

 “*DevOps Tools Explained – Everything You Need to Know Before You Start!*”

8 DevOps Tools You Need to Learn (Detailed Notes)

1 GIT – Repository & Version Control System

What is Git?

Git is a **version control system (VCS)** that helps you **track, manage, and collaborate on code changes**.

It lets multiple developers work on the same project without overwriting each other's work.

Why It's Important in DevOps:

- Every DevOps workflow starts with **source code**.
- Git ensures your application code, infrastructure code, and configuration files are stored in a **central repository** like **GitHub**, **GitLab**, or **Bitbucket**.
- Enables **collaboration, branching, rollback, and versioning**.

Common Commands:

```
git init      # Initialize repository  
git add .     # Stage changes  
git commit -m "msg"  # Save version  
git push      # Push to remote repo  
git pull      # Fetch latest code
```

Real-world Example:

When developers push code to GitHub, it automatically triggers the **CI/CD pipeline** to build and deploy the application.

2 CI/CD Tools – Continuous Integration / Continuous Deployment

What is CI/CD?

- **CI (Continuous Integration):** Automatically builds and tests your code whenever a developer pushes changes.
- **CD (Continuous Deployment/Delivery):** Automatically deploys the tested code to servers or cloud environments.

Why It's Important:

Without CI/CD, deploying software would be manual, slow, and error-prone.

These tools make delivery **faster, reliable, and repeatable**.

Popular Tools:

- **Jenkins:** Open-source automation server, widely used for custom pipelines.
- **Azure DevOps:** Cloud-based CI/CD and project management suite from Microsoft.
- **GitHub Actions:** Lightweight CI/CD directly inside GitHub repositories.

Real-world Example:

When you commit code in GitHub:

1. Jenkins or Azure DevOps pipeline builds the code
2. Runs unit tests
3. Deploys automatically to Azure Web App or Kubernetes

3 IaC – Infrastructure as Code

What is IaC?

Infrastructure as Code (IaC) means writing **code to create and manage infrastructure** (servers, VMs, networks, databases, etc.) instead of doing it manually through cloud consoles.

Why It's Important:

- Automates environment creation
- Keeps infrastructure **consistent and repeatable**
- Enables **version control** for infrastructure
- Ideal for **multi-cloud** and **disaster recovery**

Popular IaC Tools:

- **Terraform (HashiCorp)**: Cloud-agnostic; works with Azure, AWS, and GCP.
- **AWS CloudFormation**: AWS-native IaC tool.
- **Azure Bicep**: Microsoft's declarative IaC language for Azure.

Example Terraform Code:

```
resource "azurerm_virtual_machine" "vm" {  
    name    = "dev-vm"  
    location = "East US"  
  
    ...  
}
```

 This code automatically provisions a VM in Azure.

Configuration Management – **Ansible**

What is Configuration Management?

It's the process of **automating the setup, configuration, and maintenance** of servers.

Why It's Important:

Once infrastructure is created using IaC, tools like **Ansible** configure software, services, and environments consistently.

Common Tools:

- **Ansible (by Red Hat)**: Agentless, uses SSH
- **Chef / Puppet**: Agent-based older tools

Example Ansible Playbook:

```
- name: Install Apache Web Server  
  
hosts: webservers  
  
become: yes  
  
tasks:
```

```
- name: Install apache
```

```
apt:
```

```
  name: apache2
```

```
  state: present
```

-  Ansible reads YAML playbooks and executes tasks on multiple machines simultaneously.
-

5 Cloud Platforms – Azure, AWS, GCP

What is Cloud Computing?

Cloud platforms provide **on-demand infrastructure** (servers, databases, storage, networking) over the internet — no need to maintain physical data centers.

Why It's Important in DevOps:

- DevOps pipelines deploy applications directly to cloud environments.
- Enables **scalability, high availability, and cost optimization**.

Major Cloud Providers:

- **Azure (Microsoft)** – Deep integration with enterprise tools, great for hybrid cloud.
- **AWS (Amazon Web Services)** – Largest ecosystem and global reach.
- **GCP (Google Cloud Platform)** – Strong in AI/ML and data analytics.

Example:

Terraform can deploy infrastructure to any of these clouds.

Jenkins pipelines can deploy containers to **AKS (Azure Kubernetes Service), EKS (AWS), or GKE (Google)**.

6 Monitoring & Observability – Datadog

What is Monitoring?

Monitoring tools continuously track the **health, performance, and logs** of your infrastructure and applications.

Why It's Important:

- Detect performance bottlenecks
- Get alerts before outages happen
- Helps in **debugging** and **incident response**

Popular Tools:

- **Datadog:** Cloud-based all-in-one monitoring tool
- **Prometheus + Grafana:** Open-source alternatives
- **Azure Monitor / AWS CloudWatch:** Cloud-native tools

Example Use:

Datadog collects metrics from VMs, Kubernetes, and applications — and visualizes them in dashboards.

You can also set alerts like “CPU usage > 80%” or “App response time > 2s”.

7 Kubernetes – Container Orchestration

What is Kubernetes?

Kubernetes (K8s) is an **open-source system** that automates the **deployment, scaling, and management** of containerized applications (like Docker containers).

Why It's Important:

- Manages hundreds of containers efficiently
- Provides **auto-scaling, self-healing, load balancing, and rolling updates**
- Forms the backbone of **modern DevOps** and **cloud-native applications**

Core Components:

- **Pod:** Smallest deployable unit (contains containers)
- **Node:** Worker machine
- **Deployment:** Defines desired app state
- **Service:** Exposes app to network

Example:

A Jenkins pipeline builds a Docker image → pushes it to Azure Container Registry → deploys it to an **AKS cluster (Azure Kubernetes Service)**.

8 Scripting & Declarative Files – **JSON & YAML**

What are JSON and YAML?

They are **data serialization formats** used to define configuration and automation in almost every DevOps tool.

Why It's Important:

- Used in **Terraform, Ansible, Kubernetes, CI/CD pipelines, and APIs**.
- YAML is human-readable, JSON is machine-friendly.

Examples:

YAML (Ansible / Kubernetes):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
```

JSON (Terraform / APIs):

```
{
  "location": "East US",
  "vmSize": "Standard_B1s"
}
```

-  Understanding JSON/YAML is mandatory to write DevOps automation scripts.
-

Summary – DevOps Tools at a Glance

Category	Tools	Purpose	Example Usage
Version Control	Git	Manage source code	GitHub repo for app
CI/CD	Jenkins, Azure DevOps, GitHub Actions	Automate build & deploy	Auto deploy to Azure Web App
IaC	Terraform, CloudFormation, Bicep	Provision infra via code	Create VMs, Networks, Storage
Config Mgmt	Ansible	Configure servers & apps	Install packages via playbooks
Cloud	Azure, AWS, GCP	Run infra & apps	Host containers or VMs
Monitoring	Datadog	Track performance	Visualize metrics & logs
Containers	Kubernetes	Manage container workloads	Run microservices at scale
Scripting	JSON, YAML	Define automation & configs	Used in IaC, K8s, CI/CD

 **Recommended Learning Order:**

Git → CI/CD → Terraform → Ansible → Cloud (Azure/AWS/GCP) → Kubernetes → Datadog → YAML/JSON
