National Centre for Radio Astrophysics

Student Training Program 2024 Report

GMRT/TGC/001- Jun 2024

NCRA 15M Control Monitoring System (CMS) Software Build Using Docker

(Duration: Jun-Aug, 2024)

By

Setu Sameer Fulawade.

Under Supervision of:

Mr. Jitendra Kodilkar, Mr. Anil Raut

Objective: The legacy software for the NCRA 15m Radio Telescope Control and Monitoring System (CMS) operates on the RHEL 6 OS with Firefox version 57.3.2 for the user interface (UI). Due to the outdated OS and associated software components or libraries, the CMS is currently experiencing usage issues and several software upgrade problems. To address these issues and facilitate future software system upgrades and maintenance, it is essential to rebuild and containerize the software system using Docker.

This technical report details the implementation and testing of the NCRA 15m Radio Telescope Control and Monitoring System on CentOS 7, utilizing Docker.

Revision	Date	Modification/Change
Ver. 9	23/07/2024	Initial Draft Version
Ver : 1.0	07/08/2024	Reviewed Version
Ver : 1.1	12/08/2024	Final Version

Acknowledgement

I would like to express my profound gratitude to Mr. Jitendra Kodilar for giving me the opportunity to work on the 'NCRA 15M Control Monitoring System (CMS) Software Build Using Docker' project. I also extend my heartfelt thanks to Mr. Anant Khatri, the head of the department, for allowing me to undertake this project.

I am deeply grateful to the NCRA Director, Prof. Y. Gupta, the GMRT Dean, Prof. Ishwar Chandra, the Telemetry & Operation Group Coordinator, Mr. Anil Raut, and Dr. Yogesh Maan (for availing the Linux Machine '***padma***') . I would also like to thank the NCRA and GMRT staff, including Mrs. Deshmukh, who assisted me throughout my project.

Working at GMRT was a wonderful experience. I had the opportunity to learn more about GMRT's work culture and to interact with the amazing people there. Through this project, I gained valuable insights into the control and monitoring systems of the 15-meter radio telescope and the DevOps tool such as DOCKER containerisation software for compilation and deployment. This experience has significantly enhanced my technical skills and understanding of real-world applications in radio astronomy.

Executive Summary

NCRA 15m Radio Telescope Control & Monitoring System : The NCRA 15m Radio Telescope Control & Monitoring System (CMS) controls and monitors the 15m antenna Subsystems ranging from the Front-end Common-Box, Servo, Analog chain and Digital backend receiver systems to perform the radio astronomical observation by configuring or setting the RF-receiving chain systems and tracking a celestial source using the Servo-subsystem. This system runs presently using the PowerEdge R310 DELL server with RHEL 6 operating system. The system was commissioned around 2011, and in use since the past decade.

Objective : The 15m CMS system is supported by the GMRT staff located at the Khodad, hence upgradation of the CMS on the latest OS has not been possible in the past due to GMRT responsibilities and other jobs at higher priorities.

Recently one of the CMS supporting Dell-Server crashed in Feb 2024, also the maintaining and upgrading of the CMS became a difficult task as machines are old and RHEL 6 platform is no longer supported. The CMS web-based UI was running on the old version of the browser with Flash-player plugins which are not supported in the new browsers after 2020.

Therefore to upkeep the system for future use, and also to preserve the CMS code-base where efforts and cost have been put in the past, it was essential to deploy this system on the latest Linux OS. However, the latest Linux OS (Fedora, CentOS, and Ubuntu) repositories are by far upgraded with latest versions of tools and libraries which are not matching any more with the RHEL-6 OS , CMS software code-base and required compilers, associated libraries etc. Hence, the primary objective of this project is to containarise the CMS software using Docker.

Docker : DevOps (Development Operation) is one of the important buzzwords in the current trends of s/w technology. DevOps combines Developments and Operations aspects of the SDLC to improve efficiency, security, and help in faster delivery of the software compared to traditional processes. Docker is a key DevOps tool that improves software efficiency by enhancing portability and deployment. It is an open platform for developing, packaging, shipping, and running applications, using OS-independent virtualization. The Docker engine on the host machine supports containerization, allowing software packages to run seamlessly on various systems.

Project Implementation : The 15m CMS system and Web-based UI is successfully built under the docker using the CentOS-7 base-image. The CMS system implementation under docker enabled the system for future use, and can be evolvable or modifiable for the telescope control and monitoring requirements. The CMS system is validated to be working under docker, and tested for performance which shows that both the UI and CMS system can be deployed on the latest Linux OS versions, and usable in an efficient manner much like the legacy system.

Findings:

1. **Improved Efficiency:** The implementation of Docker significantly improved the efficiency of software deployment, reducing the time and effort required for setup and configuration.
2. **Enhanced Control Systems:** The newly developed control systems provided more precise and reliable control over the telescope's operations.
3. **Robust Monitoring:** The monitoring systems effectively tracked the telescope's performance, allowing for early detection and resolution of any issues.
4. **Reproducibility and Consistency:** Docker ensured a consistent software environment, which was crucial for maintaining the reliability of the control and monitoring systems.

Acronyms :

ActiveMQ	**ActiveMQ is an open source protocol developed by Apache which functions as an implementation of message-oriented middleware. Its basic function is to send messages between different applications.**
Apache-ant	**Apache Ant is a software tool for automating software build processes for Java applications**
Apache-Maven	**Maven is a build automation tool used primarily for Java projects**
CentOS	**Community Enterprise Operating System Linux OS**
CMS	**Central Control and Monitoring system**
DevOps	**DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity**
DOCKER	**Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers https://docs.docker.com/guides/docker-overview/**
DOM	**The Document Object Model (DOM) is the data representation of the objects that comprise the structure and content of a document on the web.**
Eclipse	**Eclipse is an integrated development environment used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment.**
jar	**A JAR file is a package file format typically used to aggregate many Java class files and associated metadata and resources into one file for distribution.**
JDE	**The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets**
JRE	**The Java Runtime Environment, or JRE, is a software layer that runs on top of a computer's operating system software and provides the class libraries and other resources that a specific Java program requires to run**
Mysql	**MySQL is an open-source relational database management system.**
NCRA	**National Centre For Radio Astrophysics, Pune, India**
NCRA 15m	**NCRA 15 Meter Radio Telescope**
Pom	**A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects**
RHEL	**Red Hat Enterprise Linux is a commercial open-source Linux**

	distribution developed by Red Hat for the commercial market.
XML	**Extensible Markup Language (XML) is a markup language that provides rules to define any data**

Table no.1

INDEX

1 . Introduction

1.1 About the NCRA 15m Radio Telescope:

The NCRA has built a 15 meter Radio Antenna in the East-Campus of NCRA, Pune . The Antenna observing Frequency range is from 1000 to 1450 MHz with five sub-bands of 100 MHz each. The telescope is Alt-Azimuth mounted, the upgraded data processing analog backend and digital backend can process the 200 MHz bandwidth RF-signals with Max 2K channels. The visible Sky coverage is from ~ 54 to 90 degrees of declination. The User

Figure-1 shows the NCRA 15m Radio Telescope.

Interface is provided over a web to control the telescope remotely for performing the astronomical observations.

The telescope Control and Monitoring System (CMS) is a configurable system which is developed based on the Java web-based Spring framework and Tomcat web-server. A generic and specification driven CMS was developed around 2008-2009 in collaboration with PSPL (Persistent System Private Limited) and IUCAA (Inter University Centre for Astronomy & Astrophysics) which can be configured for the optical and radio telescope. At present the CMS software runs on RHEL 6 Operating system with ***PowerEdge R310 Dell server*** machine which has completed its End of Life. Since the system of 15m antenna is supported partially by the GMRT staff located at khodad, Pune, the telescoped software upgradation has not been done on a routine basis. Therefore, running of the CMS on the latest Linux based OS version is difficult because many of the s/w components, spring-framework dependencies are no longer supported. Hence, in case of machine malfunctioning, RHEL 6 supports unavailability; it is essential to port the CMS software on a new machine with the latest OS version. To address this issue, we explore various solutions and propose a software containerisation for running the CMS software on the current generation of Linux OS based machines.

1.2 User Requirement Specifications, and S/w Requirement Specifications:

The user and software requirement specifications analysis was the first task taken for the project on "***15m Control and Monitoring System software compilation using docker***" to analyze and plan the design and development for the project. After discussing with users of 15m CMS and s/w development & maintenance stake-holders, following user requirements are listed.

URS ID	System	User Requirement	Stakeholder
URS001	UI (Front-end)	The existing Web based User Interface of 15m antenna shall run on a remote machine to access the control and monitoring of 15m antenna.	Astronomer, Engineer, Students
URS002	CMS Back-end	The RHEL 6 license and support has EoL, the CMS backend shall run on the Linux system other than RHEL 6.	CMS developer and maintenance

Table no.2

After analysing the URS, Software Requirement specifications deduced are as follows:

SRS ID	URS ID	Software Specification Requirements	Feasibility analysis and Constraints
SRS001	URS001	User Interface shall run on the latest browsers with the same GUI components appearance on the web as it runs on the legacy system.	The latest browsers such as Firefox, google-chrome etc. do not support Flash-player. Hence running on the legacy Firefox old ver 52.7.3 may be possible.
SRS002	URS001	The Web-interface shall run on the latest Linux and Windows OS	The porting of old firefox-version (52.7.3), flash-player plugins are not

			possible to port on the latest Linux or OS windows version because s/w support is not available.
SRS003	URS002	CMS Software shall run on the latest Linux OS	**(1)** The CMS s/w uses a Tomcat 6, Apache ActiveMQ (Message Queue) <Ver> , and technologies associated with Java JDK version 1.6.0_30. The s/w support is already EoL and outdated. Hence, the CentOS 7 (compatible to RHEL 6 and available) can be tried to compile the s/w. **(2)** The Docker containerisation of CentOS 7 can be made to run and support maintenance on the new Linux OS systems
SRS004	URS002	It shall be possible to modify, compile and port the CMS software on latest Linux OS	

Table no.3

1.3 Objectives:

At present the CMS system of 15m antenna facing the following issues :

- **RHEL 6 EOL:**
  - The 15-meter radio telescope at the National Centre for Radio Astrophysics (NCRA) relies on a Control and Management System (CMS) version 3.3.7 to oversee and coordinate its operations. This CMS is currently running on Red Hat Enterprise Linux (RHEL) 6. However, RHEL 6 reached its end of life on November 30, 2020 and is no longer supported, creating significant risks for the telescope's operations.
- **CMS Compatibility Issue:**
  - The end-of-life status of RHEL 6 necessitates an urgent migration to a supported operating system to ensure the security and operational integrity of the telescope. However, migrating the CMS to RHEL 9 has encountered significant compatibility issues, including deprecated dependencies and configuration conflicts. These issues prevent the CMS from functioning correctly on the newer OS, presenting a substantial challenge in maintaining the telescope's control and management capabilities.
- **Firefox Compatibility Issue:**
  - The 15-meter radio telescope relies on a monitor and control (CMS) system that utilizes outdated technology, posing significant challenges and risks to its functionality and sustainability. The current M&C system operates on Firefox 52.7.3, a version lacking critical updates and features.

- **Flash-Player Compatibility Issue:**
    - The 15-meter radio telescope at the National Centre for Radio Astrophysics (NCRA) currently relies on a Control and Management System (CMS) running on Red Hat Enterprise Linux (RHEL) 6, using Firefox version 52.7.3 that requires Flash Player for its display functionalities. With RHEL 6 and Firefox 52.7.3 being outdated and unsupported, and Flash Player no longer being supported in modern browsers.

1.4 Report Structure:

- **1. Introduction:**
    - The introduction section outlines the 15m radio telescope at NCRA's Pune campus and the development of a generic web-based Monitoring and Control (M&C) system**.** It highlights the need to modernize the M&C system due to the end-of-life status of Red Hat Enterprise Linux (RHEL) 6, which poses risks from outdated software dependencies such as Firefox and Flash Player.
- **2. Feasibility Study:**
    - This chapter evaluates the technical and operational feasibility of modernizing the 15m radio telescope's M&C system. It assesses the compatibility of existing software and hardware, explores potential solutions for overcoming identified challenges, and determines the practicality of implementing Docker technology.
- **3. Implementation:**
    - This chapter outlines the step-by-step process of implementing the proposed solution. It details the configuration and deployment of Docker containers, the setup of necessary software components, and the integration with existing systems.
- **4. Validation Testing:**
    - This chapter describes the validation and testing procedures carried out to ensure the successful deployment of the new system. It includes test cases, results, and an analysis of the system's performance and reliability.

2 . Feasibility Study

2.1 Aim:

The aim of this chapter is to evaluate the technical and operational feasibility of modernizing the 15m radio telescope's Monitoring and Control (M&C) system. This includes assessing the compatibility of existing software and hardware, exploring potential solutions for overcoming identified challenges, and determining the practicality of implementing Docker technology to achieve a stable and maintainable environment for the system.

2.2 RHEL 6 Docker Compatibility:

Based on our thorough investigation, it has been conclusively determined that installing Docker on RHEL 6 for the CMS of the 15-meter radio telescope is not feasible due to compatibility constraints. The Docker engine and its dependencies require newer versions of system libraries and kernel features that are not available in RHEL 6.

2.3 A Feasibility Study of the CMS UI and CMS Framework implementation:

2.3.1 Flash-Player Extension:

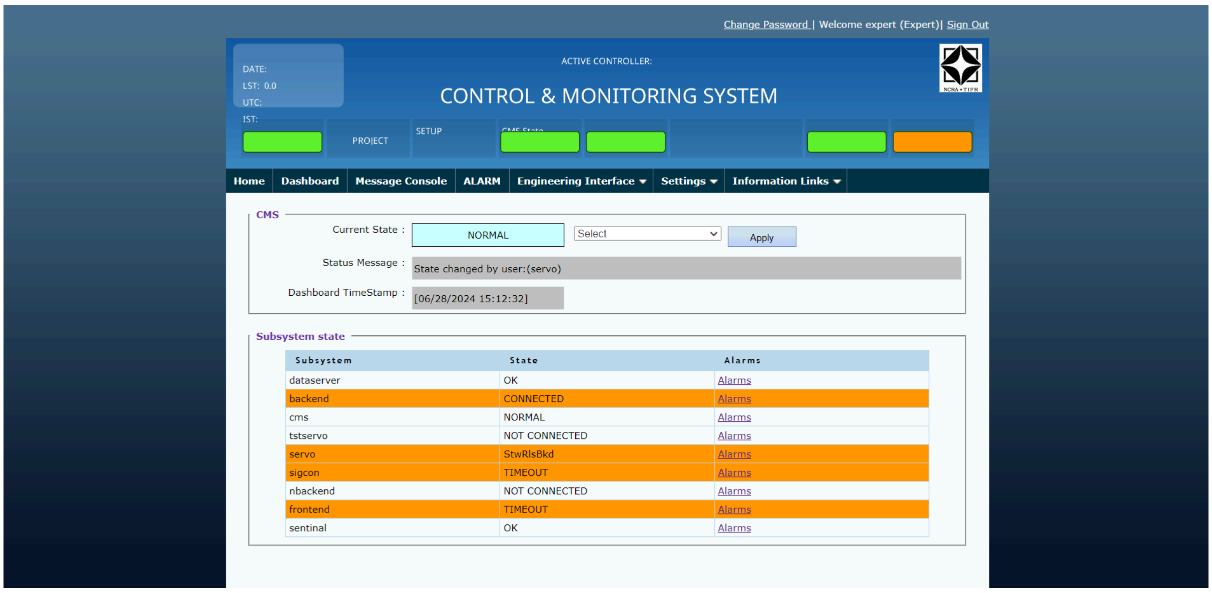
A Flash Player emulator is a software tool or system designed to replicate the functionality of Adobe Flash Player in environments where the original Flash Player is no longer supported or available. This is particularly relevant since Adobe officially ended support for Flash Player at the end of 2020, leading to widespread incompatibility with modern web browsers and operating systems.

Flash Player emulators typically aim to provide the ability to run Flash content, such as animations, games, and interactive applications, in a controlled environment that mimics the behaviour of the original Flash Player.

- **Examples**:
    - **Ruffle**: An open-source Flash Player emulator written in Rust, which aims to provide support for both older and newer Flash content by translating it to HTML5 and modern web technologies.
    - **CheerpleX**: A commercial solution that runs Flash applications within a virtualized environment, allowing businesses to continue using legacy Flash-based software.
    - **Lightspark**: An open-source project that focuses on supporting newer Flash formats and features.(add more about light spark)

- **Problem with Flash-Player extension and engines:**

  The Flash Player extension currently in use is incompatible with the CMS website, causing significant issues in functionality. **Despite being installed and enabled, the extension fails to load or display Flash content correctly**, leading to a disrupted user experience. This incompatibility could stem from outdated software, security restrictions, or the declining support for Flash technology. As a result, users encounter errors or blank spaces where interactive elements should appear, hindering the website's overall effectiveness and usability. It is crucial to explore alternative solutions.



CMS

Current State : NORMAL

Status Message : State changed by user:(servo)

Dashboard TimeStamp : [06/28/2024 15:12:32]

Subsystem state

Subsystem	State	Alarms
dataserver	OK	Alarms
backend	CONNECTED	Alarms
cms	NORMAL	Alarms
tstservo	NOT CONNECTED	Alarms
servo	StwRlsBkd	Alarms
sigcon	TIMEOUT	Alarms
nbackend	NOT CONNECTED	Alarms
frontend	TIMEOUT	Alarms
sentinal	OK	Alarms

Figure-2: Ruffle Compatibility Issue

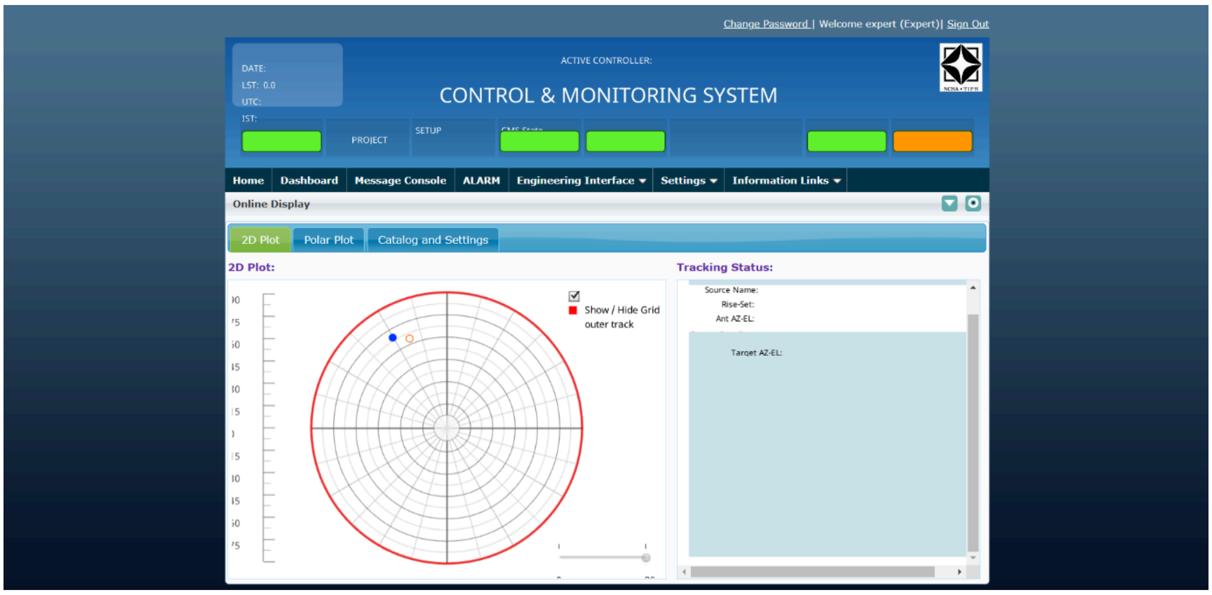
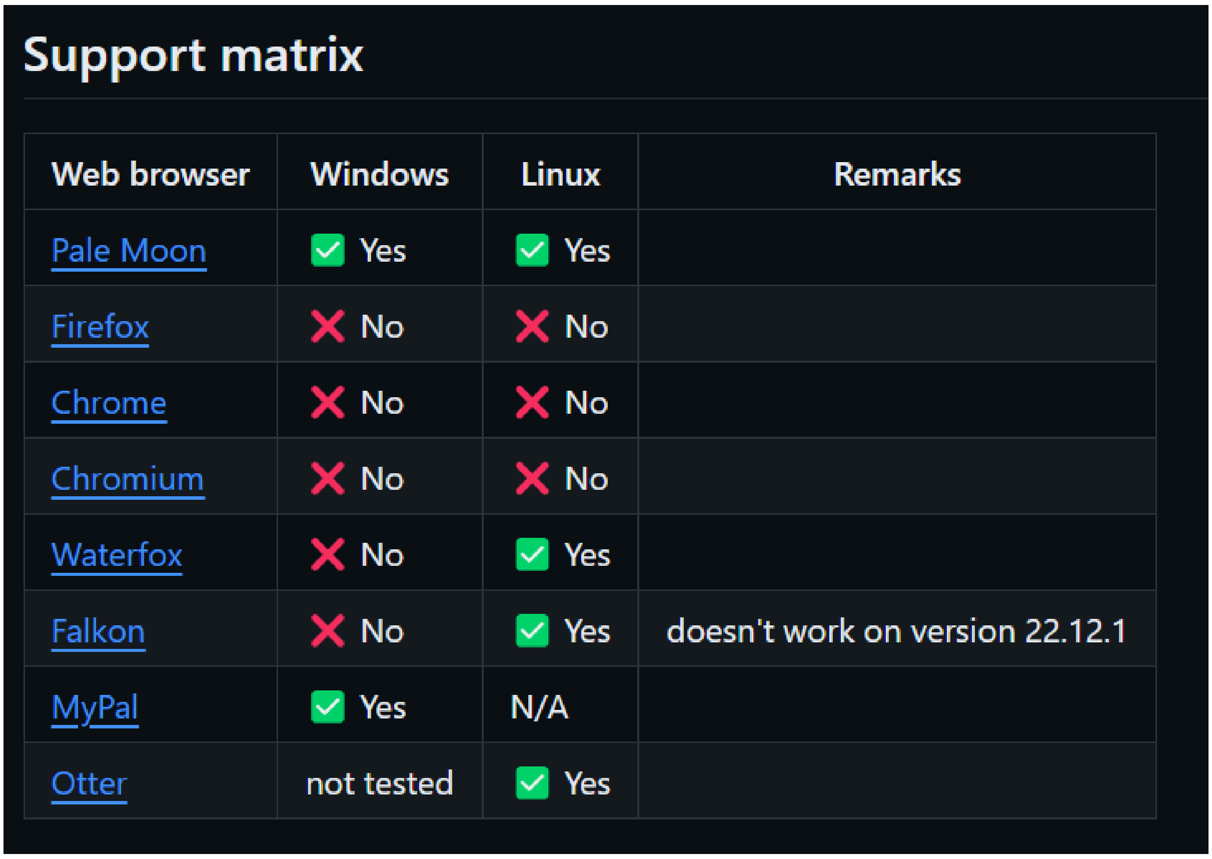


Figure-3: Ruffle Compatibility Issue



Support matrix

Web browser	Windows	Linux	Remarks
Pale Moon	✓ Yes	✓ Yes	
Firefox	✗ No	✗ No	
Chrome	✗ No	✗ No	
Chromium	✗ No	✗ No	
Waterfox	✗ No	✓ Yes	
Falkon	✗ No	✓ Yes	doesn't work on version 22.12.1
MyPal	✓ Yes	N/A	
Otter	not tested	✓ Yes	

Figure-4: LightSpark Support issue

- **Conclusion:**

  Most Flash Player emulators are only supported on the latest versions of web browsers and are designed to emulate newer versions of Flash Player. However, according to tests conducted, these emulators do not work with the Control and Monitoring System (CMS) web interface, which relies on Flash Player 10.

2.3.2 Feasibility check of The CMS Implementation:

The **CMS s/w** comprises a java based SPRING framework, and many s/w components or Java libraries which are more than ~ 15 packages. An implementation of these with latest upgraded packages available on new Linux OS versions using Redhat Package Manager (RPM) or debian packages repositories (apt) is almost difficult due to no more support availability. Hence, one of the viable options was to implement these software on the available Linux OS version which is close to the contemporary RHEL 6 packages. After exploring many available Linux flavours such as Fedora and CentOS, we selected the CentOS-7 version of Linux OS which is still available.

The CentOS-7 OS version was installed on a new machine named ‘padma’ and compatible packages such as mysql 5.5.31 , Tomcat 6.0.35, ActiveMQ 5.5.1, Apache

Maven 3.0.4, Ant Version 1.7.1 and Java 1.6.0_30 installed on it with Eclipse Helios 3.6.1. All the requisite dependencies Java Jar files, s/w code for the cms-core, cms-state, cms-batch-processing, cms-validator, cms-web were copied to this machine.

After configuring the Eclipse, and compiling using the Eclipse and Apache maven and cms-war file to run under the Tomcat 6 version is installed. The CMS was tested for its working on the CMS. And then it was decided to create a CentOS 7 docker image for deploying the CMS software.

2.4 Docker:

Docker is an open-source platform that automates the deployment, scaling, and management of applications within containers. Containers are lightweight, standalone, and executable packages that include everything needed to run a piece of software, including the code, runtime, libraries, and system tools. Docker streamlines the development-to-production workflow by providing a consistent environment across different stages of the application lifecycle. Here are some key aspects of Docker:

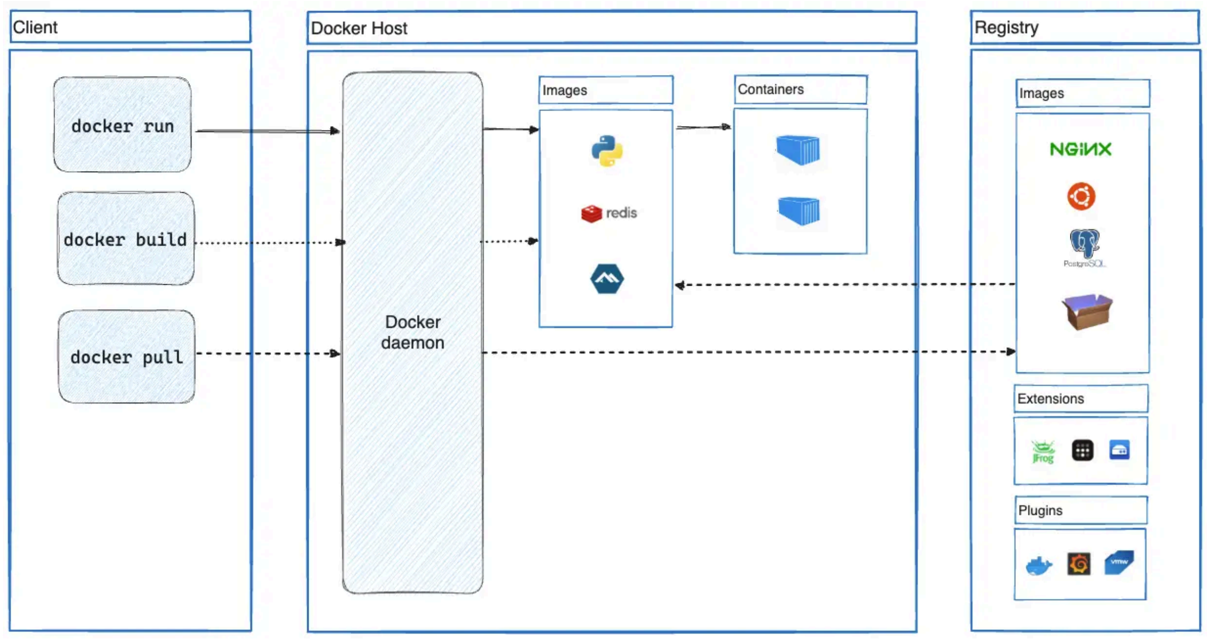


Figure-5: Docker block Diagram

- **Key features and concepts :**

  1. **Containers**: Docker containers are isolated environments that package software and its dependencies. Unlike virtual machines, containers share the host system's kernel, making them more lightweight and efficient.

  2. **Docker Engine**: The core component of Docker, the Docker Engine, is responsible for creating and running Docker containers. It includes a daemon (dockerd), a REST API, and a command-line interface (CLI).

3. **Docker Images**: Images are read-only templates used to create containers. They include the application code and the necessary runtime environment. Images are built from Dockerfiles, which contain instructions for assembling the image.
4. **Dockerfile**: A text file that contains a set of instructions to build a Docker image. It specifies the base image, application code, dependencies, and commands to run the application.
5. **Docker Hub**: A cloud-based repository where Docker users can share and distribute Docker images. It hosts official images, community images, and user-contributed images.
6. **Docker Compose**: A tool for defining and running multi-container Docker applications. Using a YAML file, Docker Compose allows you to specify how containers should be configured, linked, and orchestrated.
7. **Docker Swarm**: A native clustering and orchestration tool for Docker. It allows you to create and manage a cluster of Docker nodes as a single virtual system.
8. **Kubernetes**: Although not a part of Docker, Kubernetes is often used in conjunction with Docker for container orchestration, providing features like automated deployment, scaling, and management of containerized applications.

2.5 Why Docker?

We have chosen to use Docker to run Firefox version 52.7.3 for several compelling reasons that enhance our development and deployment processes. Docker offers a containerization solution that ensures consistency across various environments, enabling developers to replicate the same conditions from development to production. This consistency is particularly important when dealing with specific software versions, such as Firefox 52.7.3, ensuring compatibility and eliminating the "works on my machine" problem.

Docker containers are lightweight and efficient, allowing us to bundle all necessary dependencies and configurations with Firefox, creating an isolated environment that runs seamlessly on any system with Docker installed. This approach simplifies the management of dependencies and mitigates potential conflicts that could arise from differing software environments. Additionally, Docker's portability facilitates easy scaling and deployment, enabling us to quickly spin up multiple instances of Firefox for testing or scaling purposes without the overhead of traditional virtual machines.

Moreover, Docker enhances security by isolating applications, reducing the risk of interference from other processes on the host system. This isolation is crucial for maintaining the integrity and security of our CMS website, especially when dealing with outdated or legacy software like Flash Player.

In summary, using Docker to run Firefox 52.7.3 provides a robust, consistent, and efficient solution that streamlines our development workflow, enhances security, and ensures compatibility, ultimately improving the overall stability and performance of our CMS website.

- **Difference Between Docker and Virtualization:**

Docker	Virtualization
It boots in a few seconds.	It takes a few minutes for VMs to boot.
Pre-built docker containers are readily available.	Ready-made VMs are challenging to find.
Docker has a complex usage mechanism consisting of both third-party and docker-managed tools.	Tools are easy to use and more straightforward to work with. third-party.
Limited to Linux.	Can run a variety of guest OS.
Dockers make use of the execution engine.	VMs make use of the hypervisor.
It is lightweight.	It is heavyweight.
Host OS can be different from container OS.	Host OS can be different from guest OS.
Can run many docker containers on a laptop.	Cannot run more than a couple of VMS on an average laptop.
Docker can get a virtual network adapter. It can have separate IPs ad Ports.	Each VMS gets its virtual network adapter.
Sharing of files is possible.	Sharing library and files are not possible.
Lacks security measures.	Security depends on the hypervisor.
A container is portable.	VMS is dependent on a hypervisor.
It allows running an application in an isolated environment known as a container	It provides easiness in managing applications, recovery mechanisms, and isolation from the host operating system

Table no - 4

2.6 Solution Proposal:

To Address the challenges posed by the outdated technology used in the 15m Telescope monitor and control (CMS) system, a modernization strategy leveraging Docker technology can be implemented. Docker provides a lightweight, portable, and scalable solution for packaging, distributing, and running applications across different environments. Centos 7 can be used because it is derived from the sources of Red Hat Enterprise Linux (RHEL) and has docker compatibility.

- **Modernizing the Firefox and Flash Components:** Docker can be utilized to containerize the Firefox browser along with the required Flash components. This approach allows Firefox and Flash to run consistently across different operating systems without dependencies on the host system's environment. By encapsulating Firefox and Flash within Docker containers, compatibility issues and dependency concerns are minimized, ensuring a seamless and reliable browsing experience for the M&C system.

- **Compiling the CMS System Using Docker:** The CMS system can be compiled within a Docker container, providing a standardized and reproducible build environment. By creating a Docker image with the necessary build tools, libraries, and dependencies, the compilation process becomes platform-independent. This Docker image can then be used to compile the CMS system on any server, regardless of the underlying operating system. Additionally, Docker's layering mechanism facilitates efficient caching of build artifacts, speeding up the compilation process and enhancing development productivity.

- **Deployment and Distribution:** Once the CMS system is compiled and packaged using Docker, the resulting artifacts can be deployed to any server with Docker installed. Docker images provide a consistent deployment environment, ensuring that the CMS system runs reliably across diverse infrastructure setups. Furthermore, Docker Hub or a private Docker registry can be utilized for storing and distributing the Docker images, enabling seamless deployment and updates across multiple servers.

By leveraging Docker for both running the Firefox browser with Flash and compiling the CMS system, the 15M CMS system can overcome the limitations imposed by outdated technology. This modernization approach enhances flexibility, scalability, and maintainability, ultimately contributing to the efficiency and effectiveness of the 15m operations for astronomical research and observation.

3 . Implementation

3.1 About Web based NCRA 15m Control & Monitoring System (CMS):

The NCRA 15m antenna is an Alt-Azimuth mounted parabolic dish located at Pune. The web-based CMS is suitable for this antenna and as well as for the 2 Meter Optical telescope of IUCAA (Inter University Centre for Astronomy & Astrophysics). The CMS is based on an architecture which is sufficiently generic to cater to both types of telescopes with most of the differences in specification captured as configuration parameters. The system was developed around 2011.

Figure 2 shows a high level architecture diagram of the 15m antenna CMS system. The spring frame-work, used along with the web container, provides a runtime environment for the web functionalities like security, concurrency, life-cycle management, transaction and other services. The spring web controllers route the request from the C&M System user interface (CMS-UI) to the Core System Services. The CMS-UI display web-page consists of HTML and Flex components which handle the HTTP requests and responses from the telescope users. The web-server application consists of core system services that accept and validate the requests from the controller, and pass them to the appropriate service components such as batch processing, state-machine and data layer objects. The core system also provides dynamic service level integration and implementation such as user and user-group or role management, catalog management etc. The state-manager is responsible for overall behaviour of the M&C system and keeps updates of live status of all the telescope sub-systems like servo, signal conditioning, sentinel and back-end processing units. Upon raising of alarms or exceeding the threshold levels of telescope parameters, state-machine takes corrective actions to restore the telescope to normal operations.

The core-system services & state machine uses XML specifications, self-description and rule files which increase the level of configurability of the M&C system. The communication and messaging layers mainly handle the requests / responses, asynchronous events and monitoring data between the CMS and telescope subsystems wrappers using message queues and socket communication. The wrapper software is generic and configurable, and communicates to all subsystems of the telescopes. The wrapper decodes the XML packets received from the CMS, passes to hardware devices to take action, and after gathering the responses from devices, converts them to XML packets to send to the CMS.

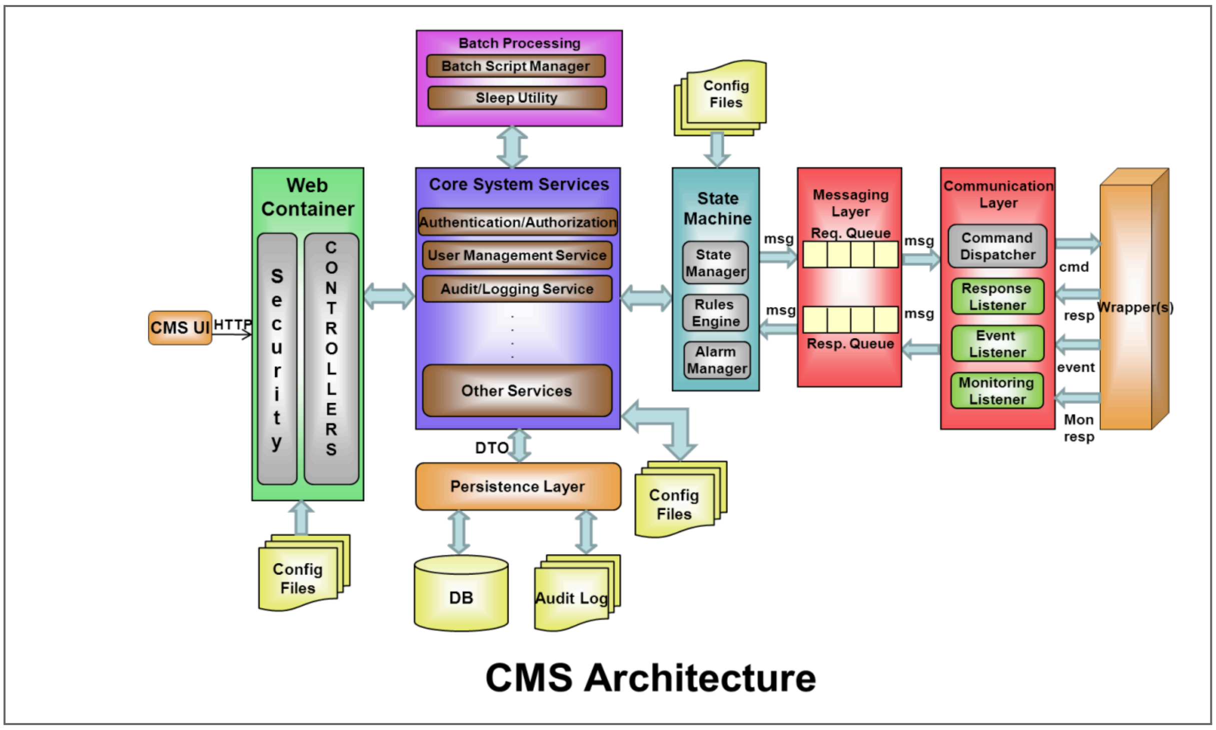


Figure 6 : The high level architecture of Control & Monitoring System of NCRA 15m Antenna

Salient Features: Our web-based CMS frame-work is an end-to-end software solution for the radio and optical telescopes M&C system which has the salient features like-

I. Context based web-browser interface
II. Modular and allow easier hardware/software upgrades
III. Scalable and configurable (via xml-DTD specification)
IV. Automatic state-restoration, exception handling using the batch/scripts
V. Gives alarms notifications through the audio, visual and emails.

At present the CMS system runs on the PowerEdge Dell 310 servers with RHEL 6 OS. The CMS has been in usage since past decades, both the Server and OS has reached its end of life, and hence raise the risk of machine failure and no more support of RHEL 6 is available. Therefore, we chose the CentOS 7 operating system on a new machine to implement the CMS. First the all spring frame required software like Tomcat web-server, ActiveMQ message passing system, Apache Maven , ant compiler and Java (1.6.0_30) were installed on the CenOS 7. Along with this all requisite system jar libraries along with Eclipse were installed on CentOS 7 were ensured to be working.

The platform CentOS 7 as a basic image for the docker then finalized, and centOS 7 Docker image from the available open-source repository system was installed. The following section gives the docker image installation details for a Linux and Windows OS platform.

3.2 Docker Installation:

Docker is an open platform for developing, shipping, and running applications.

Docker allows you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications.

By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

Docker can be downloaded by following the steps mention on the official docker website : https://docs.docker.com/get-docker/

Docker System Requirements:

1. **Linux:**
    a. 64-bit kernel and CPU support for virtualization.
    b. KVM virtualization support. Follow the KVM virtualization support instructions to check if the KVM kernel modules are enabled and how to provide access to the KVM device.
    c. QEMU must be version 5.2 or later. We recommend upgrading to the latest version.
    d. systemd init system.
    e. Gnome, KDE, or MATE Desktop environment.
        i. For many Linux distros, the Gnome environment does not support tray icons. To add support for tray icons, you need to install a Gnome extension. For example, AppIndicator.
    f. At least 4 GB of RAM.
    g. Enable configuring ID mapping in user namespaces, see File sharing.
    h. Recommended: Initialize pass for credentials management**.**
2. **Windows:**
    a. WSL version 1.1.3.0 or later.
    b. Windows 11 64-bit: Home or Pro version 21H2 or higher, or Enterprise or Education version 21H2 or higher.
    c. Windows 10 64-bit:
        i. We recommend Home or Pro 22H2 (build 19045) or higher, or Enterprise or Education 22H2 (build 19045) or higher.

ii. Minimum required is Home or Pro 21H2 (build 19044) or higher, or Enterprise or Education 21H2 (build 19044) or higher.

d. Turn on the WSL 2 feature on Windows. For detailed instructions, refer to the Microsoft documentation.

e. The following hardware prerequisites are required to successfully run WSL 2 on Windows 10 or Windows 11:

i. 64-bit processor with Second Level Address Translation (SLAT)
ii. 4GB system RAM
iii. Enable hardware virtualization in BIOS. For more information, see Virtualization.

3.3 Implementation of Firefox 52.7.3 and flash player:

- **Objective :**
  The objective of this setup was to create a CentOS 7 environment within a Docker container, install Firefox version 52.7.3 with all necessary dependencies, and configure Flash Player to enable the use of a web-based control and monitoring system for a 15-meter radio telescope.

- **Step-by-Step Process:**

- **Step 1: Download Docker Image of CentOS 7:**

  1. A Docker image of CentOS 7 was downloaded from a reliable repository.
  2. This provides a lightweight and isolated environment for running applications.
  3. Centos 7 docker images can be installed using the command `Docker pull` command.
  4. Command executed: `sudo docker pull centos:7`

- **Step 2: Update Yum Package in CentOS 7 Docker Image:**

  1. The Yum package manager was updated to ensure all system packages were current.
  2. Commands executed: `sudo yum update`

- **Step 3: Installing Latest Firefox and Uninstalling:**

This step is done to ensure that all the libraries needed to run the Firefox browser are installed automatically.

1. Command Executed:
   i. >`sudo yum install firefox`
   ii. >`sudo yum remove firefox`

- **Step 4: Install Firefox 52.7.3 with Dependencies:**

  1. Firefox version 52.7.3 was chosen due to its compatibility with Flash Player.
  2. Necessary dependencies were identified and installed to ensure proper functionality.
  3. Firefox 52.7.3 can be installed using the rpm package .
  4. Commands executed:
  5. >`sudo yum install Firefox_52_7_3.rpm .`

- **Step 5: Add Flash Player Plugins to Firefox:**

1. Adobe Flash Player plugins were downloaded and installed to enable Flash content in Firefox.
2. The flash player plugins then were copied inside the plugins folder of the browser:
3. >`root/.mozilla/plugins`
4. If the plugins folder is not present inside the .mozilla folder it can be created by using the command.
5. >`mkdir -p /root/.mozilla/plugins`
6. Flash Player was then configured within Firefox to ensure it was recognized and enabled.

- **Step 6: Access and Use Control and Monitoring System:**

1. The web-based control and monitoring system for the 15-meter radio telescope was accessed via Firefox.
2. This system relies on Flash Player for its graphical user interface (GUI).
3. Firefox successfully opened the system, allowing full functionality for control and monitoring tasks.

- **Conclusion:**
  The setup was completed successfully, allowing the control and monitoring system of the 15-meter radio telescope to function as intended within a CentOS 7 Docker environment. The use of Firefox 52.7.3, combined with the Flash Player plugin, facilitated the GUI-based operations required for the system.

This setup ensures a stable and isolated environment for managing the radio telescope, leveraging Docker's containerization capabilities.

3.4 Deployment of the CMS Software:

The 15m telescope CMS uses a java based SPRING framework along with a Tomcat Web-application server, ActiveMQ message passing system along with the many dependencies components. The description of required software components, libraries are mentioned as below :

3.4.1 Software tools and Libraries required for the CMS software:

1) **Spring framework(2.5):**

    a) a. Used as MVC framework for web based application development Various spring based web controllers are the entry point for processing of a user request. Following are few controllers CMS have used –

2) **Blaze DS (4.0):**

    a) Used to integrate flex with java based web applications and is used to update the data on various flex based widgets in CMS. This library is used only for integration purposes and does not generate any output files as such. Refer to following link for step by step guide about integrating BlazeDS into web application –

3) **Hibernate (3.2.6):**

    a) ORM framework, used for rapid development of database interactions with applications. This framework is used to hide the database level complexities from application, facilitating rapid application development. This acts as an interface between application and actual database. Various DAO classes in CMS make use of this framework.

    b) Following are few configuration files used in CMS along with brief usage description –

        i) Alarm.hbm.xml – Maps the Alarm object from application domain to “t_alarm” table on database side

        ii) User.hbm.xml - Maps the User object from application domain to “t_alarm” table on database side

4) **Flex (4.0):**

   a) It is used to create various widgets on CMS UI, which are then updated with dynamic data using push technology provided by BlazeDS

5) **Sleep Utility:**

   a) A customized batch scripting framework. Modified to suit CMS requirements like

      i) Executing commands from batch file

      ii) Conditional, looping logic

6) **Physhun (0.5.1):**

   a) Used for building and executing processes as finite State Models in J2SE and J2EE environments in CMS.

7) **MySQL (5.1):**

   a) Used as a persistent store by CMS. CMS interacts with databases via the Hibernate framework through various DAO (Data Access Object) classes in CMS. Following are few DAOs used in CMS

8) **Maven (3.0):**

   a) Build tool to build entire application. Maven is an open-source build automation and project management tool widely used for Java applications.

9) **Active MQ (5.4.3):**

   a) Messaging framework used to enqueue and dequeue the messages to increase the scalability of overall CMS. Please refer to the following URL for step by step information about Spring ActiveMQ configuration.

10) **Tomcat (6.0.35):**

   a) Tomcat is an open-source web server and servlet.It is used widely for hosting Java-based applications on the web.

11) **Java (1.6.30):**

   a) Java platform to develop java based application

12) **Style-sheet transformation (XSLT):**

a) To transform XML documents into HTML web page, thus adding capability to modify the page contents by simple change in XML document. Please refer to the "Dynamic UI Generation" section in Developer Guide for further details about this.

13)Castor (1.3.2):

a) a. It is an XML data binding framework, to convert xml to java objects and vice versa. It uses xml configuration files to convert a java object into xml and vice versa. Following are few of such files used in CMS –

14)JNA – Java Native Access:

a) Used to invoke native APIs from java, this is used in CMS to call various apis in tact calculation library.

3.4.2 Steps to compile CMS software in docker:

1. **Objective:**
   a. Create a Docker image that contains the necessary environment and compiled CMS software, ensuring it is portable and can run on any system that supports Docker.
2. **Prerequisites:**
   a. Operating System:
      i. CentOS 7 : The base operating system for the Docker host.

   b. Docker:
      i. **Docker:** Platform for developing, shipping, and running applications inside containers.
      ii. Centos7

   c. Software Packages and Libraries:
      i. **Apache Maven 3.0.4**: A build automation tool primarily used for Java projects. Install using the package manager or download from the official website.
      ii. **Apache ActiveMQ 5.5.1**: A message broker written in Java together with a full JMS client.
      iii. **Apache Ant 1.7.1**: A Java-based build tool used for automating software build processes.

iv. **Apache Tomcat 6.0.35**: An open-source implementation of the Java Servlet, JavaServer Pages, and Java Expression Language technologies.

v. **Spring Framework 3.1.0:** A comprehensive programming and configuration model for modern Java-based enterprise applications.

vi. **Eclipse 3.6.1:** An integrated development environment (IDE) used in computer programming.

vii. **Drools 2.5:** A business rule management system (BRMS) solution.

viii. **Java Runtime Environment (JRE) 1.6.30:** A prerequisite for running Java applications.

ix. **Java Development Kit (JDK) 1.6.30:** A software development environment for developing Java applications and applets.

x. **MySQL Client:** Command-line tool for connecting to MySQL databases. Ensure the client is installed to facilitate database operations.

xi. **GCC-G++ 4.4.6:** The GNU Compiler Collection, including the C++ compiler.

xii. **GCC Fortran:** The GNU Fortran compiler, part of the GNU Compiler Collection.

xiii. **Memory Analyser:** A tool for analyzing memory usage and detecting memory leaks.

xiv. **Physhun:** Used for building and executing processes as finite State Models in J2SE and J2EE environments in CMS.

xv. **JDOM**: A Java-based document object model for XML parsing

xvi. **M2 Repository**: A repository for Maven dependencies.

d. Hardware Requirements:

i. **64-bit Processor**: Ensure the system has a 64-bit processor to support CentOS 7 and Docker.

ii. **4GB RAM**: Minimum of 4GB of RAM to ensure smooth operation.

iii. **20GB Free Disk Space**: At least 20GB of free disk space to accommodate CentOS 7, Docker, and all necessary software packages and libraries.

3. **Steps to Compile the Software into Docker:**

   a. **Pull a CentOS 7 Docker Image:**

      i. >`docker pull centos:7`

   b. **Create a Docker Container from the CentOS 7 Image:**

      i. >`sudo docker run -it --net host --ulimit nofile=300000:300000 -m 3g --env="DISPLAY" --volume="/tmp/.X11-unix:/tmp/.X11-unix:rw" cmssoft:v2.0`

   c. **Update the Mirrorlist as CentOS 7 Has Reached End of Life (EOL):**

      i. >`sed -i s/mirror.centos.org/vault.centos.org/g /etc/yum.repos.d/*.repo`

      ii. >`sed -i s/^#.*baseurl=http/baseurl=http/g /etc/yum.repos.d/*.repo`

      iii. >`sed -i s/^mirrorlist=http/#mirrorlist=http/g /etc/yum.repos.d/*.repo`

   d. **Install the Required Packages and Software Referenced in Prerequisites Inside the /opt Directory in the Docker Container**

   e. **Create the Required System Links:**

      i. >`ln -s /opt/ncra/sharedlib /usr/ncra/sharedlib`

      ii. >`ln -s /opt/ncra/lib /usr/ncra/lib`

      iii. >`ln -s /opt/apache-tomcat-6.0.35 /usr/share/tomcat`

   f. **Source the PATH File in /opt Directory:**

      i. >`source /opt/PATH`

   g. **Run Eclipse and Compile the CMS Software:**

      i. >`eclipse`

      ii. create a workspace

iii. Open the project Folder and make a clean build

h. Copy the CMS.war File Inside the Tomcat Using copy.sh Scrip :

i. >`cd /opt`

ii. >`./copy.sh`

i. Run the startupshellscript.sh Script to Start Tomcat, ActiveMQ, and CMS Web App :

i. >`startupshellscript`

j. Assign the Servo, Antenna Frontend, Antenna Backend, and Antenna Server IP to the Container :

i. Replace hardcoded IP addresses with container IP within the frontend, backend, antenna server, and MySQL scripts to enable proper communication between Docker containers.

3.5 Production version :

3.5.1 The Web-based (Firefox) CMS UI:

Scripts have been developed for Linux that automate the process of downloading Docker, installing an image containing Firefox 52.7.3 with Flash Player, and running it. These scripts streamline the setup by eliminating manual installation steps, ensuring a consistent and efficient environment for our CMS website. By leveraging Docker, the scripts create isolated containers that encapsulate Firefox and its dependencies, maintaining compatibility and stability across different systems. This automation not only saves time but also reduces potential errors, enhancing the reliability of our web applications and providing a seamless user experience.

3.5.1.1 For linux :

- **How to use**

1. Open the terminal in linux-script folder and run the following command
   >`chmod +x runfirefoxdocker.sh`
2. after executing the above command execute the script using the following command inside the terminal-
   >`./runfirefoxdocker.sh`

- **Logic of scripts**

1. **`runfirefoxdocker.sh`** - Check whether docker is running or not. if docker is not running redirects to packagechecker.sh else loads the cms15m:v1 image and redirects to cmsfirefox.sh

2. **`cmsfirefox.sh`** - terminate the container and starts a new container with firefox
3. **`packagechecker.sh`** - its checks which package manager the linux machine is running, apt or yum and redirects to aptinstaller.sh or yuminstaller.sh accordingly.
4. **`aptinstaller.sh`** - installs docker and installs the docker image present inside the folder then redirects to `docker_status.sh`
5. **`yuminstaller.sh`** - installs docker and installs the docker image present inside the folder then redirects to `docker_status.sh`

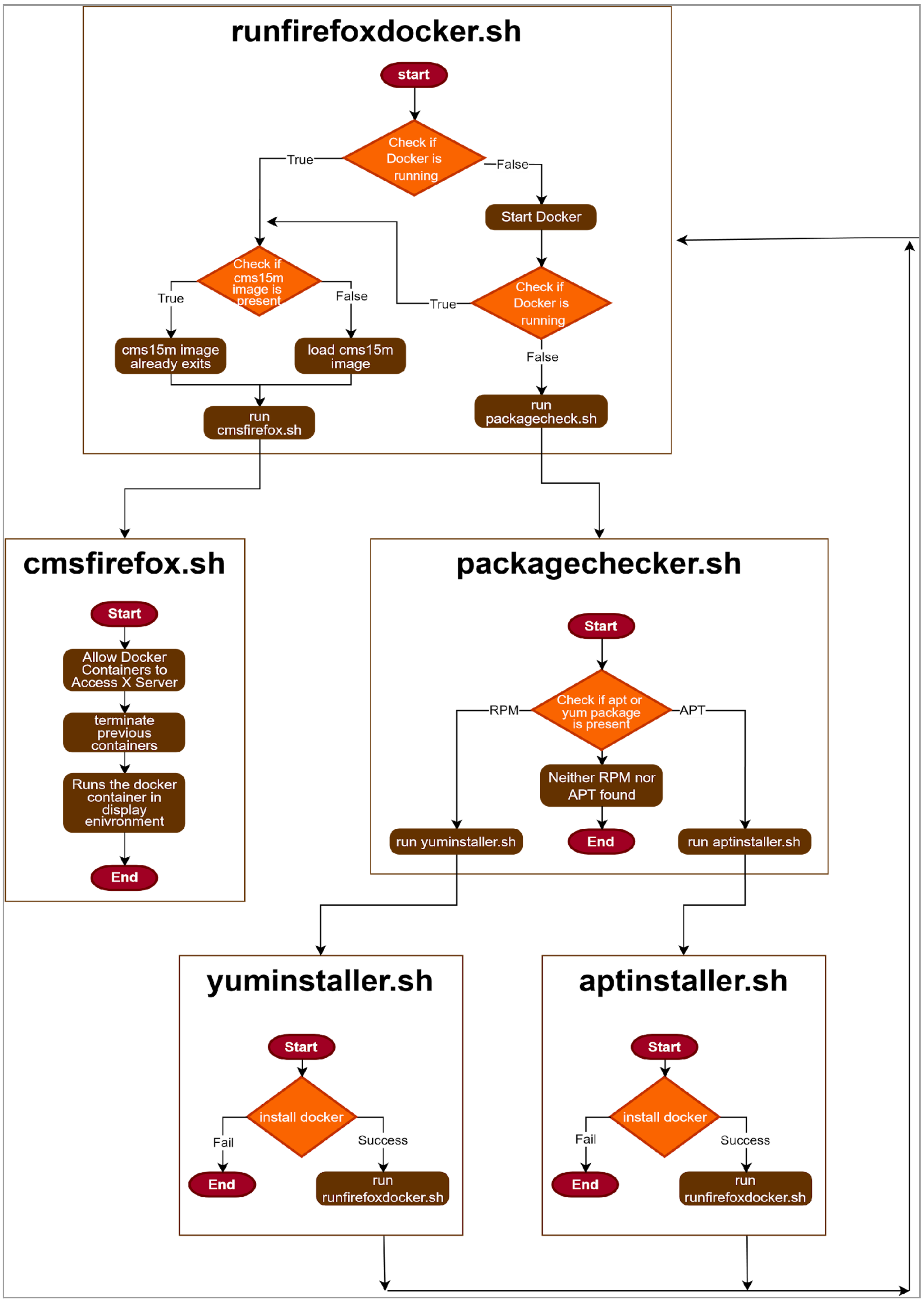


Figure-7: Logic Diagram of Script to run firefox with flash on Linux.

3.5.1.2 For Windows :

- **How to use**

  1. Run the **`cmsfirefox.bat`** file inside the folder.

- **Logic of script :**

  1. **`runfirefox.bat` -** Set PowerShell execution policy to unrestricted for the current user and run (`rundocker.ps1`).
  2. **`check_docker_running.ps1` -** Checks if docker daemon is running.
  3. **`checkwsl.ps1`** - Checks if WSL is installed and determines whether the default version is set to 1 or 2. If WSL is not installed, attempts to install it and sets the default version to 2.
  4. **`rundocker.ps1` -** uses (`check_docker_running.ps`)1 to verify if the Docker daemon is running. If the daemon is not running, it attempts to start it. If starting the daemon fails, it runs (`dockerinstaller.ps1`) to download and install Docker. Once the Docker daemon is running, it checks if the csm15m/v_1.0:v1.0 image exists. If the image is not found, it loads the image and then runs (`runxming.ps1`.)
  5. **`dockerinstaller.ps1`** - attempts to install docker from the (Docker_Desktop_Installer.exe) and redirect to the (`rundocker.ps1`).
  6. **`runxming.ps1` -** checks if Xming is running. If Xming is not running, it attempts to start it. If starting Xming fails, it redirects to (`xminginstaller.ps1`) to install Xming. Once Xming starts successfully, it redirects to (`cmsfirefox.ps1`.)
  7. **`xminginstaller.ps1` -** attempts to install xming from the (Xming-6-9-0-31-setup.exe) and redirects to (`runxming.ps`1).
  8. **`cmsfirefox.ps1` -** removes any previously running from image `csm15m/v_1.0:v1.0` and starts a new container.

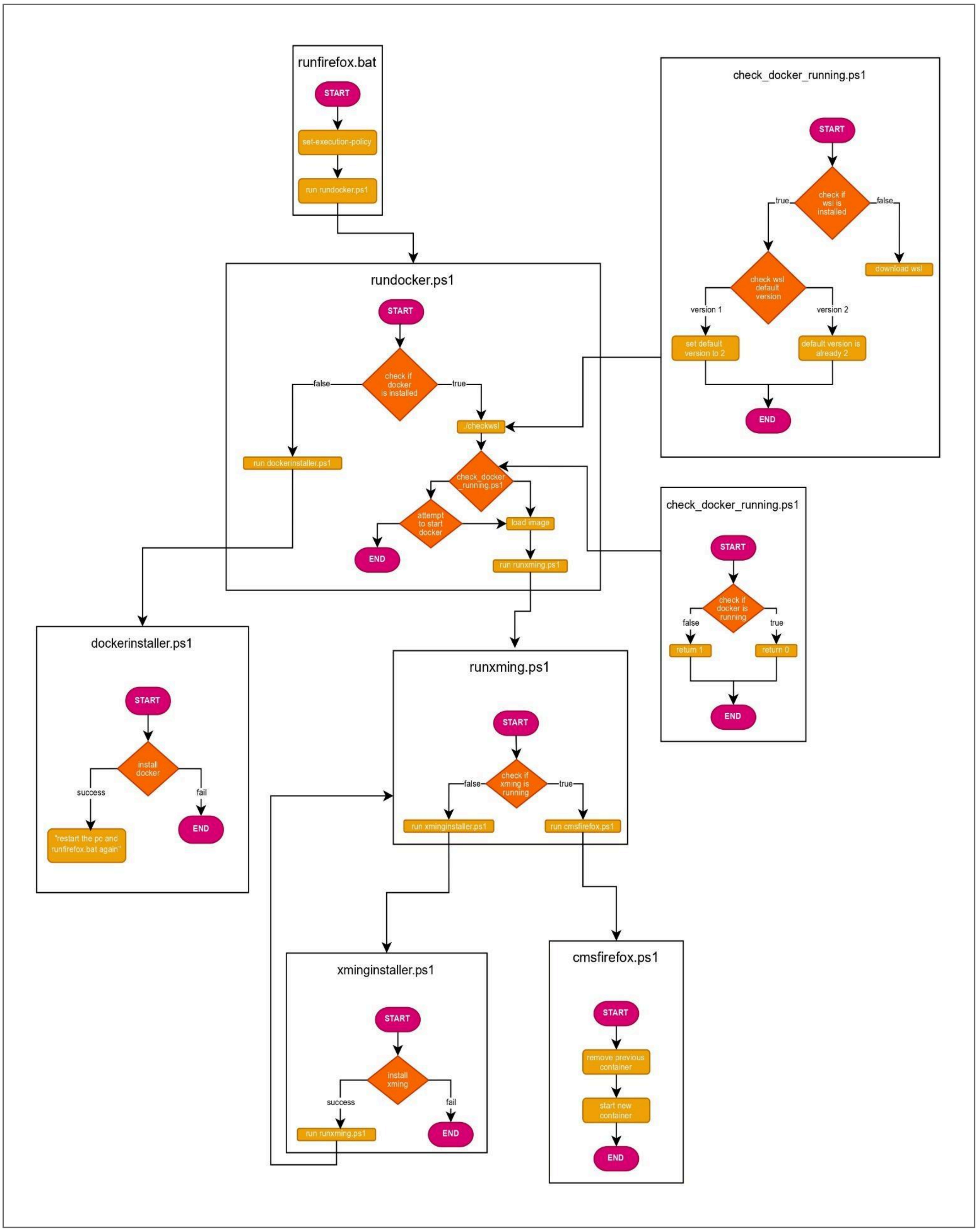


Figure-8: Logic Diagram of Script to run firefox with flash on windows.

3.5.2 CMS-Web Software Deployment:

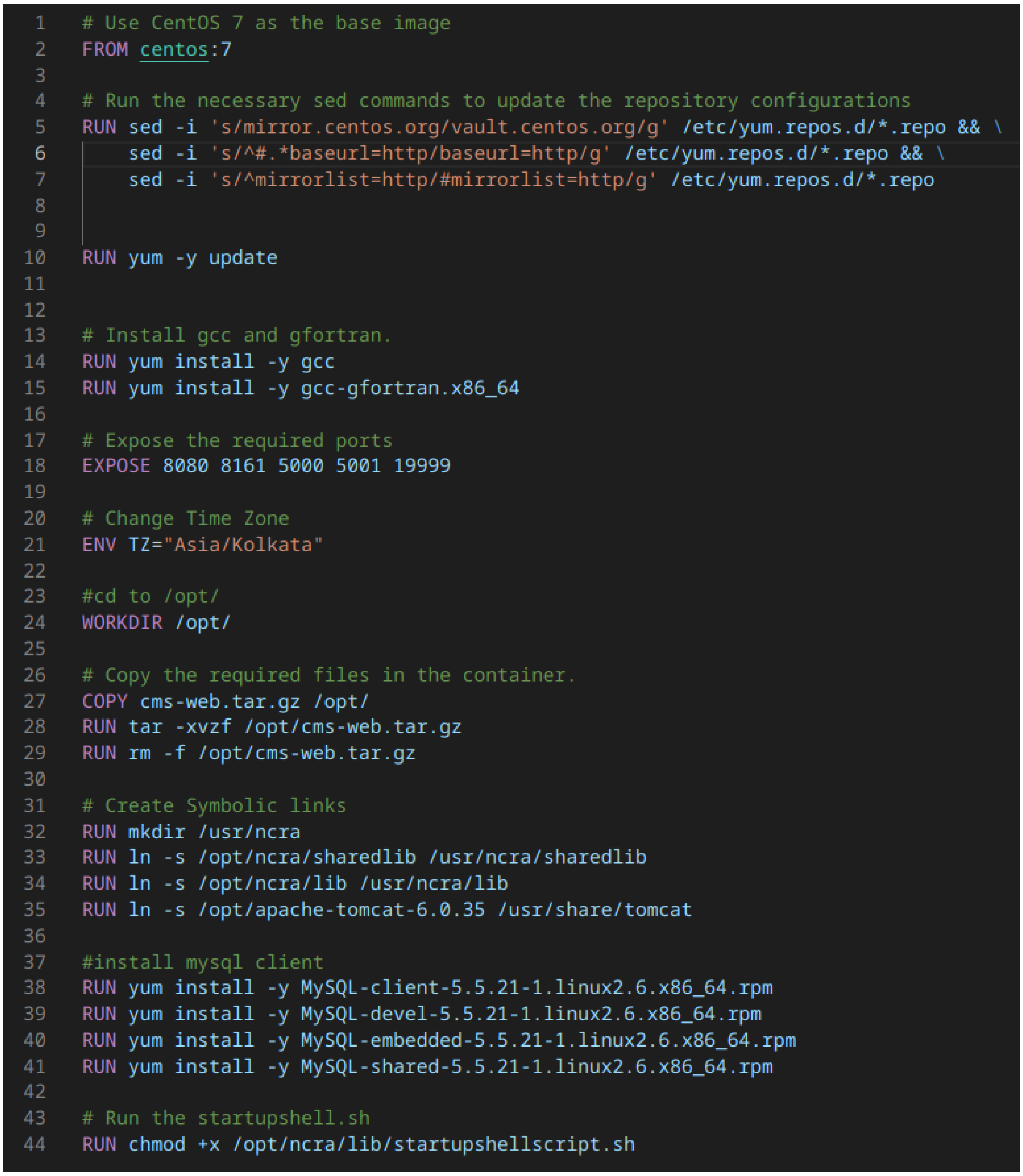
To deploy the CMS-Web software, a Dockerfile is utilized to create a minimal Docker image. This image can be built using the following command:

```
> docker build -t [imagename]:[tag] .
```

Once the image is successfully built, a Docker container can be created and run with the following command:

```
> sudo docker run -it --net host -e TZ=Asia/Kolkata --ulimit
nofile=300000:300000 -m 3g --name [CONTAINER_NAME]
--env="DISPLAY" --volume="/tmp/.X11-unix:/tmp/.X11-unix:rw"
[imagename]:[tag]
```

This process ensures a consistent and efficient deployment of the CMS-Web software, encapsulating all necessary dependencies and configurations within the Docker container. The use of Docker simplifies the deployment process, providing an isolated environment that is easy to manage and maintain.



```
# Use CentOS 7 as the base image
FROM centos:7

# Run the necessary sed commands to update the repository configurations
RUN sed -i 's/mirror.centos.org/vault.centos.org/g' /etc/yum.repos.d/*.repo && \
    sed -i 's/^#.*baseurl=http/baseurl=http/g' /etc/yum.repos.d/*.repo && \
    sed -i 's/^mirrorlist=http/#mirrorlist=http/g' /etc/yum.repos.d/*.repo


RUN yum -y update


# Install gcc and gfortran.
RUN yum install -y gcc
RUN yum install -y gcc-gfortran.x86_64

# Expose the required ports
EXPOSE 8080 8161 5000 5001 19999

# Change Time Zone
ENV TZ="Asia/Kolkata"

#cd to /opt/
WORKDIR /opt/

# Copy the required files in the container.
COPY cms-web.tar.gz /opt/
RUN tar -xvzf /opt/cms-web.tar.gz
RUN rm -f /opt/cms-web.tar.gz

# Create Symbolic links
RUN mkdir /usr/ncra
RUN ln -s /opt/ncra/sharedlib /usr/ncra/sharedlib
RUN ln -s /opt/ncra/lib /usr/ncra/lib
RUN ln -s /opt/apache-tomcat-6.0.35 /usr/share/tomcat

#install mysql client
RUN yum install -y MySQL-client-5.5.21-1.linux2.6.x86_64.rpm
RUN yum install -y MySQL-devel-5.5.21-1.linux2.6.x86_64.rpm
RUN yum install -y MySQL-embedded-5.5.21-1.linux2.6.x86_64.rpm
RUN yum install -y MySQL-shared-5.5.21-1.linux2.6.x86_64.rpm

# Run the startupshell.sh
RUN chmod +x /opt/ncra/lib/startupshellscript.sh
```

Figure-9 : Dockerfile

4. Validation Testing

4.1 Aim:

The purpose of this validation testing document is to verify that the NCRA 15m Control Monitoring System (CMS) software meets the specified requirements and functions correctly within the Docker container environment. This document outlines the testing methodology, test cases, expected results, and actual results.

4.2 Test Environment:

4.2.1 Hardware Configuration :

- **Processor** : intel i7 64-bit
- **RAM** : 16GB
- **Disk Space** : 30 GB free space

4.2.2 Software Configuration :

- **Operating System** : CentOS 7
- **Docker** : Latest version compatible with CentOS 7
- **Software Packages** : As listed in the prerequisites section

4.2.3 Network Configuration :

Name	IP Address	Ports
Servo	192.168.160.2	9000
testservo	192.168.160.2	9005
Sentinel	192.168.160.3	8000
frontend	192.168.160.3	8571
nbackend	192.168.160.8	9571
backend	192.168.160.6	9571
sigcon	192.168.160.3	7570
dataserver	192.168.160.6	9572
tomcat	192.168.160.33	8080

activemq	192.168.160.33	8161

Table no-5

4.3 Functional Testing of the CMS running under Docker :

4.3.1 Test Cases :

1. **Test Case 1 : Running cms-web application**
   a. **Objective:** Verify the running of Tomcat, ActiveMQ, and CMS web app.
   b. **Steps:**
      i. Run the `startupshell.sh` script.
   c. **Expected Result:** Tomcat, ActiveMQ, and CMS web app start without issues.
   d. **Actual Result:** [Pass] - Comments if any**.**
2. **Test Case 2 : Network Configuration**
   a. **Objective:** Verify the network configuration of the Docker container.
   b. **Steps :**
      i. Assign the servo, antenna frontend, antenna backend, and antenna server IPs to the container.
   c. **Expected Result:** Container is correctly configured with the necessary IP addresses.
   d. **Actual Result:** [Pass] - Comments if any.
3. **Test Case 3 : Basic Functionality of CMS-Web:**
   a. **Objective:** Verify the basic functionality of the CMS-web application.
   b. **Steps :**
      i. Access the CMS-web application via a web browser.
      ii. Navigate through various sections of the application.
   c. **Expected Result:** CMS-web application functions correctly with all basic features working.
   d. **Actual Result:** [Pass] - Comments if any.
4. **Test Case 4 : Connectivity with MySQL :**
   a. **Objective :** Verify if the docker container is able to connect with the MySQL server.
   b. **Step :**
      i. Install the MySQL client in the docker image
      ii. Check if it is possible to login on the web app
   c. **Expected Result :** Login in to the cms-web is successful and the app can connect with the database.
   d. **Actual result : [PASS]**
5. **Test Case 5 : Time and Date :**

   a. **Objectives :** Verify if container's time and date is similar to the Host System
   b. **Steps :**
      i. Check the date and time and container with respect to the host machine.
      ii. Use `-e TZ=Asia/Kolkata` flag while running a docker container.
   c. **Expected Result :** The container time matches the host time
   d. **Actual Result :** [Pass]
6. **Test Case 6 : Antenna Movement :**
   a. **Objective :** Verify the functionality of the antenna motors and brakes.
   b. **Steps:**
      i. Initiate the antenna movement commands from the control interface.
      ii. Observe the physical movement of the antenna.
   c. **Expected Results :** Antenna motors and brakes operate correctly without issues.
   d. **Actual Results :** [Pass]
7. **Test Case 7 : Antenna Tracking :**
   a. **Objectives :** Verify the antenna's ability to track objects correctly.
   b. **Steps :**
      i. Select a known celestial object or test signal.
      ii. Command the antenna to track the object.
      iii. Monitor the tracking accuracy and response.
   c. **Expected Result:** Antenna tracks the object correctly and maintains accurate alignment.
   d. **Actual Result :** [Pass]
8. **Test Case 8 : Sub-systems Connectivity**
   a. **Objective:** Verify connectivity and communication with all relevant subsystems.
   b. **Steps :**
      i. Establish connections with the Servo, Common-Box, Digital Backend, and Data server.
      ii. Issue commands and monitor responses.
      iii. Verify data flow and communication stability.
   c. **Expected Result:** All subsystems communicate successfully and commands are executed within ~3 seconds.
   d. **Actual Result:** [Pass]
9. **Test Case 9 : Receiver Settings**

a. **Objectives :** Verify the receiver settings for the frontend and backend systems.

b. **Steps :**

   i. Set the Frontend Radio Frequency to 1400 MHz.

   ii. Set the Common-Box attenuation to 15 dB and 0 dB.

   iii. Configure the GAB Local Oscillator to 1330 MHz.

   iv. Set the Digital Backend sampling to 1 second.

   v. Observe the spectrum plot.

c. **Expected Results :** Receiver settings are applied correctly, and the spectrum plot displays expected results.

d. **Actual Result :** [Pass]

Test Case	Objective	Expected Result	Actual Result
TC1	**Running cms-web application**	Tomcat, ActiveMQ, and CMS web app starts without issues.	PASS
TC2	**Network Configuration**	Container is correctly configured with the necessary IP addresses.	PASS
TC3	**Basic Functionality of CMS-Web:**	CMS-web application functions correctly with all basic features working.	PASS
TC4	**Connectivity with MySQL**	Successful connection with MySQL database	PASS
TC5	**Time and Date**	System time and date are correct.	PASS
TC6	**Antenna Movement**	Antenna motors and Brakes work correctly.	PASS
TC7	**Antenna Tracking**	Antenna tracks correctly on .	
TC8	**Sub-systems Connectivity**	Servo, Common-Box, Digital Backend, Data server communicating . Sentinel system networking problem.	PASS

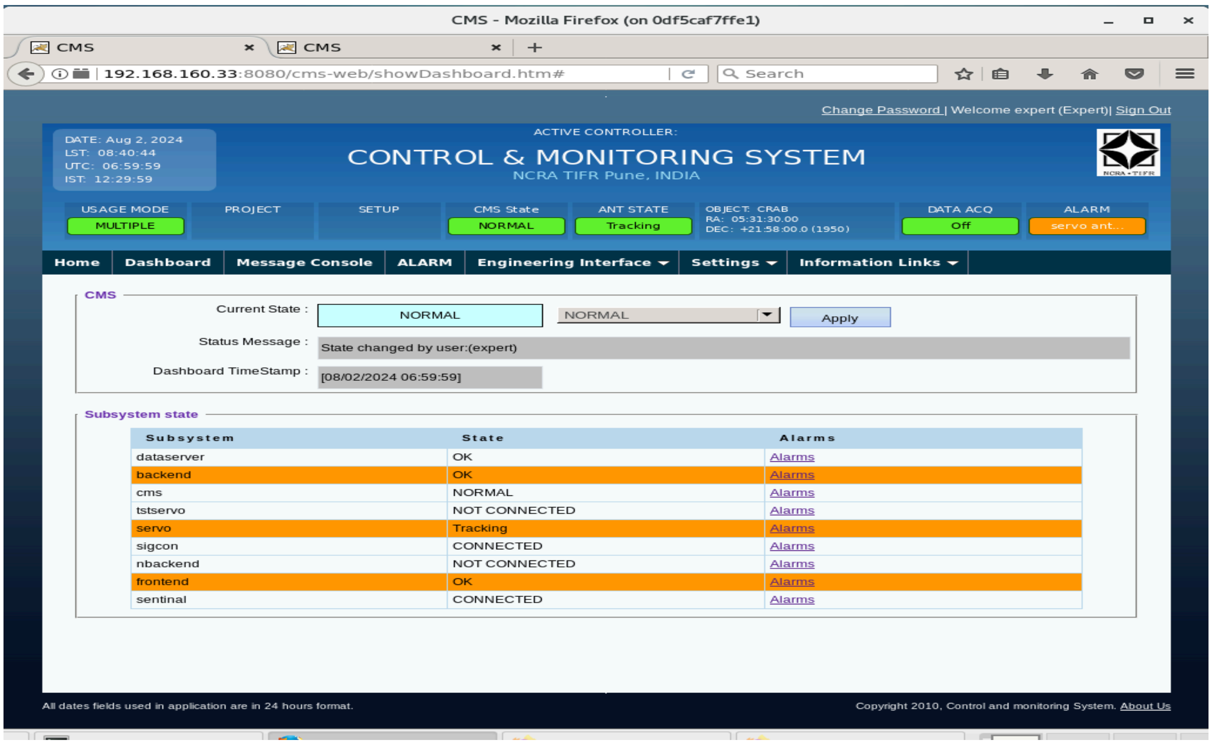
		GAB Configured to connect other system - Command issued are successful within a ~ 3 seconds of time - All Engineering parameters of the subsystems are being monitored.	
TC9	**Receiver Settings**	Frontend Radio Freq = 1400 MHz Common-Box Atten = 15 dB, 0 dB GAB Local Oscillator = 1330 MHz Digital Backend 1 sec sampling ● Spectrum plot seen	PASS

Table no - 6

4.3.2 Functional Test Details :

1. **CMS Status :**

   The CMS running in 'cmssoft V2.0' docker container shows all sub-systems are connected except Sentinel and SigCON (15m Signal Conditioning Analog Backend).


CMS - Mozilla Firefox (on 0df5caf7ffe1)
CMS
CMS
192.168.160.33:8080/cms-web/showDashboard.htm#
Search
Change Password | Welcome expert (Expert)| Sign Out
ACTIVE CONTROLLER:
CONTROL & MONITORING SYSTEM
NCRA TIFR Pune, INDIA
Home
Dashboard
Message Console
ALARM
Engineering Interface
Settings
Information Links
Apply

DATE: Aug 2, 2024
LST: 08:40:44
UTC: 06:59:59
IST: 12:29:59

USAGE MODE: MULTIPLE
PROJECT:
SETUP:
CMS State: NORMAL
ANT STATE: Tracking
OBJECT: CRAB
RA: 05:31:30.00
DEC: +21:58:00.0 (1950)
DATA ACQ: Off
ALARM: servo ant...

CMS
Current State : NORMAL
Status Message : State changed by user:(expert)
Dashboard TimeStamp : [08/02/2024 06:59:59]

Subsystem state

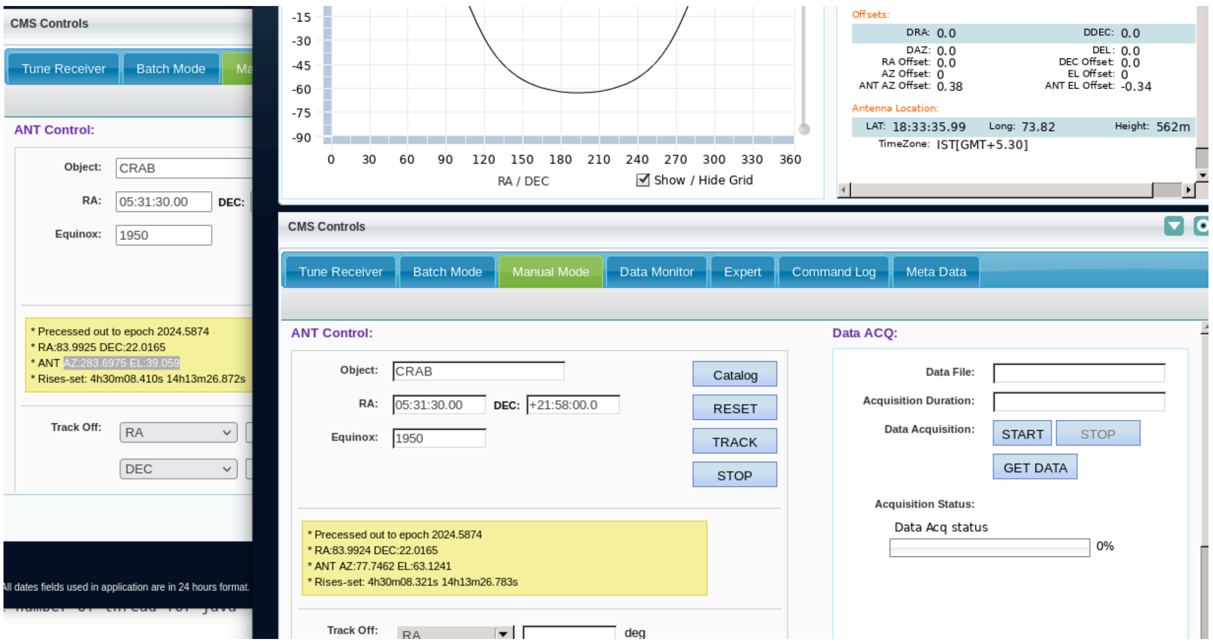
Subsystem	State	Alarms
dataserver	OK	Alarms
backend	OK	Alarms
cms	NORMAL	Alarms
tstservo	NOT CONNECTED	Alarms
servo	Tracking	Alarms
sigcon	CONNECTED	Alarms
nbackend	NOT CONNECTED	Alarms
frontend	OK	Alarms
sentinal	CONNECTED	Alarms

All dates fields used in application are in 24 hours format.

Copyright 2010, Control and monitoring System. About Us

Figure-10

2. **Equatorial to Horizon Coordinate conversion verification for the Source Tracking :**
    a. Because of Docker Image was booting with the UTC time, the Equatorial to Horizontal (Azimuth and Elevation) values were showing difference :



ANT Control:

Object: CRAB
RA: 05:31:30.00 DEC:
Equinox: 1950

* Precessed out to epoch 2024.5874
* RA:83.9925 DEC:22.0165
* ANT AZ:283.6975 EL:39.059
* Rises-set: 4h30m08.410s 14h13m26.872s

Track Off: RA
DEC

All dates fields used in application are in 24 hours format.

Offsets:
DRA: 0.0 DDEC: 0.0
DAZ: 0.0 DEL: 0.0
RA Offset: 0.0 DEC Offset: 0.0
AZ Offset: 0 EL Offset: 0
ANT AZ Offset: 0.38 ANT EL Offset: -0.34

Antenna Location:
LAT: 18:33:35.99 Long: 73.82 Height: 562m
TimeZone: IST[GMT+5.30]

ANT Control:

Object: CRAB
RA: 05:31:30.00 DEC: +21:58:00.0
Equinox: 1950

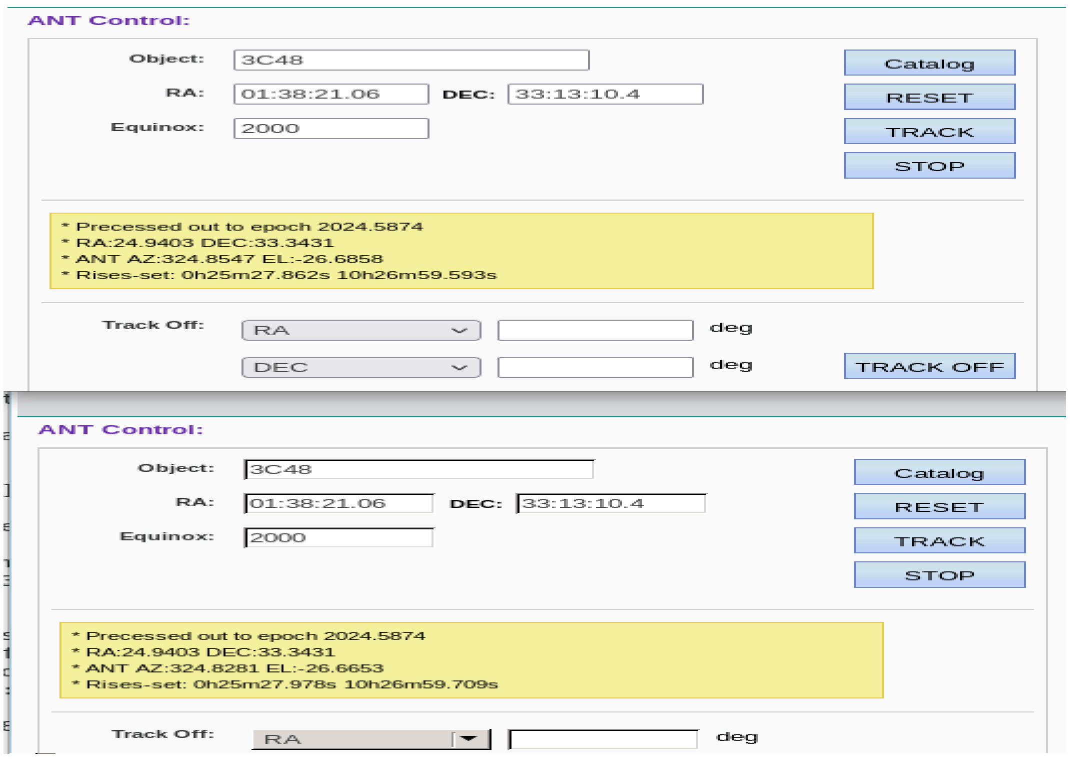
* Precessed out to epoch 2024.5874
* RA:83.9924 DEC:22.0165
* ANT AZ:77.7462 EL:63.1241
* Rises-set: 4h30m08.321s 14h13m26.783s

Track Off: RA deg

Data ACQ:
Data File:
Acquisition Duration:
Data Acquisition:
Acquisition Status:
Data Acq status 0%

Figure-11:

b. After correcting options for Booting the Docker image in the container source, Conversion is taking correctly -



ANT Control:

Object: 3C48
RA: 01:38:21.06 DEC: 33:13:10.4
Equinox: 2000

* Precessed out to epoch 2024.5874
* RA:24.9403 DEC:33.3431
* ANT AZ:324.8547 EL:-26.6858
* Rises-set: 0h25m27.862s 10h26m59.593s

Track Off: RA deg
DEC deg

ANT Control:

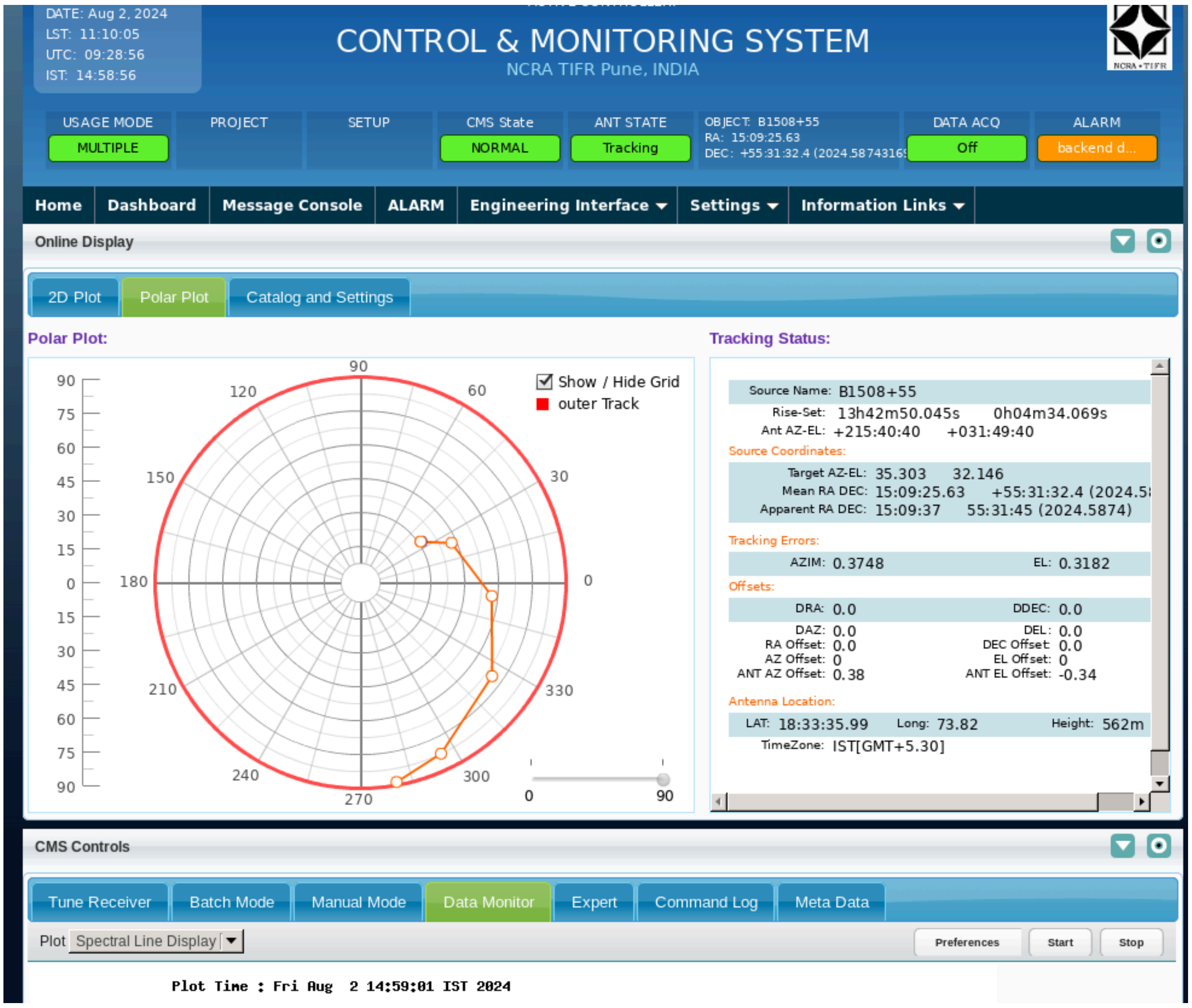
Object: 3C48
RA: 01:38:21.06 DEC: 33:13:10.4
Equinox: 2000

* Precessed out to epoch 2024.5874
* RA:24.9403 DEC:33.3431
* ANT AZ:324.8281 EL:-26.6653
* Rises-set: 0h25m27.978s 10h26m59.709s

Track Off: RA deg

Figure 12

Antenna Tracking Status :



DATE: Aug 2, 2024
LST: 11:10:05
UTC: 09:28:56
IST: 14:58:56

CONTROL & MONITORING SYSTEM
NCRA TIFR Pune, INDIA

USAGE MODE	PROJECT	SETUP	CMS State	ANT STATE	OBJECT: B1508+55	DATA ACQ	ALARM
MULTIPLE			NORMAL	Tracking	RA: 15:09:25.63 DEC: +55:31:32.4 (2024.5874316	Off	backend d...

Tracking Status:

Source Name: B1508+55
Rise-Set: 13h42m50.045s 0h04m34.069s
Ant AZ-EL: +215:40:40 +031:49:40

Source Coordinates:
Target AZ-EL: 35.303 32.146
Mean RA DEC: 15:09:25.63 +55:31:32.4 (2024.5
Apparent RA DEC: 15:09:37 55:31:45 (2024.5874)

Tracking Errors:
AZIM: 0.3748 EL: 0.3182

Offsets:
DRA: 0.0 DDEC: 0.0
DAZ: 0.0 DEL: 0.0
RA Offset: 0.0 DEC Offset: 0.0
AZ Offset: 0 EL Offset: 0
ANT AZ Offset: 0.38 ANT EL Offset: -0.34

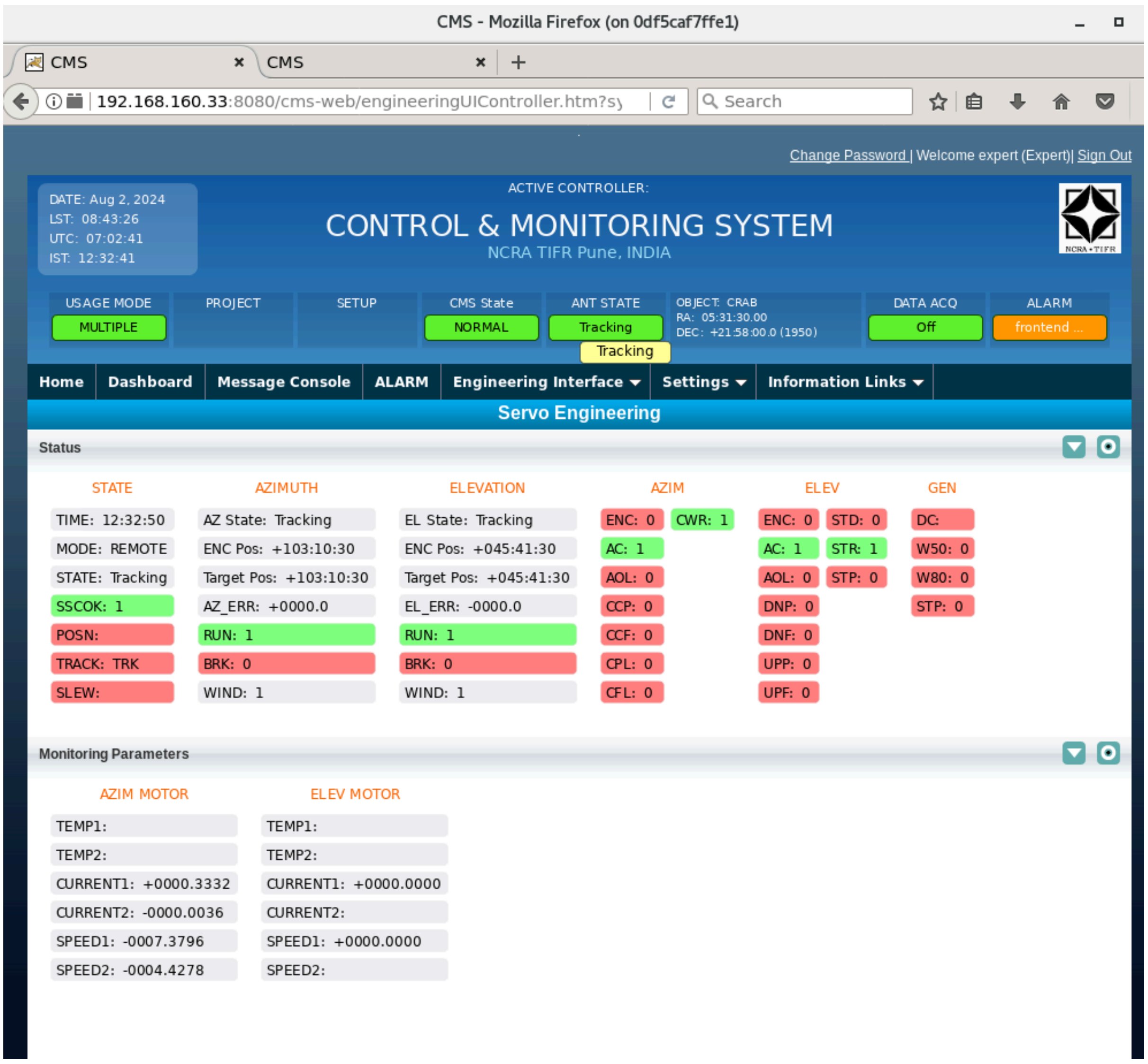
Antenna Location:
LAT: 18:33:35.99 Long: 73.82 Height: 562m
TimeZone: IST[GMT+5.30]

Plot Spectral Line Display

```
Plot Time : Fri Aug  2 14:59:01 IST 2024
```

Figure-13

Engineering Sub-system Parameters Monitoring Display :



CMS - Mozilla Firefox (on 0df5caf7ffe1)

192.168.160.33:8080/cms-web/engineeringUIController.htm?sy

Change Password | Welcome expert (Expert)| Sign Out

DATE: Aug 2, 2024
LST: 08:43:26
UTC: 07:02:41
IST: 12:32:41

ACTIVE CONTROLLER:

CONTROL & MONITORING SYSTEM

NCRA TIFR Pune, INDIA

USAGE MODE	PROJECT	SETUP	CMS State	ANT STATE	OBJECT: CRAB	DATA ACQ	ALARM
MULTIPLE			NORMAL	Tracking / Tracking	RA: 05:31:30.00 DEC: +21:58:00.0 (1950)	Off	frontend ...

Home | Dashboard | Message Console | ALARM | Engineering Interface | Settings | Information Links

Servo Engineering

Status

STATE	AZIMUTH	ELEVATION
TIME: 12:32:50	AZ State: Tracking	EL State: Tracking
MODE: REMOTE	ENC Pos: +103:10:30	ENC Pos: +045:41:30
STATE: Tracking	Target Pos: +103:10:30	Target Pos: +045:41:30
SSCOK: 1	AZ_ERR: +0000.0	EL_ERR: -0000.0
POSN:	RUN: 1	RUN: 1
TRACK: TRK	BRK: 0	BRK: 0
SLEW:	WIND: 1	WIND: 1

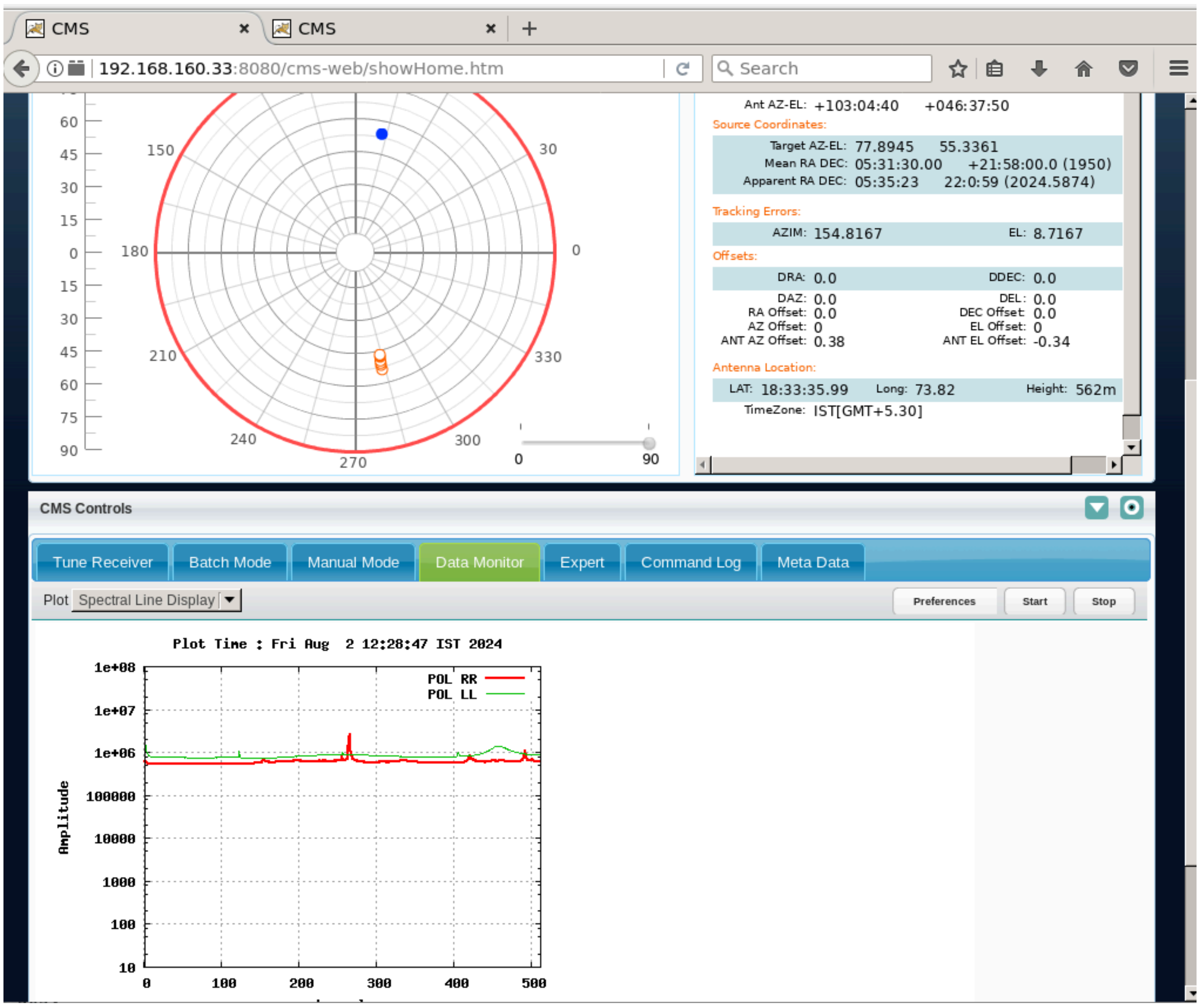
AZIM		ELEV		GEN
ENC: 0	CWR: 1	ENC: 0	STD: 0	DC:
AC: 1		AC: 1	STR: 1	W50: 0
AOL: 0		AOL: 0	STP: 0	W80: 0
CCP: 0		DNP: 0		STP: 0
CCF: 0		DNF: 0		
CPL: 0		UPP: 0		
CFL: 0		UPF: 0		

Monitoring Parameters

AZIM MOTOR	ELEV MOTOR
TEMP1:	TEMP1:
TEMP2:	TEMP2:
CURRENT1: +0000.3332	CURRENT1: +0000.0000
CURRENT2: -0000.0036	CURRENT2:
SPEED1: -0007.3796	SPEED1: +0000.0000
SPEED2: -0004.4278	SPEED2:

Figure-14

Receiver Settings and Band-plot :



Ant AZ-EL: +103:04:40 +046:37:50

Source Coordinates:

Target AZ-EL: 77.8945 55.3361
Mean RA DEC: 05:31:30.00 +21:58:00.0 (1950)
Apparent RA DEC: 05:35:23 22:0:59 (2024.5874)

Tracking Errors:

AZIM: 154.8167 EL: 8.7167

Offsets:

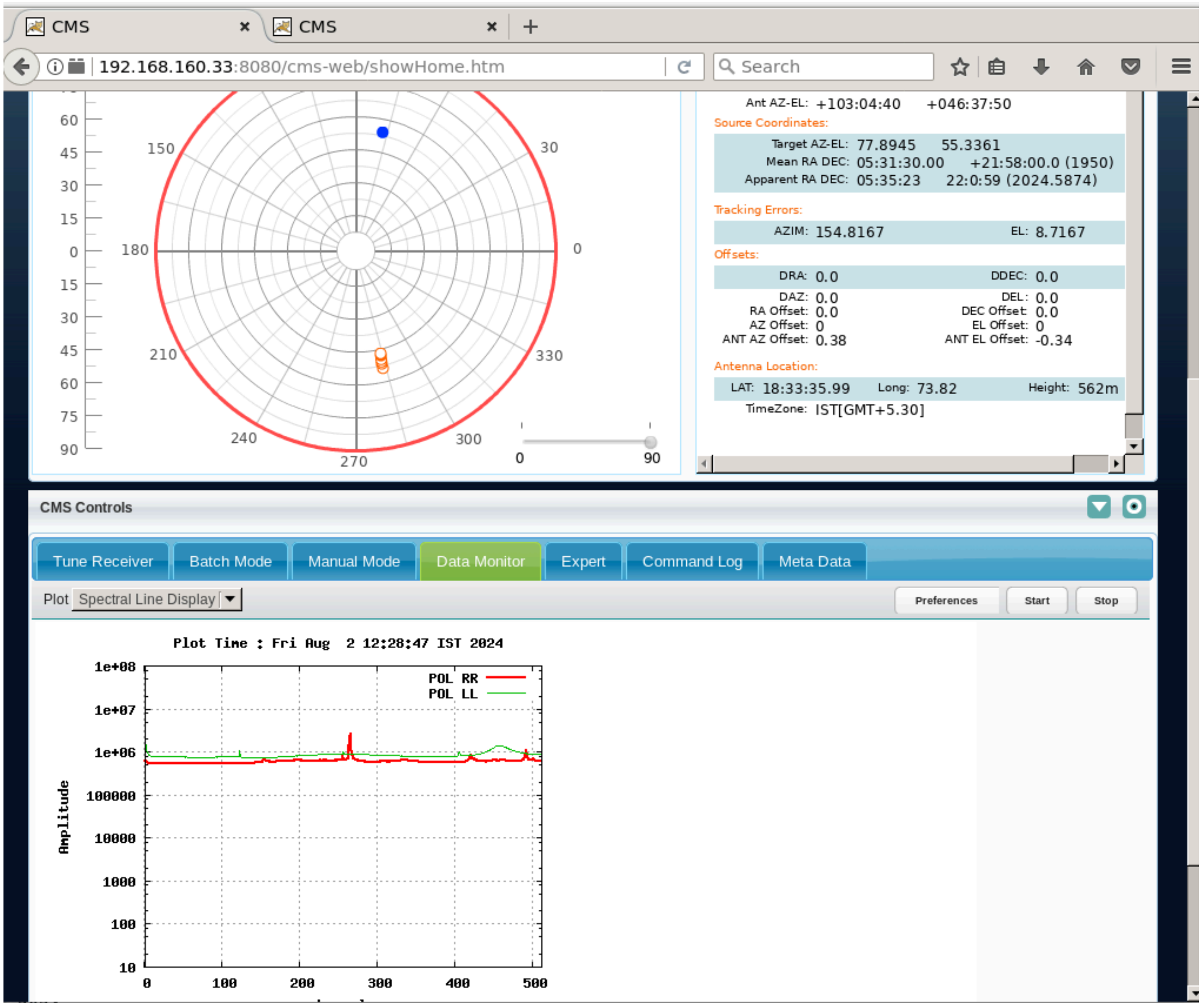
DRA: 0.0 DDEC: 0.0
DAZ: 0.0 DEL: 0.0
RA Offset: 0.0 DEC Offset: 0.0
AZ Offset: 0 EL Offset: 0
ANT AZ Offset: 0.38 ANT EL Offset: -0.34

Antenna Location:

LAT: 18:33:35.99 Long: 73.82 Height: 562m
TimeZone: IST[GMT+5.30]

Plot Time : Fri Aug 2 12:28:47 IST 2024

Figure-15:



Ant AZ-EL: +103:04:40 +046:37:50

Source Coordinates:

Target AZ-EL: 77.8945 55.3361
Mean RA DEC: 05:31:30.00 +21:58:00.0 (1950)
Apparent RA DEC: 05:35:23 22:0:59 (2024.5874)

Tracking Errors:

AZIM: 154.8167 EL: 8.7167

Offsets:

DRA: 0.0 DDEC: 0.0
DAZ: 0.0 DEL: 0.0
RA Offset: 0.0 DEC Offset: 0.0
AZ Offset: 0 EL Offset: 0
ANT AZ Offset: 0.38 ANT EL Offset: -0.34

Antenna Location:

LAT: 18:33:35.99 Long: 73.82 Height: 562m
TimeZone: IST[GMT+5.30]

```
Plot Time : Fri Aug  2 12:28:47 IST 2024
```

Figure-16:

4.3.3 Non-Functional Testing :

#	Type	Legacy System	New System running under Docker
1	Machine	PowerEdge R310 (™) Dell Rack-Server	HP Compaq 8200 (Mini Tower Computer)
2	CPU	Intel(R) Xeon(R) Quad-core CPU X3430 @ 2.40GHz RAM : 16 GB	Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz RAM : 16 GB
3	OS	RHEL 6	Docker Image - CentOS 7 Host OS - CentOS 7
4	Average CPU Consumption for 1 hour	6.9	12.48
5	Average	3.9	4.6

	Mem in % Consumption		

Table no -7

(1) CPU Consumption by the CMS Process over 1 hour :

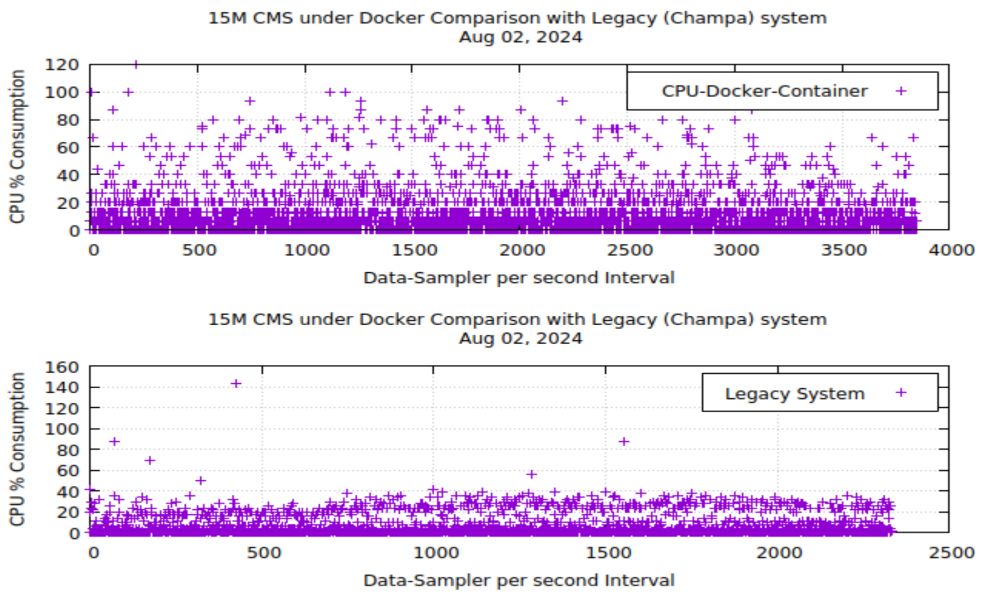


Figure-17:

(2) Memory Consumption by the CMS Process over 1 hour :

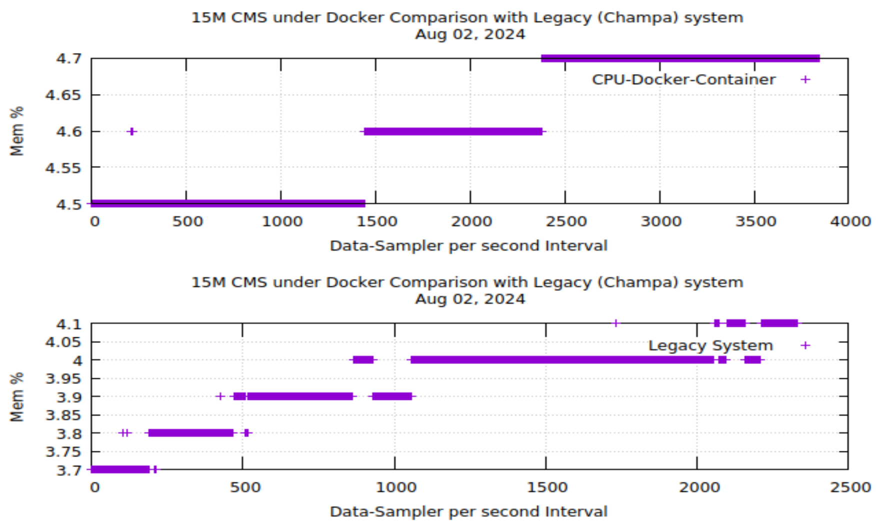


Figure-18:

Containers generally have access to the full CPU and RAM resources of the host system, with only minimal overhead. The CPU overhead for running a container is negligible, often considered close to zero. Similarly, RAM usage for containers is very efficient. While the Docker daemon itself does consume some CPU and RAM, these requirements are minimal compared to the resources needed to run applications in a virtual machine.