



RV Educational Institutions®
RV College of Engineering®

Autonomous
Institution Affiliated
to Visvesvaraya
Technological
University, Belagavi

Approved by AICTE,
New Delhi

Go, change the world

Major Project

18MCA61

on

Design and Development of Distributed Application in PaaS

Submitted by
Shivaji Rao Lokhande

1RD19MCA04

Under the Guidance
of

Dr. S Anupama Kumar
Associate Professor
Department of MCA
R V College of Engineering
Bengaluru – 560059

Mr Ashish Singh
Lead System Architect
Newfold Digital
Bengaluru - 560059

*Submitted in partial fulfillment of the requirements for the award of degree
of*

MASTER OF COMPUTER APPLICATIONS

2021 -2022

RV COLLEGE OF ENGINEERING®

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)

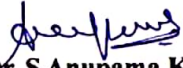
DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

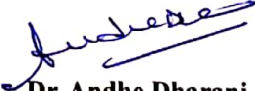
Bengaluru- 560059

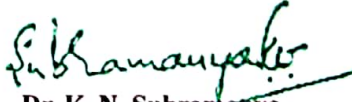


CERTIFICATE

Certified that the project work titled “Design and Development of Distributed Application in PaaS” carried out by Shivaji Rao Lokhande S G, USN : 1RD19MCA04, a bonafide student of RV College of Engineering®, Bengaluru submitted in partial fulfillment for the award of Master of Computer Applications of RV College of Engineering®, Bengaluru affiliated to Visvesvaraya Technological University, Belagavi during the year 2021-22. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirement in respect of project work prescribed for the said degree.


Dr. S Anupama Kumar
Associate Professor
Department of MCA
RVCE, Bengaluru-59


Dr. Andhe Dharani
Professor and Director
Department of MCA
RVCE, Bengaluru-59


Dr. K. N. Subramanya
Principal
RVCE, Bengaluru-59

PRINCIPAL
RV COLLEGE OF ENGINEERING
BENGALURU - 560 059

RV COLLEGE OF ENGINEERING®

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)

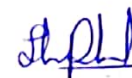
DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

Bengaluru- 560059

DECLARATION

I, **Shivaji Rao Lokhande S G**, student of sixth semester MCA in **Department of Master of Computer Applications**, RV College of Engineering®, Bengaluru declare that the project titled “**Design and Development of Distributed Application in PaaS** ” has been carried out by me. It has been submitted in partial fulfillment of the course requirements for the award of degree in **Master of Computer Applications** of RV College of Engineering®, Bengaluru affiliated to Visvesvaraya Technological University, Belagavi during the academic year **2021-22**. The matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

Date of Submission: 1-7-2022



Signature of the Student

Student Name: Shivaji Rao Lokhande S G

USN: 1RD19MCA04

Department of Master of Computer Applications

RV College of Engineering®

Bengaluru-560059



May 27, 2022

TO WHOMSOEVER IT MAY CONCERN

This is to certify that Mr. Shivaji Rao Lokhande S G was working at Endurance International Group (India) Private Limited (formerly known as Directi Web Technology Pvt. Ltd.), as an Intern since January 10, 2022, till May 27, 2022.

He has successfully completed the internship project under the guidance of Mr. Joshua Banks, VP – Infrastructure.

We found him sincere, hardworking and result oriented. He worked well as part of the team during his tenure.

We take this opportunity to thank him and wish him all the best in his future endeavors.

Regards,

For Endurance International Group (India) Private Limited.

A handwritten signature in black ink, appearing to read "Lavita Nathani".

Lavita Nathani
Vice President - Learning & Organizational Development

A handwritten signature in black ink, appearing to read "Shivaji Rao Lokhande S G".

Shivaji Rao Lokhande S G

Endurance International Group (India) Private Limited

Registered Office: Unit No. 501, 5th Floor, IT Building 3, Nesco IT Park, Nesco Complex,
Western Express Highway, Goregaon (East), Mumbai 400 063

CIN: U72300MH2012PTC226415, Tel: +91 22 67209000; Email Id: finacctind@endurance.com

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned the efforts with success.

First and foremost, I would like to extend my sincere gratitude to my project Manager **Mr. Ashish Singh** (Lead Systems Architect, Newfold Digital), also my project guide **Vasudevan M** (Senior Systems Architect, Newfold Digital) and the entire team at **Newfold Digital** for their constant guidance, continual encouragement and understanding. Without them the accomplishment of this task would have never been possible.

I would like to express my thanks to the Principal **Dr. K.N.Subramanya**, RV College of Engineering® for his encouragement that motivated me for the successful completion of the Project.

It gives me immense pleasure to thank **Dr. Andhe Dharani** Professor and Director ,Department of MCA , RV College of Engineering® for her constant support and encouragement.

I would like to express my deepest sense of gratitude to guide **Dr. S Anupama Kumar**, Associate Professor, Department of MCA, RV College of Engineering® for her constant support and guidance.

Last but not least, I would like to thank my parents and family for their unconditional love and support. It is indeed a pleasure to thank my friends who persuaded and encouraged me not only in this project but throughout the entire tenure of the course.

Shivaji Rao Lokhande
Master of Computer Applications
RV College of Engineering®
Bengaluru-59

ABSTRACT

As the businesses realize the dynamism of what can be done with their data, they are moving on from their existing resources to well-equipped Data Centers to aid better data management. The deployment and management of applications in large-scale clustered environments are always a difficult task with respect to enterprise mail operations, so to build a reliable platform that runs multiple mission critical workloads including containerized and non containerized applications the distributed data services should be portable across cloud providers or data centers, The project aims to design and develop a platform using Mesos which is known for production ready and reliable at scale than many other container-enabling technologies in the market.

The first module of this project is Statistical Analysis of Jobs using Mesos Master and Slave, Mesos follows Master and slave model in which Master is responsible to allocate the tasks for the slaves in mean while the slaves picks up the tasks and completes as instructed by the elected leader.. The second module is a Deployment of Applications using Marathon framework which is built on top of Mesos that helps for starting, stopping, and scaling applications. The third module is Cluster management using ZooKeeper which is used for maintaining centralized configuration information, leader election and to track the status of nodes in the Kafka cluster. Last module is Visualization of Metrics using Grafana which is used to analyze all the metrics of the designed infrastructure and it serves as an analytics application.

The staging setup will be completely puppetized into the production environment. This designed platform serves for the deployment of applications and it also changed the way that organizations create, ship the applications from one datacenter to another and maintain those in real-time. Container orchestration automates the deployment, maintenance, scaling as well as networking of applications. In today's world, where enterprises are required to deploy and manage multiple hosts, container orchestration can be the only rock-solid alternative.

PROJECT GUIDELINES FOR THE REPORT

As per the Non-Disclosure Agreement with the company, following are the project guidelines to abide by -

1. No mention of any client name.
2. No mention of any development methodology & models.
3. No mention of any proprietary tools used.
4. No mention of the source of the dataset used for training or any other purpose.
5. No mention of data flow between the modules.

List of Publication

Type of Publication	Details of Publication
International Journal	ICECAA - 2022 (International Conference on Edge Computing and Applications ICECAA 2022)

ICECAA 2022 - Decision: Revise and Resubmit



icecc confdesk

to me, anupamakumar

9 Jul 2022, 12:46 (2 days ago)



Dear author

Your article "**Deployment Strategy Using DevOps Methodology : A Thorough Comparison on Container Orchestration Frameworks**" is recommended for publication based on the following criteria:

Decision: Revise and Resubmit

Review comments:-

1. Deployment strategy using Devops methodology: A thorough comparison on container orchestration frameworks is the proposed title of this paper.
2. A similarity measure of 12% from a single source of literature is observed in the paper. It should be revised to less than 5%. Authors need to revise and resubmit the paper.

Reduce overall plagiarism less than 15%

Reduce single source less than 5%

Thank you

Table of Contents

CONTENTS	PAGE NO
College Certificate	i
Company Certificate	ii
Declaration by student	iii
Acknowledgement	iv
Abstract	v
Project Guidelines	vi
Table of Contents	vii
List of Tables	viii
List of Figures	ix
Chapter 1: Introduction	1-5
1.1 Project Description	2
1.2 Company Profile	4
1.3 Dissertation Organization	5
Chapter 2: Literature Review	6-13
2.1 Literature Survey	6
2.2 Existing and Proposed System	9
2.3 Tools and Technologies used	11
2.4 Hardware and Software Requirements	13
Chapter 3: Software Requirement Specifications	14-20
3.1 Introduction	14
3.2 General Description	16
3.3 Functional Requirement	17
3.4 External Interfaces Requirements	20

Chapter 4: System Design	22-28
4.1 Architectural Design	22
4.2 Context Diagram	28
Chapter 5: Detailed Design	29-36
5.1 System Design	29
5.2 Detailed design	35
Chapter 6: Implementation	37-56
6.1 Code Snippets / PDL	37
6.2 Implementation Screenshots	46
Chapter 7: Software Testing	57-64
7.1 Test cases	57
7.2 Testing and Validations	58
Chapter 8: Conclusion	65
Chapter 9: Future Enhancements	66
Bibliography	67

LIST OF TABLES

Table No	Table Label	Page No
2.1	Hardware Requirements of the Project	13
2.2	Software Requirements of the Project	13
3.1	Acronyms and Abbreviations	14
5.2	Mesos master quorum size	35
7.1	Unit Testing for Statistical Analysis of Jobs using Mesos Master and Slave	58
7.2	Boundary value testing for connection to the LAN	60
7.3	Integration Testing for Deployment of Applications using Marathon and LDAP Authentication	61
7.4	System Testing Visualization of Metrics and Nginx Reverse proxy	63

LIST OF FIGURES

Figure No	Figure Label	Page No
4.1	Block diagram of Mesos Architecture	23
4.2	Zookeeper cluster architecture ensemble	24
4.3	Zookeeper leader election architecture	25
4.4	Context Diagram of Mesos Infrastructure	28
5.1	Mail Operation Project Ecosystem	30
5.2	DFD Level - 0	32
5.3	DFD Level - 1	33
5.4	ER Diagram of Mesos Architecture	34
5.5	Mesos framework	36
6.1	Launching a Docker Container	38
6.2	Marathon app configuration	46
6.3	Marathon Application lifecycle	47
6.4	Apps deployed in Marathon	47
6.5	Marathon Application Health Check	48
6.6	Marathon Application Health Check	49
6.7	Mesos slaves tasks completion	50
6.8	Marathon Application Scaling process	51
6.9	LDAP authentication to access Marathon	52
6.10	Grafana Dashboard	53
6.11	Working with Grafana Dashboard UI	54
6.12	Grafana Dashboard Header	54
6.13	Active and silenced alerts of Grafana	55
7.1	Testing the coordination between Master and slave	59
7.2	Status of deployed apps in Marathon	62
7.3	Alert testing in Grafana	64

Chapter 1: Introduction

This chapter gives an overview of the project and its description. The company profile is also included for the reference.

The Distributed Applications(DApp) is designed to coordinate on the tasks, to access the information, users to collaborate and idea sharing and also to exchange apps via servers [1]. Distributed applications are client-server networks in which it allows user to access to information

from the cloud computing servers. In other words these are the applications or softwares that runs on multiple computers within the given network and it can be stored on cloud computing servers. During the traditional approach single systems used to run the applications but distributed applications make applications run on multiple systems and servers simultaneously at the same time for the given task or job. Enterprises and business organizations can choose container technology like Dockers to deploy and package distributed applications, Also containers can be used to run and build distributed applications in a cloud shared infrastructure.[2]

In cloud computing PaaS stands for Platform as a Service where third party cloud service providers provide the various software and hardware tools that are actually required to design and develop the required applications for the developers or users over the internet. On the PaaS infrastructure they host software and hardware tools services, with help of these kinds of services the developers are not required to install the complex in house software in order to run or build an application.

Distributed applications along with PaaS allows you to avoid the complexity and expensive cost of managing and buying the licenses of the software and also the underlying application infrastructure of container orchestration such as Mesos or Kubernetes [1]. So it basically helps developer to concentrate on the development of the application the rest is typically managed by the service providers

1.1 Project Description

Software engineers spend half of their working hours wading through the work of our forerunners , but veterans and great people of the community do emerge and just as the working individuals have learnt the lessons about writing the code , the community has learned collective lessons to work with software development at scale. It is not easy to see these larger trends when they are busy doing small bug fixes. DevOps is one of these huge trends .

It is a combination or say unification of two major traditionally separate worlds of software development into one common cycle. The problem arises during the development and deployment of the application in the large clustered environment. Docker Swarm, Kubernetes and Apache Mesos are a few various container orchestration frameworks that exist in the market. In this current era it is still a difficult task to opt the best suited optimal container orchestration solution. So the purpose of this work will help to provide analysis of the selected orchestration tool Mesos also provides the setup for data migration using DCOS.

The objectives of the project are as follows

- Identifying the optimal container orchestration framework based on the existing infrastructure.
- Building a staging setup of the required infrastructure using Mesos and Marathon
- Developing a centralized log management system using Grafana
- Achieving additional level abstraction of the servers using Nginx reverse proxy.

By referring to the above objectives the project is divided into two phases respectively to achieve the expected results.

First phase : Staging Setup

The first phase of the project is about identifying the optimal container orchestration framework from a bunch of existing tools such as Mesos, that fulfill the requirements of the project. The quorum setup , Mesos master and slave setups are configured. Marathon framework opted along with Mesos provided the interface to test and deploy applications in distributed environments which leads to scale them up as per the requirement. Lightweight Directory Access Protocol (LDAP) searches for group and consumer mapping and it has the ability to authenticate based on username or custom ID to bind to an enterprise Lightweight Directory Access Protocol (LDAP) directory with a password.

Second phase : Centralized log management and Puppetizing the setup

In the second phase of the project is to build a centralized log management system to analyze the logs and metrics for the setup. Monitoring tool Grafana is used to provide alerts in centralized log management, analytics and interactive visualization of web application of business metrics. Reverse proxy is achieved by Nginx which enables an additional level of abstraction and it also ensures the smooth network flow traffic between the clients and the provided specific servers by hiding the actual server details. Puppetizing the setup helped in automating and centralizing the configuration management process and also saved time by increasing deployment speed.

1.2 Company Profile

Newfold Digital

Newfold Digital (previously Endurance International Group) is a leading web technology company serving nearly 7 million customers globally. It provides solutions for small and medium-sized businesses. It offers domains, website builder, hostings, SEO services, customer support, and other solutions.

History

Newfold Digital (formerly Endurance International Group (EIG) and BizLand, and merged with Web.com in 2021), is an IT services company specializing in web hosting. The company was founded in 1997 and is headquartered in Burlington, Massachusetts, USA. It achieved its size by acquiring a large number of smaller companies, which it continues to operate under the original brand names, while moving their IT infrastructure to India

Mission

Newfold Mission is to empower success in a connected world with a focus on helping businesses of all sizes thrive online.

Brands

Newfold Digital owns more than 40+ brands. Among them below are few brands of Newfold family

- Bluehost
- HostGator
- Reseller Club
- Big Rock
- Network solutions
- Register.com
- web.com
- Crazy Domains

Services

Newfold digital provide various services such as

- Domains
- Webbuilders
- Cloud Services
- Reseller Hosting
- VPS Hosting
- Dedicated Hosting

1.3 Dissertation Organization

The entire report is divided into nine fragments. In Chapter Two, there will be details about existing systems or projects that are relevant to backend services, as well as which frameworks are best suited to the problems. The third chapter focuses on software requirement specifications, particularly functional and non-functional requirements. The system design, or how the clusters are created, is discussed in Chapters four and five, as well as the project's overall flow. Chapter Six focuses on the project's implementation in terms of the pseudocodes and procedures used in the project.

The software testing specifications and various types of unit testing that are required for efficient development and debugging are covered in Chapter Seven. Chapter Eight focuses on summarizing conclusions. Finally, the chapters include the project's future scope and closing remarks with the references.

Chapter 2: Literature Review

This chapter gives an overview of the references from the literature survey, Proposed system of the project, Tools and Technologies that are used and Hardware software requirements of the project.

2.1 Literature Survey

Andrea Tosatto; Pietro Ruiu; Antonio Attanasio in their paper [1] it aims to provide the overview of the container orchestration tool frameworks such Kubernetes, Docker swarm and Mesos and their existing problems, present analysis and future challenges that are to be addressed, In such manner the paper provides a insight on challenges and problems of container orchestration frameworks

The overall performance of Docker swarm is slightly worse than that of Kubernetes and Mesos because it will not provide the non containerized applications deployment as Kubernetes and Mesos does. Also the scaling up process of application is difficult in the provided infrastructure. Hence Mesos and Kubernetes provide optimal solutions for the current problems [2].

The paper [3] highlights the features of the container orchestration tools in detail for example the number of failures, scaling up time and the time required for deploying the applications in a distributed environment. By this study it is found that compared to any container orchestrated frameworks Kubernetes has better performance than any of its competitors.

The main goal is to analyze current options of Docker container orchestration and their advantages and disadvantages such as the way that docker helps for repeatable development for testing and production environments also on the other hand they are not able to run at the speed of bare metal servers. Marek, Martin Kontsek mentions that the Docker container-based virtualization approaches are frequently described as a high-performance alternative to hypervisors. [4]

The paper [5] includes discussion on Apache Mesos cluster development and the infrastructure that will be built according to the quorum standards because the quorum will be responsible for the high availability infrastructure and the main feature of Mesos in the distributed environment. So this paper highlights the infrastructure of the Apache Mesos along with the Master and Slave model.

Xiaolian Li, Yuling Jiang [6] This paper highlights the technology based on Docker application features like rapid scaling, easy configurations and security management are leading to the container cluster which is gradually moving towards a production friendly environment and the businesses are now including docker and containerization as a part of their technical stack.

MingminZhang explained about Marathon as a framework for the Apache mesos in the paper [7], Marathon provides the graphical user interface for the connected masters and slaves of Mesos container orchestration also provides the REST Api for the coordination between the cluster. Mesos combined with Marathon makes the infrastructure as same as Kubernetes infrastructure

The paper highlights about Zookeeper as a brain of the distributed infrastructure as because the zookeeper will be responsible for the coordination between the Master and Slave. It helps in case of leader election and if any failures of master the zookeeper will elect the leader and inform the slaves about the leader and keep the tasks progress by acheiving High Availability [8].

Paul Le, Alexandru Costan, Luc Bougé [9] This paper includes the importance of Apache Kafka in a distributed environment to build real time stream lines which includes storage, stream and compute analysis of both historical and real time data analysis. Apache kafka also helps in the log management part of the cluster.

Matthew Denis gives the final conclusion of the selected container orchestration framework Mesos over the other existing competitors in the paper [10] because Mesos infrastructure always helps when the business wants to do data migration of an existing infrastructure then Mesos will be the rock solid alternative than its competitors.

These survey papers helped to provide an overview of the container orchestration tool frameworks such as Kubernetes, Docker swarm and Mesos and their existing problems, present analysis and future challenges that are to be addressed, In such manner the papers provides an insight on challenges and problems of container orchestration frameworks. It also explains about Marathon as a framework for the Apache mesos, Marathon provides the graphical user interface for the connected masters and slaves of Mesos container orchestration also provides the REST Api for the coordination between the cluster.

Literature survey helped summarize the importance of docker and container tools and also it highlights the technology based on Docker application features like rapid scaling, easy configurations and security management are leading to the container cluster which is gradually moving towards the betterment and to analyze current options of Docker container orchestration and their advantages and disadvantages such as the way that docker helps for repeatable development for testing and production environments also on the other hand they are not able to run at the speed of bare metal servers.

The identified container orchestration framework Mesos comparatively has better performance over the other existing competitors in the market this is because Mesos infrastructure helps in the situation where the business wants to do data migration tasks of an existing infrastructure in such cases Mesos will be the rock solid alternative than its competitors. This is how opted system infrastructure helps to overcome the existing problems.

2.2 Existing and Proposed System

Existing system has several hurdles in the data migration process and containerized applications such as

1. Poor Knowledge of Source Data

This knowledge gap includes not being aware of the problems that exist in data, such as duplicates, missing information, misspellings and erroneous data. It can be all too easy to get complacent and assume that data can easily be configured into the parameters of a new system, however the reality could mean critical failures when it comes to user acceptance. So to ensure success, It is required to have a good understanding of the source data.

2. Underestimating Data Analysis

Due to constraints in computer systems, There may be incomplete, inaccurate and outdated information being transferred during the migration, often discovered very late in the day, even after the project has been completed. The outcome can mean not having enough time or the right resources to be able to identify and correct this data. Performing a thorough data analysis at the earliest possible occasion, usually when planning and designing your data migration can help you uncover these hidden errors.

3. Lack of Integrated Processes

Data migrations typically involve a disparate set of people using disparate technologies. Using disparate technologies can lead to failure in the transfer of data and its design between the analysis, development, testing and implementation phases. Things can get lost in translation, resulting in increased costs and wasted time. Organizations must look to utilize a platform that successfully links the critical inputs and outputs from each of the stages to help reduce error and save time and money.

4. Failure to Validate the Implementation

Like before, where the knowledge of source data is evident, It can still be difficult because of a lack of test cases. The risk of developing problems when it is often too late. Testing the migration using full volume data from the real world helps cover a wider range of possibilities and tests for the worst case scenario, which could be missed when using more convenient samples of data.

5. Application Dependencies

If users have a little development experience, they already know that these technologies each have a version the application depends on. It needs to run in an environment, which could be a development, test or a production environment. Since the environments can differ in OS, version, hardware etc, it's obvious that the application and its technologies with their respective versions should work the same on different environments.

Proposed System

The above mentioned existing problems such as poor knowledge of source data and underestimating the data analysis is achieved using the proposed identified container orchestration framework process. The integrated process between them is resolved using Marathon framework which in turn sums up to a distributed system which is loosely coupled and it solves the application dependencies between each of them and at the end all the modules are tested and validated.

It is important to note that a data migration is a challenging project and the risk involved is high. The proposed system intends to provide container orchestration framework Apache Mesos helps to scale and deploy application in low cost also providing Database storage, compute power, Data Center Operating System (DC OS) and other functionality to help businesses scale and grow, enhancing security to protect data and workloads.

This designed platform serves for the deployment of applications and it also changed the way that organizations create, ship the applications from one datacenter to another and maintain those in real-time. Mesos provides space for both containerized and non containerized applications so it will help the business organizations to choose mesos when they are working on existing projects. It will help testers to use the containerized docker images without any difficulty about the dependency issues as because they are loosely coupled.

2.3 Tools and Technologies Used

- **Mesos**

It is a one of the container orchestration solution for containerized and non containerized applications deployments [31]. It follows master and slave model where it includes specific number of masters and slaves in its infrastructure and Marathon as a framework which is build up on the Apache Mesos. Mesos provides a distributed, highly available architecture in which the Masters schedule work to be performed on the cluster, and slaves advertise available resources to the schedulers, which in turn execute tasks on the cluster. Mesos deployments compare and contrast with the traditional datacenter, and how an application-centric approach can lead to using resources more efficiently.

- **AWS (Amazon Web Services)**

It is cloud platform that is widely being used in the current are which eliminates the high cost and provides storage space, computing and scaling in an effective pay as you go payment model in which user are only required to pay on the usage of resource

- **CentOS**

It is one of the linux distributions which is primarily focused on providing a platform for open source communities to build, develop and deploy applications on cloud providers framework for hosting community and data processing solutions. It is also referred to as the Community Enterprise operating system.

- **Zookeeper**

It is a brain of the distributed infrastructure as because the zookeeper will be responsible for the coordination between the Master and Slave [24]. It helps in case of leader election and if any failures of master the zookeeper will elect the leader and inform the slaves about the leader and keep the tasks progress by achieving High Availability.

- **Grafana**

In this project used to analyze and visualize the metrics of real time data. In the project Influx database is used to store database information and it is configured with grafana to visualize the metrics and logs also active and silence alerts can be configured on grafana to get alert notifications. Grafana Alerts, If it matches the threshold it increases the count of active alerts and as soon as it is resolved it will be moved to silenced alerts. These alerts help in such a way that once the set threshold is reached it will alert the team through the opted channels it can be mail or notification as required for the user. From the alerts the operational engineers solves the issues such as storage on some server box storage limit reached, It can be server failure or Load balancing issues.

- **Docker**

It is the most popular software containerization platform in the world. It encapsulates the microservice in a Docker container [6], which can then be maintained and deployed independently. Each of these containers will be responsible for running a specific business functionality or say an individual microservice.

- **Containers**

It allows developers to package the required applications with all of its dependencies and distribute it as a single file [13] with the help of containers, developers are rest assured that their application will run on any platform without the need for any dependencies or incompatibilities, making writing and testing easier.

2.4 Hardware and Software Requirements

Hardware Requirement

Table : 2.1 Hardware Requirements of the project

Hardware	Requirement
Disk Space	32 GB Minimum and 64 GB Maximum
Processor	1.5 GHz 64 bit like NVIDIA, Hewlett-Packard
Memory(RAM)	16 GB Minimum and 32 GB Maximum
Bare-metal server	Depending on the infrastructure requirement

Software Requirement

Table : 2.2 Software Requirements of the project

Software	Requirement
Operating system	CentOS version : 8.0-1905
Browser	Firefox latest version 99.0
Platform	AWS CLI version 2
Docker	Minimum Version: 20.10.6 Maximum Version: 21.11.10
Git	Minimum Version: 2.31.1, Maximum Version: 2.8.2

Chapter 3 : Software Requirement Specification

This chapter provides a detailed description of the proposed system as well as its expected results. The system's definitions, general product description and functions, user characteristics, functional and non-functional requirements, and system limitation constraints are all included in the specification

3.1 Introduction

Software requirements specification (SRS) is a document that contains a detailed description of how the system should function. The Software Requirement Specification (SRS) stage of software development is where the requirements of the system under consideration are written down in order to lay the groundwork for the software development activities. Correctness, completeness of all essential requirements and their definitions, unambiguity, and consistency are all characteristics of a good SRS document.

Definitions, Acronyms and Abbreviations

Table 3.1 Acronyms and Abbreviations

Abbreviation	Explanation
IP	Internet Protocol
LDAP	Lightweight Directory Access Protocol
ARP	Address Resolution Protocol
DNS	Domain Name Server
MAC	Media Access Control
DC/OS	Data Center Operating System

Definitions

Container orchestration

It is a process of automating the operational efforts that are required to run the containerized and non containerized services and workloads such as container life cycle, deployment, scaling, load balancing and much more for the software teams.

AWS (Amazon Web Services)

It is a cloud platform that is widely being used in the current era which eliminates the high cost and provides storage space, computing and scaling in an effective pay as you go payment model in which users are only required to pay on the usage of resources.

Data Centre OS

It is one of the linux distributions which is primarily focused on providing a platform for open source communities to build, develop and deploy applications on cloud providers framework for hosting community and data processing solutions. It is also referred to as the Community Enterprise operating system.

Docker

It is the most popular software containerization platform in the world. It encapsulates the microservice in a Docker container, which can then be maintained and deployed independently. Each of these containers will be responsible for running a specific business functionality or say an individual microservice.

Containers

It allows developers to package the required applications with all of its dependencies and distribute it as a single file. With the help of containers, developers are rest assured that their application will run on any platform without the need for any dependencies or incompatibilities, making writing and testing easier.

3.1 General Description

This project is built on the identified container orchestration framework Mesos in which includes master and slaves and framework Marathon built over it. Marathon is responsible for providing interfaces for deployment of containerized and non containerized applications. Authentication and reverse proxy is built for that along with the centralized log management system for continuous monitoring the cluster.

3.2.1 Product Description

The product developed will be used for the users for testing and deployment of applications, The built infrastructure solves the issue of dependencies of the softwares in the distributed environment. As the developed cluster is loosely coupled the users need not to worry about the packages and dependencies of the software as they can use the containerized docker images this solves the operational time. The Grafana is developed to help in the log management of the cluster metrics.

3.2.2 Product Functions

- Mesos is responsible for the submitting the jobs to the slaves to get the status of the frameworks and tasks running on a Mesos cluster.
- Marathon provides the graphical user interface for the connected masters and slaves of Mesos container orchestration also provides the REST Api for the coordination between the cluster
- Lightweight Directory Access Protocol helps in securing and authentication in Mesosphere Marathon by verifying the provided credentials.
- Nginx reverse proxy which enables additional level of abstraction and it also ensures the smooth network flow traffic for between the clients and the provided specific servers by hiding the actual server details

- To visualize the business metrics Grafana is used as an open source tool which is used to analyze and visualize the metrics of real time data

3.2.3 User Characteristics

The application is used by software developers, testers and also by the end users. Testers for testing and developers for deployment of applications, The built infrastructure solves the issue of dependencies of the softwares in the distributed environment. As the developed cluster is loosely coupled the users need not to worry about the packages and dependencies of the software as they can use the containerized docker images this solves the operational time. For example if the tester is using a windows operating system and the developer built an application in Linux distribution such as CentOS or Ubuntu, It will be difficult for tester to run that application on windows. So the built infrastructure provides the containerized docker images which solves these issues as the package and dependencies are packed as a single docker image.

3.3 Functional Requirement

This project is built in five modules. The description of the modules is given below:

Module 1 : Statistical Analysis of Jobs using Mesos Master and Slave

Mesos is a special container orchestration solution for containerized and non containerized applications deployments. It follows master and slave model where it includes specific number of masters and slaves in its infrastructure and Marathon as a framework which is build up on the Apache Mesos.

- Input : Submitting the jobs to the slaves to perform the required task in the cluster
- Process : Zookeeper will coordinate with the master and slaves and helps to report a set of statistics and metrics of the cluster
- Output : The allocated agents executes the allotted task and reports the status of the frameworks and tasks running on a Mesos cluster.

Module 2 : Deployment of Applications using Marathon

Marathon as a framework for the Apache mesos, Marathon provides the graphical user interface for the connected masters and slaves of Mesos container orchestration also provides the REST Api for the coordination between the cluster [11]. Mesos combined with Marathon makes the infrastructure as same as Kubernetes infrastructure

- Input : Deployment of containerized Applications and deployment of non containerized applications like Nginx and Marathon LB etc
- Process : Running applications and Docker containers on top of a Mesos cluster.
- Output : Long-running Mesos tasks on Marathon and applications executes the required operations.

Module 3 : Service Authentication using LDAP

LDAP (Lightweight Directory Access Protocol) is an important open source tool and cross platform protocol that is used for directory services such as active directories for authentication provided for Marathon. The main features that can be considered are Password checking of the users and LDAP does grant access to Marathon.

- Input : Inserting the Access Directory Credentials of the user in the organization such as username and password
- Process : Securing and authentication in Mesosphere Marathon by verifying the provided credentials
- Output : On successful verification it grants access to Marathon and the dashboard access provided

Module 4 : Additional level abstraction using Nginx Reverse Proxy

Nginx is an open source web server Reverse proxy is achieved by Nginx which enables additional level of abstraction and it also ensures the smooth network flow traffic for between the clients and the provided specific servers by hiding the actual server details

- Input : The request made by the user to access the services provided and user enters the domain name of the required site that the user wants to access information
- Process : Request made by user has taken as the input and behind Nginx proxy hides the identities of servers and port number of the site on which it is running is not visible to the end user.
- Output : Response to the required request made by the user and provides the additional level of abstraction.

Module 5 : Visualization of Metrics

To visualize the business metrics Grafana is used as an open source tool which is used to analyze and visualize the metrics of real time data. In the project Influx database is used to store database information and it is configured with grafana to visualize the metrics and logs also active and silence alerts can be configured on grafana to get alert notifications.

- Input : The metrics data is stored in Influx database is provided as input for the grafana
- Process : Analyzes business metrics using time-series data from influxdb and collectd helps to process the data with grafana by being a connector between the influxdb and Grafana
- Output : Grafana results the received time series data inform creative understandable dashboards also it helps in setting up silence and active alerts

3.4 External Interfaces Requirements

User Interface

The given user interface for the particular software shall only be compatible with browsers. As the modules of this project provide a graphical interface for Marathon and Grafana, they can be accessible through the browsers which support advanced settings, for browsers like firefox.

Software Interface

- Virtual box is used for machines as well as different technologies like scapy, netifaces and libraries have been used.
- Amazon web services cloud platform is required to configure and deploy the website. The AWS space provides hosting facilities to host the site in which this enables building developing hosting within a platform.

3.5 Non-Functional Requirements

- Performance

Distributing load using Load balancer for the incoming requests The Marathon should respond in less than 5 seconds, though it may take a few seconds longer if there are a large number of responses to be fetched.

- Scalability

Dynamic scaling and predictive scaling of the applications deployed on marathon. The applications can be containerized and non containerized applications. The system is scalable because it is built on a microservices architecture. A scalable, performance microservice is one that is driven by efficiency, one that can not only handle a large number of tasks or requests at once, but can also handle them efficiently and is prepared for future tasks or requests.

- High Availability

Avoiding single point of failure by Mesos master quorum setup along with zookeeper. This provides the system architecture to be available all the time without failure this is achieved by leader election by zookeeper

- Portability

Cloud Applications are Platform independent. As the project is a built on distributed architecture it helps to deploy applications in any environment irrespective of the system configurations and dependencies. The applications are containerized here which enables and brings the portability to the project.

- Robustness

When handling exceptions, the code should not break. It should be able to deal with errors that occur during execution. If any exceptions occur in the backend code, it will take 10 to 40 seconds to restart the server.

Chapter 4: System Design Specification

System design provides an overview of the system's architecture, including how the system is connected internally, how work flows within the system, and the concept of complete system components.

4.1 Architectural Design

4.1.1 Problem Specification

DevOps is one of the major trends in the market . It is a combination or say unification of two major traditionally separate worlds of software development into one common cycle . It is not a recent act of invention rather it is a result of years of iteration. The deployment and management of applications in large-scale clustered environments are always a difficult task. Docker Swarm, Kubernetes and Apache Mesos are a few various container orchestration frameworks that exist in the market. There is still a lack of the best suited orchestration solution. The purpose of this work is to close that gap and provide the experiment findings for the same.

Deploying of the applications within the particular data center has been evolved gradually and it involves one or more physical or virtual servers. The adoption of the virtualization has made us to allow and run multiple virtual machines on a single physical server, this made the better of physical resources but applications running this way also refers that running a full pledged operating system on each of the the machines may consume resources and brings up the maintenance overhead. Docker Swarm, Kubernetes and Apache Mesos are a few container orchestration frameworks.

This designed platform serves for the deployment of applications and it also changed the way that organizations create, ship the applications from one datacenter to another and maintain those in real-time.

Mesos provides space for both containerized and non containerized applications so it will help the business organizations to choose mesos when they are working on existing projects. It will help testers to use the containerized docker images without any difficulty about the dependency issues as because it is loosely coupled.

4.1.2 Block Diagram

In the first phase of this project the block diagram shows the main components of orchestration framework Mesos. It includes Mesos master which is responsible for providing task to the slaves and Zookeeper in the cluster helps in leader election process which is in turn responsible for slaves to know about their leader as soon as Master fails and move to the standby state.

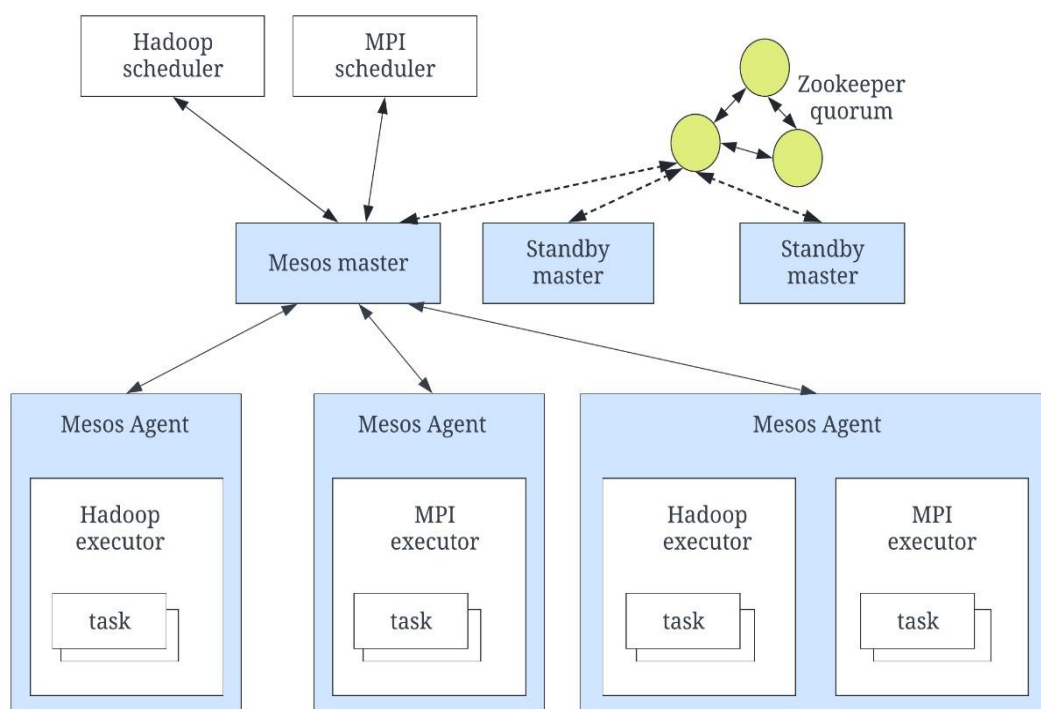


Figure 4.1: Block diagram of Mesos Architecture

4.1.3 Architectural Design

Zookeeper Ensemble

Mesos frameworks—rely on a ZooKeeper ensemble for leader election, coordination, and to maintain state. Because of these dependencies, which could be owed to ZooKeeper’s reputation in distributed systems coordination

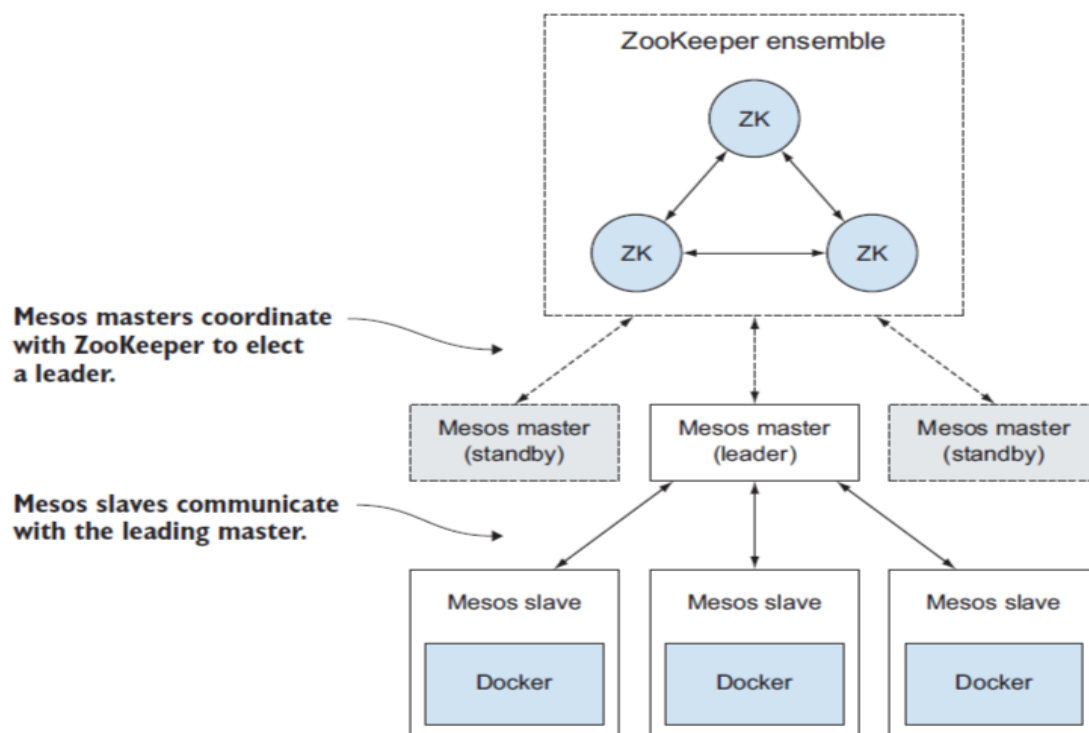


Figure 4.2 : Zookeeper cluster architecture ensemble

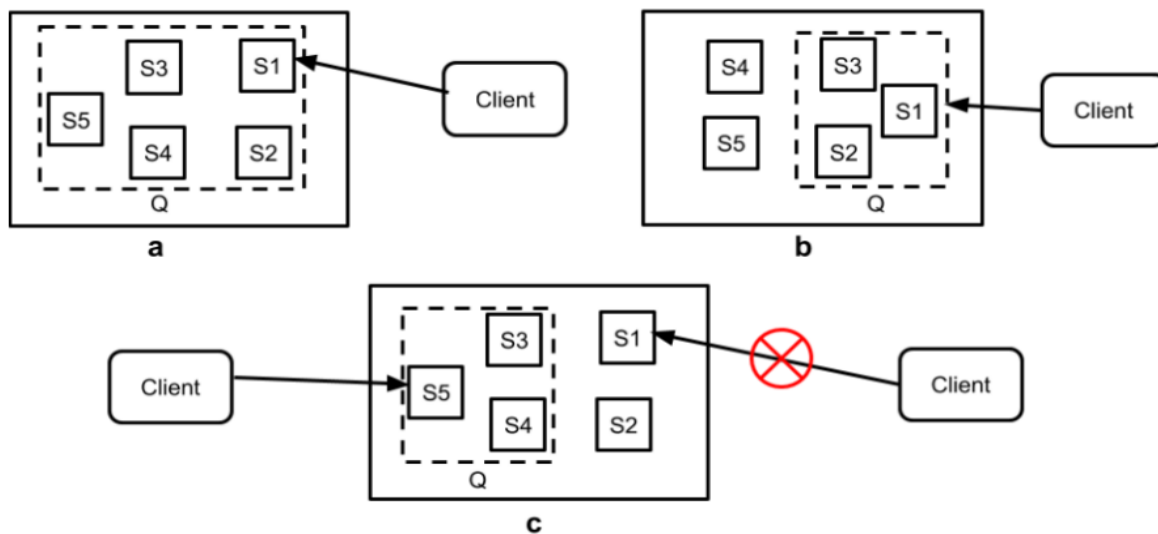


Figure 4.3 : Zookeeper leader election architecture

The idea behind this algorithm is that it always tries to elect the leader as the process with the most up-to-date transaction history and it will be voted on by the majority of nodes. Basically the zookeeper always works in quorum and to choose the head he always has to follow

The formula used to identify the quorum : $QN = \frac{(N+1)}{2}$

- QN: Minimum number of servers in quorum
- N: Total number of servers. (must be odd)
- So if we have 5 servers, the quorum must have at least 3 servers, 3 servers still count as majority (out of a fully defined quorum of 3)

Zookeeper Data Recovery

- Zookeeper keeps track of every transaction that occurs (read or write) in a file called a "transaction log" file.
- The ZooKeeper service host continues to save point-in-time copies as snapshots of the ZooKeeper tree or namespace on the local file system, and the snapshot is asynchronous.

4.1.1 Module Specification

Module Name: Statistical Analysis of Jobs using Mesos Master and Slave

This module Mesos is responsible for the submitting the jobs to the slaves to get the status of the frameworks and tasks running on a Mesos cluster. When Mesos slaves advertise available resources to the master, the master makes some decisions based on the fair share of resources assigned to the registered frameworks. The resources are then offered to a framework's scheduler, which includes the logic for accepting or declining resource offers. If a resource offer is accepted, the scheduler is responsible for launching the executors on the slaves, which in turn launch the framework's tasks.

Module Name: Deployment of Applications using Marathon

This module focuses provides the graphical user interface for the connected masters and slaves of Mesos container orchestration also provides the REST Api for the coordination between the cluster. The Marathon applications follow the application lifecycle in between that application health checks are mandatory for this aspect the testing is necessary for Marathon application deployments. The applications which are running successfully depict that the application has gone through all the checks prescribed and the application got deployed successfully.

Module Name: Service Authentication using LDAP

This module helps in securing and authentication in Mesosphere Marathon by verifying the provided credentials. A plugin for Mesosphere Marathon which authenticates users via UI/REST against an LDAP Server. The features are it Allows authentication against predefined static users, Supports LDAP and user memberships, Allows membership/group level permissions support full CRUD against Apps and Groups including ID paths, Tested against Marathon 1.1.1+ and Easy setup and JSON based configuration

Module Name: Visualization of Metrics

This module used to visualize the business metrics using Grafana. It is an open source tool which is used to analyze and visualize the metrics of real time data. Alerts can be set using Grafana alert feature, also it can have alerts sent through a number of different alert notifiers, including PagerDuty, SMS, email, VictorOps, OpsGenie, or Slack.. Grafana Alerts, If it matches the threshold it increases the count of active alerts and as soon as it is resolved it will be moved to silenced alerts.

These alerts help in such a way that once the set threshold is reached it will alert the team through the opted channels it can be mail or notification as required for the user. From the alerts the operational engineers solve the issues such as storage on some server box storage limit reached, It can be server failure or Load balancing issues. The engineers solve the issues as tickets or incidents and make the active alert as a silenced alert.

Module Name: Additional level abstraction using Nginx Reverse Proxy

This module enables an additional level of abstraction and it also ensures the smooth network flow traffic between the clients and the provided specific servers by hiding the actual server details. Nginx reverse proxy acts as an intermediate server that intercepts client requests and forwards them to the appropriate upstream backend server and subsequently forwarded a response from the server back to the client. The reverse proxy provides various benefits as an abstract layer above upstream servers

4.2 Context Diagram:

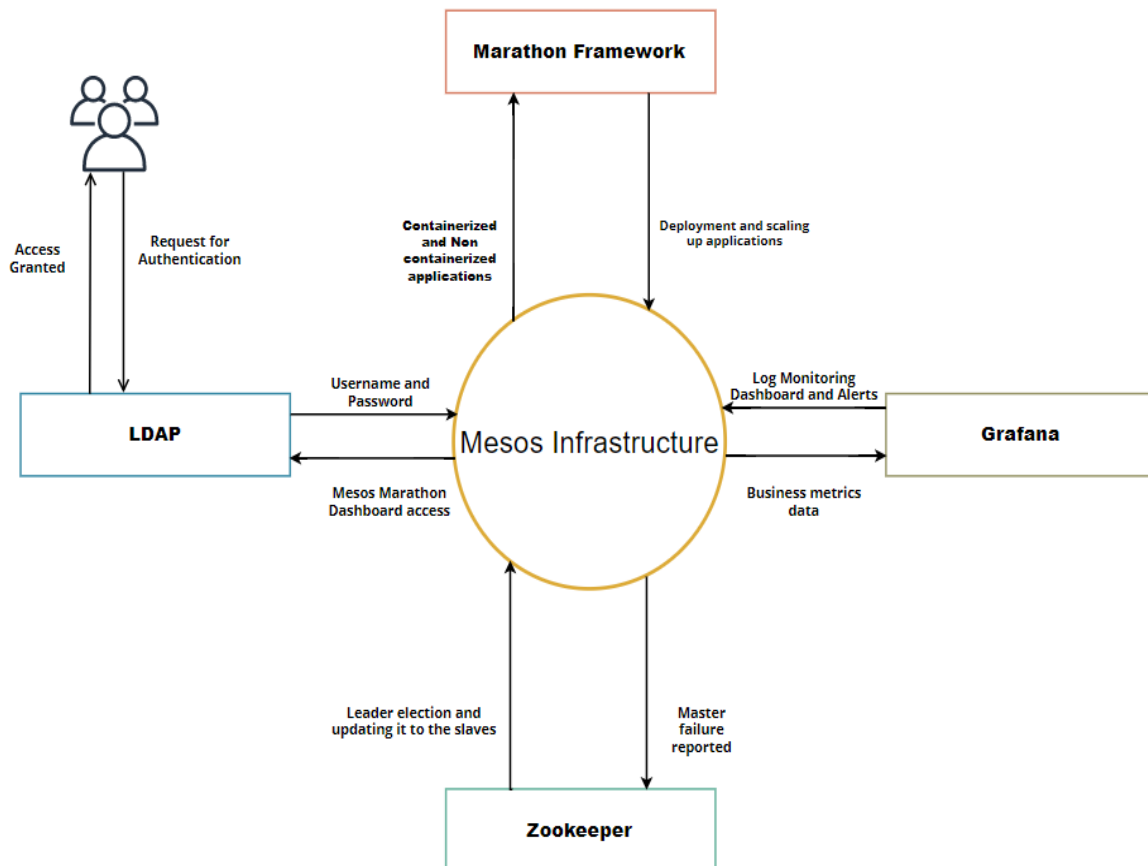


Figure 4.4 : Context Diagram of Mesos Infrastructure

The above diagram shows about interaction happening between each modules, Mesos includes master and slaves and framework Marathon built over it. Marathon is responsible for providing interfaces for deployment of containerized and non containerized applications. Authentication of users using LDAP along with the centralized log management system using Grafana for continuous monitoring of the cluster.

Chapter 5 : Detailed design

This chapter describes about the architectural design of the project which includes the Mail operation ecosystem and the working flow between the modules of the entire project architecture also it explains about the container orchestration Mesos framework along with the master and slave architecture.

5.1 System design

The focus of the object-oriented approach is on encapsulating the structure and behavior of information systems into smaller modules that mix data and process. The basic goal of Object Oriented Design (OOD) is to make system analysis and design more usable in order to increase quality and productivity.

Mail Operation Ecosystem

- The below figure gives a clear picture of the entire project and how the system is integrated with one another. Mesos is a special container orchestration solution for containerized and non containerized applications deployments. It follows master and slave model where it includes specific number of masters and slaves in its infrastructure and Marathon as a framework which is build up on the Apache Mesos.
- Deployment of applications using Marathon as a framework for the Apache mesos, Marathon provides the graphical user interface for the connected masters and slaves of Mesos container orchestration also provides the REST Api for the coordination between the cluster.

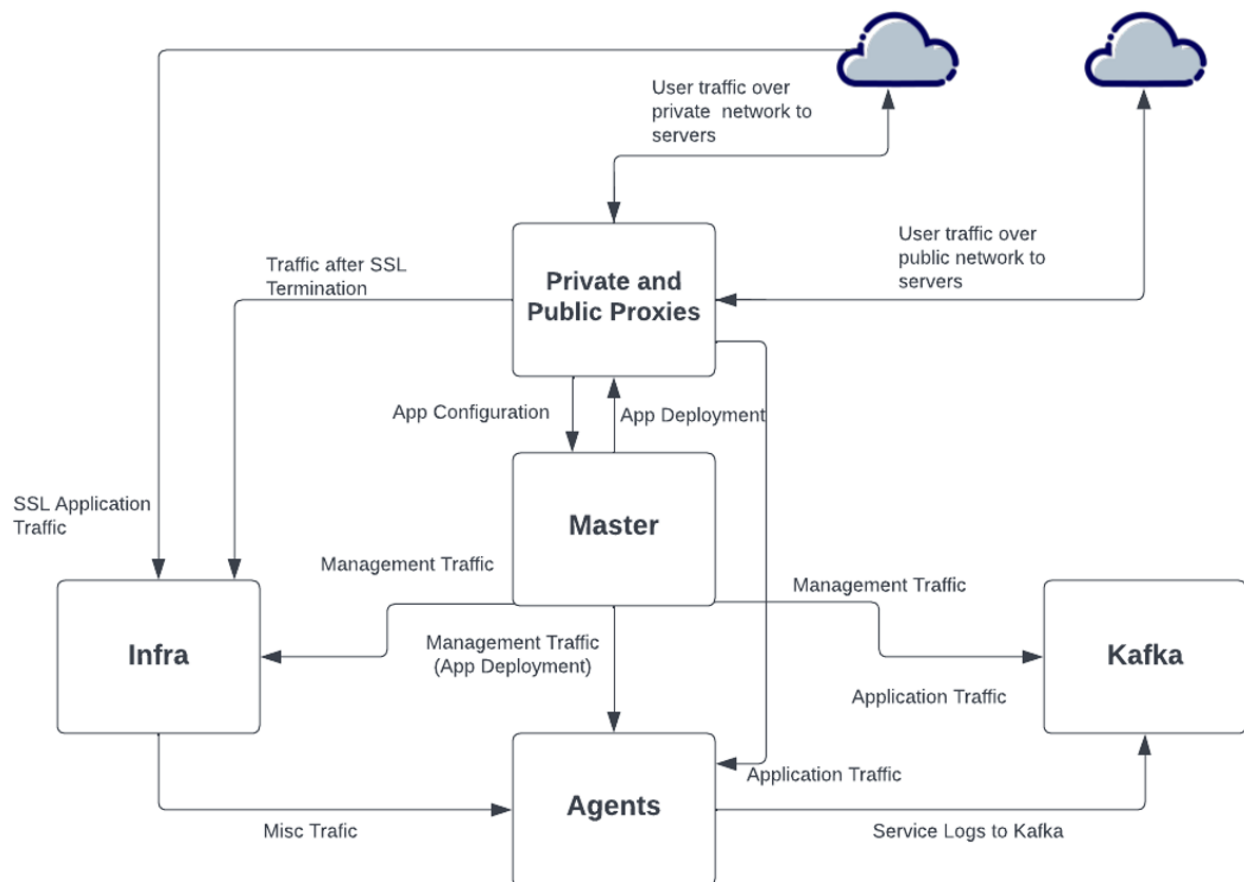


Figure 5.1 : Mail Operation Project Ecosystem

- Nginx is an open source web server Reverse proxy is achieved by Nginx which enables an additional level of abstraction and it also ensures the smooth network flow traffic for between the clients and the provided specific servers by hiding the actual server details.
- In the project Influx database is used to store database information and it is configured with grafana to visualize the metrics and logs also active and silence alerts can be configured on grafana to get alert notifications.

In general the project is divided into two phases,

- First phase involves the staging setup of the Mesos infrastructure.
 - a) Identifying the optimal container orchestration framework based on the existing infrastructure.
 - b) Mesos master and slave setup according to the quorum requirement with Zookeeper.
 - c) Setting up a Marathon framework that provides the interface to test and deploy applications in distributed environments.
 - d) LDAP authentication to grant access for Marathon.

- Second phase involves Centralized log management and Puppetizing the setup
 - a) Grafana setup for centralized log management system to analyze the logs and metrics for the setup
 - b) Nginx reverse proxy provides an additional level of abstraction of servers by hiding the server details.
 - c) Puppetizing the setup helped in automating and centralizing the configuration management process and also saved time by increasing deployment speed.

Functional Modeling

Data Flow Diagram

DFD a data-flow map is a way of serving a sail of statistics of a process or a system. The DFD additionally supplies databases about the outputs and inputs of each entity and the process itself. A data-flow map has no control flow, there're a lot of no decision rules and no loop.

DFD Level - 0

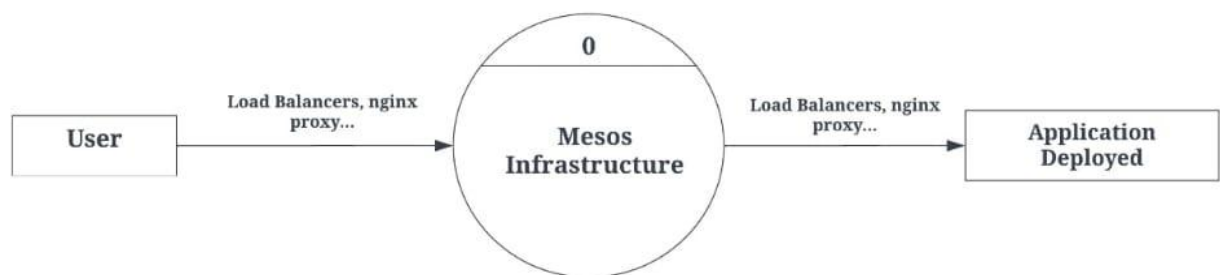


Figure 5.2 : DFD Level - 0

The above diagrams show the different levels of the Data Flow Diagram. The figure. 5.2 shows the highest level of the DFD. The internal process that takes place when a user requests for application deployment. The users here are Developers and Testers in which the applications that need to be deployed are Marathon load balancers, Reverse proxy and other required applications for these Mesos provides the required infrastructure.

DFD Level - 1

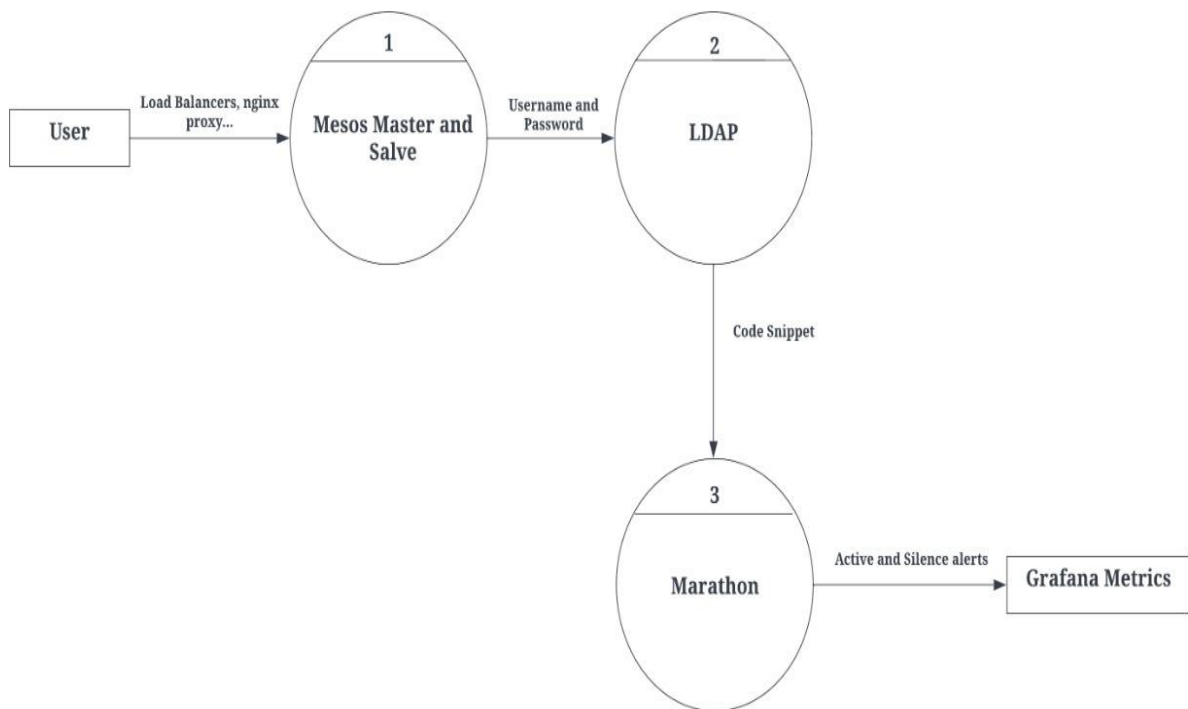


Figure 5.3 : DFD Level - 1

The figure. 5.3 shows the data that is flown between different modules, This project is built on the identified container orchestration framework Mesos in which includes master and slaves and framework Marathon built over it. Marathon is responsible for providing interfaces for deployment of containerized and non containerized applications. Authentication using LDAP and along with the centralized log management system for continuous monitoring of the cluster using Grafana.

ER Diagram

ER map stands for Entity Relationship Diagram, additionally known as ERD is a map that showcases the relationship of entity sets stored in a database. In other words, ER diagrams assist to elaborate the sensible structure of the system. ER diagrams are generated as said by three basic concepts: entities, attributes and relationships.

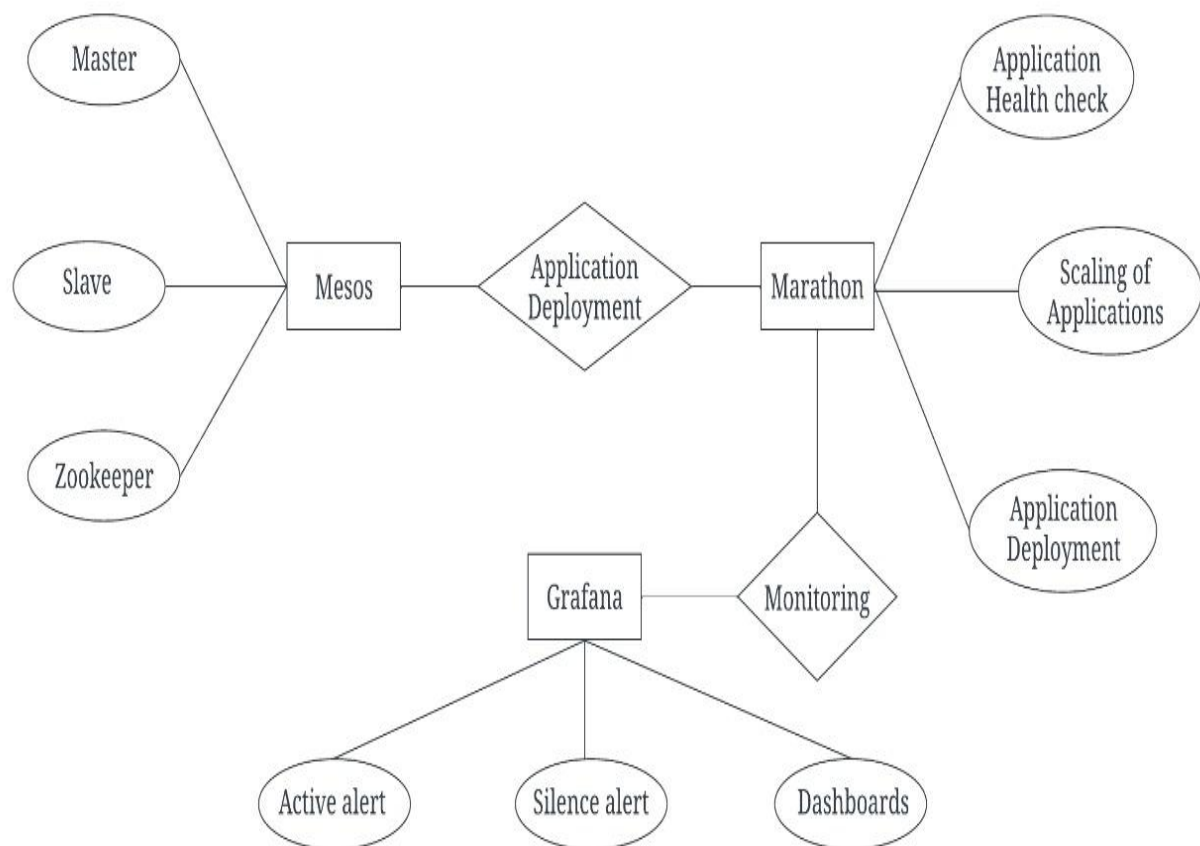


Figure 5.4 : ER Diagram of Mesos Infrastructure

5.2 Detailed design

Design decisions

This section includes the decisions that were made to set the quorum size of each Mesos master with the configuration.

At times it might be necessary to modify the number of Mesos masters running the cluster. Perhaps this is to replace failed hardware, rebuild a VM on a newer release of an operating system, or provision additional masters to improve your high availability strategy. There needs to be an odd number of masters to maintain a quorum; a majority of masters is required to make decisions for the cluster. The quorum size is set on each of the Mesos masters with the quorum configuration option.

Table 5.2 : Mesos master quorum size

Number of Mesos masters	Mesos master quorum	Number of failures tolerated
1	1	0
3	2	1
5	3	2
$2 \times N - 1$	N	$N - 1$

The idea behind this algorithm is that it always tries to elect the leader as the process with the most up-to-date transaction history and it will be voted on by the majority of nodes. Basically the zookeeper always works in quorum and to choose the head he always has to follow

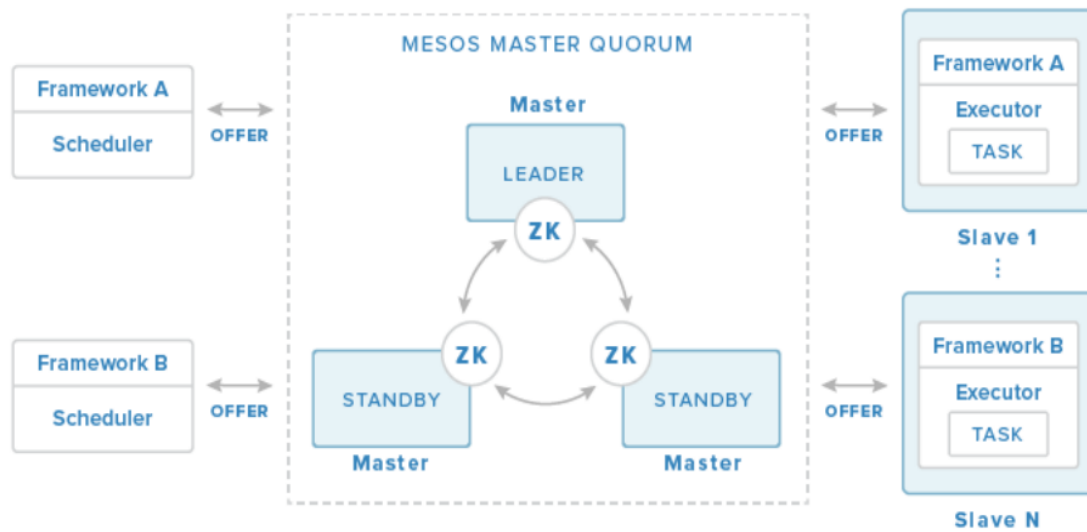


Figure 5.5 : Mesos framework

The formula used to identify the quorum : $QN = \frac{(N+1)}{2}$

- QN: Minimum number of servers in quorum
- N: Total number of servers. (must be odd)
- So if we have 5 servers, the quorum must have at least 3 servers, 3 servers still count as majority (out of a fully defined quorum of 3)

Failure to follow these basic rules could result in service interruptions. As with any production service, be sure to test changes in a pre-production environment, and proceed with caution when making changes to a production system.

Chapter 6: Implementation

6.1 Code Snippets

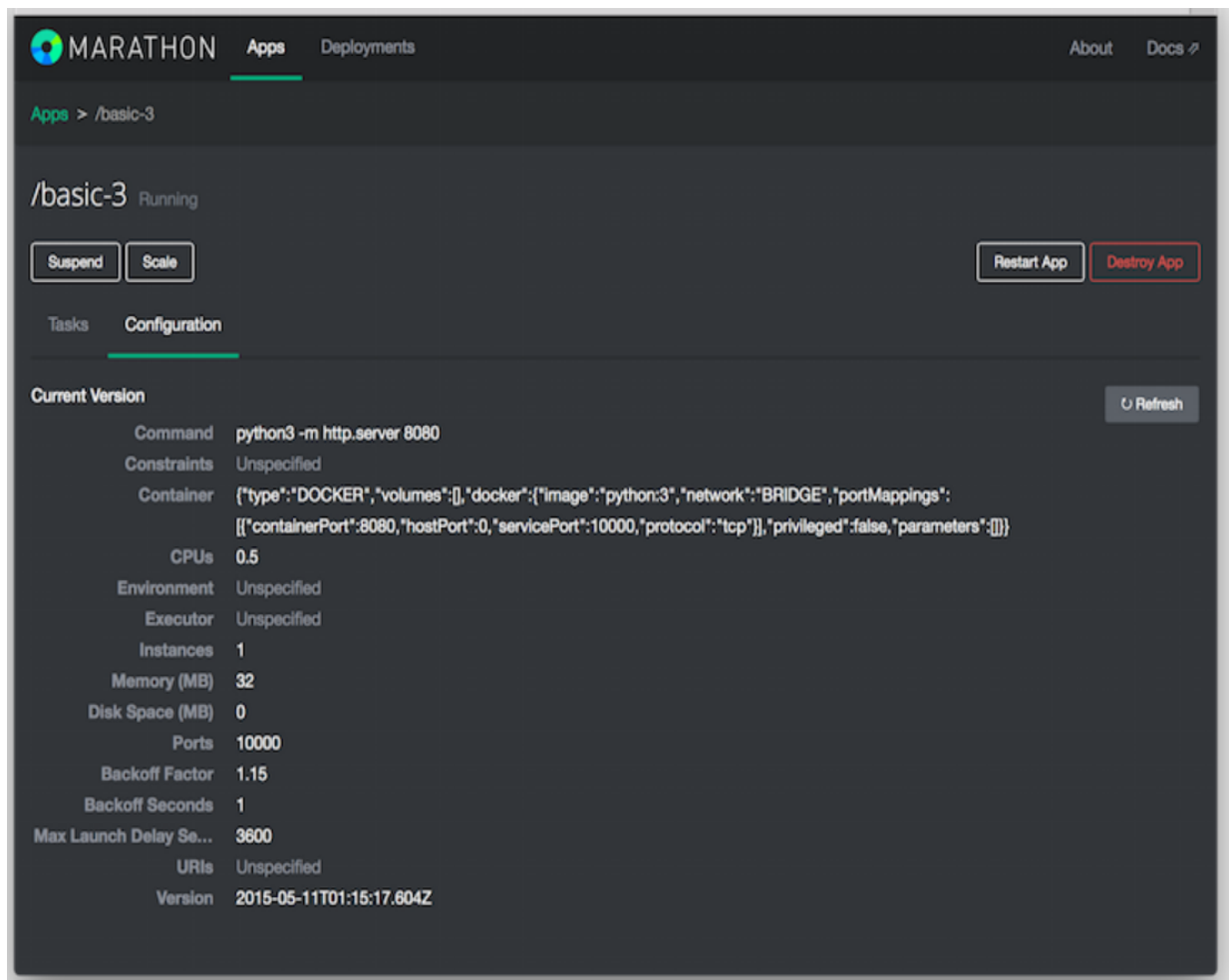
6.1.1 Deployment of Docker based applications

The below code snippet is a simple Docker app: a Python-based web server using the image `python:3`. Inside the container, the web server runs on port 8080 (the value of `containerPort`). `hostPort` is set to 0 so that Marathon assigns a random port on the Mesos agent, which is mapped to port 8080 inside the container.

```
{
  "id": "basic-3",
  "cmd": "python3 -m http.server 8080",
  "cpus": 0.5,
  "mem": 32.0,
  "networks": [ { "mode": "container/bridge" } ],
  "container": {
    "type": "DOCKER",
    "docker": {
      "image": "python:3"
    },
    "portMappings": [
      { "containerPort": 8080, "hostPort": 0 }
    ]
  }
}
```

- Marathon pods support the Mesos containerizer. The Mesos containerizer supports multiple images, such as Docker.

Marathon has launched a Python-based web server in a Docker container, which is now serving the contents of the container's root directory.



6.1 Launching a Docker Container

Here is an example of a pod pulling a Docker image from a private registry.

```
{
  "id": "/simple-pod",

  "scaling": {
    "kind": "fixed",
    "instances": 1
  },
}
```

```
"containers": [{
  "name": "container0",
  "exec": {
    "command": {
      "shell": "sleep 1000"
    }
  },
  "image": {
    "kind": "DOCKER",
    "id": "company/private-image",
    "pullConfig": {
      "secret": "configSecret"
    }
  },
  "resources": {
    "cpus": 1,
    "mem": 50.0
  }
}],
"secrets": {
  "configSecret": {
    "source": "/config"
  }
}
```

- Pods enable you to share storage, networking, and other resources among a group of applications on a single agent, address them as one group rather than as separate applications, and manage health as a unit.
- Pod instances may include one or more tasks that share certain resources (e.g. IPs, ports, volumes).
- Pods require the use of the Mesos Universal Container Runtime.

6.1.2 Creating and Managing Pods

Marathon supports the creation and management of pods. Pods enable you to share storage, networking, and other resources among a group of applications on a single agent, address them as one group rather than as separate applications, and manage health as a unit.

```
{
  "containers": [
    {
      "artifacts": [],
      "endpoints": [],
      "environment": {},
      "exec": {
        "command": {
          "shell": "sleep 1000"
        }
      },
      "healthCheck": null,
      "image": null,
      "labels": {},
      "lifecycle": null,
      "name": "sleep1",
      "resources": {
        "cpus": 0.1,
        "disk": 0,
        "gpus": 0,
        "mem": 32
      },
    },
    {
      "user": null,
```

```
    "volumeMounts": []
  },
],
"environment": {},
"id": "/simplepod2",
"labels": {},
"networks": [
  {
    "labels": {},
    "mode": "host",
    "name": null
  }
],
"scaling": {
  "instances": 2,
  "kind": "fixed",
  "maxInstances": null
},
"scheduling": {
  "backoff": {
    "backoff": 1,
    "backoffFactor": 1.15,
    "maxLaunchDelay": 3600
  },
  "placement": {
    "acceptedResourceRoles": [],

    "constraints": []
  },
}
```

```
"upgrade": {
  "maximumOverCapacity": 1,
  "minimumHealthCapacity": 1
},
"secrets": {},
"user": null,
"volumes": []
},
"networks": [
  {
    "mode": "host"
  }
]
}
```

- Task groups, or “pods”, allow a scheduler to group one or more tasks into a single workload.
- When one task is specified alongside an executor that has a unique executor ID, the task group is simply executed as a single isolated OS process; this is the simple case of a single task.

Containerizers

Marathon pods support the Mesos containerizer. The Mesos containerizer supports multiple images, such as Docker. Learn more about running Docker containers on Marathon.

The following JSON specifies a Docker image for the pod:

```
{
  "image": {
    "id": "mesosphere/marathon:latest",
    "kind": "DOCKER",
    "forcePull": false
  }
}
```

An optional `image.pullConfig` is supported too. Here is an example of a pod pulling a Docker image from a private registry:

```
{
  "id": "/simple-pod",
  "scaling": {
    "kind": "fixed",
    "instances": 1
  },
  "containers": [{
    "name": "container0",
    "exec": {
      "command": {
        "shell": "sleep 1000"
      }
    },
    "image": {
      "kind": "DOCKER",
      "id": "company/private-image",

      "pullConfig": {
        "secret": "configSecret"
      }
    }
  ]
}
```

```
    }  
  },  
  "resources": {  
    "cpus": 1,  
    "mem": 50.0  
  }  
},  
"secrets": {  
  "configSecret": {  
    "source": "/config"  
  }  
}  
}
```

6.1.3 Marathon application health checks

Mesos-level health checks (MESOS_HTTP, MESOS_HTTPS, MESOS_TCP, and COMMAND) are locally executed by Mesos on the agent running the corresponding task and thus test reachability from the Mesos executor.

```
"healthChecks": [  
  {  
    "protocol": "HTTP",  
    "path": "/ping",  
    "portIndex": 0,  
    "gracePeriodSeconds": 3,  
    "intervalSeconds": 30,  
    "timeoutSeconds": 10,  
    "maxConsecutiveFailures": 3  
  },  
]
```



```
{  
  
  "protocol": "COMMAND",  
  "command": {  
    "value": "curl -f -X GET http://$HOST:$PORT0/ping"  
  },  
  "maxConsecutiveFailures": 3  
}
```

]

- Health checks are a powerful way to ensure that the various instances, or tasks, for
- a given application is all up and healthy at the application layer.
- They play a crucial role in allowing Marathon to perform rolling upgrades of a given application—deploying a new version of an application with zero downtime.

6.2 Implementation Screenshots

- Marathon app deployment

Marathon provides the graphical user interface for the connected masters and slaves of Mesos container orchestration also provides the REST Api for the coordination between the cluster

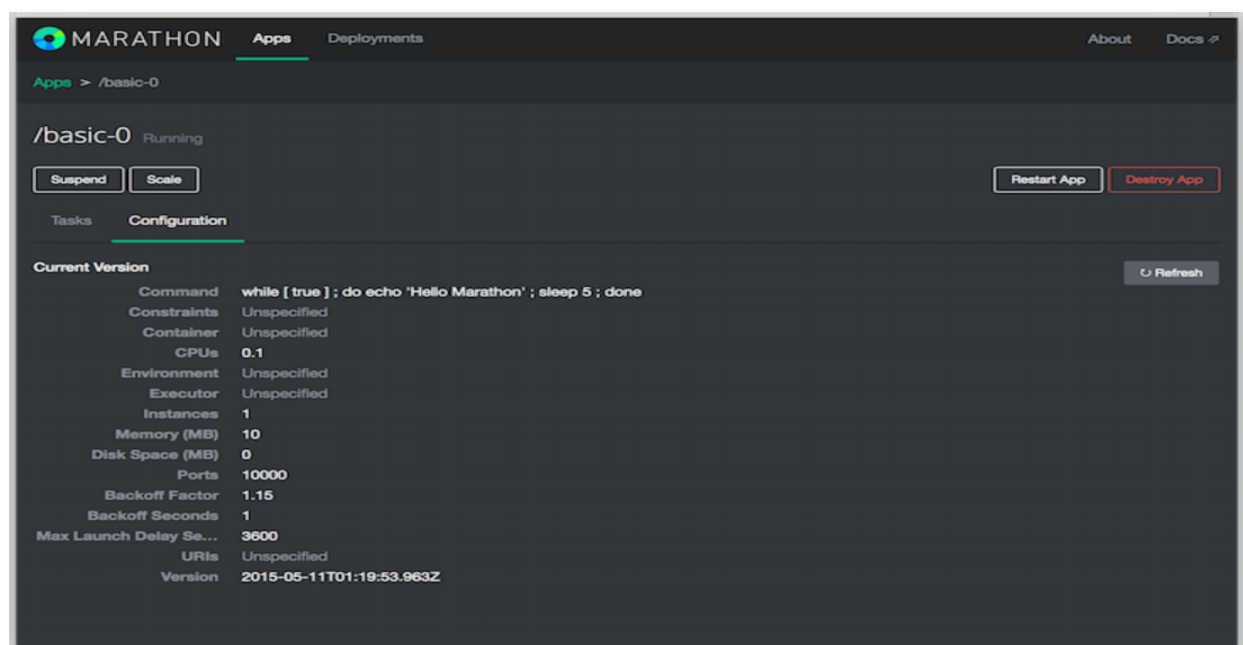


Figure 6.2 : Marathon app configuration

When it is defined and launches an application, Marathon hands over execution to Mesos. Mesos creates a sandbox directory for each task. The sandbox directory is a directory on each agent node that acts as an execution environment and contains relevant log files. The stderr and stdout streams are also written to the sandbox directory.

Marathon app lifecycle displays about the healthy and unhealthy status of deployed applications.

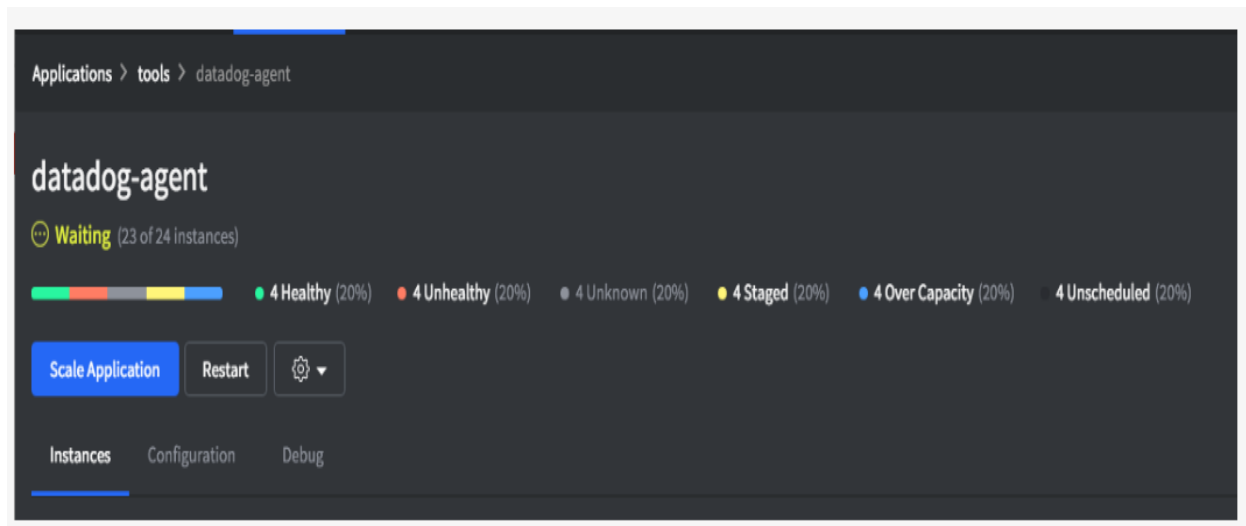


Figure 6.3 : Marathon Application lifecycle

Marathon App deployment and the status of proxy and replica applications and their usages

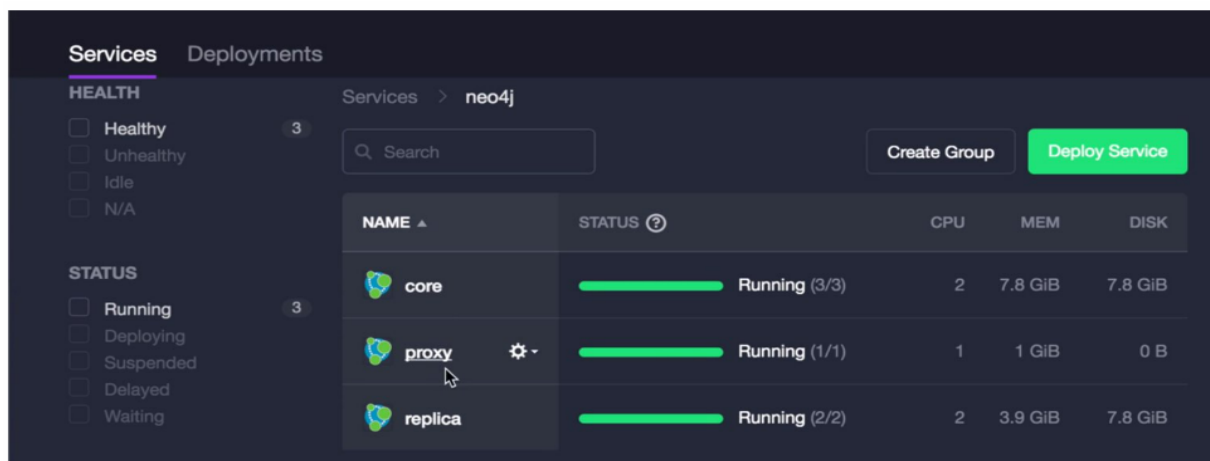


Figure 6.4 : Apps deployed in Marathon

After the health checkup progress gets completed, In the next process the app will be moved to waiting state, if the requirements are not fulfilled then the app will get delayed, else the app will get deployed in a marathon applications such as load balancer application, Python and Proxy applications are get deployed in Marathon.

The below pictures show the health check warning of the marathon applications, the TCP warning messages in the checks as warnings.

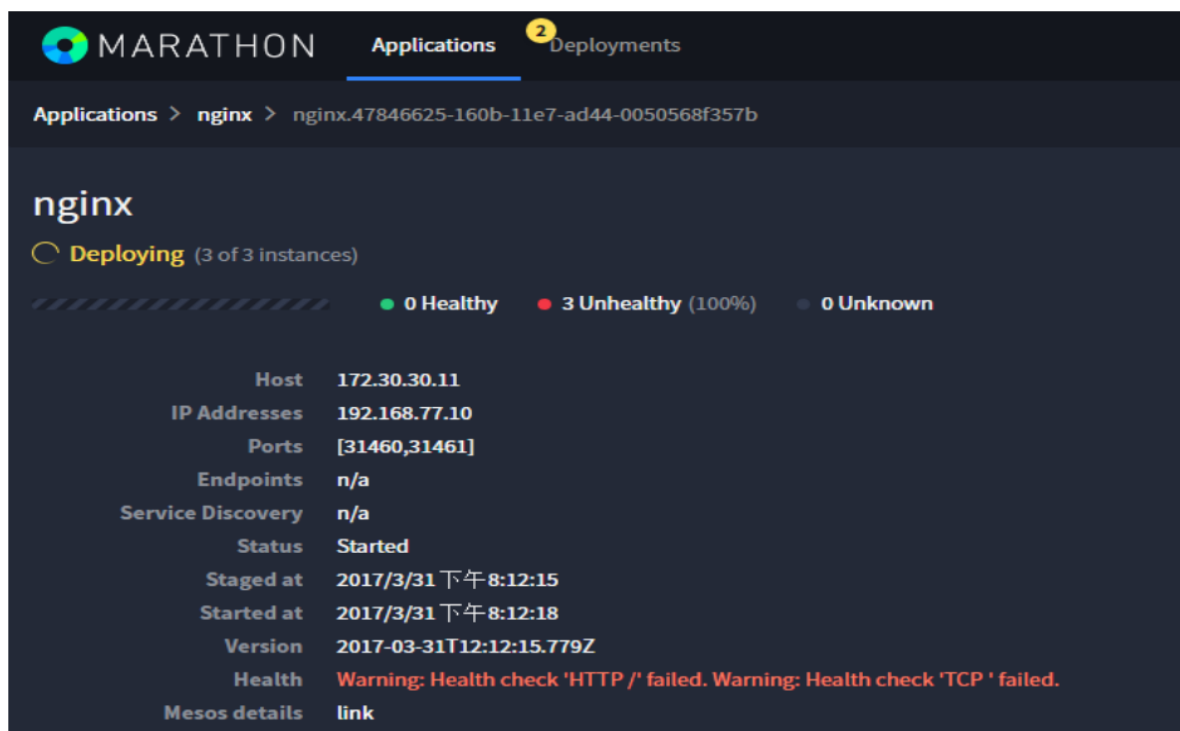


Figure 6.5 : Marathon Application Health Check while deploying

Mesos-level health checks (MESOS_HTTP, MESOS_HTTPS, MESOS_TCP, and COMMAND) are locally executed by Mesos on the agent running the corresponding task and thus test reachability from the Mesos executor.

Mesos-level health checks offer the following advantages over Marathon-level health checks:

- Mesos-level health checks are performed as close to the task as possible, so they are

not affected by networking failures.

- Mesos-level health checks are delegated to the agents running the tasks, so the number of tasks that can be checked can scale horizontally with the number of agents in the cluster.

Marathon application health check process can be seen in the below image.

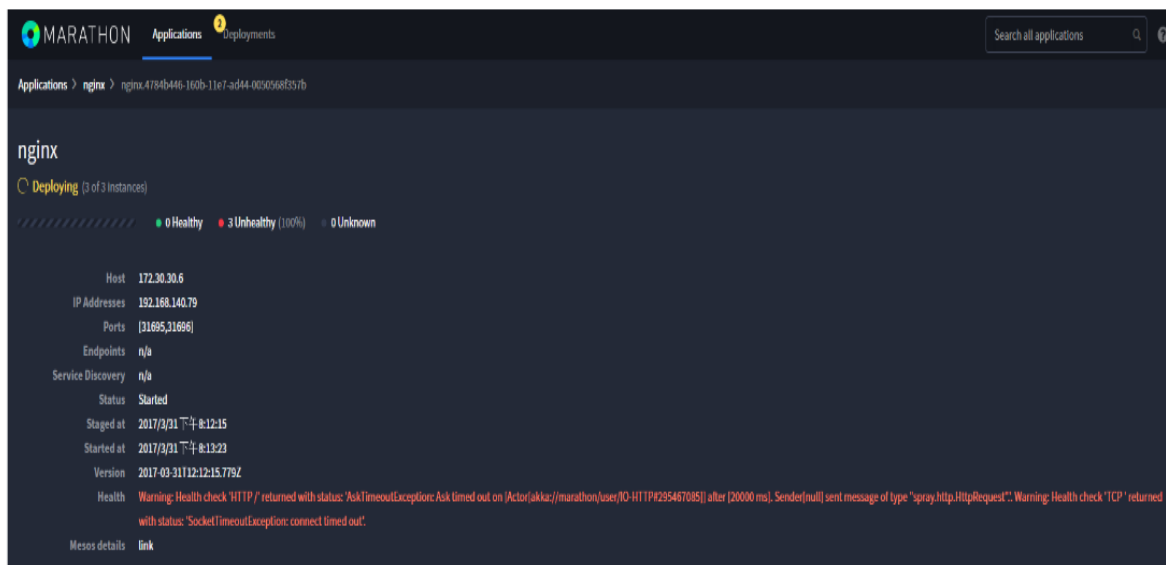


Figure 6.6 : Marathon Application Health Check

Limitations and considerations of the Marathon application health check

- Mesos-level health checks consume extra resources on the agents; moreover, there is some overhead for fork-execing a process and entering the tasks' namespaces every time a task is checked.
- The health check processes share resources with the task that they check. Your application definition must account for the extra resources consumed by the health checks.

Mesos slaves services are up and connected with the Master, containerized and non containerized applications deployments. It follows master and slave model where it includes specific number of masters and slaves.

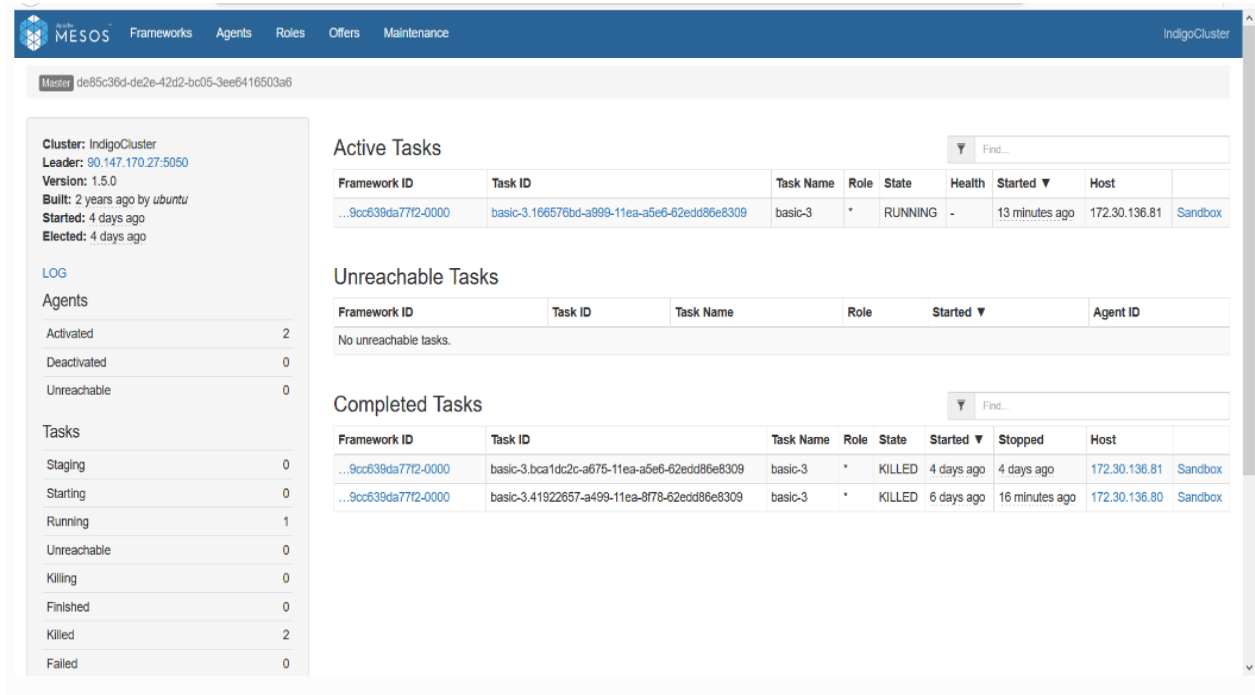


Figure 6.7 : Mesos slaves tasks completion

- Mesos provides a distributed, highly available architecture in which the Masters schedule work to be performed on the cluster, and slaves advertise available resources to the schedulers, which in turn execute tasks on the cluster.
- Mesos deployments compare and contrast with the traditional datacenter, and how an application-centric approach can lead to using resources more efficiently.

Scaling up Marathon applications's Dynamic scaling and predictive scaling of the applications deployed on marathon.

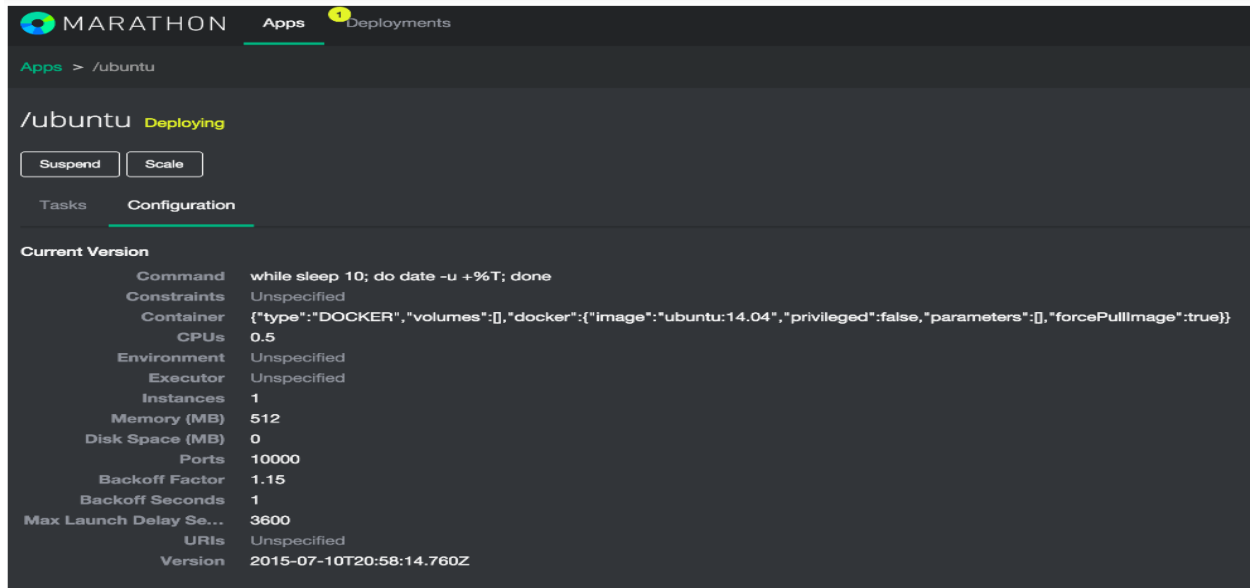


Figure 6.8 : Marathon Application Scaling process

- One of the many problems that SysOps and development teams have to tackle is application management.
- Marathon helps operations and development teams by providing a platform on which to run applications. Instead of the traditional approach of provisioning servers, deploying apps, and getting paged at 3 a.m.
- When a service goes down or when hardware fails, Marathon runs each instance of the application as a task on the Mesos cluster, automatically restarting the instance if it should fail. Marathon does this by ensuring that the workflow begins with the application, not with the server. Create the application, define the resources each instance will need, and specify the number of instances to run.

LDAP Authentication

The main features are Password checking against the external authentication engine. Automatic synchronization of usernames and emails.

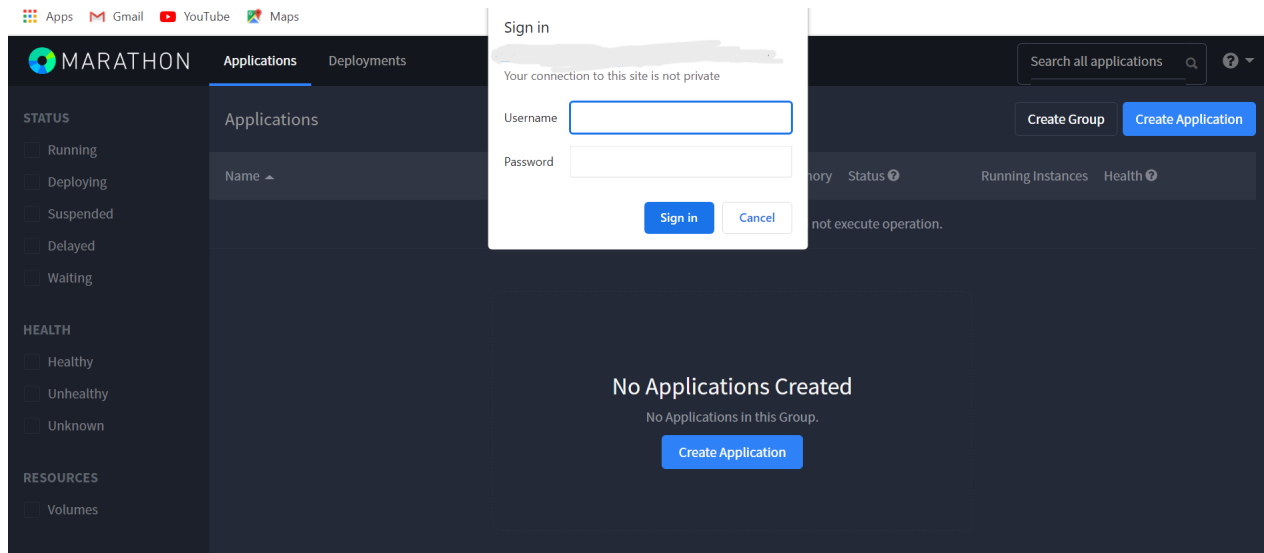


Figure 6.9 : LDAP authentication to access Marathon

A plugin for Mesosphere Marathon which authenticates users via UI/REST against an LDAP Server.

Features

- Allows authentication against predefined static users
- Supports LDAP and user memberships
- Allows membership/group level permissions support full CRUD against Apps and Groups including ID paths
- Tested against Marathon 1.1.1+
- Easy setup and JSON based configuration

Grafana Dashboards to visualize the business metrics Grafana is used as an open source tool which is used to analyze and visualize the metrics of real time data



Figure 6.10 : Grafana Dashboard

A dashboard is a set of one or more panels organized and arranged into one or more rows. Grafana ships with a variety of panels making it easy to construct the right queries, and customize the visualization so that it allows to create the perfect dashboard for requirement needs. Each panel can interact with data from any configured Grafana data source.

Dashboard snapshots are static . Queries and expressions cannot be re-executed from snapshots. As a result, if you update any variables in your query or expression, it will not change your dashboard data.

Managing dashboards

- Select a time period for a dashboard using the Time range controls in the upper right of the dashboard.
- Tag dashboards.
- Use templating to make them more dynamic and interactive.

- Use annotations to display event data across panels. This can help correlate the time series data in the panel with other events.
- Use the dashboard picker for quick, searchable access to all dashboards in a particular organization.



Figure 6.11 Working with Grafana Dashboard UI

- Zoom out time range (1)
- Time picker dropdown (2). Access relative time range options, auto refresh options and set custom absolute time ranges.
- Manual refresh option (3) Fetch new data.
- Dashboard panel (4) Click the panel title to edit panels.
- Graph legend (5) Change series colors, y-axis and series visibility directly from the legend.



Figure 6.12 Grafana Dashboard Header

- Side menu bar toggle (1): This option toggles the side menu. It provides access to features

unrelated to a dashboard such as users, organizations, data sources, and alerting.

- Dashboard dropdown (2): Use this option to view the current dashboard name. From here, you can:
 - Select another dashboard name to easily switch to that dashboard.
 - Create a new dashboard or folder, import existing dashboards, and manage dashboard playlists.
- Add panel (3): Use this option to add a new panel to the current dashboard.
- Star dashboard (4): Use this option to star (or unstar) the current dashboard. Starred dashboards show up on your own home dashboard by default. It is a convenient way to mark Dashboards that you're interested in.
- Share dashboard (5): Use this option to share the current dashboard by creating a link or create a static snapshot of it. You must save the dashboard before sharing.
- Save dashboard (6): Use this option to save the current dashboard using its current name.
- Settings (7): Use this option to manage dashboard settings and configure templates and annotations.

Grafana Alerts, If it matches the threshold it increases the count of active alerts and as soon as it is resolved, it will be moved to silenced alerts.

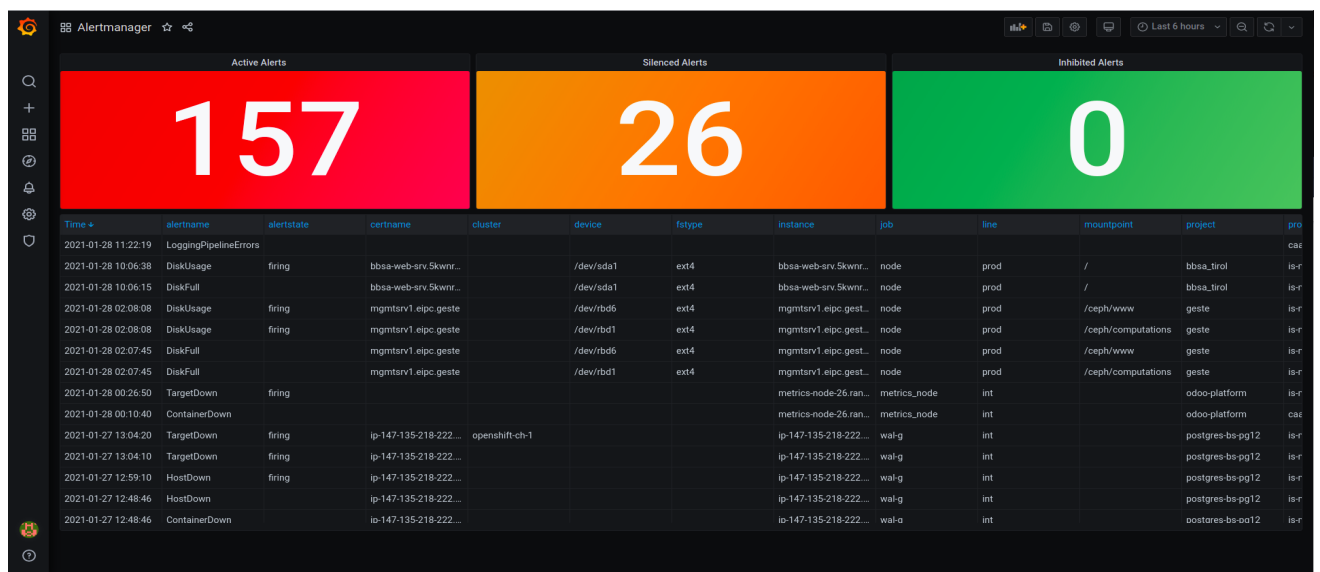


Figure 6.13 : Active and silenced alerts of Grafana

- Alerts can be set using Grafana alert feature, also it can have alerts sent through a number of different alert notifiers, including PagerDuty, SMS, email, VictorOps, OpsGenie, or Slack.
- In the grafana setup, It allows to create alerts such as Active and silence alerts.
- These alerts are helpful for the operational engineers because it is always difficult to monitor each and every server box to check about their performances and their health.
- These alerts help in such a way that once the set threshold is reached it will alert the team through the opted channels it can be mail or notification as required for the user.

From the alerts the operational engineers solves the issues such as storage on some

1. Server box storage limit reached
2. It can be server failure
3. Load balancing issues

The engineers solve the issues as tickets or incidents and make the active alert as a silenced alert.

Chapter 7 :Testing

Software testing is done to see if the system's actual result matches the expected result and to make sure the system is free of flaws. This activity is critical in ensuring that the client receives defect-free software. In order to accomplish this, the developed software is run and the system's functionality is tested. This aids in the detection of code errors and bugs, as well as filling in the gaps between missing functionalities.

Test Cases

Test cases define the set of conditions that a tester will use to determine whether or not a system under test meets the requirements and functions correctly [15]. The Test Case Template contains the following information: Test Case ID, Test Case Summary, Sample Input, Expected Output, and System Performance Remarks.

Unit testing is the process of testing each module of a system to ensure that it functions properly. Each core unit of the system is validated by testing individual units or components of the modules. Unit testing is done during the development phase, and it isolates the code section and verifies its correctness. After each unit of the system has been combined and tested as a group, integration testing is performed. This is done to expose any defects in the interfaces and to determine whether the system's components are properly integrated.

7.1 Unit Testing

Unit testing examines a single aspect of the application's logic [14]. A unit test does not test how the code interacts with dependencies or infrastructure, nor does it test the framework in which the code is written; it should be assumed that it works, or if it does not, a bug should be filed and a workaround coded. The entire unit test is run in the process and memory. It has no network, file system, or database communication. Individual developers write code, and unit tests are used to test it. This project employs the Python UNITTEST and Pyunit for unit testing.

Table 7.1 : Unit Testing for Statistical Analysis of Jobs using Mesos Master and Slave

Test Case ID	Description	Input	Expected Output	Actual Output	Remarks
01.	Validation of connection between Master and Slave	Mesos slave services up	Mesos Masters able to communicate with the slaves.	Slaves are not connected with master and not able to perform required tasks.	Fail
02.	Validation of connection between Master and Slave	Mesos slave services down	Mesos Masters able to communicate with the slaves.	Slaves are connected with master and able to perform required tasks.	Pass
03.	Verification of Master submitting jobs to slaves	submitting jobs or tasks	Report a set of statistics and metrics	Status of the frameworks and tasks running on a Mesos cluster	Pass

- The validation between the Master and slaves are important because if the connection fails then there should be a quicker resolution to that issue else it fails to achieve the required tasks.
- Zookeeper plays an important role in such event as because it is the brain of the infrastructure which helps to perform the leader election when the leader that is master failure
- Zookeeper uses leader election algorithms to choose the leader between the masters and then it advertises the elected leader to the slaves

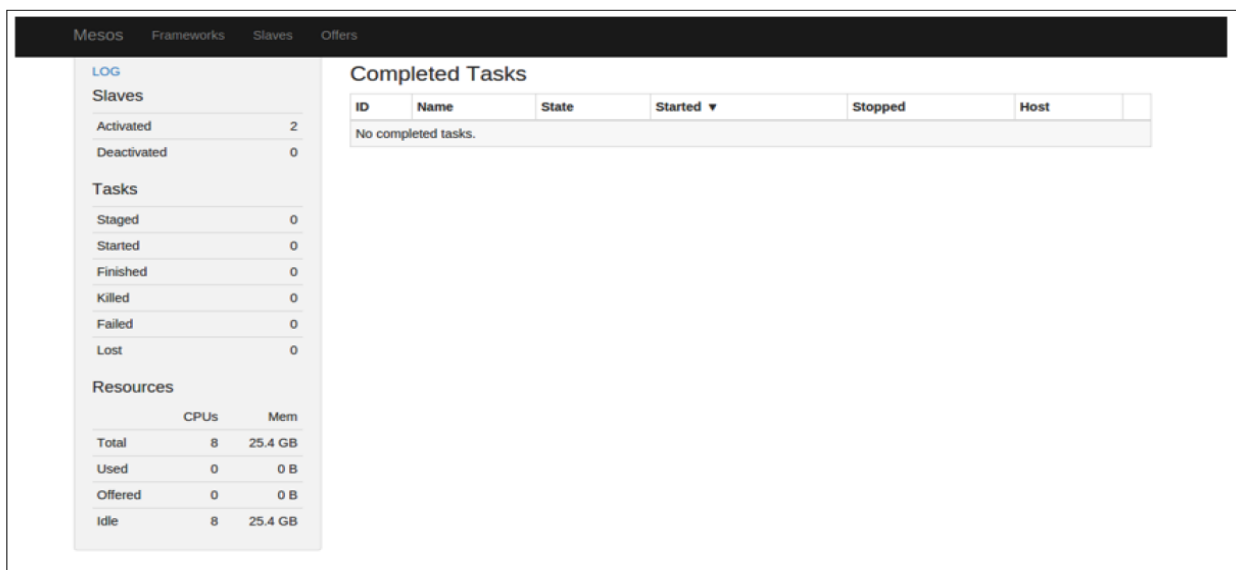


Figure 7.1 Testing the coordination between Master and slave

- When Mesos slaves advertise available resources to the master, the master makes some decisions based on the fair share of resources assigned to the registered frameworks.
- The resources are then offered to a framework's scheduler, which includes the logic for accepting or declining resource offers. If a resource offer is accepted, the scheduler is responsible for launching the executors on the slaves, which in turn launch the framework's tasks.
- This is the main reason to perform the testing between the coordination of mesos master and slaves is important

7.2 Boundary value Testing:

Boundary testing is an important aspect of testing which helps to process the testing between the extreme ends or it can be boundaries of the project between the partition of the specified input values.

Table 7.2 : Boundary value testing for connection to the LAN

Test case ID	Description	Input	Expected Output	Actual Output	Remarks
01.	Perform attack on the system which is connected to different LAN	ICMP packets	Directed to the false websites placed.	System is out of Range	Fail
02.	Perform attack on the system which is connected to same LAN	ICMP Packets	Directed to the false websites placed.	User is redirected to fraudulent site	Pass

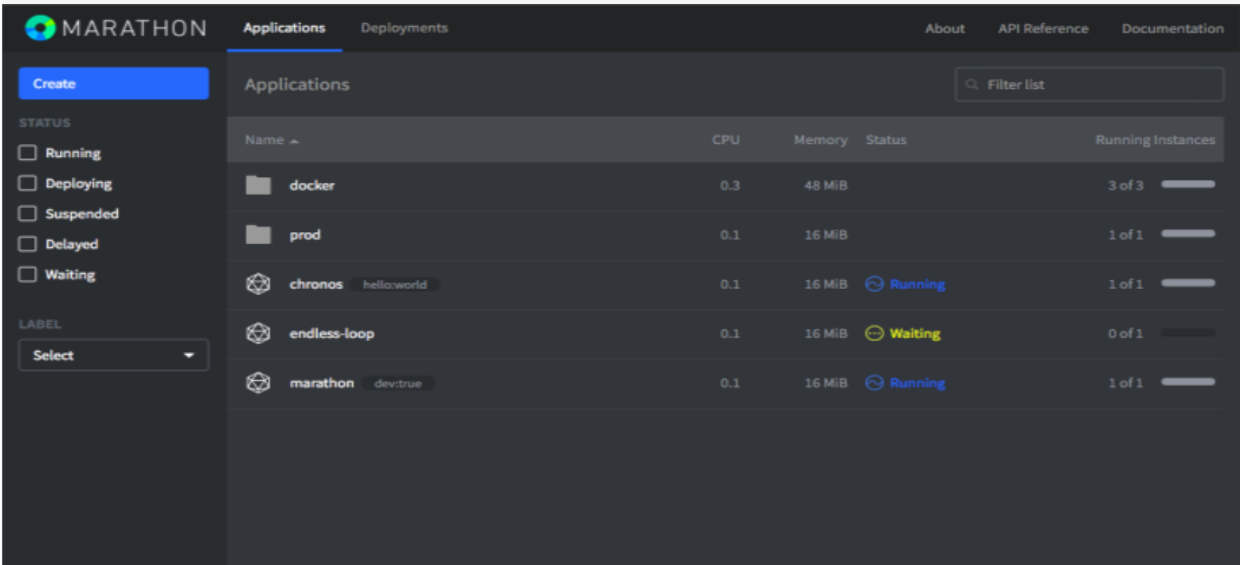
7.3 Integration Testing:

Integration testing provides a different insight for testing the project because in this testing type here the individual component is not tested testing is between a combination of modules or products. The purpose of the integration testing provides the exposure about the faults in the system and interaction between integrated units.

Table 7.3 : Integration Testing for Deployment of Applications using Marathon and LDAP Authentication

Test Case id	Description	Input	Expected output	Actual output	Remarks
01.	Deployment of Marathon LB application in Marathon	Marathon LB deployed with required parameters	Application deployed successfully and Running	Application deployed successfully and went to waiting state	Fail
02.	Deployment of applications in Marathon	Application that needs to be deployed	Application deployed successfully and Running	Application deployed successfully and Running	Pass.
03	Check for Index.html is working	Entering localhost IP in browser.	Index.html successfully hosted on localhost	Index.html successfully hosted on localhost	Pass
04	LDAP authentication access	Active Directory credentials	Grant access to Marathon	LDAP does not grants access to Marathon	Fail
05	LDAP authentication access	Active Directory credentials	Grant access to Marathon	LDAP does grants access to Marathon	pass

- The testing of deployment phase for Marathon applications is vital because if the application moves to a waiting or delayed state then the application will never get deployed. So the constant monitoring of the application deployment is important.
- The Marathon applications follow the application lifecycle in between that application health checks are mandatory for this aspect the testing is necessary for Marathon application deployments.



The screenshot shows the Marathon web interface. On the left, there's a sidebar with a 'Create' button and a 'STATUS' filter section containing checkboxes for Running, Deploying, Suspended, Delayed, and Waiting. Below that is a 'LABEL' section with a 'Select' dropdown. The main area is titled 'Applications' and contains a table with columns: Name, CPU, Memory, Status, and Running Instances. The table lists five applications: 'docker' (3 of 3 running), 'prod' (1 of 1 running), 'chronos' (1 of 1 running), 'endless-loop' (0 of 1 waiting), and 'marathon' (1 of 1 running). Each application has a small icon and a label next to its name.

Name	CPU	Memory	Status	Running Instances
docker	0.3	48 MiB		3 of 3
prod	0.1	16 MiB		1 of 1
chronos	0.1	16 MiB	Running	1 of 1
endless-loop	0.1	16 MiB	Waiting	0 of 1
marathon	0.1	16 MiB	Running	1 of 1

Figure 7.2 Status of deployed apps in Marathon

- From the above picture we can see the status of the deployed applications in Marathon, In which some applications are in waiting state and some applications are running successfully.
- The applications which are running successfully depict that the application has gone through all the checks prescribed and the application got deployed successfully.

6.4 System Testing:

System testing is a level of testing that validates a complete and fully integrated software product. The purpose of this kind of system testing is always to evaluate the end-to-end system specifications of the project and provide the results for the same.

Table 7.4 : System Testing Visualization of Metrics and Nginx Reverse proxy

Test case ID	Description	Input	Expected Output	Actual Output	Remarks
01	Nginx Reverse proxy	Client Request	Response to the request and redirect to the site	It redirects to the site with port number on which the site is running	Fail
02	Nginx Forward proxy	Client Request	Response to the request	Hides the identities of servers and responds to the client request	Pass
03	Grafana metrics active alert	Influxdb and set alert	If it matches the threshold, Increase the count of active alert	Active alert count is increased in the grafana dashboard	Pass
04	Grafana metrics silence alert	Make the required changes to the system	Silence the alert	Active alert did not get resolved and silence alert count did not get incremented	Fail
05	Grafana metrics silence alert	Make the required changes to the system	Silence the alert	Active alert got resolved and silence alert count got incremented	Pass

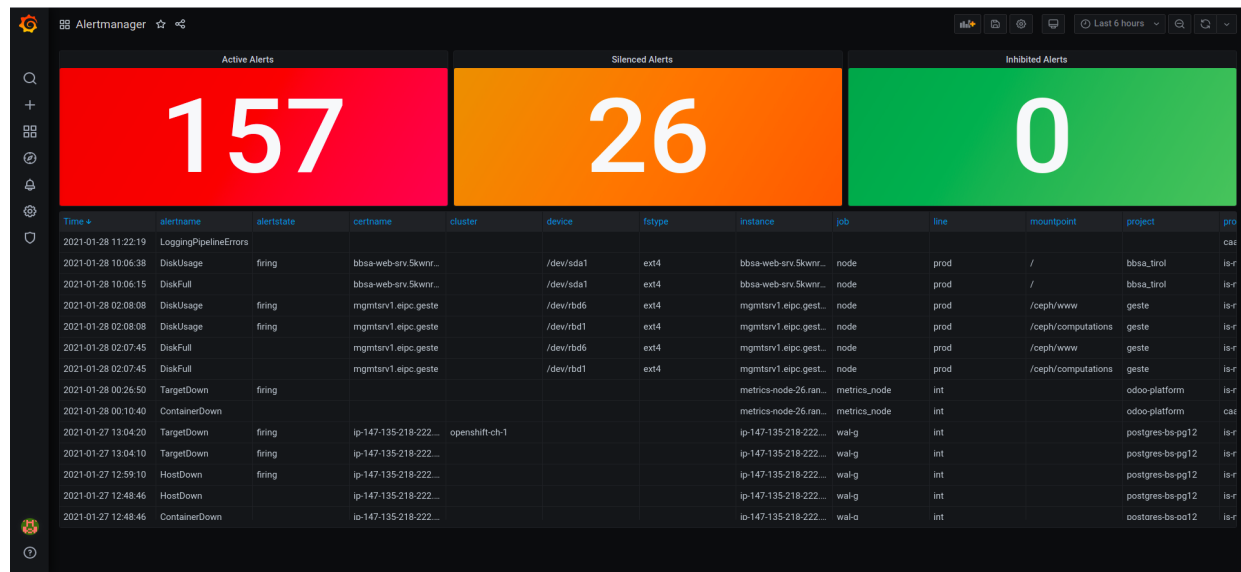


Figure 7.3 Alert testing in Grafana

- Alerts can be set using Grafana alert feature, also it can have alerts sent through a number of different alert notifiers, including PagerDuty, SMS, email, VictorOps, OpsGenie, or Slack
- Grafana Alerts, If it matches the threshold it increases the count of active alerts and as soon as it is resolved it will be moved to silenced alerts.
- These alerts help in such a way that once the set threshold is reached it will alert the team through the opted channels it can be mail or notification as required for the user. From the alerts the operational engineers solves the issues such as storage on some server box storage limit reached, It can be server failure or Load balancing issues
- The engineers solve the issues as tickets or incidents and make the active alert as a silenced alert.

Chapter 8 : Conclusion

Both Kubernetes and Mesos aim to simplify the deployment and management of applications in data centers or cloud containers. Both provide some support for businesses of various sizes. Ultimately, choose a cluster management solution that suits the unique needs of your current and future business.

- Mesos combines master and slave to report details about available resources, used resources, registered frameworks, active agents, and task status.
- Marathon framework provided the interface to test and deploy applications in distributed environments which leads to scale them up as per the requirement.
- LDAP searches for group and consumer mapping and it has the ability to authenticate based on username or custom ID to bind to an enterprise LDAP directory with a password.
- The Nginx reverse proxy provides an additional layer of abstraction and control to ensure a smooth flow of network traffic between the client and server by hiding the server details.
- The identified Container orchestration platform Mesos is built with a huge ecosystem that can improve the business productivity and it is cheaper than its alternatives.
- The infrastructure setup of Mesos and Marathon framework orchestrates computing, networking, and storage infrastructure on behalf of the user.
- Monitoring tool Grafana used to provide alerts in centralized log management, analytics and interactive visualization of web application of business metrics.
- Puppetizing the setup helped in automating and centralizing the configuration management process and also saved time by increasing deployment speed.

There has been a rapid increase in product development after the big organizations started using devops methodology, they have been constantly adding features to the products

Chapter 9 : Future Enhancements

This project aims at a distributed environment, monitoring different parts of the system and ensuring they are working properly is an extremely hard problem.

The future enhancement to the project would be adding the cloud-native approach with the microservices to take advantage of the cloud computing model. Implementation of Kubernetes and Docker to create a continuous delivery configuration for building microservices.

Using Ansible for better automating the process of development and saving time. Building a production friendly Multi-Node ZooKeeper cluster environment development. Also the more optimal quorum setup which helps to achieve high availability which is important in the modern distributed systems

Bibliography

- [1] Andrea Tosatto; Pietro Ruiu; Antonio Attanasio, 'Container-Based Orchestration in Cloud: State of the Art and Challenges' July 2021
- [2] Yao Pan; Ian Chen, Richard Sinnott, 'A Performance Comparison of Cloud-Based Container Orchestration Tools' 2nd International Conference on Computing and Communications Technologies (ICCT) Nov 2020
- [3] Eddy Truyen, Matt Bruzek, Wouter Joosen, 'Evaluation of Container Orchestration Systems for Deploying and Managing Database Clusters' International Conference on Computer Communication and Distributed env(ICCCI) July 2020
- [4] Marek, Martin Kontsek, 'Overview of Docker container orchestration tools' International Conference on Distributed Systems Nov 2019
- [5] Madhusudhan,Angel beltre, 'Exploring the Fairness and Resource Distribution in an Apache Mesos Environment' International Conference on Computer Science and Information Processing (CSIP) July 2018
- [6] Xiaolian Li, Yuling Jiang, 'Application Research of Docker Based on Mesos Application Container Cluster' 2nd International Conference on Advanced Computer Control Nov 2018
- [7] Mingmin Zhang; Mingliang Xu, 'The framework and implementation of Virtual Network Marathon' International Conference on Computer Science and Network Technology International Conference on Computing and Communication Technologies Mar 2017
- [8] Lipika Bose Goel, Rana Majumdar, 'Handling mutual exclusion in a distributed application through Zookeeper' IEEE 3rd International Conference on Communication Software and Networks Mar 2016
- [9] Paul Le, Alexandru Costan, Luc Bougé, ' A performance evaluation of Apache Kafka in support of big data streaming applications' Mar 2015

- [10] Viktor Walter-Tscharf, 'Practical Approach to Monitor Runtime Engine Statistics for Apache Spark and Docker Swarm, Grafana' International Conference on Advanced Computing and security control Dec 2014
- [11] N. Ram Ganga Charan, S. Tirupati Rao, Dr .P.V.S Srinivas, Deploying an Application on the Cloud (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No. 5, 2011
- [12]. F. Belqasmi, J. Singh, S. Y. B. Melhem and R. H. Glitho, "SOAP-Based Web Services vs. RESTful Web Services for Multimedia Conferencing Applications: A Case Study," IEEE Internet Computing, vol. 16, issue 4, pp. 54–63, July–Aug. 2012
- [13]. S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, "Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors," SIGOPS Oper. Syst. Rev., vol. 41, no. 3, Mar. 2007, pp. 275–287
- [14] Krishna, "What is Software Testing? Introduction, Definition, Basics & Types. Internet: <https://www.guru99.com/software-testing-introduction-importance.html> , 2020".
- [15] Krishna, "Unit Testing tutorial: What is, Types, tools, EXAMPLES, <https://www.guru99.com/unit-testing-guide.html> , 2020".
- [16] Simos Gerasimou, Maria Kechagia, Dimitris Kolovos, Richard Paige and Georgios Gousios, "On Software Modernisation due to Library Obsolescence", IEEE/ACM 2nd International Workshop on API Usage and Evolution (WAPI) 2018
- [17] Nasreen Sultana Quadri;Kusum Yadav, 'Performance Evaluation of AWS and IBM Cloud Platforms for Security Mechanism' International Conference on Computer Communication and Informatics (ICCCI) Dec 2018
- [18] John Ruther, Franklin clark , 'When HTTPS Meets CDN, A case of authentication in Delegated service' International Conference on Information Systems June 2017
- [19] N. Dragoni et al., "Microservices: yesterday, today, and tomorrow," ArXiv160604036 Cs, June 2016.

- [20] M. Villamizar et al., “Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures,” *Serv. Oriented Comput. Appl.*, vol. 11, no. 2, pp. 233–247, Jun. 2017.
- [21] Lipika Bose Goel; Rana Majumdar: “Handling mutual exclusion in a distributed application through Zookeeper”, 2013 International Conference on Advances in Computer Engineering and Applications, 2013, pp. 457-460, doi: 10.1109/ICACEA.2013.7164748.
- [22] Ailidani Ailijiang, Aleksey Charapko, Murat Demirbas, Bekir O. Turkkan, Tevfik Kosar: “Efficient Distributed Coordination at WAN-Scale”, 2011 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), 2011, pp. 1575-1585, doi: 10.1109/ICDCS.2011.274.
- [23] Yunli Cheng: “Architecture Design and Research Based on Distributed Configuration Center”, 2019 International Conference on Electronic Engineering and Informatics (EEI), 2019, pp. 247-250, doi: 10.1109/EEI48997.2019.00061.
- [24] M. Song, G. Luo and E. Haihon: “A Service Discovery System based on Zookeeper with Priority Load Balance Strategy”, 2019 IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC), 2019, pp. 117-119, doi: 10.1109/ICNIDC.2019.7974547.
- [25] M. Muneotmo and T. Abe: “Designing a Distributed Design Exploration Framework in the Inter-cloud Environment”, 2017 IEEE 8th International Conference on Cloud Computing, 2017, pp. 1073-1076, doi: 10.1109/CLOUD.2017.155.
- [26] Pingting Hao, Liang Hu, Jingyan Jiang, Xilong Che: “A Stochastic Adjustment Strategy For Coordination Process In Distributed Networks”, *Computing and Informatics*, 37(5), pp.1184-1208.(2016)

- [27] S. Skeirik, R. B. Bobba and J. Meseguer: "Formal Analysis of Fault-tolerant Group Key Management Using ZooKeeper", 2016 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, 2016, pp. 636-641, doi: 10.1109/CCGrid.2016.98.
- [28] Raluca Halalai, Pierre Sutra, Étienne Rivière, Pascal Felber: "ZooFence: Principled Service Partitioning and Application to the ZooKeeper Coordination Service", 2014 IEEE 33rd International Symposium on Reliable Distributed Systems, 2014, pp. 67-78, doi: 10.1109/SRDS.2014.41
- [29] The configuration tutorial provides step-by-step instructions for using the DSL: <http://aurora.apache.org/documentation/latest/configuration-tutorial>.
- [30] The configuration reference provides detailed information about the entire schema: <http://aurora.apache.org/documentation/latest/configuration-reference>.
- [31] Jiongjiong Gu Streamlining IT Agile Business Way-Preview of Enterprise Cloud Computing IT Infrastructure Platform Architecture [Z] Huawei 2013.
- [32] Baohua Yang Wangjian Dai and Yalun Cao "Introduction and Practice of Docker Technology [M]" Mechanical Industry Press 2015.
- [33] Junfeng Bian "Design and Implementation of Docker-based Resource Scheduling and Application Container Cluster Management System" Shandong University 2017.
- [34] Jing Huo Jingyan Shi Gongxing Sun and Bowen Kan Optimization of BESIII Cluster Resource Management by an Improved DRF Algorithm [Journal Papers]-nuclear electronics and Detection Technology no. 10 2014.
- [35] Zunwang Ke Jiong Yu and Bin Liao "DRF Enhancement Algorithm for Mesos Multi-Resource Scheduling Adapted to Heterogeneous Clusters" Journal Papers] Computer Applications 2016.

- [36] Shaogang Zhang and Haiyun Ma "Code for Software Engineering and Graduation Design [M]" National Defense Industry Press 2015.
- [37] Yiming Chen Xiaoqiang Kou and Yongli Wang "Design and Implementation of Docker-based Vulnerability Verification Framework" Electronic Technology Applications no. 11 2018.
- [38] Z. Li et al., "Performance Overhead Comparison between Hypervisor and Container Based Virtualization", in IEEE Int. Conf. on Advanced Information Networking and Applications (AINA), 2017, pp. 955-962.
- [39] R. Dua, A. R. Raja, and D. Kakadia, "Virtualization vs Containerization to Support PaaS", in 2014 IEEE Int. Conf. on Cloud Engineering, 2014.
- [40] R. Morabito et al., "Evaluating Performance of Containerized IoT Services for Clustered Devices at the Network Edge", IEEE Internet of Things Journal, vol. 4, pp. 1019-1030, 2017.
- [41] R. Morabito, "A performance evaluation of container technologies on Internet of Things devices", in 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WORKSHOPS), pp. 999-1000.
- [42] Z. Nikdel et al., "DockerSim: Full-stack simulation of container-based Software-as-a-Service (SaaS) cloud deployments and environments", in 2017 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), 2017, pp. 1-6.

DR SAK_Shivaji

ORIGINALITY REPORT

18%

SIMILARITY INDEX

14%

INTERNET SOURCES

5%

PUBLICATIONS

5%

STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	3%
2	J. W. Mar, L. A. Schmit. "Some Structural Penalties Associated With Thermal Flight", Journal of Fluids Engineering, 1957 Publication	3%
3	docs.huihoo.com Internet Source	3%
4	www.ijres.org Internet Source	1%
5	en.wikipedia.org Internet Source	1%
6	lume.ufrgs.br Internet Source	1%
7	Submitted to St Joseph Engineering College, Mangalore Student Paper	1%
8	docs.mesosphere.com Internet Source	1%