**Software Architecture Document**

**DXC Technology Managed Services For Workloads On Public Cloud**

****

SE 4485.001

Dr. Weichen Wong

***Group 2:***

*Ihfaz Tajwar*

*Zunayed Siddiqui*

*Po-Yu Liu*

*Yeswanth Bogireddy*

*Zachary Tarell*

# Abstract

Service providers are constantly challenged to accept customers' workload as-is and manage it on behalf of the customer. The management can range from monitoring, patching, and security aspects of the workload.

* The security could be shut off certain ports or setup security scans for Denial of Service attack, etc.
* The monitoring could leverage the cloud provider monitoring but would alert based on certain rules
* The patching could use cloud providers specific to patch from vulnerabilities, routine service packs, etc.

Initially, the workloads may be untouched as in refactored ([7 R’s](https://dev.to/wingkwong/7-strategies-for-migrating-applications-to-aws-131d)) to reduce the timelines but over time it might get refactored over time to reduce costs or increase stability/performance

# Table of Contents

[**Abstract**](#_46fnpa3t5y1k) **2**

[**Table of Contents**](#_8ebvkubztzvy) **3**

[**List of Figures**](#_l5yze7h7le8s) **4**

[**List of Tables**](#_hdxmx63d2ue8) **4**

[**Introduction**](#_6vg3jevz44yf) **4**

[**Architecture**](#_b5nghmjuaeg) **5**

[**Technology, Software, and Hardware Used**](#_df93enksq2xt) **6**

[**References**](#_ny7t4utzy5fr) **6**

# List of Figures

Figure 1. Client-Server Architecture

# List of Tables

N/A

# Introduction

Our team has been tasked with implementing a makeManaged script for DXC Technology to monitor, make patchwork, and manage security aspects of their workload. This project management plan will include all of the aspects from beginning to end on how our team will create and implement DXC Technology’s goal from our script. Even though the requirements list out three different phases of this project, our sponsor has made clear that the minimum viable product will only need to contain the monitoring aspect of our project and if time permits move onto patchwork as well as security, load balancing, and firewall monitoring.

**PURPOSE AND SCOPE**

The purpose of monitoring software through Infrastructure as a System (IaaS) is to benefit many different companies that require help desk support, patchwork, and security. Since SaaS and PaaS are already managed by cloud providers, we will only be using IaaS. The scope will range from a working monitoring script able to communicate with a cloud service and relay information, notifications, and patchwork to DXC Technology. This will lighten the workload of their employees by automating a lot of tasks and opening up free time to allocate resources elsewhere.

**OVERVIEW**

A brief overview of the makeManaged program will be to use a virtual machine (VM) to communicate to AWS, Azure, and/or GCP cloud services over IaaS to monitor and the help desk and send notifications and email alerts or patch any necessary fixes.

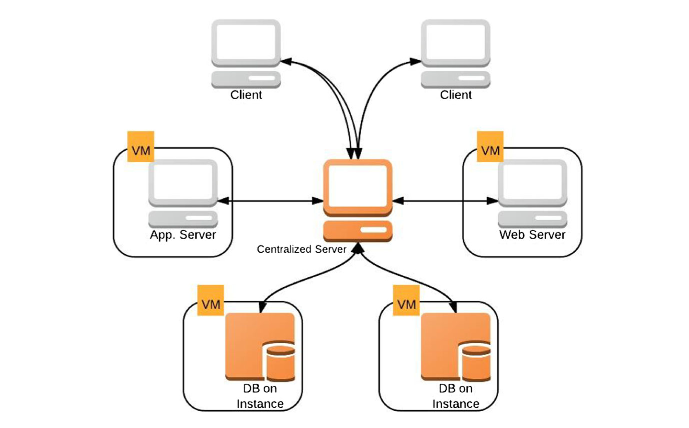
# Architecture

ARCHITECTURAL STYLE(S) USED

We will be using a client-server architecture because of the event listener feature the monitoring tool will feature. Once events occur within the cloud provider, it will be received by our monitoring tool which will then send appropriate responses to ServiceNow

ARCHITECTURAL MODEL

*Figure 1. Client-Server Architecture*



# 

## Technology, Software, and Hardware Used

Our project is fully cloud based which means we would not be requiring any hardware other than a computer and an internet connection for the programmer and the user to use our services.

**Software**

* Jira testing
* VM - AWS
* GitHub
* JAVA - C
* Ec2 or Google
* Monitoring tool - Zabbix
* Nagios
* Postman for API calls
* Swagger - for API calling that is connected to spring boot and we’ll also have a UI
* Spring boot
* Jenkins - deploying
* JDK/JRE
* IntelliJ
* Maven - dependency management

## Rationale

Based on our conversation with our sponsor, we determined that this model would best fit our needs in monitoring the cloud providers. The client can use this core architecture to add more features like reporting and patching as needed. This model exemplifies usability, extensibility, and availability, therefore was the best one we could choose.

# Evidence

B

# References

“An overview of the Commercial Cloud Monitoring Tools: Research Dimensions, Design Issues, State-of-the-Art.” <https://my.ece.msstate.edu/faculty/skhan/pub/A_K_2014_COMP.pdf>