INFRAMIND SEASON-4

USE CASE

AUTONOMOUS IT

PARTICIPANT

AVIK KUNDU

NAME	CT NUMBER	EMAIL ID
Avik Kundu		1828008@kiit.ac.in

GitHub link:

https://github.com/Lucifergene/TCS-Inframinds-2021

Presentation link:

https://www.youtube.com/watch?v=rEt8-QxM8sw

Contents

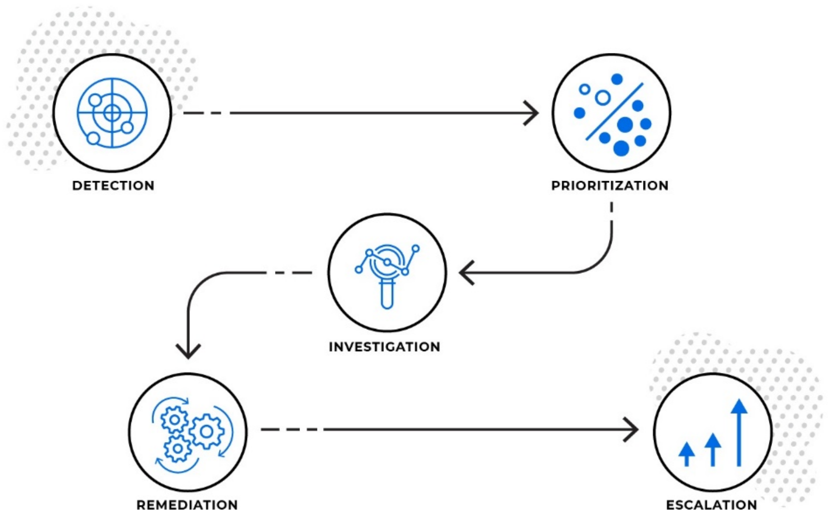
Introduction:

Digital transformation is critical for any enterprise that wants to win in today's market. It requires meaningful IT investments in new applications, clouds, services, infrastructure, and delivery models. These add complexity and scale that compound the already heavy burden on IT Operations, which in turn increases the risk of painful IT disruptions.

Service outages, lost revenue, angry customers, and runaway costs put at risk the very technology-enabled innovations meant to keep your enterprise competitive. For IT Operations to transform, it must intelligently automate and scale to keep up with a transforming IT reality. The current generation of automation technologies – based on static rules and largely built by hand are not a fit for modern, and complex IT environments. It is time for the next generation of automation.

Autonomous Operations (AO) is a new category of enterprise software, powered by machine learning, that intelligently automates incident management for large and complex enterprise IT environments. Autonomous Operations is the next generation of IT automation. It moves enterprise IT from the legacy and rules-based solutions of the past to a future of fully autonomous operations. AO eases the burden on IT Operations by combining humans and machines to dramatically improve time to resolution.

AO increases service availability while driving operational costs down. Uninterrupted operations allow more time for development of technology-enabled innovations.



Problem Statement:

One of the Retail customer is looking to automate various phases of **Infrastructure Lifecycle i.e., Design, Build, Operate & Optimize.** Overall objectives of the **Autonomous Infrastructure** are to improve agility of the business requirement and **reduce manual intervention while managing the overall lifecycle of application**.

Overall solution should consist of:

1. Create the WordPress application stack – **2 Servers, 1 ELB (1-LB, 1-Apache PHP, 1-MySql**)
2. Patch the system to latest kernel and security updates.
3. Create the **test Blog site**.
4. Create self-healing automation which will monitor the Apache-PHP and MySQL process and in case it is not running/hung it will stop/start the daemon/process.
5. Add Apache-PHP server in case of increase in **CPU utilization above 70-80%**

Software Development Lifecycle:

An SDLC is the series of phases, that takes a business case to completion. It involves the identification of issues followed by implementing solutions with recurring efforts.

Deciding how to do things is as important as actually doing them. SDLC involves fixing broken inefficiencies and reduce overall overheads and costs, reason: it helps a business stand out against competitors who require more time (or cost) for the same level of work or/and quality.

Overall, it is about improving the processes, recognizing the need for change, getting sponsorship from senior management and setting-up and forming a process group.

SDLC Classic Waterfall Model

Classical waterfall model is the basic **software development life cycle** model. It is very simple but idealistic. Earlier this model was very popular but nowadays it is not used. But it is very important because all the other software development life cycle models are based on the classical waterfall model.

Classical waterfall model divides the life cycle into a set of phases. This model considers that one phase can be started after completion of the previous phase. That is the output of one phase will be the input to the next phase. Thus, the development process can be considered as a sequential flow in the waterfall. Here the phases do not overlap with each other. The different sequential phases of the classical waterfall model are shown in the below figure:

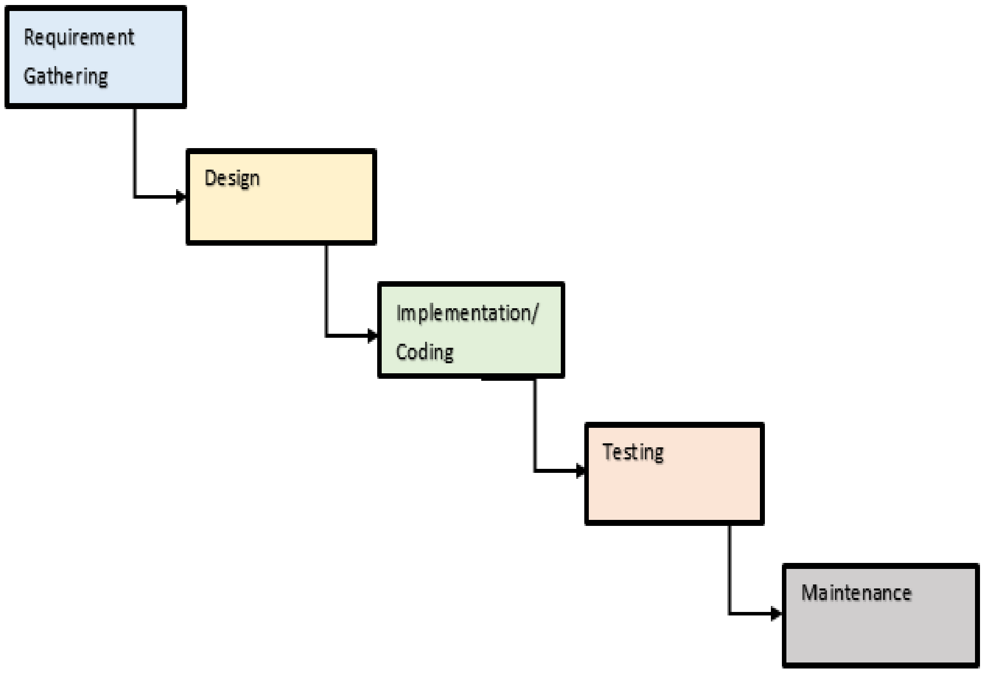


Fig. SDLC Lifecycle

Phase 1: Requirements Gathering & Analysis

The first phase involves understanding what needs to design and what is its function, purpose, etc. Here, the specifications of the input and output or the final product are studied and marked.

Business Objective

"Developing Highly Scalable and Secure WordPress Infrastructure on Cloud Services".

Software Requirements

- The infrastructure build should be automated.
- Systems must be regularly patched with latest updates.
- The infrastructure must have self-healing / Auto-scaling properties.
- The WordPress site should guarantee Zero Downtime Deployment

Phase 2: System Design

The requirement specifications from the first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and helps in defining overall system architecture. The software code to be written in the next stage is created now.

Microsoft Azure Cloud

1. Azure has currently 3 Availability Zones in India- Central, South and West India
2. Azure is 4-12% cheaper than AWS, and it also offers some extra properties which makes it better than AWS.
3. Azure gives stronger and faster PaaS capabilities which nowadays is more important part of Cloud infrastructure.
4. With the Enterprise Agreement, enterprises can typically obtain significant incentives for using Azure.
5. More than 95 percent of Fortune 500 companies use Azure.

Terraform – Infrastructure as Code

Infrastructure-as-Code (IaC) is a practice that has become mainstream with the growing popularity of public cloud providers, such as AWS, Google, and Microsoft. In a nutshell, it consists of managing a set of resources (computing, network, storage, etc.) using the same approach developers use to manage application code.

Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can manage existing and popular service providers as well as custom in-house solutions.

HashiCorp
Terraform

Ansible – Configuration Management

Ansible is an open-source configuration management and provisioning tool, like Chef, Puppet or Salt. It uses SSH to connect to servers and run the configured tasks.

Ansible lets you control and configure nodes from a single machine. What makes it different from other management software is that Ansible uses SSH infrastructure.

RED HAT®
ANSIBLE®
Automation

WordPress Architecture

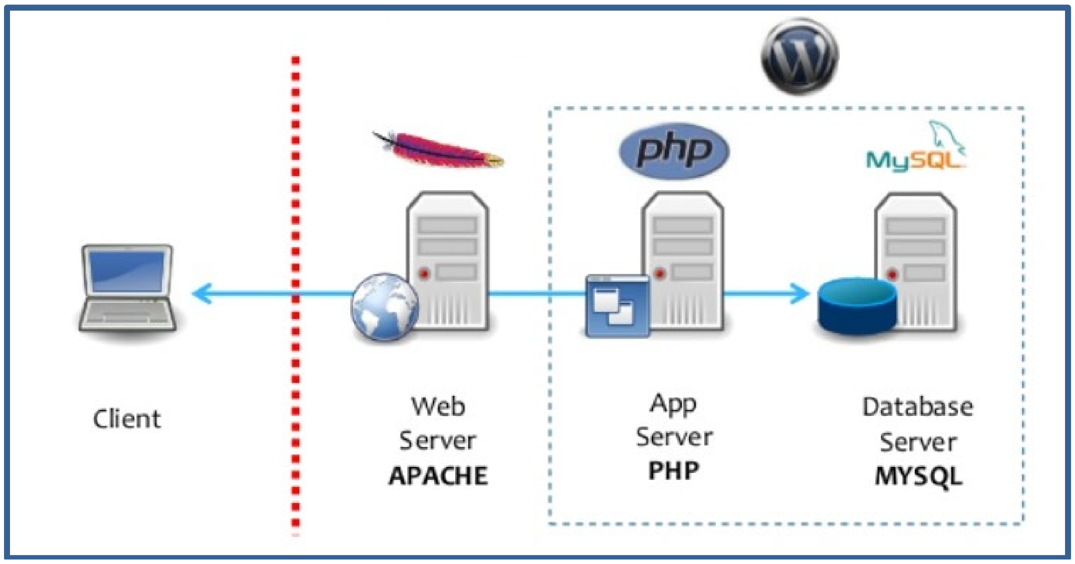


Fig. WordPress Architecture

Azure Architecture

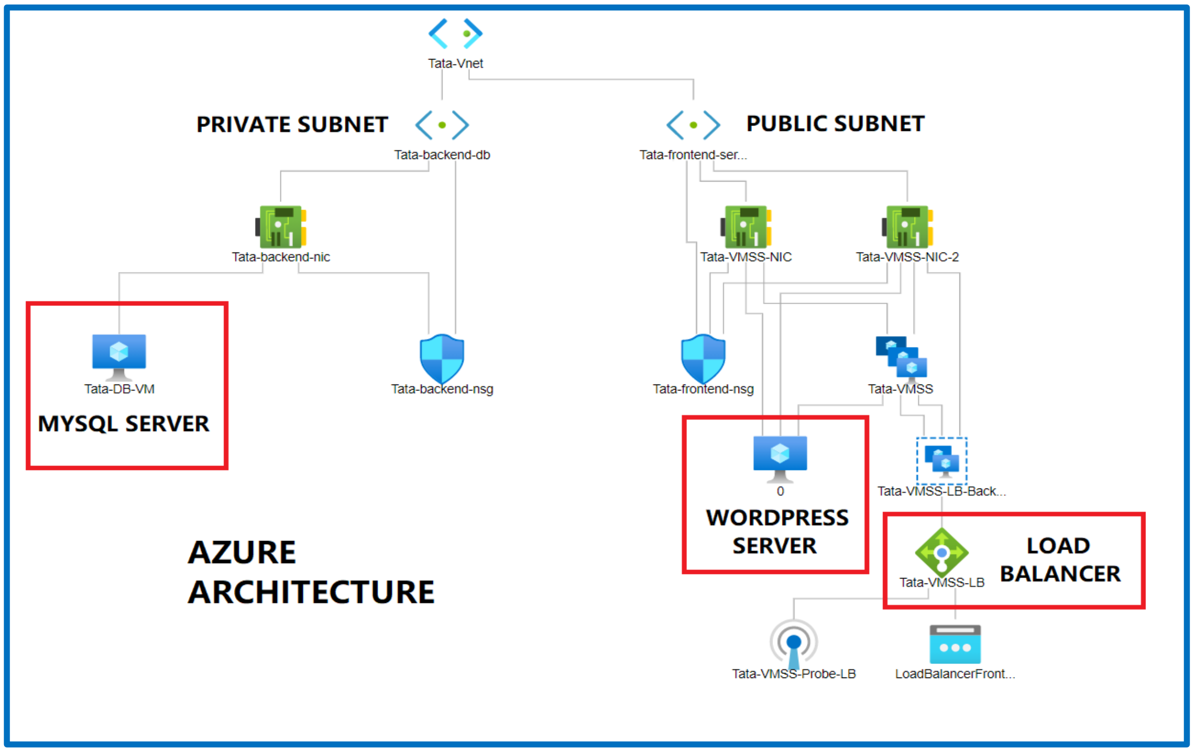


Fig. Azure Architecture

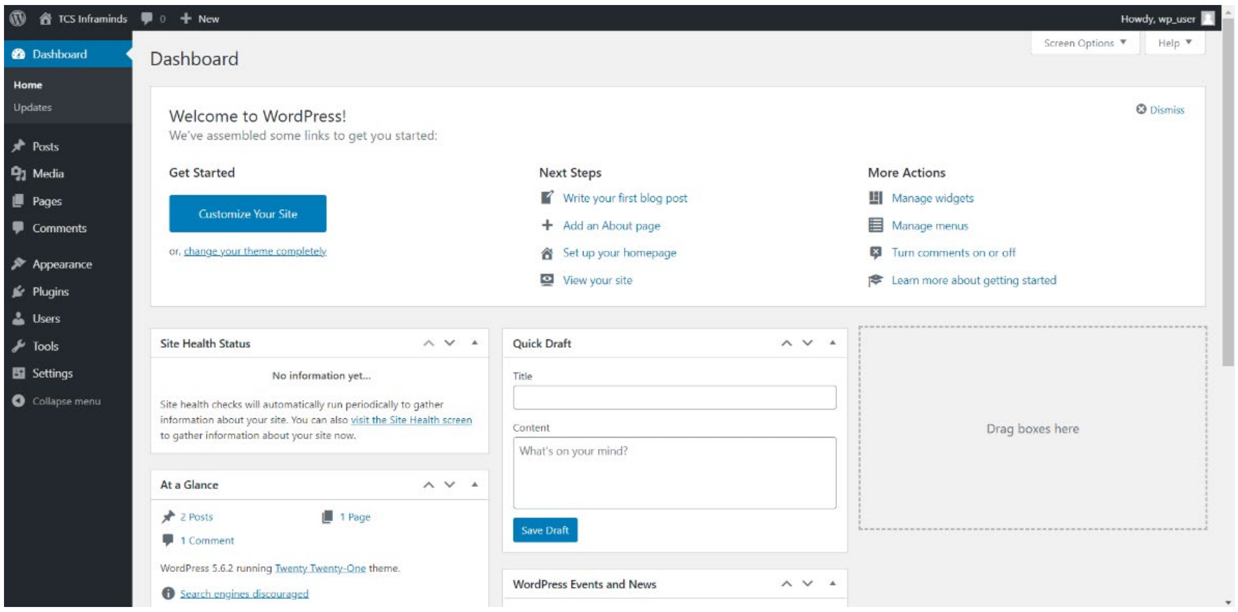
Phase 3: Implementation

With inputs from system design, the system is first developed in small programs called units, which are integrated into the next phase.

In this phase, we developed the infrastructure on Azure Cloud through the help of Terraform. This code functions as a documentation of the infrastructure as well as for maintenance purposes. While creating the infrastructure, special attention was given to secure the Database from attacks. For that, we had setup 2 subnets, one which had active outgoing internet connection while in the other subnet, outgoing internet was closed. In this subnet, we installed the VM for the Database since Databases do not need to be connected to the outside world, rather they must be only interacting with the resources within the Virtual Network. Thus, we ensured Security in the WordPress infrastructure.

After setting up the infrastructure on Azure, we used Ansible as the configuration management tool to patch the servers. For that, we had to connect our VM to the Virtual Network of the infrastructure and then finally executed the code to perform the tasks.

After that, we setup the MySQL and the WordPress servers in separate VMs that were connected to a Public and Private Subnet, respectively.



Welcome to WordPress!
We've assembled some links to get you started:

or, change your theme completely

No information yet...

Site health checks will automatically run periodically to gather information about your site. You can also visit the Site Health screen to gather information about your site now.

What's on your mind?

WordPress 5.6.2 running Twenty Twenty-One theme.

Fig. WordPress Dashboard

The next task was to setup an Auto-Scaling Service so that the infrastructure becomes self-healing and can guarantee Zero Downtime Tolerance. For that, we used Azure Virtual Machine Scale Sets. There we adjusted the scaling rules such that once the CPU Utilization would reach 70%, automatically new instance would be launched, and once the utilization decreases, the instance would automatically get shutdown.

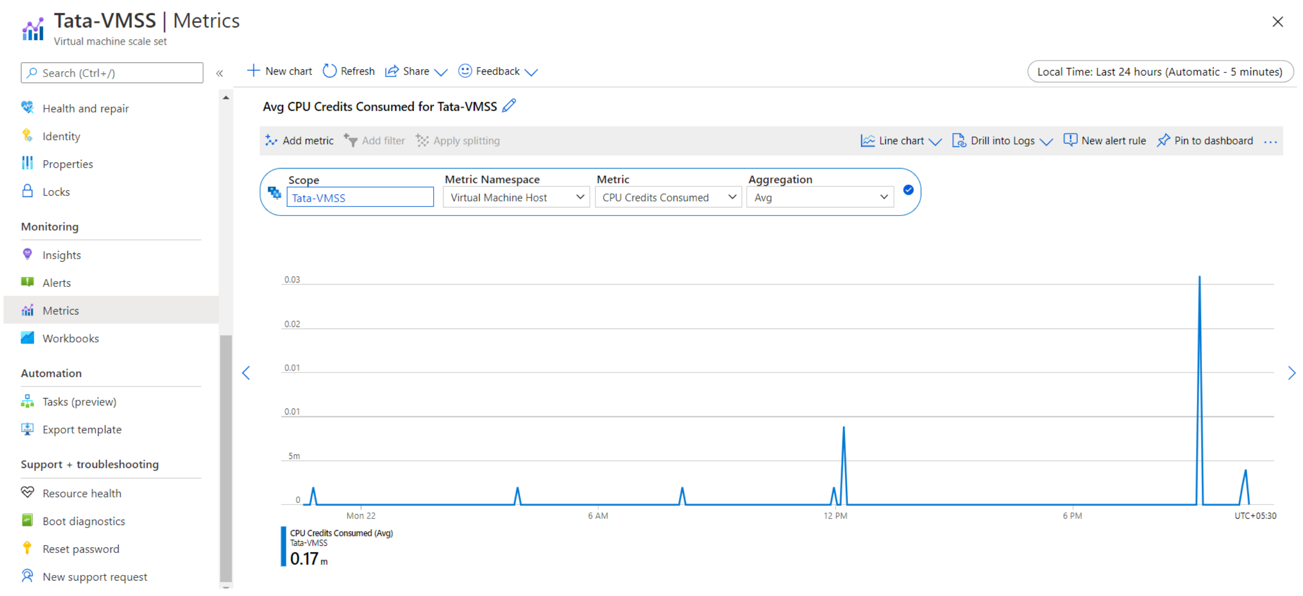
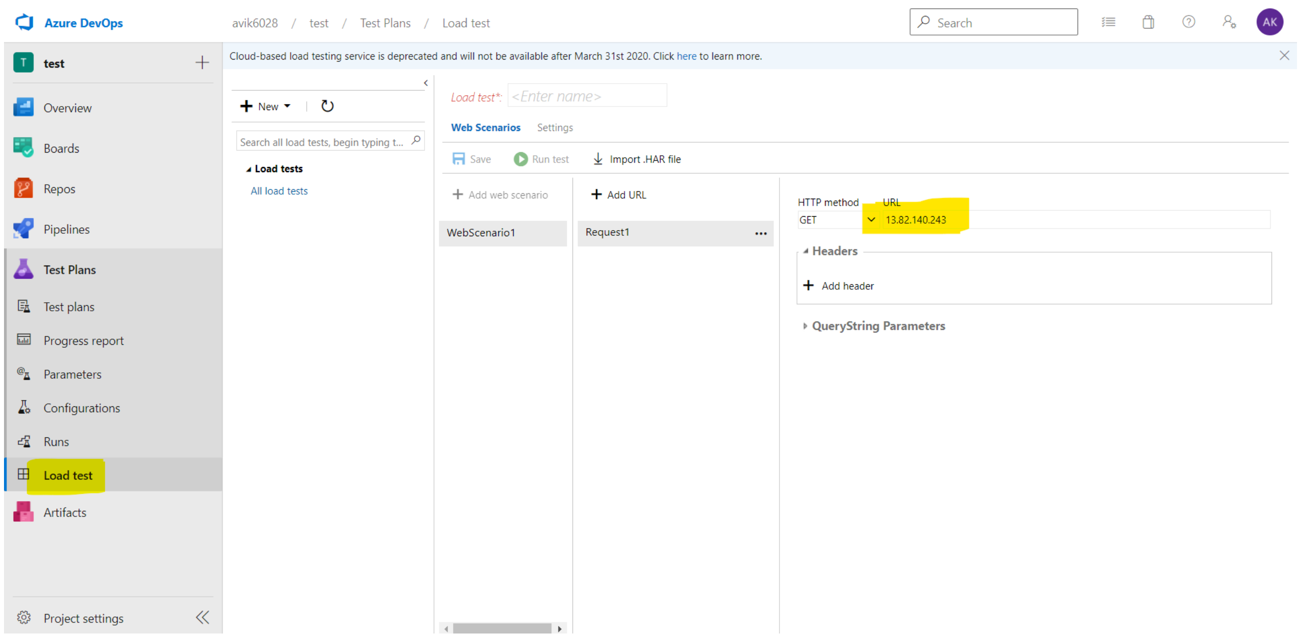


Fig. Monitoring Scale Set Performance

Phase 4: Testing

All the units developed in the implementation phase are integrated into a system after testing of each unit. The software designed, needs to go through constant software testing to find out if there are any flaws or errors. Testing is done so that the client does not face any problem during the installation of the software.

To test whether the auto-balancing service is working properly, we performed a load test on the scale set with the help of Azure DevOps Tools.
We saw a creation of VM once the load test began and once the test was closed, the VM shut down.



Cloud-based load testing service is deprecated and will not be available after March 31st 2020. Click here to learn more.

HTTP method: GET
URL: 13.82.140.243

Fig. Azure DevOps Platform for Load Testing

Phase 5: Maintenance

This step occurs after installation and involves making modifications to the system or an individual component to alter attributes or improve performance. These modifications arise either due to change requests initiated by the customer, or defects uncovered during live use of the system. The client is provided with regular maintenance and support for the developed software.

Since we had set up codes for infrastructure as well as for patching and upgrading systems, it would become easier to maintain the infrastructure.

Conclusion

Thus, we had successfully prepared the deliverable as per as the business requirement stated. By the end of the prototype, we have also found some more future optimizations which we can surely perform. For example, rather than managing the Database ourselves we can easily use the Database-as-a-Service feature of Azure. Then we can set up CDN for caching the website to effectively decrease the load time.

We can finally say that the prototype we have developed is fully autonomous.

Thank You