# A PRELIMINARY REPORT ON

**INTELLIGENT INFRASTRUCTURE ALERTS**

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE

OF

**BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)**

SPONSORED BY



##### SUBMITTED BY

**TEJASHREE DADABHAU SHETE** **EXAM NO.:** **B150204380**

**SHARAYU RAJARAM SHINDE EXAM NO.: B150204382**

**VIDYA VISHNU SONAWANE EXAM NO.: B150204394**

**RASHMI ARVIND VAGHA EXAM NO.: B150204400**



## DEPARTMENT OF COMPUTER ENGINEERING

## Cummins College of Engineering for Women, Pune

**KARVENAGAR, PUNE-411052**

## SAVITRIBAI PHULE PUNE UNIVERSITY

## 2018 -2019



**CERTIFICATE**

This is to certify that the project report entitles

**INTELLIGENT INFRASTRUCTURE ALERTS**

Submitted by

**TEJASHREE DADABHAU SHETE EXAM NO. : B150204380**

**SHARAYU RAJARAM SHINDE EXAM NO. : B150204382**

**VIDYA VISHNU SONAWANE EXAM NO. : B150204394**

**RASHMI ARVIND VAGHA EXAM NO. : B150204400**

are bonafide students of this institute and the work has been carried out by them under the supervision of **Prof. S.P.MENGALE** and it is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University, for the award of the degree of **Bachelor of Engineering** (Computer Engineering).

**(Prof. S. P. MENGALE)** **(Dr. Supriya Kelkar)**

Guide Head,

Department of Computer Engineering Department of Computer Engineering

**(Dr. M. B. Khambete)**

Director, Cummins College of Engineering for Women Pune – 52

Place: Pune

Date:

**ACKNOWLEDGEMENT**

It gives us great pleasure in presenting the preliminary project on

**Intelligent Infrastructure Alerts**

We would like to take this opportunity to thank our internal guide ***Prof. S. P. Mengale*** for giving us all the help and guidance we needed. We are really grateful to him for his kind support. His valuable suggestions were very helpful.

We are also grateful to ***Prof. S. M. Kelkar,*** Head of Computer Engineering Department, Cummins College of Engineering for Women for her indispensable support, suggestions.

In the end our special thanks to our external guide, ***Mr Rahul Nema*** from IBM India Pvt. Ltd. for providing his valuable time, support, guidance and resources.

Tejashree Shete

Sharayu Shinde

Vidya Sonawane

Rashmi Vagha

(BE Computer Engineering)

**ABSTRACT**

Nowadays Cloud Computing is used in various domains. It is a technology which uses the internet and central remote servers to maintain data and applications. **Cloud computing** allows consumers and businesses to use applications without installation and access their personal files at any computer with internet access.

One of the key elements of Cloud Computing is the deployment model. Deployment is the process of making software available and ready for use. Once the application is deployed using IBM Cloud Automation Manager, it may encounter multiple issues like low storage, thrashing, crashing of containers, excess CPU utilization etc. Currently these events are not communicated properly to user. Irrelevant information about events and false alarms will take the user off track.

Hence we proposed a system that aims to capture the events and communicate them back to user. On capturing the events, they also need to be prioritized. This can be done using IBM Cloud Event Management. It helps to know which is the biggest problem that needs to be solved first. A conversational BOT is built using IBM Watson API to publish errors and warnings to the user. It enables your DevOps and IT Operations teams to resolve application, service and infrastructure problems promptly.

**TABLE OF CONTENTS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr.No.** | | | **Title of Chapter** | | **Page No.** |
| 01 | | | **Introduction** | | 8 |
|  | 1.1 | | Motivation | | 9 |
|  | 1.2 | | Problem Definition | | 10 |
| 02 | | | Literature Survey | | 11 |
| 03 | | | Software Requirements Specification | | 13 |
|  | | 3.1 | **Introduction** | | 14 |
|  | |  | 3.1.1 | Project Scope | 14 |
|  | |  | 3.1.2 | Assumptions and Dependencies | 14 |
|  | | 3.2 | Functional Requirements | | 15 |
|  | |  | 3.2.1 | System Feature 1(Functional Requirement) | 15 |
|  | |  | 3.2.2 | System Feature2 (Functional Requirement) | 15 |
|  | | 3.3 | External Interface Requirements (If Any) | | 17 |
|  | |  | 3.3.1 | User Interfaces | 17 |
|  | |  | 3.3.2 | Software Interfaces | 18 |
|  | |  | 3.3.3 | Communication Interfaces | 19 |
|  | | 3.4 | Nonfunctional Requirements | | 21 |
|  | |  | 3.4.1 | Performance Requirements | 21 |
|  | |  | 3.4.2 | Security Requirements | 21 |
|  | |  | 3.4.3 | Software Quality Attributes | 21 |
|  | | 3.5 | **System Requirements** | | 22 |
|  | |  | 3.5.1 | Database Requirements | 22 |
|  | |  | 3.5.2 | Software Requirements(Platform Choice) | 22 |
|  | |  | 3.5.3 | Hardware Requirements | 23 |
|  | | 3.6 | Analysis Models: SDLC Model to be applied | | 24 |
|  | | 3.7 | Implementation Plan | | 25 |
| 04 | | | System Design | | 26 |
|  | | 4.1 | **MVC Architecture** | |  |
|  | | 4.2 | **UML Diagrams** | |  |
|  | |  | 4.2.1 Data Flow Diagram Level-0 | | 27 |
|  | |  | 4.2.2 Data Flow Diagram Level-1 | | 28 |
|  | |  | 4.2.3 Data Flow Diagram Level-2 | | 29 |
|  | |  | 4.2.4 Use Case Diagram  4.2.5 Sequence Diagram  4.2.6 Deployment Diagram  4.2.7 Activity Diagram | | 30  31  32  33 |
| 05 | | | Other Specification | | 34 |
|  | | 5.1 | Advantages | | 35 |
|  | | 5.2 | Applications | | 36 |
| 06  6.1  6.2 | | | **Conclusions & Future Work**  Conclusion  Future Work  **References** | | 40  41  41 |
|  | | | **Glossary** | | 43 |
|  | | | **Appendix- A** | |  |
|  | | | **Appendix- B** | |  |

|  |  |
| --- | --- |
|  | **LIST OF FIGURES** |
|  |  |
|  |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| **Figure** | **Illustration** | **Page No.** |
|  | | |
| 3.1  4.1  4.2.1 | Analysis Model  MVC Architecture  Data Flow Diagram Level-0 | 28 |
| 4.2.2 | Data Flow Diagram Level-1 | 29 |
| 4.2.3 | Data Flow Diagram Level-2 | 30 |
| 4.2.4 | Use Case Diagram | 31 |
| 4.2.5 | Sequence Diagram | 32 |
| 4.2.6  4.2.7 | Deployment Diagram  Activity Diagram | 33  34 |

**CHAPTER 1**

**INTRODUCTION**

**1.1 Motivation**

DevOps (Developers + Operations) deploy and operate the applications. They also troubleshoot and resolve issues in dev, test and production environments. DevOps teams want to be more self-sufficient at managing the incidents when application performance is poor or negatively impacted and get distracted by incomplete or irrelevant alert information.

After the application is deployed, various incidents like low storage, excess CPU utilization, thrashing, crashing of containers, etc. Due to low storage and crashing of containers application may get slowed down and leads to service outage. CPU can get damaged due to excess utilization.

If these events are not properly communicated to DevOps team and administrator, then they won’t able to identify which event has most impact on the system. It leads to more issues if they are not solved immediately. Also it may happen that they spend more time on resolving issues which have least priority.

The endeavour of the project is to capture events with the help of IBM Cloud Event Management (CEM), convey these events to DevOps team through IBM Watson and manage them with the help of IBM Cloud Automation Manager (CAM).

**1.2 Problem Definition**

IBM Cloud Automation Manager (CAM) is a cloud management solution on IBM Cloud Private (ICP) for deploying cloud infrastructure with an optimized user experience. This infrastructure can be present across different cloud providers.

CAM helps to create and edit templates and services that implement common business patterns and to deploy them in the cloud environment.

The provisioned services include deployment of micro services Kubernetes containers and virtual machines across multiple cloud providers.  Post deployment, users can manage and access the instances from the CAM user interface.

The provisioned VM's and micro-service containers over their lifetime can encounter multiple issues e.g. low storage, thrashing, crashing of containers etc. The provisioned VM’s and containers publish different events and health status periodically. The endeavour of the proposed project is two-fold:

1. Leveraging IBM Cloud Event Manager to capture these events and direct them to deployed service instances of CAM
2. Develop a conversation bot by leveraging IBM Watson API's that can process natural language queries regarding the health status of the provisioned VM's/ containers, publishing errors and warnings to the user and respond to different events received from the event manager.

**CHAPTER 2**

**LITERATURE SURVEY**

Cloud computing is one of the most useful and widespread technologies. As mentioned in the IEEE paper **“Application Deployment and Management in the Cloud”** by Marco Miglierina cloud computing is a revolutionary paradigm that allows to acquire inﬁnite resources on demand and pay only for the actual use. The cloud can really help companies in responding to market demand where customers, developers and system administrators are almost conﬂicting among each other where the shared ﬁnal objective is that of making the product available to the user. One of the main challenges in cloud computing is of the heterogeneity of the offer. More and more cloud services are growing from different providers with no standard yet deﬁned. Customers are not interested in the product itself anymore; rather they look at the services delivered together with the product. Today, as we will see, in successful companies the key word is continuous delivery. The business environment is forcing companies to shift their focus from stability and efficiency to agility and innovation. If they do not keep up with today’s fast changing demand, by shortening work cycles and being creative, they can disappear in few years. Cloud computing is the direct response to the need for agility. IT companies can now focus on adding business value to their products, without thinking about hardware provisioning and management.

According to **“Cloud application deployment with transient failure recovery”** by Ioannis Giannakopoulos, Ioannis Konstantinou, Dimitrios Tsoumakos and Nectarios Koziris application deployment is a crucial operation for modern cloud providers. The ability to dynamically allocate resources and deploy a new application instance based on a user-provided description in a fully automated manner is of great importance for the cloud users as it facilitates the generation of fully reproducible application environments with minimum effort. However, most modern deployment solutions do not consider the error-prone nature of the cloud. Network glitches, bad synchronization between different services and other software or infrastructure related failures with transient characteristics are frequently encountered. Even if these failures may be tolerable during an application’s lifetime, during the deployment phase they can cause severe errors and lead it to failure.

In the book **“Cloud Application Architectures”** by George Reese, it is said that, disaster recovery is the practice of making a system capable of surviving unexpected or extraordinary failures. A disaster recovery plan, for example, will help your IT systems survive a fire in your data center that destroys all of the servers in that data center and the systems they support. For that, monitoring your cloud infrastructure is extremely important. You cannot replace a failing server or execute your disaster recovery plan if you don’t know that there has been a failure. The trick, however, is that your monitoring systems cannot live in either your primary or secondary cloud provider’s infrastructure. They must be independent of your clouds. To complete the disaster recovery scenario, you need to recognize when a disaster has happened and have the tools and processes in place to execute your recovery plan.

From the book **“The Enterprise Cloud”** by James Bond**,** it is evident that the information every Chief Information Officer needs in order to make the business and technical decisions to finally execute on their journey to cloud computing is getting updated trends and definitions in cloud computing, deployment models, and for building or buying cloud services. Also, discovering challenges in cloud operations and management not foreseen by early adopters as well as identifying security threats and vulnerabilities unique to the cloud and employing a cloud management system for your enterprise (private or multi-provider hybrid) cloud ecosystem are of utmost importance.

The official IBM Cloud Automation Manager website <https://www.ibm.com/in-en/marketplace/cognitive-automation> provides information regarding Cloud Automation Manager. Also, it gives a brief description of how using IBM Cloud Automation Manager can benefit the business by empowering developers and administrators to meet business demands. This site also contains the features of IBM CAM which are self-service, governance and control and automated provisioning.

The website referred for IBM Cloud Event Management is https://www.ibm.com/in-en/marketplace/cloud-event-management. It contains information as to how Cloud Event Management automatically correlates events into prioritized incident views. It also notifies the right person at the right time, with integrated, automated notifications. This initiates a fast response and keeps everyone in sync. In addition to this, it maintains runbooks which are used to resolve similar events which may occur in the future. Another advantage of Cloud Event Management is that it provides a Centralized Knowledge Base for DevOps team.

In order to communicate the captured events back to the user, one of the possible alternatives is using a chat bot. To develop this chat bot, IBM has developed a tool namely IBM Watson. As mentioned on the website for IBM Watson which is <https://www.ibm.com/watson/about/index.html>, powered by the latest innovations in machine learning, Watson lets one learn more with less data.  One can integrate AI into the most important business processes, informed by IBM’s rich industry expertise. Also, one can build models from scratch, or leverage APIs and pre-trained business solutions.

**CHAPTER 3**

**SOFTWARE REQUIREMENT SPECIFICATION**

**3.1 Introduction**

**3.1.1 Project scope**

The endeavour of the proposed project is two-fold:

1. Leveraging IBM Cloud Event Manager to capture and prioritize these events and direct them to deployed service instances of CAM.

This module aims to capture the events. These events are given as input to CAM. Further, they are filtered and prioritized by CEM so as to decide which event must be worked on first.

1. Develop a conversation bot by leveraging IBM Watson API's that can process natural language queries regarding the health status of the provisioned VM's/ containers.

Bot can convey the errors and warnings to the user using and respond to different events received from the event manager.

The proposed project can only publish errors and warnings to the user and respond to different events received from the event manager. Currently, these errors are not analyzed and upcoming failure conditions are not predicted.

**3.1.2 Assumptions and dependencies**

1. User should have an account on IBM Bluemix (IBM Cloud Platform) to operate IBM Cloud Event Management.
2. The applications might be deployed on different cloud platforms like AWS, Google Cloud Platform, Microsoft Azure, etc.
3. The administrators and DevOps teams which will be using this system should have necessary skills to operate CAM user interface.
4. User should have an active internet connection.
5. The proposed project will be integrated in the existing IBM’s Cloud Automation Manager.

**3.2 Functional Requirements:**

**3.2.1 System Features Module 1 (Functional Requirement)**

1. Quick problem determination:

Receive thousands of events from various sources and consolidate them into prioritized incidents that tell user what is really causing faults in the application environment.

1. Bridge the gap between Developers and Operations teams:

Relieve pressure on teams by automatically routing incidents to the right people and bringing together Development and Operations teams to resolve complex faults fast.

1. Restore service quickly:

Using the alerts and warnings conveyed by chat bot, immediate actions will be taken by the DevOps team to resolve the issues and restore the service quickly.

**3.2.2 System Features Module 2 (Functional Requirement)**

1. Natural Language Processing:

Analyze text to extract meta-data from content such as concepts, entities, keywords, categories, relations, semantic roles using natural language understanding. With custom annotation models developed using Watson Knowledge Studio, identify industry/domain specific entities and relations in unstructured text.

1. Communicating events to user:

Chat bot conveys the health status of the provisioned Virtual machines, containers along with errors and warnings to the user and responds to different events received from the event manager.

# 3.3 External Interface Requirements

**3.3.1 User Interfaces**

* IBM Watson:
* **Features:**
* **Simple and easy-to-use interface**

This interface allows building bot quickly and reliably.

* **Connects to messaging channels**

Easily integrate with multiple social and messaging channels.

* **Knows what your bot is missing**

Watson recommends what to train next, based on frequently asked but unanswered questions.

* **Provide complex responses**

Address more questions with answers previously hidden in your documentation by integrating with Watson Discovery Service.

* **Services:**
* **Watson Assistant**

Watson Assistant is an offering for building conversational interfaces into any application or device. Most chat bots try to mimic human interactions, which can frustrate users when a misunderstanding arises. Watson Assistant is more intelligent in such cases. It knows when to search for an answer from a knowledge base, when to ask for clarity, and when to direct you to a human.

* **Watson Discovery**

Watson Discovery Service enables you to rapidly work on your unstructured data. Discovery implies the infrastructure, scale and algorithm challenges associated with analyzing large datasets.

**3.3.2 Software Interfaces**

* **IBM Cloud Private**:

IBM CloudPrivate is an application platform for developing and managing on-premises, containerized applications. It is an integrated environment for managing containers that include the container orchestrator (Kubernetes), a private image repository, a management console, and monitoring frameworks.

IBM Cloud Private is a reliable and scalable cloud platform that runs on your infrastructure. It’s built on open-source frameworks, with common services for self-service deployment, monitoring, logging and security.

With IBM Cloud Private, development and administrative teams share a flexible cloud environment to create new micro-service based applications, modernize existing apps and securely integrate between the two applications. IBM Cloud Private complements the IBM Cloud by providing consistent runtimes, services and management capabilities.

* **IBM Bluemix:**

It is a cloud [platform as a service (PaaS)](https://en.wikipedia.org/wiki/Platform_as_a_service) developed by [IBM](https://en.wikipedia.org/wiki/IBM). It supports several [programming languages](https://en.wikipedia.org/wiki/Programming_language) and servicesas well as integrated [DevOps](https://en.wikipedia.org/wiki/DevOps) to build, run, deploy and manage applications on the cloud. Bluemix runs on [ICP](https://en.wikipedia.org/wiki/SoftLayer) infrastructure.

* **IBM Event Management:**

IBM Cloud Event Management automatically correlates events into prioritized incidents. It also notifies administrator at the appropriate time, with integrated, automated notifications. This leads to a fast response and keeps everyone in sync. Previously encountered errors and their solutions are recorded in runbooks. Runbooks contain structured, manual and automated steps to help resolve the underlying problems described in incidents. We can create our own custom runbooks and manage existing catalog of runbooks.

These runbooks are referred to recommend probable solutions which might help in resolving incidents quickly. This is one of the most important guideline for the administrator.

**3.3.3 Communication Interfaces**

* **Representational State Transfer** (**REST**) **API**

A REST API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data. It also referred to as a RESTful web service which is based on representational state transfer (REST) technology, an architectural style and approach to communicate. It is often used in web services development.

**3.4 Non-functional Requirements:**

**3.4.1 Performance Requirements**

System should be able to scale to get alerts across multiple cloud providers and virtual machines.

**3.4.2 Security Requirements**

The credentials to access the machines and the logs will be communicated over a secure network.

**3.4.3 Software Quality Attributes**

* **Agility**: To build a scalable and highly reliable monitoring system that works across tens of thousands of containers.
* **Compatibility**: The system should compatible with any cloud provider and VM.
* **Configurability**: Chef is a configuration management tool which is integrated in CAM, responsible to deploy and configure simple and complex software from templates (also called cookbooks).
* **Self-service:** Provide self-service access for developers and the ability to automate the application lifecycle using prebuilt automation packs and communicate the same through Watson API.
* **Governance and control**: Maintain control through effective governance and intelligent insights for a security-rich and compliant IT environment in collaboration with the user by communicating through chat bot.

**3.5 System Requirements:**

**3.5.1 Database Requirements**

* **MongoDB:**

It is an open source NoSQL database which is used to store unstructured data.The logs and information related to events is stored in database.

**3.5.2 Software Requirements (Platform choice)**

* **Ubuntu:**

Ubuntu is Linux based free and open source operating system.

Version: 16.04

* **Docker:**

A tool designed to make it easier to create, deploy, and run applications by using containers.

Version: 17.12

* **Kubernetes:**

Kubernetes is an open-source container-orchestration system for automating deployment, scaling and management of containerized applications.

* **JavaScript:**

JavaScript is a programming language designed for creating network-centric applications.

* **Node Js:**

Node.js is a platform for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

* **Loopback:**

It is an open source Node.js framework that enables you to quickly create the dynamic end to end REST APIs and connect to backend systems such as databases, REST services.

**3.5.3 Hardware Requirements**

**Servers**:

Intel® Xeon(R) CPU E5-2620 0 @ 2.00 GHz x 24

Disk size- 586 GB

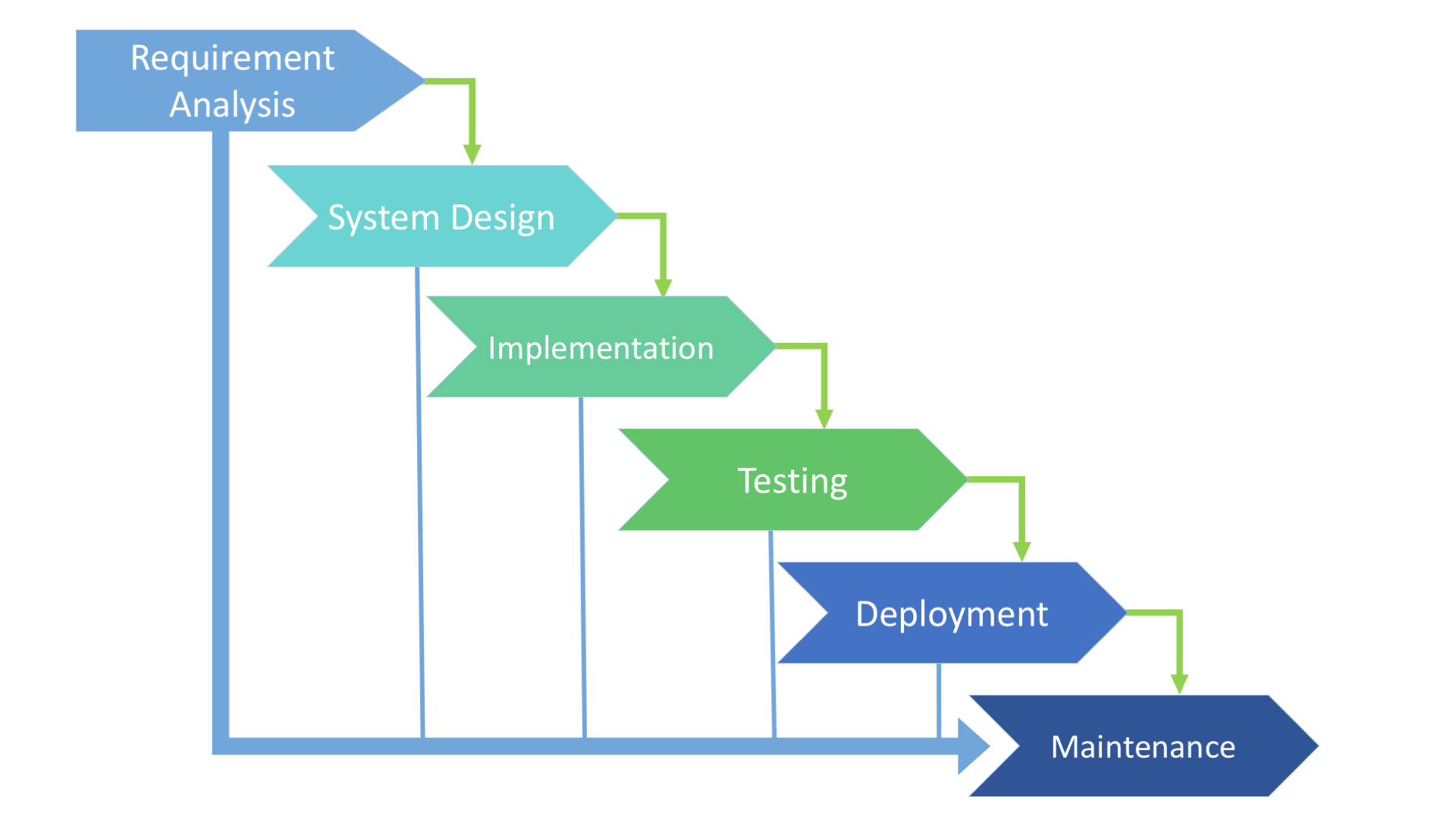
**3.6 Analysis Model: SDLC Model to be applied**

**Iterative Model**:

In the **Iterative model**, **iterative** process starts with a simple implementation of a small set of the software requirements and **iteratively** enhances the evolving versions until the complete system is implemented and ready to be deployed.

For our project we intend to start with managing applications deployed on AWS and ICP.

Further increments shall include integration with multiple cloud service providers. After that, events on VMs can also be managed.



**Figure: 3.1**

**3.7 Implementation Plan:**

Major tasks in the project

1. Selection of project statement :

This includes the selection of the problem statement, deciding the objective of the project and literature review, getting the clear idea of the project, contribution of the project towards society, clear objective of the project.

1. Feasibility and Scope :

This task includes end users of the project and the projected cost of the product.

This also includes the time required for the project. It means whether the project can be completed in the given time. Prioritization of the risks and the backup plan for the risks are decided.

1. Requirement Analysis :

It includes the complete domain information, consistent requirements of the project along with resources budgets and schedule is clearly defined in the requirement analysis task. And whether all the requirements specified are traceable by the system or not is also decided.

1. Design :

Reflection of the requirements in the architecture of the system does system addresses all the issues from the requirements, design support of the system both product as well as project goals. All the structural diagrams, sequence diagrams, behavioural diagrams are must be clear in the task

1. Modeling:

The software development process model to be used is decided, clear matching of the objects and their attributes, clear analysis of the project requirement is done in the task.

Functional dependencies, NP-completeness of the problem and interface between modules is made clear in this task.

1. Implementation :

Implementation task includes the actual development i.e. the coding part of the project. It contains complete and correct implementation of the design, well-structured code. All functions of the software should be coded, correct language for coding should be selected and finally the code should be optimized.

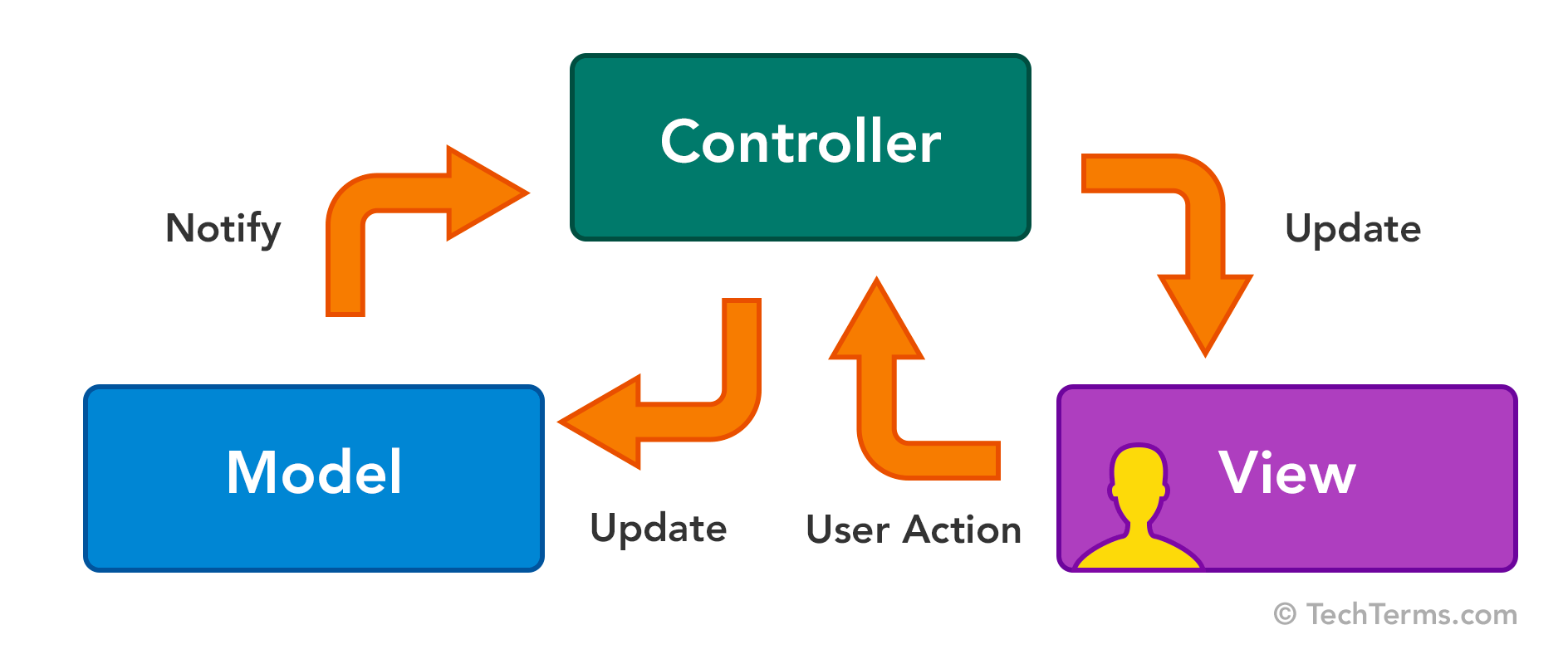
**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 Model-View-Controller (MVC) Architecture:**

The proposed system is designed by following MVC architecture. The Model-View-Controller is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components is built to handle specific development aspects of an application.

## MVC Components:



**Figure: 4.1**

### **Model**

### The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. The logs and information related to events is extracted and relevant data is stored in database.

### **View**

### The View component is used for all the UI logic of the application. In this system, the conversational chat bot will convey health status of the system to user in order to take appropriate actions.

### **Controller**

### Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. IBM Cloud Event Manager acts as a controller for the proposed system. It interacts with IBM Cloud Automation Manager to manage the events.

**4.2 UML Diagrams:**

**4.2.1 Dataflow diagram Level 0**

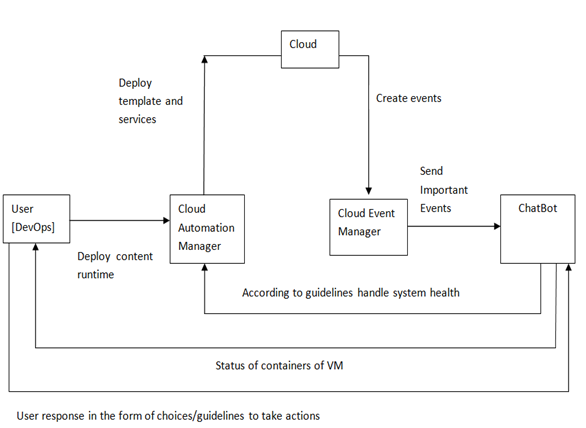


Fig. 4.2.1 – Dataflow diagram level 0

**4.2.2 Dataflow diagram Level 1**

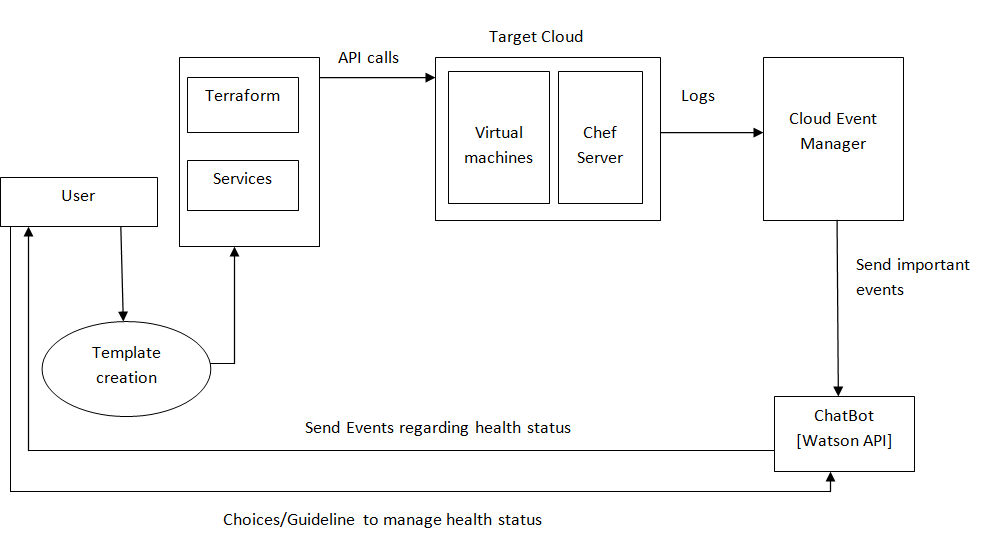


Fig. 4.2.1 – Dataflow diagram level 1

**4.2.3 Dataflow diagram Level 2**

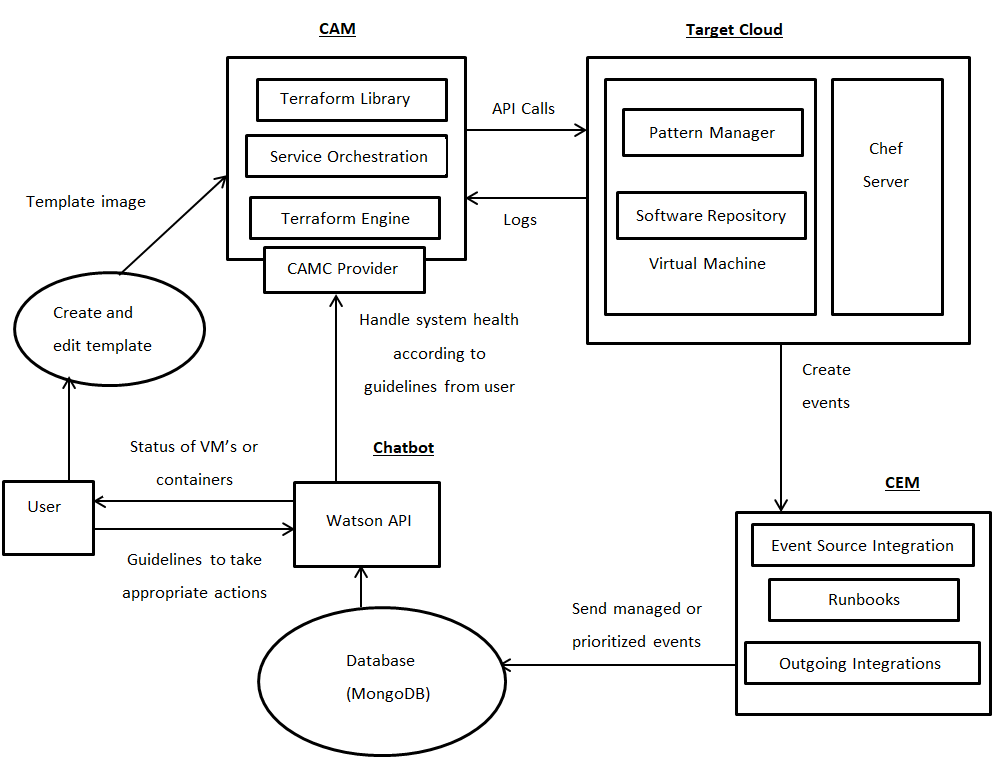


Fig. 4.2.1 – Dataflow diagram level 2

**4.2.4 Use case diagram**

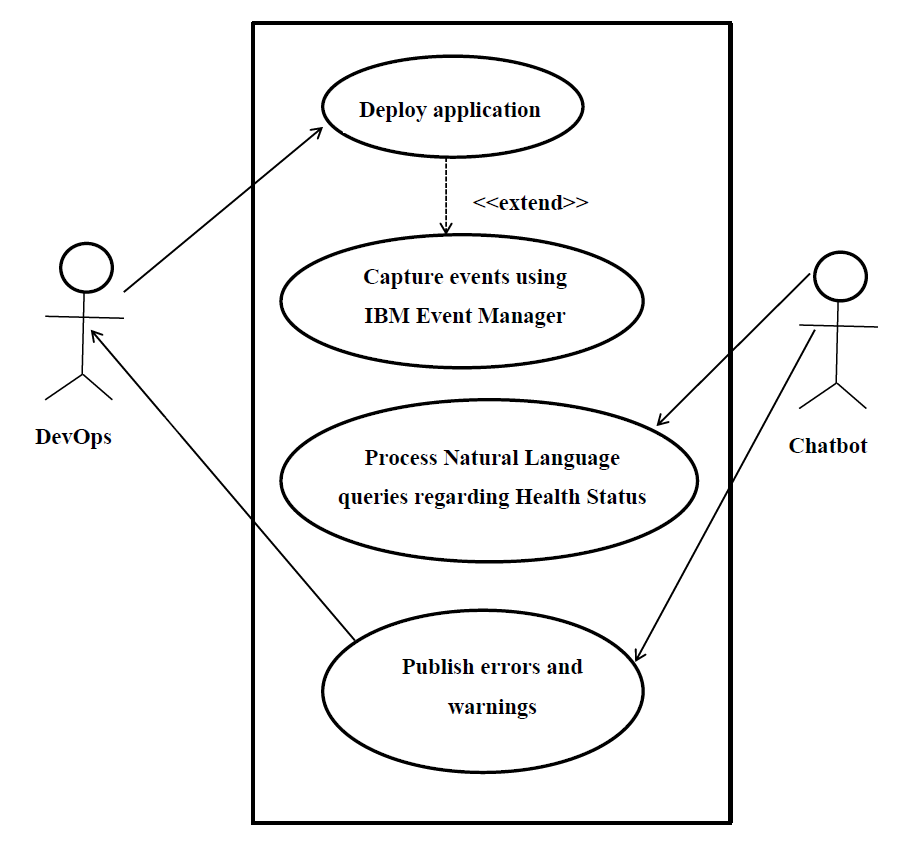


Fig. 4.2.1 – Use case diagram

**4.2.5. Sequence diagram**

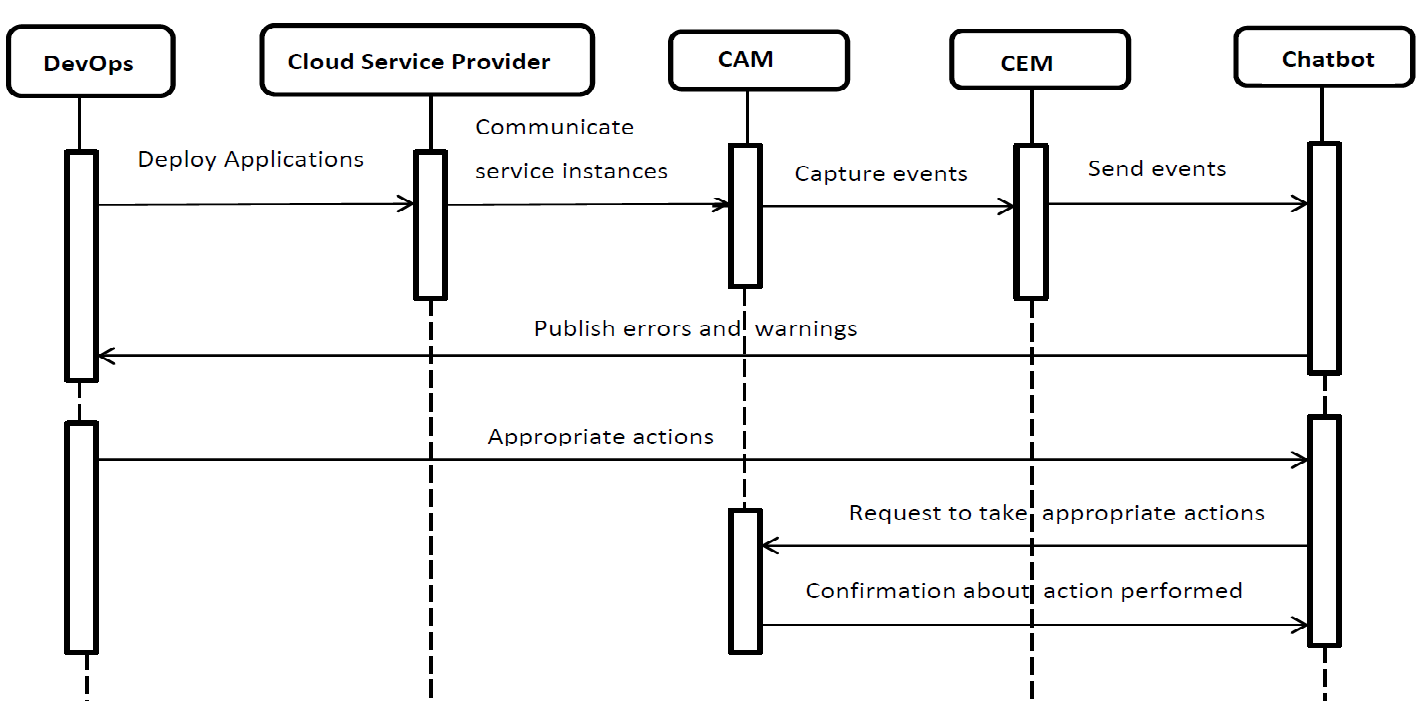


Fig. 4.2.1 – Sequence diagram

**4.2.6. Deployment diagram**

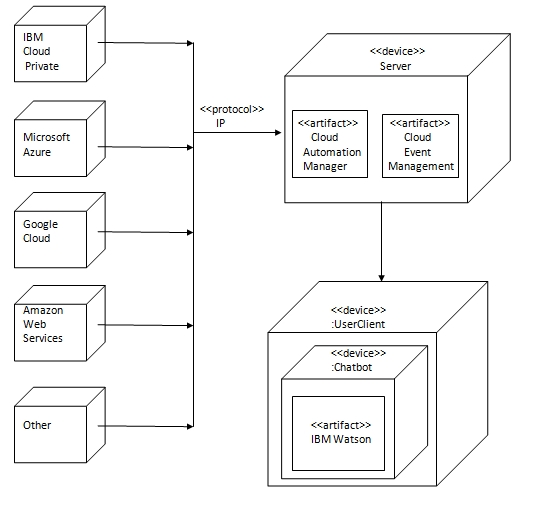
****

Fig. 4.2.1 – Deployment diagram

**4.2.7. Activity diagram**

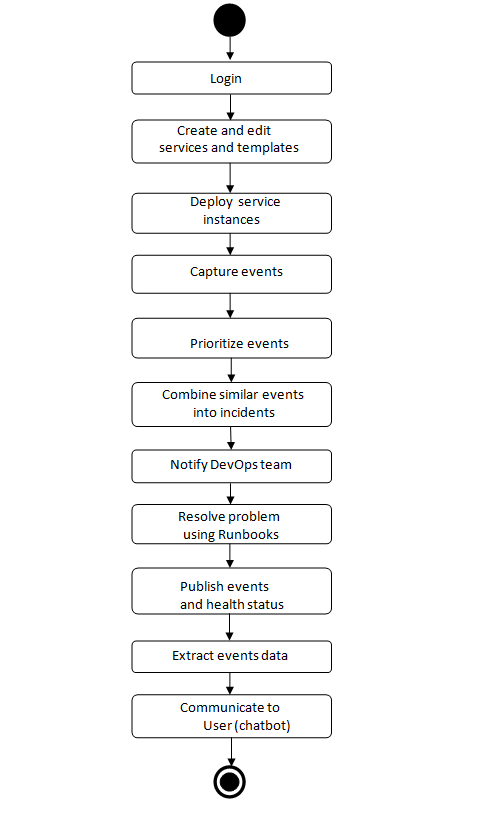
****

Fig. 4.2.1 – Activity diagram

|  |  |
| --- | --- |
|  |  |

**CHAPTER 5**

**OTHER SPECIFICATION**

**5.1 Advantages:**

1. It helps in **identifying problems** from multiple events from different sources according to publish – subscribe model and see how the events are related and communicate it to the administrator.
2. There may be different events that are reported. End-users are notified regarding **prioritizes incidents** and thus guides to focus on critical incidents.
3. It provides **access to runbooks** which contains solutions to solve complex and routine problems faster and communicate it to the administrator.
4. It gives the facility of having a centralized knowledge base for the DevOps team to resolve issues.

**5.2 Applications:**

This project can be used in multiple situations when a company needs management of the deployed application. The end users can be app programmers, web programmers, database admins, micro-service architects as well as service composers or configuration managers.

Few of the scenarios where this project can be used are as follows.

* **Thrashing**:

Thrashing in computing is an issue caused when virtual memory is in use. It occurs when the virtual memory of a computer is rapidly exchanging data for data on hard disk. As the main memory gets filled, additional pages need to be swapped in and out of virtual memory. The swapping causes a very high rate of hard disk access. Thrashing can potentially result in total collapse of the hard drive of the computer.

So, this project aims to-

1. Capture the thrashing events with help of CEM.
2. According to rules set threshold value for segregating the events.
3. Store the event details in database.
4. Notify these events to the administrator through chat bot.

Thus, it helps to avoid losses and prevent the application from going down and ultimately improves business value.

* **CPU utilization**:

CPU utilization refers to a computer's usage of processing resources, or the amount of work handled by a CPU. Actual CPU utilization varies depending on the amount and type of managed computing tasks. Certain tasks require heavy CPU time, while others require less because of non-CPU resource requirements. If CPU utilization exceeds a given limit, it may cause severe damage.

So, this project aims to:

1. Capture the incidences of excess CPU utilization.
2. Check with the threshold rules.
3. If CPU utilization is more, then store in database.
4. Convey to the administrator the details regarding the event.

Thus, it helps administrative teams to take preventive measures and satisfy their customer’s needs.

* **Crashing of containers:**

Components virtualize applications in specific containers. They create an isolation boundary at the application level. A crash, in the context of computing, is an event wherein the computer application stops functioning properly. It mostly occurs when:

* Hardware has failed in a non-recoverable fashion
* Operating system data have become corrupted
* Recovery from an error is not possible without loss of data

An application crash can result in an unexpected exit from the application.

So, this project helps to:-

1. Capture events regarding unexpected exit from application.
2. Determine its severity with the help of predefined rules and threshold.
3. If severe, convey it to the administrator.

Thus, it helps to identify the crashed container and accordingly proceed further to take preventive measures.

**CHAPTER 6**

**CONCLUSION AND FUTURE WORK**

**6.1 Conclusion**

The proposed project empowers developers and administrators to meet business demands. This platform allows efficient management and delivery of services through end-to-end automation while enabling developers to build applications aligned with enterprise policies. Using IBM Watson, developers can optimize their landscapes within minutes. They can achieve flexibility, gain speed through self-service access and maintain control through effective, enforceable governance and intelligent insights to ensure a safe and compliant IT environment.

**6.2 Future Work**

Certain elements in this project leave scope for further development. The proposed project can only publish errors and warnings to the user and respond to different events received from the event manager. Currently, these errors are not analyzed.

The future plan is to come up with a better technique of running analytics on the event data to derive patterns of meaningful insights. A failure prediction based on predictive analytics models can identify an upcoming failure condition in advance. Then, based on this detection, a proactive prescriptive action can be taken.

**REFERENCES**

* IEEE Papers –
* Marco Miglierina, “Application Deployment and Management in the Cloud”, 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 2014
* Ioannis Giannakopoulos , Ioannis Konstantinou, Dimitrios Tsoumakos and Nectarios Koziris, “Cloud application deployment with transient failure recovery”, Journal of Cloud Computing: Advances, Systems and Applications, 2018
* Books –
* George Reese, Cloud Application Architectures, 2009, O'Reilly Media, ISBN: 9780596157647
* James Bond, The Enterprise Cloud, 2015, O'Reilly Media, ISBN: 9781491907832
* Websites –
* <https://www.ibm.com/in-en/marketplace/cognitive-automation>
* <https://searchcloudcomputing.techtarget.com/definition/cloud-automation>
* <https://www.ibm.com/in-en/marketplace/cloud-event-management>
* <https://www.marutitech.com/advantages-of-natural-language-generation/>
* <https://www.ibm.com/watson/about/index.html>
* <https://developer.ibm.com/recipes/tutorials/integration-of-ibm-watson-conversation-service-to-your-java-application/>

**GLOSSARY**

1. Agility:

It refers to the effective and rapid response to change yielding rapid and incremental delivery of software.

1. Application Programming Interface (API):

It is a set of functions and procedures that allow the creation of applications which access the features or data of an operating system, application or other service.

1. Amazon Web Services (AWS):

Amazon Web Services is a subsidiary of Amazon.com that provides on-demand cloud computing platforms to individuals, companies and governments, on a paid subscription basis.

1. Bot

A bot is an automated program that runs over the internet, Some bots run automatically, while others only execute commands when they receive specific inputs.

1. Cloud Automation Manager (CAM):

It is a multi-cloud, self-service management platform that empowers developers and administrators to meet business demands through end-to-end automation.

1. Cloud Event Management (CEM):

It provides a consolidated view of problems that occur with your services, applications and infrastructures so that you can resolve the problems more efficiently.

1. Chef :

It is a tool that can streamline the task of configuring and maintaining a company’s servers and integrating with cloud- based platforms.

1. Containers:

They create a unique virtual space called a “sandbox” that separates an application from others in the same environment.

1. DevOps:

It is a software development methodology that combines software development (*Dev*) with information technology operations (*Ops*).

1. IBM Cloud Private (ICP):

It is an application platform for developing and managing on-premises, containerized applications.

1. Machine learning:

It is application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.

1. Micro-services:

A distinctive method of developing software systems those try to focus on building single function modules with well-defined interfaces and operations.

1. Model-View-Controller (MVC):

The Model-View-Controller (**MVC**) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller.

1. Orchestration:

It describes automated arrangement, coordination, and management of complex computer systems and services.

1. Runbooks:

It is a compilation of routine procedures and operations that the system administrator or operator carries out.

1. Terraform:

It enables to safely and predictably create, change and improve infrastructure by codifying APIs which can be shared amongst team members.

1. Virtual Machine (VM):

It is an emulation of a computer system that provides functionality of physical computer.

1. Watson:

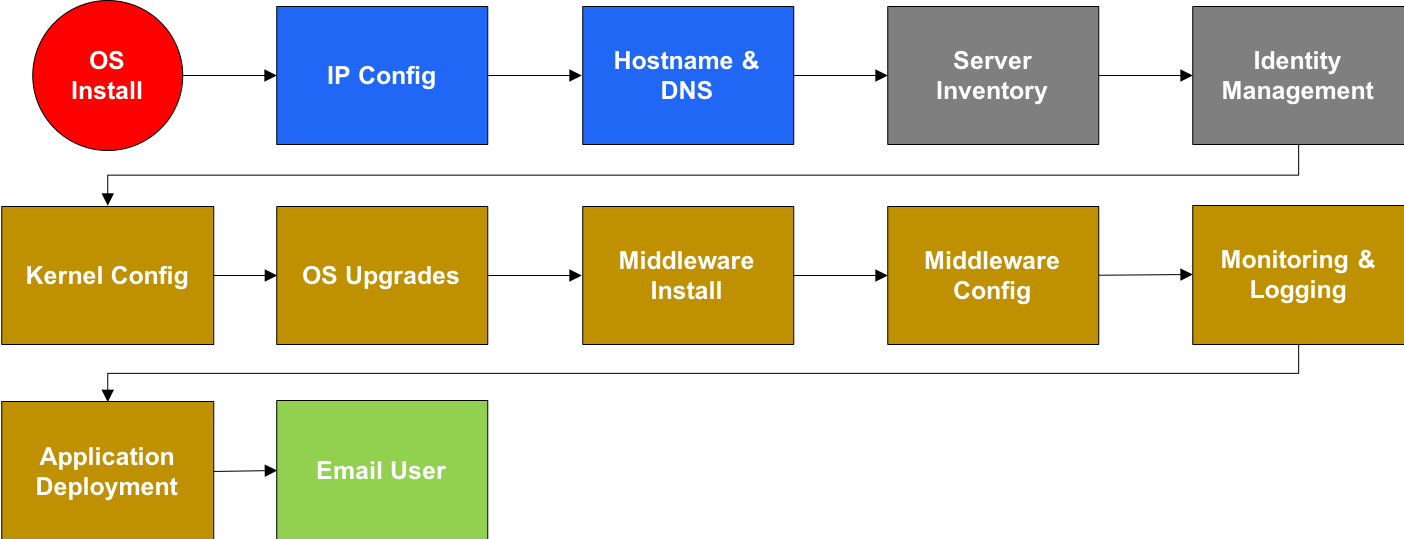
It is a question-answering computer system capable of answering questions posed in natural language.

**APPENDIX - A**

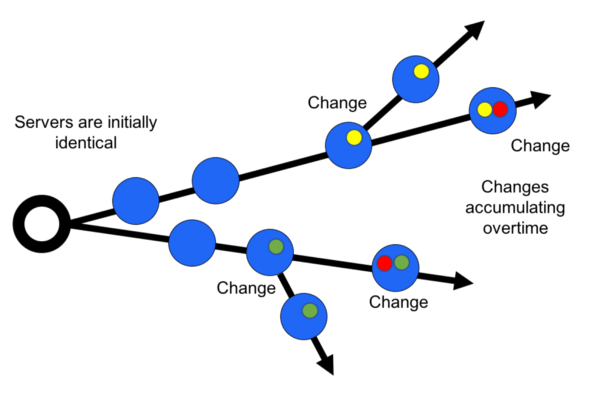
1. The Cloud Automation Management feature can be used to perform a predefined set of actions as follows:
   1. Administrators can automate most (if not all) of the processes that come with provisioning environments, avoiding manual processes that are tedious and error prone. With CAM and Terraform, administrators have to simply declare how they want the environment to look, launch the provisioning, and forget all about it. Terraform will figure out how to deploy.

Imagine having an application made out of 10 VMs and then having to repeat these steps for each and every environment in a typical enterprise scenario: Dev, QA, UAT, Production and DR, that’s roughly a total of 50 VMs. CAM’s automation reduces provisioning time from weeks to hours.

Following image displays a typical deployment procedure:



* 1. CAM solves the issue of configuration drifts. Gone are the days when applications don’t behave uniformly when the environments they sit on are supposedly identical. Configuration drifts happen when somebody, somewhere messed up and never reflected the config change to rest of the environments. This happens quite often in a typical Dev, QA, User Acceptance Testing, Production and DR deployment scenario.



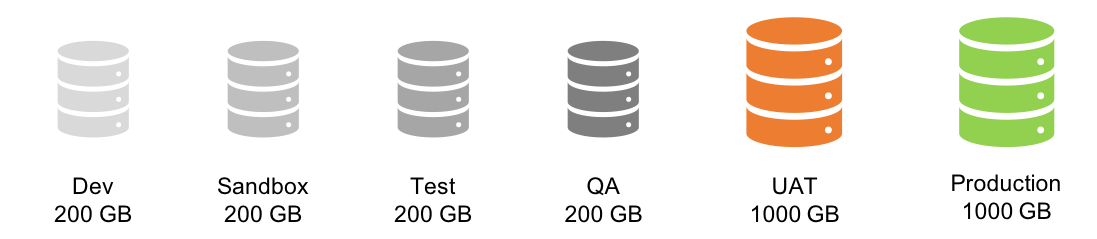
And because Terraform templates are text files that are stored in Git, it simplifies integration into existing DevOps

* 1. Just like application code, infrastructure should be **versioned**. Infrastructure encompasses everything in a software-defined environment, so in the example of VMware we’re talking, amongst other things, Storage I/O Control, OS customizations, core count, and also sometimes the type of CPU the VM is emulating to enable and disable processor features.

Infrastructure doesn’t stop there, it also includes /etc/hosts files, IP configuration, DNS entries, iptables rules, apt-get/yum repos and then, perhaps, middleware configuration might also be part of the responsibility of the infrastructure team, depending on the organization structure.

Tracking and versioning all aspects of an environment while consolidating VM and container artifacts allow configurations to be branched, forked, rolled-back and then promoted from Dev, QA, UAT, and then Production alongside every other element in an app development delivery process.

* 1. Infrastructure as Code allows Ops to **rebuild instead of fixing** when things go wrong. It has been proven time and again that rebuilding an entire environment and then rolling back to a known working version will save time and effort helping achieving SLOs and SLAs.
  2. It **saves on disk space**. Currently, organizations rely on VM templates, VM snapshots to quickly provision environments. This is an operational nightmare and wastes disk space, which incidentally opened the doors for storage appliance companies to introduce expensive duplication technologies.



Infrastructure as Code allows for replacing clones of VM images with text files that store descriptive declarative statement that reflect configurations and end targets.

1. The events that occur on the nodes are prioritized by IBM Event management and convey the same through IBM Watson to the client. IBM Watson API, internally uses Natural language processing to communicate these events occurred and take the response or guidelines from the clients. According to the guidelines, the appropriate actions will be taken by CAM.

**APPENDIX – B**

**PLAGIARISM REPORT**

