QA Consulting DevOps February 2020

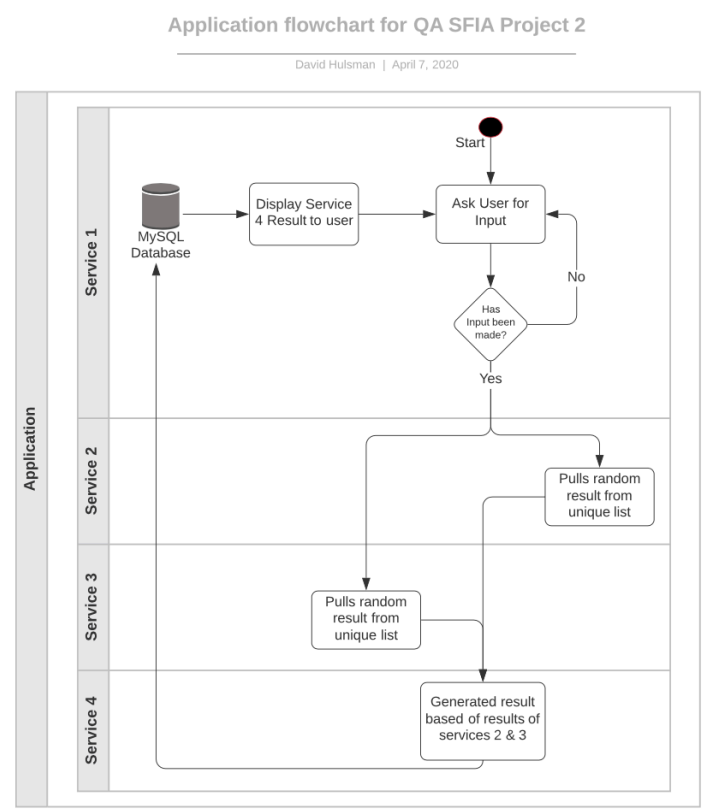
SFIA Project 2 – Rapper Name Generator

By David Hulsman

1. Project Brief

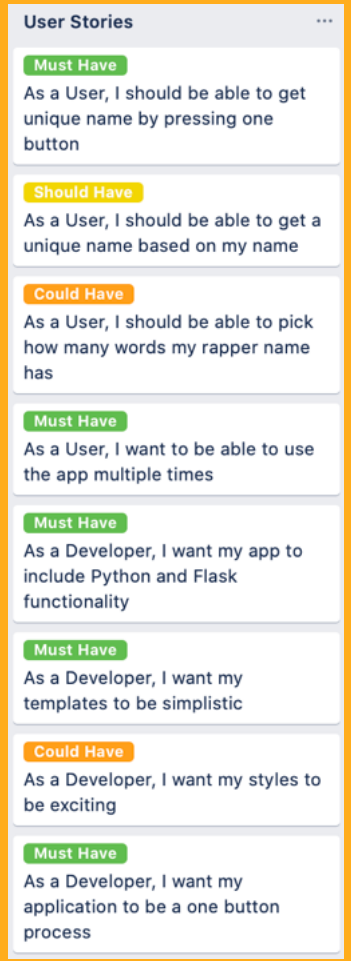
- Create a micro-service orientated architecture for a application:
  - Service 1: Used to render Jinja2 templates for interaction, connect to the other services, and finally persisting some data into a SQL Database
  - Service 2/3: Used to generate 'Objects'
  - Service 4: Used to create an 'Object' based of the results of Services 2 and 3 using interesting logic
- 2 Different implementations between Services 2, 3 and 4
- Deployment using containerisation and a orchestration tool
- Advanced CI practices using Jenkins and Ansible
- Refined skills of services and tools used in the previous project

Create a micro-service orientated architecture for a application



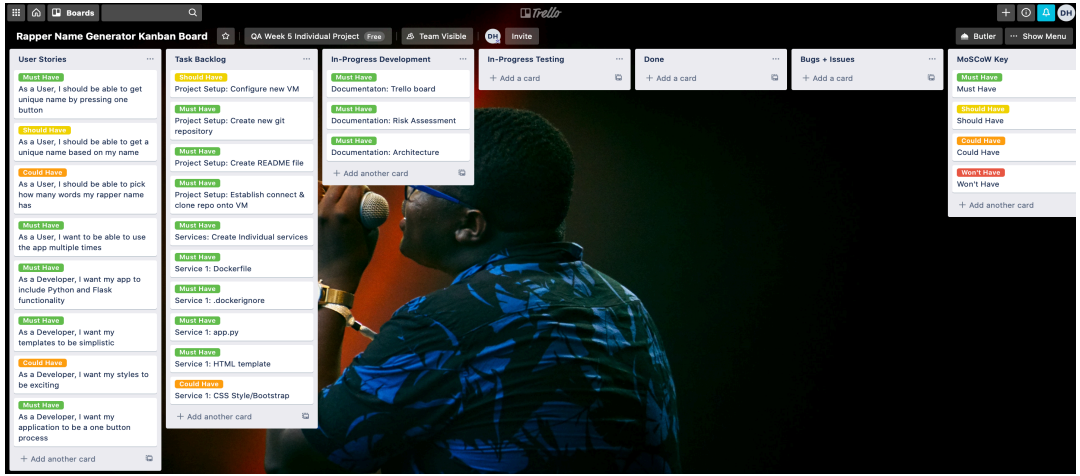
My Solution

- Simple light-hearted rapper name generator
- Less focus on app functionality/styling
- More focus on the use of multiple nodes, networks and objects
- SQL and NGINX implemented from the start
- Further use of CI/CD Practices and building on feedback from previous project
- Target MVP and plan for further improvements if possible



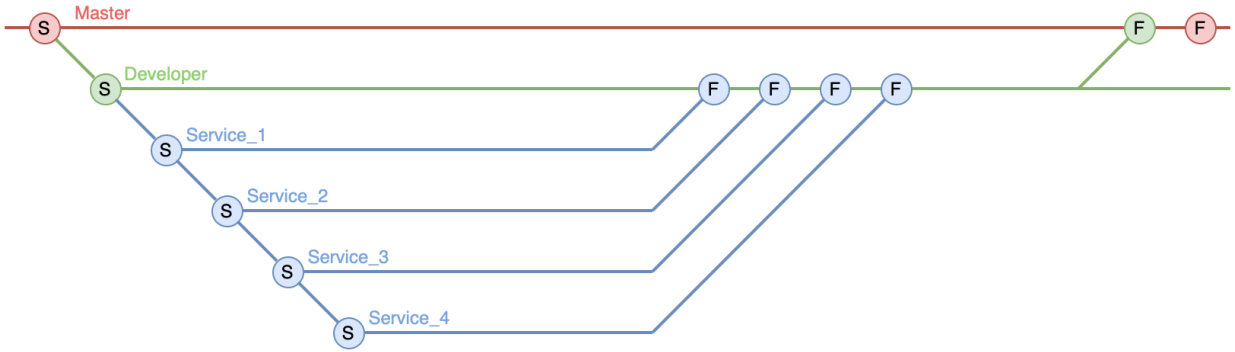
My Solution - Kanban

- Used Trello yet again as I now understand its principles
- Use of MoSCoW prioritisation again to identify important MVP related tasks



My Solution – Feature Branch

- Build upon past experience with feature branch model
- Include branches for each service that will push into developer once completed
- Delete old branches to streamline development
- More active commit cycle, over 140 commits



My Solution – SQL Design

- Very simple design that allows for basic insert and read functions
- Includes full validation of columns and PRIMARY KEY
- Allows for easy scalability and addition of new tables

Entity Relationship Diagram for QA SFIA Project 2

David Hulsman | April 6, 2020

Entities		
PK	id	INT(3)
	name_old	VARCHAR(28)
	name_new	VARCHAR(28)

MoSCoW Method Key
Must Have (Green)
Should Have (Yellow)
Could Have (Orange)
Won't Have (Red)

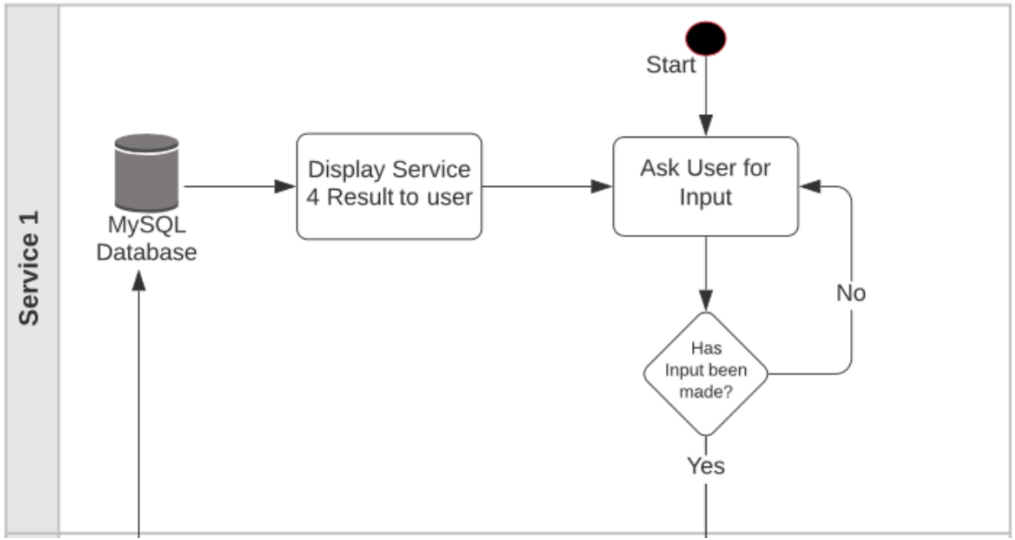
Risk Assessment

1.1.3	Change in project specification	depending on the lateness of a change, as well as the serious of said change, this will massive effect the project from all aspects from architecture, to methodologies used. **This in combination with other risks can cause massive issues**	Change in requirements from internal/external sources.	1	5	5
Project Development						
1.2.1	Poor accessibility to appropriate software/ hardware	Poor or no access to related hardware and/or software can effect both the accessibility of the project. files, as well as the progression towards expected milestones.	Limited access to sufficient machines, little to no cloud storage capability, downtime on related services and privacy concerns.	1	2	2
1.2.2	**Poor internet access/connection**	**Poor internet access or access can prohibate my acess to learning content, VM access and video meetings, which can impact various areas from project progression, relationship management and learning potential**	**Poor location regarding connection, low bandwidth in household, wireless connection as appose to wired.**	4	4	16

Full versions of original risk assessment and revised version available at https://github.com/HavidDulsman/RapperNameGenerator/tree/master/documentation

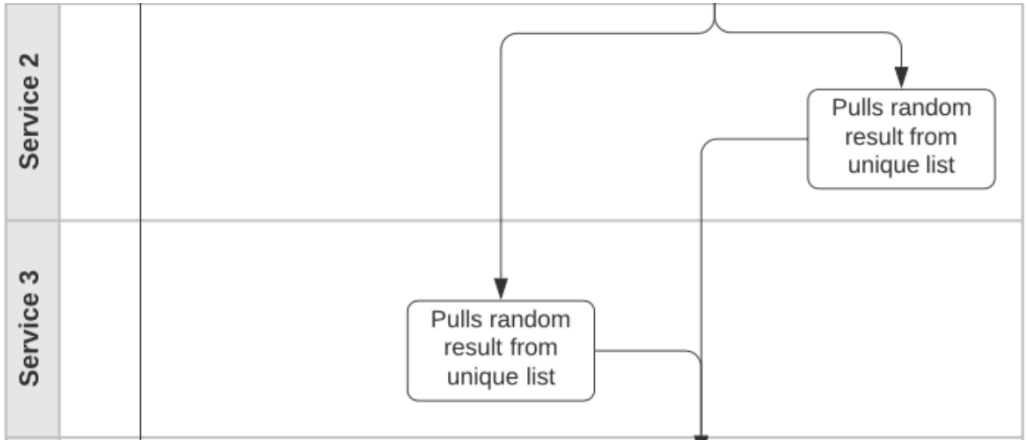
Service 1: Frontend Application

- Service responsible for the frontend interface
- Only service users will directly interact with.
- Linked directly with SQL Database via form
- Displays result from Service 4 as output.
- Utilises port 5000



Service 2 & 3:

- Utilise port 5001 & 5002 respectively
- Contain similar functions what take a random word from unique CSV files
- Once completed, results can be referenced in later service



service_2 > application > first.txt

```
1   Lil,Big,The,Yung,Da,Dockta,Hot,Eazy,Almighty,Ol Dirty,Ghostface,Ch
```

Service 4:

- Utilises port 5003
- References functions in services 2 and 3 to create unique rapper name
- End result can then be referenced in service 1
- Result also imputed into SQL database.


```python
@app.route('/generated3', methods=['GET'])
def generated3():
    firstname = requests.get('http://part_one:5001/generated1')
    secondname = requests.get('http://part_two:5002/generated2')
    response = firstname.text + " " + secondname.text
    print(response)
    return response
```

- Used for the containerization of the services used in this application
- Docker Swarm allows me to create a network of 2 or more nodes to run a application together
- Docker Stack allows me to deploy containers and images onto old or new machines, allowing them to host the services

```
version: '3.7'
services:
    front_end:
        image: dhulsman/service_1:latest
        environment:
            - MYSQLHOST=${MYSQLHOST}
            - MYSQLUSER=${MYSQLUSER}
            - MYSQLPASSWORD=${MYSQLPASSWORD}
            - MYSQLDB=${MYSQLDB}
        deploy:
            replicas: 3
        networks:
            - service_network
    part_one:
        image: dhulsman/service_2:latest
        deploy:
            replicas: 3
        networks:
            - service_network
    part_two:
        image: dhulsman/service_3:latest
```

- Allows for the automation of software and application deployment
- My application uses a playbook to install the necessary packages and tools onto Manager and Worker Nodes

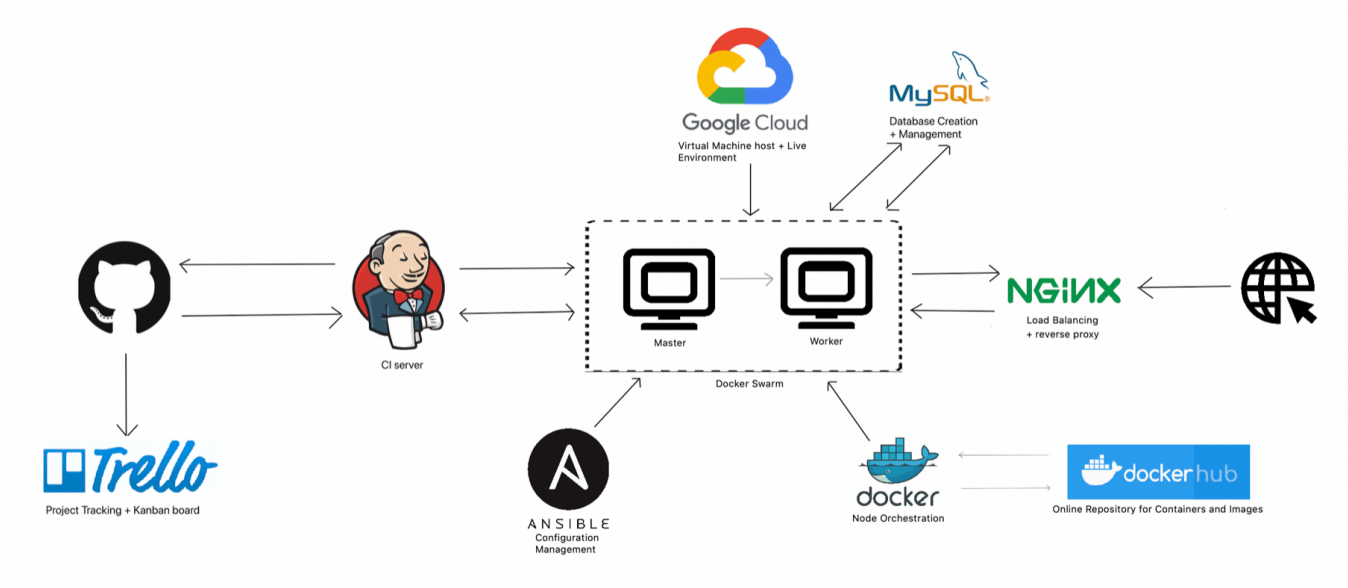
```
export ANSIBLE_HOST_KEY_CHECKING=False

[servers]
35.246.119.11 ansible_user=jenkins ansible_ssh_private_key_file=/var/lib/jenkins/.ssh/id_rsa ansible_python_interpreter=/usr/bin/python3

[node]
35.246.119.11 ansible_user=jenkins ansible_ssh_private_key_file=/var/lib/jenkins/.ssh/id_rsa ansible_python_interpreter=/usr/bin/python3
```

NGINX

- NGINX acts as a reverse proxy for my services, allowing me to hide the ports from public viewing, adding a layer of security to the app
- NGINX was implemented from the start of the application
- A unique NGINX service is pulled from my dockerhub account and used in configuration with my other service images/containers

```
FROM nginx:latest
WORKDIR /etc/nginx/
RUN rm ./conf.d/default.conf
COPY ./reverse_proxy.conf ./conf.d/
```

Development: CI Pipeline & used tools



Testing PyTest

```python
def test_node_manager():
    http = urllib3.PoolManager()
    r = http.request('GET', url)
    assert 200 == r.status
```

```
============================= test session starts ==============================
platform linux -- Python 3.6.9, pytest-5.4.1, py-1.8.1, pluggy-0.13.1
rootdir: /var/lib/jenkins/workspace/rapper-pipeline@2
collected 7 items

testing/test.py .......                                                  [100%]

============================== 7 passed in 0.39s ===============================
```

Testing Coverage

- Now includes a HTML Coverage sheet!

```
testing/test.py                                                                                                                    51
0    100%
-------------------------------------------------------------------------------------------------------------------------------------
---------------------
TOTAL                                                                                                                           41907
25829     38%
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Improving on previous project feedback

- Efforts where made to improve testing coverage and functionality
- Inclusion of Risk Assessment Matrix, as well as evolution of risk assessment
- (Hopefully) better time management regarding presentation
- Further advancements in tools and services used in the previous project

Project retrospective

What went well

- Effective use of docker, swarm and stack
- NGINX Implementation from the start
- Automation on multiple levels
- Easy-to-setup for new nodes

```
upstream rapperapp {
    server  front_end:5000;
}
server {
    listen  80;
    server_name localhost;

    location / {
        proxy_pass http://rapperapp;
    }
}
```

Project retrospective

What went wrong

- Poor Project Scope
- Poor testing continues...
- Inconsistent security measures, especially for new nodes

```
[servers]
35.246.119.11 ansible_user=jenkins ansible_ssh_private_key_file=/var/lib/jenkins/.ssh/id_rsa ansible_python_interpreter=/usr/bin/python3
```

Project retrospective

Future Improvements

- More use of nodes
- CRUD Implementation
- Selenium or other program assisted testing

Thanks!

Now is time for demonstration of my application

Find me on GitHub at @

By David Hulsman

*I do not own any of the logos or assets used in this PowerPoint