World Cities and Places - A Microservices Application

Abstract goes here

Donald Knuth

Dennis Ritchie

B.Sc.(Hons) of Science in Computing in Software Development

May 2, 2020

Final Year Project

Advised by: Dr Alan Turing
Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)

About this project

Abstract A brief description of what the project is, in about two-hundred and fifty words.

Authors Explain here who the authors are.

Contents

List of Figures

Chapter 1

Introduction

The introduction should be about three to five pages long. Make sure you use references [2]
and more [3]
and the last [4]

Chapter 2

Context

- Provide a context for your project.
- Set out the objectives of the project
- Briefly list each chapter / section and provide a 1-2 line description of what each section contains.
- List the resource URL (GitHub address) for the project and provide a brief list of the main elements at the URL.

2.1 Filler

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam mi enim, interdum ut elit lobortis, bibendum tempus diam. Etiam turpis ex, viverra tristique finibus nec, feugiat at metus. Curabitur tempus gravida interdum. Donec ac felis a lorem scelerisque elementum. Vestibulum sit amet gravida tortor, a iaculis orci. Nam a molestie augue. Curabitur malesuada odio at mattis molestie. In hac habitasse platea dictumst. Donec eu lectus eget risus hendrerit euismod nec at orci. Praesent porttitor aliquam diam, eu vestibulum nisl sollicitudin vel. Nullam sed egestas mi.

Quisque vel erat a justo volutpat auctor a nec odio. Sed rhoncus augue sit amet nisl tincidunt, vitae cursus tellus efficitur. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque et auctor dui. Fusce ornare odio ipsum, et laoreet mi molestie sed. Cras at massa sit amet ipsum gravida aliquam. Nulla suscipit porta imperdiet. Fusce eros neque, bibendum sit amet consequat non, pulvinar quis ipsum.

2.1.1 More filler

Donec fermentum sapien ac rhoncus egestas. Nullam condimentum condimentum eros sit amet semper. Nam maximus condimentum ligula. Praesent faucibus in nisi vitae tempus. Sed pellentesque eleifend ante, ac malesuada

nibh dapibus nec. Phasellus nisi erat, pulvinar vel sagittis sed, auctor et magna. Quisque finibus augue elit, consequat dignissim purus mollis nec. Duis ultricies euismod tortor, nec sodales libero pellentesque et. Interdum et malesuada fames ac ante ipsum primis in faucibus.

Donec id interdum felis, in semper lacus. Mauris volutpat justo at ex dignissim, sit amet viverra massa pellentesque. Suspendisse potenti. Praesent sit amet ipsum non nibh eleifend pretium. In pretium sapien quam, nec pretium leo consequat nec. Pellentesque non dui lacus. Aenean sed massa lacinia, vehicula ante et, sagittis leo. Sed nec nisl ac tellus scelerisque consequat. Ut arcu metus, eleifend rhoncus sapien sed, consequat tincidunt erat. Cras ut vulputate ipsum.

Curabitur et efficitur augue. Proin condimentum ultrices facilisis. Mauris nisi ante, ultrices sed libero eget, ultrices malesuada augue. Morbi libero magna, faucibus in nunc vitae, ultricies efficitur nisl. Donec eleifend elementum massa, sed eleifend velit aliquet gravida. In ac mattis est, quis sodales neque. Etiam finibus quis tortor eu consequat. Nullam condimentum est eget pulvinar ultricies. Suspendisse ut maximus quam, sed rhoncus urna.

2.2 Filler

Phasellus eu tellus tristique nulla porttitor convallis. Vestibulum ac est eget diam mollis consectetur. Donec egestas facilisis consectetur. Donec magna orci, dignissim vel sem quis, efficitur condimentum felis. Donec mollis leo a nulla imperdiet, in bibendum augue varius. Quisque molestie massa enim, vitae ornare lacus imperdiet non. Donec et ipsum id ante imperdiet mollis. Nullam est est, euismod sit amet cursus a, feugiat a lectus. Integer sed mauris dolor.

Mauris blandit neque tortor, consequat aliquam nisi aliquam vitae. Integer urna dolor, fermentum ut iaculis ut, semper eu lacus. Curabitur mollis at lectus at venenatis. Donec fringilla diam ac risus imperdiet suscipit. Aliquam convallis quam vitae turpis interdum, quis pharetra lacus tincidunt. Nam dictum maximus lectus, vitae faucibus ante. Morbi accumsan velit nec massa tincidunt porttitor. Nullam gravida at justo id viverra. Mauris ante nulla, eleifend vitae sem vitae, porttitor lobortis eros.

Cras tincidunt elit id nisi aliquam, id convallis ex bibendum. Sed vel odio fringilla, congue leo quis, aliquam metus. Nunc tempor vehicula lorem eu ultrices. Curabitur at libero luctus, gravida lectus sed, viverra mi. Cras ultrices aliquet elementum. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Sed metus ante, suscipit sit amet finibus ut, gravida et orci. Nunc est odio, luctus quis diam in, porta molestie magna. Interdum et malesuada fames ac ante ipsum primis in faucibus. Mauris pulvinar lacus odio, luctus tincidunt magna auctor ut. Ut fermentum nisl rhoncus, tempus nulla eget, faucibus tortor. Suspendisse

eu ex nec nunc mollis pulvinar. Nunc luctus tempus tellus eleifend porta. Nulla scelerisque porttitor turpis porttitor mollis.

Duis elementum efficitur auctor. Nam nisi nulla, fermentum sed arcu vel, posuere semper dui. Fusce ac imperdiet felis. Aenean quis vestibulum nisl. Integer sit amet tristique neque, at suscipit tortor. Morbi et placerat ante, vel molestie dui. Vivamus in nibh eget massa facilisis accumsan. Nunc et purus ac urna fermentum ultrices eget sit amet justo. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Cras elementum dui nunc, ac tempor odio semper et. Ut est ipsum, sollicitudin eleifend nisl eu, scelerisque cursus nunc. Nam at lectus vulputate, volutpat tellus vel, pharetra mauris. Integer at aliquam massa, at iaculis sem. Morbi nec imperdiet odio. In hac habitasse platea dictumst.

Mauris a neque lobortis, venenatis erat ut, eleifend quam. Nullam tincidunt tellus quis ligula bibendum, a malesuada erat gravida. Phasellus eget tellus non risus tincidunt sagittis condimentum quis enim. Donec feugiat sapien sit amet tincidunt fringilla. Vivamus in urna accumsan, vehicula sem in, sodales mauris. Aenean odio eros, tristique non varius id, tincidunt et neque. Maecenas tempor, ipsum et sollicitudin rhoncus, nibh eros tempus dolor, vitae dictum justo massa in eros. Proin nec lorem urna. In ullamcorper vitae felis sit amet tincidunt. Maecenas consectetur iaculis est, eu finibus mi scelerisque et. Nulla id ex varius, ultrices eros nec, luctus est. Aenean ac ex eget dui pretium mattis. Ut vitae nunc lectus. Proin suscipit risus eget ligula sollicitudin vulputate et id lectus.

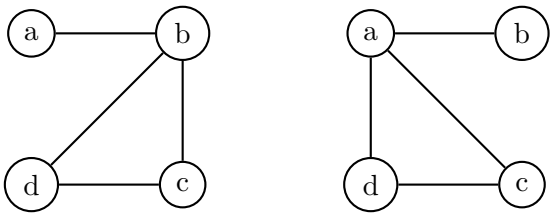
Chapter 3

Methodology

About one to two pages. Describe the way you went about your project:

- Agile / incremental and iterative approach to development. Planning, meetings.
- What about validation and testing? Junit or some other framework.
- If team based, did you use GitHub during the development process.
- Selection criteria for algorithms, languages, platforms and technologies.

Check out the nice graphs in Figure 3.2, and the nice diagram in Figure **??**.



G_1 G_2

Figure 3.1: Nice pictures

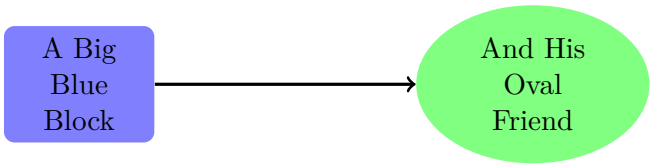


Figure 3.2: Nice pictures

Chapter 4

Technology Review

4.1 Ahutentication Service Database

We need a database on where the data can be accessed quickly and reliably by index. We are also looking for a database that can be replicated. A relational SQL database fits the requirements. From MariaDB, MySQL, PostgreSQL, and oracle. MySQL and PostgresSQL have been selected for further analysis.

4.1.1 MySql

MySQL 5.6 [5]

Pros

- Oracle has brought more investment into MySQL, meaning there is a future with it.
- It is a solid product. MySQL 5.6 is a reliable product with all the features fully tested.
- There are many teams from Oracle working in MySQL.

Cons

- Not so mature as other relations databases like PostgreSQL. This means that it has fewer features than the other more mature database systems.
- Not fully open source anymore, in theory, is open source, but in practice, Oracle has taken over.
- Many have replaced MySQL for MariaDB. Since oracle takes over MySQL, many big names like RedHat Enterprise Linux, Fedora, have moved to MariaDB.

4.1.2 PostgreSQL [1]

Pros

- Reach libraries for transactions. Fully documented with comments made it easy to know what part of the code does what and how it is done.
- Many adjustable parameters make the system easier to personalize.
- Easy to extend, if extra features are needed is possible to add the feature. Secure and reliable, also security can be personalized and extendible.

Cons

- Open source, not owned for any organization that takes care of it.
- Slow performance, there have been reported issues with performance and backup recovery.

Conclusion

There is not a real need for a feature-rich database; the database that we need to implement is simple, and the main aspect to consider is the replication—also, some of the cons of MySQL that be considered as advantages. Because MySQL is robust and has all the functionality needed, it has been chosen as the best option.

About seven to ten pages.

- Describe each of the technologies you used at a conceptual level. Standards, Database Model (e.g. MongoDB, CouchDB), XMl, WSDL, JSON, JAXP.
- Use references (IEEE format, e.g. [1]), Books, Papers, URLs (timestamp) – sources should be authoritative.

4.2 XML

Here's some nicely formatted XML:

```
<this>
  <looks lookswhat="good">
    Good
  </looks>
</this>
```

Chapter 5

System Design

5.1 Ahuthentication Service

This service provides user authentication. It is composed of three components: the hash service, the database, and the main service (Figure 5.1).

The password is store securely, is hashed and salted using the hash service and then the hash and salt is stored in the database.

The database is replicated using the master follower topology. Create, update, and delete operations are always performed in the master, read operations are performed in followers.

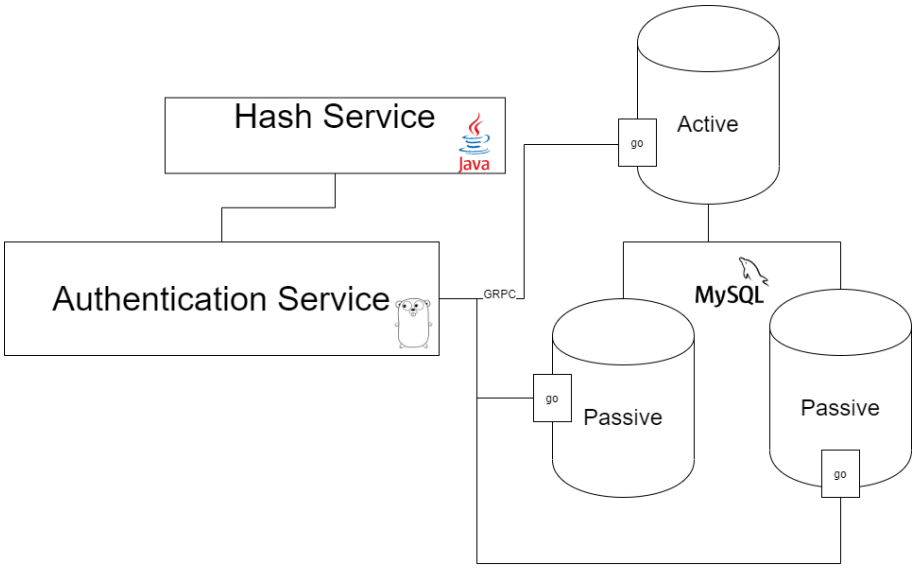


Figure 5.1: Authentication Service- UML.

5.1.1 Database Replication

Replication it has been set up using MySQL 5.7 (See Appendix B.1.4), we have set up a master and a replicate running in different Virtual Machine using Azure services. Using the instruction in Appendix B.1.4 is possible to add more followers, for do tables in master need to be locked for a few minutes.

The load balancing is done using Round Robin, and standards go grpc librarie. helps in performance because the most used operation is to get user data or token to check authentication.

Is implemented as client-side replication, the client chooses a server one at the time to make the calls, the client in this situation is the authentication service, and the server is the authentication dba. The load balancing is implemented in the authentication service(Figure 5.3).

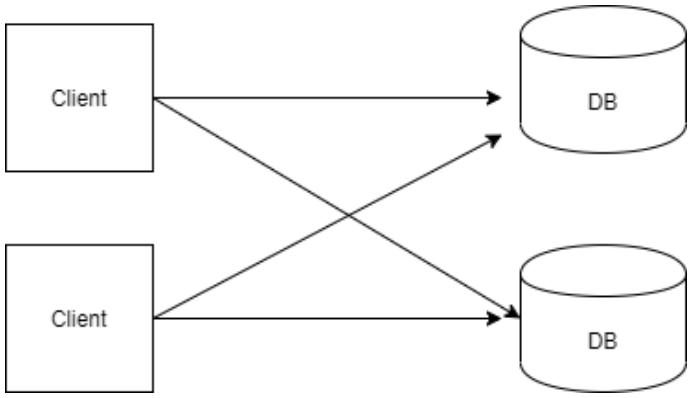


Figure 5.2: Authentication Service- Create User Sequence Diagram.

5.1.2 Endpoints

Create User

Users create an account, and when this happens, a new entry is created in the authentication database. Also, a new entry is created in the profiles database (Figure 5.3). To create a user, this service needs to communicate with the profiles service.

Login user

To login a user, we perform the followings steps (Figure 5.4):

- Get user data from the authentication database.
- Check the password using the hash service.

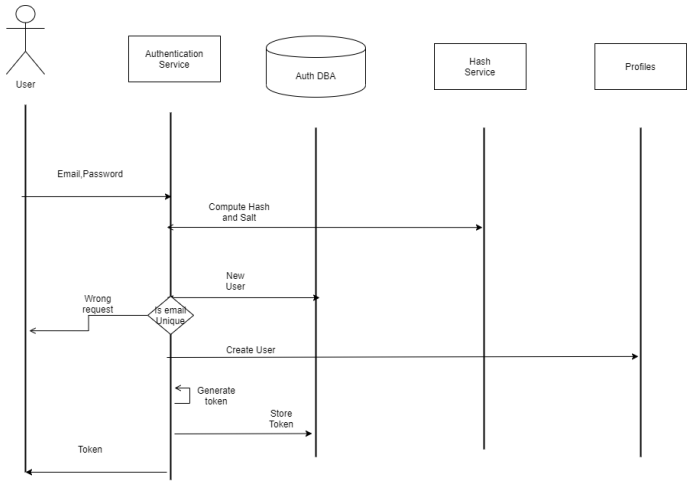


Figure 5.3: Authentication Service- Create User Sequence Diagram.

- If the password is valid will generate a unique token, it stores the token in the database and sends the response to the client.

Check Token

The most used endpoint, here is where replication plays a role for fast checking tokens. This is used in the most requests to all services to ensure security across the application (Figure 5.5).

Log out

The request includes the token, and that token is removed from the sessions table in the authentication database.

5.1.3 Authentication DBA

This program provides access to the authentications database. Is written in go and runs in a docker container, it connects to a MySQL database running in localhost. The application communicates with the main authentication service through a grpc interface.

The database

The authentication database store the necessary user data for authentication and login.

It is composed of two tables: the authentication table and the sessions table. The authentication table contains the user name, the password hash, and the password salt(Figure 5.6).

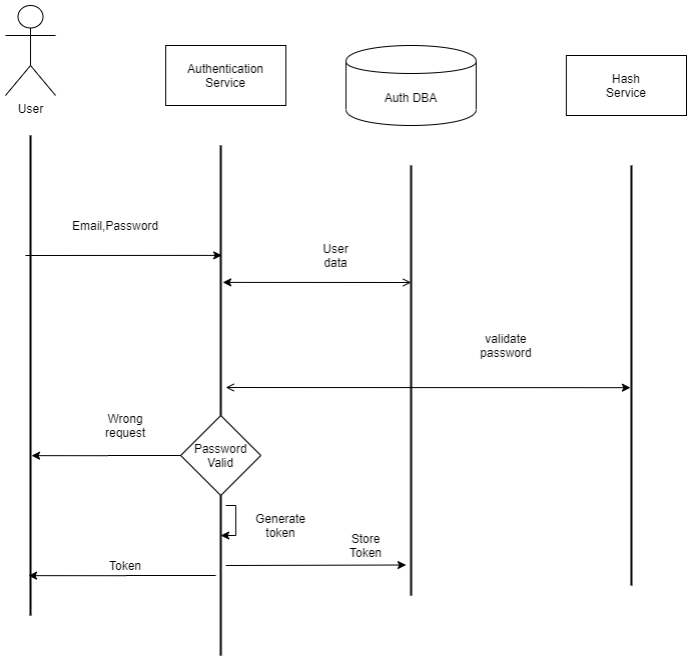


Figure 5.4: Authentication Service- Login Sequence Diagram.

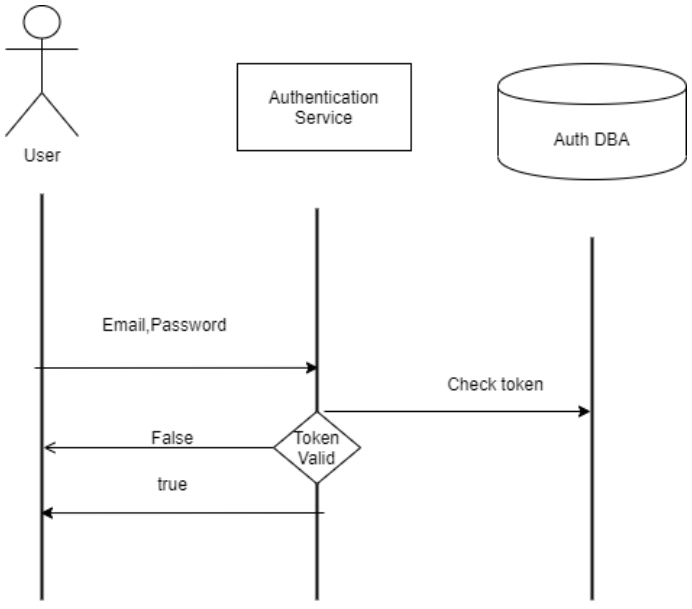


Figure 5.5: Authentication Service- Check Token Sequence Diagram.

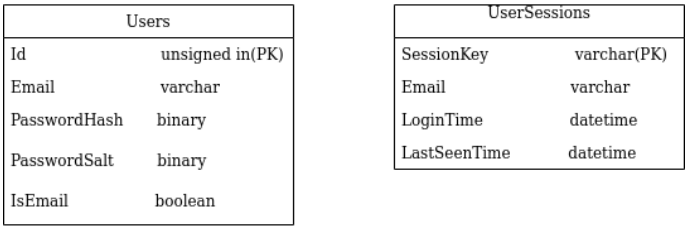


Figure 5.6: Authentication DBA- Authentication Database.

The user name is the email and is the unique identifier for all the systems, is not the primary key of the database, but is indexed for quick access. For this database, an extra integer field is added as the primary key. The password hash and salt are 32 bytes binaries strings. The sessions table uses a unique session key as a primary key for a quick check if a session exists.

When a user login a session is created and stored in this table. The user then can log in to the application using that session. When the user logs out, the session is deleted from the database.

Endpoints

- AddUser() : Create a new user in the database. When the user creates the account, it will create the profile automatically in the profiles database.
- GetUser(): Returns the user data used to authenticate the user.
- UpdateUser(): Update the user data, is used for changing the password.
- CreateSeassion(): Create a new session in the database. Used when the user login using the password.
- GetSeassion() : Return a user session if exist. Used to check if the session exists so the user can log in without the password.
- DeleteSession : Delete user session if exits. Used when the user logs out from the device.

5.1.4 The Hash Service

This service creates a password salted password hash using a randomly generated salt. It has been adapted from a Distributed Systems project from semester 7. It checks the password in fix amount of time for security reasons. Attackers can guess passwords guess by comparing the time it takes to validate a password.

5.2 Profiles Service

As many pages as needed.

- Architecture, UML etc. An overview of the different components of the system. Diagrams etc... Screen shots etc.

Column 1	Column 2
Rows 2.1	Row 2.2

Table 5.1: A table.

Chapter 6

System Evaluation

As many pages as needed.

- Prove that your software is robust. How? Testing etc.
- Use performance benchmarks (space and time) if algorithmic.
- Measure the outcomes / outputs of your system / software against the objectives from the Introduction.
- Highlight any limitations or opportuni-ties in your approach or technologies used.

Chapter 7

Conclusion

About three pages.

- Briefly summarise your context and ob-jectives (a few lines).
- Highlight your findings from the evalua-tion section / chapter and any opportuni-ties identified.

Bibliography

[1] S. Dhruv, "Pros and cons of using postgresql for application development." `https://www.aalpha.net/blog/pros-and-cons-of-using-postgresql-for-application-development/`. [Online; accessed 12-Novemer-2019].

[2] A. Einsteion, "Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]," *Annalen der Physik*, vol. 322, no. 10, pp. 891–921, 1905.

[3] M. Goossens, F. Mittelbach, and A. Samarin, *The LaTeX Companion*. Reading, Massachusetts: Addison-Wesley, 1993.

[4] D. Knuth, "Knuth: Computers and typesetting."

[5] L. Beatty, "The pros and cons of mysql." `https://www.smartfile.com/blog/the-pros-and-cons-of-mysql/`. [Online; accessed 12-Novemer-2019].

Appendices

Appendix A

Docker

A.1 Install Docker in Ubuntu Using Command Line

A.1.1 Setup repository

- Update packages

```
$ sudo apt-get update
```

- Install packages to allow apt to use a repository over HTTPS:

```
$ sudo apt-get install \
  apt-transport-https \
  ca-certificates \
  curl \
  gnupg-agent \
  software-properties-common
```

- Add Docker's official GPG key :

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg
↪  | sudo apt-key add -software-properties-common
```

- Verify that you now have the key with the fingerprint 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88, by searching for the last 8 characters of the fingerprint.

```
$ sudo apt-key fingerprint 0EBFCD88
```

- set up the stable repository:

```
$ sudo add-apt-repository \
  "deb [arch=amd64]
  ↪ https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) \
  stable"
```

A.1.2 Install Docker Community

- Install the latest version of Docker Engine - Community and container, or go to the next step to install a specific version:

```
  $ sudo apt-get install docker-ce docker-ce-cli
  ↪ containerd.io
```

- Verify that Docker Engine - Community is installed correctly by running the hello-world image.

```
  $ sudo docker run hello-world
```

A.2 Run Image Using Docker Hub

- Create repository in Docker Hub.

  https://docs.docker.com/docker-hub/repos/

- Build Image (Local machine):

```
$ sudo docker image build -t
↪ docker-hub-user-name/image-name:version-tag .
```

  Example:

```
$ sudo docker image build -t joseretamal/hash-service:1.0
↪ .
```

- Push Image (Local machine):

```
$ sudo docker push
↪ docker-hub-user-name/image-name:version-tag
```

  Example:

```
$ sudo docker push  joseretamal/hash-service:1.0
```

- Pull Image (Remote machine):

```
$ sudo docker pull
↪ docker-hub-user-name/image-name:version-tag
```

Example:

```
$ sudo docker pull joseretamal/hash-service:1.0
```

- Run image (Remote machine):

  Opening a port and restart on crash or reboot:

```
$ sudo docker run -d -p internal-port:open-port --restart
↪  always --name instance-name
↪  user-name/image-name:version-tag
```

Example:

```
$ sudo docker run -d -p 5151:5151 --restart always --name
↪  hash-service joseretamal/hash-service:1.0
```

  Allowing instance to full network acess (allows acess to local host):

```
$ sudo docker run -d -p  --network="host" --restart
↪  always --name instance-name
↪  user-name/image-name:version-tag
```

Example:

```
$ sudo docker run -d -p  --network="host" --restart
↪  always --name hash-service
↪  joseretamal/hash-service:1.0
```

- Stop instance: (Remote machine):

```
$ sudo docker rm --force instance-name
```

Example:

```
$ sudo docker rm --force hash-service
```

- Check logs: (Remote machine):

```
$ sudo docker logs instance-name
```

Example:

```
$ sudo docker logs hash-service
```

- Bash into the container: (Remote machine):

```
$ sudo docker exec -it instance-name bash
```

Example:

```
$ sudo docker exec -it hash-service bash
```

Appendix B

MySql

B.1 Install Mysql in Linux Using Command Line

B.1.1 Install MySQL-shell

- Make sure you do not skip the step for updating package information for the MySQL APT repository:

```
$ sudo apt-get update
```

- Install MySQL Shell with this command:

```
$ sudo apt-get install mysql-shell
```

B.1.2 Install MySql server

```
$ sudo apt-get install mysql-server
```

B.1.3 Uninstall MySql server

```
$ sudo apt-get remove --purge mysql*
$ sudo apt-get purge mysql*
$ sudo apt-get autoremove
$ sudo apt-get autoclean
$  sudo apt-get remove dbconfig-mysql
$ sudo apt-get dist-upgrade
```

B.1.4 Setup Replication

https://www.digitalocean.com/community/tutorials/how-to-set-up-master-slave-replication-in-mysql

setup master

- Edit the mysql config file,for open the file using vi:

```
$sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf
```

- make the followings changes to the the file, if the field are missing they must be added or if they are commented un commented:

```
server-id       = 1
log_bin                 = /var/log/mysql/mysql-bin.log
binlog_do_db  = replica1
sudo mysql_secure_installation
```

- Restart MySQL:

```
$ sudo service mysql restart
```

- Create user for replication and give permissions:

```
$ sudo mysql -u root
mysql>GRANT REPLICATION SLAVE ON *.* TO
↪  'slave_user'@'%' IDENTIFIED BY 'password';
FLUSH PRIVILEGES;
```

- Get master status, after select the database in one MySQL seasiion :

```
 mysql>FLUSH TABLES WITH READ LOCK;
```

- then open another MySQL seasion(keep the other open):
- Get master status, after select the database in one MySQL seasiion :

```
mysql>SHOW MASTER STATUS;
| File             | Position | Binlog_Do_DB |
+------------------+----------+--------------+--
| mysql-bin.0001580 |      154 | user_login   |
+------------------+----------+--------------+----
```

Note the file (mysql-bin-0001580) and the position.

- After take note of file name and position tables can be unlocked :

```
mysql>UNLOCK TABLES;
```

setup slave

- Edit slave config file :

```
$ sudo vi /etc/mysql/my.cnf
```

Make the following modifications:

```
server-id               = 2
relay-log               =
↪  /var/log/mysql/mysql-relay-bin.log
log_bin                 = /var/log/mysql/mysql-bin.log
binlog_do_db            = newdatabase
```

- Restart MySQL service :

```
$ sudo service mysql restart

```

- Config slave in mysql shell:

```
mysql> CHANGE MASTER TO
MASTER_HOST='104.40.206.141',
MASTER_USER='repl',
MASTER_PASSWORD='password',
MASTER_LOG_FILE='mysql-bin.000160',
MASTER_LOG_POS= 2439;
```

- Start slave

```
mysql> START SLAVE;
```

- Check status

```
mysql> SHOW SLAVE STATUS\G
```

Appendix C

Neo4J

C.1 Neo4j With Docker

C.1.1 Install

- Pull the latest image from docker hub(`https://hub.docker.com/_/neo4j/`):

```
$ sudo docker pull neo
```

- Run Neo4j:

```
$ sudo docker run \
--name neo4j \
-p7474:7474 -p7687:7687 \
-d \
-v $HOME/neo4j/data:/data \
-v $HOME/neo4j/logs:/logs \
-v $HOME/neo4j/import:/var/lib/neo4j/import \
-v $HOME/neo4j/plugins:/plugins \
--env NEO4J_AUTH=neo4j/test \
neo4j:latest
```

C.1.2 Access bash console:

- Access image bash:

```
$ sudo docker exec -it neo4j bash
```

- Access neo4j bash:

```
$ cypher-shell -u neo4j -p test
```