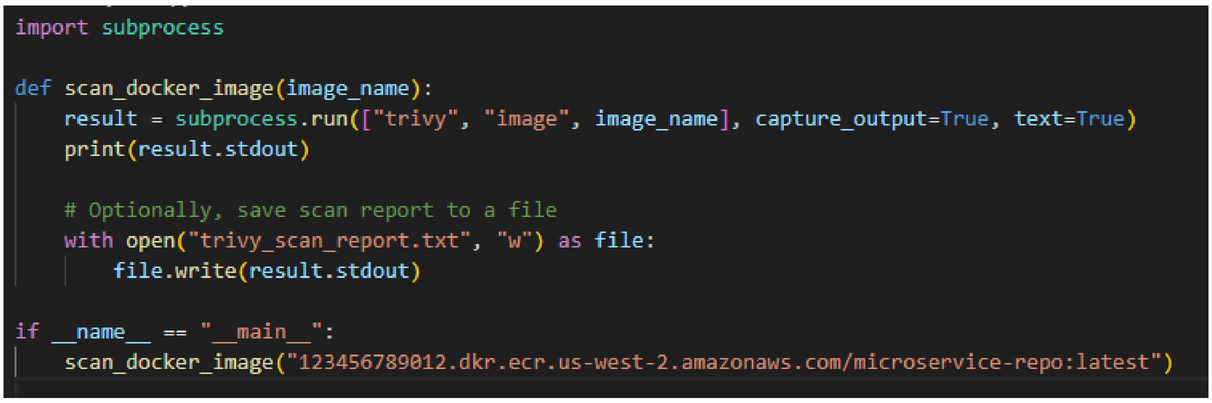
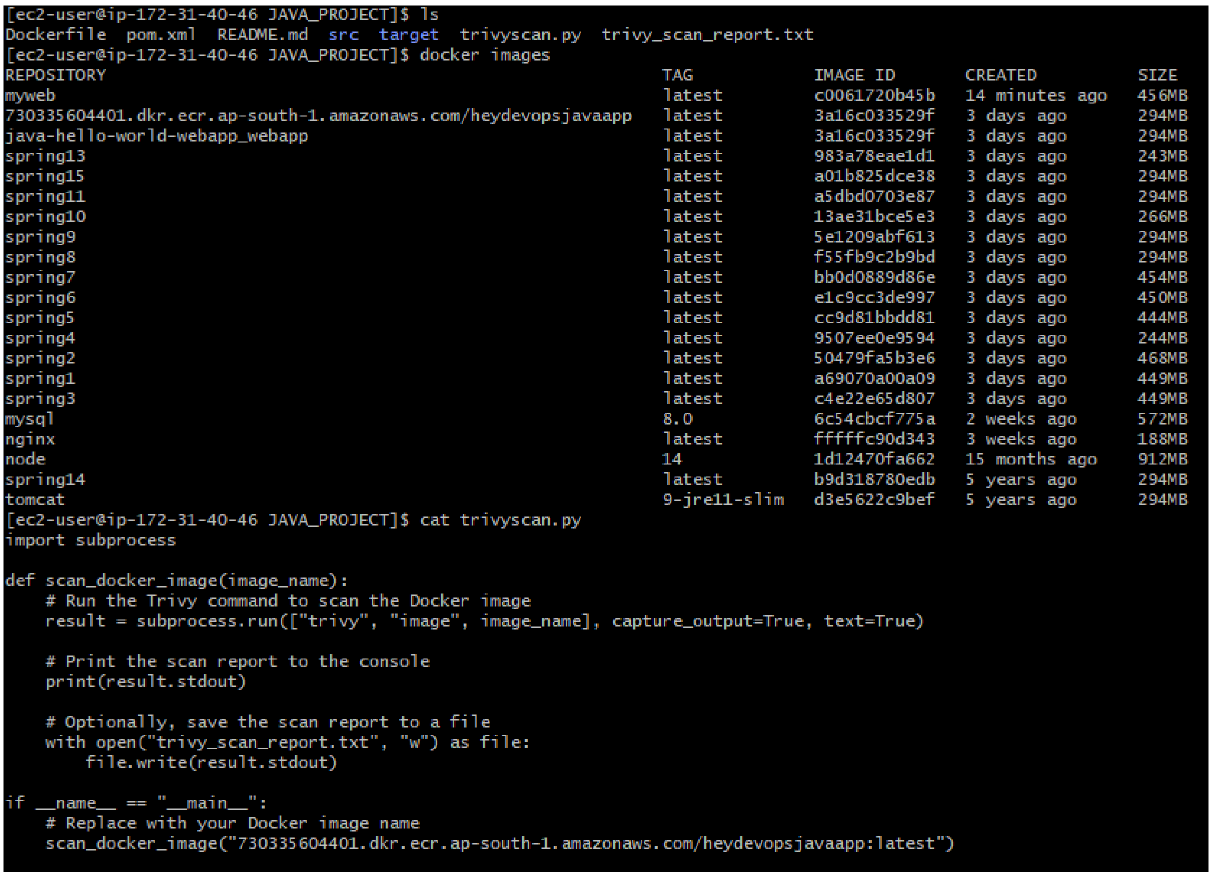
Scenario Based Python Automation Ideas

Scenario 1: Automated Trivy Scan

- **Scenario**: Scan Docker images for vulnerabilities using Trivy.



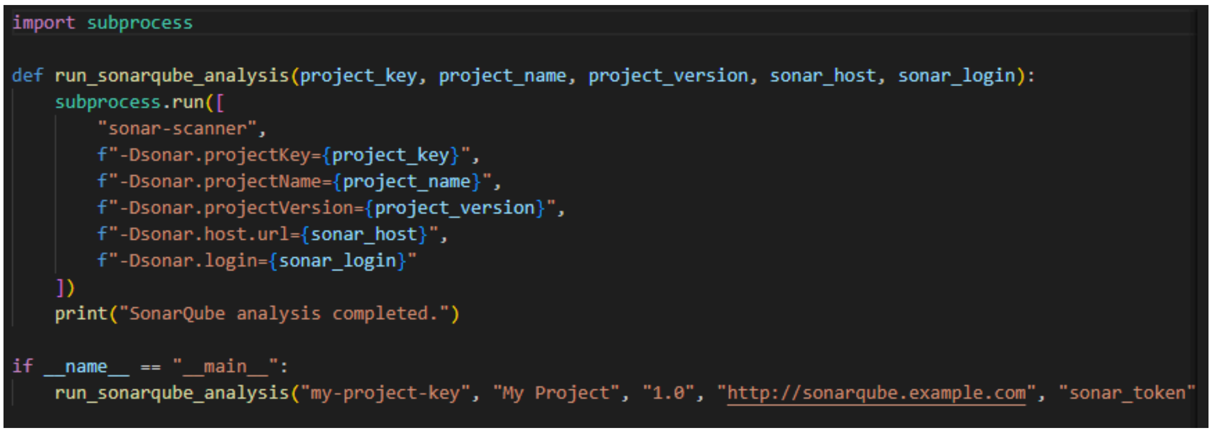
```python
import subprocess

def scan_docker_image(image_name):
    result = subprocess.run(["trivy", "image", image_name], capture_output=True, text=True)
    print(result.stdout)

    # Optionally, save scan report to a file
    with open("trivy_scan_report.txt", "w") as file:
        file.write(result.stdout)

if __name__ == "__main__":
    scan_docker_image("123456789012.dkr.ecr.us-west-2.amazonaws.com/microservice-repo:latest")
```



```
[ec2-user@ip-172-31-40-46 JAVA_PROJECT]$ ls
Dockerfile  pom.xml  README.md  src  target  trivyscan.py  trivy_scan_report.txt
[ec2-user@ip-172-31-40-46 JAVA_PROJECT]$ docker images
REPOSITORY                                                 TAG            IMAGE ID       CREATED          SIZE
myweb                                                      latest         c0061720b45b   14 minutes ago   456MB
730335604401.dkr.ecr.ap-south-1.amazonaws.com/heydevopsjavaapp   latest   3a16c033529f   3 days ago       294MB
java-hello-world-webapp_webapp                             latest         3a16c033529f   3 days ago       294MB
spring13                                                   latest         983a78eae1d1   3 days ago       243MB
spring15                                                   latest         a01b825dce38   3 days ago       294MB
spring11                                                   latest         a5dbd0703e87   3 days ago       294MB
spring10                                                   latest         13ae31bce5e3   3 days ago       266MB
spring9                                                    latest         5e1209abf613   3 days ago       294MB
spring8                                                    latest         f55fb9c2b9bd   3 days ago       294MB
spring7                                                    latest         bb0d0889d86e   3 days ago       454MB
spring6                                                    latest         e1c9cc3de997   3 days ago       450MB
spring5                                                    latest         cc9d81bbdd81   3 days ago       444MB
spring4                                                    latest         9507ee0e9594   3 days ago       244MB
spring2                                                    latest         50479fa5b3e6   3 days ago       468MB
spring1                                                    latest         a69070a00a09   3 days ago       449MB
spring3                                                    latest         c4e22e65d807   3 days ago       449MB
mysql                                                      8.0            6c54cbcf775a   2 weeks ago      572MB
nginx                                                      latest         fffffc90d343   3 weeks ago      188MB
node                                                       14             1d12470fa662   15 months ago    912MB
spring14                                                   latest         b9d318780edb   5 years ago      294MB
tomcat                                                     9-jre11-slim   d3e5622c9bef   5 years ago      294MB
[ec2-user@ip-172-31-40-46 JAVA_PROJECT]$ cat trivyscan.py
import subprocess

def scan_docker_image(image_name):
    # Run the Trivy command to scan the Docker image
    result = subprocess.run(["trivy", "image", image_name], capture_output=True, text=True)

    # Print the scan report to the console
    print(result.stdout)

    # Optionally, save the scan report to a file
    with open("trivy_scan_report.txt", "w") as file:
        file.write(result.stdout)

if __name__ == "__main__":
    # Replace with your Docker image name
    scan_docker_image("730335604401.dkr.ecr.ap-south-1.amazonaws.com/heydevopsjavaapp:latest")
```

Scenario 2: Automated SonarQube Analysis

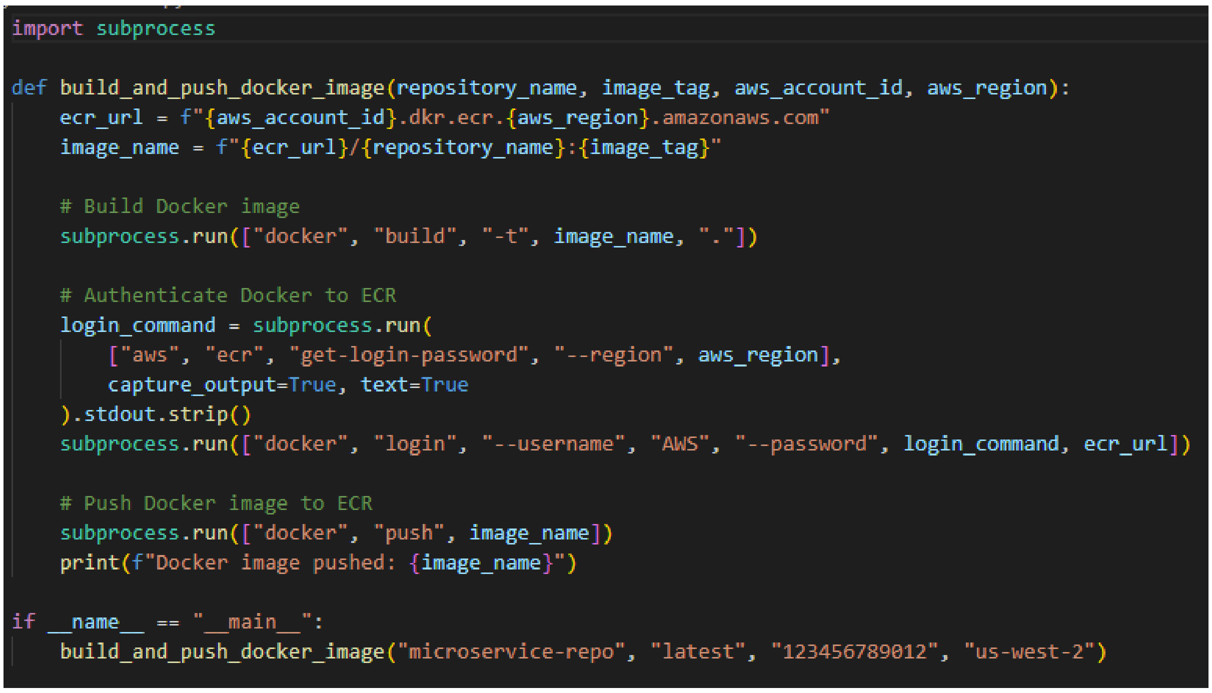
- **Scenario**: Perform code quality analysis using SonarQube.



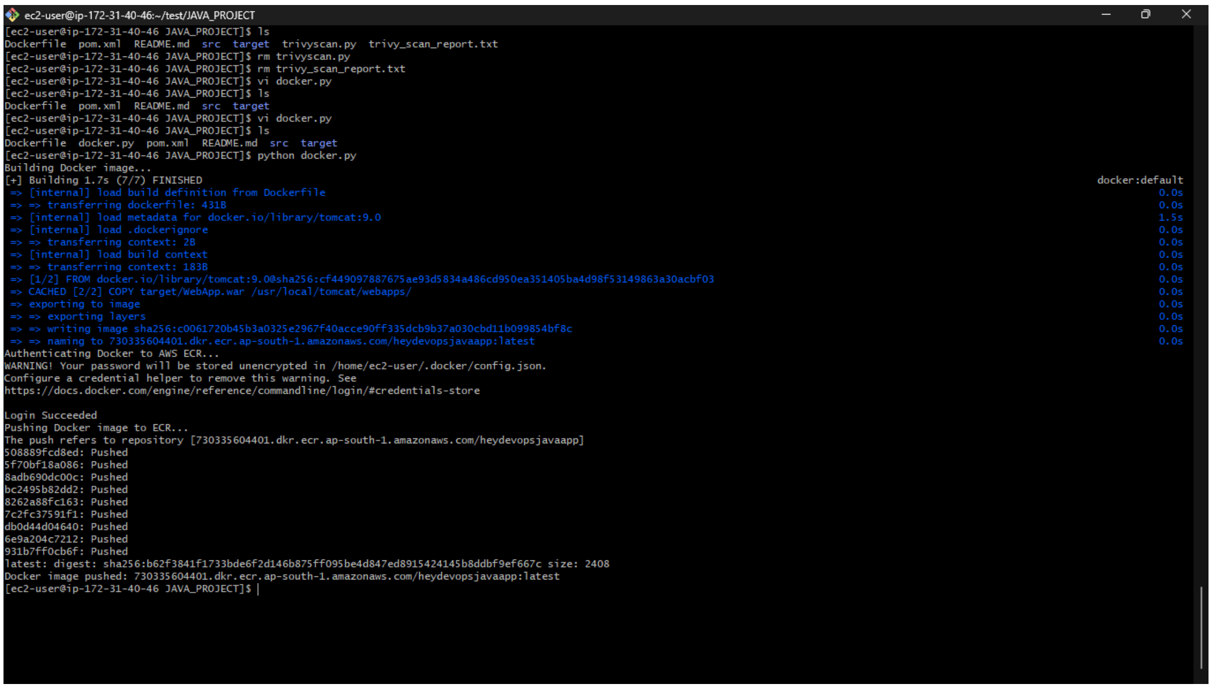
```python
import subprocess

def run_sonarqube_analysis(project_key, project_name, project_version, sonar_host, sonar_login):
    subprocess.run([
        "sonar-scanner",
        f"-Dsonar.projectKey={project_key}",
        f"-Dsonar.projectName={project_name}",
        f"-Dsonar.projectVersion={project_version}",
        f"-Dsonar.host.url={sonar_host}",
        f"-Dsonar.login={sonar_login}"
    ])
    print("SonarQube analysis completed.")

if __name__ == "__main__":
    run_sonarqube_analysis("my-project-key", "My Project", "1.0", "http://sonarqube.example.com", "sonar_token"
```

Scenario 3: Automated Docker Image Build and Push

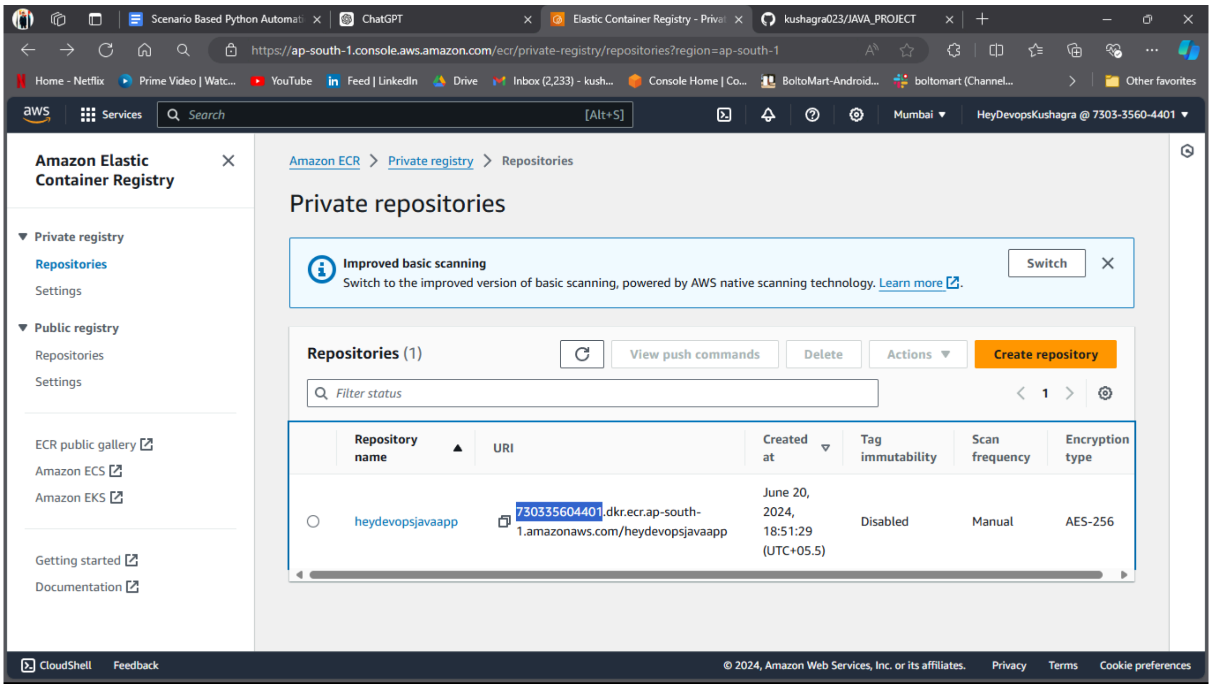
- **Scenario**: Automatically build and push Docker images to ECR when new artifacts are published.



```python
import subprocess

def build_and_push_docker_image(repository_name, image_tag, aws_account_id, aws_region):
    ecr_url = f"{aws_account_id}.dkr.ecr.{aws_region}.amazonaws.com"
    image_name = f"{ecr_url}/{repository_name}:{image_tag}"

    # Build Docker image
    subprocess.run(["docker", "build", "-t", image_name, "."])

    # Authenticate Docker to ECR
    login_command = subprocess.run(
        ["aws", "ecr", "get-login-password", "--region", aws_region],
        capture_output=True, text=True
    ).stdout.strip()
    subprocess.run(["docker", "login", "--username", "AWS", "--password", login_command, ecr_url])

    # Push Docker image to ECR
    subprocess.run(["docker", "push", image_name])
    print(f"Docker image pushed: {image_name}")

if __name__ == "__main__":
    build_and_push_docker_image("microservice-repo", "latest", "123456789012", "us-west-2")
```



```
ec2-user@ip-172-31-40-46:~/test/JAVA_PROJECT
[ec2-user@ip-172-31-40-46 JAVA_PROJECT]$ ls
Dockerfile  pom.xml  README.md  src  target  trivyscan.py  trivy_scan_report.txt
[ec2-user@ip-172-31-40-46 JAVA_PROJECT]$ rm trivyscan.py
[ec2-user@ip-172-31-40-46 JAVA_PROJECT]$ rm trivy_scan_report.txt
[ec2-user@ip-172-31-40-46 JAVA_PROJECT]$ vi docker.py
[ec2-user@ip-172-31-40-46 JAVA_PROJECT]$ ls
Dockerfile  pom.xml  README.md  src  target
[ec2-user@ip-172-31-40-46 JAVA_PROJECT]$ vi docker.py
[ec2-user@ip-172-31-40-46 JAVA_PROJECT]$ ls
Dockerfile  docker.py  pom.xml  README.md  src  target
[ec2-user@ip-172-31-40-46 JAVA_PROJECT]$ python docker.py
Building Docker image...
[+] Building 1.7s (7/7) FINISHED                                                                        docker:default
 => [internal] load build definition from Dockerfile                                                              0.0s
 => => transferring dockerfile: 431B                                                                              0.0s
 => [internal] load metadata for docker.io/library/tomcat:9.0                                                     1.5s
 => [internal] load .dockerignore                                                                                 0.0s
 => => transferring context: 2B                                                                                   0.0s
 => [internal] load build context                                                                                 0.0s
 => => transferring context: 183B                                                                                 0.0s
 => [1/2] FROM docker.io/library/tomcat:9.0@sha256:cf449097887675ae93d5834a486cd950ea351405ba4d98f53149863a30acbf03  0.0s
 => CACHED [2/2] COPY target/WebApp.war /usr/local/tomcat/webapps/                                                 0.0s
 => exporting to image                                                                                            0.0s
 => => exporting layers                                                                                           0.0s
 => => writing image sha256:c0061720b45b3a0325e2967f40acce90ff335dcb9b37a030cbd11b099854bf8c                      0.0s
 => => naming to 730335604401.dkr.ecr.ap-south-1.amazonaws.com/heydevopsjavaapp:latest                            0.0s
Authenticating Docker to AWS ECR...
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
Pushing Docker image to ECR...
The push refers to repository [730335604401.dkr.ecr.ap-south-1.amazonaws.com/heydevopsjavaapp]
508889fcd8ed: Pushed
5f70bf18a086: Pushed
8adb690dc00c: Pushed
bc2495b82dd2: Pushed
8262a88fc163: Pushed
7c2fc37591f1: Pushed
db0d44d04640: Pushed
6e9a204c7212: Pushed
931b7ff0cb6f: Pushed
latest: digest: sha256:b62f3841f1733bde6f2d146b875ff095be4d847ed8915424145b8ddbf9ef667c size: 2408
Docker image pushed: 730335604401.dkr.ecr.ap-south-1.amazonaws.com/heydevopsjavaapp:latest
[ec2-user@ip-172-31-40-46 JAVA_PROJECT]$ |
```



Private repositories

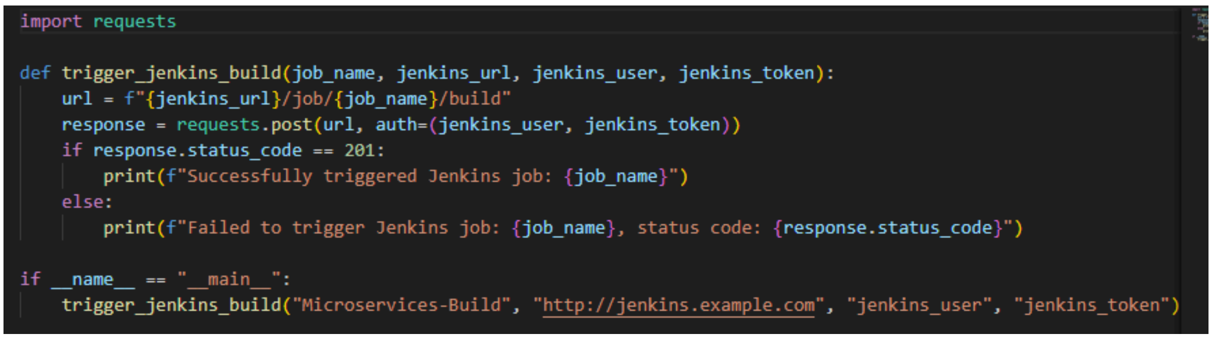
Improved basic scanning
Switch to the improved version of basic scanning, powered by AWS native scanning technology. Learn more.

Repositories (1)

	Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type
○	heydevopsjavaapp	730335604401.dkr.ecr.ap-south-1.amazonaws.com/heydevopsjavaapp	June 20, 2024, 18:51:29 (UTC+05.5)	Disabled	Manual	AES-256

Scenario 4: Automated Jenkins Pipeline Trigger

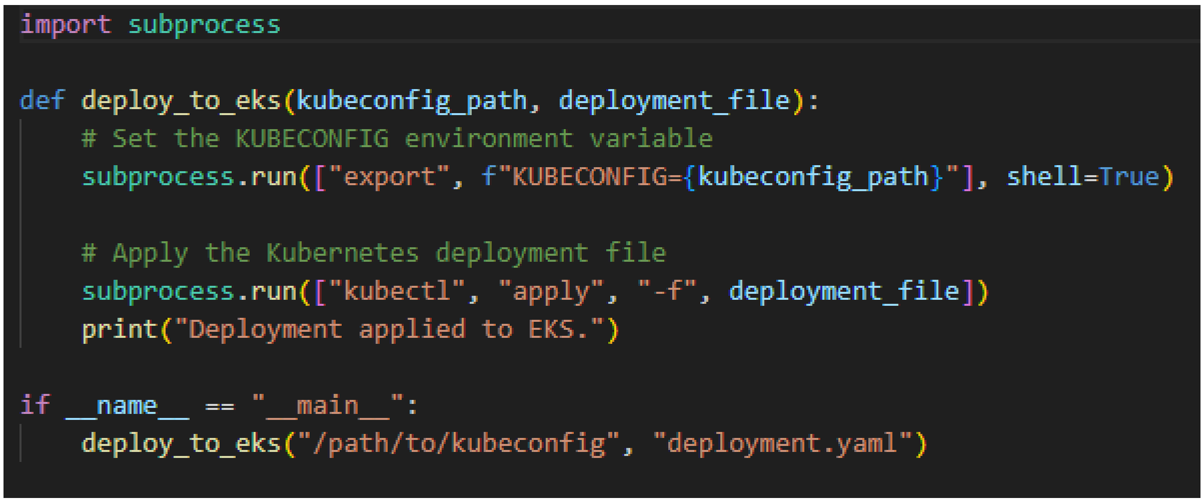
- **Scenario**: Trigger Jenkins pipeline when a new commit is pushed to the repository.



```python
import requests

def trigger_jenkins_build(job_name, jenkins_url, jenkins_user, jenkins_token):
    url = f"{jenkins_url}/job/{job_name}/build"
    response = requests.post(url, auth=(jenkins_user, jenkins_token))
    if response.status_code == 201:
        print(f"Successfully triggered Jenkins job: {job_name}")
    else:
        print(f"Failed to trigger Jenkins job: {job_name}, status code: {response.status_code}")

if __name__ == "__main__":
    trigger_jenkins_build("Microservices-Build", "http://jenkins.example.com", "jenkins_user", "jenkins_token")
```

Scenario 5: Automated Kubernetes Deployment

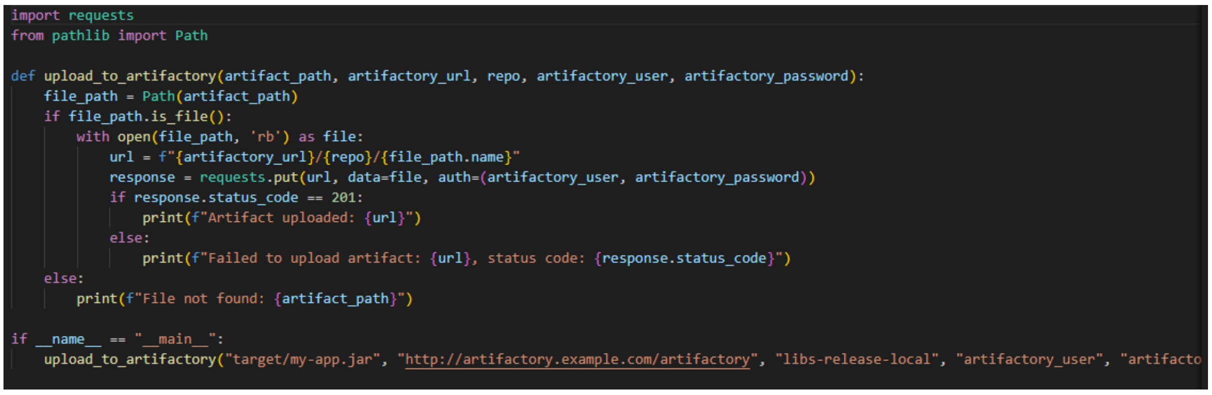
- **Scenario**: Deploy Docker images to EKS.



```python
import subprocess

def deploy_to_eks(kubeconfig_path, deployment_file):
    # Set the KUBECONFIG environment variable
    subprocess.run(["export", f"KUBECONFIG={kubeconfig_path}"], shell=True)

    # Apply the Kubernetes deployment file
    subprocess.run(["kubectl", "apply", "-f", deployment_file])
    print("Deployment applied to EKS.")

if __name__ == "__main__":
    deploy_to_eks("/path/to/kubeconfig", "deployment.yaml")
```

Scenario 6: Automated Artifact Upload to Artifactory

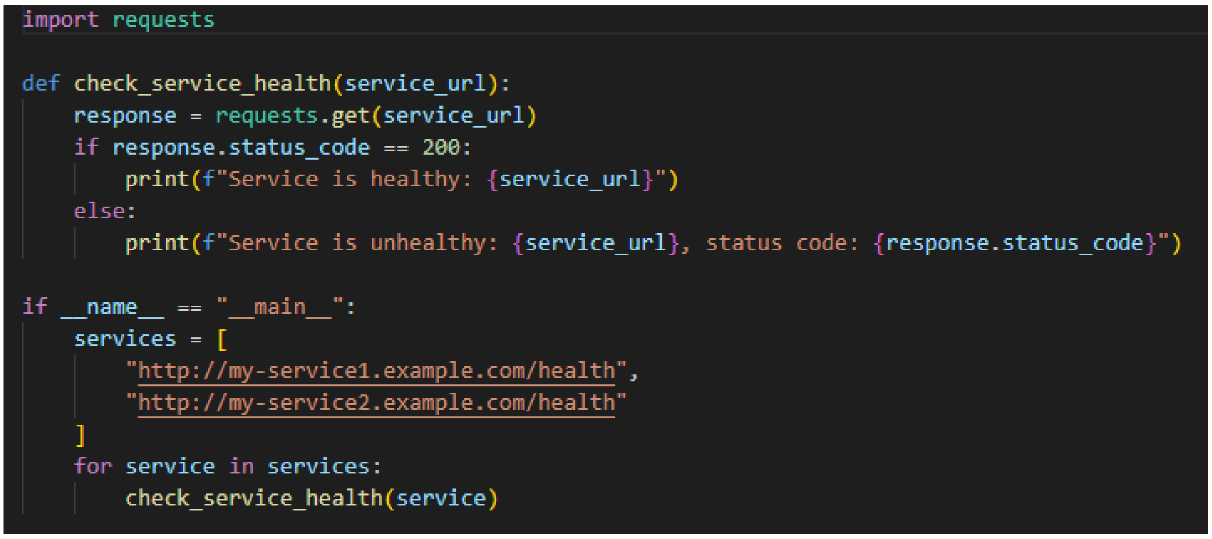
- **Scenario**: Upload built JAR files to Artifactory.



```python
import requests
from pathlib import Path

def upload_to_artifactory(artifact_path, artifactory_url, repo, artifactory_user, artifactory_password):
    file_path = Path(artifact_path)
    if file_path.is_file():
        with open(file_path, 'rb') as file:
            url = f"{artifactory_url}/{repo}/{file_path.name}"
            response = requests.put(url, data=file, auth=(artifactory_user, artifactory_password))
            if response.status_code == 201:
                print(f"Artifact uploaded: {url}")
            else:
                print(f"Failed to upload artifact: {url}, status code: {response.status_code}")
    else:
        print(f"File not found: {artifact_path}")

if __name__ == "__main__":
    upload_to_artifactory("target/my-app.jar", "http://artifactory.example.com/artifactory", "libs-release-local", "artifactory_user", "artifacto
```

Scenario 7: Automated Environment Health Check

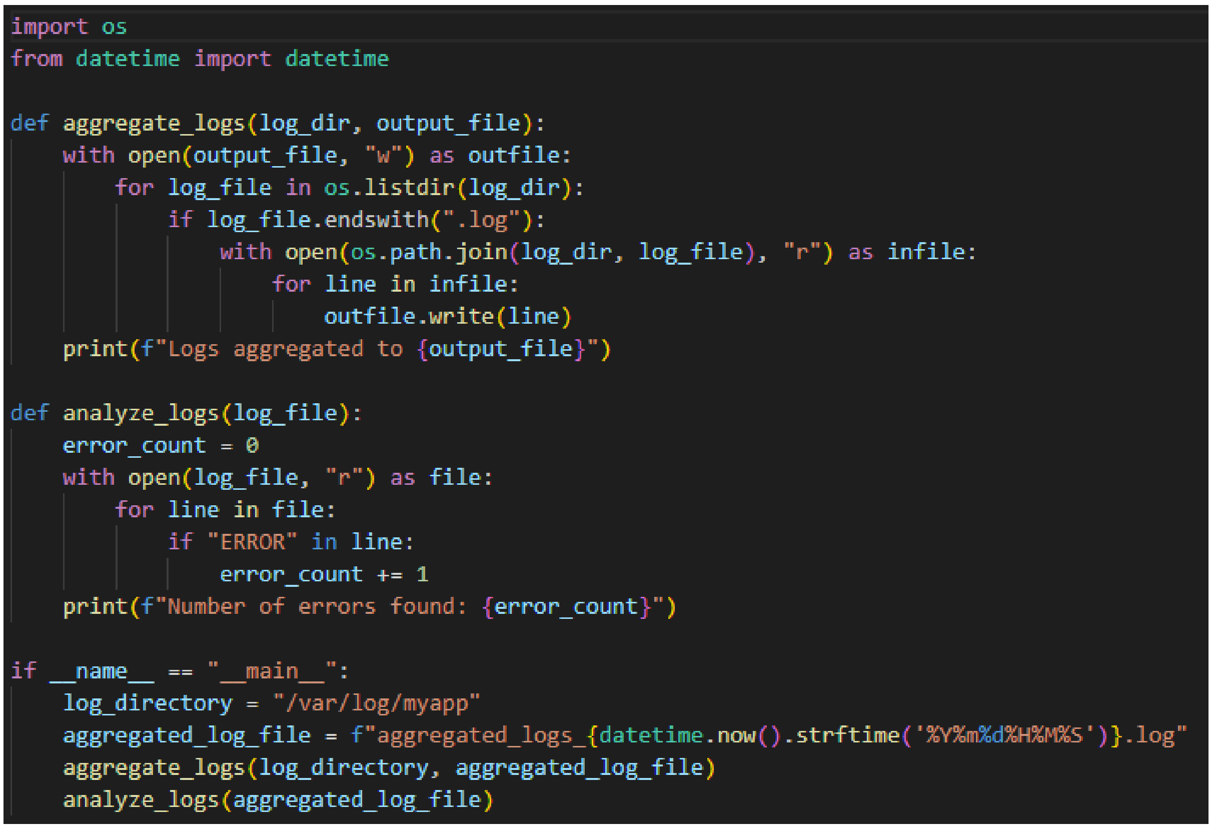
- **Scenario**: Regularly check the health of deployed services in EKS.



```python
import requests

def check_service_health(service_url):
    response = requests.get(service_url)
    if response.status_code == 200:
        print(f"Service is healthy: {service_url}")
    else:
        print(f"Service is unhealthy: {service_url}, status code: {response.status_code}")

if __name__ == "__main__":
    services = [
        "http://my-service1.example.com/health",
        "http://my-service2.example.com/health"
    ]
    for service in services:
        check_service_health(service)
```

Scenario 8: Automated Log Aggregation and Analysis

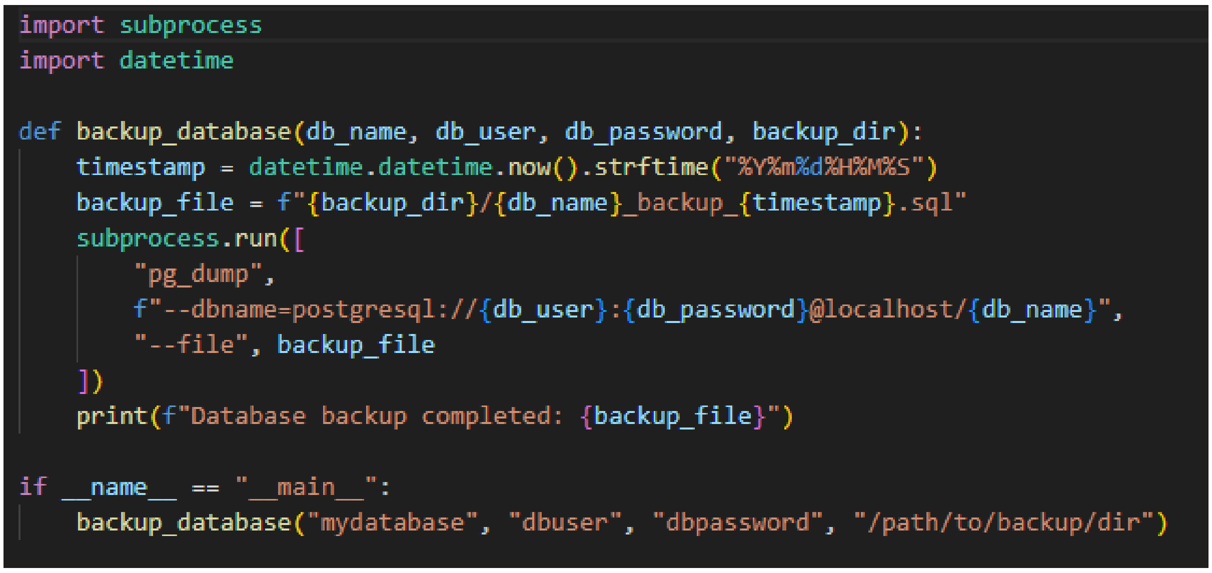
- **Scenario**: Aggregate and analyze logs from different services.



```python
import os
from datetime import datetime

def aggregate_logs(log_dir, output_file):
    with open(output_file, "w") as outfile:
        for log_file in os.listdir(log_dir):
            if log_file.endswith(".log"):
                with open(os.path.join(log_dir, log_file), "r") as infile:
                    for line in infile:
                        outfile.write(line)
    print(f"Logs aggregated to {output_file}")

def analyze_logs(log_file):
    error_count = 0
    with open(log_file, "r") as file:
        for line in file:
            if "ERROR" in line:
                error_count += 1
    print(f"Number of errors found: {error_count}")

if __name__ == "__main__":
    log_directory = "/var/log/myapp"
    aggregated_log_file = f"aggregated_logs_{datetime.now().strftime('%Y%m%d%H%M%S')}.log"
    aggregate_logs(log_directory, aggregated_log_file)
    analyze_logs(aggregated_log_file)
```

Scenario 9: Automated Database Backup

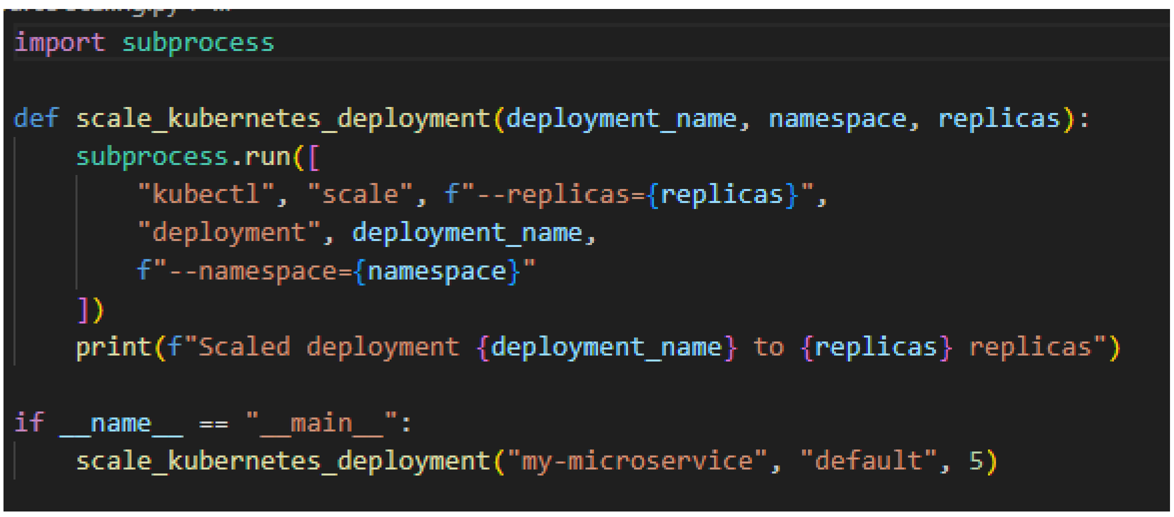
- **Scenario**: Regularly back up your production database.



```python
import subprocess
import datetime

def backup_database(db_name, db_user, db_password, backup_dir):
    timestamp = datetime.datetime.now().strftime("%Y%m%d%H%M%S")
    backup_file = f"{backup_dir}/{db_name}_backup_{timestamp}.sql"
    subprocess.run([
        "pg_dump",
        f"--dbname=postgresql://{db_user}:{db_password}@localhost/{db_name}",
        "--file", backup_file
    ])
    print(f"Database backup completed: {backup_file}")

if __name__ == "__main__":
    backup_database("mydatabase", "dbuser", "dbpassword", "/path/to/backup/dir")
```

Scenario 10: Automated Resource Scaling

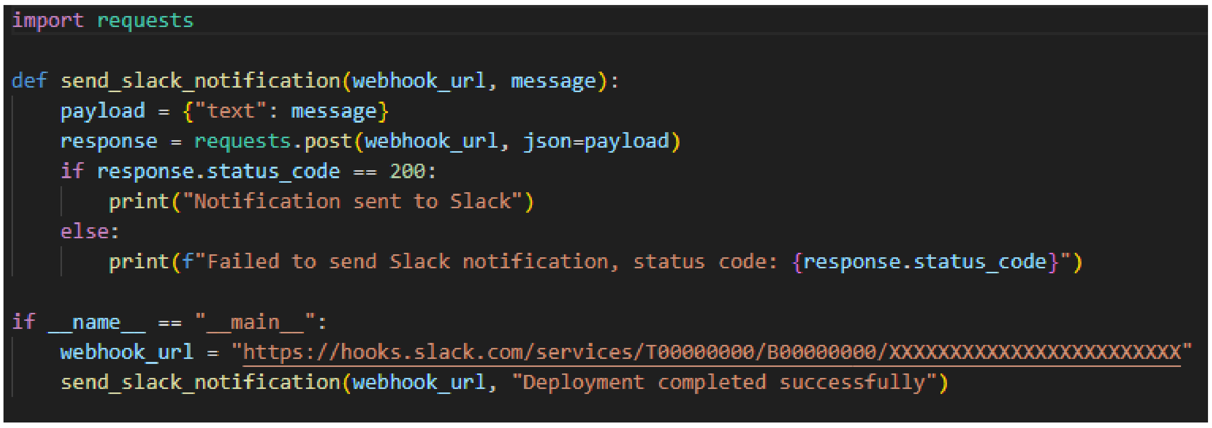
- **Scenario**: Automatically scale Kubernetes resources based on usage metrics.



```python
import subprocess

def scale_kubernetes_deployment(deployment_name, namespace, replicas):
    subprocess.run([
        "kubectl", "scale", f"--replicas={replicas}",
        "deployment", deployment_name,
        f"--namespace={namespace}"
    ])
    print(f"Scaled deployment {deployment_name} to {replicas} replicas")

if __name__ == "__main__":
    scale_kubernetes_deployment("my-microservice", "default", 5)
```

Scenario 11: Automated Slack Notifications

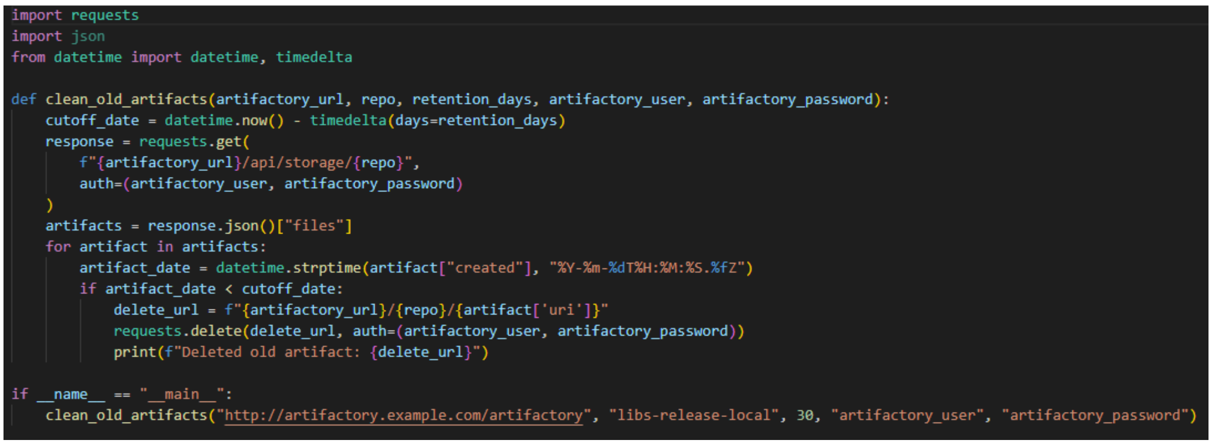
- **Scenario**: Send notifications to Slack for various events (e.g., build failures, successful deployments).



```python
import requests

def send_slack_notification(webhook_url, message):
    payload = {"text": message}
    response = requests.post(webhook_url, json=payload)
    if response.status_code == 200:
        print("Notification sent to Slack")
    else:
        print(f"Failed to send Slack notification, status code: {response.status_code}")

if __name__ == "__main__":
    webhook_url = "https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXXXXXXXXXXXXXXXX"
    send_slack_notification(webhook_url, "Deployment completed successfully")
```

Scenario 12: Automated Artifact Cleanup

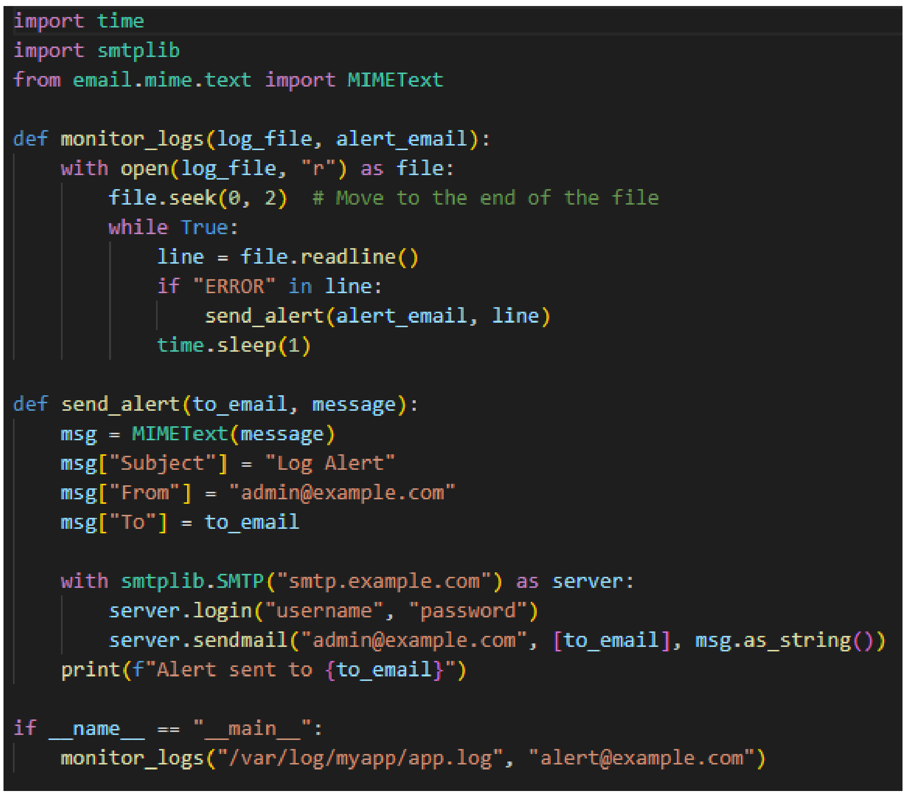
- **Scenario**: Clean up old artifacts from Artifactory to save storage space.



```python
import requests
import json
from datetime import datetime, timedelta

def clean_old_artifacts(artifactory_url, repo, retention_days, artifactory_user, artifactory_password):
    cutoff_date = datetime.now() - timedelta(days=retention_days)
    response = requests.get(
        f"{artifactory_url}/api/storage/{repo}",
        auth=(artifactory_user, artifactory_password)
    )
    artifacts = response.json()["files"]
    for artifact in artifacts:
        artifact_date = datetime.strptime(artifact["created"], "%Y-%m-%dT%H:%M:%S.%fZ")
        if artifact_date < cutoff_date:
            delete_url = f"{artifactory_url}/{repo}/{artifact['uri']}"
            requests.delete(delete_url, auth=(artifactory_user, artifactory_password))
            print(f"Deleted old artifact: {delete_url}")

if __name__ == "__main__":
    clean_old_artifacts("http://artifactory.example.com/artifactory", "libs-release-local", 30, "artifactory_user", "artifactory_password")
```

Scenario 13: Automated Log Monitoring and Alerting

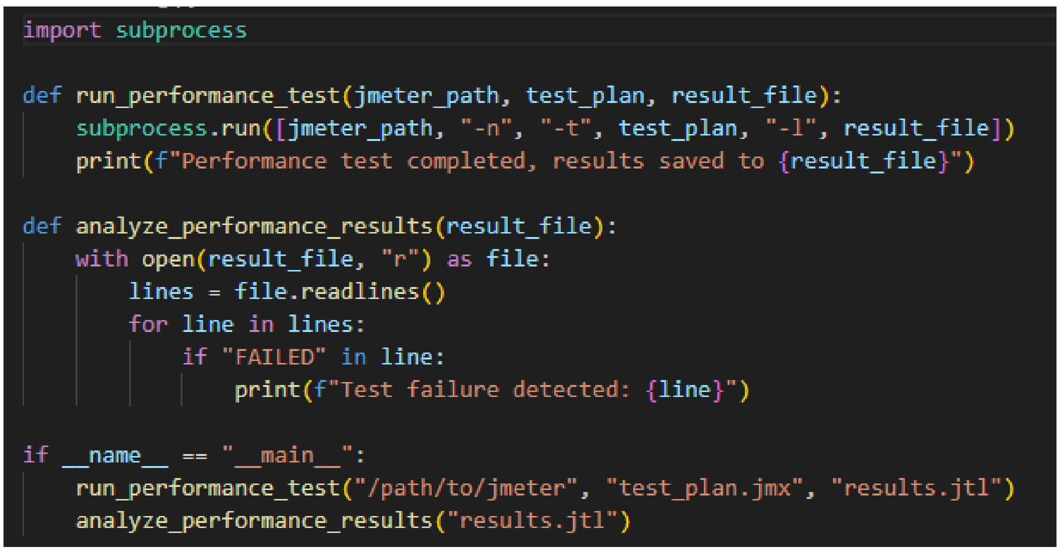
- **Scenario**: Monitor application logs and send alerts for specific patterns or errors.



```python
import time
import smtplib
from email.mime.text import MIMEText

def monitor_logs(log_file, alert_email):
    with open(log_file, "r") as file:
        file.seek(0, 2)  # Move to the end of the file
        while True:
            line = file.readline()
            if "ERROR" in line:
                send_alert(alert_email, line)
            time.sleep(1)

def send_alert(to_email, message):
    msg = MIMEText(message)
    msg["Subject"] = "Log Alert"
    msg["From"] = "admin@example.com"
    msg["To"] = to_email

    with smtplib.SMTP("smtp.example.com") as server:
        server.login("username", "password")
        server.sendmail("admin@example.com", [to_email], msg.as_string())
    print(f"Alert sent to {to_email}")

if __name__ == "__main__":
    monitor_logs("/var/log/myapp/app.log", "alert@example.com")
```

Scenario 14: Automated Performance Testing

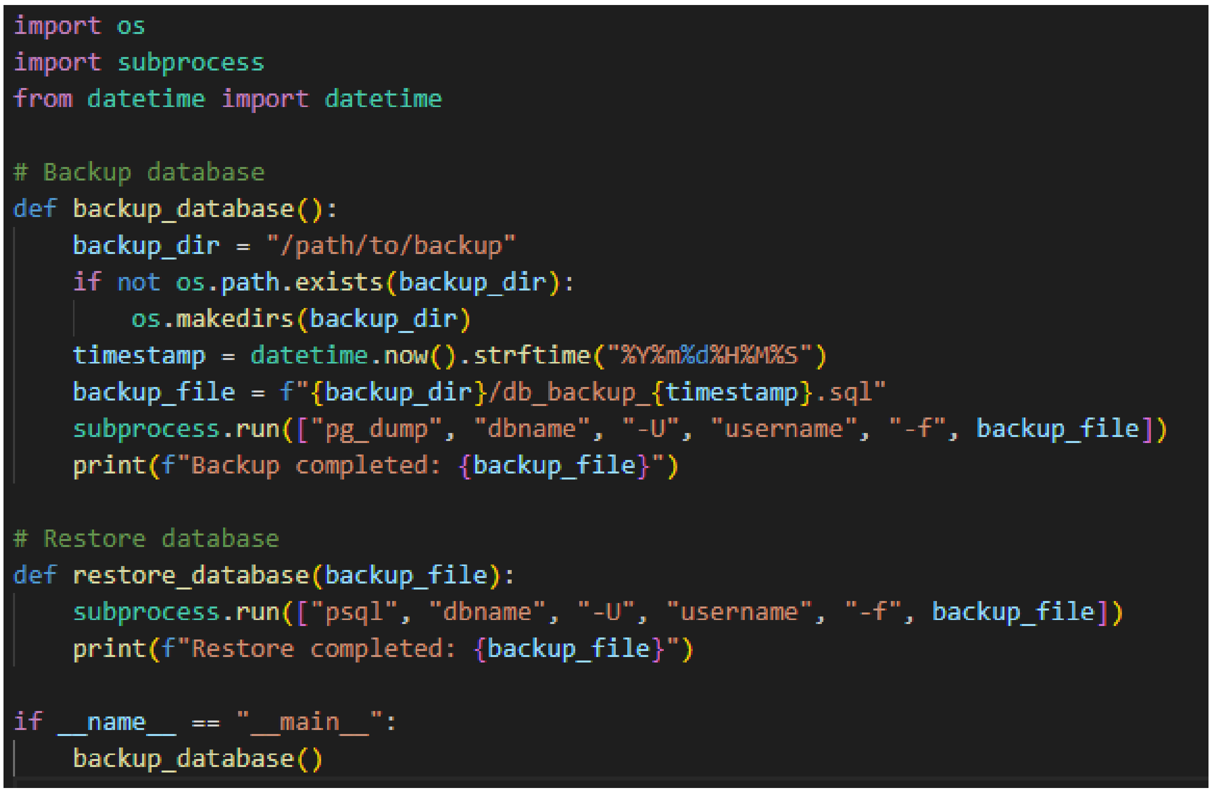
- **Scenario**: Run performance tests on your application using a tool like JMeter and analyze the results.



```python
import subprocess

def run_performance_test(jmeter_path, test_plan, result_file):
    subprocess.run([jmeter_path, "-n", "-t", test_plan, "-l", result_file])
    print(f"Performance test completed, results saved to {result_file}")

def analyze_performance_results(result_file):
    with open(result_file, "r") as file:
        lines = file.readlines()
        for line in lines:
            if "FAILED" in line:
                print(f"Test failure detected: {line}")

if __name__ == "__main__":
    run_performance_test("/path/to/jmeter", "test_plan.jmx", "results.jtl")
    analyze_performance_results("results.jtl")
```

Scenario 15. Automated Backup and Restore

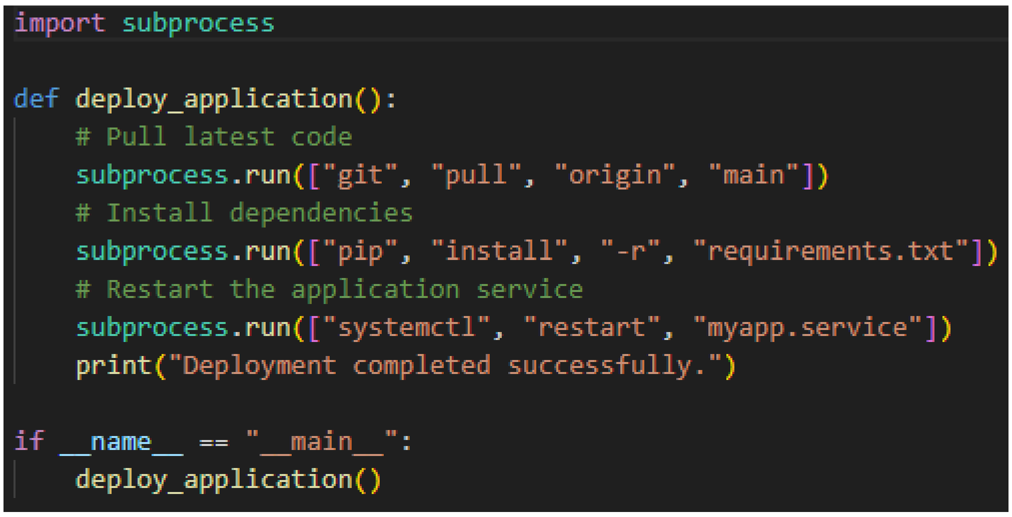
- **Scenario**: Regularly backup databases and restore them in case of data loss.



```python
import os
import subprocess
from datetime import datetime

# Backup database
def backup_database():
    backup_dir = "/path/to/backup"
    if not os.path.exists(backup_dir):
        os.makedirs(backup_dir)
    timestamp = datetime.now().strftime("%Y%m%d%H%M%S")
    backup_file = f"{backup_dir}/db_backup_{timestamp}.sql"
    subprocess.run(["pg_dump", "dbname", "-U", "username", "-f", backup_file])
    print(f"Backup completed: {backup_file}")

# Restore database
def restore_database(backup_file):
    subprocess.run(["psql", "dbname", "-U", "username", "-f", backup_file])
    print(f"Restore completed: {backup_file}")

if __name__ == "__main__":
    backup_database()
```

Scenario 16. Automated Deployment

- **Scenario**: Deploy application to servers automatically when new code is pushed to the repository.



```python
import subprocess

def deploy_application():
    # Pull latest code
    subprocess.run(["git", "pull", "origin", "main"])
    # Install dependencies
    subprocess.run(["pip", "install", "-r", "requirements.txt"])
    # Restart the application service
    subprocess.run(["systemctl", "restart", "myapp.service"])
    print("Deployment completed successfully.")

if __name__ == "__main__":
    deploy_application()
```