

Naveed Anwar

Architecting Scalable Web Application with Scalable Cloud Platform

Helsinki Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

28 February 2018

PREFACE

Allah Almighty, the lord of majesty and honor; first of all, I would like to thank you for giving me the courage and strength to undertake this Master of Engineering program and accomplish this thesis project. I am thankful to my mother Mrs. Naziran Begum, for her definite support and motivation that I can complete this work in parallel to full-time employment. Her best wishes, prayers, sincere advise and time management guidelines were really helpful throughout my academic and professional career. It was a challenging and time-consuming task for me requiring considerable time and attention. In such challenging situations, I thought about my father, Malik Mohammad Anwar (late) to get inspiration from his personality that “everything is possible with your full involvement and dedication”. I would like to thank him for such captivated and self-motivational theories and practices.

My special gratitude goes to my beloved wife Mrs. Shabana Naveed for her encouragement to complete this work, providing the peaceful environment for studies and understanding the importance of my work. Without her moral support, I couldn't have completed this work as well as several other projects related to my professional career. I would like to thanks to my two wonderful princesses; Hamnah Naveed, Marifah M. Naveed, and my little prince Mohammad Zakaria for their comprehensive understanding and not demanding too much time during my studies. My brothers and sisters (NTSISUS) living far away and praying for my success, I would like to acknowledge their encouragement and motivation as well.

My sincere thanks also go to my thesis supervisor and examiner Ville Jääskeläinen for his precious guidance from the selection of study options to accomplish this thesis work. His prompt and valuable insights have always been very helpful throughout this research work.

Last but not least, I would like to thanks to my senior colleague and department manager, Christer Magnusson for his moral support during my studies. He always appreciates my educational activities and believes that academic updates introduce new ideas and solution that makes difference in the services we offer.

Stockholm, 28 February 2018
Naveed Anwar

Author(s) Title	Naveed Anwar Architecting Scalable Web Application with Scalable Cloud Platform
Number of Pages Date	92 pages + 2 appendices 28 February 2018
Degree	Master of Engineering
Degree Programme	Information Technology
Specialisation option	Networking and Services
Instructor(s)	Ville Jääskeläinen, Head of Master's program in IT
<p>World Wide Web (WWW) has achieved a significant role in information sharing, communication and service delivery. Online identity and existence have become an essential part of the success of the enterprises and web-based applications are growing constantly, causing rapid growth in the web traffic. This rapid growth is dynamic in nature and often unpredictable in terms of resource demands, seamless response time and service delivery. The classic approach of traditional infrastructure provisioning involves service interruption and capital expenditure to purchase new servers or upgrade the existing system infrastructure. This is why solution architects are focusing on the innovation of new systems to ensure operational continuity of unpredictable system by means of scalable system design and considering “cloud services” for building the scalable web applications.</p> <p>Flexibility, availability, and scalability are some of the attributes of cloud platform that introduces concepts, best practices, tools and techniques to design, implement and operate a cloud platform to scale web applications dynamically. Servers can be provisioned automatically when required (increase in the resource demand) and destroyed when there is no need (decline in the resource demand) with highly virtualized and scalable cloud platform.</p> <p>Among several cloud service providers, Amazon is one of the famous cloud service providers that offer a wide range of computing services in a virtualized environment. This study concerns the design and implementation of a scalable web application using Amazon Web Services (AWS) cloud platform. This study provides a granular understanding of how servers in the cloud are scaled when stressed out using benchmarking tools and the whole process remain transparent to the application consumer. The performed experiments, results, and analysis presented in this report shows that proposed (scalable) cloud architecture is proficient to manage application demand and saving overall infrastructure investment.</p>	
Keywords	Cloud Computing, Scalable Cloud Platform, Web Application Scalability, Cloud Load Balancer, Virtualization, JMeter

Table of Contents

Preface

Abstract

List of Figures

List of Tables

List of Abbreviations

1	Introduction	1
1.1	Overview	1
1.2	Problem Statement	2
1.3	Methodology	3
1.4	Project Scope	4
1.5	Structure of this Thesis	4
2	The Legacy System	6
2.1	Overview of the Legacy System	6
2.2	Scalability Analysis with Experimental Workload	9
3	Cloud Computing	11
3.1	Defining Cloud Computing	11
3.2	Essential Characteristics of the Cloud Computing	12
3.3	Cloud Classifications and Service Models	14
3.4	Cloud Service Usage and Deployment Models	19
3.5	Virtualization of Compute Resources	23
3.6	Types of Virtualization	24
3.7	Scalability	26
3.8	Little's Law	27
3.9	Scalability Layers	28
3.10	Scalability Design Process	29
3.11	Scaling Approaches	31
4	Scalable Cloud Architecture for a Web Application	32
4.1	Scalable Web Application Reference Architecture	32
4.2	Load Balancing Tier	33
4.3	Application Tier	35
4.4	Database Tier	36
4.5	DNS Server and User Requests	37

4.6	Monitoring and Alerts	38
4.7	Management Node	40
5	Implementation	42
5.1	Motivation for Using Amazon Web Services	42
5.2	Create an Amazon Account	43
5.3	Create Virtual Server using Amazon Elastic Cloud Compute (EC2)	45
5.4	Install and Configure Apache Web Server (LAMP Stack)	52
5.5	Install and Configure WordPress Application with Amazon Linux	54
5.6	Create an Image from Linux EC2 Instance	56
5.7	Create Auto Scaling Group and Launch Configuration	58
5.8	Create Elastic Load Balancer	63
5.9	Performance Measurement Tool (JMeter)	67
5.10	Response Time Assertion	69
5.11	Creating Performance Test Plan in JMeter	70
5.12	JMeter Concurrency Level and Best Practices	74
6	Results and Analysis	76
6.1	Experiment Environment	76
6.2	Experiment Workload and Data Collection	77
6.3	Results	82
6.4	Scalability Analysis	85
7	Discussions and Conclusions	89
7.1	Conclusion	89
7.2	Future Work	91

References

Appendices

Appendix 1. Preparing Experimental Environment with JMeter

Appendix 2. Troubleshooting DNS Name Change Problem

List of Figures

Figure 1. Summary Report (Legacy System)	10
Figure 2. Cloud Classifications, Everything as a Service [4, Fig. 1.7].	14
Figure 3. Scope and Control of Cloud Service Model [11].	15
Figure 4. IaaS, Scope and Control [13], [14].	16
Figure 5. PaaS, Scope and Control [16].	17
Figure 6. SaaS and FaaS, Scope and Control [20].	18
Figure 7. Public Cloud [22, Fig. 4.17].	19
Figure 8. Private Cloud [22, Fig. 4.19].	20
Figure 9. Community Cloud [22, Fig. 4.18].	21
Figure 10. Hybrid Cloud [22, Fig. 4.20].	22
Figure 11. A Basic Virtual Machine Monitor / Hypervisor [26, Fig 1.1]	23
Figure 12. Hardware Abstraction [26, Fig. 2.6].	24
Figure 13. Scalability Layers [34, Fig 1.1]	28
Figure 14. Scalability Design Process [34, Fig 1.11]	29
Figure 15. Scalable Web Application Reference Architecture	32
Figure 16. Connection Rate Curve of the Load Balancer [43, Fig 22-8].	34
Figure 17. Server Response Time [43, Fig 22-10].	35
Figure 18. CloudWatch Monitoring (CPU Utilization)	39
Figure 19. CloudWatch Basic and Detailed Monitoring	40
Figure 20. Create AWS Account [73]	44
Figure 21. AWS Account Registration Page	44
Figure 22. AWS Services Dashboard (Partial Screenshot)	45
Figure 23. EC2 Dashboard (Partial Screenshot)	46
Figure 24. Configure Instance Details	46
Figure 25. Review Instance Launch	47
Figure 26. Select an Existing Key Pair or Create a New Key Pair	48
Figure 27. Creating a New Key Pair	48
Figure 28. EC2 Instance Launch Status	49
Figure 29. EC2 Dashboard Instance Information	49
Figure 30. SSH Client Selection	50
Figure 31. Connecting to EC2 Instance directly from web browser	51
Figure 32. Linux AMI Login with Ec2-User	52
Figure 33. Creating an Amazon Machine Image	56
Figure 34. Creating an Amazon Machine Image (Properties)	57
Figure 35. Available Amazon Machine Images (AMIs)	58

Figure 36. Amazon EC2 Auto-Scaling Mechanism [64, Fig 11.7]	59
Figure 37. Automatic Recovery of EC2 Instance [64, Fig 11.2]	60
Figure 38. Welcome to Auto Scaling	61
Figure 39. Create Auto Scaling Group and Launch Configuration	61
Figure 40. Configure Auto Scaling Group Details	62
Figure 41. VPC, Region, Availability Zone [64, Fig 11.6]	62
Figure 42. AWS Load Balancing	64
Figure 43. Load Balancer Types	64
Figure 44. Configure Health Check	65
Figure 45. Associate Load Balancer to Auto Scaling Group	67
Figure 46. The Anatomy of a JMeter Test [59].	68
Figure 47. Response Time Assertion	69
Figure 48. High Load Simulation with JMeter	70
Figure 49. Add Thread Group	71
Figure 50. Thread Group Properties	72
Figure 51. Executing JMeter Test	73
Figure 52. Experiment Environment of the Legacy System	76
Figure 53. Experiment Environment of the Scalable Cloud Platform	77
Figure 54. Ramp-Up Period Representation	79
Figure 55. Experiment Results in Tree Format in JMeter	80
Figure 56. Experiment Results in Table Format in JMeter	81
Figure 57. JMeter Summary Report	81
Figure 58. Defining Minimum and Maximum Number of EC2 Instances	86
Figure 59. Auto Scaling Activity History	87
Figure 60. Connection Draining Configuration	87
Figure 61. Download Apache JMeter	Appendix 1
Figure 62. Java SE Development Kit Demos and Samples Downloads	Appendix 1
Figure 63. Advance System Properties (Microsoft Windows)	Appendix 1

List of Tables

Table 1. Hardware and Software Specifications of Current System	6
Table 2. Capacity Analysis of the Current Infrastructure Components.	8
Table 3. Software and Hardware Specifications of the Management Node	40
Table 4. Installing LAMP Stack (Terminal Commands)	53
Table 5. Installing WordPress Application (Terminal Commands)	54
Table 6. Summary of the Experiment Workloads.	78
Table 7. Experiment Results of Legacy System.....	83
Table 8. Experiment Results of the Scalable Cloud Platform	84
Table 9. Scalability Analysis of the Cloud Platform	85
Table 10. Scale Out Time of EC2 Instances	88

List of Abbreviations

AMI	Amazon Machine Image
AWS	Amazon Web Services
CCs	Concurrent Connections
CMS	Content Management System
CSV	Comma-Separated Values
CPU	Central Processing Unit
CSP	Cloud Service Provider
DB	Data Base
DBMS	Database Management System
DNS	Domain Name Server
DoS	Denial of Service
EC2	Elastic Compute Cloud
EIP	Elastic Internet Protocol
ELB	Elastic Load Balancing
ERP	Enterprise Resource Planning
ESX/ESXi	Elastic Sky X / Elastic Sky X Integrated (VMWare Hypervisor)
FaaS	Framework as a Service
FTP	File Transfer Protocol
GB	Giga Bit
GUI	Graphical User Interface
HaaS	Hardware as a Service
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IaaS	Infrastructure as a Service
IDE	Integrated Development Environment
I/O	Input / Output
IP	Internet Protocol
IP v4	Internet Protocol Version 4
IP v6	Internet Protocol Version 6
IT	Information Technology
KPI	Key Performance Indicator
KVM	Kernel-based Virtual Machine
JMS	Java Message Service
JDBC	Java Database Connectivity

LAMP	Linux, Apache, MySQL, PHP
LAN	Local Area Network
LB	Load Balancer
MB	Megabit
NIST	National Institute of Standards and Technology
OS	Operating System
PaaS	Platform as a Service
PC	Personal Computer
RAM	Read Only Memory
RPC	Remote Procedure Call
S3	Simple Storage Service (Amazon Cloud Storage)
SaaS	Software as a Service
SAN	Storage Area Network
SLA	Service Level Agreement
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSD	Solid State Drive
SSH	Secure Shell
TCP	Transmission Control Protocol
TPS	Transactions Per Second
UTC	Coordinated Universal Time (UTC)
URL	Uniform Resource Locator
UX	User Experience
vCore	Virtual Core
VLAN	Virtual Local Area Network
VM	Virtual Machine
VMM	Virtual Machine Monitor
VPN	Virtual Private Network
VPS	Virtual Private Server
WWW	World Wide Web
XML	Extensible Markup Language

1 Introduction

This chapter provides the study background, aims, and objectives of this project goal, a description of the problem statement and the research question.

1.1 Overview

Information Technology (IT) has reinforced the concept of information processing more efficiently and effectively over the Internet with the evolution of cloud computing. Cloud computing uses Internet technologies to deliver a wide range of computing services offered by several Cloud Service Providers (CSPs). Engineering and scientific applications, big data analysis, data mining, gaming, finance, social media and many other computing activities that require scalable infrastructure can benefit from cloud computing.

Cloud computing provides a platform to deploy scalable applications, that can be provisioned with the increase in the demand or with intensive resource utilization. Business needs are changing rapidly and often are dynamic in nature. That's why high availability and responsiveness of web applications are some of the core aspects of designing modern web applications. Cloud computing services are not limited to the web-based applications but cover the full range of computing activities, e.g. data and storage solutions, the dedicated virtual private server (VPS) are few to mention. The scalable cloud platform is capable to allocate resources in a timely manner at the time of high demand (scale in) and terminating the allocated resources when there is deterioration in the demand (scale out). All the technical details remain transparent from the end user. The users of cloud service are mainly concerned if the cloud services meet their needs and want to maintain budget by paying for the consumed resources. Scalable cloud platform helps customers and service owners to reduce the cost of computing, and at the same time provides an immense capacity of computing resources when required.

The classic approach was to launch a standard web application according to existing business needs and then follow the application maintenance and testing life cycle. These phases require modifications, emission of errors and even sometimes re-design everything to meet the new challenges e.g. user demands, high workloads, and responsiveness. Now business entails its online platform to be scalable in order to sustain the unpredictable growth in terms of resource utilization and the number of concurrent requests

to a particular web application. With an optimized scalable cloud architecture, computing resources and cloud infrastructure can accommodate all of the application's lifecycle phases. This approach provides a consistent context to shape an application from its concept into development, production to maintenance and gradually to the end of life. As a result, scalability in modern day web applications is more relevant now than ever and has achieved attention by solution architects, IT professionals and researchers. Techniques like fault tolerance, cloud computing, distributed computing, load balancing, and virtualization help not only in scalability; they are also very effective in achieving high availability.

1.2 Problem Statement

The legacy approach to cope with the unpredictability is to over-provision the resources to manage the web traffic load. With this approach, the web application under consideration managed to sustain availability in case of a heavy load of web traffic. This approach does not effectively utilize the available resources with the decline in demand, and unused resources remain in the idle state. Due to the presence of unutilized resources, the overall solution was not a cost efficient and not a recommended approach for infrastructure provisioning. Also, a system downtime was involved when a particular hardware component required an upgrade or a malfunctioning component needed a replacement. During system downtime, the target web application remains unavailable causing loss of revenue because no user requests were facilitated during the maintenance process.

This legacy approach was not a desirable approach to manage the system load under high resource utilization. In contrast, a scalable cloud architecture provides a perfect platform to deploy scalable applications, that can be provisioned with an increase in the demand and decommission them when consumed resources are no longer required. With dynamic provisioning, customers pay only for the consumed resources for the period of time the specified resources were in use. Due to the dynamic resource provisioning, no computing resource remains in an idle state when there is a decline in demand (by decommissioning the allocated resources).

The ability of a system to handle higher (often unpredictable) workload without compromising its specified performance is referred to as scalability. The main objective of this thesis is to study how to design a scalable cloud platform to implement and test a scalable web application. The goal is also to examine how to scale up the resources since for a cloud-based application accessed by an unpredictable number of users may result in

a very high resource demand. Similarly, how to revoke the allocated resources which are no longer required is an essential part of the solution. This helps to reduce the overall cost by removing the additional resources and paying only for the consumed resources.

Research Problem

This research identifies the scalability issues in the legacy infrastructure model and evaluates the proposed scalable cloud-based architecture using Amazon Web Services (AWS). This platform was designed to meet the performance of a modern rapidly evolving web-based application. Especially it tries to answer the following research question:

“Is it possible to architect a scalable web application using a cloud platform to dynamically manage the increased workload by provisioning the required resources and terminate the assigned resources when there is a decline in the resource demand?”

This study demonstrates on how to both scale in and scale out for a cloud-based application used by a random number of users. Infrastructure provisioning problems in non-scalable systems i.e. over-provisioning the compute resources, service interruption and over budgeting can be overcome with dynamic scaling and a cloud load balancer.

1.3 Methodology

Different research strategies and methods have been developed to support the researchers in creating and presenting their findings in a well-structured manner. These research strategies and methods are useful for empirical study as well as in design science while exploring problems of a practical nature and defining requirements and investigation artifacts. This study uses multiple methods (also known as pragmatic studies) to solve the research problem. The pragmatic approach provides opportunity to combine and mix different data collection methods to analyse the data, and multiple perspectives to interpret the results. Following steps were involved to solve the design problem:

- Define the problem.
- Gather required information.

- Generate multiple potential solutions.
- Selection of the solution.
- Implement the solution.
- Analyze the solution.

At the initial stage of this project, typical quantitative research methods were investigated but rejected later on because quantitative methods focus on very specific procedures and are not suitable for the purpose of this project goal. For instance, quantitative research methods are mainly used to employ mathematical and statistical models, theories, and hypothesis. That's why a pragmatic approach was followed because of the opportunity to apply and combine any of the available approaches, methods, techniques, and procedures.

1.4 Project Scope

The proposed (scalable) solution in this report was implemented and evaluated using cloud platform; Amazon Web Services (AWS). The security of the cloud infrastructure components such as virtual servers, networking components, and multi-tenancy issues is not included in this report. Similarly, backup and disaster recovery mechanism adopted by the AWS is not included in this report.

The major focus of this study was to design and implement a scalable web application using cloud platform to dynamically scale in with an increase in the resource demand and scale out when there is a decline in the resource demand.

1.5 Structure of this Thesis

The thesis has been divided into 7 sections. Chapter 1 introduces the problem, objective and scope of this project. Chapter 2 provides information about the existing system (legacy) and analyses of the current system to identify the scalability problems in the existing system. Chapter 3 provides a theoretical background of the topic i.e. general information to understand the cloud computing, the role of the virtualization in the cloud computing and scalability patterns. Chapter 4 describes the proposed architecture for the scalable application and details about the design and participating components. Based on the presented architecture, Chapter 5 defines the implementation of the proposed solution

with the cloud platform. This chapter also provides the required steps that were performed in order to deploy the scalable web application using the cloud platform. Chapter 6 provides the information about the experiment workloads, results of the experiments and scalability analysis. Chapter 7 concludes this thesis and presents the conclusion of this thesis based on the experimental findings, and highlights the areas where further research may be conducted.

Additional information regarding how to download and setup the JMeter (performance measurement tool) is available in Appendix 1. Commands used to repair the broken installation of WordPress application are listed in Appendix 2.

|

2 The Legacy System

This chapter provides the introduction to the legacy (non-scalable) system, its software and hardware specifications and an experiment environment which was used to find the key weaknesses and scalability related issues.

2.1 Overview of the Legacy System

The legacy system described in this chapter is based on a laboratory setup. This environment was comprised on one dedicated machine that was used to present the existing legacy model of infrastructure provisioning. Apache web server was configured to host WordPress web application running on LAMP stack (Linux, Apache, MySQL, PHP). The same machine was hosting MySQL database and performing functions of a web and database server. The software and hardware specifications of this environment are listed in Table 1.

Table 1. Hardware and Software Specifications of Current System

Hardware Specifications		Software Specifications	
Processor (CPU)	1 vCPU	Operating System	Linux Server
Memory (RAM)	1 GB	System Type	X64 (64 bit)
Hard Disk	8 GB (SSD)	Web Application	WordPress

Table 1 summarizes the hardware capacity and software configuration of the legacy system. This machine was configured with a legacy infrastructure model and the system is not aware how to handle the heavy load and an unpredictable number of user requests. Also, with the growth in the business, there is no pre-defined mechanism that can be used to scale the existing system to meet the business requirements. One of the business requirement is to facilitate a user request within 3 seconds regardless of the load of the system.

Service and maintenance related tasks require a system termination or power down in most cases which results in service interruption because the server is offline during a maintenance. Examples of such tasks include; replacing a malfunctioning component, upgrading system memory (RAM), installation of hard disks with increased capacity or

replacing currently installed hard (sequential) drives with fast Solid State Drive (SSD). The current system was also subject to experience downtime as a reboot often required when there was any major software update applied; either an application software or an operating system upgrade.

Demand Analysis of the Laboratory Setup

The purpose of this analysis was to understand how the load on the web application affects to system usage, average peak time of application utilization, CPU, network load, RAM consumption and other infrastructure components. The number of estimated users accessing the application under analysis was not aware in advance. The increase in demand takes place infrequently, such as with new product launch, promotions or annual sales. However, the current demand can lead to a high sudden load, and this analysis helped to understand the application behaviour if it can cope with this sudden load.

The system under test tends to decrease in performance with the fluctuation in the high resource utilization. This system uses the legacy approach of infrastructure provisioning and wasn't capable to automatically scale the resources with increase in the resource demand.

Capacity Analysis of the Laboratory Setup

Capacity analysis of the laboratory setup was conducted in order to estimate the capacity of the laboratory setup. Since the expected demand is dynamic in nature, it was important to estimate the current capacity for the X number of users at a given time while maintaining the business requirement of 2 seconds as service level agreement (SLA). This means that user requests should be facilitated in 2 seconds, failure doing this results breach in business SLA. Table 2 given below provides threshold values for the laboratory setup that can help to understand if the provisioned resources are over or underutilized.

Table 2. Capacity Analysis of the Current Infrastructure Components.

Resource	Performance parameter	Evaluation Criteria
CPU	High utilization	Determine if the CPU utilization is 75 % or above for a certain time frame alongside a number of requests to be processed. The value of CPU cores needs to be adjusted to cope with high CPU load.
Memory	High page rate	In Linux systems, the physical memory is divided into pages and these pages are allocated to different processes. State of these pages determines the high page rate i.e.; free (unused) or busy (allocated to process).
	Swap space	Even a system is equipped with an adequate amount of physical memory, Linux kernel uses the swap space to move the memory pages which are not used frequently. The system will be slow if not enough RAM is available and the kernel is forced to continuously shuffle memory pages to swap and back to RAM. Consider to increase in the memory capacity if swap is used 70% or above.
Storage	Low space	A system is considered to have low disk space if the operating system files span over 70 % of the entire hard disk.

Analysis of the system capacity was conducted to find the potential bottlenecks in the existing system as well as to provide an estimation of the expected load the current system can handle while maintaining its performance.

2.2 Scalability Analysis with Experimental Workload

To understand the scalability related issues, the legacy web application architecture was analysed by conducting several performance evaluation experiments using benchmarking tool. These experiments were effective in the collection of the infrastructure usage statistics. Following sections provides a description of these experiments.

Defining Experiment Metrics

Among several performances related parameters, this study was focused on the system throughput, response time and resource utilization (CPU, RAM, Network). Throughput represents the number of successful requests that a web server was capable to manage. Response time, also known as execution time or assertion time; represent the time spend to facilitate a web request. All these experiments were conducted with 2 seconds as criteria of success for the response time. If the request was fulfilled within 2 seconds, it was considered as a successful request, otherwise marked as false.

Workload Generation

Performance of the web server was examined by generating heavy load with performance measurement tool called JMeter (described in chapter 5). Existing legacy system was evaluated by simulating concurrent user requests to the web server. Multiple threads represent concurrent connections where each thread executes test plan independently of the other threads in a test plan. Chapter 5, section 5.11 “Creating Performance Test Plan in JMeter” provides details about the elements of the JMeter test plan.

Following workloads were used in these experiments:

- 1 – 25
- 1 – 50
- 1 – 100
- 1 – 1,000
- 1 – 2,000

Above mentioned workloads served as an input to measure the system throughput and response time of the legacy system. The legacy system in the test environment continue to decline in the performance and resulted in breach of the 3 seconds SLA. Following figure shows the Summary Report for the legacy system for the 2,000 users that was generated with JMeter.

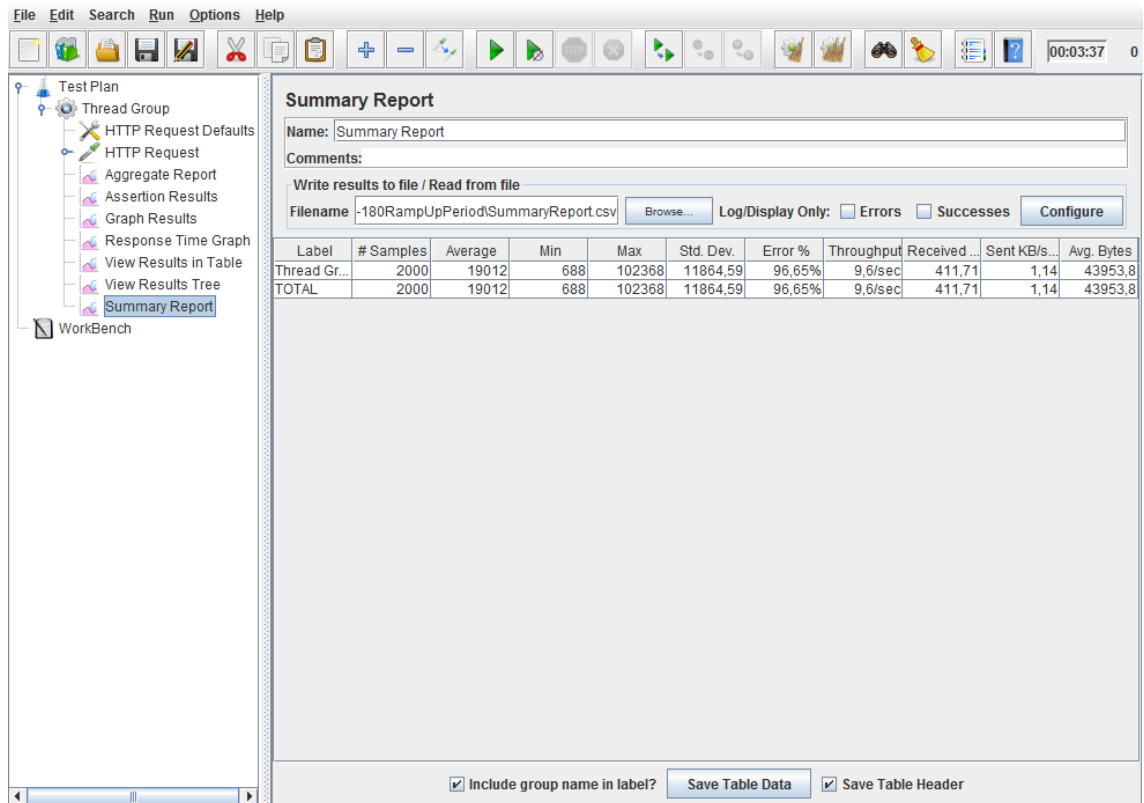


Figure 1. Summary Report (Legacy System)

As presented in the Figure 1, the legacy system produced 96.65 % of error when processing the 2,000 users. This caused the service interruption (web applications was unavailable) because the legacy system wasn't capable to scale the resources to manage the resource demand.

Chapter 6 provides the detailed description of the tests, number of simulated users, and analysis of the results after applying the same workload on the web application hosted in the scalable cloud platform.

3 Cloud Computing

This chapter provides the theoretical background of the cloud computing including, definition, characteristics, different computing and deployment models and virtualization of computing resources. Description of scalability and its attributes are also presented in this chapter to get basic insights on the scalability.

3.1 Defining Cloud Computing

Cloud Computing has emerged as a vital service in the computing industry. Modern business needs are changing dynamically and considering cloud computing as a credible fit to fulfill their needs while maintaining the budget. Due to flexibility and a wide range of services offered by cloud computing, many of the existing applications are likely to move to the cloud solutions. With cloud computing, computational resources are not physically present at the consumer location, rather accessed over the Internet from the client computer. Cloud service provider takes responsibility for the uptime, service availability, backup, and disaster recovery procedures, and upgrade and maintenance related tasks. Scalable cloud platform helps customers and service owners to reduce the cost of computing, and at the same time have an immense capacity of computing resources when required. Previously occupied resources can be automatically terminate when there is a decline in the resource demand [1].

According to Amazon; one of the pioneers of the cloud service providers defines the cloud computing in simplest form as an on-demand delivery of compute resources through cloud platform over the internet and paying only for the consumed resources [2]. From the past few years, cloud computing services have been frequently adopted by single users for a private usage, and small and large enterprises for professional usage. This widespread usage of the cloud computing technology has resulted in several definitions depending upon their business needs, but the central idea is the same. Among several definitions, National Institute of Standards and Technology (NIST) defines cloud computing as:

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models [3].

Rosenberg and A. Mateos use the following approach to describe the cloud computing:

Computing services offered by a third party, available for use when needed, that can be scaled dynamically in response to changing needs. Cloud computing represents a departure from the norm of developing, operating and managing IT systems. From the economic perspective, not only does adoption of cloud computing has the potential of providing enormous economic benefit, but it also provides much greater flexibility and agility [4].

The cloud platform uses virtualization technology to make efficient use of the hardware components and is capable to allocate resources in a timely manner at the time of high demand. With this approach, the user of a cloud service pays only what is actually consumed. All the technical details regarding infrastructure provisioning remain transparent to the end user [5]. The users of cloud service are mainly interested in the services and solutions provided by the cloud service providers and are not concerned on how the service is actually maintained. They are also not concerned about the technical details such as number of servers, power supplies, and security of the datacentres. However, users are concerned about the security and availability of the cloud services.

3.2 Essential Characteristics of the Cloud Computing

Following section summaries, the essential characteristics of the cloud computing. These characteristics are the principles of the cloud computing, also known as pillars of cloud computing.

On-Demand Self-Service

With on-demand self-service, a cloud service consumer is capable to customize the computing resources themselves directly from the web browser, usually by interacting with some type of “Admin Console” or “Dashboard”. Users are capable to perform these task without an interaction of the cloud service providers.

Network Access

Computing resources are accessible and available to any subscribing user of the cloud service over the public network; such as the Internet. Users can access the resources

regardless of the client device they are using, for example; workstations, tablets, and mobile phones. The quality of the network connection may limit the usage of the cloud services. High speed Internet connection with low delays are essential for many applications.

Resource Pooling

Cloud computing makes possible to utilize the available resources dynamically that can be allocated to several users and re-assign according to the consumer demand. This model is called multi-tenant model. Multi-tenancy is a result of trying to achieve an economic gain in cloud computing by utilizing virtualization and allowing resource sharing dynamically [6]. These computing resources can include storage, processing, memory, and network bandwidth. Generally, the user of cloud service has no information where and how these resources are maintained by service provider's datacentres. In some cases, users need to know the location of datacentres for legal reasons. With resource pooling, physical resources are shared among cloud users with a fine layer of abstraction that users remain unconcerned if the service and resources are being shared with others [7], [8]. The cloud service providers (CSP) are responsible for the resource pooling.

Rapid Elasticity

Rapid elasticity refers to the scalable feature of the cloud platform. In order to exploit the elasticity of a cloud infrastructure, the applications need to be able to scale in (adding additional resources) and scale out (removing those resources which are no longer required). The elasticity of cloud platform is considered as one of the most important characteristics of the cloud computing [9].

Measured Service

The main idea of the measured services is the economy of the scale and refers to pay only for the consumed resources or per-usage business model. This pricing model and elasticity of the compute resources provides proficient use of the capital and agility [10]. To maintain the transparency of what is consumed, different cloud service providers are offering tools to maintain billing alarms to limit the resource consumption.

3.3 Cloud Classifications and Service Models

Cloud computing can be classified in different ways depending on the type of service they offer and technical implementation. Generally, cloud services are denoted as “X as a Service”, where X can represent as either Hardware, Infrastructure, Platform, Framework, Application and sometimes Datacentres. Important aspects and key characteristics of different kinds of cloud offerings are highlighted in Figure 2.

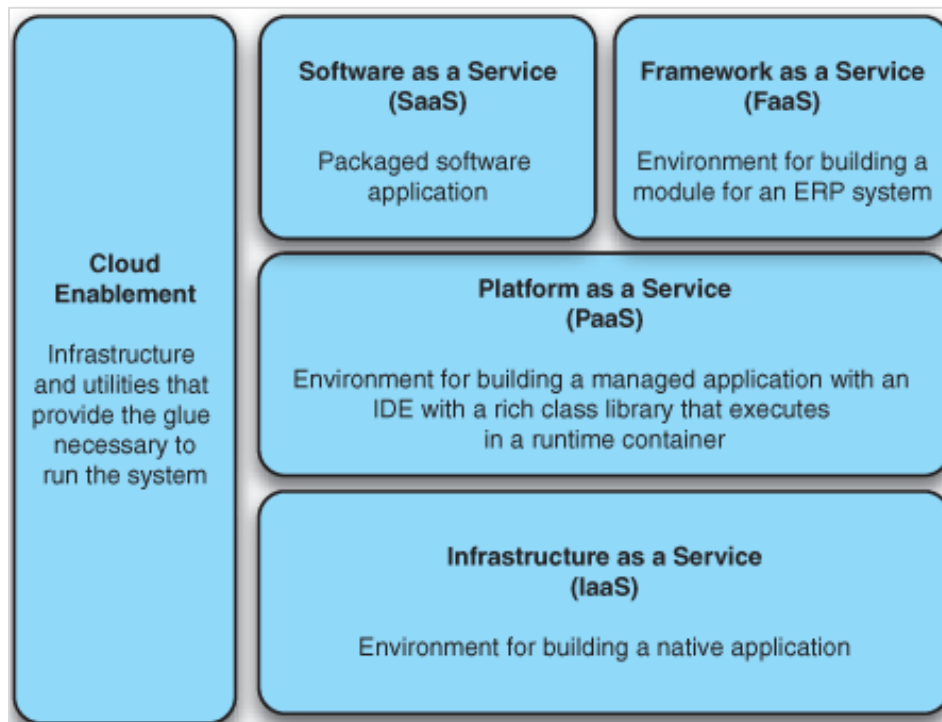


Figure 2. Cloud Classifications, Everything as a Service [4, Fig. 1.7].

In Figure 2, cloud services are classified based on the type of resource and service offered to the cloud service users. Level of flexibility and complexity varies at different layers. This is usually represented by scope the and control model of cloud layers and is shown in Figure 3.

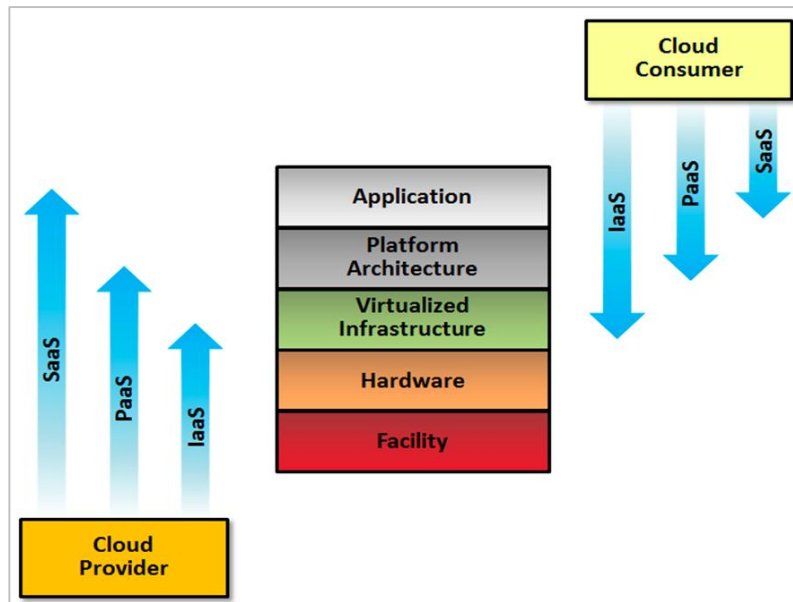


Figure 3. Scope and Control of Cloud Service Model [11].

Scope and control refer to the different levels of scope and control to the producer and consumer for each deployment model. A brief introduction to each service models of the cloud computing is described in the following sections.

Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS), or sometimes called Hardware as a Service (HaaS); is a form of a cloud computing which provides on-demand physical and virtual computing resources e.g. storage, network, firewall, and load balancers. To provide virtual computing resources, IaaS uses some form of hypervisor, such as Xen, KVM, VMware ESX/ESXi, Hyper-V.

A user of IaaS is operating at the lowest level of features available and with the least amount of pre-packaged functionality. An IaaS provider supplies virtual machine images of different operating system variations. These images can be tailored by the developer to run any custom or packaged application. These applications run natively on the selected OS (Operating System) and can be saved for a particular purpose [4]. The user can use instances of these virtual machine images whenever needed by starting the particular instance. The use of these images is typically metered and charged in hour-long increments. Storage and bandwidth are also consumable commodities in an IaaS environment, with storage typically charged per gigabyte per month and bandwidth

charged for both inbound and outbound traffic [12]. Figure 4 represents the scope and control of IaaS model and is depicted below:

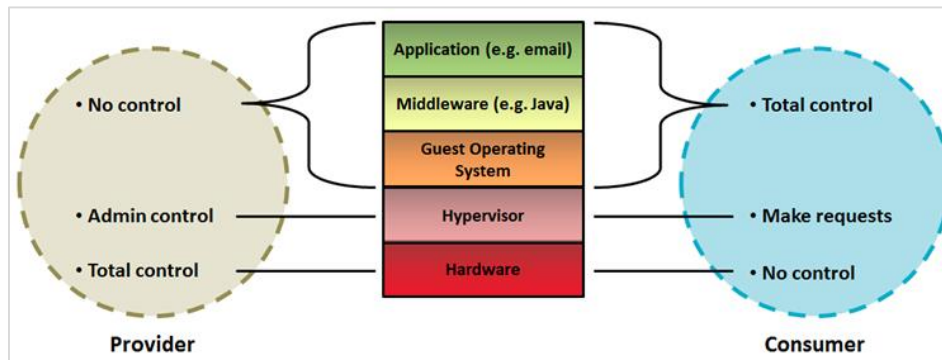


Figure 4. IaaS, Scope and Control [13], [14].

As shown in Figure 4, the provider maintains total control over the physical hardware and administrative control over the hypervisor layer. A consumer may make requests to the cloud (including the hypervisor layer) to create and manage new VMs (Virtual Machines) but these requests are privileged only if they conform to the provider's policies over resource assignment. Through the hypervisor, the provider will typically provide interfaces to networking features (such as virtual network switches) that consumers may use to configure custom virtual networks within the provider's infrastructure. A user of a cloud service maintains full control over the guest virtual machine's operating system and same is applied to the software running on the guest operating system [15].

Platform as a Service

Platform as a Service (PaaS) is a class of cloud computing services which allows users to develop, run, and manage applications without taking care of the underlying infrastructure. With PaaS, users can simply focus on building their applications, which is a great help to developers. PaaS provides access to the deployed applications and sometimes hosting configurations of the cloud environment but users of this service do not control the physical resources and operating system neither hypervisor.

Figure 5 represents the scope and control structure of the PaaS model and is shown below:

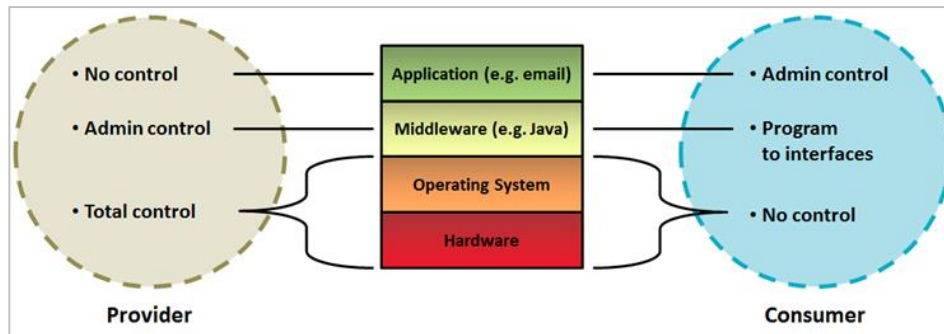


Figure 5. PaaS, Scope and Control [16].

Figure 5 illustrates how control and management responsibilities are shared in PaaS. The centre depicts a traditional software stack comprising layers for the hardware, operating system, middleware, and application. The provider operates and controls the lowest layers such as operating system and hardware. The provider also controls networking infrastructure such as LANs and routers between datacentres. The provider allows consumer access to middleware through programming and utility interfaces. These interfaces provide the execution environment where consumer applications run and provide access to certain resources such as CPU cycles, memory, persistent storage, data stores, databases, and network connections. The provider determines the circumstances under which consumer application code gets activated, and monitors the activities of consumer programs for billing and other management purposes [17].

Software as a Service (SaaS) and Framework as a Service (FaaS)

SaaS refers to services and applications that are available on an on-demand basis. Perhaps the most commonly used cloud service for general purpose is SaaS, which represents the availability of provider's applications to cloud users. The user can access these services from client devices via web browsers over the Internet [18]. In SaaS, the consumer has no control over the cloud infrastructure components as these resources are controlled by the cloud service providers. The consumers only have limited control of user-specific application configurations [19], as represented by Figure 6.

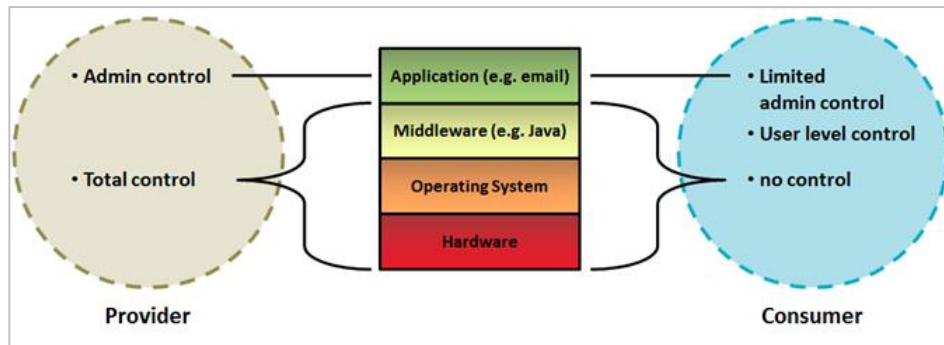


Figure 6. SaaS and FaaS, Scope and Control [20].

Figure 6 illustrates how control and management responsibilities are shared. In SaaS, the cloud provider controls most of the software stack. A provider is responsible for deploying, configuring, updating, and managing the operation of the application so it provides expected service levels to consumers. A provider's responsibilities include also enforcing acceptable usage policies, billing, and problem resolution to mention few. To meet these obligations a provider must exercise final authority over the application. Middleware components may provide database services, user authentication services, identity management, account management, and much more [21]. In general, however, a cloud consumer needs and possesses no direct access to the middleware layer. Similarly, consumers require and generally possess no direct access to the operating system layer or the hardware layer.

FaaS is an environment adjunct to a SaaS offering and allows developers to extend the pre-built functionality of the SaaS applications as represented in Figure 2. Force.com is an example of a FaaS that extends the Salesforce.com SaaS offering. FaaS offerings are useful specifically for augmenting and enhancing the capabilities of the base SaaS system. [4].

3.4 Cloud Service Usage and Deployment Models

There are four fundamental deployment models of the cloud computing: Public Cloud, Private Cloud, Community Cloud and Hybrid Cloud. These service and deployment models refer to sharing, scalability, security, and cost of the resources within the cloud. Cloud deployment models distinguish cloud environment by ownership, access level and the number of cloud service users [22]. The following sections provide a brief introduction of each mode.

Public Cloud

A public cloud is owned by the cloud service provider (also known as a hosting provider). The cloud service provider provides cloud resources for an organization and users in an organization interact and access the resources over the Internet. The cloud vendor may share its resources with multiple organizations, or with the public. Figure 7 shows an illustration of the public cloud.

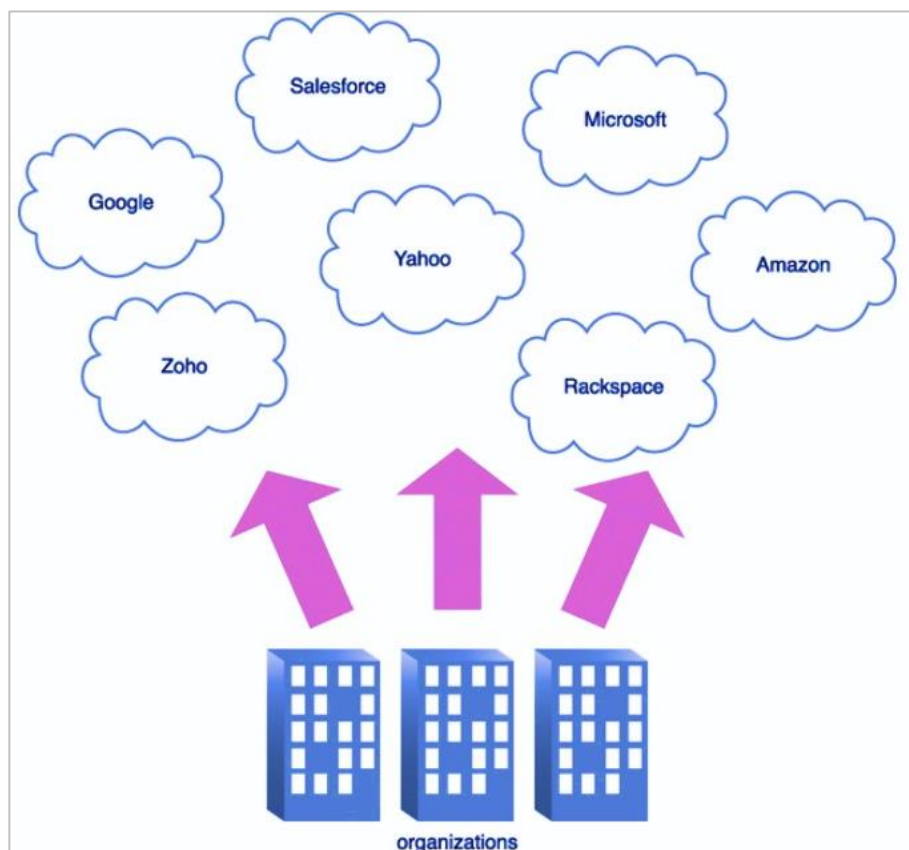


Figure 7. Public Cloud [22, Fig. 4.17].

As shown in Figure 7, several organizations are represented as cloud consumers while accessing the cloud solutions hosted by different cloud service providers.

Private Cloud

A private cloud operates only within one organization on a private network and is a highly secure form of the cloud computing model. It provides cloud functionality to external customers or specific internal departments, such as accounting or human resource department. By creating a private cloud, an organization provides a pool of resources for the infrastructure and the applications are shared with each end user as a tenant with the respective resources that they need. A typical representation of a private cloud is presented in Figure 8.

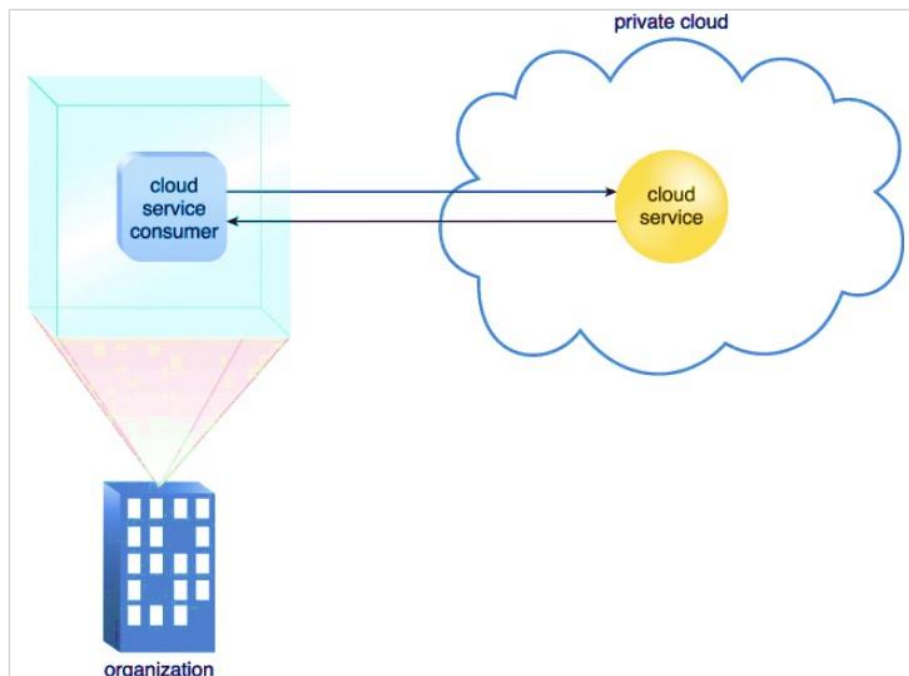


Figure 8. Private Cloud [22, Fig. 4.19].

As shown in Figure 8, the organization is comprised of the on-premises environment and a cloud user consumes the same organization's cloud resources by means of an internal private network. When considering a private cloud implementation, an organization should evaluate carefully whether building its own private cloud is the right strategy. Depending on various factors, such as cost, availability of in-house skills, compliance, and the Service Level Agreement (SLA), it may be better to outsource the hosting of the infrastructure [23].

Community Cloud

A community cloud is quite similar to the public cloud but distinguished by its access to the specified community rather than to the public. The community cloud may be jointly owned by one or few organizations with legitimate need of shared concerns [24]. Figure 9 represents a graphical representation of a community cloud model.

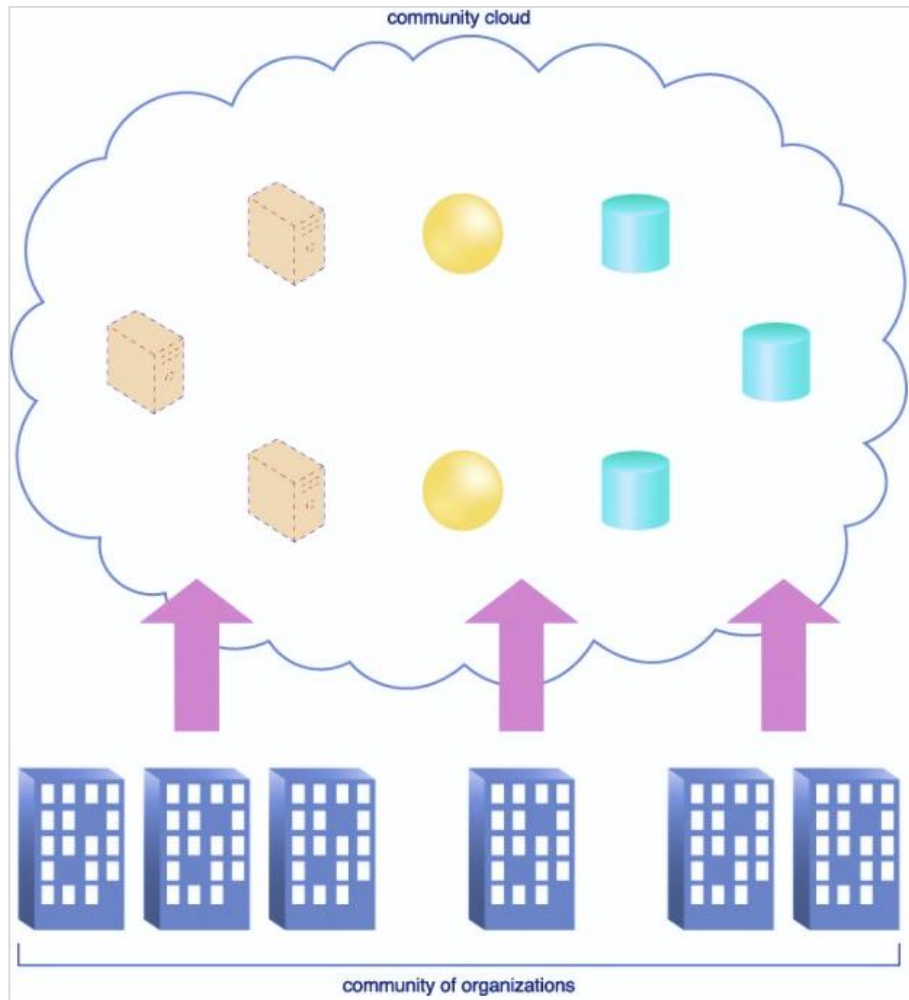


Figure 9. Community Cloud [22, Fig. 4.18].

In above figure, a community of cloud consumers is accessing the IT resources offered from a community cloud.

Hybrid Cloud

A hybrid cloud is a combination of private and public deployment models. In a hybrid cloud, specific resources are run or used in a public cloud, and others are run or used in a private cloud [25]. A hybrid cloud offers benefits from both private and public cloud

models. This may be a preferable strategy for an organization with an interest to control and manage some of workloads locally but also still want to leverage some of the benefits of cost, efficiency, and scale available from a public cloud model. Figure 10 represents a typical structure of a hybrid cloud environment.

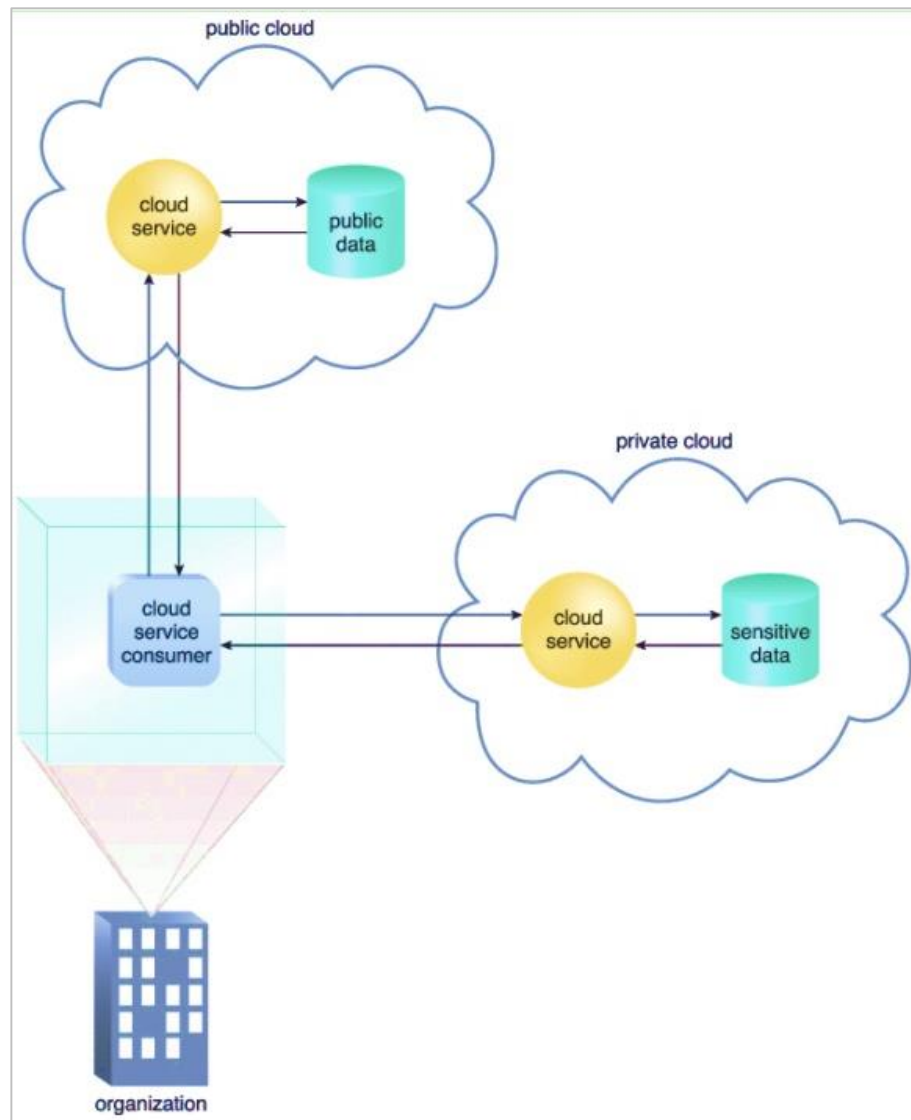


Figure 10. Hybrid Cloud [22, Fig. 4.20].

As shown in Figure 10, an organization is consuming IT resources from both public and private clouds.

3.5 Virtualization of Compute Resources

Virtualization is one of the revolutionary, widely accepted technology and one of the most significant pillars of the cloud computing and is defined as:

Virtualization in computing often refers to the abstraction of some physical component into a logical object. By virtualizing an object, you can obtain some greater measure of utility from the resource the object provides. For example, Virtual LANs (local area networks), or VLANs, provide greater network performance and improved manageability by being separated from the physical hardware. Likewise, storage area networks (SANs) provide greater flexibility, improved availability, and more efficient use of storage resources by abstracting the physical devices into logical objects that can be quickly and easily manipulated [26].

It was 1974 when Gerald J. Popek and Robert P. Goldberg first introduced the framework that provides information regarding the virtualization requirements, attributes and a Virtual Machine Monitor (VMM), also known as hypervisor [27]. A hypervisor is a core software that provides virtualization environment for virtual machines (VMs) to operate [26]. A basic concept of a VMM is illustrated in Figure 11 below.

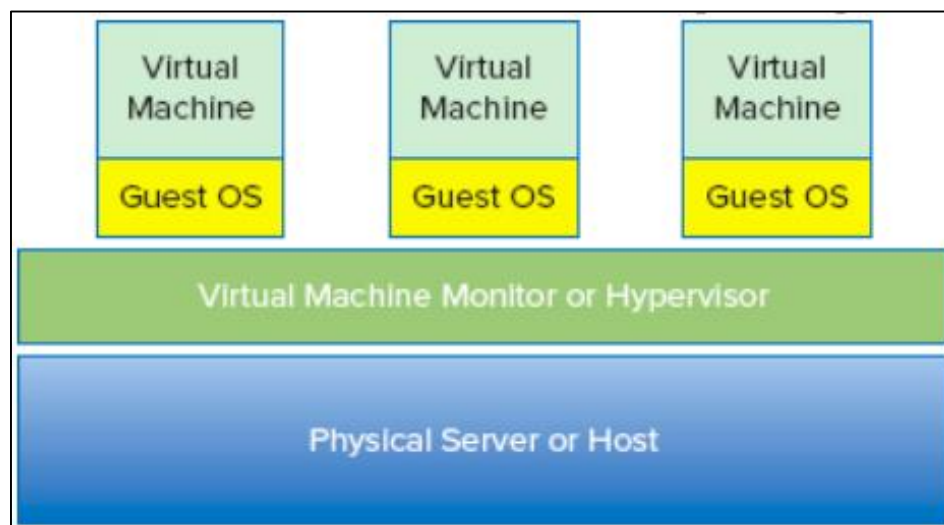


Figure 11. A Basic Virtual Machine Monitor / Hypervisor [26, Fig 1.1]

Figure 11 illustrates that hypervisor or virtual machine monitor (VMM) is running on the top of the physical layer, while each virtual machines (VMs) are running on the top of a hypervisor. This also clarifies that guest OS is communicating with the hypervisor, not to the physical hardware. This is the excellence of a hypervisor to hide all the hardware configuration details from the user to give an idea that VM is running independently. Figure 12 represents the concept of the hardware abstraction.

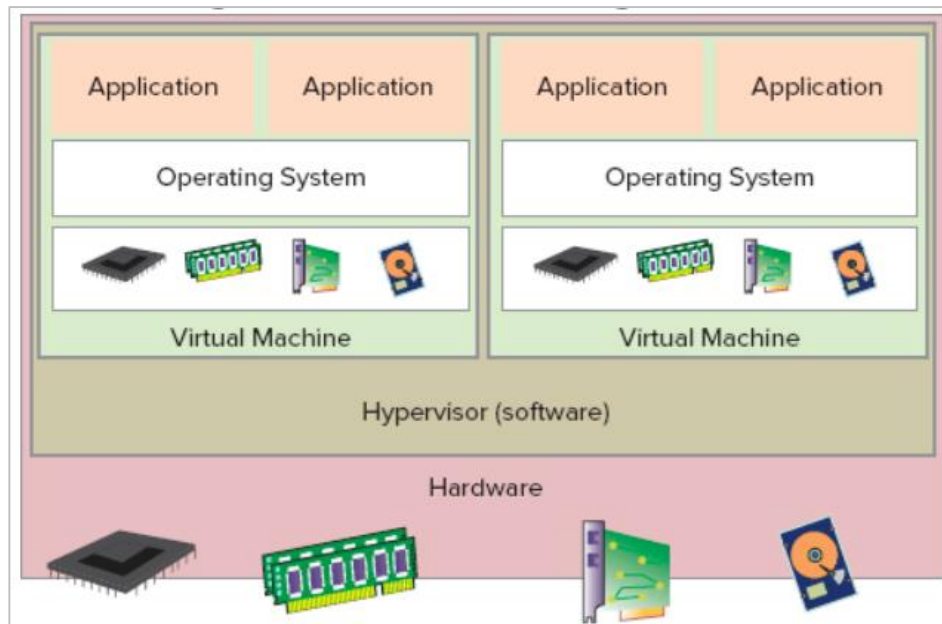


Figure 12. Hardware Abstraction [26, Fig. 2.6]

Figure 12 shows that a hypervisor resides between the hardware and virtual machines and provides a way for VM to communicate with and exchange computing resources. Requests generated by guest VMs are served by the hypervisor in a timely manner and with an adequate resource allocation.

3.6 Types of Virtualization

Virtualization can be offered at different hardware layers such as CPU (Central Processing Unit), Disk, Memory, File systems, etc. This means there will be different types of business and user needs that will be facilitated by a particular cloud service [28]. This includes the technical setup carried out and maintained by a cloud service provider. Fundamental types of virtualization include the following:

- Platform Virtualization.
- Network Virtualization.
- Storage Virtualization.

A typical cloud user is not concerned about products and technologies, rather about servicing and consuming resources based on SLA. Actually, users require little or sometimes no knowledge of the details of how a particular cloud service is implemented, hardware specifications, architecture, number of CPU's, and so on. What makes it important

for a cloud user is to understand what the service is and how to use this service via management portal or a self-service portal. Following section briefly explains these fundamental types of the virtualization.

Platform Virtualization

This type of the virtualization deals with the abstraction of the computer resources. The main idea of this technology is to communicate and interact with virtual machine monitor or hypervisor instead of the operating system itself. With this approach, physical resources can be used to form multiple virtual machines (VMs) that can independently run on the physical server. Each individual virtual machine or instances, performs the compute tasks independently and providing such an illusion to the user that the resources weren't being shared by anyone, hence abstracting those details from the user. Maximum utilization of physical resources save power and energy are few of the attributes that platform virtualization offers.

Network Virtualization

The main principle of the network virtualization is the same as of platform virtualization, the ability to run several isolated networks where each network can perform tasks transparently from other networks. It is quite common for VMs to have a specific network that they can use to communicate and share resources while maintaining isolation from the other VMs by means of virtual networks. Depending upon the selected hypervisor, there may be different approaches and options for the network virtualization.

Storage Virtualization

Storage virtualization is the ability to use and mix multiple storage devices regardless of physical hardware and logical volume structure and abstracting all the underlying details from the user. This technology allows storage administrators to divide and distribute the storage in a well-structured manner. With heterogeneous storage devices, business-critical applications and valuable information that requires fast processing can be hosted on significantly fast and efficient storage medium, such as solid-state disks (SSDs). For other types of data where speed and performance is not a primary concern, a storage administrator can utilize relatively slow disks (low in price). Another feature powered by storage virtualization is the availability of file-based access to data no matter where the actual data is actually stored. Consumers usually remain unaware about the fact that where the files are actually stored, how the storage has been configured and the types of disk involved (rotational disks or SSDs) [29], [30].

3.7 Scalability

The effective allocation and management of the compute resources to ensure enough resources are available for an application is called scalability. According to B. Wilder, scalability is defined as:

The scalability of an application is a measure of the number of users it can effectively support at the same time. The point at which an application cannot handle additional users effectively is the limit of its scalability. Scalability reaches its limit when a critical hardware resource runs out, though scalability can sometimes be extended by providing additional hardware resources. The hardware resources needed by an application usually include CPU, memory, disk (capacity and throughput), and network bandwidth [31].

The underlying concept of scalability is concerned with the capability of a system to cope with an increased load while maintaining the overall system performance. Scalability elements include the following:

- Application and its Ecosystem.
- Increased Workload.
- Efficiency

Application scalability involves various components including hardware and software and is evaluated at different levels. For a web application, its primary workload is handling HTTP requests for a certain time period. The requested workload should be adaptable if the allocated resources are according to the normal workload, meaning the system performance is not compromised when processing the web requests. Usually, the web traffic is dynamic in nature and sometimes the number of expected web requests are not known in advance and the system is subject to failure in terms of request processing. The efficiency of a web application, therefore, includes the throughput, Service Level Agreement (SLA), number of executed transactions per seconds (TPS) and response time.

In terms of application scalability, general description of the scalability refers to the concurrent application users and a desired response time. A number of the concurrent users generates the activity and demands resources to be available with an acceptable response time. Response time refers to the time it takes between request generation and request fulfillment.

3.8 Little's Law

Little's theorem [32] is related to the capacity planning of the system and provides a foundation for scalable systems. This theorem is well known in queuing theory due to its theoretical and practical importance [33].

S. K. Shivakumar describes the little's theorem in terms of scalability and defined as:

For a system to be stable, the number of input requests should be equal to the product of the request arrival rate and the request handling time [34].

Formal notation of the Little's Law [34] is as follows:

$$L = \lambda \times W$$

L =Average number of requests in a stable system.

λ =Average request arrival rate.

W =Average time to service the request.

Scalability primarily deals with the optimization of the average time to service the request (W) using infrastructure and software components. To understand the above equation, consider an example scenario with following assumptions:

- Number of concurrent requests in 1 second= 100.
- Time spend on each request= 0.5 second.

Now the average number of request the system may handle can be determined as:

- $100 \times 0.5 = 50$ requests.

This shows that to increase the number of requests that can be facilitated concurrently, optimization in the request servicing time is required, represented by W in above equation. In today's dynamic and rapidly growing era, the need for scalable web applications is required more than ever. Scalable systems also determine how the business can manage the future growth. For example, an online business web application starts responding slowly because the system is not designed to cope with the unexpected spike in web traffic. Similarly, an online business may drop potential business deals due to poor user experience during the sale season because of the immense increase in web traffic.

3.9 Scalability Layers

Understanding the layers involved in establishing end-to-end scalability is the first step in the understanding the scalability. Figure 13 depicts the scalability request processing chain based on a sequence and a contribution order. This includes a request generation from user's web browser to organization's infrastructure such as security appliances, application load balancer, and other network components. System software or operating system receives the user request and routes it to the requested web server to deliver the request to the appropriate web application.

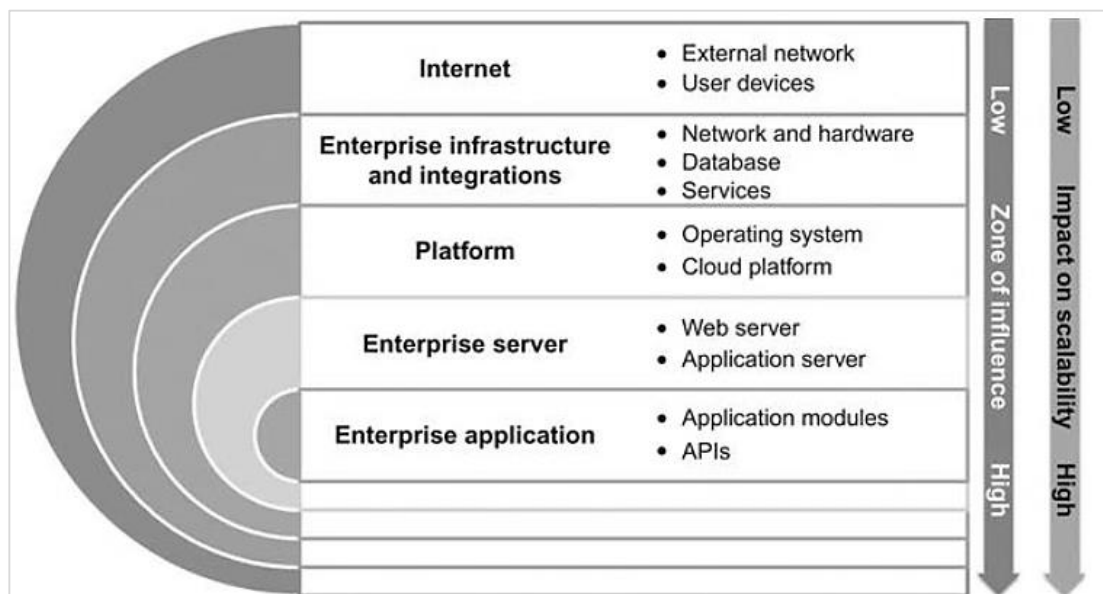


Figure 13. Scalability Layers [34, Fig 1.1]

The above diagram represents the abstraction from underlying computing components such as shared networks, security infrastructure, database management system (DBMS) and Enterprise Resource Planning (ERP) systems. In Figure 13, Enterprise infrastructure and integrations represent this abstraction. Understanding the scalability layers helps to recognize scalability challenges and potentials issues in a system under consideration. In the context of enterprise web application, control of some scalability layers are outside the scope of an enterprise, such as the Internet layer, depicted in Figure 13. However, some layers offer high control and opportunity to fine tune the scalability, e.g. enterprise application layer, assuming other layers are equal, comprising on the internet and client infrastructure. Though the above diagram is used to represent the scalability layers, the same analysis is applied for other quality attributes including the availability of a web application [34].

3.10 Scalability Design Process

Depending upon the business requirements and nature of the web application, scalability needs to be considered at various levels. For example, a business SLA (Service Level Agreement) includes that a particular web application should have a response time of X seconds (for example 2 seconds), and capable to manage Y number of TPS (Transactions Per Second). Commonly used scalability design steps are depicted in Figure 14.

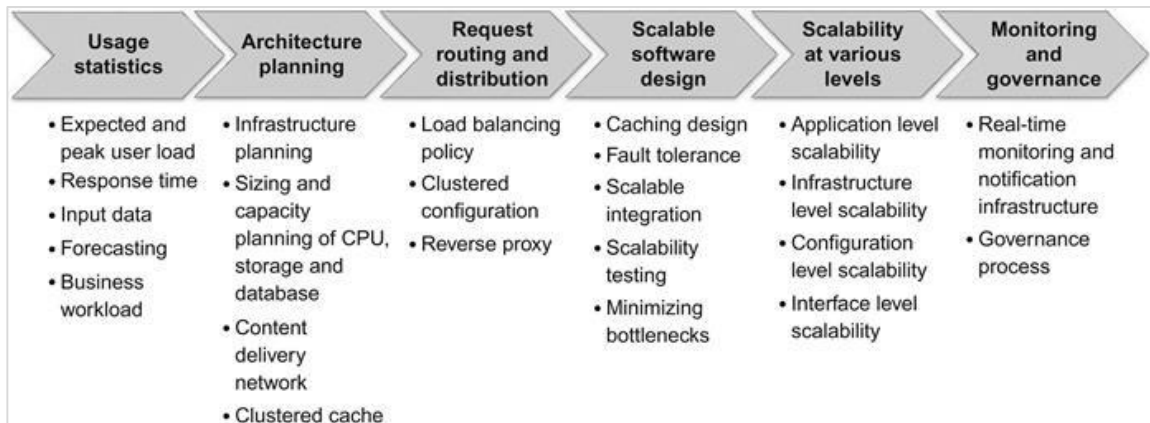


Figure 14. Scalability Design Process [34, Fig 1.11]

As depicted in Figure 14, the fundamental stages in scalability design process can be carried out at various scalability layers in designing components. These stages are applicable at the time of infrastructure planning and while designing software modules.

Scalability metrics represents variables to be monitored over a defined time frame. As stated earlier, depending on the business and user requirements, scalability attributes of the proposed system may vary. Therefore, it's very important to understand the key performance indicators (KPIs) about the system under consideration [35].

Following are few examples of the KPIs:

- Maximum number of TPS (Transactions Per Second).
- A total number of concurrent logins at a given time.
- Expected response time to facilitate user request.
- Task completion time.
- User traffic per availability zone (geographic region).

These statistics help to design a scalable system very close to the actual requirements. These scalability measures provide an insight into how to design a scalable system especially for the dynamic application usage and growth in the business.

Infrastructure planning is another critical factor and deals with capacity and sizing of the components. Following points need to be considered to achieve optimal performance:

- Analysis of the current demand.
- Analysis of the current capacity.
- Planning for the future capacity.

After collecting the information about the current and future demand and workload on a particular system, next stage should be the estimation of the current capacity to determine if it meets the demand. Evaluation of the provisioned resources is also required to understand if resources are over or underutilized by establishing the threshold and benchmark values. Several hardware and software vendors nowadays provide information about the minimum requirements and recommended configurations. This information can also help when planning the infrastructure capacity and estimate the optimized capacity. Even with capacity planning and provisioning the estimated hardware resources, still, one needs to evaluate the web application with some benchmarking tools. Reason for this is that typically all factors that cannot be determined accurately enough during the capacity planning phase.

In terms of high availability and performance, it is recommended to implement load sharing mechanism that will help to distribute the load to make efficient use of hardware. Load balancing can have several forms depending upon the nature of the application and allowed budget. Scalability monitoring governance refers to the quality measures that are conducted to manage error handling in the system. Well-defined rules and monitoring alarms are used to notify the service provider in case there is an error in the infrastructure or if an application is unavailable due to some reasons. Monitoring alerts can further be categorized into specific components, such as CPU utilization, network load, memory monitoring, database, and app monitoring.

3.11 Scaling Approaches

The applied scalability methodology used to provide additional hardware resources defines the following two scalability approaches, provided application can utilize those newly assigned resources effectively.

- Vertically Scale Up.
- Horizontally Scale Out.

The following section provides a brief introduction of those scalability approach.

Vertically Scale Up

This scalability approach is also known as vertical scaling or scaling up. The underlying concept of this approach is to improve the application capacity by additional hardware resources within the same box. The box is also known as compute node or a virtual machine (VM) running the application logic. The operations performed in vertical scaling include increasing system memory, an additional number of CPU cores and other similar activities. Due to its low risk, this approach has been used as a most common way to provide additional resources and maintaining the budget with modest hardware improvements. Even with the availability of the hardware resources, it is not guaranteed that if the application can gain the advantage of these hardware resources. A downtime is also expected in this approach because hardware changes often require the system to be shut down which cause service interruption.

Horizontally Scale Out

This scalability approach is also known as scaling out or horizontal scaling. It is based with the increasing application capacity by adding more data nodes, such as new virtual machine instances. In most of the cases, the newly provisioned nodes provide the exact capacity as was with the existing node. When comparing with vertical scaling, the architectural challenges and level of complexity involved tend to be more apparent in horizontal scaling because the scope is shifted from individual node to several nodes. Horizontal scaling tends to be more complex than vertical scaling, and has a more fundamental influence on application architecture [31].

4 Scalable Cloud Architecture for a Web Application

This chapter presents the proposed scalable architecture for a web-based application configured to run and deploy with the scalable cloud platform. Following sections describe the different tiers and components involved in this reference architecture. Attributes of the management workstation, also known as management node are also mentioned briefly to understand the purpose of the management node.

4.1 Scalable Web Application Reference Architecture

This section describes the overall design for the web-based application that was intended to implement using the scalable cloud platform. WordPress, an online content management system [37] was selected as an example web application to host on cloud platform running on Apache Web Server [38]. Figure 15 illustrates this reference architecture model and associated tiers.

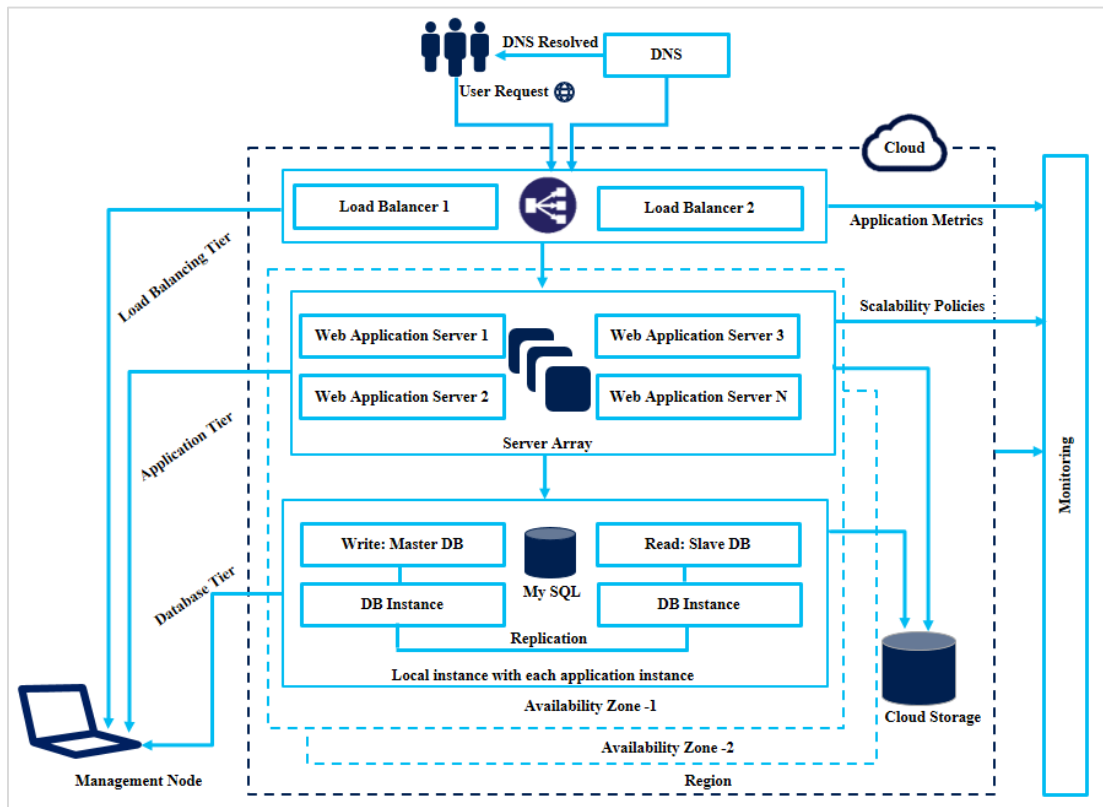


Figure 15. Scalable Web Application Reference Architecture

The architecture presented in Figure 15 looks similar to the traditional three-tier application model architecture [39] with some enhancements. Following sections of this chapter explores the tiers involved in this reference architecture while chapter 5 covers the actual implementation with Amazon Web Services cloud platform.

4.2 Load Balancing Tier

The first tier depicted in the scalable web application reference model (Figure 15) is the load balancing tier. The concept and the implementation of the load balancer implementation is not a new practice. Load balancing has been used in several systems with different types and needs of the load to be balanced. Generic examples include client-server load balancing, network infrastructure like routers to distribute the load across multiple paths that are directed to the same destination [40]. The purpose of the load balancing tier in the scalable cloud platform is to distribute the application load among server array that participates in the particular load balancer. With load balancer implementation, problems like single node failure can be reduced and results in application availability and responsiveness [41]. When using scalable cloud platforms, such as Amazon Web Services [42], application servers can easily have an association to cloud load balancer and can increase/decrease the number of required servers depending upon the resource demand. User requests for the web applications received by the load balancer and forwarding user requests to the member servers are one of the important tasks of load balancers. Assuming an application server started to malfunction, user requests should be forwarded to another node in the load balancing tier. Load balancers together with scaling policies help maintain web application highly available without impacting on the overall performance.

Traffic Pattern and Load Balancer

In terms of receiving and establishing Concurrent Connections (CCs), the Central Processing Unit (CPU) is actively involved to facilitate the huge amount of the web requests. The performance of a load balancer is related to the compute capacity of the target load balancer, as shown in Figure 16.

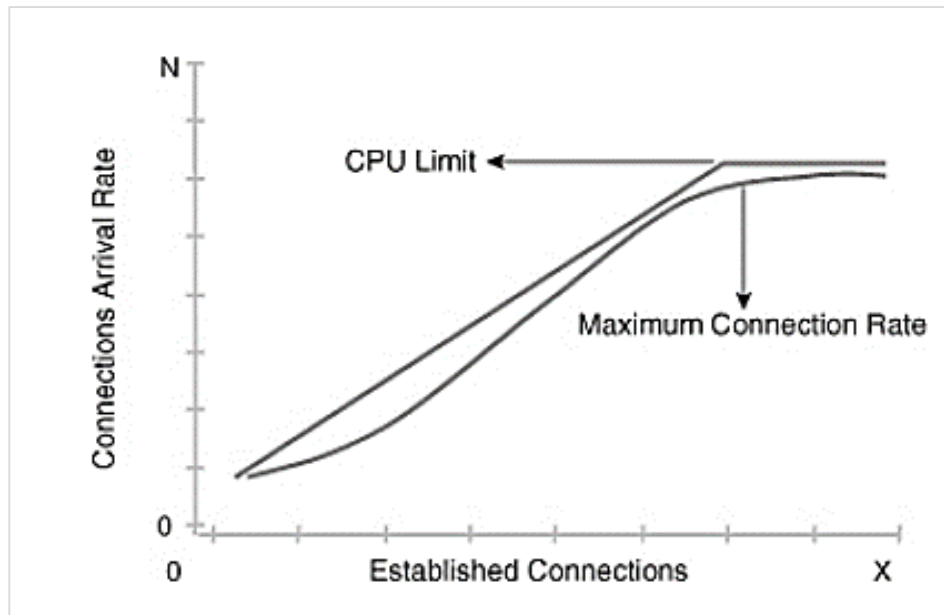


Figure 16. Connection Rate Curve of the Load Balancer [43, Fig 22-8].

As depicted in Figure 16, with the increases in the new connections, the number of the connections a load balancer can manage increases, resulting in the high CPU utilization. This also indicates that a load balancer has reached its capacity with the flattened curve. Response time is another crucial pattern when considering the performance of a load balancer and web application. Response time is usually measured in milliseconds and is referred as:

The elapsed time between the end of an application layer request (the user presses the Enter key) and the end of the response (the data is displayed in the user's screen) [43].

Response time also helps to estimate the capacity of a system with measurable methods to determine if the requested content is available to the client. Also, it measures the amount of time (in milliseconds) the users have to wait to receive the contents.

Figure 17. Illustrates the server response time.



Figure 17. Server Response Time [43, Fig 22-10].

The response time is not a direct metric to evaluate the load balancer and because it's typically used to measure the performance of a web server. However, a web server traffic is distributed and regulated with a load balancer, causing response time to act as an indirect performance indicator for the load balancer [44].

4.3 Application Tier

The second tier in the scalable web application reference architecture is the application tier, comprising of the application servers and associated server array, as shown in Figure 15. Web Application Servers 1 to N; represents the virtual machines, called EC2 (Elastic Cloud Compute) instances in terms of Amazon Web Services (AWS) [46]. The minimum number of application servers recommended in this reference architecture is two EC2 instances running in different availability zones. AWS compute resources are hosted in different geographical locations that constitute a region and availability zones. A region is a separate geographical area, and each region has multiple isolated locations that are known as availability zones [47]. In Figure 15, the availability zones are represented by "Availability Zone-1" and "Availability Zone-2". Both availability zones are located in the same "Region". Ireland region is one of the AWS regions in Europe [48] that was selected for the implementation and experiments.

Apache web server was used to host the WordPress web application running on AWS EC2 instances. The multiple application servers shown in Figure 15 are actually a clone of the virtual machine that provides exactly the same configuration as if it was the original virtual machine. These EC2 instances were deployed in different availability zones in order to make web application highly available. If all servers are located in one availability zone and there is service interruption in that particular zone, all the resources will be out-of-service. Having resources in multiple zones helps to reduce this risk in service interruption. The size of the server array expands with additional servers with an increase in the resource demands (scale in) and decreases when there is a decline in the current demand (scale out).

With AWS, the minimum and a maximum number of EC2 instances can be defined as part of the scaling policies. The minimum size of the available instances can be increased any time by cloud administrators to meet the business needs and application demands. The maximum size defines the limit of the server array and scaling policies should continue to provision the EC2 instances until the maximum limit has been reached. As soon as the scaling policies determine the decline in the current demand, assigned resources are released until the number of EC2 instances reaches to the minimum instances defined. This helps to maintain the budget as a cloud user is billed only for the consumed resources over a certain period of time.

4.4 Database Tier

The database is the third tier in the scalable web application architecture illustrated in Figure 15 and one of the critical component in the infrastructure planning and design. Database tier plays a significant role and in today's era of information age, there is very rare chance for a web application to run without a database. In terms of heavy load on a web server, the number of database connections that a database management system allows to facilitate user requests determines the performance of a database management system (DBMS). For the purpose of this study, MySQL database was selected due to following significant characteristics [50]:

- Speed
- Ease of Use.

- Structured Query Language (SQL) Support.
- Connectivity and Security.
- Low Cost and High Availability.

Figure 15 illustrates the recommended best practice for the web application database when deployed in the cloud platform. Although the cloud service provider is responsible for the uptime and ensuring that hardware is running flawlessly, still the best practice is to deploy the MySQL database in master/slave model. This approach guarantees the application availability if the master database becomes unavailable. As discussed in the application tier, recommended practice is to deploy them in multiple availability zones that span across the region. Database redundancy is quite important as a database is the location that stores the data about each transaction as well as metadata. Depending on the nature of the application, a database may be more disk write intensive while some databases may tend to more read oriented. This is why it is important to design the database tier as multiple instances to reduce the potential data loss that results in service unavailability.

4.5 DNS Server and User Requests

A user interacts the cloud service with a web browser that sends a request to the web server [51]. A typical user identifies a resource on the internet with its distinguished name, for example; `www.MyUserFriendlyDomainName.Com`. Each distinguished identity has been assigned with a unique address, called Internet Protocol (IP) address. An IP address constitutes a network and hostname segments and in terms of networking, user requests are routed over the Internet based on this unique IP address. An IP v4 (version 4) is still the widely used IP addressing scheme though there exists IP v6 as well. An IP v4 packet consists of 32-bit addressing scheme and the standard format is; `nn: nn: nn: nn` [52]. This format of IP address is difficult to remember and write when accessing web applications. To eliminate this problem, Domain Name System (DNS) is used that leverage the user-friendly name and maps the IP address of the respective domain name. However, when a user interacts with a web browser and sends a query to a web server, an actual resource on the Internet must be located with the IP address and browsers need to forward the request to the correct IP address. DNS performs this transformation of the user-friendly domain name to IP address and performs domain name resolution.

By default, Amazon Web Services (AWS) provides a generic domain name based on instance types and configurations. This web address is a user-friendly but still long string to remember, for example, EC2-52-50-7-237.eu-west-1.compute.amazonaws.com. It is possible to obtain a static IP addresses from AWS that can be mapped to a domain name. This static IP address is called elastic IP address (EIP) in AWS terminology and designed for the dynamic cloud computing. An elastic IP address is IP v4 IP address which can be accessed from the Internet and associated with one particular AWS account [53]. Amazon cloud computing provides one static IP v4 address at free of charge. Additional IP addresses are subject to an additional cost but applied only for the consumed time duration [54].

This study was conducted with the public DNS name generated by the load balancer and was similar to the “http://demo-lb1-244513860.eu-west-1.elb.amazonaws.com”. In the scalable reference architecture (Figure 15) client’s requests are directed to load balancing tier, not to the web server itself. It was due to the fact that load balancer should distribute the received requests to the application tier and servers that form the server array.

4.6 Monitoring and Alerts

Monitoring of the cloud infrastructure is another important design aspect that requires attention and proper implementation. Cloud service providers are responsible for the continuous monitoring of their cloud infrastructure. The consumer of the cloud computing, however, requires to maintain monitoring system that sends warning and alerts as soon a breach of the critical resource(s) has been observed. Amazon elastic cloud compute (EC2) uses monitoring service called Amazon CloudWatch. With Amazon CloudWatch, compute resources and applications can be monitored. This service helped to collect logs files, monitor and track the metrics, send automatic alarms (e-mail notifications) and perform an action depending on the change in AWS resources [55]. AWS CloudWatch service provides following features and benefits:

- Monitor Amazon EC2 Instances.
- Monitor AWS Load Balancers, Databases.
- Monitor Custom Metrics.
- Monitor and Store Logs.

- Alarms Configurations (E-Mail Notifications).
- Plot Graphs and Statistics for Analysis.
- Monitor and React to Resource Changes.

For a scalable web application, it was necessary to configure the monitoring for the compute resources and scalability metrics that checks the system health periodically and perform the required action(s) as defined in the policies. For example, if CPU utilization was X % for Y seconds; CloudWatch were configured to consider it as a breach of the desired threshold and mark it as unhealthy. Additionally, certain actions can be performed automatically such as provisioning a new instance of the virtual machine to mitigate the load or replace it with the unhealthy instance.

Figure 18 shows an example of CloudWatch Monitoring details with a default time interval of 5 minutes.

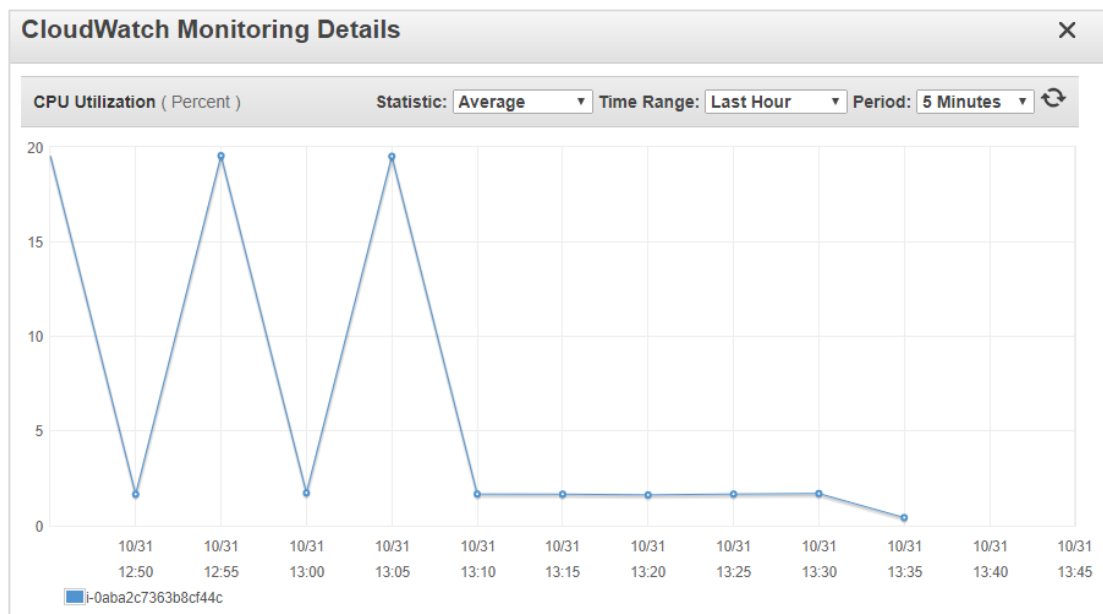


Figure 18. CloudWatch Monitoring (CPU Utilization)

The CloudWatch service is enabled by default for the EC2 instances in basic mode with a default time interval of 5 minutes. This default setting can be modified by selecting “Enable Detailed Monitoring” as shown in Figure 19 and configuring the desired metrics with custom intervals.

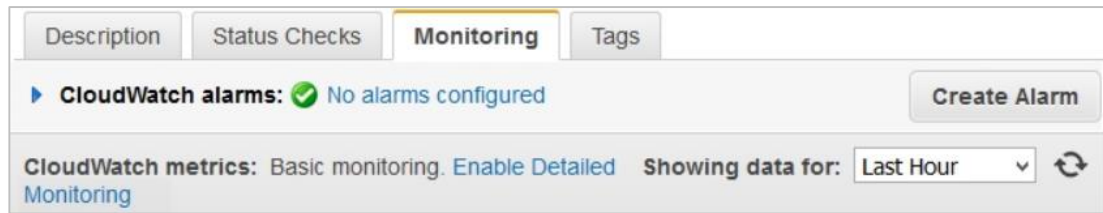


Figure 19. CloudWatch Basic and Detailed Monitoring

Enabling detailed monitoring will incur additional service charges for the EC2 resource monitoring. As depicted in Figure 19, Monitoring tab provides the information about the EC2 instances as well as the option to configure the monitoring alarms to send custom notifications [56].

4.7 Management Node

A management node is a client device in a local environment (on-premises), also known as technician machine. It is used to perform administrative tasks e.g. cloud infrastructure management, performance monitoring, and compliance auditing. This machine is used to interact with the cloud platform, configuring virtual machines, applying scalability policies, test the performance of the web application and scalability metrics. Hardware and software specifications of management node are listed in the below Table 3.

Table 3. Software and Hardware Specifications of the Management Node

Software and Hardware Specifications of the Management Node (On-Premises)					
Hardware Specifications		System Software		Application Software	
Processor (CPU)	Intel Core i3	Operating System	Windows 10 Pro	Microsoft Office	2016
Memory (RAM)	4 GB	System Type	X64-based PC	Java	8 U152
Hard Disk	500 GB	OS Build	14393.693	JMeter	3.3

Table 3 summarizes the software and hardware specifications of the management node. The same machine was used to conduct the experiments to evaluate the performance of the web server by generating heavy load.

Performance testing was crucial in order to determine if the web application under test satisfies high load requirements. It was also used to analyze the overall server performance under heavy load. Stress testing determines the responsiveness, throughput, reliability, and scalability of a system under a given workload. Results of a testing determine the quality of a system and serves as input to implement workload strategies [57]. Apache JMeter is one of the tools designed for this kind of purposes. Apache JMeter is a free, open source, and cross-platform desktop application from the Apache Software Foundation [58]. Some of the features of the Apache JMeter are [59]:

- Performance tests of different server types, including web (HTTP and HTTPS), SOAP, database, LDAP, JMS, mail, and native commands or shell scripts.
- Complete portability across various operating systems.
- Full-featured Test IDE that allows fast test plan recording, building, and debugging.
- Dashboard report for detailed analysis of application performance indexes and key transactions.
- In-built integration with real-time reporting and analysis tools, such as Graphite, InfluxDB, and Grafana, to name a few.
- Complete dynamic HTML reports.
- Graphical User Interface (GUI).
- HTTP proxy recording server.
- Caching and offline analysis/replaying of test results.
- A live view of results as testing is being conducted.

Apache JMeter is a Java-based application that requires Java Standard Edition (SE) Development Kit 8 [60] as a prerequisite. At the time of writing this report, Apache JMeter 3.3 was the latest version available, although Java 9 was the latest version but it was not supported yet with JMeter 3.3 [61]. Appendix 1 (Preparing Experiment Environment with JMeter) provides information on how to download, install, configure and execute JMeter as well as required dependencies.

5 Implementation

This chapter covers the implementation of the proposed scalable web application architecture (described in chapter 4). This provides detailed information for all the stages involved with possible screenshots, step-by-step procedures, and references to the product documentation for further details. Among different cloud service providers, Amazon Web Services (AWS) was selected as a scalable cloud platform for this implementation and experiments. This chapter begins with a motivation statement for using AWS and characteristics that makes AWS distinctive for the cloud administrators and other technology specialists.

5.1 Motivation for Using Amazon Web Services

Amazon Web Services (AWS) is a scalable public cloud platform with a wide range of services and solutions. Following are few quotes that define the AWS and list its essential characteristics and competitive advantages.

Amazon Web Services (AWS) is actually a *huge array of services* that can affect consumers, Small to Medium-Sized Business (SMB), and enterprises. Using AWS, you can do everything from backing up your personal hard drive to creating a full-fledged IT department in the cloud [62].

Amazon Web Services (AWS) is a public cloud provider. It provides *infrastructure and platform services* at a *pay-per-use* rate. This means you get on-demand access to resources that you used to have to buy outright. You can get *access to enterprise-grade services* while only paying for what you need, usually down to the hour [63].

Amazon Web Services or AWS is a comprehensive public cloud computing platform that offers a variety of *web-based products and services* on an *on-demand and pay-per-use basis*. AWS was earlier a part of the e-commerce giant Amazon.com, and it wasn't until 2006 that AWS became a separate entity of its own. Today, AWS operates globally with data centers located in USA, Europe, Brazil, Singapore, Japan, China, and Australia. AWS provides a variety of mechanisms, using which the end users can connect to and leverage its services, the most common form of interaction being the *web-based dashboard* also called as *AWS Management Console* [56].

A wide range of AWS services can be utilized to host a scalable web application (scope of this study), enterprise applications, data mining and big data solutions. Due to the distinctive characteristics of AWS, services are being used by government agencies, medical institutes, scientific and engineering organizations. A user can access the AWS cloud resources from client machine and web interface such as a web browser.

The term web service means services can be controlled via a *web interface*. The web interface can be used by machines or by humans via a graphical user interface. The most prominent services are EC2, which offers virtual servers, and S3, which offers storage capacity [64].

The end user is not required to purchase additional software tools to access the AWS web interface or Management Console. Clients using desktop and mobile operating systems can interact with standard web browsers across multiple platforms. This means clients using Microsoft Windows Operating System [65], Open Source Operating Systems such as Linux [66] and Ubuntu [67], and Apple Macintosh clients using macOS [68] can easily interact with the AWS web services.

For education and training purposes, AWS Educate has introduced student package to give an opportunity to evaluate the AWS cloud platform with an academic user account [69]. Depending on the scope and nature of the project, educational institutes, educators, and students can apply for AWS Educate account to access and master the AWS cloud platform almost for free [70]. GitHub is a development platform used to host and review the code and manage projects. Sharing of the code has been made easy by the GitHub. GitHub Education also offers varied services for students via Student Developer Pack [71]. This student developer pack from GitHub [72] includes offerings from partners, and AWS is one of them. Together with GitHub Education, AWS Educate offering provide value-added service for students, for example, bonus and additional credits.

5.2 Create an Amazon Account

The first step was to create an Amazon account (if not done already). This study was based on the AWS Free Tier [73]. This free tier allows new users to consume some compute resources for free within the certain time frame, usually, 12 months [74], and some resources are free forever [75], [76]. Figure 20 illustrates the look and feel of the AWS account registration.

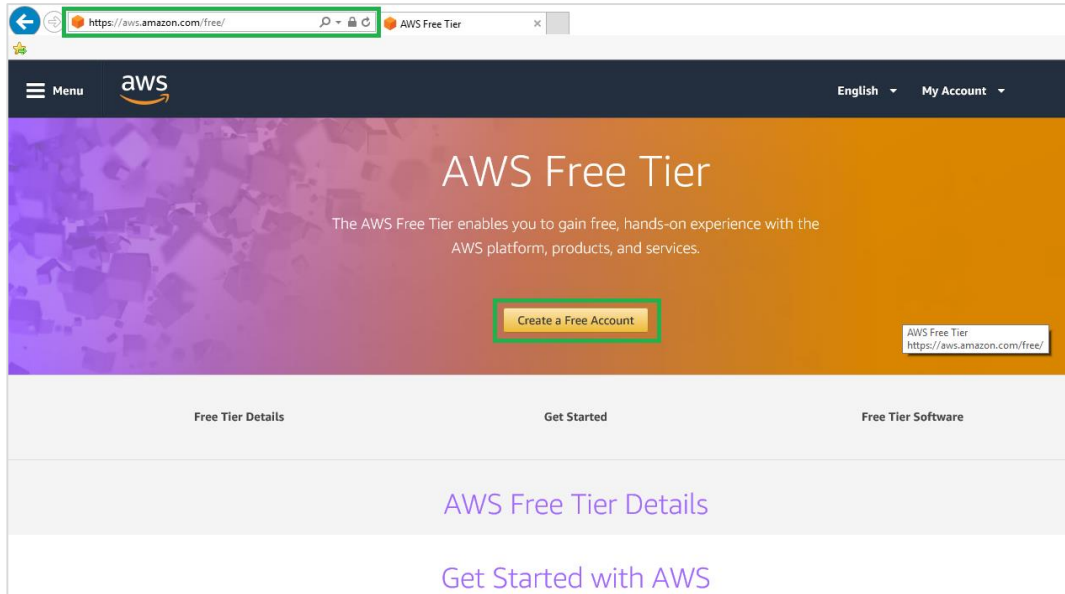


Figure 20. Create AWS Account [73]

As depicted in Figure 20, AWS account registration can be initiated by clicking on the “Create Free Account” button. Once clicked, a user registration form is presented looks shown in the Figure 21.

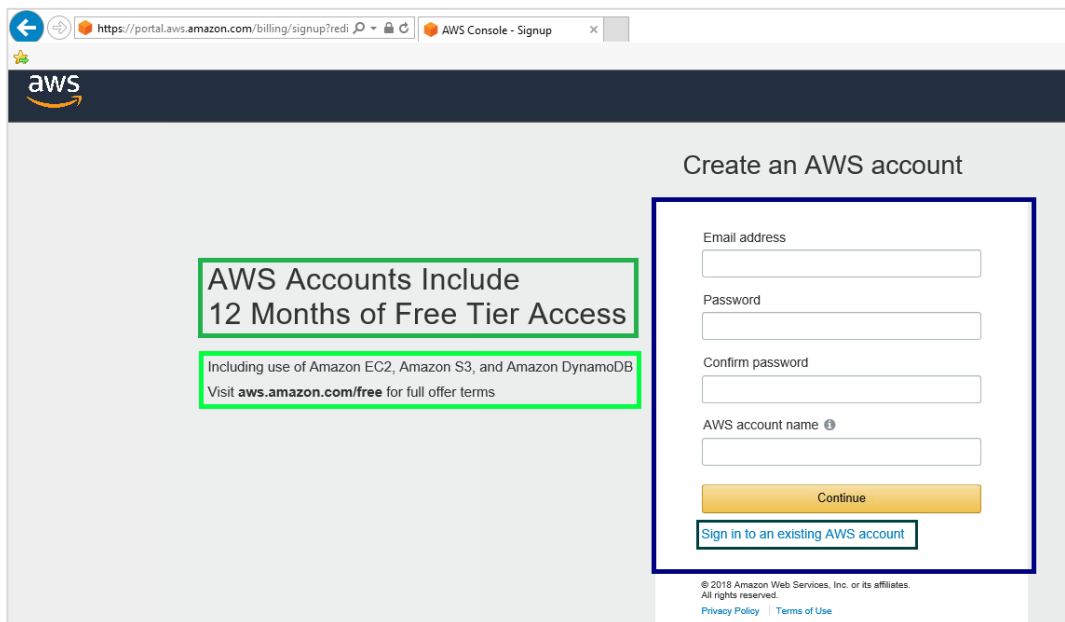


Figure 21. AWS Account Registration Page

On the user information page, some basic information and a brief description of the free tier is available as highlighted in Figure 21. If academic or student account (institutional

email address) is used, AWS requires to provide the credit card information to verify the user identity. Even with a private user email address, the user will not be charged until the consumed resources exceed the free tier limits.

5.3 Create Virtual Server using Amazon Elastic Cloud Compute (EC2)

Once the sign-up process has been completed, the next step is to launch configure the EC2 instances. Following procedure describes how to create and run and access an EC2 instance.

1. Access the AWS console at <http://console.aws.amazon.com>.
2. Sign-in with the credentials specified as part of AWS sign-up procedure.

AWS Services dashboard is shown in Figure 22.

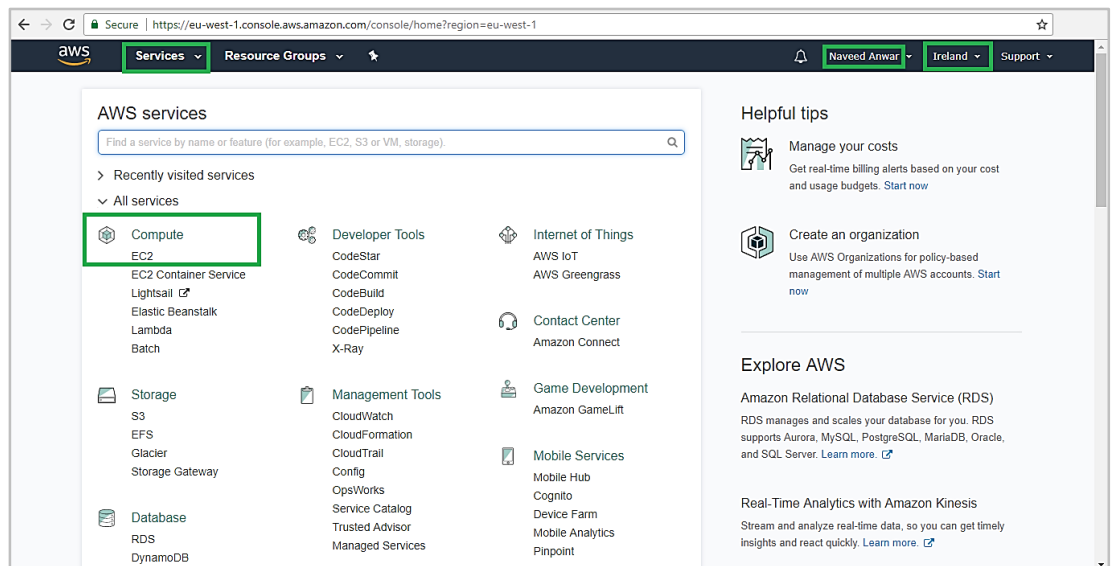


Figure 22. AWS Services Dashboard (Partial Screenshot)

- Click on the EC2 under Compute category. This opens the EC2 dashboard as shown in Figure 23.

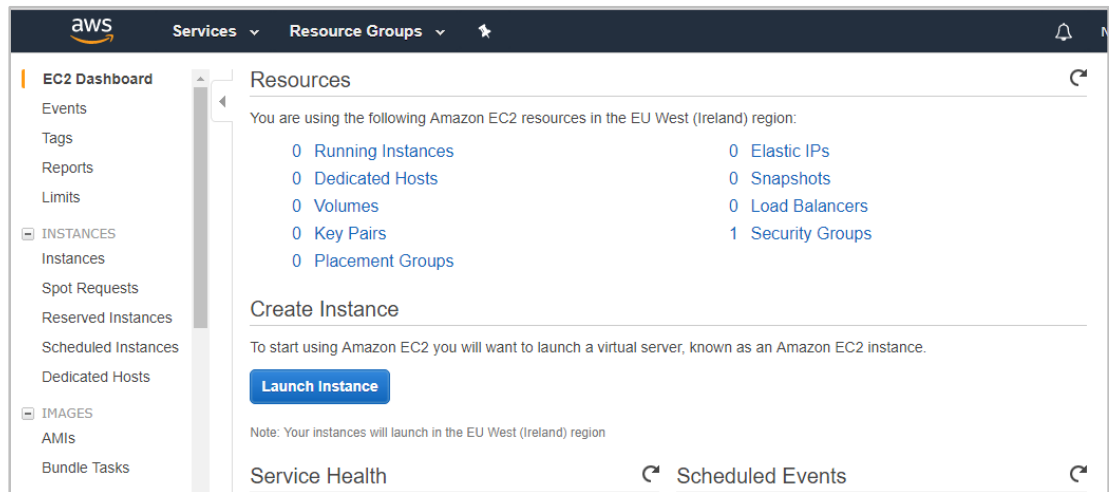


Figure 23. EC2 Dashboard (Partial Screenshot)

Depending upon the usage, the resource listing may vary from what is shown in Figure 23. In this case, there were no running instances at the moment.

- Click “Launch Instance” on the EC2 dashboard to create a new virtual machine (EC2 instance). Following steps needs to be carried out to create an Amazon EC2 Linux instance.
 - Step 1: Choose an Amazon Machine Image AMI → Amazon Linux AMI
 - Step 2: Choose an Instance Type → T2 Micro (free tier eligible)
 - Step 3: Configure Instance Details → Accept default settings

Figure 24. Configure Instance Details

Select Enable to “Auto-assign Public IP” and Click Next: Add Storage. Default settings were adequate enough to proceed to the next step as depicted in Figure 24. “Auto-assign Public IP” can be selected as “Enable”. Click Next to proceed to next step.

- Step 4: Add Storage → Accept Default

Default size was 8 GB SSD disk, which was sufficient for this study goals.

- Step 5: Add Tag → Optional Tag
- Step 6: Configure Security Group → Existing Security Group (Default)

Here choose “Select an existing security group” to select the “default” security group and then click review and launch.

- Step 7: Review Instance Launch → Launch

Step 7: Review Instance Launch
Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

▼ **AMI Details** [Edit AMI](#)

Amazon Linux AMI 2017.09.1 (HVM), SSD Volume Type - ami-760aaa0f
The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.
Root Device Type: ebs Virtualization type: hvm

▼ **Instance Type** [Edit instance type](#)

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

▼ **Security Groups** [Edit security groups](#)

Security Group ID	Name	Description
sg-e72f5d81	default	default VPC security group

[Cancel](#) [Previous](#) [Launch](#)

Figure 25. Review Instance Launch

This screen provides the instance details that were going to be created. If everything is set correctly, click on the “Launch” to bring this instance up and running, as shown in Figure 25.

5. Key Pair Selection / Creation

A key pair is required to connect to the EC2 instances. Select a key pair (if already exists) or create a new one, as shown in Figure 26.

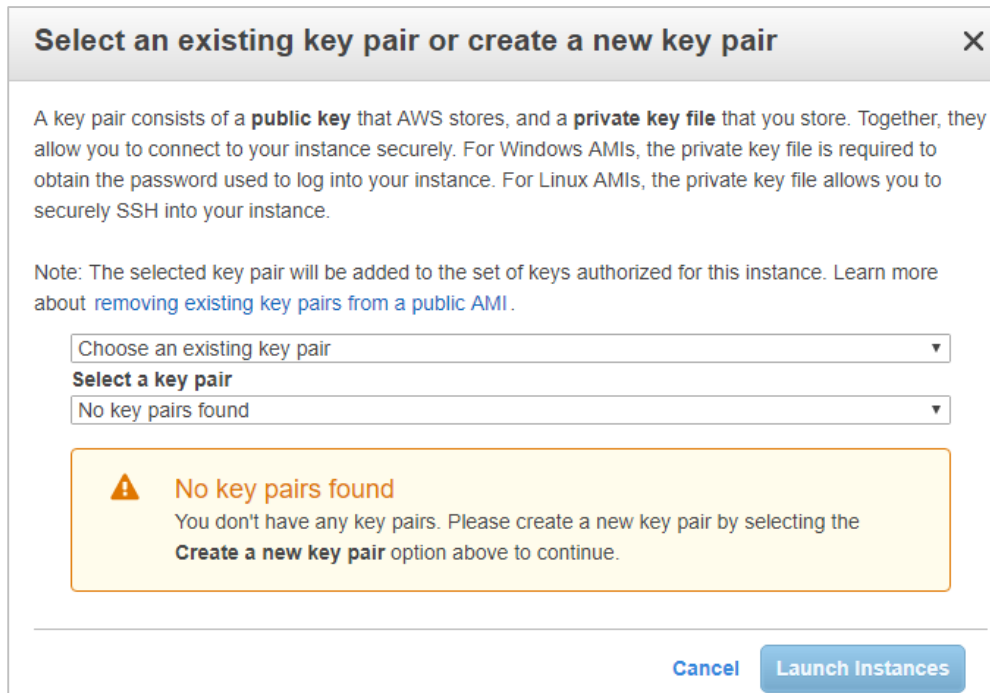


Figure 26. Select an Existing Key Pair or Create a New Key Pair

As depicted in Figure 26, either the existing key pair can be selected (browse to the location where the key pair file is saved) or create a new key pair.

6. Assuming there is no existing key pair, create a new key pair and download it on the management node, as illustrated in Figure 27 shown below:

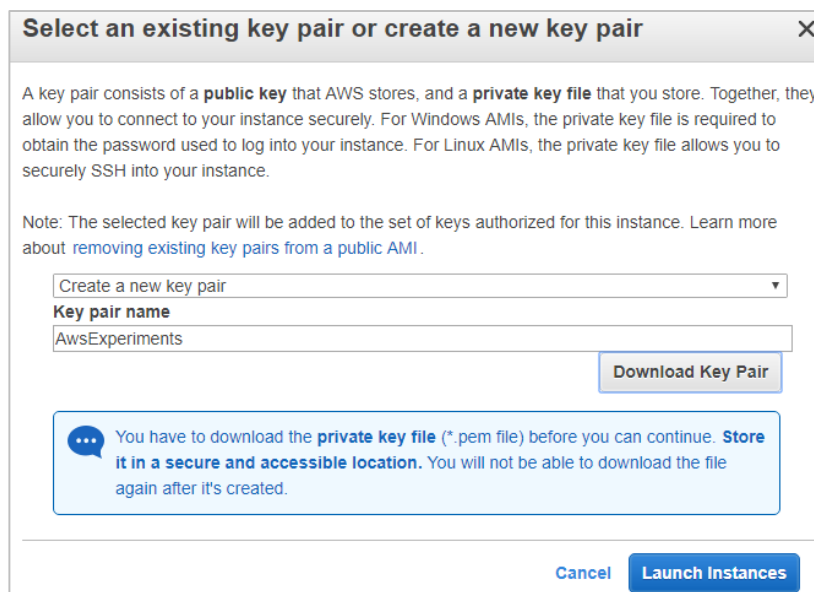


Figure 27. Creating a New Key Pair

Give it a suitable key pair name while selecting “Create a new key pair” and click on the “Download Key Pair” button, as depicted in Figure 27.

7. Click on the “Launch Instances” button.

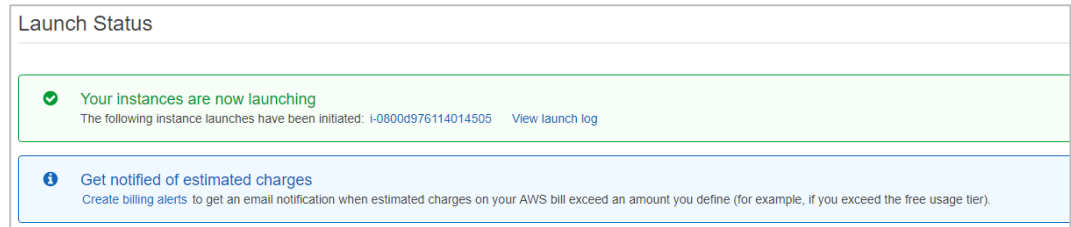


Figure 28. EC2 Instance Launch Status

Figure 28 shows the status of the launch instance in green color. Additionally, launch status provides an option to view the launch logs.

Clicking on the instance status brings the EC2 dashboard and shows any available instances (see Figure 29).

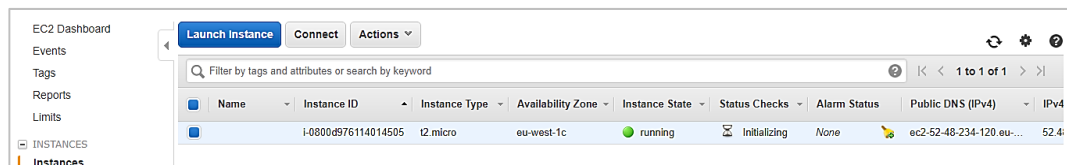


Figure 29. EC2 Dashboard Instance Information

Figure 29 illustrates the Instance ID, information about the availability zone, instance type (T2 Micro), health checks and the public DNS name generated automatically.

At this stage, the EC2 virtual server has been created and is ready for the configuration. For more information and details about the Amazon EC2 instance, Amazon product documentation for EC2 provides a good source for further reading [77].

Connecting to EC2 Instance

To perform actions such as installing updates, packages, and web server configuration; one needs to connect to EC2 instance from the client device. Depending upon the operating system of the management node or the client device the built-in SSH client can be used on Linux Unix systems. However, on Microsoft Windows Operating Systems, a third party SSH client called “putty” was required to perform the SSH operations [78]. Another

option was to install and enable Java in web browser [79], [80] and connect to EC2 instance using a web browser such as Microsoft Internet Explorer [81].

In this study a web browser (Microsoft's Internet Explorer) was used to connect to EC2 instances. Java was installed on the management node running on Windows 10 operating system (described in Chapter 4, section 4.7).

Following steps were required in order to connect to the newly created virtual machine and start the actual configuration.

1. Select Instance node from the EC2 dashboard.
2. Select the desired instance and click on "Connect".
3. Two options were presented, first was the "standalone SSH client" and second was the "Java SSH Client directly from browser" (which requires Java), as shown in Figure 30.

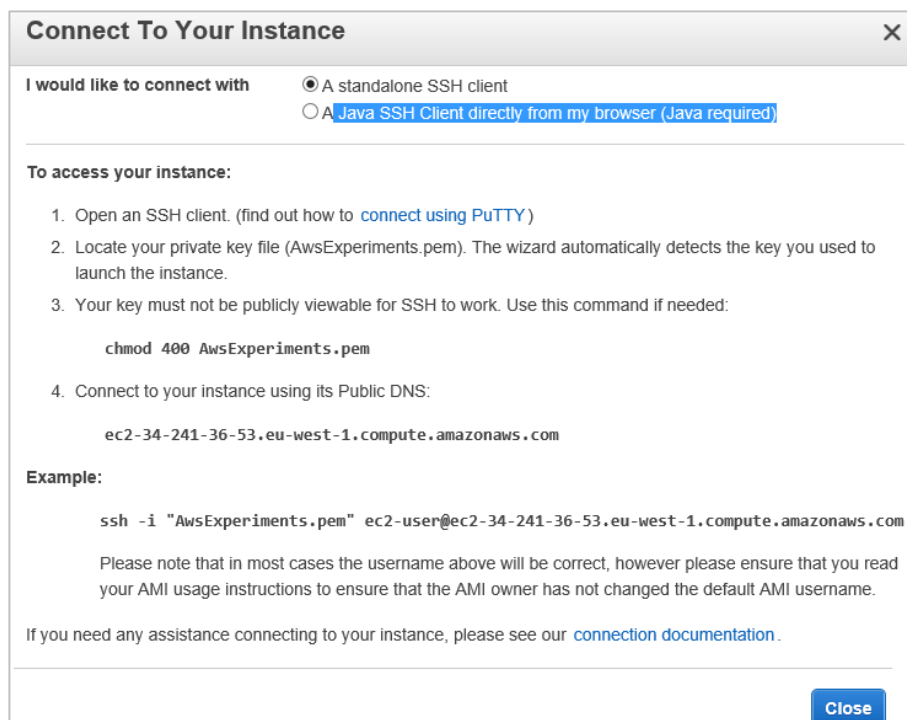


Figure 30. SSH Client Selection

Figure 30 shows the available SSH options to connect to the virtual machine (EC2 Instance).

4. Select Java SSH client from a web browser and provide a path of the key pair and select Launch SSH Client. Figure 31 shows this process.

Connect To Your Instance [X]

I would like to connect with A standalone SSH client
 A Java SSH Client directly from my browser (Java required)

Enter the required information in the fields below to connect to your instance. AWS automatically detects the key pair name, and Public DNS for your instance. You need to enter the location and name of the .pem file containing your private key.

Public DNS ec2-34-241-36-53.eu-west-1.compute.amazonaws.com
User name ec2-user
Key name AwsExperiments.pem
Private key path C:\AmazonAWS_Experiments\
Save key location Store in browser cache

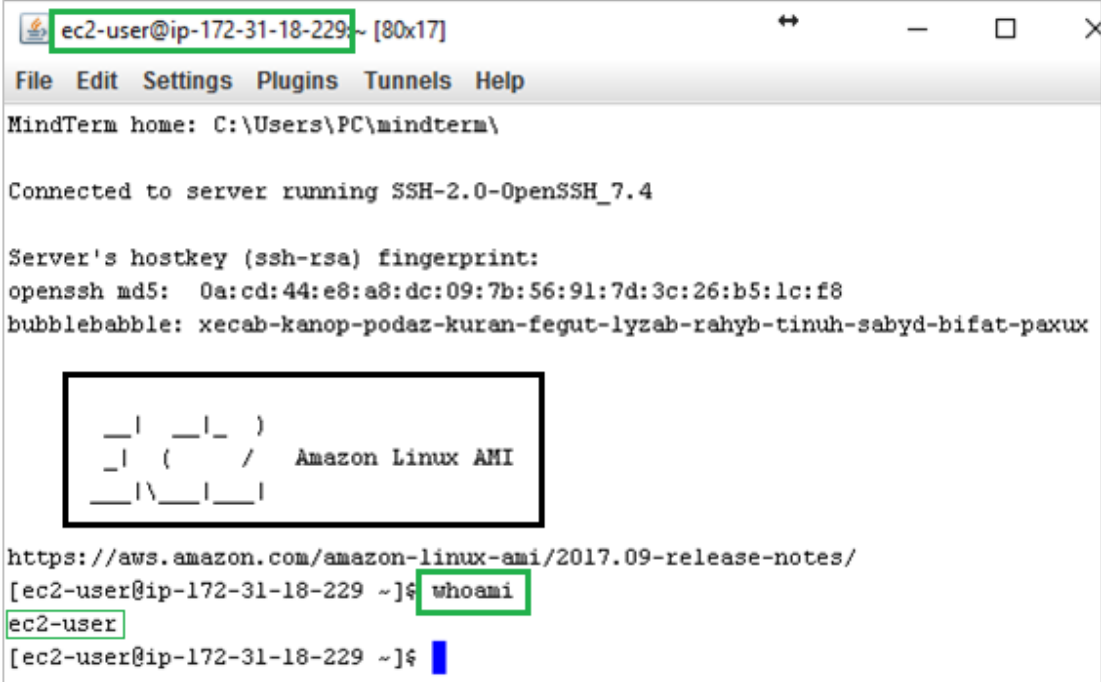
Launch SSH Client

Close

Figure 31. Connecting to EC2 Instance directly from web browser

As depicted in Figure 31, after selecting target instance, connecting to an EC2 instance requires a username (ec2-user), key name (AwsExperiments.pem), and private key path (C: \AmazonAWS_Experiments\AwsExperiments.pem). Optionally, the path to the key location can be saved in browser cache to allow the web browser to remember this path for future connections.

5. If the key name and key location are valid, the connection should be established between the client device and a virtual machine running on a cloud platform. After successful connection, ec2-user (the default username) should be logged in, as shown in Figure 32.



```

ec2-user@ip-172-31-18-229 ~ [80x17]
File Edit Settings Plugins Tunnels Help
MindTerm home: C:\Users\PC\mindterm\

Connected to server running SSH-2.0-OpenSSH_7.4

Server's hostkey (ssh-rsa) fingerprint:
openssh md5: 0a:cd:44:e8:a8:dc:09:7b:56:91:7d:3c:26:b5:1c:f8
bubblebabble: xecab-kanop-podaz-kuran-fegut-lyzab-rahyb-tinuh-sabyd-bifat-paxux

  _|  _|_ )
  _| (    /  Amazon Linux AMI
  __|\__|__|

https://aws.amazon.com/amazon-linux-ami/2017.09-release-notes/
[ec2-user@ip-172-31-18-229 ~]$ whoami
ec2-user
[ec2-user@ip-172-31-18-229 ~]$

```

Figure 32. Linux AMI Login with Ec2-User

By default, the Linux AMI (Amazon Machine Image) sign-in with `ec2-user`. This can be confirmed by with Linux terminal command “`whoami`”, which should return the currently logged-in user, as illustrated in Figure 32.

5.4 Install and Configure Apache Web Server (LAMP Stack)

Once the connection to the EC2 instance has been established using a web browser (Internet Explorer), Apache web server configuration can be started from the command line. PHP, MySQL and other required components for the web server is called LAMP stack when running on Linux operating system. The procedure mentioned in this implementation was derived from the AWS documentation [78] and based on the Amazon Linux instance. Apart from the installation itself, it was also required for the security group to allow access to certain ports such as SSH (port 22), HTTP (port 80), and HTTPS (port 443). Following Table 4 summaries the command lines with a brief description required to install LAMP stack on Amazon Linux virtual machine [78].

Table 4. Installing LAMP Stack (Terminal Commands)

No.	Linux Terminal Commands with description (LAMP Installation)
1	<p><i>sudo yum update -y</i></p> <p>This command checks and install updates (if any).</p>
2	<p><i>sudo yum install -y httpd24 php70 mysql56-server php70-mysqld</i></p> <p>This command installs multiple software packages and all related dependencies at the same time.</p>
3	<p><i>sudo service httpd start</i></p> <p>This command starts the Apache web server service.</p>
4	<p><i>sudo chkconfig httpd on</i></p> <p>Some services do not start automatically when system boots or restarts, above command, ensure that Apache web server service initializes at the system startup.</p>
5	<p><i>chkconfig --list httpd</i></p> <p>This command verifies if the service was actually running or not.</p>
<p>Note: At this stage, the installation of the Apache web server can be tested by typing the public DNS name or the IP address in a web browser. Resulting page should be Apache default home page because no contents have replicated yet in the /var/www/html. The Amazon Linux Apache document root is /var/www/html; which, by default is owned by the root. This can be viewed by typing: <code>ls -l /var/www</code>.</p>	
6	<p><i>sudo usermod -a -G apache ec2-user</i></p> <p>This adds the user account ec2-user to the apache group.</p>
7	<p><i>exit</i></p> <p>You have to log out or exit to pick up the new group specified earlier.</p>
8	<p><i>groups</i></p> <p>This command verifies if the apache group membership and output should look like: <code>ec2-user wheel apache</code></p>
9	<p><i>sudo chown -R ec2-user:apache /var/www</i></p> <p>Change the group ownership of /var/www and contents to the apache group.</p>
10	<p><i>sudo chmod 2775 /var/www</i></p> <p><i>find /var/www -type d -exec sudo chmod 2775 {} \;</i></p> <p>Add group write permissions, change the directory permissions of /var/www and its sub-directories.</p>
11	<p><i>find /var/www -type f -exec sudo chmod 0664 {} \;</i></p> <p>This command was used to add group writes permissions, recursively change the file permissions of /var/www and its subdirectories.</p>
12	<p><i>sudo service mysqld start</i></p> <p>Like Apache web server, MySQL service can start with above example.</p>
13	<p><i>sudo mysql_secure_installation</i></p>

	Here we can secure the MySQL installation, when prompted, type the desired password to set for the MySQL installation.
14	<i>sudo chkconfig mysqld on</i>
	This command ensures that MySQL service should start on every boot.

At this stage, Apache web server has been installed together with MySQL database and PHP scripting language. The above command lines also include required configuration for the ec2-user account.

5.5 Install and Configure WordPress Application with Amazon Linux

At this point, the prerequisites for the web application have been installed (LAMP stack) and configured the security group have been configured to allow traffic on port 80 and 443 (HTTP and HTTPS) and the SSH access have been enabled on port 22. Now it's time to install and configure the web application, WordPress; an online content management system [37]. Table 5 summaries the command-lines that were used to install the WordPress application in this implementation. These command lines were extracted from the AWS tutorial that describes WordPress blog hosting on Amazon Linux instance [79].

Table 5. Installing WordPress Application (Terminal Commands)

No.	WordPress Application Installation (Terminal Commands)
1	<i>wget https://wordpress.org/latest.tar.gz</i> This command was used to download the WordPress.
2	<i>tar -xzf latest.tar.gz</i> This command extracts and unzips the installation package, called WordPress.
3	<i>sudo service mysqld start</i> This command was used to start the MySQL server.
4	<i>mysql -u root -p</i> Log in to the MySQL server as the root user. Enter your MySQL root password when prompted. This must be the same password as specified during MySQL installation.
5	<i>CREATE USER 'wordpress-user'@'localhost' IDENTIFIED BY 'MyP@ssW0rd';</i> Above command should create a user and password for your MySQL database.
6	<i>CREATE DATABASE `wordpress-db`;</i> wordpress-db was the name of the database in above example.
7	<i>GRANT ALL PRIVILEGES ON `wordpress-db`.* TO "wordpress-user"@"localhost";</i> Grant full privileges for the database to the WordPress user that was created earlier.
8	<i>FLUSH PRIVILEGES;</i> Flush the MySQL privileges to pick up all of your changes.

9	Exit Exit the MySQL client.
10	cd wordpress/ cp wp-config-sample.php wp-config.php Copy the wp-config-sample.php file to a file called wp-config.php.
11	nano wp-config.php define ('DB_NAME', 'wordpress-db'); define ('DB_USER', 'wordpress-user'); define ('DB_PASSWORD', 'YourPassword'); cd .. In the editor, find the line that defines DB_NAME, DB_USER, DB_PASSWORD, once done save the file and exit the text editor. Change the directory with cd .. to reach to the root listing.
12	cp -r wordpress/* /var/www/html/ Copy the contents of the wordpress installation directory (but not the directory itself) if the intention was to run wordpress to at your document root.
13	sudo nano /etc/httpd/conf/httpd.conf In text editor, find the section that starts with <Directory "/var/www/html">. Find the text >>>AllowOverride None and change with AllowOverride All WordPress permalinks need to use Apache .htaccess files to work properly, but this was not enabled by default on Amazon Linux. Use above procedure to allow all #overrides in the Apache document root.
14	sudo chown -R apache /var/www Change the file ownership of /var/www and its contents to the apache user.
15	sudo chgrp -R apache /var/www Change the group ownership of /var/www and its contents to the apache group.
16	sudo chmod 2775 /var/www find /var/www -type d -exec sudo chmod 2775 {} \; Change the directory permissions of /var/www and its subdirectories to add group write permissions and to set the group ID on future subdirectories.
17	find /var/www -type f -exec sudo chmod 0664 {} \; Recursively change the file permissions of /var/www and its subdirectories to add group writes permissions.
18	sudo service httpd restart Restart the Apache web server to pick up the new group and permissions.
19	sudo chkconfig httpd on sudo chkconfig mysqld on Use the chkconfig command to ensure that the httpd and mysqld services start at every system boot.
20	<i>Enter the public DNS name of your instance in a web browser. WordPress installation screen will pop up, with options to specify site title, user name, and password.</i>

Now the WordPress web application has been installed and functional. Further adjustments and customizations e.g. change the theme, installing updates can be performed via admin control panel. Amazon documentation provides details regarding configuring the newly created blog, increase the capacity and how to change the domain name [80].

Troubleshooting DNS Name Change Problem

EC2 instances receive public DNS name by default and the WordPress installation was automatically configured using the public DNS name. When the instance has been powered off or restarted, previously assigned public DNS changes randomly. After that the public address that was originally configured was not more available and configurations still pointed to the old URL [81]. One solution is to assign an elastic IP, static IP for EC2 instance, otherwise one needs to repair the WordPress with wp-cli. Wp-cli is a command line tool from the WordPress and used to repair the WordPress installation problem due to change in the DNS name [82]. Appendix 2, *Troubleshooting DNS Name Change Problem* provides the commands required to repair the WordPress installation.

5.6 Create an Image from Linux EC2 Instance

Now the Linux virtual machine has been configured to run the web application, hosted on the cloud platform. Once all the required software has been installed and configurations have been completed, an image of this virtual machine can be captured. This procedure is known as AMI (Amazon Machine Image) on Amazon terminologies. Whenever there would be a need for additional resources, this AMI can be launched containing all required software ready to use [83]. Following steps needs to be carried out to create an image of EC2 instance [84].

1. Login to AWS management console (if not logged-in already).
2. Select EC2 services and access the EC2 dashboard.
3. Click on Instances from the side navigation tree menu.
4. Details of the available EC2 instance should appear in the main pane.
5. Select the desired instance and click on the: Actions → Image → Create Image.

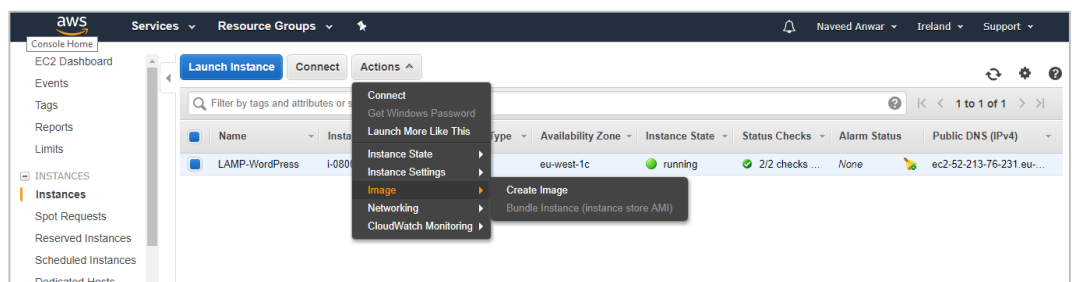


Figure 33. Creating an Amazon Machine Image

Figure 33 illustrates the options to create an image of EC2 instance.

6. Create Image dialog box appears now and looks similar to the figure given below. Give a suitable name and description and click on “Create Image”.

Create Image

Instance ID ⓘ i-0800d976114014505

Image name ⓘ MyVM-LAMP-WordPress

Image description ⓘ Apache,MySQL,PHP,WordPress

No reboot ⓘ

Instance Volumes

Volume Type ⓘ	Device ⓘ	Snapshot ⓘ	Size (GiB) ⓘ	Volume Type ⓘ	IOPS ⓘ	Throughput (MB/s) ⓘ	Delete on Termination ⓘ	Encrypted ⓘ
Root	/dev/xvda	snap-0465ac035e2dac85d	8	General Purpose SSD (GP2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Total size of EBS Volumes: 8 GiB
When you create an EBS image, an EBS snapshot will also be created for each of the above volumes.

Figure 34. Creating an Amazon Machine Image (Properties)

Figure 34, shows the properties of the Create Image wizard, here name and description of the image can be specified. Hardware customization would also possible, such as the size of the hard disk. In this implementation, the default 8 GB was selected to create the image.

7. Image creation request is acknowledged and a confirmation dialog will pop up.
8. Newly created AMI now available for use and listed under Images → AMIs. Figure 34 depicts a partial screenshot of the EC2 dashboard, showing available AMIs.

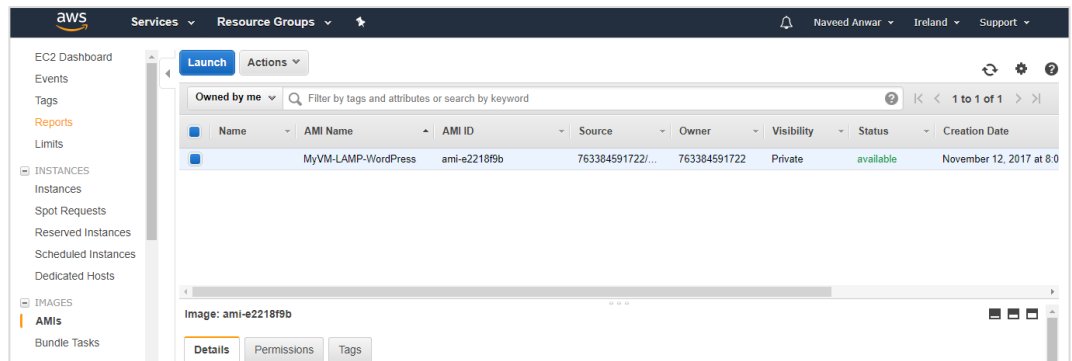


Figure 35. Available Amazon Machine Images (AMIs)

Figure 35 shows the newly created Amazon Machine Image, this is the copy of the virtual machine that was configured with the Apache web server to host WordPress web application. Date of creation and current status are available in Figure 35.

5.7 Create Auto Scaling Group and Launch Configuration

Auto-scaling is one of the distinctive features of Amazon cloud platform available as part of EC2 service. Auto-scaling ensures that specified number of EC2 instances are always available. As soon as there is a breach at software or hardware level is detected, a new instance is automatically launched and it becomes available dynamically. Auto-scaling constitutes on following two elements:

- **Launch Configuration:** A launch configuration contains the necessary information required to launch a virtual machine. A typical launch configuration includes the size of the virtual server and Amazon Machine Image (AMI) to launch.
- **Auto-Scaling Group:** An auto-scaling group is used to configure EC2 service representing the number of virtual machines to be started with a particular launch configuration. Resource monitoring and subnet where to launch the new instance is also determined by the auto-scaling group.

The following figure depicts a high-level view of how auto-scaling service to ensures that specified number(s) of the health EC2 instances are available for the service.

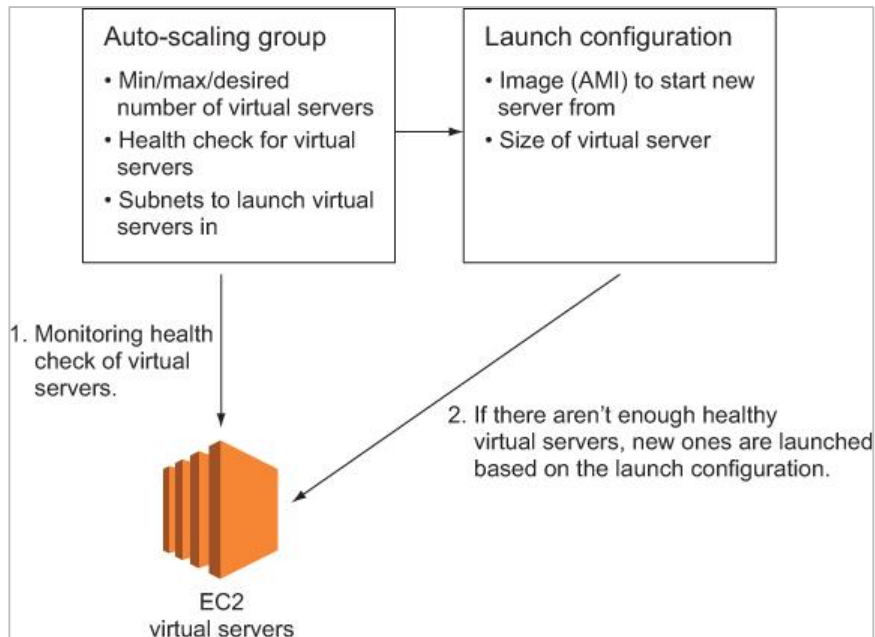


Figure 36. Amazon EC2 Auto-Scaling Mechanism [64, Fig 11.7]

Figure 36 shows the auto-scaling feature to ensure at least a single instance (at minimum) should always be available to serve the user requests. With a properly configured auto-scaling and launch configuration, EC2 auto-scaling service can provision and de-commission the EC2 instances. New instances can be launched dynamically, either with an increase in resource demand or if an existing virtual server started to malfunction.

Figure 37 shows an example scenario where a new instance is launched due to hardware failure in the currently running instance.

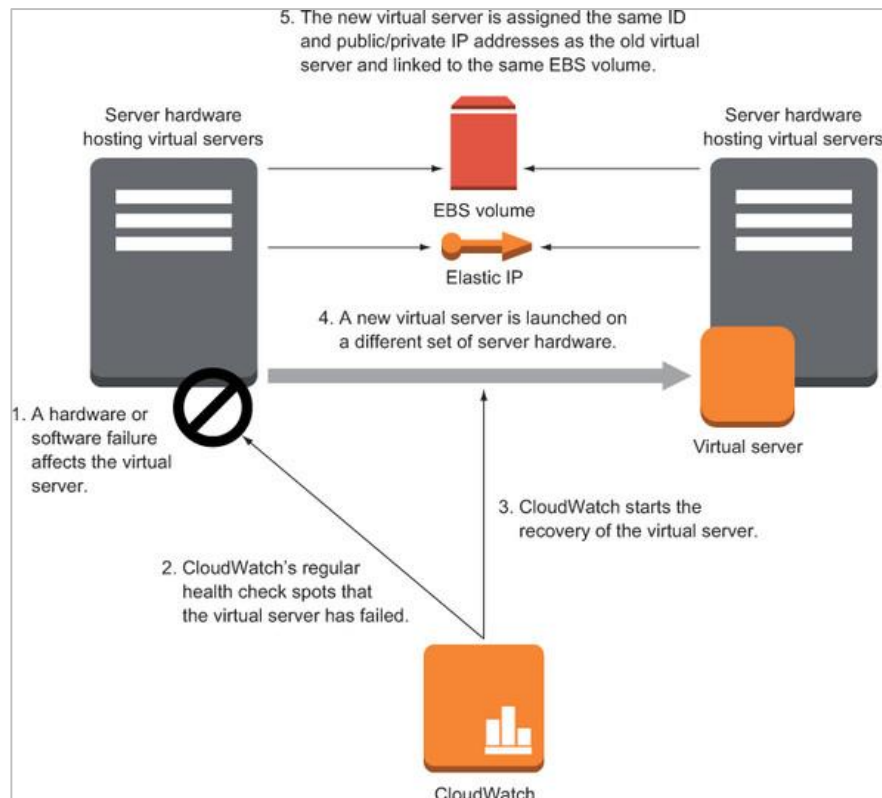


Figure 37. Automatic Recovery of EC2 Instance [64, Fig 11.2]

Figure 37 depicts the process of launching a new virtual server. This event was triggered by an alert generated by CloudWatch that records the unhealthy threshold. Depending on the scalability metrics and CloudWatch alarms, one or multiple instances of virtual machines (AMIs) can be launched with the auto-scaling service.

Following steps were required to complete this implementation of the auto-scaling.

1. Login to AWS management console (if not logged-in already).
2. Select EC2 services and access the EC2 dashboard.
3. From side navigation menu, select Auto Scaling.

The welcome screen should look something similar to the following figure.

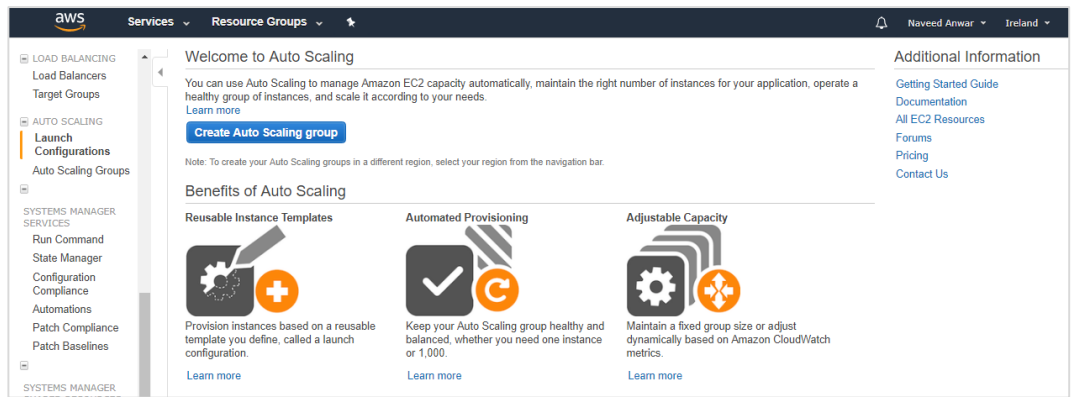


Figure 38. Welcome to Auto Scaling

Figure 38 shows the welcome screen of auto-scaling and provides the option to “Create Auto Scaling Group”.

4. Click on “Create Auto Scaling Group”. If no existing launch configuration is found, “Create Auto Scaling Group” begins with the “Create Launch Configuration as the first step.

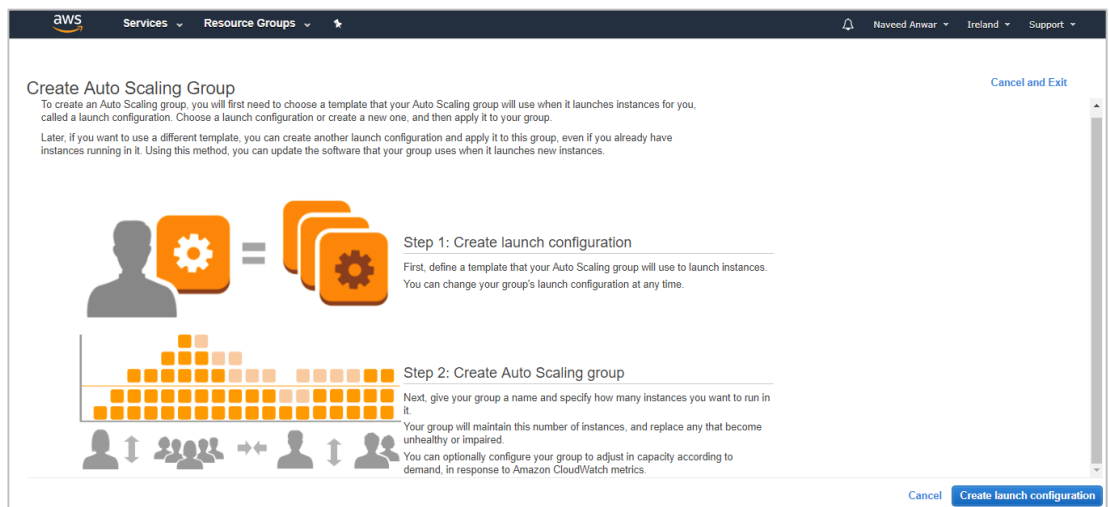


Figure 39. Create Auto Scaling Group and Launch Configuration

As illustrated in Figure 39, auto-scaling service demands a launch configuration (step 1) before an auto-scaling group can be created (step 2).

5. Click on “Create Launch Configuration” and follow on-screen instructions.

- After completing the launch configuration wizard, click on the “Create Auto Scaling Group”.

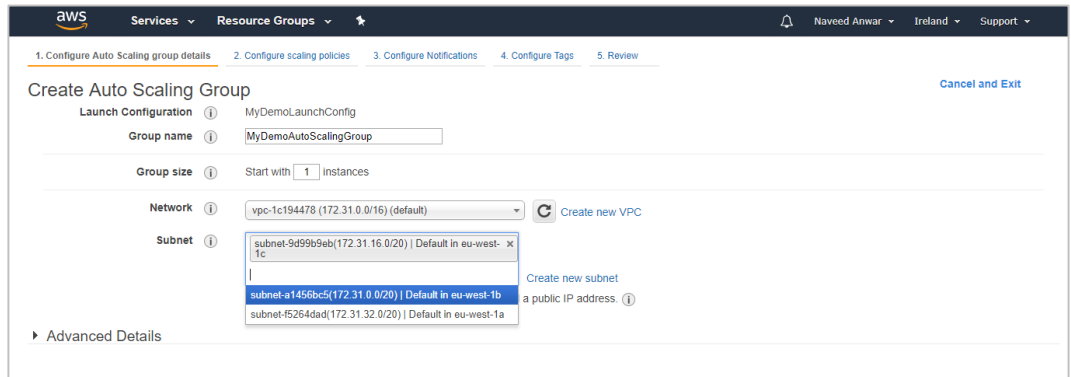


Figure 40. Configure Auto Scaling Group Details

Figure 40 illustrates the details of the auto scaling group. Select the default VPC and all available subnets, this procedure makes an application highly available across multiple availability zones, as illustrated in Figure 40.

The group size, also known as desired capacity, can be configured and it determines the number of the instance(s) that this auto scaling group should have at any time. This implementation uses two instances to start with, however, this can span to multiple instances across the availability zone to ensure high availability. The concept of availability zone and region can be illustrated with the following figure.

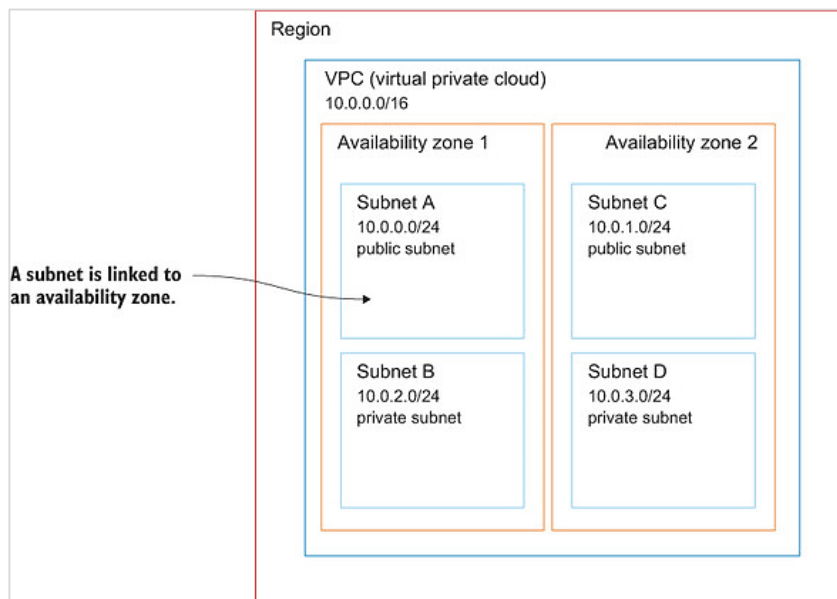


Figure 41. VPC, Region, Availability Zone [64, Fig 11.6]

As illustrated in Figure 41, a VPC (Virtual Private Cloud) is bound to a region. Subnets are linked to availability zones which should always be within a VPC. Configuring multiple subnets in distinguished availability zones is the recommended approach for highly available and scalable applications.

7. Next click on the “Configure Scaling Policies”. There should be two options available:

(i) Keep this group at its initial size.

(ii) Use scaling policies to adjust the capacity of this group.

Select “Use scaling policies to adjust the capacity of this group” and define the minimum and maximum size of the group.

This implementation consists of 3 (minimum) and 9 (maximum), however, this can be changed according to business needs and current demands.

8. Follow the on-screen instructions and complete all the steps until review screen appears.

9. Review all the configurations and proceed further to create the auto scaling group.

5.8 Create Elastic Load Balancer

After configuring the auto scaling launch configuration and auto-scaling group, next step is to create an elastic load balancer. Amazon defines the elastic load balancer as:

A load balancer accepts incoming traffic from clients and routes requests to its registered targets (such as EC2 instances) in one or more Availability Zones. The load balancer also monitors the health of its registered targets and ensures that it routes traffic only to healthy targets. When the load balancer detects an unhealthy target, it stops routing traffic to that target and then resumes routing traffic to that target when it detects that the target is healthy again [85].

There are three types of elastic load balancers supported by Amazon Web Services. These three types are:

- Application Load Balancer.
- Network Load Balancer.
- Classic Load Balancer.

This implementation uses the classic load balancer provided by AWS. J. Nadon defines load balancer and highlights the difference between the elastic load balancer and classic load balancer as:

Elastic Load Balancers are endpoint devices that handle web traffic and can balance that traffic between multiple EC2 virtual server instances. In 2016, AWS introduced a new type of load balancer with enhanced features and benefits called an Application Load Balancer. At this point, AWS started referring to the Elastic Load Balancer offering as a Classic Load Balancer [86].

Following steps were required to create a classic load balancer and register this load balancer to the auto scaling group.

1. Login to AWS management console (if not logged-in already).
 2. Select EC2 services and access the EC2 dashboard.
 3. From side navigation menu, select Load Balancing → Load Balancer
- This should look similar to the Figure 42, given below:

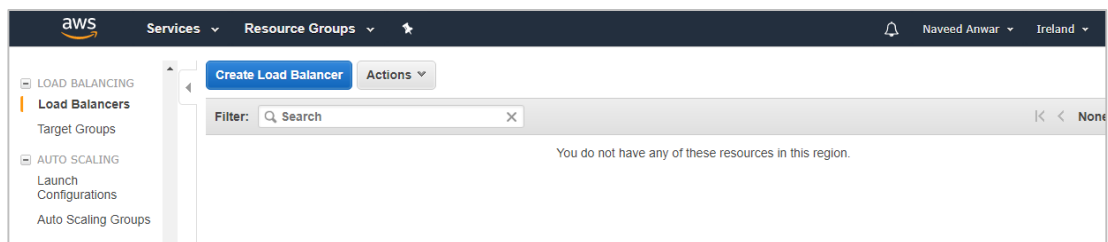


Figure 42. AWS Load Balancing

4. Click “Create Load Balancer”, and resulting screen look similar to the Figure 43, depicted below.

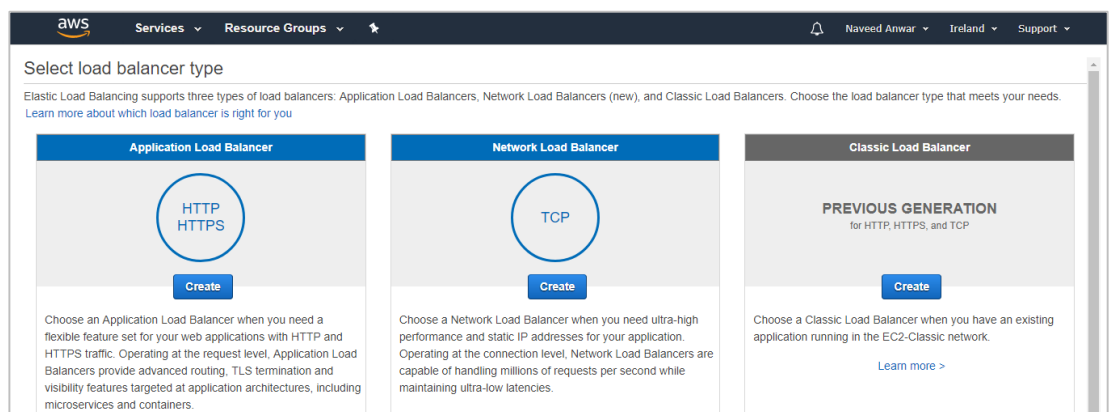


Figure 43. Load Balancer Types

Figure 43 illustrates the types of a load balancer that Amazon cloud platform provides. This implementation uses the classic load balancer.

5. Follow the on-screen instruction to complete the creation of a load balancer.
6. Define Load Balancer
 - Give Load Balancer a suitable name.
 - Select the VPC group under “Create LB Inside”, this should allow access to port 80 from everywhere.
 - Select “Enable advanced VPC configuration” and select the available subnets.
7. Assign Security Groups

A security group can be a default group or one created by the user with necessary configurations, but it should be the same throughout the implementation. This example uses a security group that allows access to port 80 from everywhere and port 22 only to Administrator’s IP.
8. Configure Security Settings

No changes are required, accept the default settings and proceed further.
9. Configure Health Check

This is one of the very critical stage and it requires ping protocol and port and path for the application under consideration. This implementation uses HTTP as a ping protocol, port 80 and ping path “/” which specifies the root of the web application as shown in Figure 44.

The screenshot displays the AWS Management Console interface for configuring a health check. At the top, there is a navigation bar with the AWS logo, 'Services', 'Resource Groups', and user information. Below this is a progress bar with seven steps: 1. Define Load Balancer, 2. Assign Security Groups, 3. Configure Security Settings, 4. Configure Health Check (highlighted), 5. Add EC2 Instances, 6. Add Tags, and 7. Review.

The main content area is titled 'Step 4: Configure Health Check'. It includes a descriptive paragraph: 'Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to instances that pass the health check. If an instance fails the health check, it is automatically removed from the load balancer. Customize the health check to meet your specific needs.'

The configuration fields are as follows:

- Ping Protocol:** A dropdown menu set to 'HTTP'.
- Ping Port:** A text input field containing '80'.
- Ping Path:** A text input field containing '/'.

Below these fields is the 'Advanced Details' section, which contains four rows of configuration:

- Response Timeout:** A text input field with '2' and the unit 'seconds'.
- Interval:** A text input field with '5' and the unit 'seconds'.
- Unhealthy threshold:** A dropdown menu set to '2'.
- Healthy threshold:** A dropdown menu set to '4'.

At the bottom right of the page, there are three buttons: 'Cancel', 'Previous', and 'Next: Add EC2 Instances'.

Figure 44. Configure Health Check

Figure 44 illustrates the options required to configure health checks. Following advanced options are required to be configured according to the application needs and are listed below:

- **Response Timeout** = Time to wait when receiving a response from the health check.
- **Interval** = Amount of time between health checks (5 secs - 300 sec).
- **Unhealthy Threshold** = Number of consecutive health check failures before declaring an EC2 instance unhealthy.
- **Healthy Threshold** = Number of consecutive health check successes before declaring an EC2 instance healthy.

10. Add EC2 Instances

At this stage, the instances that would be part of load balancer can be selected, but this can be done later on as well. For this implementation, the EC2 instance was added after configuring the auto scaling policies. Following two properties of the “Availability Zone Distribution” needs some introduction and described as:

- **Enable Cross-Zone Load Balancing** = Cross-Zone Load Balancing distributes traffic evenly across all the back-end instances in all Availability Zones.
- **Enable Connection Draining** = The number of seconds to allow existing traffic to continue flowing for the X number of seconds provided by administrator. When **Connection Draining** is enabled, Auto Scaling will wait for outstanding requests to complete before terminating instances [87].

11. Add Tag

An optional step, if some special tags need to be associated with the EC2 instances.

12. Review

Summary of load balancer is now presented for review. Click on “Create” if everything is correctly set, otherwise click on “previous” button to make any adjustment.

13. Associate Load Balancer to Auto Scaling Group

Auto Scaling Group → Details → Edit

14. Select this newly created Load Balancer and save settings, as shown in Figure 45, given below.

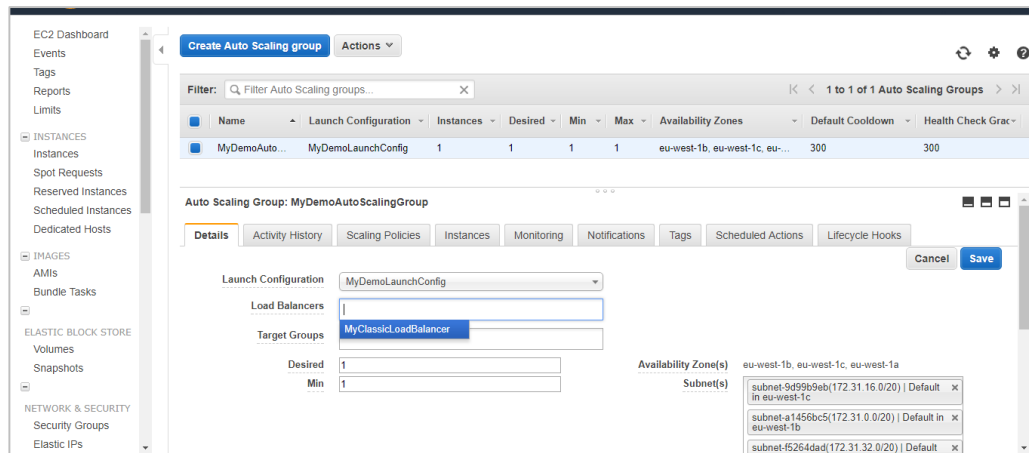


Figure 45. Associate Load Balancer to Auto Scaling Group

Figure 45 describes how to associate a load balancer with the auto scaling group. Click on “Save” to actually commit the required modifications.

5.9 Performance Measurement Tool (JMeter)

Amazon Cloud Platform was used for the actual implementation and local environment was used to conduct the experiments. Apache JMeter was installed on the local machine (on premise), known as a management node or a technician machine as described in section 4.7. JMeter is a Java-based tool for performance measurement and benchmarking of the web servers. A JMeter test plan consists of a number of elements that constitute a test plan in JMeter. Figure 46 describes the anatomy of a JMeter test [59, Anatomy of a JMeter test].

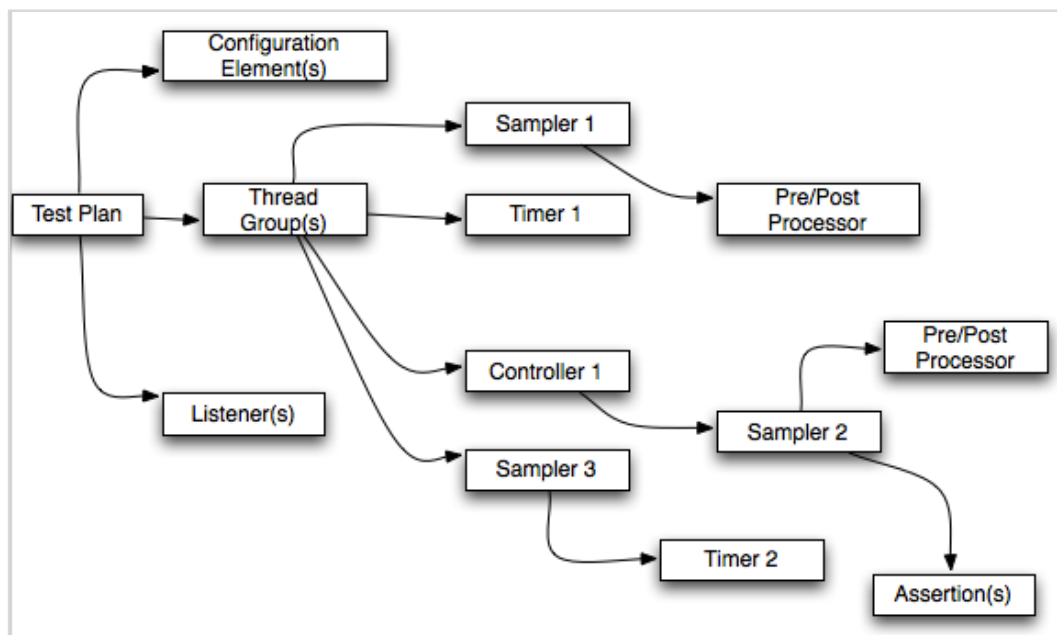


Figure 46. The Anatomy of a JMeter Test [59].

As depicted in Figure 46, there are several components that help to generate data to a test plan (thread group, timer, assertions) and these built-in components helped to extract and save responses (listeners). A test plan is the core element of a JMeter script that acts as a placeholder for the other vital components such as threads, configuration elements, timers, pre-processors, postprocessors, assertions, and listeners.

To generate heavy workloads, thread groups are used which represents the number of users (threads in JMeter terminology) that JMeter uses for the execution of a test plan and how long it will take to initialize each user request (Ramp-Up Period). Regardless of the number of threads (users), each thread executes a test plan independently from other threads. Samplers are built-in components that send requests to the server and record the response. JMeter is equipped with following samplers:

- HTTP request.
- JDBC request.
- LDAP request.
- SOAP/XML-RPC request.
- Web service (SOAP) request.
- FTP request.

Listeners are very useful because they help to gather results from a running test and can save these results in a CSV or XML file for further analysis. There is no restriction in the order of precedence for the listeners. They can be added anywhere in a test plan but important to know is that they will collect data only from those elements that are below their level.

5.10 Response Time Assertion

Assertions are one of the most significant components that are frequently used to verify the expected performance of the application by calculating the response time received from the target server. For instance, duration assertion tests if the response was received within a specified time frame (milliseconds). Results of response assertion can either be “success” (if the response is within the time frame) or “fail” (if the response is not within the time frame). For example, take an example of 300 milliseconds as a test case criteria and send requests to JMeter. If response time is within 3 seconds, the test is passed otherwise it will return fail. Figure 47 depicts this concept.

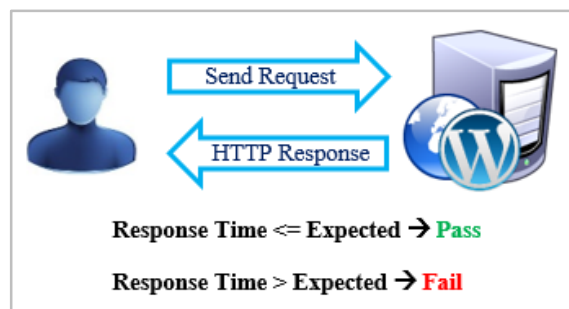


Figure 47. Response Time Assertion

Figure 47 illustrates the concept of the duration assertion, also known as response time assertion. Exploring all JMeter elements and test plan components in detail is out of the scope of this project. For more information on JMeter elements and test composition, [59] provides useful information on this topic and official product documentation is also a good source of information [88].

5.11 Creating Performance Test Plan in JMeter

To compare the performance of the proposed system with the legacy setup, several experiments were conducted. As stated before, JMeter was used to benchmark the web application. JMeter tests were used to generate heavy load by simulation concurrent user requests. Figure 48 shows the setup of heavy load simulation with JMeter.

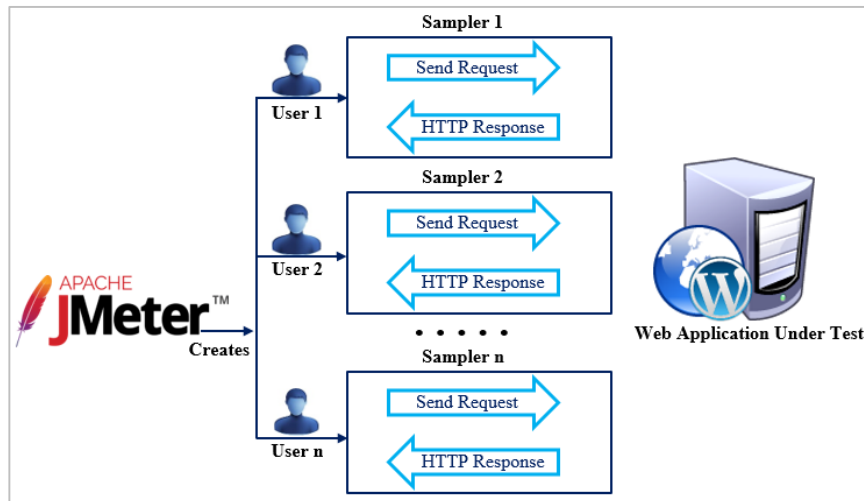


Figure 48. High Load Simulation with JMeter

As depicted in Figure 48, JMeter simulates user requests and tracks the HTTP responses for the WordPress web allocation.

Following steps were required to create a performance test plan in JMeter.

Add Thread Group

1. Start JMeter in GUI Mode
2. Select Test Plan
3. Add Thread Group: Right click on the Test Plan → Add → Thread (Users) → Thread Group, as shown in below figure.

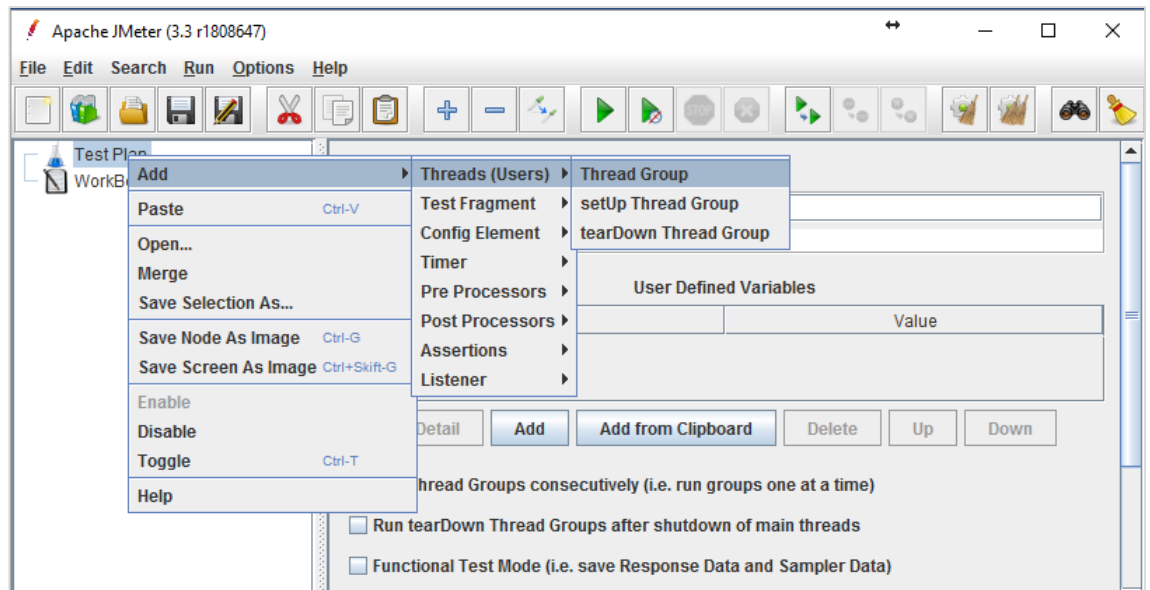


Figure 49. Add Thread Group

Figure 49 illustrates the graphical representation of the JMeter Thread Group.

4. Enter the Thread Properties according to desired testing criteria, as shown in the following figure.

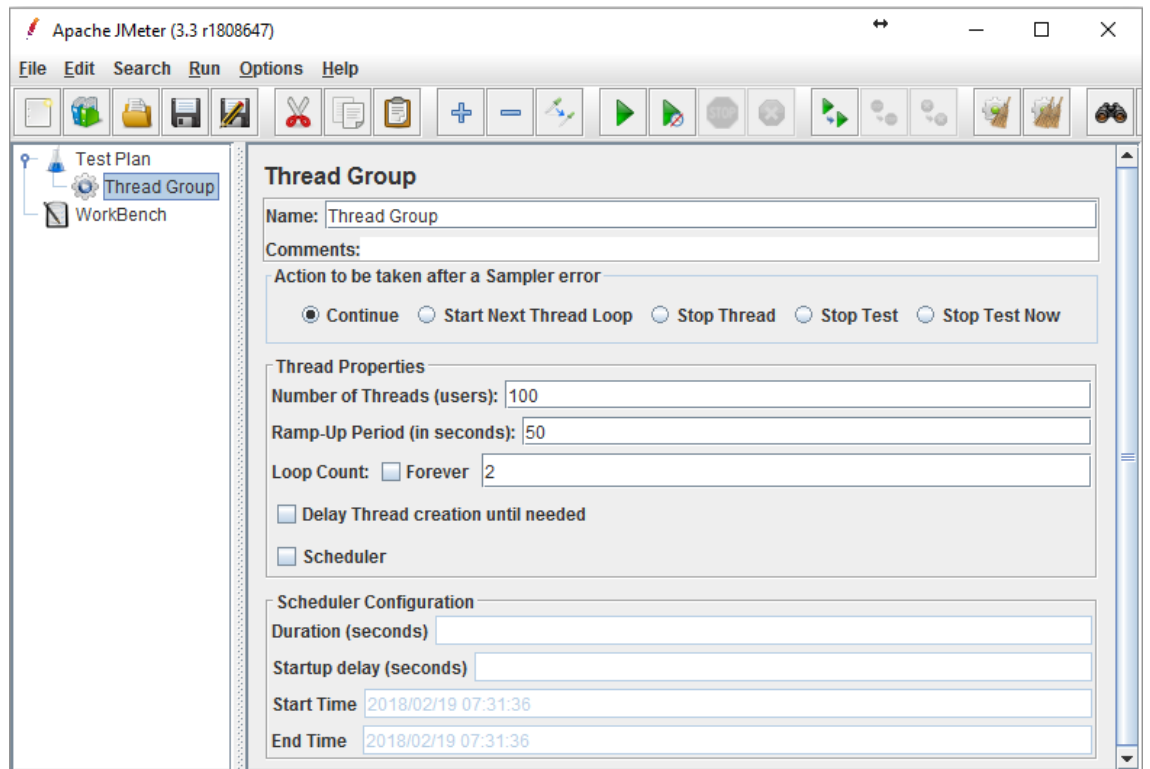


Figure 50. Thread Group Properties

A number of Threads (users), Ramp-Up Period and Loop count are the properties of Thread (which represents users), as shown in Figure 50 and described as:

- Number of Threads: 100 (*Number of users connects to target website: 100*)
- Ramp-Up Period: 50 (*delay before starting new user*)
- Loop Count: 2 (*Number of time to execute testing*)

Adding JMeter Elements

1. HTTP Request Default

Right-click on the Thread Group and selecting:

Add → Config Element → HTTP Request Defaults.

2. In the HTTP Request Defaults control panel, enter the Website name under test.

3. HTTP Request

Right-click on Thread Group and select:

Add → Sampler → HTTP Request.

4. Add Graph Results

Right-click Test Plan, Add → Listener → Graph Results

Right-click Test Plan, Add → Listener → View Results in Table

Right-click Test Plan, Add → Listener → View Results in Tree

Right-click Test Plan, Add → Listener → Response Time Graph

Right-click Test Plan, Add → Listener → Summary Report

Add Duration Assertion

1. Right-clicking on the Thread Group and select:
Add → Assertion → Duration Assertion
2. This should bring up Duration Assertion details pane.
3. Enter the time in milliseconds in the Duration Assertion field.

Execute Test

JMeter test can be executed by clicking on the Play button or keyboard shortcut key combination (Ctrl + R), as depicted in the Figure 52 below.

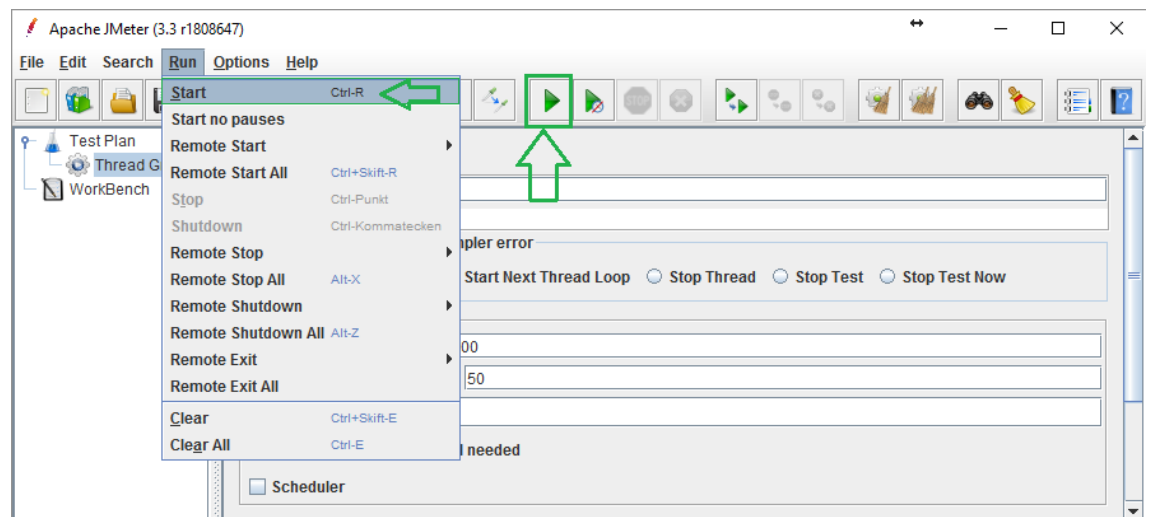


Figure 51. Executing JMeter Test

Figure 52 shows the available options to execute a JMeter test from the graphical user interface (GUI).

Test results started to populate the result area providing an opportunity to view the results in real time. However, due to huge resource consumption with the graphical user interface, a recommended approach to execute the test is to use command line mode of the JMeter [59].

5.12 JMeter Concurrency Level and Best Practices

A thread group controls the threads (users) that will be created by JMeter to simulate simultaneous users. Each thread represents one user using the application under test. If there are two or more Thread Groups in the same Test Plan, each Thread Group will execute completely independently from each other. Multiple Thread Groups within the same Test Plan simulate groups of concurrent, individual connections to the target web server [90].

According to the JMeter user manual and best practices, the hardware capabilities and Test Plan design will both impact the number of threads that can run effectively with JMeter.

JMeter has an option to delay thread creation until the thread starts sampling, i.e. after any thread group delay and the ramp-up time for the thread itself. This allows for a very large total number of threads, provided that not too many are active concurrently [91].

Ramp-Up Period defines how long it will take JMeter to get all threads running. For example, if there are 100 threads and a ramp-up period of 50 seconds, then each successive thread will be delayed by 0.5 second ($50 / 100$). In 50 seconds, all threads would be up and running. If Ramp-Up Period is configured to 0, JMeter will start running immediately without any delays. If there are more than one user, having Ramp-Up Period to 0 will cause all users to start running immediately and simultaneously. Starting all threads at once may cause the application to crash due to instance and rapid load. This is why, Ramp-Up Period of 0 is not considered a good approach specially when testing with large number of users (threads). Ramp-Up Period also should not be too high as it makes the throughput indicator void. The best policy is to make the Ramp-Up Period long enough to avoid large workload as the test begins, but short enough to allow the last one to start running before finishing the first one. Also, for the larger workloads, if Ramp-Up Period is 0, the web application may consider the JMeter requests as denial-of-service attack (DoS attack) because all users are started at once. DoS attack is a

process to make a server overloaded by sending tremendous amount of user requests with intentions to make a network resource unavailable to its intended users. For testing purposes, Ramp-Up Period can be configured same as the number of Threads, this will result in 1 second delay ($\text{Ramp-Up Period} / \text{Threads}$) before facilitating next thread and can be adjusted according to the specific needs [90].

Throughput is one of the important performance indicator when evaluating the application performance. It signifies number of transactions or requests that can be made in a given period of time. It is a useful measurement to check the load capacity of the server. Though one should not purely depend on the throughput metrics and it needs to viewed in conjunction with latency and response times [92].

6 Results and Analysis

This chapter describes what was found using the performance evaluation tool (JMeter) as described in the Implementation chapter, section 5.11. Several experiments were conducted to collect the desired data on both the legacy system and the (proposed) highly scalable system. Following sections describe experiment workflows, analyzed data and compiled results as well as the description of the experimental environment.

6.1 Experiment Environment

Amazon Cloud Platform was used for the actual implementation (chapter 5) and local environment was used to conduct the tests by simulating workloads. This local environment is represented by the management node also known as technician machine (described in section 4.7) and was used to conduct the experiments with JMeter (described in 5.9). The same management node was also used to create the performance test plans in JMeter (section 5.11) and save the experiment results in CSV file format. Following figure depicts the experimental environment of the legacy system.

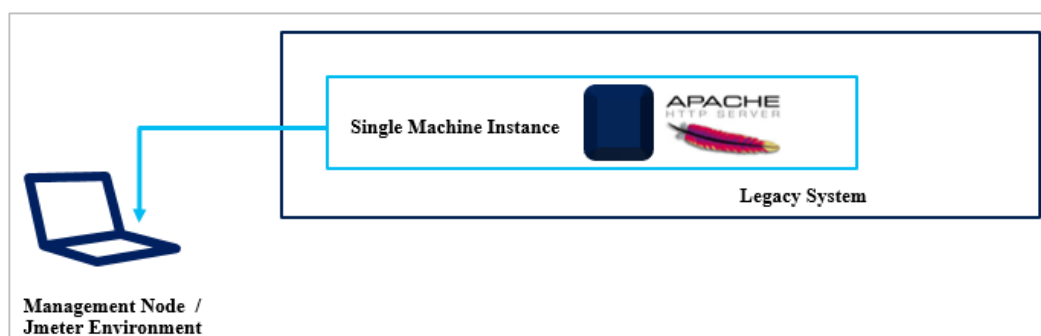


Figure 52. Experiment Environment of the Legacy System

As shown in Figure 52, JMeter is installed and configured on the management node that was used to perform the experiments to test the performance of the legacy system by applying different workloads. The legacy system comprises on a single machine and running Apache Web Server to host WordPress web application.

Next stage was to conduct the same experiments with the scalable cloud platform. Figure 53 illustrates the experimental environment for the scalable cloud platform.

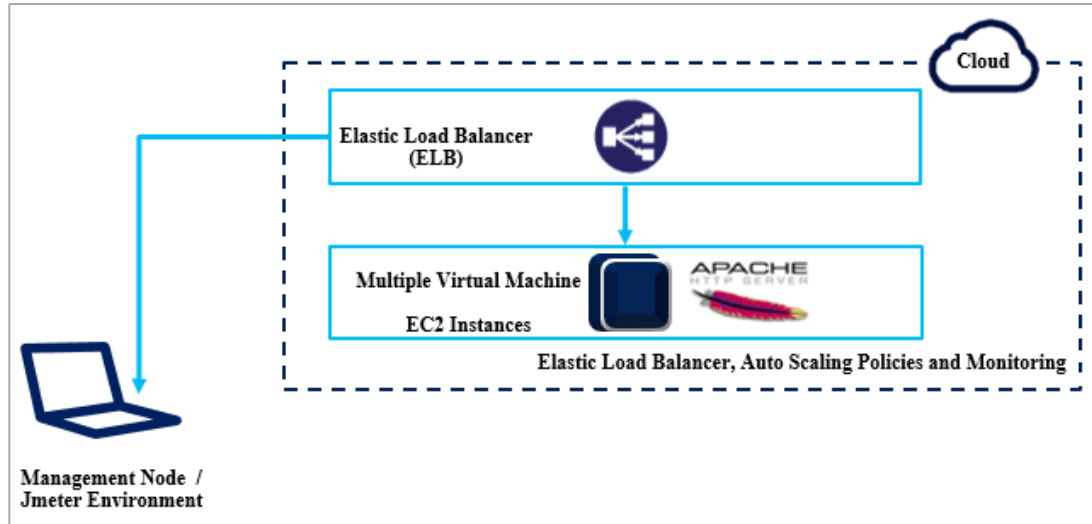


Figure 53. Experiment Environment of the Scalable Cloud Platform

As depicted in Figure 53, the web application is running on cloud platform and elastic cloud load balancer, auto scaling policies and monitoring services are configured for the multiple virtual machine instances. As with the legacy system, the management node was used to perform the experiments as well as save the results. Details of the applied workflow is available in the following section.

6.2 Experiment Workload and Data Collection

To evaluate the performance of the proposed system, different experiments were conducted by simulating heavy workloads against the web server. The same workloads were applied to the existing (legacy) system that was used to identify the scalability problems in the legacy system (described in chapter 2). Table 6 provides a summary of the experiment workloads.

Table 6. Summary of the Experiment Workloads.

Summary of the Experiment Workloads						
Total Simulated Users	25	50	100	500	1,000	2,000
Number of Threads (Users)	25	50	100	500	1,000	2,000
Ramp-Up Period (Seconds)	0	0	0	0	100	500
Loop Count	1	1	1	1	1	1
Duration Assertion (Milliseconds)	3000	3000	3000	3000	3000	3000

Table 6 presents the experiment workloads that were applied to the legacy system (and the proposed (scalable) system). Following elements constitute the experiment workloads:

Number of Threads = The number of concurrent users accessing the web application.

Loop Count = Number of time to execute the test.

Duration Assertion = Represents time in milliseconds to set the criteria if the response was received within the acceptable time frame (as described in section 5.10, Response Time Assertion).

Ramp-Up Period = Ramp-Up Period represents the time in seconds that tell JMeter to delay before facilitating next user. The following figure represents the idea of the ramp-up time.

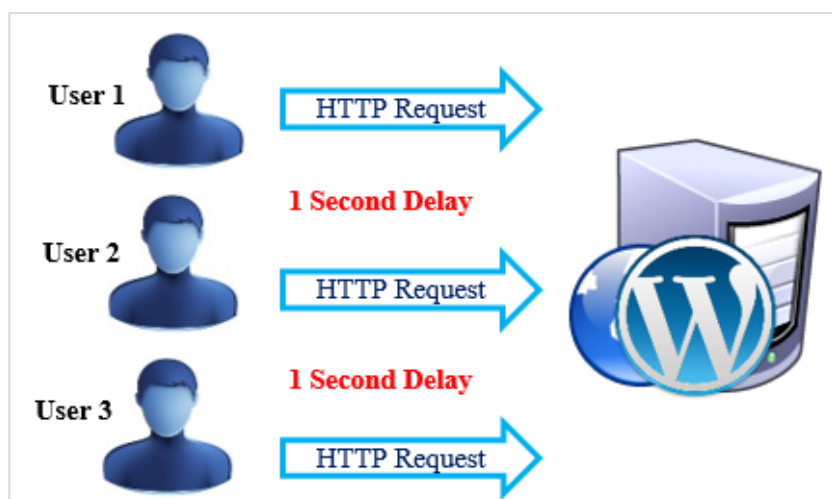


Figure 54. Ramp-Up Period Representation

Figure 54, depicts the idea of Ramp-Up Period. Ramp-Up Period tells JMeter to wait for X seconds before creating a new user request. For instance, the first entry in the experiment workload represents 0 as Ramp-Up Period and the number of threads (users) are 25. This tells JMeter to delay 0 second before starting new user ($0 / 25 = 0$), meaning all Threads will initialize simultaneously.

If the Ramp-Up Period is 0, JMeter will initialize all users at once (concurrently) without any delays. As described in section 5.12, having 0 Ramp-Up Period for the large number of users is not considered a good practice because it may cause the application to crash by starting all users as once. This is the reason that non-zero Ramp-Up Period is configured for the 1,000 and 2,000 users (100 and 500 respectively). When processing 1,000 Threads, JMeter waits for the 0.10 seconds before starting the new Threads ($100 / 1,000$). The last workload is to generate 2,000 users and in this case JMeter waits for the 0.25 seconds before starting new threads ($500 / 2,000$). With these workloads multiple thread within the same Test Plan simulate groups of concurrent, individual connections to the web application. Concurrency is still available even a non-zero Ramp-Up Period is defined because JMeter waits to start the new thread (according to time specified by Ramp-Up Period) while previously started thread is still under progress. For 2,000 users, JMeter starts the first thread and wait for the 0.25 seconds ($500 / 2,000$) to start the second thread while the previously started thread is still running. It is also practical to have large number of web requests but not necessarily that all uses should request the resources exactly the same time (there will be delays in milliseconds).

The number of time the test is repeated is specified by the Loop Count (Threads x Loop Count). If a Loop Count is configured to 2 and the Threads are 25, then JMeter will simulate the 50 user request ($25 \times 2 = 50$). The applied workloads in this implementation is configured to repeat only once as represented by 1 in Table 6.

Data Collection with JMeter

JMeter can represent results in table, tree and graph formats. When using JMeter in graphical mode (GUI), live results can observe as well. Figure 55 shows an example of a JMeter test illustrating the success and failure of the response time criteria.

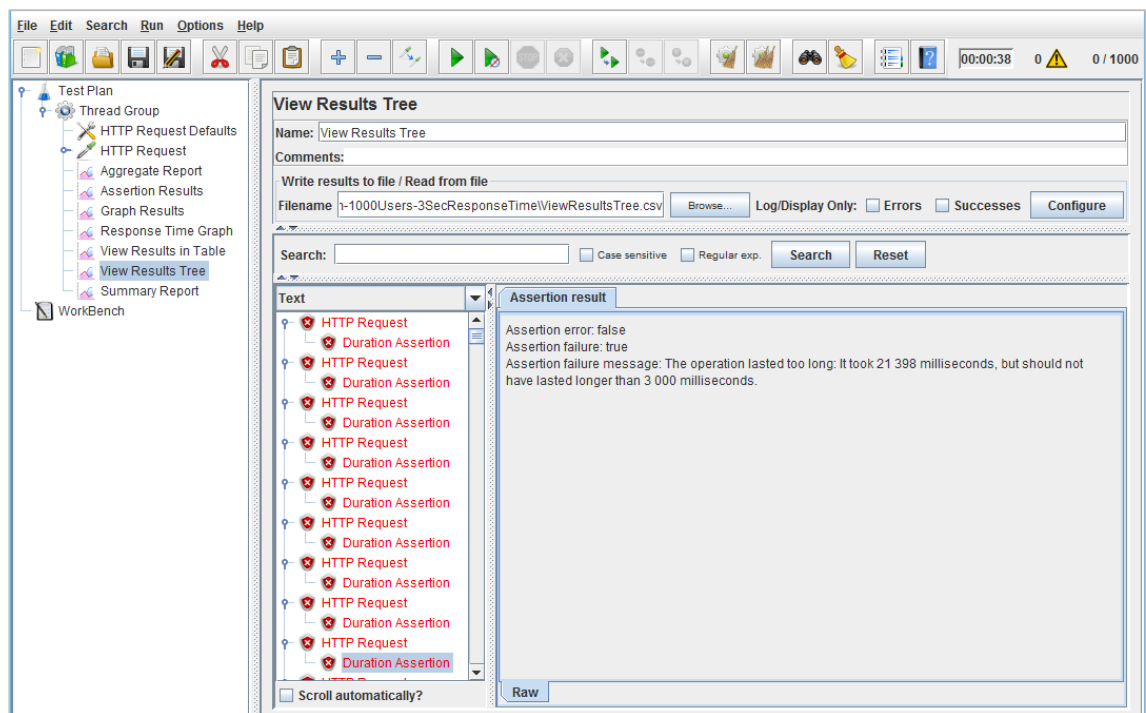


Figure 55. Experiment Results in Tree Format in JMeter

Figure 55 depicts the HTTP requests results based on the assertion time. If the HTTP request was facilitated within the defined criteria then results were marked as successful, otherwise as fail. Green and red colors were used to distinguish between the success and failures of the HTTP requests. Similarly, Figure 56 depicts the JMeter output in the table format.

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename: cySystem-1000Users-3SecResponseTimeTableResults.csv

Log/Display Only: Errors Successes

Sample #	Start Time	Thread Name	Label	Sample Tim...	Status	Bytes	Sent Bytes	Latency	Connect Ti
1	16:25:50.732	Thread Grou...	HTTP Request	1966	✓	54595	153	1929	
2	16:25:50.732	Thread Grou...	HTTP Request	1969	✓	54595	153	1929	
3	16:25:50.977	Thread Grou...	HTTP Request	2156	✓	54595	153	2059	
4	16:25:51.019	Thread Grou...	HTTP Request	2107	✓	54595	153	2022	
5	16:25:50.764	Thread Grou...	HTTP Request	2651	✓	54595	153	2540	
6	16:25:50.802	Thread Grou...	HTTP Request	2611	✓	54595	153	2499	
7	16:25:51.006	Thread Grou...	HTTP Request	2467	✓	54595	153	2383	
8	16:25:50.803	Thread Grou...	HTTP Request	2680	✓	54595	153	2596	
9	16:25:51.014	Thread Grou...	HTTP Request	2486	✓	54595	153	2396	
10	16:25:50.713	Thread Grou...	HTTP Request	2794	✓	54595	153	2696	
11	16:25:50.975	Thread Grou...	HTTP Request	2533	✓	54595	153	2441	
12	16:25:50.709	Thread Grou...	HTTP Request	2798	✓	54595	153	2688	
13	16:25:50.991	Thread Grou...	HTTP Request	2522	✓	54595	153	2423	
14	16:25:50.802	Thread Grou...	HTTP Request	2694	✓	54595	153	2600	
15	16:25:50.991	Thread Grou...	HTTP Request	2518	✓	54595	153	2418	
16	16:25:51.016	Thread Grou...	HTTP Request	4107	✗	54595	153	4013	
17	16:25:50.803	Thread Grou...	HTTP Request	4381	✗	54595	153	4286	
18	16:25:50.763	Thread Grou...	HTTP Request	5125	✗	222	153	5125	
19	16:25:51.014	Thread Grou...	HTTP Request	4906	✗	222	153	4906	
20	16:25:50.634	Thread Grou...	HTTP Request	5286	✗	222	153	5286	
21	16:25:50.990	Thread Grou...	HTTP Request	4930	✗	222	153	4930	
22	16:25:51.015	Thread Grou...	HTTP Request	4906	✗	222	153	4906	
23	16:25:50.989	Thread Grou...	HTTP Request	4941	✗	222	153	4941	
24	16:25:50.991	Thread Grou...	HTTP Request	4939	✗	222	153	4939	
25	16:25:50.713	Thread Grou...	HTTP Request	5217	✗	222	153	5217	

Figure 56. Experiment Results in Table Format in JMeter

In Figure 56 JMeter presents results in table format and marked the “Status” in green and red colours to represent the success and failure of the HTTP requests with details about the sample number, start time and thread name. JMeter also provides the “Summary Report” of the experiments that includes throughput, error, and standard deviation as shown in the Figure 57.

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename: smLegacySystem-1500Users-3SecResponseTime-75RampU

Log/Display Only: Errors Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
Thread Gro...	1500	3736	95	8932	1882.22	98.53%	18.4/sec	137.50	2.75	7653.0
TOTAL	1500	3736	95	8932	1882.22	98.53%	18.4/sec	137.50	2.75	7653.0

Include group name in label? Save Table Data Save Table Header

Figure 57. JMeter Summary Report

Figure 57 shows “Summary Report” generated by JMeter. JMeter provides options to save these results for future analysis by writing the results in CSV-format (Comma Separated Values) and plotting graphs of the experiment results. The user can select the location where the CSV file should be saved. Similarly, JMeter can plot graphs from the test results with a customized time interval.

6.3 Results

This section describes and analyzes the findings based on different experiments. WordPress web application was deployed on the Amazon cloud platform. Performance comparison of the proposed solution with the traditional (legacy) infrastructure was made based on these experimental results.

The ability of a web server to handle the heavy load was represented by throughput. As mentioned in section 5.12, throughput represents the transactions or requests that can be made in a given period of time. It is important to note that one should not purely depend on the throughput metrics and it needs to be viewed in conjunction with latency and response times. Response time and business SLA of 3 seconds was represented by Assertion Time in JMeter. JMeter calculates the error rate and present total percentage of the error in Summary Report together with the Throughput (as shown in Figure 57). This means that high throughput and reduced percentage of the error indicates the high performance of the system. Similarly, if a system shows high throughput but high error rate as well, this identifies decline in the system performance because it took too long to process the user requests.

Table 7 represents the results of the performance of the web application running on the legacy system (non-scalable) with experiment workloads as mentioned in section 6.2.

Table 7. Experiment Results of Legacy System

Experiment Results of Legacy System (Non-Scalable)			
Total Simulated Users (Threads)	Assertion Time (Milliseconds)	Throughput (Per second)	Error (%)
1 – 25	3000	5.0	40.00
1 – 50	3000	6.8	74.00
1 – 100	3000	14.3	86.00
1 – 500	3000	17.1	97.00
1 – 1,000	3000	9.8	74.30
1 – 2,000	3000	9.6	96.65

Table 7 represents the experimental results of the performance of the web server hosted on a single computer (i.e. legacy implementation).

The legacy system continued to decline in the performance with the increase in the applied workload. The applied workload for the first 25 users resulted in an overall error rate of 40%, representing those requests that were failed to respond in a timely manner (3000 milliseconds). The web server performance continued to decline as 74.30% of the accumulated error was recorded when simulated 1,000 users. Eventually, the web server almost stopped responding when applied workload of 2,000 users as represented by 96.65% error rate and Throughput of 9.6, as shown in Table 7. The legacy system was subject to decline in the performance with the increase in the workloads, causing the unavailability of the web application and service termination.

After collecting the results from the legacy system, the same workloads were applied to the proposed solution (scalable) hosted on cloud platform adhering to the scalability metrics and Elastic Load Balancer (ELB). This load balancer dynamically distributed the load on registered EC2 instances (virtual machines) and configured the new virtual machine(s) according to the auto-scaling policies (as described in chapter 5). Results of the

experiments after deploying the web application with the scalable cloud platform are presented in Table 8.

Table 8. Experiment Results of the Scalable Cloud Platform

Experiment Results of the Scalable Cloud Platform			
Total Simulated Users (Threads)	Assertion Time (Milliseconds)	Throughput (Per second)	Error (%)
1 – 25	3000	10.2	0.00
1 – 50	3000	11.6	32.00
1 – 100	3000	16.9	59.00
1 – 500	3000	4.6	92.80
1 – 1,000	3000	10.0	0.00
1 – 2,000	3000	4.0	0.00

Table 8 represents the results of the experiments after deploying the web application using the scalable cloud platform.

Compared to the results of the cloud implementation, as represented in Table 8, significantly better results were observed as the error rate for the first 25 users was 0.0 % (for legacy system it was 40.0 %). For the 50 users, an error of 32.0 % was logged, representing those request that failed the SLA of 3000 milliseconds (as presented in Table 8). This minor error did not cause the web application to decline in the performance and was considered as an neglectable error when comparing to the legacy system (Table 7), where 74.0 % of the accumulated error was logged for the 50 users. For the 500 users with Ramp-Up Period of 0, the scalable system produced the error rate of 92.80 % which is too high as well. However, this can be controlled by increasing the number of EC2 instances to produce the lower error rate. This implementation used 3 instances as “minimum” that can be increased by a cloud administrator to meet the business needs.

The scalable system resulted in an error of 0.0 % when processing the 1,000 user request with a Ramp-Up Period of 100. While the legacy system was subject to 74.30 % of the error. Similarly, the scalable system showed 0.0 % error for the 2,000 users (with

Ramp-Up Period of 500) while the legacy system produced 96.65 % of error when processing the same workload.

6.4 Scalability Analysis

With auto-scaling policies and a scaling group, it was possible to limit the number of the nodes to be provisioned (EC2 Instances) by defining the minimum and maximum in the auto-scaling configuration. The resources defined as “minimum” were present all the time while the additional resources continued to become available as soon as the increase in the current demand was discovered. This increase in the current demand was simulated by generating heavy workloads (described in section 6.2). This process continues until the “maximum” number of the resources to be provisioned has been reached.

This implementation was comprising of the 3 (minimum) and 9 (maximum) number of the EC2 instances as described in section 5.7. Table 9 presents the number of the EC2 instances that were participating to manage the increased workload.

Table 9. Scalability Analysis of the Cloud Platform

Scalability Analysis of the Cloud Platform				
Total Simulated Users (Threads)	Virtual Machines (EC2 Instances)	Assertion Time (Milliseconds)	Throughput (Per second)	Error (%)
1 – 25	3	3000	10.2	0.00
1 – 50	3	3000	11.6	32.00
1 – 100	3	3000	16.9	59.00
1 – 500	3	3000	4.6	92.80
1 – 1,000	3	3000	10.0	0.00
1 – 2,000	9	3000	4.0	0.00

An auto-scaling event was triggered when a scalability metric reached the threshold limit for a specific time period as defined in the scaling policy.

For the first 1,000 users, only 3 virtual machines (EC2 instances) were servicing the requests (the minimum number of instances the Auto Scaling group should have at any

time). However, for the increased workload (2,000), the number of EC2 instances increased to 6 and continued to increase until a limit of the maximum number of EC2 Instances reached as represented in Table 9. This shows that no more virtual machines were provisioned because the maximum limit of the EC2 instances that a scaling policy can provision had been reached. However, the size of the scaling group can be increased (or decreased) whenever required to adopt the new requirements.

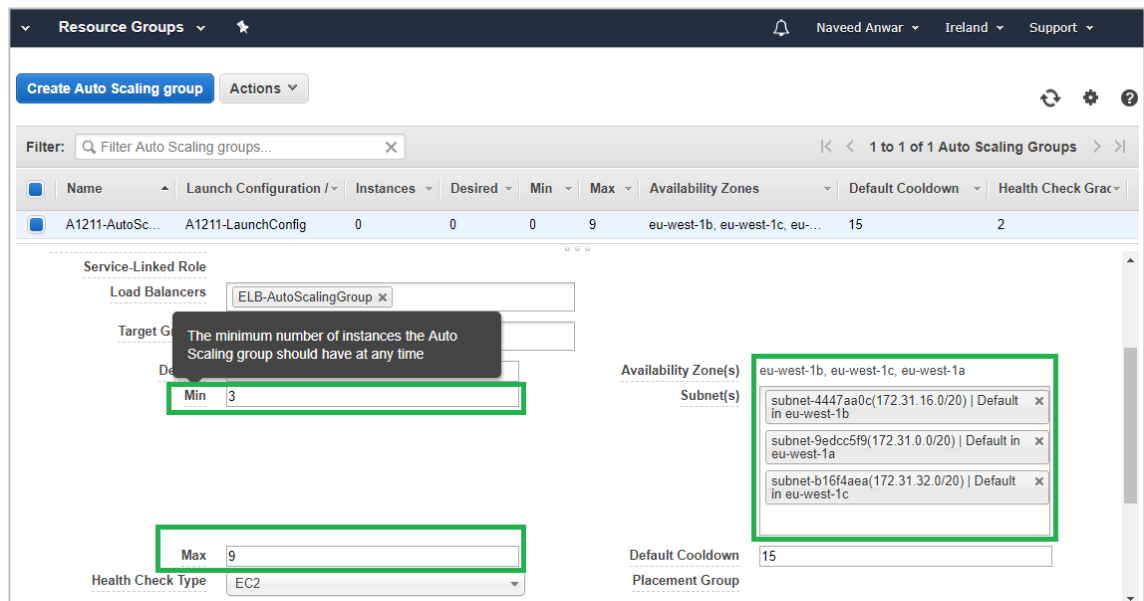


Figure 58. Defining Minimum and Maximum Number of EC2 Instances

Figure 58 represents the configuration of the minimum and maximum number of EC2 Instances. This implementation uses all available availability zones (eu-west-1b, eu-west-1c, eu-west-1a) to make application highly available.

These newly created EC2 instances were scaled out automatically i.e. decommissioned with the decline in the resource demand. Before retiring a particular EC2 instance, auto-scaling service checks and waits until all open connections have been closed. Figure 59 represents a partial screenshot of the Auto Scaling Activity History.

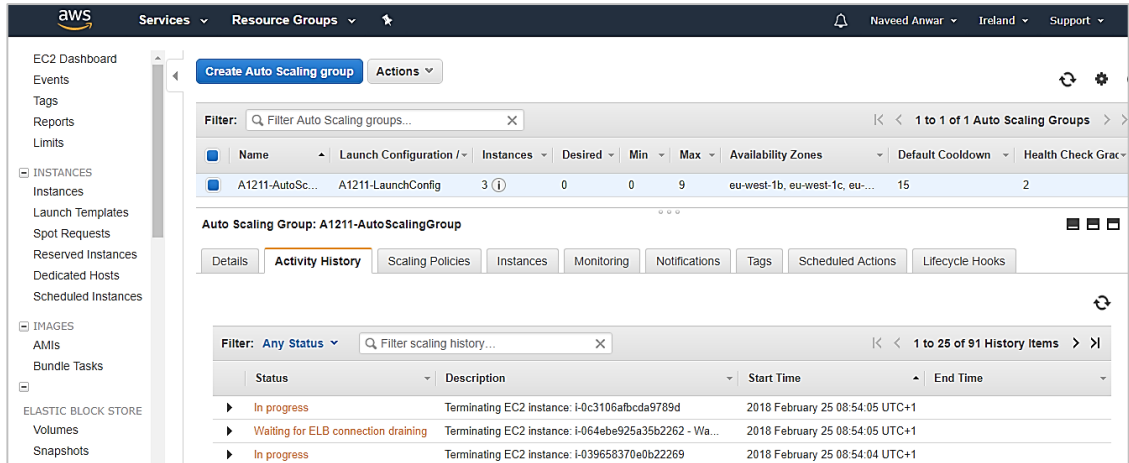


Figure 59. Auto Scaling Activity History

As depicted in Figure 59, termination of two instances is in progress while one is marked as “Waiting for the ELB connection draining”. Connection Draining allows existing requests to complete before the load balancer shifts traffic away from a deregistered or unhealthy back-end instance. This value of the Connection Draining is configurable and can be adjusted any time by selecting the appropriate load balancer as shown in below figure.

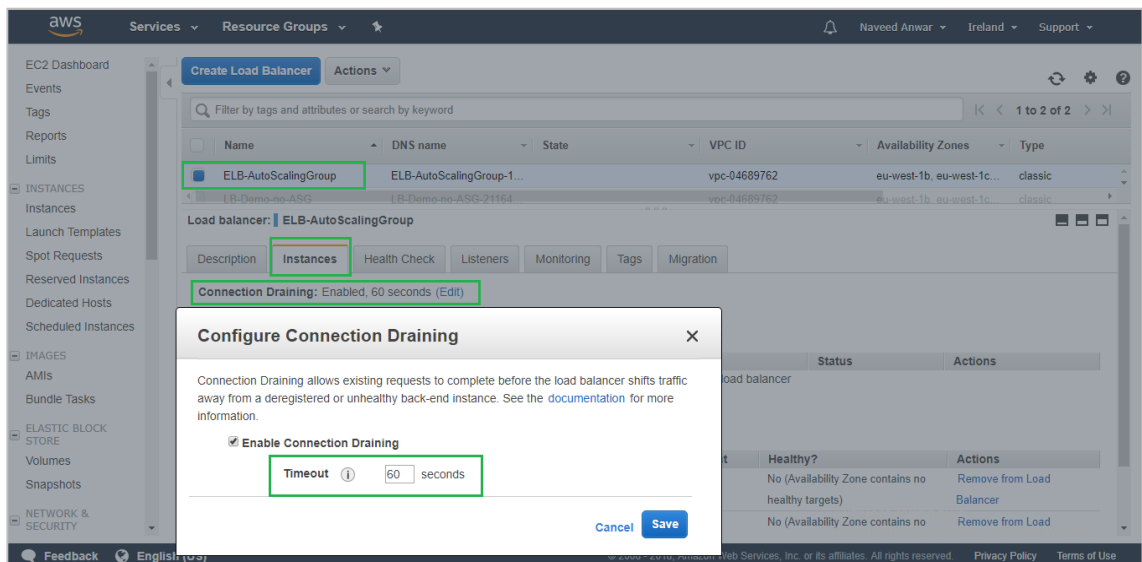


Figure 60. Connection Draining Configuration

Figure 60 represents the Connection Draining configuration which defines the number of seconds to allow existing traffic to continue flowing for the X number of seconds provided by administrator.

Table 10 shows the time in seconds to terminate the EC2 instances when there was a decline in the resource demand.

Table 10. Scale Out Time of EC2 Instances

Scale Out Time (EC2 instances)		
Termination Start Time (UTC + 1)	Termination End Time (UTC + 1)	Cause of Termination (Description)
08:32:10	08:33:50	A monitor alarm ReducedResourceUtilization in state ALARM triggered policy ReducedResourceUtilization changing the desired capacity from 9 to 6. An instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 6 to 3.
08:32:10	08:33:29	A monitor alarm ReducedResourceUtilization in state ALARM triggered policy ReducedResourceUtilization changing the desired capacity from 9 to 6. An instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 6 to 3.
08:32:10	08:33:58	A monitor alarm ReducedResourceUtilization in state ALARM triggered policy ReducedResourceUtilization changing the desired capacity from 9 to 6.
08:33:09	08:34:32	A monitor alarm ReducedResourceUtilization in state ALARM triggered policy ReducedResourceUtilization changing the desired capacity from 6 to 3. An instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 6 to 3.
08:33:09	08:34:28	A monitor alarm ReducedResourceUtilization in state ALARM triggered policy ReducedResourceUtilization changing the desired capacity from 6 to 3. An instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 6 to 3.
08:33:09	08:34:49	A monitor alarm ReducedResourceUtilization in state ALARM triggered policy

		<p>ReducedResourceUtilization changing the desired capacity from 6 to 3. An instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 6 to 3.</p>
--	--	--

With auto-scaling policies, the previously provisioned resources (EC2 Instances) were terminated dynamically with the decline in the resource demand, as presented in Table 10. With dynamic scaling, the proposed solution tends to be cost-effective as the user of the cloud platform is charged only for the consumed resources for a certain amount of the time. Also, the increase or decrease in the resource demand was executed without the intervention of the cloud administrator.

7 Discussions and Conclusions

A number of experiments were performed to evaluate the performance of the proposed cloud-based solution (scalable) and comparing the results with the legacy infrastructure provisioning approach. The same workloads were applied to both systems by simulating concurrent user requests (as described in section 6.2). The results presented in chapter 6 showed that the legacy system was not capable to manage the heavy load and the performance declined rapidly with increasing total number of simulated users. Eventually, the system almost stopped responding and a decrease in the system throughput was observed. This behavior reduced the quality of the user experience because the resources users were requesting were not responding (service unavailable). When comparing to the cloud platform, significantly improved results were observed when the same web application was deployed using the cloud platform; consisting on the virtual machine instances, a cloud load balancer, and auto-scaling policies.

7.1 Conclusion

The main objective of this research project was to design, implement and evaluate a scalable cloud architecture to host a web application (WordPress). This study also examined the scenarios of scaling up (to provision more resources) and scaling down (terminating the unwanted resources). Experiments were conducted to solve the research problem, answer the research question, and evaluate the proposed solution. Required

data were collected with JMeter; a platform-independent, open source, and very commonly used performance measurement tool.

Comparison of Table 7 and Table 8 (chapter 6) showed that overall performance of the web application can be improved with the scalable cloud platform without a decline in the performance and maintaining the budget by paying only for the consumed resources. With the traditional approach of infrastructure provisioning, an existing system may be upgraded to some extent both in terms of hardware and software to increase the capacity of the system. Eventually, there comes a point when no more upgrades are possible. For instance, the legacy system had two memory slots which were already in use and it was impossible to improve the system performance. It is also important to note that the system must be powered off during maintenance which results in service interruption as well.

For dynamic nature of web applications, there may be seasonal high workloads, i.e. the application is highly demanded only a few occasions during a year. This causes the resources of the legacy system to be underutilized and remains in the idle state until an increase in demand is taking place. This approach results in an overall expensive system due to over-provisioning and underutilization of the compute resources.

The expected results were achieved with the proposed solution using scalable cloud platform. The implemented scaling policies comprises on the number of the virtual machines that should be available at a given time. It was possible to define the maximum number of the virtual machine instances to be initialized dynamically with the increase in the current demand. Similarly, a mechanism to terminate these additional resources was defined with the decline in the current demand. With these features the overall cost can be reduced as cloud service provider only charge for the consumed resources. This answers the research question that using a cloud platform, a web application can handle high workload dynamically by allocating additional resources and decommissioning these resources when no longer required. To provide user experience with good quality, connection draining was defined (section 5.8) that waits for the current transactions to complete before retiring the resources which were currently servicing the user request. Connection draining defines the number of seconds to allow existing traffic to continue flowing the number of seconds provided by administrator.

The proposed solution is generic in nature that may be used in any enterprise aiming to build IT infrastructure in the cloud and make their web services more reliable. This applies to the dynamic nature of web traffic or when the company is going to start a new business where the expected number of users or web traffic is not known in advance. The proposed solution presented in this thesis was implemented with EC2 instance type T2 Micro, eligible to use as free for 12 months under free tier membership. The results may vary if another instance type would be used to replicate this implementation. Similarly, the results would be different if scaling metrics are changed. For instance, this implementation used a duration assertion (response time) as success criteria of 2 seconds (2000 milliseconds). Amazon cloud services are paid services; however, it was possible to perform these experiments under the student license offered by AWS Educate and Student Developer Pack from GitHub Education. This educational package was worthy enough to perform this implementation and experiments. To use this proposed solution in a production environment, it may require a paid subscription or subject to incur charges after the expiration of the free tier. Open source software was used during experiments and testing, which were free under GNU General Public License. Legal and governmental issues were not discussed in this project as this study did not specifically involve a particular business or an organization. When implemented in a production environment, it may require exposing the company information to the legal and national authorities due to local cyber laws and the nature of the business.

7.2 Future Work

Further research may be carried out to broaden the performance of a web application for unknown users with an additional caching tier in the reference architecture for the scalable web application. The existing solution was implemented with auto-scaling service and a cloud load balancer via AWS console web page. Additional exploration can be performed to automate this implementation with scripting languages and AWS command line tools. Information security is another major area of consideration that deals with the security architecture of the web application. With the increase in the online traffic, there is an increase in the information security risks as well. Financial institutes, consumer banks, e-commerce business are focusing on the security and privacy of the information. The proposed solution can be studied from the information security perspectives to design highly scalable and secure web applications.

Besides expanding the research area, a comparison of the multiple cloud service providers may be performed as well. For example, the same implementation can be evaluated with Microsoft Azure cloud platform and compare the performance of web server running on multiple cloud platforms to perform a cost analysis.

References

- [1] D. Rountree, I. Castrillo and I. Books24x7, *The Basics of Cloud Computing: Understanding the Fundamentals of Cloud Computing in Theory and Practice*. 2014;2012; 2013.
- [2] "What is Cloud Computing? - Amazon Web Services", Amazon Web Services, Inc., 2017. [Online]. Available: <https://aws.amazon.com/what-is-cloud-computing/>. [Accessed: 20- Dec- 2017].
- [3] P. Mell and T. Grance, *The NIST Definition of Cloud Computing*, National Institute of Standards and Technology (NIST), Gaithersburg, MD, 800–145, 2011.
- [4] Rosenberg and A. Mateos, *The cloud at your service*. Greenwich, Conn.: Manning, 2011.
- [5] D. C. Marinescu *et al*, "An approach for scaling cloud resource management," *Cluster Computing*, vol. 20, (1), pp. 909-924, 2017.
- [6] R. Mietzner *et al*, "Combining different multi-tenancy patterns in service-oriented applications," *2009 IEEE International Enterprise Distributed Object Computing Conference*, 2009. DOI: 10.1109/EDOC.2009.13.
- [7] B. Tang, R. Sandhu and Q. Li, "Multi-tenancy authorization models for collaborative cloud services," *Concurrency and Computation: Practice and Experience*, vol. 27, (11), pp. 2851-2868, 2015.
- [8] J. Zhu *et al*, "Towards bandwidth guarantee in multi-tenancy cloud computing networks," *2012 20th IEEE International Conference on Network Protocols (ICNP)*, 2012. DOI: 10.1109/ICNP.2012.6459986.
- [9] S. Heinzl and C. Metz, "Toward a cloud-ready dynamic load balancer based on the apache web server," *2013 Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2013. DOI: 10.1109/WETICE.2013.63.
- [10] T. Baars *et al* Heusden, "Chargeback for cloud services", *Future Generation Computer Systems*, vol. 41, pp. 91-103, 2014.
- [11] Introduction to Cloud Computing: IEEEx: CloudIntro.x, "Part 2: Dynamic Interactions and Computing Architectures," May 14, 2016. [Online]. Available: <https://courses.edx.org/courses/course-v1:IEEEx+CloudIntro.x+2T2016/course/> [Accessed: Dec. 25, 2017].

- [12] "The Cloud at Your Service - Documents", *Docslide.net*, 2017. [Online]. Available: <https://docslide.net/documents/the-cloud-at-your-service.html>. [Accessed: 25- Dec- 2017].
- [13] "My i:MOOC", *Imooc.co.kr*, 2017. [Online]. Available: http://www.imooc.co.kr/local/main/study_common_micro.php?id=2325§ion=7040&module=23508. [Accessed: 25- Dec- 2017].
- [14] "Figure 15: IaaS Provider/", *yumpu.com*, 2017. [Online]. Available: <https://www.yumpu.com/en/document/view/14248355/cloud-computing-synopsis-and-recommendations/51>. [Accessed: 25- Dec- 2017].
- [15] A. Molnar, M. Drongelen, P. Subramanian, V. Harsh, R. Messier and R. Sammut, "NIST Cloud Computing Synopsis and Recommendations | Cloud Computing | Platform As A Service", *Scribd*, 2017. [Online]. Available: <https://www.scribd.com/document/95221991/NIST-Cloud-Computing-Synopsis-and-Recommendations>. [Accessed: 25- Dec- 2017].
- [16] Introduction to Cloud Computing: IEEEEx: CloudIntro.x, "Part 2: Dynamic Interactions and Computing Architectures PaaS Dynamics and Software Stack Control," May 14, 2016. [Online]. Available: <https://courses.edx.org/courses/course-v1:IEEEEx+CloudIntro.x+2T2016/course/> [Accessed: Dec. 25, 2017].
- [17] A. Pokahr and L. Braubach, "Elastic component-based applications in PaaS clouds," *Concurrency and Computation: Practice and Experience*, vol. 28, (4), pp. 1368-1384, 2016.
- [18] "Chapter 14: Cloud Computing Security Essentials and ... - NIST Web Site", *Ws680.nist.gov*, 2017. [Online]. Available: http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=919233. [Accessed: 25- Dec- 2017].
- [19] "01-2-Service_and_Deployment_Infrastructure_and_Consumer_View", *Ki.fpv.ukf.sk*, 2017. [Online]. Available: http://www.ki.fpv.ukf.sk/~mdrlik/cloud/01-2-Service_and_Deployment_Infrastructure_and_Consumer_View.txt. [Accessed: 25- Dec- 2017].
- [20] Introduction to Cloud Computing: IEEEEx: CloudIntro.x, "Part 2: Dynamic Interactions and Computing Architectures SaaS Interaction Dynamics and Software Stack Control," May 14, 2016. [Online]. Available: <https://courses.edx.org/courses/course-v1:IEEEEx+CloudIntro.x+2T2016/course/> [Accessed: Dec. 25, 2017].
- [21] K. Tam and R. Sehgal, "A cloud computing framework for on-demand forecasting services," in 2014. DOI: 10.1007/978-3-319-11167-4_35.

- [22] T. Erl, R. Puttini and Z. Mahmood, *Cloud Computing: Concepts, Technology & Architecture*. (1st ed.) 2013.
- [23] B. Schultz, "PUBLIC VS. PRIVATE CLOUDS WHY NOT BOTH?" *Network World*, vol. 28, (7), pp. 20, 2011.
- [24] K. Jamsa, *Cloud computing*. Burlington: Jones & Bartlett Learning, 2013.
- [25] M. v. d. Berg, *Managing Microsoft Hybrid Clouds*. (1st ed.) 2015.
- [26] Portnoy and I. Books24x7, *Virtualization Essentials*. (1. Aufl. ed.) 2012.
- [27] G. Popek and R. Goldberg, "Formal requirements for virtualizable third generation architectures", *Communications of the ACM*, vol. 17, no. 7, pp. 412-421, 1974.
- [28] R. Dittner and D. Rule Jr, *Best Damn Server Virtualization Book Period*. (1st ed.) 2011.
- [29] D. Kusnetzky, *Virtualization: A Manager's Guide*. (1st ed.) 2011.
- [30] Z. H. Shah, *Windows Server 2012 Hyper-V: Deploying the Hyper-V Enterprise Server Virtualization Platform*. (1st ed.) 2013.
- [31] B. Wilder, *Cloud architecture patterns*. Sebastopol, Calif.: O'Reilly, (1st ed.) 2012.
- [32] J. R. Hauser and G. L. Urban, *From Little's Law to Marketing Science: Essays in Honor of John D.C. Little*. 2016.
- [33] John D. C. Little, "Little's Law as Viewed on Its 50th Anniversary," *Operations Research*, vol. 59, (3), pp. 536-549, 2011.
- [34] S. K. Shivakumar, *Architecting high performing, scalable and available enterprise web applications*, 1st ed. Waltham, MA: Elsevier, 2015.
- [35] *Amazon CloudWatch User Guide*. Amazon Web Services, 2017. [Online]. Available: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/acw-ug.pdf#CW_Support_For_AWS. [Accessed: 12- Nov- 2017]. pp. 104-111
- [36] M. Handa and S. Sharma, "Cloud Computing: A study of cloud architecture and its patterns," *International Journal of Engineering Research and Applications*, vol. 5, (5), pp. 11-16, 2015.
- [37] "WordPress.com: Create a website or blog", *WordPress.com*, 2017. [Online]. Available: <https://wordpress.com>. [Accessed: 12- Nov- 2017].
- [38] The Apache Software Foundation. (2017). *Welcome! - The Apache HTTP Server Project*. [online] Available at: <https://httpd.apache.org/> [Accessed 3 Nov. 2017].

- [39] J. M. Gallagher and S. C. Ramanathan, "Choosing a Client/Server Architecture: A Comparison of Two-and Three-Tier Systems," *Information Systems Management*, vol. 13, (2), pp. 7-13, 1996.
- [40] C. Koppurapu, *Load balancing servers, firewalls, and caches*. New York: Wiley, 2002.
- [41] M. Syme and P. Goldie, *Optimizing Network Performance with Content Switching: Server, Firewall, and Cache Load Balancing*. (First ed.) 2003;2004.
- [42] Amazon Web Services, Inc. (2017). *Amazon Web Services (AWS) - Cloud Computing Services*. [online] Available at: <https://aws.amazon.com/> [Accessed 3 Nov. 2017].
- [43] M. Arregoces *et al*, *Data Center Fundamentals: Understand Data Center Network Design and Infrastructure Architecture, Including Load Balancing, SSL, and Security*. 2004.
- [44] P. Killelea, *Web Performance Tuning*. China: 2002.
- [45] Amazon Web Services, Inc. (2017). *Elastic Compute Cloud (EC2) – Cloud Server & Hosting – AWS*. [online] Available at: <https://aws.amazon.com/ec2/> [Accessed 3 Nov. 2017].
- [46] Amazon Web Services, Inc. (2017). *Amazon Web Services (AWS) - Cloud Computing Services*. [online] Available at: <https://aws.amazon.com/> [Accessed 3 Nov. 2017].
- [47] "Regions and Availability Zones - Amazon Relational Database Service", *Docs.aws.amazon.com*, 2018. [Online]. Available: <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.RegionsAndAvailabilityZones.html>. [Accessed: 07- Jan- 2018].
- [48] "AWS Regions and Endpoints - Amazon Web Services", *Docs.aws.amazon.com*, 2018. [Online]. Available: <https://docs.aws.amazon.com/general/latest/gr/rande.html>. [Accessed: 07- Jan- 2018].
- [49] S. Naik, *Concepts of Database Management System*. (1st ed.) 2013.
- [50] P. DuBois, *MySQL*. Upper Saddle River, NJ: Addison-Wesley, 2013.
- [51] L. Sikos and I. Books24x7, *Web Standards: Mastering HTML5, CSS3, and XML*. (1st ed.) 2011; 2012. DOI: 10.1007/978-1-4302-4042-6.
- [52] E. Kralicek, *The Accidental Sys Admin Handbook: A Primer for Early Level IT Professionals*. 2016.
- [53] "Elastic IP Addresses - Amazon Elastic Compute Cloud", *Docs.aws.amazon.com*, 2018. [Online]. Available: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html>. [Accessed: 07- Jan- 2018].

- [54] "EC2 Instance Pricing – Amazon Web Services (AWS)", *Amazon Web Services, Inc.*, 2018. [Online]. Available: https://aws.amazon.com/ec2/pricing/on-demand/#Elastic_IP_Addresses. [Accessed: 11- Jan- 2018].
- [55] "Amazon CloudWatch - Cloud & Network Monitoring Services", *Amazon Web Services, Inc.*, 2018. [Online]. Available: <https://aws.amazon.com/cloudwatch/>. [Accessed: 12- Jan- 2018].
- [56] Y. Wadia, *AWS Administration – the Definitive Guide*. (1st ed.) 2016.
- [57] V. Garousi, "A Genetic Algorithm-Based Stress Test Requirements Generator Tool and Its Empirical Evaluation", *IEEE Transactions on Software Engineering*, vol. 36, no. 6, pp. 778-797, Nov. 2010.
- [58] "Apache JMeter - Apache JMeter™", *Jmeter.apache.org*, 2018. [Online]. Available: <http://jmeter.apache.org>. [Accessed: 04- Nov- 2017].
- [59] B. Erinle, *Performance Testing with JMeter 3 - Third Edition*. Packt Publishing, 2017.
- [60] "Java SE Development Kit 8 - Downloads", *Oracle.com*, 2018. [Online]. Available: <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>. [Accessed: 05- Nov- 2017].
- [61] "Apache JMeter - Download Apache JMeter", *Jmeter.apache.org*, 2018. [Online]. Available: http://jmeter.apache.org/download_jmeter.cgi. [Accessed: 04- Nov- 2017].
- [62] J. P. Mueller, *AWS for Admins for Dummies*. (2;1; ed.) 2016.
- [63] R. Udell and L. Chan, *AWS Administration Cookbook (1)*. (1st ed.) 2017.
- [64] M. Wittig and A. Wittig, *Amazon Web Services in Action*. (1st ed.). Manning Publications, 2015.
- [65] "Windows | Official Site for Microsoft Windows 10 Home, S & Pro OS, laptops, PCs, tablets & more", *Microsoft.com*, 2018. [Online]. Available: <https://www.microsoft.com/en-us/windows/>. [Accessed: 12- Jan- 2018].
- [66] "What is Linux?", *Linux.com | The source for Linux information*, 2018. [Online]. Available: <https://www.linux.com/what-is-linux>. [Accessed: 12- Jan- 2018].
- [67] "Ubuntu PC operating system | Ubuntu", *Ubuntu.com*, 2018. [Online]. Available: <https://www.ubuntu.com/desktop>. [Accessed: 12- Jan- 2018].
- [68] "macOS - What is macOS", *Apple*, 2018. [Online]. Available: <https://www.apple.com/macos/what-is/>. [Accessed: 12- Jan- 2018].
- [69] "AWS Educate", *Amazon Web Services, Inc.*, 2017. [Online]. Available: <https://aws.amazon.com/education/awseducate/>. [Accessed: 10- Nov- 2017].

- [70] "Apply for AWS Educate", *Amazon Web Services, Inc.*, 2017. [Online]. Available: <https://aws.amazon.com/education/awseducate/apply/>. [Accessed: 10- Nov- 2017].
- [71] "GitHub Student Developer Pack", *GitHub Education*, 2017. [Online]. Available: <https://education.github.com/pack>. [Accessed: 10- Nov- 2017].
- [72] "Build software better, together", *GitHub*, 2018. [Online]. Available: <https://github.com/>. [Accessed: 12- Jan- 2018].
- [73] "AWS Free Tier", *Amazon Web Services, Inc.*, 2017. [Online]. Available: <https://aws.amazon.com/free/>. [Accessed: 12- Nov- 2017].
- [74] "Free Tier Limits - AWS Billing and Cost Management", *Docs.aws.amazon.com*, 2017. [Online]. Available: <https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/free-tier-limits.html>. [Accessed: 12- Nov- 2017].
<https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/billing-free-tier.html>
- [75] "Using the Free Tier - AWS Billing and Cost Management", *Docs.aws.amazon.com*, 2017. [Online]. Available: <https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/billing-free-tier.html>. [Accessed: 12- Nov- 2017].
- [76] "Learn What to Do When Your Free Tier Period is Expiring", *Amazon Web Services, Inc.*, 2017. [Online]. Available: <https://aws.amazon.com/premiumsupport/knowledge-center/free-tier-expiring/>. [Accessed: 12- Nov- 2017].
- [77] "Getting Started with Amazon EC2 Linux Instances - Amazon Elastic Compute Cloud", *Docs.aws.amazon.com*, 2017. [Online]. Available: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html#ec2-get-started-overview. [Accessed: 12- Nov- 2017].
- [78] "Tutorial: Installing a LAMP Web Server on Amazon Linux - Amazon Elastic Compute Cloud", *Docs.aws.amazon.com*, 2017. [Online]. Available: <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/install-LAMP.html>. [Accessed: 12- Nov- 2017].
- [79] "Tutorial: Hosting a WordPress Blog with Amazon Linux - Amazon Elastic Compute Cloud", *Docs.aws.amazon.com*, 2017. [Online]. Available: <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/hosting-wordpress.html>. [Accessed: 12- Nov- 2017].
- [80] "Tutorial: Hosting a WordPress Blog with Amazon Linux - Amazon Elastic Compute Cloud", *Docs.aws.amazon.com*, 2017. [Online]. Available:

- <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/hosting-wordpress.html#wordpress-next-steps>. [Accessed: 12- Nov- 2017].
- [81] "Changing The Site URL « WordPress Codex", *Codex.wordpress.org*, 2017. [Online]. Available: https://codex.wordpress.org/Changing_The_Site_URL. [Accessed: 12- Nov- 2017].
- [82] "Tutorial: Hosting a WordPress Blog with Amazon Linux - Amazon Elastic Compute Cloud", *Docs.aws.amazon.com*, 2017. [Online]. Available: <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/hosting-wordpress.html#wordpress-troubleshooting>. [Accessed: 12- Nov- 2017].
- [83] "Amazon Machine Images (AMI) - Amazon Elastic Compute Cloud", *Docs.aws.amazon.com*, 2017. [Online]. Available: <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html#ami-using>. [Accessed: 12- Nov- 2017].
- [84] "Amazon Machine Images (AMI) - Amazon Elastic Compute Cloud", *Docs.aws.amazon.com*, 2017. [Online]. Available: <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html#creating-an-ami>. [Accessed: 12- Nov- 2017].
- [85] "How Elastic Load Balancing Works - Elastic Load Balancing", *Docs.aws.amazon.com*, 2018. [Online]. Available: <https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/how-elastic-load-balancing-works.html>. [Accessed: 14- Jan- 2018].
- [86] J. Nadon, *Website Hosting and Migration with Amazon Web Services*. Berkeley, CA: Apress, 2017.
- [87] "Tutorial: Create a Classic Load Balancer - Elastic Load Balancing", *Docs.aws.amazon.com*, 2017. [Online]. Available: <http://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-getting-started.html#create-load-balancer>. [Accessed: 13- Nov- 2017].
- [88] Software Foundation. (2017). *Apache JMeter - User's Manual*. [online] Available at: <http://jmeter.apache.org/usermanual/index.html> [Accessed 7 Nov. 2017].
- [89] "The GNU General Public License v3.0- GNU Project - Free Software Foundation", *Gnu.org*, 2017. [Online]. Available: <https://www.gnu.org/licenses/gpl-3.0.en.html>. [Accessed: 16-Sep-2017].
- [90] E. H. Halili and I. Books24x7, *Apache JMeter: A Practical Beginner's Guide to Automated Testing and Performance Measurement for Your Websites*. 2008.

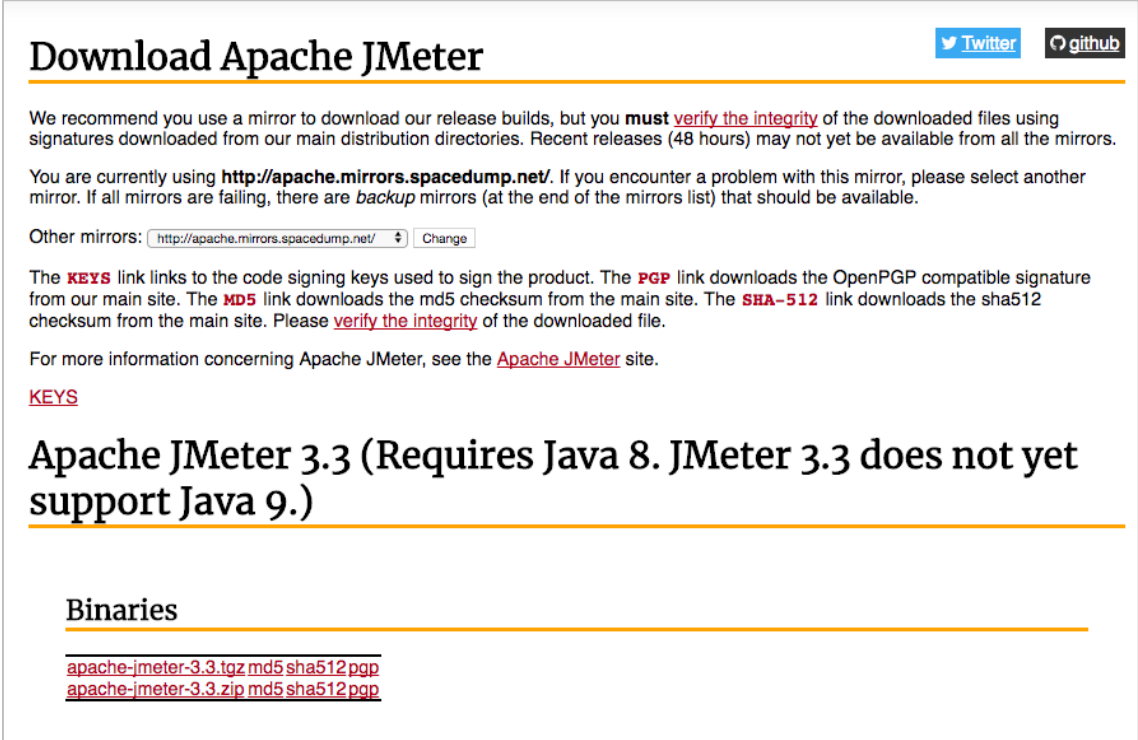
- [91] "Apache JMeter - User's Manual: Best Practices", *Jmeter.apache.org*, 2018.
[Online]. Available: <http://jmeter.apache.org/usermanual/best-practices.html>.
[Accessed: 17- Feb- 2018].
- [92] E. H. Halili and E. Halili, *apache Jmeter*. (1st ed.) 2008.

Preparing Experiment Environment with JMeter

This appendix provides information on how to download, install, configure and execute JMeter as well as required dependencies.

Prerequisite for Installing JMeter

At the time of writing this report, Apache JMeter 3.3 was the latest version available. Java 8 is required to run the JMeter 3.3, although Java 9 was the latest version available but wasn't supported yet, figure A shows the partial screenshot of the product download page (http://jmeter.apache.org/download_jmeter.cgi)



Download Apache JMeter [Twitter](#) [github](#)

We recommend you use a mirror to download our release builds, but you **must verify the integrity** of the downloaded files using signatures downloaded from our main distribution directories. Recent releases (48 hours) may not yet be available from all the mirrors.

You are currently using <http://apache.mirrors.spacedump.net/>. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are *backup* mirrors (at the end of the mirrors list) that should be available.

Other mirrors:

The **KEYS** link links to the code signing keys used to sign the product. The **PGP** link downloads the OpenPGP compatible signature from our main site. The **MD5** link downloads the md5 checksum from the main site. The **SHA-512** link downloads the sha512 checksum from the main site. Please **verify the integrity** of the downloaded file.

For more information concerning Apache JMeter, see the [Apache JMeter](#) site.

KEYS

Apache JMeter 3.3 (Requires Java 8. JMeter 3.3 does not yet support Java 9.)

Binaries

[apache-jmeter-3.3.tgz md5 sha512 pgp](#)
[apache-jmeter-3.3.zip md5 sha512 pgp](#)

Figure 61. Download Apache JMeter

Figure 53 depicts the partial screenshot of the product download page. On this download page, the Java 8 requirement is also highlighted.

Installing Java 8

Follow these steps to download and install the Java Standard Edition (SE) Development Kit 8.

1. Go to the following link [5].
<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
2. Download the Java JDK compatible to the operating system (Microsoft Windows, Linux, macOS, Solaris).
Figure 54 shows the available installation binaries for the Java SE Development Kit 8u152.

Java SE Development Kit 8u152		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
<input type="radio"/> Accept License Agreement <input type="radio"/> Decline License Agreement		
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.94 MB	jdk-8u152-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.88 MB	jdk-8u152-linux-arm64-vfp-hflt.tar.gz
Linux x86	168.99 MB	jdk-8u152-linux-i586.rpm
Linux x86	183.77 MB	jdk-8u152-linux-i586.tar.gz
Linux x64	166.12 MB	jdk-8u152-linux-x64.rpm
Linux x64	180.99 MB	jdk-8u152-linux-x64.tar.gz
macOS	247.13 MB	jdk-8u152-macosx-x64.dmg
Solaris SPARC 64-bit	140.15 MB	jdk-8u152-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.29 MB	jdk-8u152-solaris-sparcv9.tar.gz
Solaris x64	140.6 MB	jdk-8u152-solaris-x64.tar.Z
Solaris x64	97.04 MB	jdk-8u152-solaris-x64.tar.gz
Windows x86	198.46 MB	jdk-8u152-windows-i586.exe
Windows x64	206.42 MB	jdk-8u152-windows-x64.exe

Figure 62. Java SE Development Kit Demos and Samples Downloads

3. Accept the license agreement and select the installation setup.
4. Double click on setup and follow on-screen instructions.

Configuring Java_Home (Microsoft Windows)

Follow these steps to configure the Java_Home on Microsoft Windows 10.

1. Go to Control Panel.
2. Click on System.

3. Click on Advanced System settings.
4. Select Advanced tab clicks Environment Variables (as shown in below figure).

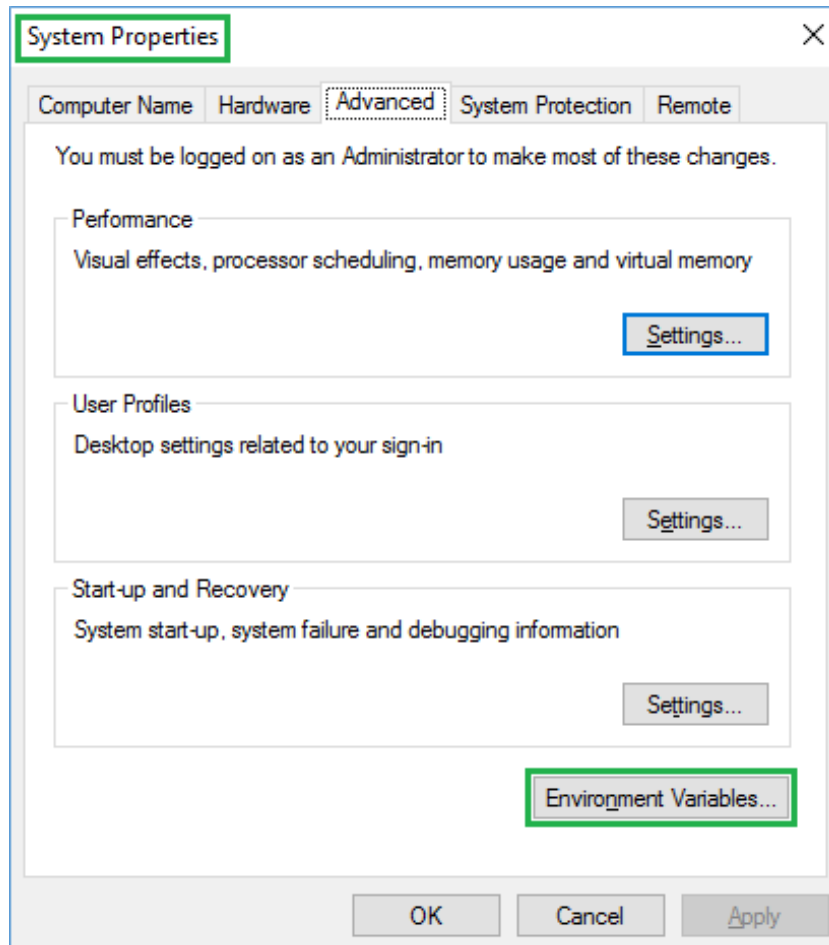


Figure 63. Advance System Properties (Microsoft Windows)

5. In System variables, add a new Java_Home variable and point it to the JDK installed folder.

Configuring Java_Home (Apple macOS)

Follow these steps to configure the Java_Home on Apple's desktop operating system (macOS).

1. Open a terminal window.
2. Check/query the currently set Java_Home with following terminal command:
echo \$JAVA_HOME

3. If Java_Home was not configured already, nothing will be returned from the above command, which is the default behavior in macOS. This can be solved with the following terminal command:
echo "export JAVA_HOME=/usr/libexec/java_home" >> ~/.profile . ~/.profile
4. This will cause JAVA_HOME to be set on start-up (rather than just the current session) and immediately add it.

Execution Modes of JMeter

Depending upon the operating system, JMeter can be invoked via a windows batch file (.bat) on Microsoft Windows operating system. Similarly, for Linux / Unix based systems, shell script (.sh) is used to launch the JMeter.

Following are the three modes the JMeter can be executed:

- i. GUI Mode
- ii. Command Line Mode
- iii. Server Mode

A performance test plan can be created using the GUI mode, while the execution may be using the command line mode (recommended) as well as direct from the GUI.

Troubleshooting DNS Name Change Problem

Following steps were involved to repair the public DNS name problem that results in broken WordPress installation.

No.	Repairing Public DNS Name Change Problem (Terminal Commands)
1	<p><code>curl -O https://raw.githubusercontent.com/wp-cli/builds/gh-pages/phar/wp-cli.phar</code></p> <p>Above command is used to download the wp-cli.</p>
2	<p><code>curl localhost grep wp-content</code></p> <p>Find the URL of the <i>OLD</i> site, can use <i>curl</i> to find it with the above command.</p>
3	<p><code>php wp-cli.phar search-replace 'old_site_url' 'new_site_url' --path=/path/to/wordpress/installation --skip-columns=guid</code></p> <p>Path is WordPress installation directory, (usually <code>/var/www/html</code> or <code>/var/www/html/blog</code>). Use search-replace replace the old reference with the new public DNS name. Following is an example of this procedure.</p> <pre>php wp-cli.phar search-replace ec2-34-241-85-53.eu-west-1.compute.amazonaws.com ec2-34-250-114-30.eu-west-1.compute.amazonaws.com --path=/var/www/html --skip-columns=guid</pre>

After performing this procedure, the WordPress installation should be functional because the broken URL (uniform resource locator) has been updated.

```
ec2-user@ip-172-31-9-146:~ [192x39]
File Edit Settings Plugins Tunnels Help
[ec2-user@ip-172-31-9-146 ~]$ php wp-cli.phar search-replace ec2-34-244-106-199.eu-west-1.compute.amazonaws.com ec2-34-244-88-34.eu-west-1.compute.amazonaws.com --path=/var/www/html --skip-columns=guid
-----+-----+-----+-----+
| Table      | Column      | Replacements | Type |
-----+-----+-----+-----+
| wp_commentmeta | meta_key    | 0            | SQL |
| wp_commentmeta | meta_value  | 0            | SQL |
| wp_comments    | comment_author | 0           | SQL |
| wp_comments    | comment_author_email | 0         | SQL |
| wp_comments    | comment_author_url | 0          | SQL |
| wp_comments    | comment_author_IP | 0          | SQL |
| wp_comments    | comment_content | 0          | SQL |
| wp_comments    | comment_approved | 0          | SQL |
| wp_comments    | comment_agent | 0          | SQL |
| wp_comments    | comment_type | 0          | SQL |
| wp_links      | link_url    | 0            | SQL |
| wp_links      | link_name   | 0            | SQL |
| wp_links      | link_image  | 0            | SQL |
| ...          | ...        | ...         | ... |
| wp_terms     | name       | 0            | SQL |
| wp_terms     | slug       | 0            | SQL |
| wp_usermeta   | meta_key   | 0            | SQL |
| wp_usermeta   | meta_value | 0            | PHP |
| wp_users     | user_login | 0            | SQL |
| wp_users     | user_nicename | 0          | SQL |
| wp_users     | user_email | 0            | SQL |
| wp_users     | user_url   | 0            | SQL |
| wp_users     | user_activation_key | 0        | SQL |
| wp_users     | display_name | 0          | SQL |
-----+-----+-----+-----+
Success: Made 3 replacements.
[ec2-user@ip-172-31-9-146 ~]$
```

The above figure shows the replacement of the public DNS name for the EC2 Instance.