User is working on a mini project related to automated memory monitoring and ticket creation, and plans to push it on GitHub.Here’s a refined version of the project, combining memory monitoring, JIRA ticket creation, and email notifications. I've also added real-life issue case details in the JIRA ticket creation section and integrated the email sending functionality into the process. This should be easy to follow for anyone looking to implement a similar solution. Here's the code:

**Automated Memory Monitoring and Ticket Creation with JIRA and Email Notification**

**Objective:**  
This project automates the monitoring of server memory on a Linux machine and creates JIRA tickets when memory is low. It also sends an email to the reporter of the ticket, asking for manager approval before proceeding further.

**Project Structure:**

1. **Prerequisites:**
   * Python 3.x
   * Libraries: **paramiko**, **jira**, **smtplib**, **email.mime.multipart**
   * JIRA API access: You need a JIRA API key and authentication details.
   * SSH access to the Linux server.
2. **Dependencies Installation:** Run the following command to install required Python libraries:

bashCopy code

pip install paramiko jira

1. **GitHub Repository Setup:** Create a new repository on GitHub and push the code there for easy access and version control.

**Code Implementation:**

pythonCopy code

import time

import paramiko

from jira import JIRA

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

# Server and SSH credentials

hostname = "192.168.146.128"

username = "abdullah"

password = "123@Abd..."

# JIRA User credentials and API Key

user = "abdullahbinaminmeo@gmail.com" # Your JIRA username

api\_key = 'WRITE\_YOUR\_JIRA\_API\_KEY' # Your JIRA API key

server = "https://al-nafi.atlassian.net" # JIRA server URL

# Connect to JIRA

jira = JIRA(server=server, basic\_auth=(user, api\_key))

# SSH Client Setup

client = paramiko.SSHClient()

client.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())

client.connect(hostname=hostname, username=username, password=password)

# Command to check available memory

cmd = "free -g"

std\_in, std\_out, std\_error = client.exec\_command(cmd)

# Output of the command

cmdout = std\_out.readlines()

for line in cmdout:

print(line)

# Extract available memory from the command output

available\_memory = cmdout[1].split()[6]

# Check if the available memory is less than or equal to 1GB

if int(available\_memory) <= 1:

print(f"Alert: Available memory on server {hostname} is critically low ({available\_memory}GB).")

# Prepare the issue details

Summary = f"Memory Available: {available\_memory}GB"

Description = f"""Dear Team,

A memory shortage has been detected on the server {hostname}.

Available memory: {available\_memory}GB.

This is a critical issue that requires your immediate attention.

Please review and take appropriate action to resolve the memory shortage.

Thank you,

Al-Nafi Support Team

"""

# Issue details for creating the JIRA ticket

issue\_dict = {

'project': {'key': 'IIP'},

'issuetype': {'name': 'Request'},

'description': Description,

'summary': Summary,

'priority': {'name': 'Highest'},

'timetracking': {'originalEstimate': '1h'},

'assignee': {"accountId": "712020:8eced984-01f8-4569-b7e6-dead12e79cc7"} # Specify stakeholder account ID

}

# Create the JIRA issue

new\_issue = jira.create\_issue(fields=issue\_dict)

print(f'Ticket created and assigned to the stakeholder. Issue ID: {new\_issue.key}')

# Send Email to Reporter for Manager Approval

email\_subject = f"Approval Request for Ticket {new\_issue.key}: {Summary}"

email\_body = f"""

<p>Dear Reporter,</p>

<p>Thank you for writing to us.</p>

<p>We have received your JIRA ticket <strong>{new\_issue.key}: {Summary}</strong> for a server access request.</p>

<p>Please provide the approval from your reporting head for proceeding further. Our team is looking into the issue and we will get back to you as quickly as possible.</p>

<p>In case of immediate assistance, please dial the Al-Nafi Direct lines for a quick response.</p>

<p>We look forward to interacting soon!</p>

<p>Best regards,<br>Al-Nafi Support Team</p>

"""

# Fetch the reporter's email

reporter\_email = new\_issue.fields.reporter.emailAddress

# Send the approval request email

my\_mail = "your\_email@gmail.com"

password = "your\_app\_password"

msg = MIMEMultipart()

msg['Subject'] = email\_subject

msg['From'] = my\_mail

msg['To'] = reporter\_email

msg.attach(MIMEText(email\_body, 'html'))

# SMTP server setup

with smtplib.SMTP('smtp.gmail.com') as connection:

connection.starttls() # Secure connection

connection.login(user=my\_mail, password=password)

connection.send\_message(msg)

print(f"Approval email sent to {reporter\_email}")

else:

print(f"Memory status on {hostname} is optimal. Available memory: {available\_memory}GB.")

**Explanation of Code:**

1. **SSH Connection to the Server:**
   * The **paramiko** library is used to connect to a remote Linux server and execute the **free -g** command to check available memory.
2. **JIRA Integration:**
   * The script uses the **jira** library to create a JIRA ticket when the available memory is below 1GB. The ticket includes a description of the issue and is assigned to the relevant stakeholder.
3. **Email Notification:**
   * When a ticket is created, an email is sent to the reporter requesting approval from their manager to proceed further with the ticket. The email includes the ticket details and instructions for the approval process.
4. **Real-Life Scenario for the Issue:**
   * In the real-world scenario, when the server's memory falls below a critical threshold, it may affect the performance and stability of applications hosted on the server. A quick resolution is necessary to avoid downtime. The ticket created highlights this issue and ensures it is handled promptly.

**How to Run the Project:**

1. **Install Dependencies:** Make sure to install the required libraries (**paramiko**, **jira**, **smtplib**).
2. **JIRA Setup:**
   * Create a JIRA project if you don't already have one (in this case, **IIP**).
   * Generate an API key for authentication.
3. **Email Setup:**
   * Configure the email address and app-specific password for the email account used to send notifications.
4. **Run the Script:**
   * Once everything is set up, you can run the Python script to monitor the server's memory. The script will check the available memory and create a JIRA ticket if necessary.
5. **GitHub:**
   * Push the entire project to GitHub for version control and sharing.

**Final Thoughts:**

This project demonstrates the power of automation in IT support. By integrating server monitoring, issue tracking (via JIRA), and email notifications, the script provides a streamlined process for addressing critical memory issues and ensures that the required stakeholders are notified promptly. It is highly customizable for different IT environments and can be adapted to handle other types of system monitoring.