FRUITSHOP

ITONK Q4/2016

Adi Tahirovic, 10626
Bilal Kais, 11756
Khaled Saied, 11638
Rameez Syed, 10500

REPORT

May 2016

Advisor: Kasper Nissen and Martin Jensen

AARHUS
UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

FRUITSHOP

ADI TAHIROVIC, BILAL KAIS, KHALED SAIED, RAMEEZ SYED

ITONK Q4/2016

Report
Department of Engineering
Computer Engineering
Aarhus University

May 2016

Adi Tahirovic, Bilal Kais, Khaled Saied, Rameez Syed
Fruitshop, Report, © May 2016

ABSTRACT

Short summary of the contents in English...

SAMMENFATNING

Kort sammenfatning på dansk...

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

Part I

THE PROPER STRUCTURE OF A THESIS

The following reflects what I believe to be a good structure for a report or a thesis in experimental computer science. It contains a natural progression from the general to the specific, and from the work of others to the work by the authors, each chapter forming the foundation of the next.

1 INTRODUCTION

The Fruitshop is a webshop which consists of 4 different Spring boot services namely; Homepage, Basket, Search, and Contact service.

Homepage service
The Homepage service is the simplest service but could have big load of requests to it. Everyone entering the website needs to be navigated through the Homepage service. The Homepage service consists of a search bar and links to all the other services within the Fruitshop domain.

Basket service
The basket service is used to place and delete orders. The basket gives an overview of the products chosen to be bought. The user can press the pay button to pay for the chosen products and thereby complete the transaction.

Search service
The search service gives the user opportunity of searching the entire list of available products within the scope of Fruitshop. The end result of search service is a picture of the product which was looked up through the search service.

Contact service
The contact service gives a user the opportunity of contacting the Fruitshop via e-mail. The service is there to provide the user with helpful information in case anything goes wrong or in case of questions.

The main focus of this project is to understand, design & implement the following areas:

1. Spring boot service
2. Docker container
3. Kubernetes

The Fruitshop has spring boot services which are packed in Docker containers to make sure that the service runs inside the same environment and thereby is not dependent of underlying system and libraries.

Furthermore, Kubernetes is used to make the webshop more reliable, stable and fault tolerant. Kubernetes creates pods which contain replica of a services on different nodes within the cluster. Kubernetes keeps track of the pods, if a pod dies then it creates a new pod on one of the nodes and it also handles increase or decreases the number of replicated pods depending on the load on the webshop.

2

SPRING CLOUD AND BOOT

Report and prototype requirements - topics to be covered

Topic 1: Write a brief introduction to the Spring Boot and Cloud Frameworks. Design, develop, test, and document a functional proof-of-concept prototype using the Spring Boot and Cloud Framework. The prototype should be able to:

- be configured remotely, so that every client fetches configuration at boot-time from a central configuration server
- contain a simple graphical user interface
- leverage the gateway API pattern to distribute requests to specific services
- contain at least three data delivering services

Microservices is a new architectural style that aims to realize software systems as a package of small services, each deployable on different platform. Every single Microservice has its own process while communicating with other services via lightweight mechanisms like RESTFull APIs.

All the services has business capability which can utilize various programming languages and data stores. A system has a microservices architecture when that system is composed of several services without any centralized control.

Resilience to failure is another characteristic of microservices as every request in this new setting gets translated to several service calls through the system. To have a fully functional microservices architecture and to take advantage of all of its benefits, the following components have to be utilized. Most of these components address the complexities of distributing the business logic among the services:

- Configuration Server: It is one of the principles of Continuous Delivery to decouple source code from its configuration. It enables us to change the configuration of our application without redeploying the code. As a microservices architecture have so many services, and their re-deployment is going to be costly, it is better to have a configuration server so that the services could fetch their corresponding configurations.
- Service Discovery: In a microservices architecture, there exist several services that each of them might have many instances in

order to scale themselves to the underlying load. Thus, keeping track of the deployed services, and their exact address and port number is a cumbersome task. The solution is to use a Service Discovery component in order to get the available instances of each service.

3

DOCKER

Topic 2: Write a brief introduction to Docker containers. Design, develop, test, and document a functional proof-of-concept prototype using Docker containers. This builds on top of topic 1 and should be able to

- build docker containers of each service in topic 1
- run docker containers locally and on a Raspberry Pi
- deploy images to Docker Hub

Docker containers allows packaging of software applications with all its dependencies and libraries. It makes it easier to create, deploy and run the applications. The layered file Docker file system can be seen below in figure 1.

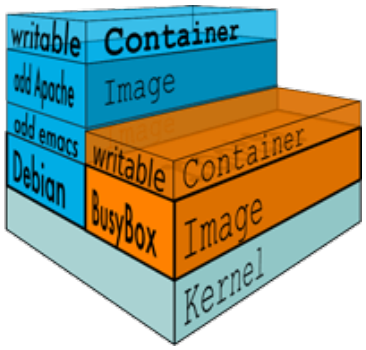
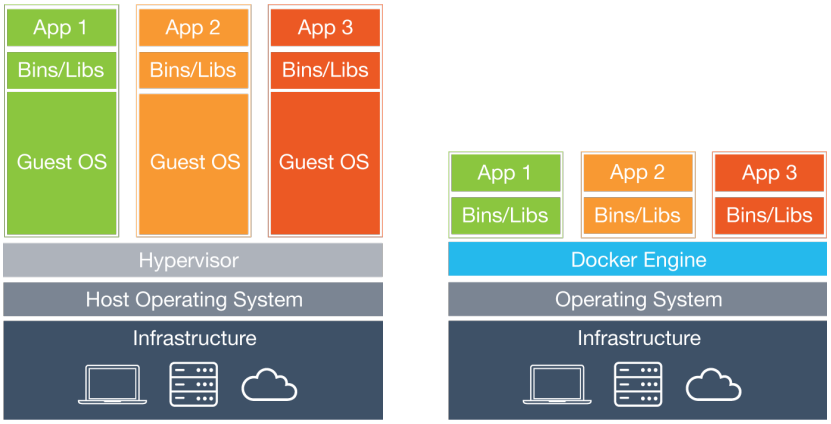


Figure 1: Overview of container architecture with complete file system

The main purpose of Docker is to ship the whole application with it's whole environment configuration as a single package so that it easily can run on other machines. By doing that the Docker ensures that the application will run on any other machine (Linux, OSX, Windows) regardless of any customized settings the particular machine might have.

Docker has resemblance to virtual machines in a way as it can be seen in figure 2, but Docker allows applications to use the same Linux kernel as the system that they're running on. By using Docker one can bypass creating virtual machine for every application. An algorithm can be run in docker using any Linux compatible language such as C, Python, Matlab etc. and be compiled using any Linux compatible libraries without causing version or library conflicts with other algorithms.



(a) Vm diagram (b) Docker diagram

Figure 2: Docker vs Vm

A docker container can be run at any time with the confidence of that the docker container's computational environment will be identical. The docker projects remains backwards compatible.

There 2 significantly different ways of building Docker containers:

1. Interactively
2. Dockerfile

Building a container interactively make it possible to install libraries and configure the environment from a shell just like in a typical Linux environment. The modifications can be saved via commits identical to Git, it's possible to track changes and see the status of modification.

On the other hand Dockerfile builds a container entirely through commands. A Dockerfile identifies a source container to start from (typically a basic Linux installation), then records a series of commands to install libraries and configure the environment. Dockerfiles can also load other Dockerfiles allowing for environment layering and concise organization of various software development projects. Dockerfiles can be versioned using a standard source control versioning tool like Git, allowing revision tracking and archiving of the computational environment.

3.0.1 *PoC of Docker container*

All the services in this project are packed using Docker containers. The Docker containers has all the necessary components and libraries to run a particular service.

In this project the Dockerfile technique is used to create Docker containers. [**Docker2014**]

Listing 1: Dockerfile with command to be carried out

```
From java:8
COPY /target/Homepage-0.0.1-SNAPSHOT.jar /data/
EXPOSE 8080
WORKDIR /data/
CMD ["java","-jar","Homepage-0.0.1-SNAPSHOT.jar"]
```

As it can be seen in the listing 1 the Dockerfile consists of commands which targets the files placed in the target folder.

```
COPY /target/Homepage-0.0.1-SNAPSHOT.jar /data/
```

The command line above specifies that the .jar file should be copied to the container

```
EXPOSE 8080
```

The command line above instructs Docker to expose the default Spring Boot port on port 8080.

```
WORKDIR /data/
```

The command line above instructs Docker to change the work directory to where the .jar file is placed.

```
CMD ["java","-jar","Homepage-0.0.1-SNAPSHOT.jar"]
```

The last command line instructs the container to run the .jar file.

After having done that its possible to build the image. The docker build command does all the heavy-lifting by creating docker image from Dockerfile. This is shown below:

```
$ docker build -t rh-th/Homepage .
```

After building the run command is used to run the image inside Docker container.

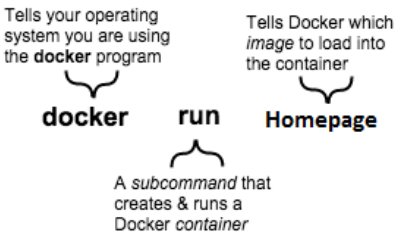


Figure 3: Container explainer

4 KUBERNETES

Topic 3: Write a brief introduction to the Kubernetes container cluster manager. Design, develop, test, and document a functional proof-of-concept prototype using the Kubernetes container cluster manager.

This topic builds on top of the previous two topics 1 and 2 and should be able to:
- deploy all containerized services in Kubernetes on the Raspberry Pi cluster
- scale pods in Kubernetes
- update pods with rolling-update

What is Kubernetes?
Kubernetes is an open-source container management system developed by Google. Kubernetes is used for automating management of containers such as scaling, loadbalancing and scheduling containers between nodes in a cluster. Containers are run inside something called a pod. A pod is a group of containers that are scheduled onto the same host. Pods are used for gathering containers that are tightly coupled to each other under the same host such communication between containers becomes easier. Every pod has it's own ip address. Kubernetes schedules, deploy and scale pods such they among other ensure that the load is balanced in the cluster.

The containers in the nodes are replicated, so if one node fails Kubernetes will create a new pod on another node and run the container within this newly created pod. If there suddenly comes a lot of load on a specific node it will make extra *replicaset* of the pods so that the load is distributed between these pods and thereby the load is more balanced.

4.1 POC OF KUBERNETES

5 RESILIENCE

Topic 4: Write a brief introduction to Resilience in microservice architectures and to how Gatling load testing framework can be used for testing. Run load testing scenarios against the handed out containers and describe the effects of using patterns. - run different load testing scenarios handed out - hand-in generated reports and key metrics on BlackBoard - hand in exercise report - run load-testing against your project architecture to see the effects

The word resilience mean "The capacity to recover quickly from difficulties". there are many definitions of the resilience, one of the definitions is taken from the book "The Resilience of Networked Infrastructure Systems". Fiskel (2003) defines a resilient system as a system that has the ability to return to a stable equilibrium state after a perturbation.

There are different categories of potential disruptions to system that create the need to implement the resilience in systems. In the figure 5 below the sources of disruptions is showed.

- Human Factor
  - UNDER
- Natural Factors
  - dfrgh
- Organizational factors
  - dfghdfsgh
- Technical Factors
  - dfrgh

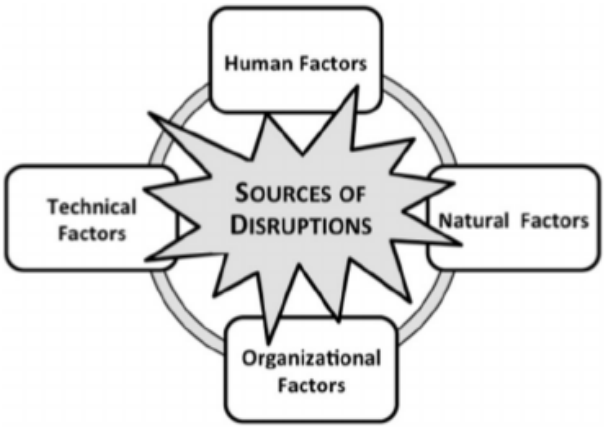


Figure 4: Sources of disruptions

6

DOMAIN NAME SERVICE AND SERVICE DISCOVERY

Topic 5: Write a brief introduction to Domain Name Service and Service Discovery and use these descriptions to put perspective on your report. The perspectives should include: - How may Domain Name Services be beneficial in cluster and cloud computing? - How may Service Discovery be beneficial in cluster and cloud computing?

6.1 DNS

The Domain Name Service (DNS) is a service that allows humans readable language converting to IP address. In other word DNS allows to type names into the Web browser like www.youtube.com and automatically find that address on the Internet (IP), instead of type the IP-address of the web site.

The DNS organizes its servers into a hierarchy figure 5 shows the hierarchy. For the Internet, so-called root name servers reside at the top of the DNS hierarchy. The root servers are responsible for holding information about all the top level domains, it is the starting point for every name lookup operation. Top level domains are divided into two groups:

- **Generic Top Level Domains (gTLD) .com, .edu, .net, .org, .mil etc.**
- **Country Code Top Level Domain (ccTLD) e.g. .us, .ca, .tv , .uk etc.**

Each ccTLD identifies a particular country and is two letters long.

At the domain level or user DNS you have the resource you are looking for.??

Recursive Query Vs Iterative Query in DNS

A recursive query is a kind of query, in which the DNS server, who received the query, will do the entire job fetching the answer, and giving it back to client. If DNS server is not able to resolve the requested query then it forwards the query to another DNS server until it gets an answer or the query fails. In an iterative query, the name server, will not go and fetch the complete answer for the query. If the queried DNS server doesn't have an exact match for the queried name, the best possible information it can return is a referral, which might have the answer. The DNS client can then query the DNS server for which it obtained a referral.

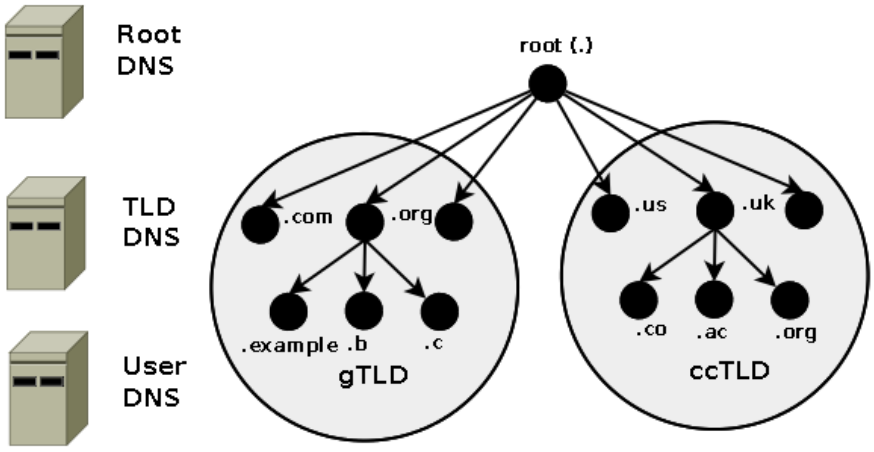


Figure 5: DNS hierarchy

Cached queries

DNS caching allows any DNS server or client to locally store the DNS records and re-use them in the future. The Ip and domain is stored in local cache called stub-resolver for a period. By that, the overall network usage is reduced and thereby higher efficiency is achieved. The number of packets sent out also gets reduced and thereby a lower latency.

How may Domain Name Services be beneficial in cluster and cloud computing?

By having redundant Domain Name Services, the concept of single point of failure is avoided, in the case of failure of a single or multiple DNS services. Hence by having redundant DNS in cloud computing it becomes fault tolerant.

To create additional resilience each root-server typically has multiple instances (copies) spread throughout the world. Each instance has the same IP address but data is sent to the closest instance using a process called anycasting.

6.2 SERVICE DISCOVERY

Service discovery protocols (SDP) are network protocols which allows automatic finding available services, on the same network. The service discovery requires a common language to allow software agents to make use of each other's services without the need for user intervention.

Service Discovery in a Microservices Architecture

The concern not having the service discovery in a microservices Architecture could for instance in the case of you have some code to invoke a service that has a REST API. To make a request on a specific

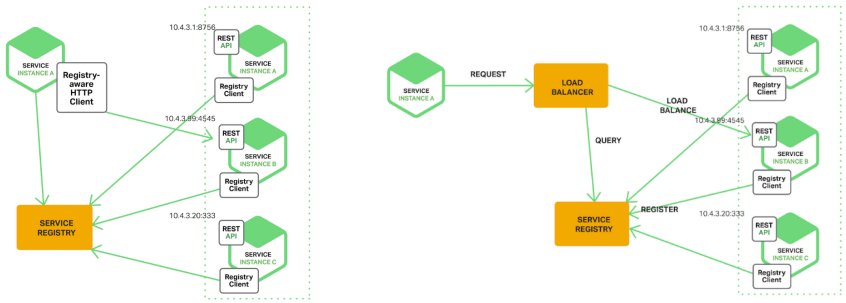
service you need to know it's IP address and port number. In a cloud-based microservices application the problem occurs since the services runs inside the pods, and the pods have dynamically assigned IP addresses and Ports. The reason why pods have dynamically network location is based on failures, upgrades and auto-scaling. There are two service discovery patterns called client-side discovery and server-side discovery, to solve the issue described above.

The client-Side Discovery Pattern

One of the approaches to service discovery is the client-side discovery pattern. In the client-side pattern the client has the responsible of determining the network locations of available services, and furthermore has the responsible of load-balancing requests. In the service registry the client can query for available services. The service registry then return a list of available services to the client, by using the load-balancing algorithm the client makes request for the available service. Services periodontally send a heartbeat to the service registry to say I'am in live. The network location for a specific service is removed from the service registry when it terminate. The client needs to have the business logic of the load-balancer, which is a cons that requires a lot of traffic on the client.

The server-side discovery pattern

The other approach to service discovery is the server-side discovery pattern. The approach is almost the same like in Client-Side Discovery Pattern, the difference is that client makes a request to service via load balancer, the load balancer then queries the service registry and return a list of available services to the load balancer. The load balancer then routes each request to the available service. The client only needs to concern to send requests to the load-balancer, and nothing else this is the pros of server-side discovery pattern. Figure 6 shows the differences between client-Side Discovery Pattern and the server-side discovery pattern



(a) client-Side Discovery Pattern

(b) server-side discovery pattern

Figure 6: Client service discovery vs Server service discovery

How may Service Discovery be beneficial in cluster and cloud computing?

7 CONCLUSION

This, then is the grand summary of what you have accomplished. You may well imagine that many readers will read your Introduction, and then skip to the Conclusion, and if, and only if, those two parts are interesting, might be tempted to read the rest. A consequence is that you should ensure that the reader will gain a good overall understanding of what you have done by reading only the conclusion. Thus, this is a place to summarise all that has gone before, before finally concluding on the results of your experiments and the validity of your hypotheses. It is also important to ensure that the Introduction (which in all likelihood was written first) still aligns closely with the conclusions reached.

If you so desire, this is also where you might add a section on Future Work, where you point in the directions that should be followed to complete the work you have already accomplished.

Part II

APPENDIX

APPENDIX TEST

Lorem ipsum at nusquam appellantur his, ut eos erant homero concludaturque. Albucius appellantur deterruisset id eam, vivendum partiendo dissentiet ei ius. Vis melius facilisis ea, sea id convenire referrentur, takimata adolescens ex duo. Ei harum argumentum per. Eam vidit exerci appetere ad, ut vel zzril intellegam interpretaris.

More dummy text.

A.1 APPENDIX SECTION TEST

Test: Table 1 (This reference should have a lowercase, small caps A if the option `floatperchapter` is activated, just as in the table itself → however, this does not work at the moment.)

LABITUR BONORUM PRI NO	QUE VISTA	HUMAN
fastidii ea ius	germano	demonstratea
suscipit instructior	titulo	personas
quaestio philosophia	facto	demonstrated

Table 1: Autem usu id.

A.2 ANOTHER APPENDIX SECTION TEST

Equidem detraxit cu nam, vix eu delenit periculis. Eos ut vero constituto, no vidit propriae complectitur sea. Diceret nonummy in has, no qui eligendi recteque consetetur. Mel eu dictas suscipiantur, et sed placerat oporteat. At ipsum electram mei, ad aeque atomorum mea. There is also a useless Pascal listing below: Listing 2.

Listing 2: A floating example (`listings` manual)

```
for i:=maxint downto 0 do
begin
{ do nothing }
end;
```

DECLARATION

Put your declaration here.

Aarhus, May 2016

Adi Tahirovic

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Final Version as of May 26, 2016 (`classicthesis`).