Shell/Bash Scripting in DevOps

Benefits of Shell Scripting in DevOps

1. **Automation**: Shell scripts can automate repetitive tasks, saving time and reducing human error.
2. **Efficiency**: They enable quick execution of complex tasks, improving operational efficiency.
3. **Consistency**: Scripts ensure consistent execution of tasks, reducing variability.
4. **Flexibility**: Shell scripts can interact with various tools and services, making them highly versatile.
5. **Scalability**: Automating tasks with shell scripts allows for scalable operations in large environments.

1. Automating Deployment on Kubernetes

Use Case: Automating the deployment process of an application on a Kubernetes cluster.

Scenario: Deploying a web application to a Kubernetes cluster, including creating the necessary resources like deployments, services, and config maps.

Shell Script Example:

Sh:

```
#!/bin/bash

# Define variables
NAMESPACE="myapp-namespace"
DEPLOYMENT_NAME="myapp-deployment"
IMAGE="myapp-image:latest"

# Create Kubernetes namespace
echo "Creating Kubernetes namespace..."
kubectl create namespace $NAMESPACE

# Apply ConfigMap
echo "Creating ConfigMap..."
kubectl apply -f - <<EOF
apiVersion: v1
kind: ConfigMap
metadata:
  name: myapp-config
  namespace: $NAMESPACE
data:
  APP_ENV: "production"
EOF

# Apply Deployment
echo "Creating Deployment..."
kubectl apply -f - <<EOF
apiVersion: apps/v1
```

```
kind: Deployment
metadata:
  name: $DEPLOYMENT_NAME
  namespace: $NAMESPACE
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp-container
        image: $IMAGE
        ports:
        - containerPort: 80
        envFrom:
        - configMapRef:
            name: myapp-config
EOF

# Apply Service
echo "Creating Service..."
kubectl apply -f - <<EOF
apiVersion: v1
kind: Service
metadata:
  name: myapp-service
  namespace: $NAMESPACE
spec:
  selector:
    app: myapp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
  type: LoadBalancer
EOF

echo "Deployment completed successfully!"
```

Output:

```
Creating Kubernetes namespace...
namespace/myapp-namespace created
Creating ConfigMap...
configmap/myapp-config created
Creating Deployment...
deployment.apps/myapp-deployment created
Creating Service...
service/myapp-service created
Deployment completed successfully!
```

2. Interacting with APIs (GitHub Example)

Use Case: Automating the creation of GitHub issues via the GitHub API.

Scenario: A script that interacts with the GitHub API to create new issues in a repository based on input parameters.

Shell Script Example:

Sh:

```
#!/bin/bash

# Define GitHub credentials and repository details
GITHUB_USER="your-username"
GITHUB_REPO="your-repo"
GITHUB_TOKEN="your-token"
ISSUE_TITLE="New issue title"
ISSUE_BODY="Description of the new issue"

# Create the issue using the GitHub API
curl -u $GITHUB_USER:$GITHUB_TOKEN -X POST -H "Content-Type:
application/json" \
  -d '{
    "title": "'$ISSUE_TITLE'",
    "body": "'$ISSUE_BODY'"
  }' \
  https://api.github.com/repos/$GITHUB_USER/$GITHUB_REPO/issues
```

Output:

Json:

```
{
  "id": 123456789,
  "number": 1,
  "title": "New issue title",
  "state": "open",
  "body": "Description of the new issue",
  "user": {
    "login": "your-username",
    ...
  }
}
```

3. Monitoring with AWS CloudWatch

Use Case: Monitoring CPU utilization of an EC2 instance and sending alerts if it exceeds a threshold.

Scenario: A script that uses AWS CLI to set up CloudWatch alarms for monitoring EC2 instances.

Shell Script Example:

Sh:

```
#!/bin/bash

# Define variables
INSTANCE_ID="i-1234567890abcdef0"
ALARM_NAME="HighCPUUtilization"
ALARM_THRESHOLD=80

# Create CloudWatch alarm
aws cloudwatch put-metric-alarm --alarm-name $ALARM_NAME --metric-name
CPUUtilization --namespace AWS/EC2 --statistic Average --period 300 --
threshold $ALARM_THRESHOLD --comparison-operator GreaterThanThreshold --
dimensions Name=InstanceId,Value=$INSTANCE_ID --evaluation-periods 2 --
alarm-actions arn:aws:sns:us-east-1:123456789012:my-sns-topic --unit
Percent

echo "CloudWatch alarm created successfully!"
```

Output:

```
CloudWatch alarm created successfully!
```

4. Monitoring and Alerting

Use Case: Monitoring disk usage and sending alerts if usage exceeds a threshold.

Scenario: Ensuring that disk usage on a server does not exceed a critical limit to avoid potential issues.

Shell Script Example:

Sh:

```
#!/bin/bash

# Define threshold (in percentage)
THRESHOLD=80

# Get the current disk usage
DISK_USAGE=$(df / | grep / | awk '{ print $5 }' | sed 's/%//g')

# Check if the disk usage exceeds the threshold
if [ $DISK_USAGE -gt $THRESHOLD ]; then
  # Send an alert (e.g., email or logging)
  echo "Disk usage is at ${DISK_USAGE}%, which is above the threshold of
${THRESHOLD}%!" | mail -s "Disk Usage Alert" admin@example.com
fi
```

Output:

```
Disk usage is at 85%, which is above the threshold of 80%!
```

5. Interacting with APIs

Use Case: Automating the process of creating new issues in a project management tool like Jira.

Scenario: A script that interacts with the Jira API to create new issues based on input parameters.

Shell Script Example:

Sh:

```
#!/bin/bash

# Define Jira credentials and project details
JIRA_URL="https://your-jira-instance.atlassian.net"
JIRA_USER="your-email@example.com"
JIRA_API_TOKEN="your-api-token"
JIRA_PROJECT="PROJ"
ISSUE_SUMMARY="New issue summary"
ISSUE_DESCRIPTION="Description of the new issue"

# Create the issue using the Jira API
curl -u $JIRA_USER:$JIRA_API_TOKEN -X POST -H "Content-Type:
application/json" \
  --data '{
    "fields": {
       "project": {
          "key": "'$JIRA_PROJECT'"
       },
       "summary": "'$ISSUE_SUMMARY'",
       "description": "'$ISSUE_DESCRIPTION'",
       "issuetype": {
          "name": "Task"
       }
   }
}' $JIRA_URL/rest/api/2/issue/
```

Output:

```
Json:

{
  "id": "10001",
  "key": "PROJ-123",
  "self": "https://your-jira-instance.atlassian.net/rest/api/2/issue/10001"
}
```

6. Executing Tasks Based on Conditions

Use Case: Performing different actions based on the status of a service.

Scenario: A script that checks the status of a web service and restarts it if it is not running.

Shell Script Example:

Sh:

```
#!/bin/bash

# Define the service name
SERVICE_NAME="apache2"

# Check the service status
SERVICE_STATUS=$(systemctl is-active $SERVICE_NAME)

# Perform actions based on the service status
if [ "$SERVICE_STATUS" != "active" ]; then
  echo "Service $SERVICE_NAME is not running. Restarting the service..."
  sudo systemctl restart $SERVICE_NAME
  echo "Service $SERVICE_NAME restarted."
else
  echo "Service $SERVICE_NAME is running."
fi
```

Output:

```
Service apache2 is not running. Restarting the service...
Service apache2 restarted.
```

Shell scripting is an indispensable tool in the DevOps toolkit, enabling automation, enhancing monitoring, and simplifying interactions with APIs. These examples demonstrate how shell scripts can streamline operations, improve efficiency, and ensure reliability in a DevOps environment. Let's harness the power of shell scripting to drive automation and innovation in our DevOps practices !