SSC23-S2-03

Lessons Learned on Building and Operating Software Defined CubeSat for Scalable Satellites

Minsik Lee
IBM
1 N Castle Dr, Armonk, NY 10504; 201-580-1101
mslee@us.ibm.com

Naeem Altaf
IBM
11501 Burnet Road, Austin, TX 78758; 512.657.6328
naltaf@us.ibm.com

Hanna Sohn
Georgia Institute of Technology
266 Ferst Dr, Atlanta, GA 30313; 201-580-1419
hsohn37@gatech.edu

ABSTRACT

Software Defined CubeSats are small and low-cost satellites that utilize software-based technologies for their operations and functionalities. While traditional hardware based CubeSats in large-scale systems have limitations for scalability due to their fixed environments, Software Defined CubeSats offer many advantages such as flexibility, scalability, cost-effectiveness, reliability, and rapid deployments.

IBM Endurance CubeSat is a Software Defined CubeSat that aims to pave the way for future space missions. On May 25th, 2022, IBM Endurance CubeSat launched successfully in Florida, USA, and now it provides CubeSat as a Service to users who need to conduct space missions or experiments. The CubeSat uses virtualization technology and cloud-based infrastructure to perform data processing, equipped with sensors and a camera. In order to provide a stable CubeSat as a Service, the Mission Control System has continuously improved and operated with 184 successful jobs uploaded from the ground and executed dynamically in space. Development and automation tools have also been developed and operated for end users to develop code for themselves. Currently, the CubeSat is provided as a Service for various scientific research experiments in the fields of device security, Earth observation, and sensor analysis from schools and higher education.

The CubeSat uses containers for packaging and deploying applications. Containers are a type of virtualization technology that allows applications and their dependencies to be packaged together in a single and self-contained unit, which helps deploy and manage applications in different environments, such as cloud computing, edge computing, and space-based systems. This feature of hardware independence performs data processing and storage tasks in a highly flexible and dynamic manner. Containers also enable easy updates and upgrades to satellites' hardware capabilities, responding quickly to changing mission requirements and user needs.

In this paper, we will show how to implement and operate CubeSat as a Service by utilizing Software Defined Environment and Container/Virtualization Technologies. In addition, we will also demonstrate how to automate the process using CI/CD (Continuous Integration and Continuous Deployment) pipeline. First, we will describe Virtualized Software Runtime consisting of an off-the-shelf Operating System, Podman, and MicroShift. Podman is an open-source container technology that is widely used for deploying applications in cloud and edge computing environments. We used Podman to deploy applications on the CubeSat, running complex applications and services in a scalable and efficient manner. MicroShift is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. Second, we will introduce a CubeSat Mission Control Center for Software Defined CubeSats. The CubeSat Mission Operations Center is a centralized facility responsible for controlling and monitoring the operations of CubeSats. Third, we will show CI/CD pipeline that has

a set of practices and tools to automate the software development process, from code creation to deployment. The CI/CD pipeline helps to automate the software development and deployment process. This centralized software life cycle management enables frequent updates and improvements to keep pace with users' changing requirements and needs.

INTRODUCTION

Work In Progress.

DESIGN OF ENDURANCE CUBESAT

IBM Endurance CubeSat

Work In Progress.

Requirements

Work In Progress.

Architecture Overview

Work In Progress.

Virtualized Software Runtime

Work In Progress.

CI/CD (Continuous Integration and Continuous Deployment) Pipeline

Work In Progress.

IMPLEMENTATIONS

Overview

Work In Progress.

Virtualized Software Runtime

Work In Progress.

CI/CD (Continuous Integration and Continuous Deployment) Pipeline

Work In Progress.

LESSONS LEARNED

Work In Progress.

CONCLUSION

Work In Progress.

REFERENCES

Work In Progress.