Python for DevOps 🚀

Why Python in DevOps? 🤔

- **Easy to Learn** 👶
- **Cross-Platform** 🌍
- **Vast Libraries** 📚
- **Active Community** 🤝

Key Use Cases of Python in DevOps 🔧

- **CI/CD Automation**: Automate pipelines with Jenkins, GitLab CI.
- **Infrastructure as Code (IaC)**: Manage infrastructure via Python with Terraform & AWS Boto3.
- **System Monitoring**: Create alerts with Nagios, Prometheus, or Grafana.
- **API Integrations**: Interact with APIs for cloud services (AWS, Azure).

Core Python Concepts for DevOps 🧠

1. **Data Types** 🔢
    - int, float for numbers
    - str for text
    - list, tuple, set for collections
    - dict for key-value pairs
2. **Functions & Modules** 🔄
    - Define reusable code blocks with def ✏️
    - Split large code into modules for better organization 📂
3. **File Operations** 📂

```
1. with open("file.txt", "w") as f:
2.     f.write("Hello, DevOps!")
```

EXPLOREWITHASHOK

4. Error Handling ⚠️

```
try:
    result = 10 / 0
except ZeroDivisionError:
    print("Can't divide by zero!")
```

5. Automation with Boto3 🤖

```
import boto3

# Create an S3 client
s3 = boto3.client('s3')

# Create a new S3 bucket
bucket_name = 'my-new-bucket'
s3.create_bucket(Bucket=bucket_name)

print(f'S3 bucket "{bucket_name}" created successfully!')

```

6. Log Management 📝

Use Python to parse, analyze, and manage logs for real-time monitoring.

Real-World Examples of Python in DevOps 🌟

1. CI/CD Automation 🚀

Scenario: Automating Jenkins Job Creation

Solution:

```
import requests
from requests.auth import HTTPBasicAuth

jenkins_url = 'http://localhost:8080'
job_name = 'example-job'
jenkins_user = 'admin'
jenkins_token = 'your_api_token'

job_config = '''
<project>
  <builders>
    <hudson.tasks.Shell>
      <command>echo "Hello, DevOps!"</command>
    </hudson.tasks.Shell>
  </builders>
</project>
'''

response = requests.post(
    f'{jenkins_url}/createItem?name={job_name}',
```

EXPLOREWITHAS

```
    data=job_config,
    headers={'Content-Type': 'application/xml'},
    auth=HTTPBasicAuth(jenkins_user, jenkins_token)
)

if response.status_code == 200:
    print(f'Job "{job_name}" created successfully!')
else:
    print(f'Failed to create job: {response.status_code}')

```

2. Infrastructure as Code (IaC) 🌎

Scenario: Managing AWS S3 Buckets

Solution:

```
import boto3

# Initialize S3 client
s3 = boto3.client('s3')

# Create a new S3 bucket
bucket_name = 'my-new-bucket'
s3.create_bucket(Bucket=bucket_name)

print(f'S3 bucket "{bucket_name}" created successfully!')

```

3. System Monitoring 📊

Scenario: Sending Alerts with Prometheus

Solution:

```
from prometheus_client import start_http_server, Summary

# Create a metric to track processing time
REQUEST_TIME = Summary('request_processing_seconds', 'Time spent processing request')

# Decorate function with metric
@REQUEST_TIME.time()
def process_request():
    # Simulate request processing
    import time
    time.sleep(2)

if __name__ == '__main__':
    # Start Prometheus metrics server
    start_http_server(8000)

    # Simulate processing requests
    while True:
        process_request()

```

4. API Integrations 🔗

Scenario: Interacting with AWS EC2

Solution:

```python
import boto3

# Initialize EC2 client
ec2 = boto3.client('ec2')

# Launch a new EC2 instance
response = ec2.run_instances(
    ImageId='ami-0abcdef1234567890',
    MinCount=1,
    MaxCount=1,
    InstanceType='t2.micro',
    KeyName='my-key-pair'
)

instance_id = response['Instances'][0]['InstanceId']
print(f'EC2 instance launched with ID: {instance_id}')

```

5. Backup and Restore Databases 📦

Scenario: Backup MySQL Database Daily

Solution:

```python
import subprocess
from datetime import datetime

# Backup configuration
db_name = 'your_database'
user = 'your_username'
password = 'your_password'
backup_dir = '/path/to/backup'
timestamp = datetime.now().strftime('%Y%m%d%H%M%S')
backup_file = f'{backup_dir}/{db_name}_{timestamp}.sql'

# Command to create backup
backup_command = f'mysqldump -u {user} -p{password} {db_name} > {backup_file}'

# Run the backup command
subprocess.run(backup_command, shell=True, check=True)

print(f'Database backup created at {backup_file}')

```

6. Automating Server Configuration 🔧

Scenario: Automatically Install and Configure Nginx on a Server

Solution:

```python
import paramiko

# Server connection details
hostname = 'your_server_ip'
port = 22
username = 'your_username'
password = 'your_password'

# Commands to run
commands = [
    'sudo apt update',
    'sudo apt install -y nginx',
    'sudo systemctl start nginx',
    'sudo systemctl enable nginx'
]

# Create SSH client
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
ssh.connect(hostname, port, username, password)

for command in commands:
    stdin, stdout, stderr = ssh.exec_command(command)
    print(stdout.read().decode())
    if stderr.read().decode():
        print(f"Error: {stderr.read().decode()}")

ssh.close()

```

7. Deploying Applications with Docker 🐳

Scenario: Deploy a Dockerized Application to a Remote Server

Solution:

```python
import paramiko

# Server connection details
hostname = 'your_server_ip'
port = 22
username = 'your_username'
password = 'your_password'

# Docker commands
docker_commands = [
    'sudo docker pull your_image:latest',
    'sudo docker stop your_container || true',
    'sudo docker rm your_container || true',
    'sudo docker run -d --name your_container your_image:latest'
]

# Create SSH client
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
ssh.connect(hostname, port, username, password)

for command in docker_commands:
    stdin, stdout, stderr = ssh.exec_command(command)
    print(stdout.read().decode())
    if stderr.read().decode():
```

```
        print(f"Error: {stderr.read().decode()}")

ssh.close()

```

8. Handling Alerts and Notifications 📬

Scenario: Send an Email Notification on Critical Errors

Solution:

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

# Email configuration
smtp_server = 'smtp.example.com'
smtp_port = 587
sender_email = 'your_email@example.com'
receiver_email = 'admin@example.com'
password = 'your_password'

# Email content
subject = 'Critical Error Notification'
body = 'A critical error occurred. Please check the logs for details.'
msg = MIMEText(body)
msg['Subject'] = subject
msg['From'] = sender_email
msg['To'] = receiver_email

# Send the email
with smtplib.SMTP(smtp_server, smtp_port) as server:
    server.starttls()
    server.login(sender_email, password)
    server.sendmail(sender_email, receiver_email, msg.as_string())

print('Notification email sent successfully')

```

9. Creating Custom Command-Line Tools 🛠️

Scenario: A Python CLI Tool to Monitor System Resource Usage

Solution:

```python
import argparse
import psutil

# Function to get resource usage
def get_resource_usage():
    cpu_usage = psutil.cpu_percent()
    memory_info = psutil.virtual_memory()
    return cpu_usage, memory_info.percent

# CLI setup
parser = argparse.ArgumentParser(description='Monitor system resource usage.')
parser.add_argument('--cpu', action='store_true', help='Show CPU usage')
parser.add_argument('--memory', action='store_true', help='Show memory usage')
```

```
14. args = parser.parse_args()
15.
16. cpu_usage, memory_usage = get_resource_usage()
17.
18. if args.cpu:
19.     print(f'CPU Usage: {cpu_usage}%')
20. if args.memory:
21.     print(f'Memory Usage: {memory_usage}%')
22.
```

If you like the content follow me on LinkedIn: https://www.linkedin.com/in/ashok-sana

Follow my Whatsapp & telegram community:
https://chat.whatsapp.com/BzX1aruZIH645l29LxvgO3

https://t.me/ExplorewithAshok

Happy learning.....!

EXPLOREW