4-1-1: Examining DevOps Tools

After completing this episode, you should be able to:

- Identify and explain the importance of DevOps tools, given a scenario

Description: In this episode, the learner will examine various tools used in DevOps environments to automate and streamline development and deployment. We will examine Ansible, Docker, the ELK Stack, Git, and more.

- Describe the significance of DevOps tools
    - The tools that automate and streamline various stages of development and deployment
    - Faster releases
    - Higher quality software
    - Efficient operations.
- Ansible
    - Ansible automates configuration management, application deployment, and task automation using simple YAML templates.
    - Its agentless design reduces complexity and errors.
- Docker
    - Docker creates, deploys, and runs applications in containers, ensuring consistency across environments.
    - It enhances scalability, CI/CD processes, and collaboration.
- Elasticsearch, Logstash, and Kibana (ELK) stack
    - Elasticsearch: Search and analytics engine for large data volumes.
    - Logstash: Ingests, transforms, and sends data to Elasticsearch.
    - Kibana: Visualizes data through charts and dashboards.
    - The ELK stack centralizes logging and real-time analysis.
- Git
    - Git is a version control system that tracks code changes, supports collaboration, and facilitates branching and merging.
    - It is essential for team development and code integrity.
- GitHub Actions
    - GitHub Actions automates workflows within GitHub repositories using YAML files.
    - It enhances CI/CD processes and integrates with GitHub events.
- Grafana
    - Grafana visualizes metrics from multiple data sources with customizable dashboards and alerts.
    - It aids in monitoring system performance and decision-making.
- Jenkins
    - Jenkins automates CI/CD processes, allowing frequent code integration and early issue detection.
    - Its plugin ecosystem supports various development stages.
- Kubernetes
    - Kubernetes orchestrates containerized applications, automating deployment, scaling, and management.
    - It ensures reliability and efficiency in large-scale environments.
- Terraform
    - Terraform defines and provisions infrastructure as code, supporting multiple cloud providers.
    - It improves consistency, reduces errors, and facilitates collaboration.

Additional Resources

- Ansible

```
# ansible-playbook.yml
- name: Install and start Apache
  hosts: webservers
  become: yes
  tasks:
    - name: Ensure Apache is installed
      apt:
        name: apache2
        state: present
    - name: Ensure Apache is running
      service:
        name: apache2
        state: started
```

- Docker

```
# Dockerfile
FROM python:3.8-slim
WORKDIR /app
COPY . /app
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
```

- ELK Stack (Logstash)

```
# logstash.conf
input {
  file {
    path => "/var/log/syslog"
    start_position => "beginning"
  }
}
output {
  elasticsearch {
    hosts => ["localhost:9200"]
    index => "syslog-%{+YYYY.MM.dd}"
  }
}
```

- Git

```
# Git commands
git init
git add .
git commit -m "Initial commit"
git branch -M main
git remote add origin <repository_url>
git push -u origin main
```

- GitHub Actions

```
# .github/workflows/ci.yml
name: CI
on: [push, pull_request]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Set up Python
        uses: actions/setup-python@v2
        with:
          python-version: 3.8
```

```
      - name: Install dependencies
        run: pip install -r requirements.txt
      - name: Run tests
        run: pytest
```

- Grafana (Dashboard JSON)

```
{
  "dashboard": {
    "id": null,
    "title": "Sample Dashboard",
    "panels": [
      {
        "type": "graph",
        "title": "CPU Usage",
        "targets": [
          {
            "expr": "node_cpu_seconds_total",
            "format": "time_series"
          }
        ],
        "datasource": "Prometheus"
      }
    ]
  }
}
```

- Kubernetes

```
# deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
```

- Jenkins (using groovy)

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                sh 'make build'
            }
        }
        stage('Test') {
            steps {
                sh 'make test'
            }
        }
        stage('Deploy') {
```

```
            steps {
                sh 'make deploy'
            }
        }
    }
}
```

- Terraform

```
# main.tf
provider "aws" {
  region = "us-west-2"
}

resource "aws_instance" "example" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"

  tags = {
    Name = "example-instance"
  }
}
```