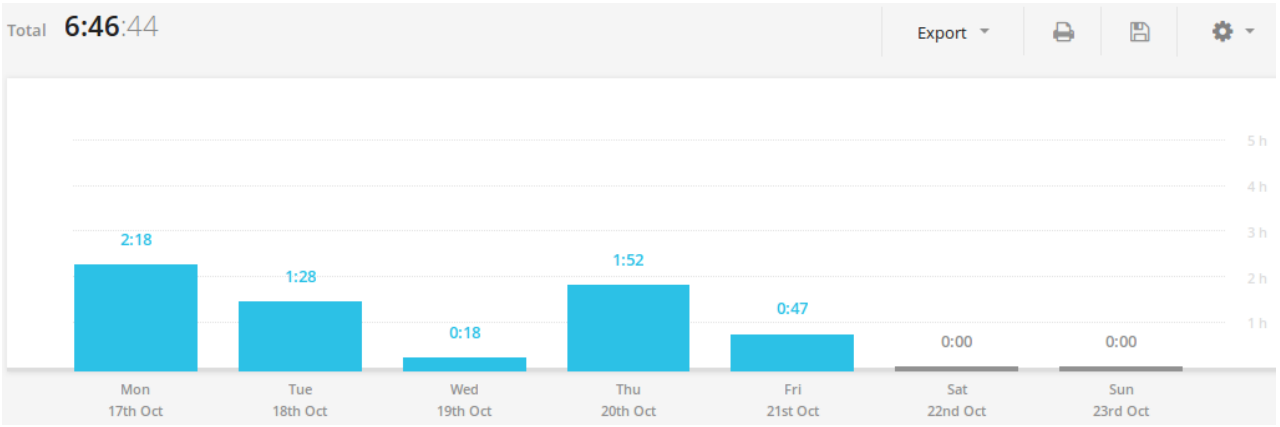


Figure 59

Tasks completed

- Week6 log
- layout project plan
- layout critical path
- Sprint Gantt chart
- Summer Gantt chart
- Autumn report project plan section
- Commit report to github
- Two page sumary of project

6.5 Week 8 summary

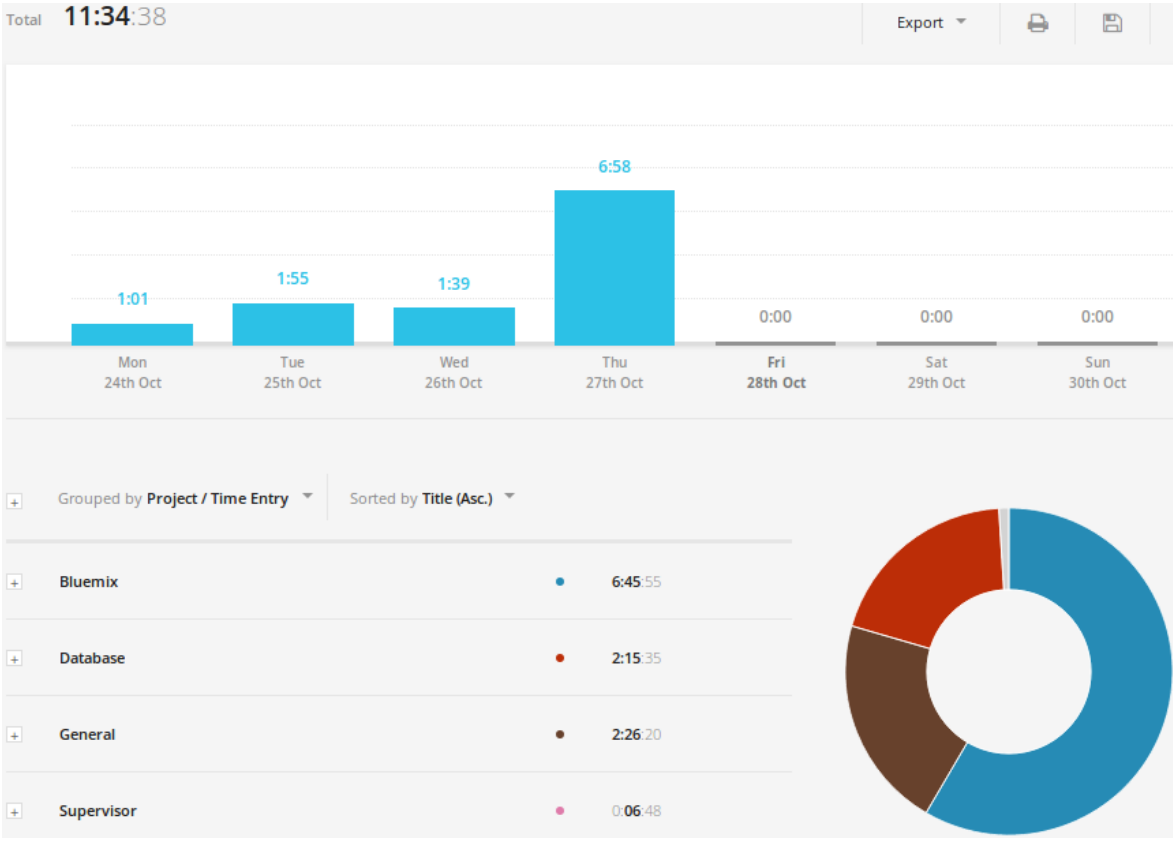


Figure 60

Tasks completed

- Week 7 log
- Create Bluemix account
- Email supervisor
- Simulate IoT sensor database
- Install Bluemix CLI
- Install Bluemix CLI plugins
- Docker Bluemix demo

7 Requirements of facilities and materials

7.1 Financial requirements

This project requires a Bluemix subscription. The subscription is within the budget constraints set by the University of Limerick. Further examination of the budget constraints will be assessed in the spring semester.

Bluemix services

Service	Features	Price
SDK for Node.js	375 GB-hours free per month (shared across all runtimes)	$0.07 USD/GB-hour
Auto-Scaling	Free service plan for the Auto-Scaling service	Free
IBM Graph	• 500 MB of free data storage • 25,000 API calls free per month	• $15.00 USD/GB • $0.20 USD/1000 API Calls
IBM Cloudant NoSQL DB for Bluemix	2 GB of free data storage 50,000 light API calls free per month 10,000 heavy API calls free per month	$1.00 USD/GB $0.03 USD/1000 light API calls $0.15 USD/1000 heavy API calls
Watson Analytics	IoT simulation with Node Red	$30 USD a month

8 Bibliography

[1]"About - The New Stack", *The New Stack*, 2016. [Online]. Available: http://thenewstack.io/about/. [Accessed: 03- Nov- 2016].

[2]"What is Docker?", *Docker*, 2015. [Online]. Available: https://www.docker.com/what-docker. [Accessed: 14- Oct- 2016].

[3]"Docker Swarm", *Docker*, 2016. [Online]. Available: https://www.docker.com/products/docker-swarm. [Accessed: 03- Nov- 2016].

[4]"IBM", *Docker*, 2015. [Online]. Available: https://www.docker.com/IBM. [Accessed: 14- Oct- 2016].

[5]"Microsoft", *Docker*, 2015. [Online]. Available: https://www.docker.com/microsoft. [Accessed: 14- Oct- 2016].

[6]"Get Started with Docker and AWS", *Docker*, 2015. [Online]. Available: http://www.docker.com/aws. [Accessed: 14- Oct- 2016].

[7]"Google Cloud Computing, Hosting Services & APIs | Google Cloud Platform", *Google Developers*, 2016. [Online]. Available: https://cloud.google.com/. [Accessed: 03- Nov- 2016].

[8]"Kubernetes - What is Kubernetes?", *Kubernetes.io*, 2016. [Online]. Available: http://kubernetes.io/docs/whatisk8s/. [Accessed: 03- Nov- 2016].

[9]"Apache Mesos", *Mesos.apache.org*, 2016. [Online]. Available: http://mesos.apache.org/. [Accessed: 03- Nov- 2016].

[10]"Home | Mantl.io", *Mantl.io*, 2016. [Online]. Available: https://mantl.io/. [Accessed: 03- Nov- 2016].

[11]"Cloud Foundry | The Industry Standard for Cloud Applications", *Cloud Foundry*, 2016. [Online]. Available: https://www.cloudfoundry.org/. [Accessed: 03- Nov- 2016].

[12]"About | Open Container Initiative", *Opencontainers.org*, 2016. [Online]. Available: https://www.opencontainers.org/about. [Accessed: 03- Nov- 2016].

[13]"About - Cloud Native Computing Foundation", *Cloud Native Computing Foundation*, 2016. [Online]. Available: https://www.cncf.io/about. [Accessed: 03- Nov- 2016].

[14]"Open source software for creating private and public clouds.", *OpenStack*, 2016. [Online]. Available: https://www.openstack.org/. [Accessed: 03- Nov- 2016].

[15]"Learn About NoSQL Databases | NoSQL vs SQL, Benefits", *DataStax*, 2016. [Online]. Available: http://www.datastax.com/nosql-databases. [Accessed: 03- Nov- 2016].

[16]"Neo4j: The World's Leading Graph Database", *Neo4j Graph Database*, 2016. [Online]. Available: https://neo4j.com/. [Accessed: 03- Nov- 2016].

[17]"What is a Graph Database? A Property Graph Model Intro", *Neo4j Graph Database*, 2016. [Online]. Available: https://neo4j.com/developer/graph-database/. [Accessed: 03- Nov- 2016].

[18]"Linkurious - Understand the connections in your data", *Linkurious*, 2016. [Online]. Available: http://linkurio.us/about/. [Accessed: 03- Nov- 2016].

[19]"KeyLines Network Visualization Software", *Cambridge Intelligence*, 2016. [Online]. Available: http://cambridge-intelligence.com/keylines/. [Accessed: 03- Nov- 2016].

[20]"IBM Cloudant - IBM Cloud Data Services- IBM Analytics", *Ibm.com*, 2016. [Online]. Available: http://www.ibm.com/analytics/us/en/technology/cloud-data-services/cloudant/. [Accessed: 03- Nov- 2016].

[21]"About | Node.js", *Nodejs.org*, 2016. [Online]. Available: https://nodejs.org/en/about/. [Accessed: 03- Nov- 2016].

[22]"AngularJS — Superheroic JavaScript MVW Framework", *Angularjs.org*, 2016. [Online]. Available: https://angularjs.org/. [Accessed: 03- Nov- 2016].

[23]"What is Java and why do I need it?", *Java.com*, 2016. [Online]. Available: https://java.com/en/download/faq/whatis_java.xml. [Accessed: 03- Nov- 2016].

[24]"The Perl Programming Language - www.perl.org", *Perl.org*, 2016. [Online]. Available: https://www.perl.org/. [Accessed: 03- Nov- 2016].

[25]"PHP: What is PHP? - Manual", *Php.net*, 2016. [Online]. Available: http://php.net/manual/en/intro-whatis.php. [Accessed: 03- Nov- 2016].

[26]"What is IBM Watson?", *Ibm.com*, 2016. [Online]. Available: http://www.ibm.com/watson/what-is-watson.html. [Accessed: 03- Nov- 2016].

[27]"Node-RED", *Nodered.org*, 2016. [Online]. Available: https://nodered.org/. [Accessed: 03- Nov- 2016].

[28]"IBM - WebSphere MQ Telemetry", *Www-03.ibm.com*, 2016. [Online]. Available: http://www-03.ibm.com/software/products/en/wmq-telemetry. [Accessed: 03- Nov- 2016].

[29]"MQTT", *Mqtt.org*, 2016. [Online]. Available: http://mqtt.org/. [Accessed: 03- Nov- 2016].

[30]"Serverless Architectures", *martinfowler.com*, 2016. [Online]. Available: http://martinfowler.com/articles/serverless.html. [Accessed: 03- Nov- 2016].

[31]"OpenWhisk", *OpenWhisk*, 2016. [Online]. Available: https://developer.ibm.com/openwhisk/. [Accessed: 03- Nov- 2016].

[32]"Learn REST: A Tutorial", *Rest.elkstein.org*, 2016. [Online]. Available: http://rest.elkstein.org/. [Accessed: 03- Nov- 2016].

[33]"IBM Graph - API", *Ibm-graph-docs.ng.bluemix.net*, 2016. [Online]. Available: https://ibm-graph-docs.ng.bluemix.net/api.html. [Accessed: 15- Oct- 2016].

[34]"IBM developerWorks : geospatial analytics service", *Ibm.com*, 2016. [Online]. Available: http://www.ibm.com/developerworks/topics/geospatial__analytics__service/index.html. [Accessed: 03- Nov- 2016].

[35]"IBM Bluemix - DevOps", *Ibm.com*, 2016. [Online]. Available: https://www.ibm.com/cloud-computing/bluemix/devops. [Accessed: 03- Nov- 2016].

[36]"IBM Knowledge Center", *Ibm.com*, 2016. [Online]. Available: https://www.ibm.com/support/knowledgecenter/SS4GSP_6.1.3/com.ibm.udeploy.admin.doc/topics/security_ch.html. [Accessed: 03- Nov- 2016].

[37]"IBM Bluemix Docs", *Console.ng.bluemix.net*, 2016. [Online]. Available: https://console.ng.bluemix.net/docs/security/index.html#security. [Accessed: 03- Nov- 2016].

[38]"Load Impact - IBM Bluemix", *Console.ng.bluemix.net*, 2016. [Online]. Available: https://console.ng.bluemix.net/catalog/services/load-impact. [Accessed: 03- Nov- 2016].

[39]"Auto-Scaling - IBM Bluemix", *Console.ng.bluemix.net*, 2016. [Online]. Available: https://console.ng.bluemix.net/catalog/services/auto-scaling. [Accessed: 03- Nov- 2016].

[40]"About SoftLayer | Corporate Details and Information", *Softlayer.com*, 2016. [Online]. Available: http://www.softlayer.com/about-softlayer. [Accessed: 03- Nov- 2016].

[41]"About Trello", *Trello.com*, 2016. [Online]. Available: https://trello.com/about. [Accessed: 03- Nov- 2016].

[42]"Toggl - Free Time Tracking Software", *Toggl.com*, 2016. [Online]. Available: https://toggl.com/about. [Accessed: 03- Nov- 2016].

[43]"Build software better, together", *GitHub*, 2016. [Online]. Available: https://github.com/about. [Accessed: 03- Nov- 2016].

[44]T. Flynn, "16117743/INS-Thesis-Documentation", *GitHub*, 2016. [Online]. Available: https://github.com/16117743/INS-Thesis-Documentation/. [Accessed: 04- Nov- 2016].