DevOps Tools Explained in 25 Minutes

Tools & Topics we are covering in this Video

- **Git & GitHub –** Tools for version control and collaborative source code management.
- **Bash & Shell Scripting** - Automating tasks and managing system operations efficiently.
- **Python for Automation and Scripting** - Versatile language for task automation and scripting.
- **Linux based OS** - Reliable and secure platform for development and deployment.
- **Docker** - Containerization tool for creating and managing portable application environments.
- **Kubernetes** - Orchestrating and managing containerized applications at scale.
- **AWS for Cloud Computing** - Scalable and flexible cloud services for computing and storage.
- **CI-CD Pipelines** - Streamlining software delivery through automated build, test, and deploy.
- **Jenkins & GitHub Actions** - Tools for automating and managing CI/CD workflows.
- **Terraform** - Infrastructure as Code (IaC) tool for provisioning and managing resources.
- **Ansible** - Configuration management and automation tool for IT environments.
- **ELK Stack** - Suite for centralized logging, monitoring, and visualization of data.
- **Prometheus** - Open-source monitoring and alerting toolkit for applications.
- **Grafana** - Visualization tool for monitoring metrics and creating dashboards.
- **GitOps & Argo CD** - Declarative approach to CI/CD using Git as the source of truth.

What is Git & GitHub ?

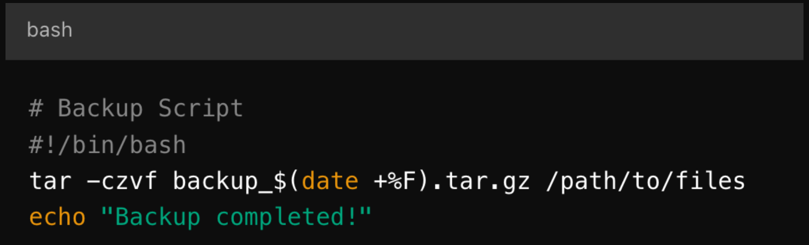
- **Definition**
  - Git is a distributed version control system for tracking changes in source code.
  - GitHub is a cloud-based platform for hosting and collaborating on Git repositories.
- **Examples**:
  - **Version Control**: Track changes in your codebase, revert to previous versions, and collaborate with others.
  - **Practical Use**:
    - Create a new repository
    - Commit the changes in the Repository
    - Push the changes in the repository

```
git init
git add .
git commit -m "Initial commit"
git remote add origin <repository_url>
git push -u origin main
```

Bash & Shell Scripting

- **Definition**:
  - Bash is a Unix shell used for command-line interface operations and scripting.
  - Shell scripting automates repetitive tasks.
- **Examples**:
  - **Automation**: Write a script to back up files.
- **Practical Use**:
  - Perform the backup of file in specified format by writing a shell script.
  - Run the script as "backup-script.sh"

BASH
THE BOURNE-AGAIN SHELL



bash

```bash
# Backup Script
#!/bin/bash
tar -czvf backup_$(date +%F).tar.gz /path/to/files
echo "Backup completed!"
```

Python for Automation and Scripting

- **Definition**:
  - Python is widely used for scripting, task automation, and data processing.
- **Examples**:
  - **Web Scraping**: Use `BeautifulSoup` to extract data from websites.
- **Practical Use**:
  - Web Scraping
  - Automation scripts to create S3 bucket in AWS

```python
import requests
from bs4 import BeautifulSoup

response = requests.get("https://example.com")
soup = BeautifulSoup(response.content, "html.parser")
print(soup.title.text)
```

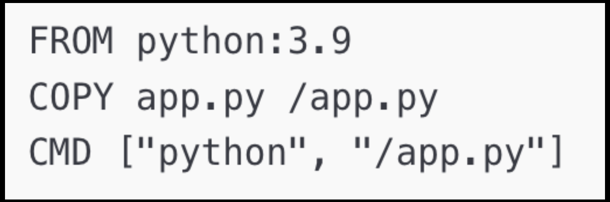
Linux-based OS

- **Explanation**:
  - Linux is an open-source operating system kernel used for servers, development, and cloud computing.
- **Examples**:
  - **Server Management**: Use `crontab` to schedule tasks.
  - Ubuntu OS, Debian, RHEL
- **Practical Use**
  - Server Management
  - Hosting applications on web servers such as Apache, Nginx

```
# Add a cron job to run a script daily
crontab -e
# Add the following line:
0 2 * * * /path/to/script.sh
```

Docker & Its Purpose

- Docker is an open-source platform designed to simplify the development, deployment, and management of applications by using **containerization**.
- Containers are lightweight, portable units that package an application and its dependencies together, ensuring consistency across different environments.
- Key Features of Docker:
  - **Containerization**: Docker containers encapsulate everything an application needs to run (code, runtime, libraries, and configurations) in a single package.
  - **Portability**: Containers can run consistently across different environments, such as development, testing, and production, whether on a developer's laptop, a server, or in the cloud.
  - **Isolation**: Each container operates in its own isolated environment, ensuring that applications don't interfere with each other.
  - **Efficiency**: Containers share the host operating system's kernel, making them more lightweight and faster to start compared to virtual machines (VMs).



```
FROM python:3.9
COPY app.py /app.py
CMD ["python", "/app.py"]
```

Kubernetes

- Kubernetes (K8s) is an open-source platform for automating the deployment, scaling, and management of containerized applications. It organizes containers into **pods**, manages their lifecycle, and ensures high availability and scalability.
- Kubernetes is widely used to simplify running applications in modern, distributed systems.
- **Key Features**:
  - **Orchestration**: Manages containers across a cluster of machines.
  - **Scaling**: Automatically adjusts resources based on demand.
  - **Self-Healing**: Restarts or replaces failed containers.
  - **Load Balancing**: Distributes traffic across containers.
- Example:
  - Imagine you're running an e-commerce app with **frontend**, **backend**, and **database** components in containers. Kubernetes can:
    - Deploy these containers across multiple servers.
    - Ensure the app scales automatically during high traffic (like a sale).
    - Restart failed components without manual intervention.

AWS (Amazon Web Services)

- AWS (Amazon Web Services) is a cloud computing platform offering scalable solutions for computing, storage, databases, networking, AI, and security.
- Key services include **EC2 (virtual servers), S3 (storage), RDS (databases), Lambda (serverless computing), and CloudWatch (monitoring).** AWS enables businesses to build, deploy, and scale applications efficiently with a pay-as-you-go model.
- AWS provides a wide range of cloud services. Here are some key examples:
    - **Compute**: EC2 (virtual servers), Lambda (serverless computing)
    - **Storage**: S3 (object storage), EBS (block storage), Glacier (archival storage)
    - **Databases**: RDS (managed SQL databases), DynamoDB (NoSQL), Redshift (data warehousing)
    - **Networking**: VPC (private cloud), Route 53 (DNS service)
    - **AI/ML**: SageMaker (machine learning), Rekognition (image recognition), Comprehend (NLP)
    - **Security**: IAM (access management), Shield (DDoS protection)
    - **Monitoring**: CloudWatch (logging & monitoring), CloudTrail (audit logs)
- AWS is widely used for hosting websites, running enterprise applications, big data analytics, and AI-driven solutions.

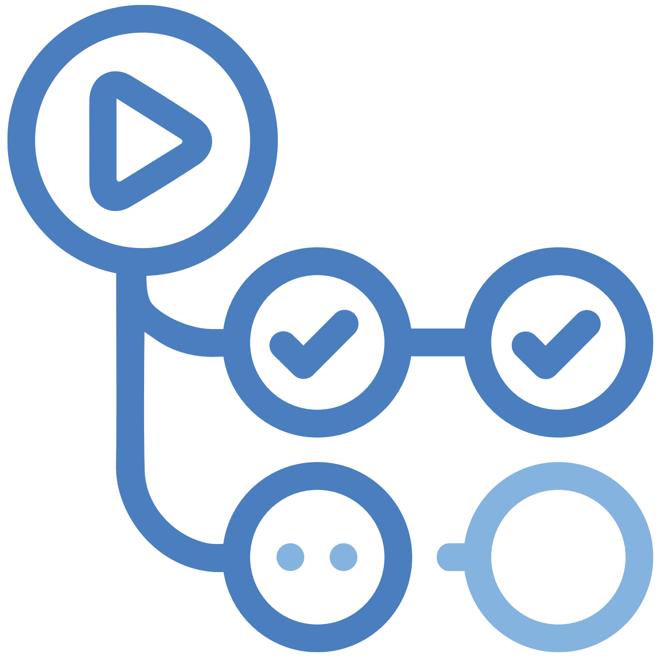
CI-CD Pipelines

- CI/CD (Continuous Integration & Continuous Deployment) pipelines automate software development, testing, and deployment. They help deliver updates faster and with fewer errors.
- Key Components:
    - **Continuous Integration (CI)**: Automates code integration and testing (e.g., Jenkins, GitHub Actions).
    - **Continuous Deployment (CD)**: Automatically deploys tested code to production (e.g., AWS CodeDeploy, GitLab CI/CD).
    - **Version Control**: Manages code changes (e.g., Git, GitHub, Bitbucket).
    - **Build & Testing**: Automates code compilation and testing (e.g., Jenkins, CircleCI).
    - **Deployment & Monitoring**: Deploys code and tracks performance (e.g., Kubernetes, AWS CodePipeline).
- CI/CD pipelines improve software quality, speed up releases, and reduce manual errors.

Jenkins for CI-CD Pipelines

- Jenkins is an open-source **CI/CD automation tool** used for building, testing, and deploying applications.
- **Key Features:**
  - **Pipeline Automation**: Uses scripted or declarative pipelines.
  - **Plugin Support**: Integrates with Git, Docker, Kubernetes, etc.
  - **Scalability**: Supports distributed builds across multiple machines.
  - **Self-Hosted**: Runs on-premises or in the cloud.
- Jenkins helps automate software delivery, improving efficiency and reliability.

GitHub Actions for CI-CD Pipelines



- GitHub Actions is a **CI/CD tool** built into GitHub for automating workflows directly from repositories.
- **Key Features:**
  - **YAML-Based Workflows**: Define CI/CD pipelines as code.
  - **Integration with GitHub**: Automates builds, tests, and deployments on code changes.
  - **Reusable Actions**: Share and reuse workflows across projects.
  - **Cloud-Based**: No need for self-hosted servers.
- GitHub Actions makes CI/CD seamless for GitHub projects, improving development speed and automation.

Terraform
(Infrastructure as Code Tool)

- **Terraform** is an **Infrastructure as Code (IaC)** tool that automates cloud resource provisioning. It enables users to define and manage infrastructure using code.
- Key Features:
  - **Declarative Configuration**: Uses HCL (HashiCorp Configuration Language) to define infrastructure.
  - **Multi-Cloud Support**: Works with AWS, Azure, GCP, Kubernetes, and more.
  - **State Management**: Tracks infrastructure changes via a state file.
  - **Modular & Reusable**: Allows reusing configurations for efficiency.
  - **Automation**: Automates provisioning, scaling, and updates.
- Terraform simplifies infrastructure management, ensuring consistency, scalability, and version control.

Ansible for Infrastructure Configuration

- Ansible is an **open-source automation tool** used for configuration management, application deployment, and orchestration.
- **Key Features:**
  - **Agentless**: No need to install software on target systems.
  - **YAML-Based Playbooks**: Uses simple, human-readable configuration files.
  - **Multi-Platform Support**: Works with Linux, Windows, cloud, and containers.
  - **Idempotent Execution**: Ensures consistent results without repeated changes.
  - **Scalability**: Manages thousands of servers efficiently.
- Ansible simplifies IT automation, reducing manual tasks and improving system reliability.

ELK Stack

- ELK Stack (Elasticsearch, Logstash, Kibana) is a **log management and data analytics** platform used for real-time monitoring and visualization.
- **Key Components:**
  - **Elasticsearch**: A search and analytics engine for storing and querying logs.
  - **Logstash**: A data processing pipeline that collects, transforms, and sends logs to Elasticsearch.
  - **Kibana**: A visualization tool for analyzing and displaying log data.
- **Key Features:**
  - **Centralized Logging**: Aggregates logs from multiple sources.
  - **Real-Time Monitoring**: Helps detect issues quickly.
  - **Scalability**: Handles large volumes of log data.
  - **Integration**: Works with cloud, containers, and security tools.
- ELK Stack is widely used for **log analysis, security monitoring, and performance tracking**.

Prometheus

Prometheus is an **open-source monitoring and alerting tool** designed for collecting and analyzing metrics from applications and infrastructure.

- **Key Features:**
  - **Time-Series Data Storage**: Stores metrics with timestamps for trend analysis.
  - **Pull-Based Monitoring**: Collects data from targets using HTTP endpoints.
  - **Powerful Querying (PromQL)**: Allows flexible data analysis.
  - **Alerting (Alertmanager)**: Sends notifications based on defined conditions.
  - **Scalability**: Works well in dynamic environments like Kubernetes.
- Prometheus is widely used for **performance monitoring, system health tracking, and cloud-native observability**.

Grafana

- Grafana is an **open-source data visualization and monitoring tool** used for creating interactive dashboards.
- **Key Features:**
    - **Multi-Source Support**: Integrates with Prometheus, Elasticsearch, InfluxDB, AWS, and more.
    - **Custom Dashboards**: Provides real-time graphs, charts, and alerts.
    - **Alerting System**: Sends notifications via email, Slack, and other channels.
    - **User Access Control**: Manages permissions for teams.
    - **Extensible Plugins**: Supports additional data sources and visualization panels.
- Grafana is widely used for **monitoring infrastructure, applications, and business metrics**.

GitOps

- GitOps is a **DevOps approach** that uses **Git as the single source of truth** for managing infrastructure and application deployments.
- **Key Features:**
  - **Declarative Infrastructure**: Configurations are stored in Git repositories.
  - **Automated Deployments**: Changes in Git trigger updates in the environment.
  - **Version Control**: Enables rollback and auditability.
  - **Improved Collaboration**: Ensures consistency across teams.
- GitOps enhances **automation, reliability, and security** in cloud-native deployments.

ArgoCD

- ArgoCD is a **GitOps-based continuous deployment tool** for Kubernetes.
- **Key Features:**
  - **Declarative Deployment**: Syncs Kubernetes clusters with Git repositories.
  - **Automated Rollbacks**: Reverts to previous versions on failure.
  - **Real-Time Monitoring**: Provides a UI and CLI for tracking application state.
  - **Multi-Cluster Management**: Deploys across multiple Kubernetes clusters.
- ArgoCD simplifies **Kubernetes application deployment, monitoring, and version control**.

What's Next

- In the next video I will be explaining the DevOps Architecture and this would be the last tutorial of the introduction to DevOps
- After this, I will start giving the in-depth knowledge of tools, we will proceed with streamlined tutorial of the DevOps
- Python will be parallel tutorial and its video will be uploaded in separate Playlist, and I will be referencing the relevant topics with the python tutorials as well.