|  |
| --- |
| **“Expert Cloud Consulting” -**  **User, Process, Resource management in Linux**  07.October.2025    Contributed by M Bindu  Approved by Rushi (In Review)  Expert Cloud Consulting  Office #811, Gera Imperium Rise,  Hinjewadi Phase-II Rd, Pune, India – 411057 |

“Expert Cloud Consulting”

User, Process, Resource management in Linux

### 1.0 Contents

Contents

[1.0 Contents 1](#_Toc210746032)

[2.0 General Information: 2](#_Toc210746033)

[3.0 Document Overview: 3](#_Toc210746034)

[4.0 Steps / Procedure 4](#_Toc210746035)

****

### 

### 2.0 General Information:

#### 2.1 Document ( Linux system administration )

|  |  |
| --- | --- |
| **Ticket(s) Name** | **Url** |
| Setup the EC2 server | NA |

#### 2.2 Document Purpose

This document outlines the procedures and guidelines for setting up and managing Jenkins and Tomcat services on an Ubuntu server. It covers user creation, process prioritization, CPU usage limitation, automated maintenance scheduling through shell scripting, resource monitoring, and reverse proxy configuration using Apache or Nginx. The objective is to simulate real-world DevOps practices in process, user, and resource management

#### 2.3 Document Revisions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date** | **Version** | **Contributor(s)** | **Approver(s)** | **Section(s)** | **Change(s)** |
| 07/Oct/2025 | 1.0 | M Bindu | Rushi | All Sections | New Document Created |
|  |  |  |  |  |  |

#### 2.4 Document References

The following artifacts are referenced within this document. Please refer to the original documents for additional information.

|  |  |  |
| --- | --- | --- |
| **Date** | **Document** | **Filename / Url** |
| 2025 | Setup EC2 instance | <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-launch-tutorials.html> |
| 2025 | Scripting for maintenance window | NA |
| 2025 | Reverse proxy for Jenkins using nginx | <https://www.youtube.com/watch?v=B62QSbPhh1s> |

### 

### 3.0 Document Overview:

This document provides a comprehensive overview of the setup and configuration of Jenkins and Tomcat on an Ubuntu server. It explains how to create and manage dedicated system users for each service, assign process priorities, and restrict CPU utilization to ensure optimal performance. Additionally, it covers the implementation of automation scripts for Tomcat maintenance scheduling, real-time resource monitoring for CPU and RAM, and the configuration of a reverse proxy using Apache or Nginx. The goal of this document is to demonstrate practical DevOps concepts related to user, process, and resource management in a real-world server environment.

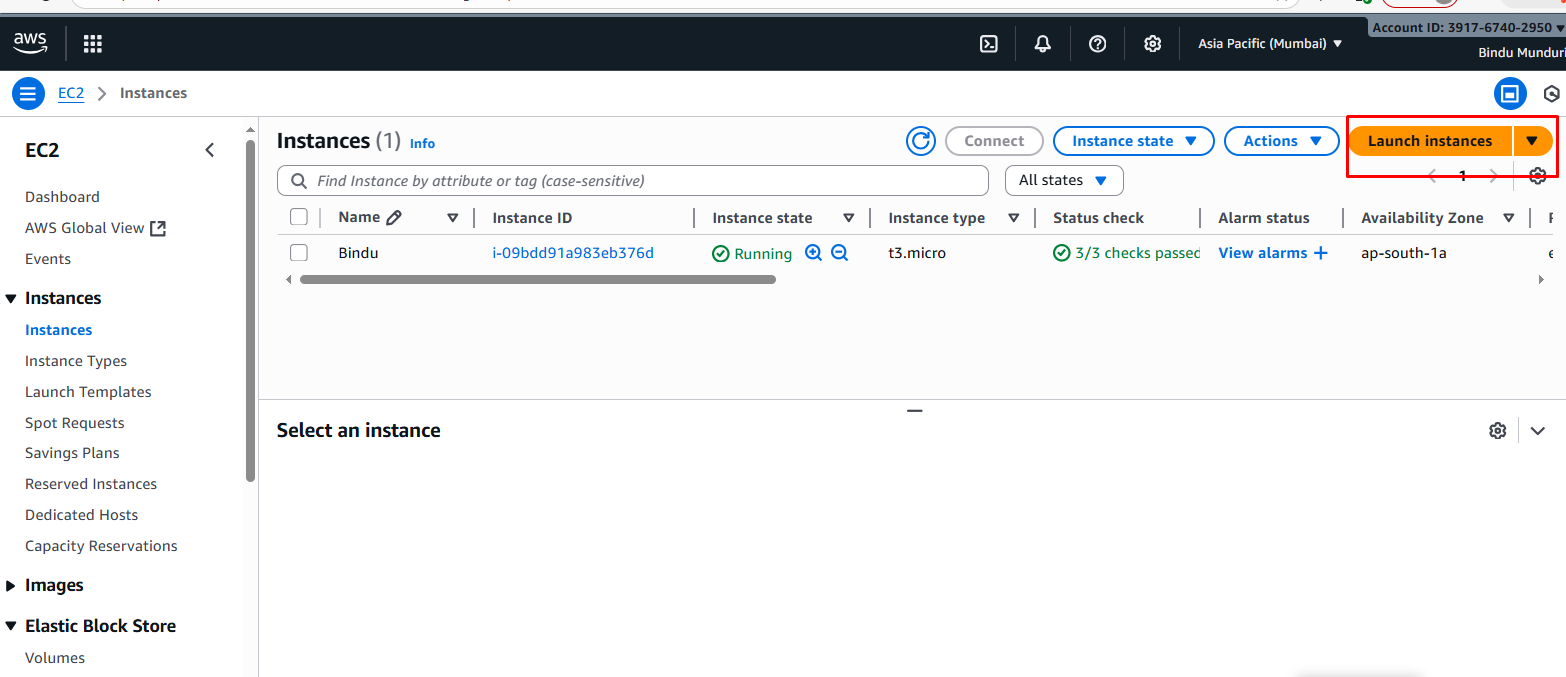
### 4.0 Steps / Procedure

#### 4.1 : Setup the ubuntu server

The following procedure has been done during the installation of ubuntu server:

Log in to the AWS Management Console and navigate to the EC2 dashboard.

Click on the "Launch Instance" button to start the process of launching a new EC2 instance.



#### 4.2: Choose an Amazon Machine Image (AMI)

Select an instance type, configure your instance details (such as the number of instances, and storage)

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

#### 4.3: Key-Pair Configuration

Select an instance type and create a new key-pair as name is sandbox-jenkins-keypair.

A screenshot of a computer

AI-generated content may be incorrect.

#### 4.4: Network settings

##### 4.4.1: VPC Configuration:

Select the vpc and subnet for the ec2 instance.

A screenshot of a computer

AI-generated content may be incorrect.

##### 4.4.2: Security Group Configuration

create a new security group with required security rules

A screenshot of a computer

AI-generated content may be incorrect.

Specified Security group rule for this ec2 instances

#### 4.5: Launch Instance

click on Launch instance

A screenshot of a computer

AI-generated content may be incorrect.

Instances are created and they are ready to use.

A screenshot of a computer

AI-generated content may be incorrect.

#### 4.5: SSH Configuration

Log in ec2 instance using SSH client.

A screenshot of a computer

AI-generated content may be incorrect.

Successfully able to connect the ec2-Instance by using ssh client.

A screenshot of a computer screen

AI-generated content may be incorrect.

#### 4.6: Install Jenkins & Tomcat on Ubuntu Server

Create Jenkins User:

# Create jenkins user with home directory and bash shell

sudo useradd -m -s /bin/bash Jenkins

# Set password for jenkins user

sudo passwd jenkins # Enter password: Demo

Create Tomcat User:

# Create tomcat user with home directory and bash shell

sudo useradd -m -s /bin/bash tomcat

Set password for tomcat user

sudo passwd tomcat # Enter password: Demo

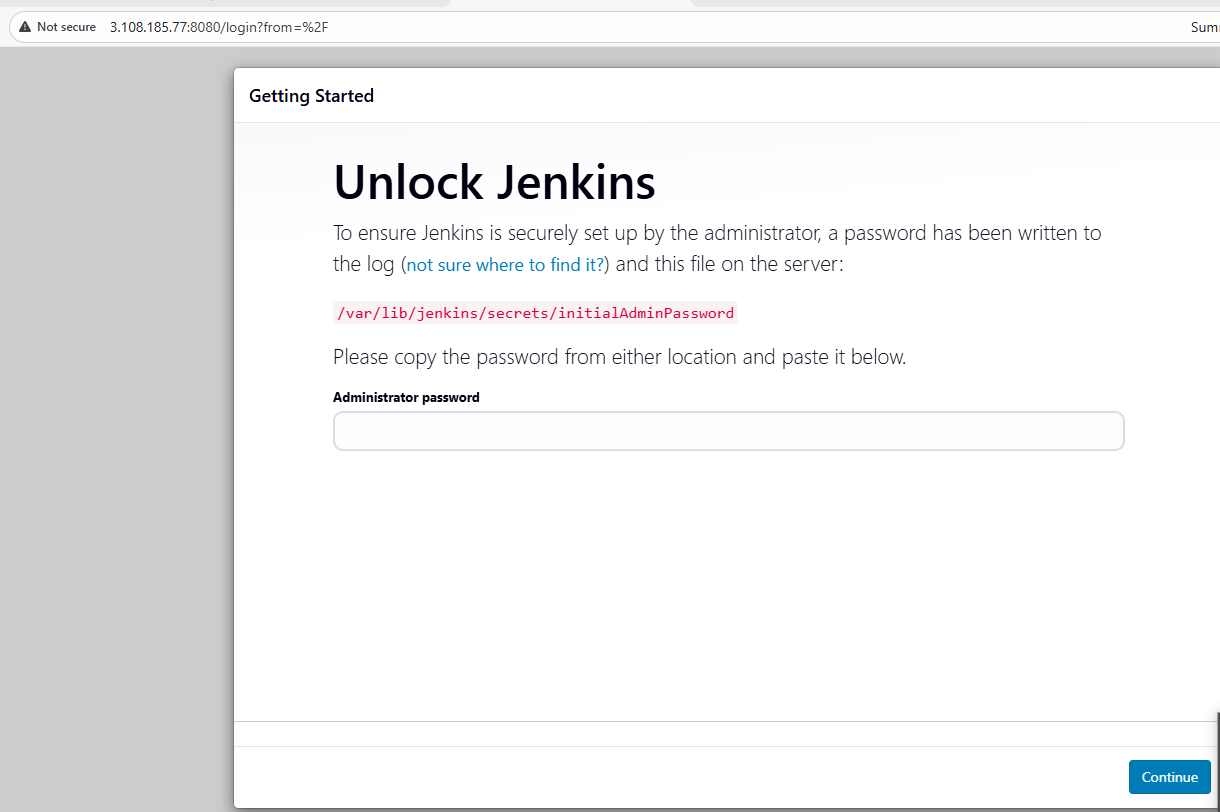
Why are we creating two separate users (jenkins and tomcat)?

This is a fundamental security practice known as principle of least privilege and service isolation security If one service (e.g., Tomcat) is compromised, the attacker will have the permissions of the tomcat user only, and will not be able to affect the jenkins user's files or process Isolation: It prevents conflicts. Each service has its own dedicated user, group, home directory, and file permissions, ensuring they do not interfere with each other accountability It makes logging and auditing clearer, as all actions are tied to a specific service user

To install Jenkins and Tomcat on Ubuntu Server. we need to follow below commands:

|  |
| --- |
| //To install Jenkins on Ubuntu Server  first, we have to install java  sudo apt update  sudo apt install openjdk-17-jdk -y then installation of Jenkins  sudo apt install Jenkins  //installation of Tomcat  Downloading Tomcat  wget -c <https://downloads.apache.org/tomcat/tomcat-9/v9.0.34/bin/apache-tomcat-9.0.34.tar.gz>  install Tomcat  sudo tar xf apache-tomcat-9.0.34.tar.gz -C /opt/tomcat  start Tomcat  sudo systemctl start Tomcat   //Check the status of the Jenkins & Tomcat  sudo snap services amazon-ssm-agent |

Jenkins is working properly on 8080:



By default , Jenkins runs on port 8080

Tomcat configuration :

Why did we change the default port of Tomcat from 8080 to 8081?

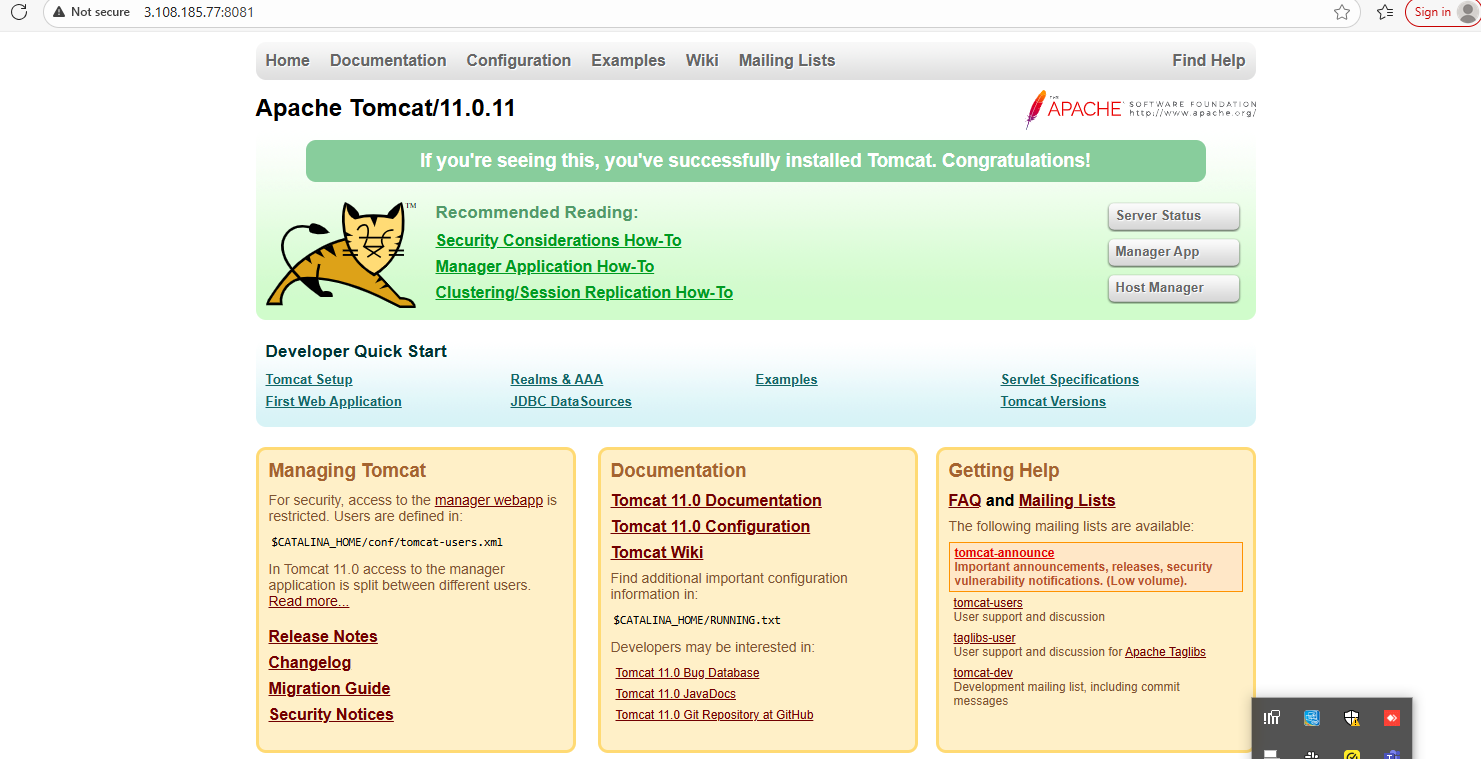
Because only one process can listen on a specific port at a time. Since Jenkins was configured to use its default port (8080), Tomcat would fail to start if it also tried to use 8080. Changing Tomcat to 8081 avoids this port conflict.

A screenshot of a computer

AI-generated content may be incorrect.

Tomcat is running on port 8081 because Jenkins is already using port 8080, and each service needs a unique port to avoid conflicts.

Tomcat is running on port 8081:



4.7 Processes allocated to Jenkins should have more priority than the Tomcat process:

“Processes allocated to Jenkins are given higher priority than Tomcat to ensure that Jenkins jobs, which are critical for continuous integration and deployment, get more CPU resources when the server is under load. This prevents Jenkins from being delayed or slowed down by Tomcat processes running simultaneously. By managing process priorities, we can maintain optimal performance and ensure important automation tasks are completed reliably. It also helps in balancing server resources effectively between different services running on the same machine.”

How I apply Nice values:

“I assigned a **nice value of -5 to Jenkins** and **10 to Tomcat**. A lower nice value (-5) gives Jenkins **higher CPU scheduling priority**, ensuring its critical CI/CD tasks run faster. A higher nice value (10) for Tomcat lowers its priority, so it only uses leftover CPU resources, preventing it from slowing down Jenkins.”

Commands to apply :

sudo renice -n -5 $(pgrep -u jenkins java)

sudo renice -n 10 $(pgrep -u tomcat java)

A computer screen with white text

AI-generated content may be incorrect.

Jenkins is assigned higher process priority than Tomcat to ensure it gets more CPU time, so critical CI/CD tasks run smoothly without being slowed down by Tomcat processes.

4.8 Restict jenkins to use 20% of CPU and Tomcat to use 30% of CPU:

“Jenkins is restricted to 20% of CPU and Tomcat to 30% to ensure that no single service consumes excessive system resources. This helps prevent server overload and ensures that both services can run simultaneously without affecting each other’s performance. By setting CPU limits, we maintain a balanced and stable environment where critical tasks like Jenkins builds and Tomcat applications can operate reliably. It also allows other system processes to have sufficient CPU availability, improving overall server efficiency.”

Commands to apply for CPUquota:

sudo cpulimit -e java -l 20 -b -u jenkins

sudo cpulimit -e java -l 30 -b -u tomcat

A screenshot of a computer

AI-generated content may be incorrect.

Jenkins is limited to 20% CPU and Tomcat to 30% to prevent any one service from overusing resources, ensuring both run smoothly and the server stays stable

CPU Limit Behavior:

When we increase the CPU limit (like from 15% to 20%), the service restarts and gets a new process ID. There might be a temporary CPU spike during startup, but then it settles down and stays under the 20% and 30% limit

5.0 Script for Tomcat maintenance window and Resource monitoring:

Scripts are used to automate repetitive tasks, such as:

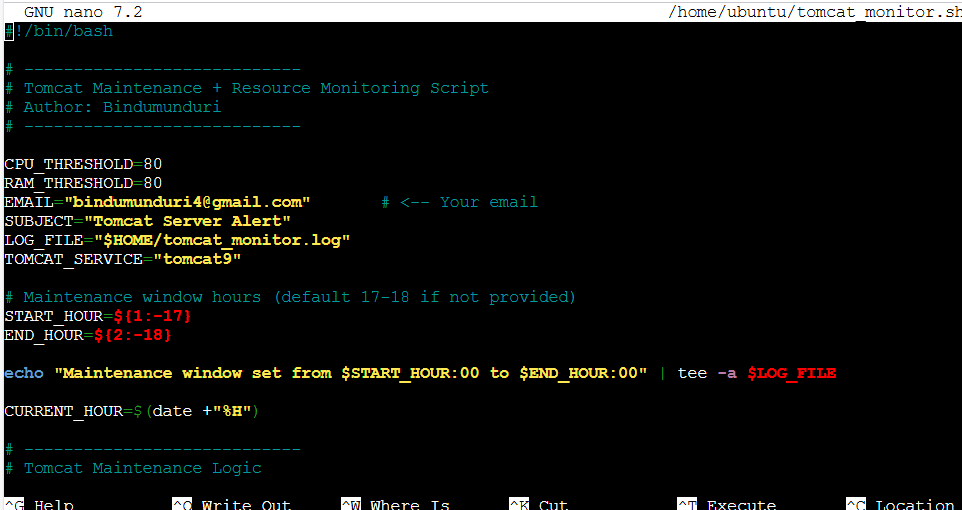
* Stopping Tomcat during maintenance hours,
* Restarting it after updates,
* Monitoring CPU and RAM in real-time,

This reduces manual work and human error, and ensures consistent reliability

Why we wrote script :

We wrote the script to automate repetitive and critical tasks on the server. It ensures that Tomcat is stopped and started automatically during the maintenance window, and it monitors CPU and RAM usage in real time. This reduces manual effort, prevents human errors, and provides logs and alerts to maintain server stability and performance

The script part is



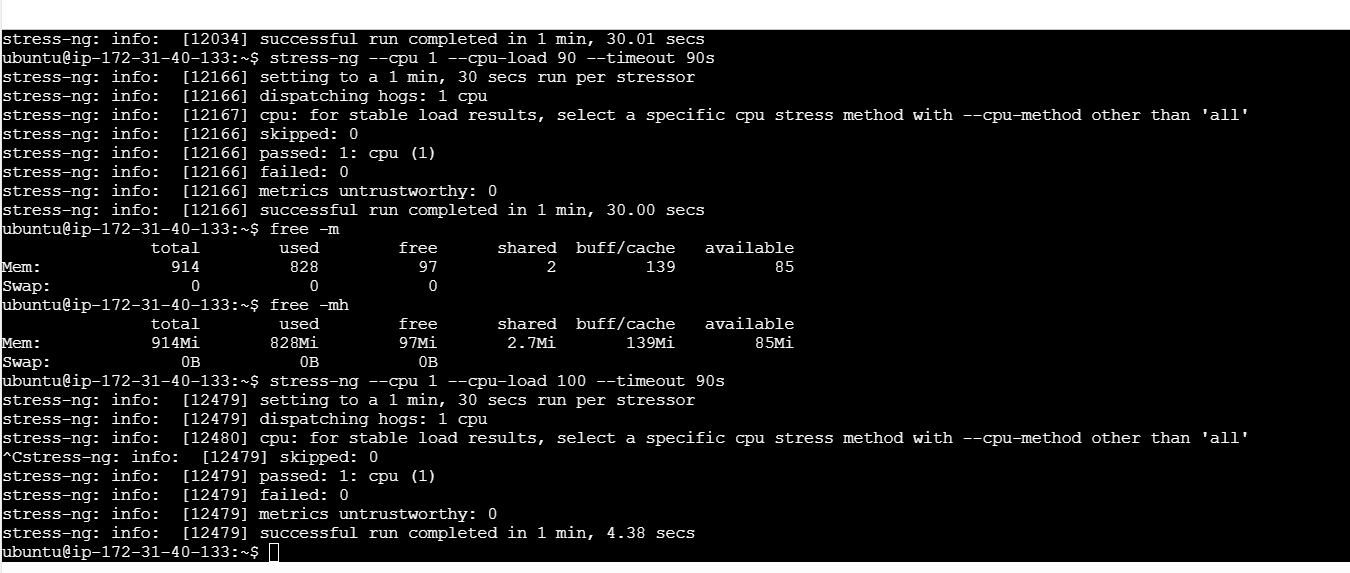
A screenshot of a computer program

AI-generated content may be incorrect.

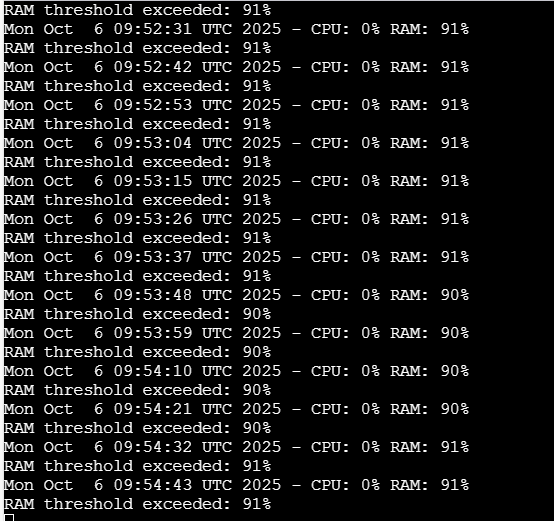
A computer screen shot of text

AI-generated content may be incorrect.

Scripting part is done. Now, we will check in background :



Now, The CPU shows 0% because the system is currently idle or the monitored process isn’t using CPU at that moment. RAM is at 90%, which exceeds the defined threshold of 80%, so the script correctly triggered a RAM alert. This demonstrates that the resource monitoring script is working as intended — it can detect high memory usage even when CPU usage is low



CPU is 0% because the system is idle, but RAM is 90%, exceeding the threshold. The script correctly triggered a RAM alert, showing the monitoring works as intended

6.0 Reverse proxy for Jenkins using Nginx:

What is Reverse proxy ?

* A reverseproxy is a server that sits between clients and backend servers.
* It receives client requests and forwards them to the appropriate backend service (like Jenkins or Tomcat).
* After processing, it sends the response back to the client.

Why we use a Reverse Proxy in this task

* **Multiple applications on the same server** (Jenkins on 8080, Tomcat on 8081) can be accessed easily using a single domain.
* **Improves security**: hides the internal server details from clients.
* **Load balancing**: can distribute requests between servers (useful if you scale).
* **SSL termination**: can handle HTTPS connections at the proxy level.

We are doing Reverse proxy using Nginx :

1. Install Nginx:

sudo apt install nginx

Nginx is running on port 80 :

A screenshot of a computer

AI-generated content may be incorrect.

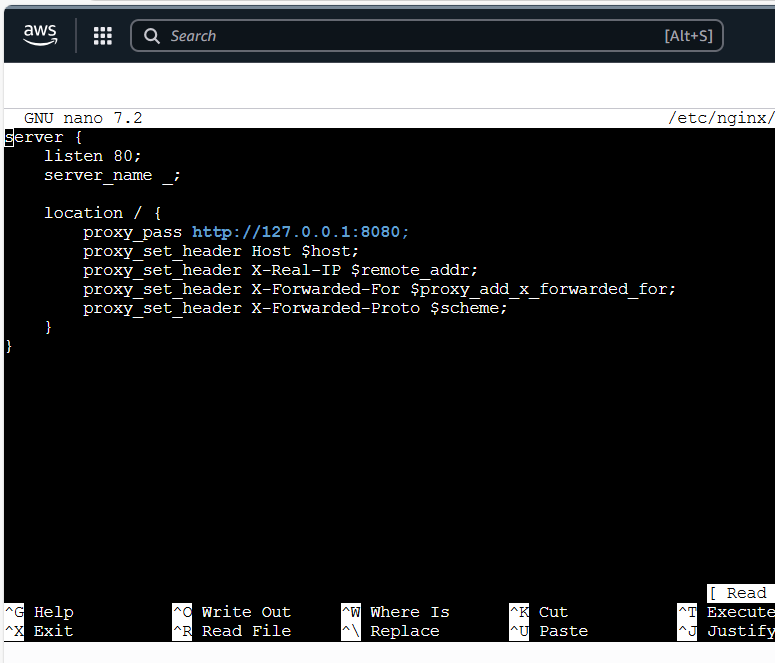
Now, Jenkins is running on port 8080

A screenshot of a computer

AI-generated content may be incorrect.

We saved reverse proxy configuration file here

sudo nano /etc/nginx/conf.d/jenkins.conf



We changed Jenkins to port 80 because Nginx listens on port 80 as a reverse proxy and forwards traffic to backend services like Jenkins

Now, the configuration file syntax is ok and successful:

A screenshot of a computer screen

AI-generated content may be incorrect.

Now, Jenkins is running on port 80, Reverse proxy worked here

A screenshot of a computer

AI-generated content may be incorrect.

Reverse proxy ensures smooth access to multiple services, improves security, and allows centralized management of server traffic — exactly how production environments are configured in real life.

### 