BENCHMARKING THE PERFORMANCE OF LIGHT WEIGHT CONTAINER (LXC)

A Project Report

Submitted by:

SAKSHI MISHRA (PU-011161517A12)

AVINASH KUMAR (PU-026161517A12)

In partial fulfilment for the award of the degree

Of

BACHELOR OF TECHONOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

at

SCHOOL OF RESEARCH AND TECHNOLOGY

PEOPLE'S UNIVERSITY

AYODHYA BYPASS ROAD, BHOPAL, MP (INDIA)-462037

(DECEMBER - 2018)

CERTIFICATE

This is to certify that the project titled **"Benchmarking the Performance of Light Weight Container (LXC)"** is the bona fide work carried out by **Sakshi Mishra(PU-011161517A12) and Avinash Kumar(PU-026161517A12),** students of B Tech (CSE) with specialization in Cloud Computing & Virtualization of SORT, People's University Bhopal during the academic year 2018-19, in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (Computer Science and Engineering)

Guided By	**Approved By**	**Forwarded By**
Mr. Anjul Rai **Assistant Professor** IBM-ICE Programme	**Mr. Ankit Dwivedi** In-Charge IBM-ICE Programme People's University	**Dr. Anshuman Sharma** Principal SORT, People's University Bhopal

DECLARATION

We hereby declare that the project entitled **"Benchmarking the Performance of Light Weight Container (LXC)"** submitted for the B. Tech. (CSE-CCV) degree is our/my original work and the project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles.

Place: Bhopal **Signature of the Student**

Date:

Sakshi Mishra (PU-011161517A12)

Avinash Kumar (PU-026161517A12)

ACKNOWLEDGEMENT

We take this occasion to thank God, almighty for blessing us with his grace and taking our endeavor to a successful culmination. We extend our sincere and heartfelt thanks to our esteemed guide, **Mr. Anjul Rai**, for providing us with the right guidance and advice at the crucial junctures and for showing me the right way. We also take this opportunity to express a deep sense of gratitude to our class coordinator **Ms. Priyanka Pande** for their cordial support, valuable suggestions and guidance. We extend our sincere thanks to our respected **Head of the division Mr. ANKIT DWIVEDI**, for allowing us to use the facilities available. We would like to thank the other faculty members also, at this occasion. Last but not the least, we would like to thank our friends and family for the support and encouragement they have given us during the course of our work.

ABSTRACT

Benchmarking the performance of Lightweight Container (LXC) is the project that aims to carry out performance analysis of LXC .This project is used to create light weight monitoring system for the Docker Host using Grafana and Prometheus. Project investigation aims to answer which scenarios LXC can offer a better performance than hypervisor-based virtualization or even equal to native environments for HPC applications. For this, we have conduct experiments with traditional benchmarks considering different VM allocation possibilities: many VMs on the same host competing or coopering for resource usage, and VMs allocated in distinct hosts without resource sharing. In this work, we analyze the performance of a container-based virtualization solution - Linux Container (LXC)

Against a hypervisor-based virtualization solution - KVM - under HPC activities. For our experiments, we consider CPU and communication (network and inter-process communication) performance, and results show the hypervisor type can impact distinctly in performance according to resource used by application.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
GLANCE ON PROJECT

1.1 INTRODUCTION

Cloud computing makes extensive use of virtual machines (VMs) because they permit workloads to be isolated from one another and for the resource usage to be somewhat controlled. However, the extra levels of abstraction involved in virtualization reduce workload performance, which is passed on to customers as worse price/performance. New advances in container-based virtualization simplify the deployment of applications while continuing to permit control of the resources allocated to different applications.

In this project, we explore the performance of traditional virtual machine deployments, and contrast them with the use of Linux containers. We use a suite of workloads that stress CPU, memory, storage, and networking resources. We use XenServer as a representative hypervisor and Docker as a container manager. Our results show that containers result in equal or better performance than VMs in almost all cases. Both VMs and containers require tuning to support I/O-intensive applications. We also discuss the implications of our performance results for future cloud architectures [3].

Virtual machines are used extensively in cloud computing. In particular, the state-of-the-art in Infrastructure as a Service (IaaS) is largely synonymous with virtual machines. Cloud platforms like Amazon EC2 make VMs available to customers and also run services like databases inside VMs. Many Platform as a Survive (PaaS) and Software as a Service (SaaS) providers are built on IaaS with all their workloads running inside VMs. Since virtually all cloud workloads are currently running in VMs, VM performance is a crucial component of overall cloud performance. Once a hypervisor has added overhead, no higher layer can remove it. Such overheads then become a pervasive tax on cloud workload performance. There have been many studies showing how VM execution compares to native execution and such studies have been a motivating factor in generally improving the quality of VM technology. Container-based on

Virtualization presents an interesting alternative to virtual machines in the cloud. Virtual Private Server providers, which may be viewed as a precursor to cloud computing, have used containers for over a decade but many of them switched to VMs to provide more consistent performance. Although the concepts underlying containers such as namespaces are well understood, container technology languished until the desire for rapid deployment led PaaS providers to adopt and standardize it, leading to a renaissance in the use of containers to provide isolation and resource control. Linux is the preferred operating system for the cloud due to its zero price, large ecosystem, good hardware support, good performance, and reliability. The kernel namespaces feature needed to implement containers in Linux has only become mature in the last few years since it was first discussed [1].

1.2 BACKGROUND OF PROJECT

Linux containers (LXCs) are rapidly becoming the new "unit of deployment" changing how we develop, package, deploy and manage applications at all scales (from test / dev to production service ready environments). This application life cycle transformation also enables fluidity to once frictional use cases in a traditional hypervisor Virtual Machine (VM) environment. For example, developing applications in virtual environments and seamlessly "migrating" to bare metal for production. Not only do containers simplify the workflow and life cycle of application development / deployment, but they also provide performance and density benefits which cannot be overlooked [1].

At the forefront of Linux Container tooling we have Docker -- a LXC framework / runtime which abstracts out various aspects of the underlying realization by providing a pluggable architecture supporting various storage types, LXC engines / providers, etc.. In addition to making LXC dead easy and fun, Docker also brings a set of capabilities to the table which make containers more productive including; automated builds (make files for LXC images), versioning support, fully featured REST API + CLI, the notion of image repositories and more.

Before going any further, let me reiterate some of the major benefits of Linux Containers from a Docker perspective:

- Fast
  - Runtime performance at near bare metal speeds (typically 97+ percent or bare metal -- a few ticks shaven off for bean counters).
  - Management operations (boot, stop, start, reboot, etc.) in seconds or milliseconds.
- Agile
  - VM-like agility -- it's still "virtualization".
  - Seamlessly move between virtual and bare metal environments permitting new development workflows which reduce costs (e.g. develop on VMs and move to bare metal in the "click of a button" for production).
- Flexible
  - Containerize a "system" (OS less the kernel).
  - Containerize "application(s)".
- Lightweight
  - Minimal per container penalty which equates to greater density and hence greater returns on existing assets -- imagine packing 100s or 1000s of containers on a single host node.
- Inexpensive
  - Open source -- free -- lower TCO.
  - Supported with out-of-the-box modern Linux kernels.
- Ecosystem
  - Growing in popularity -- just checkout the google trends for docker or LXC

Linux Containers take a completely different approach than system virtualization technologies such as VMWare, KVM and Xen, which started by booting separate virtual systems on emulated hardware and then attempted to lower their overhead via Para virtualization and related mechanisms. LXC started out with an efficient mechanism and added isolation, resulting in a system virtualization mechanism as scalable and portable as chroot, capable of simultaneously supporting thousands of emulated systems on a single server while also providing lightweight virtualization options to routers and smart phones. LXC, it later morphed into its own container runtime environment. At a high level, Docker is a Linux utility that can efficiently create, ship, and run containers.

CHAPTER 2
SYSTEM ANALYSIS

In this chapter, we will discuss about Benchmarking the performance of Lightweight Container (LXC) is the project that aims to carry out performance analysis of LXC The functional and non-functional requirements are included in SRS part to provide complete description and overview of system requirement before the developing process is carried out. Besides that, existing vs proposed provides a view of how the proposed system will be more efficient than the existing one.

2.1 PROJECT OBJECTIVE

Modern cloud infrastructure uses virtualization to isolate applications, optimize the utilization of hardware resources and provide operational edibility. However, conventional virtualization comes at the cost of resource overhead. Container-based virtualization could be an alternative as it potentially reduces overhead and thus improves the utilization of datacenters. This paper presents the results of a Marco-benchmark performance comparison between the two implementations of these technologies, namely Xen and LXC, as well as a discussion on their operational ex-ability. These days, most cloud computing datacenters run hypervisors on top of their physical machines. A hypervisor is a piece of computer software that creates and runs virtual machines. With these hypervisors, and the virtual machines that run on them, system administrators are able to optimize the use of available physical resources and canine individual parts of application infrastructure. A typical setup is displayed schematically in

Figure 2.1 With the use of virtualization, resources can be consumed more electively than conventional bare-metal setups, which use physical machines for isolating deferent parts of application infrastructure. Still efficiency could be increased even further. A hypervisor will run multiple kernels on a single physical machine; therefore the isolation of applications and processes is expensive. Mills stated that 1,500 terawatt-hours of power per year is used to power

cloud computing datacenters, that is about 10% of the worlds energy consumption, and this number is climbing. If compute resources could be used more efficiently that could have a big impact.

Benchmarking the performance of Lightweight Container (LXC) is the project that aims to carry out performance analysis of LXC .This project is used to create light weight monitoring system for the Docker Host using Grafana and Prometheus. Project investigation aims to answer which scenarios LXC can offer a better performance than hypervisor-based virtualization or even equal to native environments for HPC applications. For this, we have conduct experiments with traditional benchmarks considering different VM allocation possibilities: many VMs on the same host competing or coopering for resource usage, and VMs allocated in distinct hosts without resource sharing. In this work, we analyze the performance of a container-based virtualization solution - Linux Container (LXC) - against a hypervisor-based virtualization solution - KVM - under HPC activities. For our experiments, we consider CPU and communication (network and inter-process communication) performance, and results show the hypervisor type can impact distinctly in performance according to resource used by application[1][2][9].

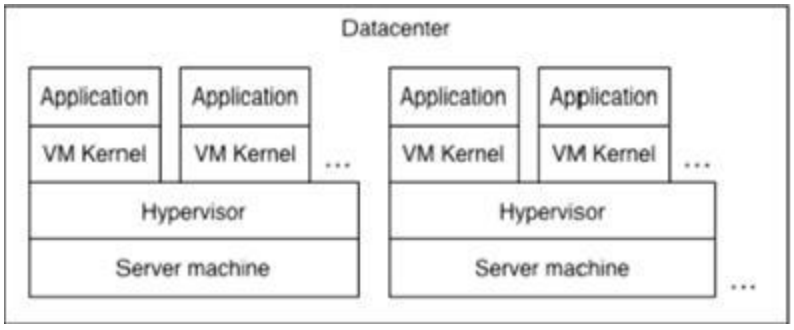

Figure 2.1. A schematic overview of virtual machines in a datacenter.

2.2 XEN SERVER

Xen is an open-source type-1 or baremetal hypervisor, which makes it possible to run many instances of an operating system or indeed different operating systems in parallel on a single machine (or host). Xen is the only type-1 hypervisor that is available as open source. Xen is used as the basis for a number of different commercial and open source applications, such as: server virtualization, Infrastructure as a Service (IaaS), desktop virtualization, security applications, embedded and hardware appliances. Xen enables users to increase server utilization, consolidate server farms, reduce complexity, and decrease total cost of ownership. Here are some of the Xen Project hypervisor's key features:

- Small footprint and interface (is around 1MB in size). Because it uses a microkernel design, with a small memory footprint and limited interface to the guest, it is more robust and secure than other hypervisors.
- Operating system agnostic: Most installations run with Linux as the main control stack (aka "domain 0"). But a number of other operating systems can be used instead, including NetBSD and Open Solaris.
- Driver Isolation: The Xen Project hypervisor has the capability to allow the main device driver for a system to run inside of a virtual machine. If the driver crashes, or is compromised, the VM containing the driver can be rebooted and the driver restarted without affecting the rest of the system.
- Para virtualization: Fully Para virtualized guests have been optimized to run as a virtual machine. This allows the guests to run much faster than with hardware extensions (HVM). Additionally, the hypervisor can run on hardware that doesn't support virtualization extensions [4][8].

2.3 PROMETHEUS & GRAFANA

Prometheus is an open-source system monitoring and alerting toolkit originally built at Sound Cloud. Since its inception in 2012, many companies and organizations have adopted Prometheus, and the project has a very active developer and user community. It is now a standalone open source project and maintained independently of any company. To emphasize this, and to clarify the project's governance structure, Prometheus joined the Cloud Native Computing Foundation in 2016 as the second hosted project, after Kubernetes.

Prometheus's main features are:

- a multi-dimensional data model with time series data identified by metric name and key/value pairs
- a flexible query language to leverage this dimensionality
- no reliance on distributed storage; single server nodes are autonomous
- time series collection happens via a pull model over HTTP
- pushing time series is supported via an intermediary gateway
- targets are discovered via service discovery or static configuration
- multiple modes of graphing and dash boarding support

Grafana is an open source, feature rich, powerful, elegant and highly-extensible analytics and monitoring software that runs on Linux, Windows and MacOS. It is a de facto software for data analytics, being used at Stack Overflow, eBay, PayPal, Uber and Digital Ocean – just to mention but a few.

It supports 30+ open source as well as commercial databases/data sources including MySQL, PostgreSQL, Graphite, and Elastic search, OpenTSDB, Prometheus and Influx DB. It allows you to dig deeply into large volumes of real-time, operational data; visualize, query, set alerts and get insights from your metrics from different storage locations.

Grafana is the leading graph and dashboard builder for visualizing time series infrastructure and application metrics, but many use it in other domains including industrial sensors, home

automation, weather, and process control. It provides a powerful and elegant way to create, explore, and share dashboards and data with your team and the world [6][13][11][10].

2.4 EXISTING VS PROPOSED SYSTEM

Existing benchmarking methods are time consuming processes as they typically benchmark the entire Virtual Machine (VM) in order to generate accurate performance data, making them less suitable for real-time analytics. The research in this paper is aimed to surmount the above challenge by presenting DocLite - Docker Container-based Lightweight benchmarking tool. DocLite explores lightweight cloud benchmarking methods for rapidly executing benchmarks in near real-time. DocLite is built on the Docker container technology, which allows a user-defined memory size and number of CPU cores of the VM to be benchmarked. The tool incorporates two benchmarking methods - the first referred to as the native method employs containers to benchmark a small portion of the VM and generate performance ranks, and the second uses historic benchmark data along with the native method as a hybrid to generate VM ranks. The proposed methods are evaluated on three use-cases and are observed to be up to 91 times faster than benchmarking the entire VM. In both methods, small containers provide the same quality of rankings as a large container. The native method generates ranks with over 90% and 86% accuracy for sequential and parallel execution of an application compared against benchmarking the whole VM. The hybrid method did not improve the quality of the rankings significantly. Linux Containers take a completely different approach than system virtualization technologies such as VMWare, KVM and Xen, which started by booting separate virtual systems on emulated hardware and then attempted to lower their overhead via Para virtualization and related mechanisms. LXC started out with an efficient mechanism and added isolation, resulting in a system virtualization mechanism as scalable and portable as chroot, capable of simultaneously supporting thousands of emulated systems on a single server while also providing lightweight virtualization options to routers and smart phones. LXC, it later morphed into its own container runtime environment. At a high level, Docker is a Linux utility that can efficiently create, ship, and run containers.

CHAPTER 3
PROJECT DESIGN

3.1 VMWARE

VMware is a company that was established in 1998 and provides different software and applications for virtualization. It has become one of the key providers of virtualization software in the industry. VMware's products can be categorized in two levels: desktop applications and server applications.

VMware was founded in 1998 by five different IT experts. The company officially launched its first product, VMware Workstation, in 1999, which was followed by the VMware GSX Server in 2001. The company has launched many additional products since that time.

VMware's desktop software is compatible with all major OSs, including Linux, Microsoft Windows, and Mac OS X. VMware provides three different types of desktop software:

- VMware Workstation: This application is used to install and run multiple copies on instances of the same operating systems or different operating systems on a single physical computer machine.
- VMware Fusion: This product was designed for Mac users and provides extra compatibility with all other VMware products and applications.
- VMware Player: This product was launched as freeware by VMware for users who do not have licensed VMWare products. This product is intended only for personal use.

VMware's software hypervisors intended for servers are bare-metal embedded hypervisors that can run directly on the server hardware without the need of an extra primary OS. VMware's line of server software includes:

- VMware ESX Server: This is an enterprise-level solution, which is built to provide better functionality in comparison to the freeware VMware Server resulting from a lesser system overhead. VMware ESX is integrated with VMware vCenter that provides additional solutions to improve the manageability and consistency of the server implementation.

- VMware ESXi Server: This server is similar to the ESX Server except that the service console is replaced with BusyBox installation and it requires very low disk space to operate.

VMware Server: Freeware software that can be used over existing operating systems like Linux or Microsoft Windows [2].

3.2 XENSERVER DESIGN

Citrix XenServer is an open source server virtualization platform based on the Xen hypervisor. Citrix also offers a supported version that you can purchase, with two options: Standard and Enterprise.

This platform is used by virtualization administrators to deploy, host and manage VMs. It's also used to distribute hardware resources -- CPU, memory, networking, storage -- to VMs.

Key features of Citrix XenServer aim to ease virtualization infrastructure management. VM templates are a significant aspect of this. For example, you can create VM templates from snapshots. Another key feature of this platform is XenMotion, which allows you to live migrate VMs between hosts. With the Enterprise version of XenServer 7.1, you can also live patch hosts with no downtime. Centralized support for Open vSwitch is another important feature of the XenServer platform.

The XenServer management console, which is a Windows-based client, sets this platform apart from other Xen stacks because it gives you the ability to manage VMs running on multiple hosts in a resource pool. Using this enterprise-level management console, you can install, provision and run VMs on a XenServer host. Admins can also configure a remote storage repository for VMs.

Critics of XenServer argue that its ecosystem isn't as established as those of its major competitors, but the fact that there is an open source version of this hypervisor makes it a good choice for smaller IT shops. Another drawback of Citrix XenServer is that it lacks built-in automation capabilities. In order to automate tasks, you have to use PowerShell scripts. In addition, compared to its competitors, XenServer doesn't have the same level of integration or available certifications.

Citrix XenServer backup and recovery is made easier with VM templates and snapshots. Also, both the Standard and Enterprise versions include HA, which enables localized VM recovery in the event of a failure, and Site Recovery Manager, which allows admins to recover VMs from a major hardware failure in the CPU, networking and/or storage layers[4].

3.3 PROMETHEUS AND GRAFANA DESIGN

As time series database, prometheus is often used in the world of monitoring system in combination with grafana, which is at the very basic a visualization tool for time series data. I will show you, how to get them up and running with the easiest way.

Grafana

Grafana documentation is awesome, please check out there for several installation possibilities, I will use the installation with Docker for simplicity. Because I want to change some of the default configs for later use, I will mount the configuration file with the Docker run.

Prometheus

As well for prometheus, I will take the simple path and install it with Docker, again we are using volumes in order to keep persistence and to be able to change the configuration without diving in the container.

But before we can run the Docker container, we have to prepare the following configs:

Grafana configuration

Create a file called grafana.ini and copy & past official default configuration in there. It is good to run the config file as volume because of future changes, that you might want to do.

Prometheus configuration

In order to tell Prometheus, where to take the metrics, we need to write a config file as well.

- create a file called prometheus.yml

Paste this very basic code example in it, note that at this time you should have an application running, that exposes metrics to prometheus. In my example on localhost using port 8080, check out here the code. You can take localhost:9090, which exposes metrics about the prometheus application itself[10][11].

Linux Containers take a completely different approach than system virtualization technologies such as VMWare, KVM and Xen, which started by booting separate virtual systems on emulated hardware and then attempted to lower their overhead via Para virtualization and related mechanisms. LXC started out with an efficient mechanism and added isolation, resulting in a system virtualization mechanism as scalable and portable as chroot, capable of simultaneously supporting thousands of emulated systems on a single server while also providing lightweight virtualization options to routers and smart phones. LXC, it later morphed into its own container runtime environment. At a high level, Docker is a Linux utility that can efficiently create, ship, and run containers.

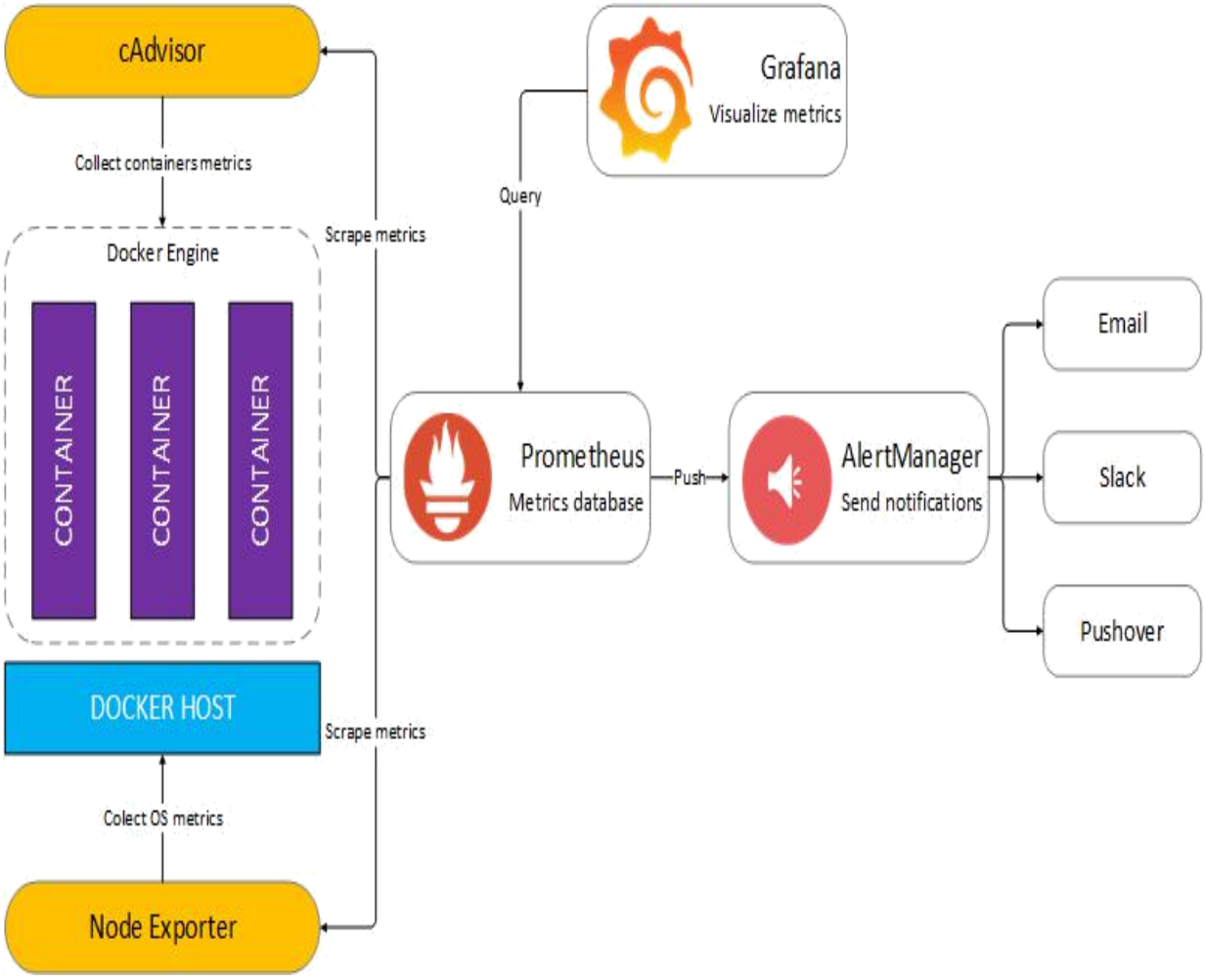

Figure 3.1 Setup monitoring for docker and container

Chapter 4
Project Implementation

4.1 VMware installation

Before downloading and installing VMware Workstation:

- Ensure that your physical machine meets the system requirements for VMware Workstation. For more information, see System Requirements:
- Ensure that you are using a supported guest operating system. For more information, see the VMware Compatibility Guide.

To download VMware Workstation:

1. Navigate to the VMware Workstation Download Center.
2. Based on your requirements, click **Go to Downloads** for VMware Workstation for Windows or VMware Workstation for Linux.
3. Click **Download Now**.
4. If prompted, log in to your My VMware profile. If you do not have a profile, create one. For more information, see How to create a My VMware profile (2007005).
5. Ensure that your profile is complete and enter all mandatory fields. For more information, see How to update your My VMware profile (2086266).
6. Review the End User License Agreement and click **Yes**.
7. Click **Download Now**.

- **Installing VMware Workstation.**

To install VMware Workstation on a Windows host:

1. Log in to the Windows host system as the Administrator user or as a user who is a member of the local Administrators group.
2. Open the folder where the VMware Workstation installer was downloaded. The default location is the **Downloads** folder for the user account on the Windows host.

Note: The installer file name is similar to VMware-workstation-full-*xxxx-xxxx*.exe, where xxxx-xxxx is the version and build numbers.

3. Right-click the installer and click **Run as Administrator**.
4. Select a setup option:
   - **Typical**: Installs typical Workstation features. If the Integrated Virtual Debugger for Visual Studio or Eclipse is present on the host system, the associated Workstation plug-ins are installed.
   - **Custom**: Lets you select which Workstation features to install and specify where to install them. Select this option if you need to change the shared virtual machines

4.2 Docker Installation

- INSTALL DOCKER CE

1. Install the *latest version* of Docker CE, or go to the next step to install a specific version:
2. ***$ sudo yum install docker-ce***

   If prompted to accept the GPG key, verify that the fingerprint matches060A 61C5 1B55 8A7F 742B 77AA C52F EB6B 621E 9F35, and if so, accept it.
   Got multiple Docker repositories?
   If you have multiple Docker repositories enabled, installing or updating without specifying a version in the yum install or yum update command always installs the highest possible version, which may not be appropriate for your stability needs.

   Docker is installed but not started. The Docker group is created, but no users are added to the group.

3. To install a specific version of Docker CE, list the available versions in the repo, then select and install:

a. List and sort the versions available in your repo. This example sorts results by version number, highest to lowest, and is truncated:

$ yum list docker-ce --showduplicates | sort -r

docker-ce.x86_64 18.09.0.ce-1.el7.centos docker-ce-stable

The list returned depends on which repositories are enabled, and is specific to your version of CentOS (indicated by the .el7 suffix in this example).

b. Install a specific version by its fully qualified package name, which is the package name (docker-ce) plus the version string (2nd column) up to the first hyphen, separated by a hyphen (-), for example,docker-ce-18.03.0.ce.

$ sudo yum install docker-ce-<VERSION STRING>

Docker is installed but not started. The docker group is created, but no users are added to the group.

4. Start Docker.
5. ***$ sudo systemctl start docker***
6. Verify that docker is installed correctly by running the hello-world image
7. ***$ sudo docker run hello-world***
   This command downloads a test image and runs it in a container. When the container runs, it prints an informational message and exits.

Docker CE is installed and running. You need to use sudo to run Docker commands. Continue to Linux postinstallto allow non-privileged users to run Docker commands and for other optional configuration steps[14][15].

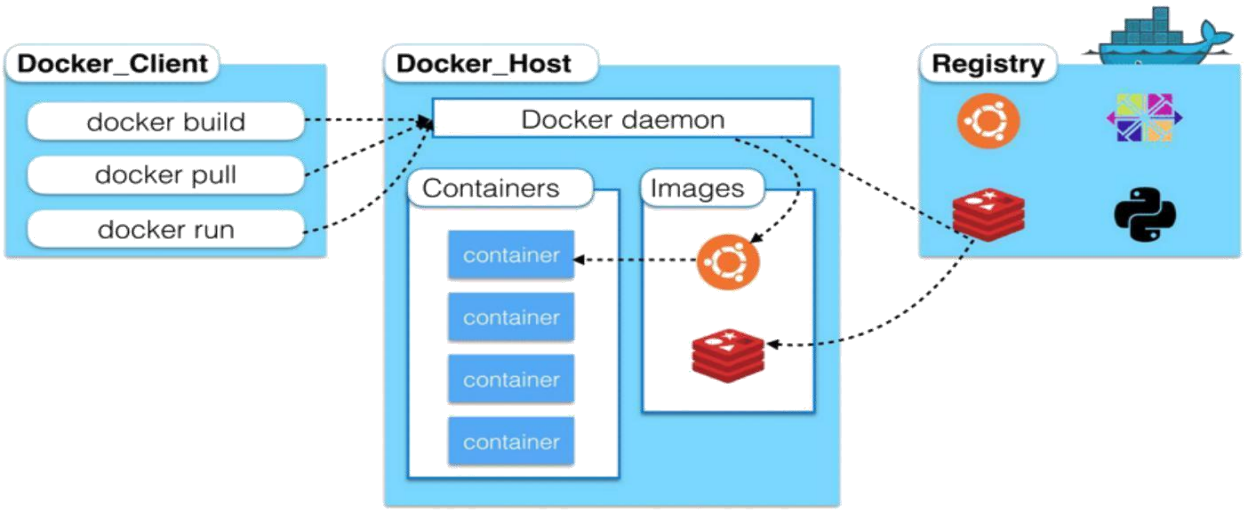

Figure 4.1 Docker And Container architecture

4.3 WORKING OF PROMETHEUS AND GRAFANA

4.3.1 PROMETHEUS CONFIGURATION

Prometheus is a time-series database with a UI and sophisticated querying language (PromQL). Prometheus can scrape metrics, counters, gauges and histograms over HTTP using plaintext or a more efficient protocol.

When the /metrics endpoint is embedded within an existing application it's referred to as instrumentation and when the /metrics endpoint is part of a stand-alone process the project call that an Exporter.

Node exporter

One of the most widely used exporters is the NodeExporter. When NodeExporter is run on a host it will provide details on I/O, memory, disk and CPU pressure. You can run the NodeExporter as

a Docker container, but it needs so many additional flags that the project recommends you run it directly on a host being monitored.

The built-in exporter

Prometheus provides its own set of metrics - in effect dog-fooding. This is built-in and is usually configured to be scraped (or collected) by default.

Community-supported exporters

An interesting exporter that was put together around Docker's Berlin summit is the Docker Hub and Github exporter by Edward Marshall. These exporters surface metrics from the Docker Hub or Github sites by querying APIs periodically and then relaying the values.

Getting Prometheus with Docker

Prometheus is written in Golang and can be consumed as single statically-compiled binary with no other dependencies. The project also packages up the binary with a sensible configuration in a Docker container[10][11][12].

Run Prometheus in Docker

Let's define a Docker Compose which will let us keep our command-lines simple and repeatable:

You can’t access the endpoint URLs directly if you use Docker for Mac or Docker for Windows.

Learn techniques for monitoring applications and servers with Prometheus

Run Prometheus in Docker

Monitor and instrument a sample Golang application

Understand how to deploy Prometheus with Docker stacks

Figure 4.2 Prometheus UI Interface

4.3.2 GRAFANA CONFIGURATION

To run Grafana open your browser and go to http://localhost:3000/. 3000 is the default http port that Grafana listens to if you haven't configured a different port.

There you will see the login page. Default username is admin and default password is admin. When you log in for the first time you will be asked to change your password. We strongly encourage you to follow Grafana's best practices and change the default administrator password. You can later go to user preferences and change your user name

How to add a data source

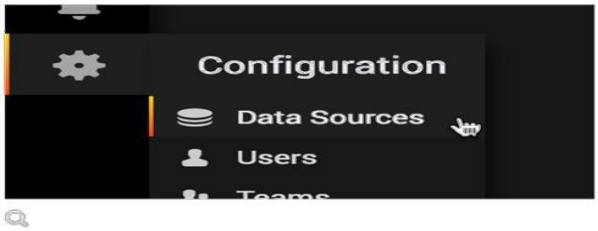

Figure 4.3 Grafana Data source

Before you create your first dashboard you need to add your data source.

First move your cursor to the cog on the side menu which will show you the configuration menu. If the side menu is not visible click the Grafana icon in the upper left corner. The first item on the configuration menu is data sources, click on that and you'll be taken to the data sources page where you can add and edit data sources. You can also simply click the cog.

Click Add data source and you will come to the settings page of your new data source.

First, give the data source a Name and then select which Type of data source you'll want to create, see Supported data sources for more information and how to configure your data source.

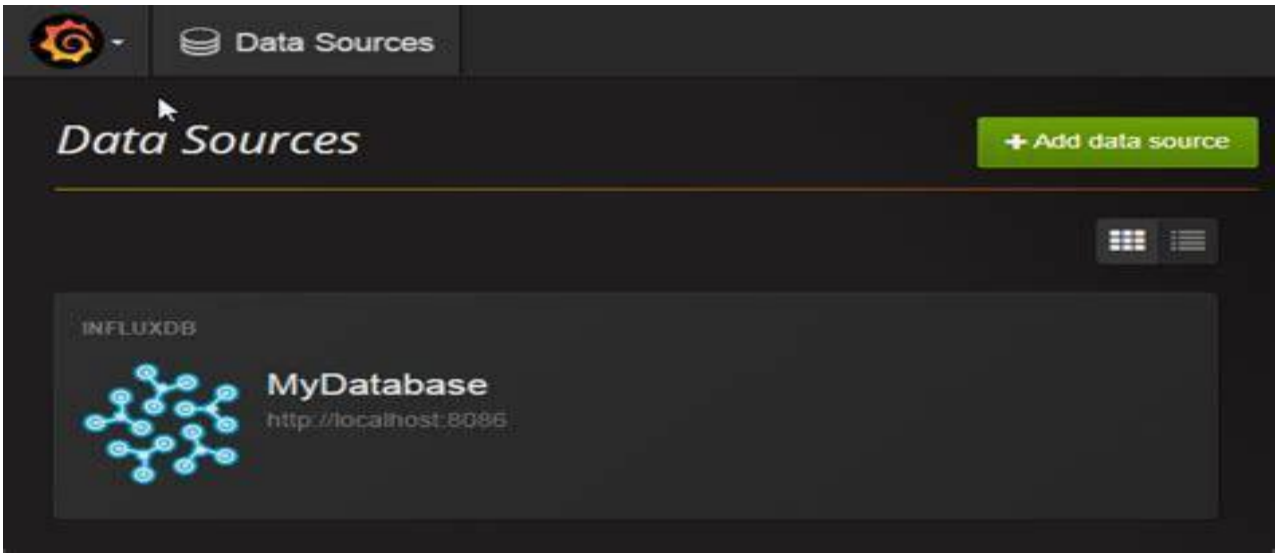

Figure 4.4 Grafana Data source Dashboard

First, give the data source a Name and then select which Type of data source you'll want to create, see Supported data sources for more information and how to configure your data source.

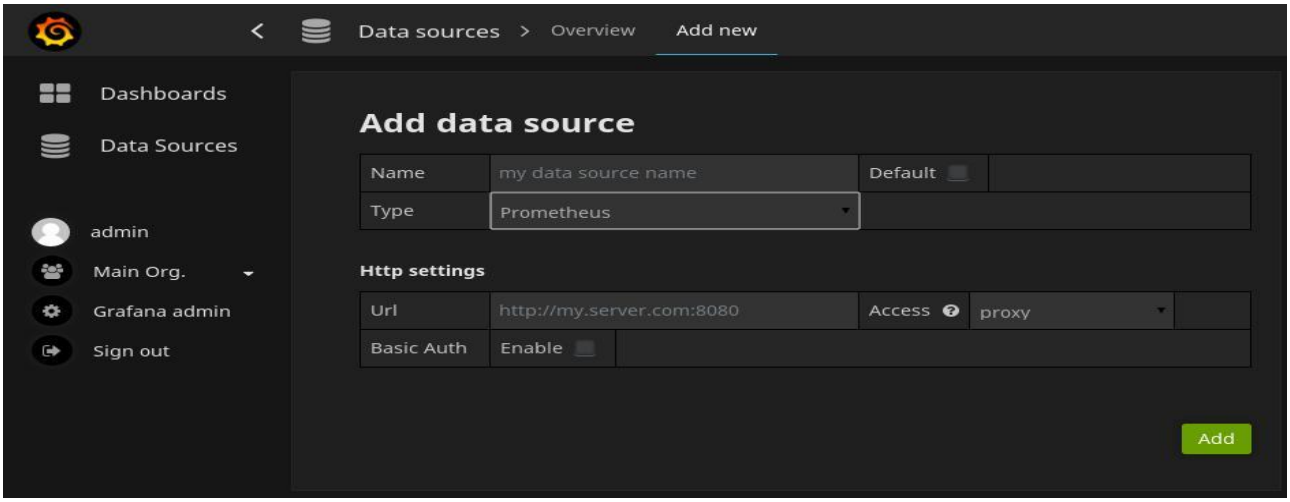

Figure 4.5 Add Grafana Data source

After you have configured your data source you are ready to save and test.

Basic Concepts

Read the Basic Concepts document to get a crash course in key Grafana concepts. Top header

Let's start with creating a new Dashboard. You can find the new Dashboard link on the right side of the Dashboard picker. You now have a blank Dashboard. The image above shows you the top header for a Dashboard.

1. Side menu bar toggle: This toggles the side menu, allowing you to focus on the data presented in the dashboard. The side menu provides access to features unrelated to a Dashboard such as Users, Organizations, and Data Sources.

2. Dashboard dropdown: This dropdown shows you which Dashboard you are currently viewing, and allows you to easily switch to a new Dashboard. From here you can also create a new Dashboard or folder, Import existing Dashboards, and manage Dashboard playlists.

3. Add Panel: Adds a new panel to the current Dashboard

4. Star Dashboard: Star (or unstart) the current Dashboard. Starred Dashboards will show up on your own Home Dashboard by default, and are a convenient way to mark Dashboards that you're interested in.

5. Share Dashboard: Share the current dashboard by creating a link or create a static Snapshot of it. Make sure the Dashboard is saved before sharing.

6. Save dashboard: The current Dashboard will be saved with the current Dashboard name.
7. Settings: Manage Dashboard settings and features such as Templating and Annotations.

Dashboards, Panels, the building blocks of Grafana…

Dashboards are at the core of what Grafana is all about. Dashboards are composed of individual Panels arranged on a grid. Grafana ships with a variety of Panels. Grafana makes it easy to construct the right queries, and customize the display properties so that you can create the perfect Dashboard for your need. Each Panel can interact with data from any configured Grafana Data Source (currently Graphite, Prometheus, Elastic search, Influx DB, OpenTSDB, MySQL, PostgreSQL, Microsoft SQL Server and AWS Cloud watch). The Basic Concepts guide explores these key ideas in detail[10][11][12].

Figure 4.6 Grafana Dashboard.

1. Zoom out time range
2. Time picker dropdown. Here you can access relative time range options, auto refresh options and set custom absolute time ranges.
3. Manual refresh button. Will cause all panels to refresh (fetch new data).
4. Dashboard panel. You edit panels by clicking the panel title.
5. Graph legend. You can change series colors, y-axis and series visibility directly from the legend
6. Get alert notifications instantly when your metrics stop reporting or show unexpected behavior – get notifications via Pager Duty, VictorOps, HipChat, Slack, Email, Web hooks and more. A full alert history allows you to track incidents.

Adding & Editing Graphs and Panels

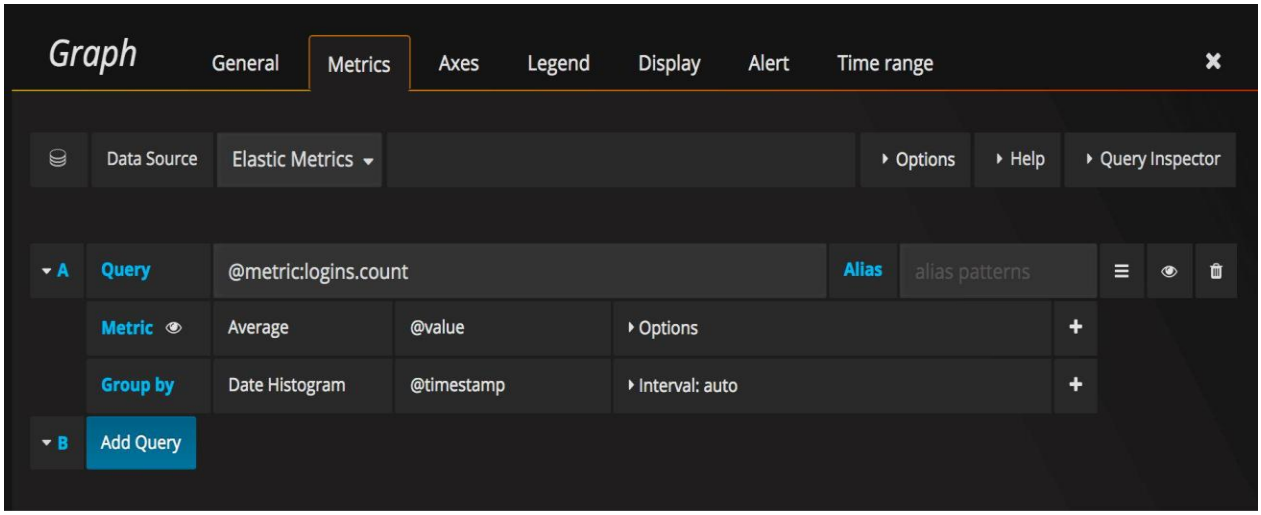

Figure 4.7 Grafana Graph

1. You add panels by clicking the Add panel icon on the top menu.
2. To edit the graph you click on the graph title to open the panel menu, then Edit.
3. This should take you to the Metrics tab. In this tab you should see the editor for your default data source.
4. When you click the Metrics tab, you are presented with a Query Editor that is specific to the Panel Data Source. Use the Query Editor to build your queries and Grafana will visualize them in real time.
5. Grafana is a slick, feature-rich graph and dashboard editor with an intuitive UI. View your metrics on beautiful, interactive dashboards in real-time. We process billions of data points every day and organize your metrics with advanced data views and filtering.

Drag-and-Drop panels

You can Drag-and-Drop Panels by simply clicking and holding the Panel title, and drag it to its new location. You can also easily resize panels by clicking the (-) and (+) icons.

Figure 4.8 Grafana Drag-and-Drop panels

Tips and shortcuts

- Click the graph title and in the dropdown menu quickly change span or duplicate the panel.
- Click the Save icon in the menu to save the dashboard with a new name
- Click the Save icon in the menu and then advanced to export the dashboard to json file, or set it as your default dashboard.
- Click the colored icon in the legend to select series color
- Click series name in the legend to hide series
- Ctrl/Shift/Meta + Click legend name to hide other series

Grafana loves the keyboard

- Ctrl+S Saves the current dashboard

- Ctrl+F Opens the dashboard finder / search
- Ctrl+H Hides all controls (good for tv displays)

Hit Escape to exit graph when in full screen or edit mode

4.4 Flow of Project

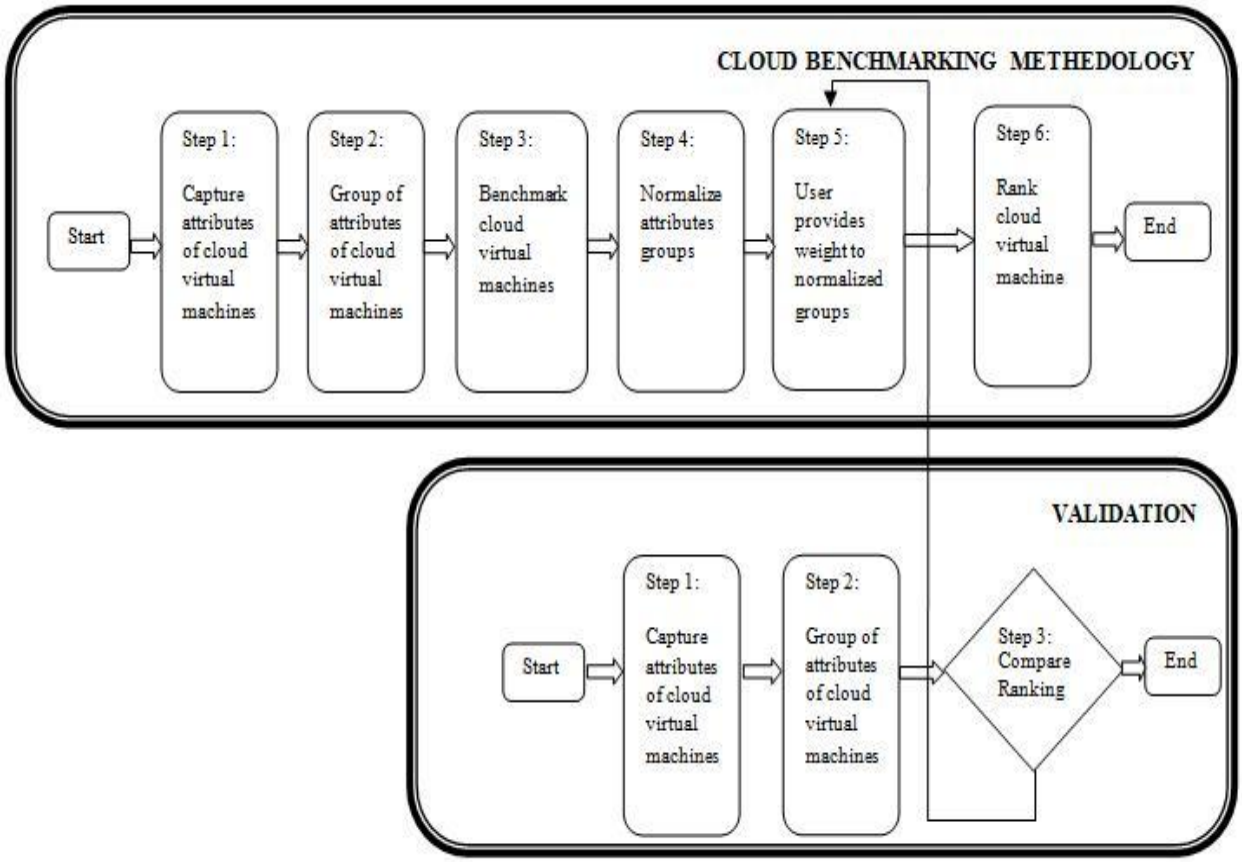

Figure 4.9 Flow of Project

CHAPTER 5
TESTING

Our application collect, analyze and keep all kinds of log-files. The main goal of the environment is to perform primary load testing.

So, the situation is as follows:

- Our service is written in Go and it also has client-server architecture.
- Service can write data in multiple storages so that multiple worker instances can read your data in parallel. This is very important for building test environment.
- Developers need a possibility to troubleshoot test environment fast and safely.
- We have to test network component interaction in distributed environment on several network nodes. For that we need to analyze.
- Traffic flow between clients and servers.
- We need to control resources consumption and make sure daemon remains stable under high load.
- And, of course, we want to see all possible metrics in real time and based on testing results.

As a result, we decided to build Docker test environment and its related technologies. This allows us to fulfill our requests and to effectively use hardware resources without necessity to buy separate server for each separate component. In this case hardware resources can be: a separate server, a set of servers or even a developer's laptop.

Test Environment Architecture

For a start let's consider main architecture's components:

- Arbitrary quantity of server instances of our application.
- Arbitrary quantity of agents.

- Separate environments with data storages such as: Elastic Search, MySQL or PostgreSQL.
- Load generator (we have implemented simple stress-generator, but it is possible to use any other, for example, Yandex.Tank or Apache Benchmark).

Test environment should be easy to scale and support.

We have built distributed network environment with the help of Docker containers, which isolate internal and external services, and docker-machine, that allows establishing isolated test environment.

As a result test environment architecture looks like:

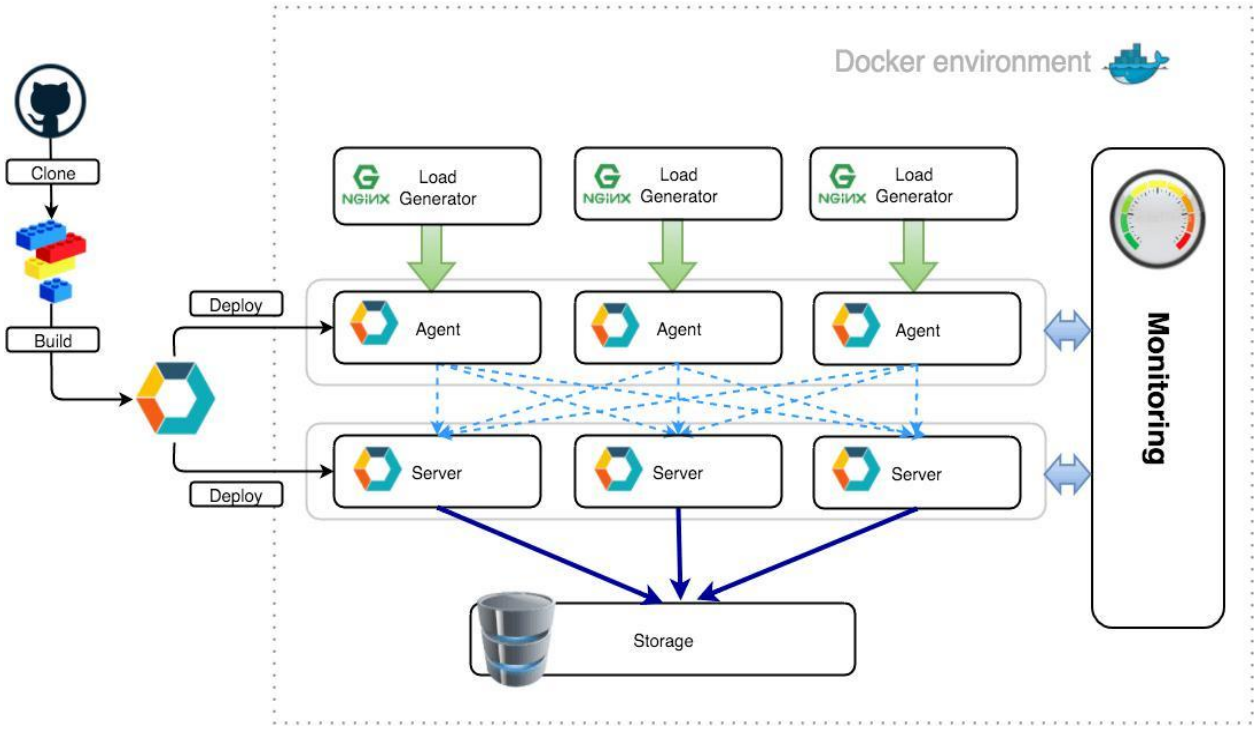

Figure 5.1 Testing Flow

For environment visualization we use Weave Scope, because it's a very convenient and clearly arranged service for Docker containers monitoring

REFERENCE

[1]An Updated Performance Comparison of Virtual Machines and Linux Containers Wes Felter, Alexandre Ferreira, Ram Rajamony, Juan Rubio IBM Research, Austin, TX

[2]Virtualization and Containerization of Application Infrastructure: A Comparison Mathijs Jeroen Scheepers University of Twente

[3] Virtualization Overview. http://www.vmware.com/pdf/virtualization.pdf.

[4] Xen Project Software Overview. http://wiki.xen.org/wiki/Xen Overview.

[5]SoftLayer introduces bare metal cloud. http://www.softlayer.com/press/ release/96/softlayer-introduces-bare-metal-cloud, Oct 2009.

[6]PXZ—parallel LZMA compressor using liblzma. https://jnovy. fedorapeople.org/pxz/, 2012.

[7] Advanced multi layered unification filesystem. http://aufs.sourceforge. net, 2014.

[8] Jens Axboe. Flexible IO Tester. http://git.kernel.dk/?p=fio.git;a=summary.

[9]Lxc, linux conainers. URL http://linuxcontainers.org/. Accessed:2018-10-30.

[10] Prometheus. URL https://prometheus.io/docs/introduction/overview/.Accessed:2018-10-9

[11]Grafana. URL https://grafana.com . Accessed:2018-10-9

[12]Sysbench benchmark. https://launchpad.net/sysbench.

[13]The “nuttcp” Network Performance Measurement Tool. http://www. nuttcp.net/.

[14]Michael H• uttermann. DevOps for Developers, volume 1.Springer, 2012.

[15]Docker. URL https://www.docker.io/. Accessed: 2018-10-30.