|  |  |
| --- | --- |
| **Topic** | DevOps |
| **Document Name** | DEVOPS-EX-01 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Difficulty Level** | | | |
| **Beginner** | **Junior** | **Senior** | **Expert** |
| □ | ■ | □ | □ |

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Ver | Comments |
| 01.07.2025 | Alican Çağdaş | 1.0 | Initial Draft |
| 19.07.2025 | Mennan Tekbir | 1.1 | Revisions |

# DevOps

## Exercise DEVOPS-EX-01:

Please answer the questions below

1. Please describe Configuration as Code (CaC) and Infrastructre as Code (IaC) with an example
2. What are the main differences between tools like Terraform, Ansible, and CloudFormation?
3. What is Docker Compose and what are its main use cases?
4. What is the default network type in Docker Compose, and how do services communicate with each other by default?
5. Given the following partial docker-compose.yml file, fill in the missing parts to define a simple web application with one backend (using the python:3.9 image) and one frontend (using the nginx:latest image) service. The backend should be accessible to the frontend on port 5000.

version: '3.8'  
 services:  
   backend:  
 image: python:3.9  
 # Add necessary configuration for the backend service  
  
   frontend:  
 image: nginx:latest  
 # Add necessary configuration for the frontend service  
 # Ensure it can communicate with the backend on port 5000

1. What is the difference between Continuous Integration (CI) and Continuous Deployment (CD)?
2. What is a pipeline? In a typical CI/CD pipeline, what steps would you include to ensure code quality and safe deployment?
3. How would you configure a pipeline to deploy only when code is merged to the main branch, but run tests on every pull request?
4. You have deployed Prometheus and Grafana using Docker Compose to monitor your application. However, when you open Grafana, you cannot see any metrics from Prometheus. List possible reasons why Grafana cannot display metrics from Prometheus in this setup and suggest concrete steps or configuration changes to resolve this issue.
5. List 6 commonly used DevOps tools and briefly describe them.
6. Regarding to Branching strategies in Version Control Systems, which would you prefer and why?
7. Write 10 widely used git commands and describe them

## Solution of DEVOPS-EX-01:

Please answer all question.

Create a LinkedIn post for the answer of question MODULO[calculator](https://www.omnicalculator.com/math/remainder)( your phone number, 12 ) + 1. For instance, if your phone number is 5339635384 please make a post for question 5 that describes your answer with additional info, image, video, links etc.

Best of Luck

**1. Configuration as Code (CaC) vs Infrastructure as Code (IaC)**

* **CaC**: Manages system settings and app configs (e.g., Ansible installs and configures Nginx).
* **IaC**: Provisions infrastructure (e.g., Terraform provisions AWS EC2 and VPC).

**2. Terraform vs Ansible vs CloudFormation**

| **Tool** | **Type** | **Purpose** | **Language** | **Multi-Cloud** | **State Mgmt** |
| --- | --- | --- | --- | --- | --- |
| Terraform | IaC | Infra provisioning | HCL | Yes | Yes |
| Ansible | CaC / light IaC | Config management | YAML | Yes | No |
| CloudFormation | IaC (AWS) | AWS Infra provisioning | JSON/YAML | No | Yes |

**3. Docker Compose**

* Define/manage multi-container apps in YAML.
* **Use Cases**:
  + Dev environments
  + Dependency orchestration (DB, backend, etc.)
  + Testing setups

**4. Docker Compose Networking**

* **Default**: user-defined bridge network
* **Communication**: via service name (backend:5000)

**5. Example docker-compose.yml (Backend + Frontend)**

version: '3.8'

services:

backend:

image: python:3.9

container\_name: backend

ports:

- "5000:5000"

command: python -m http.server 5000

networks:

- appnet

frontend:

image: nginx:latest

container\_name: frontend

depends\_on:

- backend

ports:

- "80:80"

networks:

- appnet

volumes:

- ./default.conf:/etc/nginx/conf.d/default.conf

networks:

appnet:

**6. CI vs CD**

| **Type** | **Full Form** | **Description** |
| --- | --- | --- |
| CI | Continuous Integration | Automatically test and integrate code |
| CD | Continuous Deployment | Automatically deploy tested code to staging/prod |

**7. Typical CI/CD Pipeline Stages**

1. Checkout code
2. Install dependencies
3. Static code checks (lint, security)
4. Run tests
5. Build artifacts
6. Deploy to staging
7. Manual approval
8. Deploy to production

**8. Conditional Pipeline Logic**

* **Tests**: On all PRs
* **Deploy**: Only on main branch merge

if: github.ref == 'refs/heads/main' && github.event\_name == 'push'

**9. Grafana Can't See Prometheus Metrics – Troubleshooting**

* Prometheus not added as Grafana source
* Wrong Prometheus URL (fix: http://prometheus:9090)
* Containers not in same network
* Wrong query time range
* Prometheus config missing scrape target

**10. Common DevOps Tools**

| **Tool** | **Purpose** |
| --- | --- |
| Jenkins | CI/CD automation |
| Docker | Containerization |
| Kubernetes | Container orchestration |
| Terraform | Infrastructure as Code |
| Ansible | Configuration management |
| Grafana | Visualization and monitoring |

**11. Branching Strategy**

* **Preferred**: GitHub Flow or Trunk-Based
* Small frequent merges, easier CI/CD
* Git Flow (alternative) for release-heavy teams

**12. 10 Git Commands**

| **Command** | **Description** |
| --- | --- |
| git init | Init a repo |
| git clone | Clone remote repo |
| git status | Show status of working directory |
| git add . | Stage all changes |
| git commit -m "msg" | Commit staged changes |
| git push origin | Push to remote branch |
| git pull | Pull from remote |
| git checkout -b | Create & switch to new branch |
| git merge | Merge branch into current |
| git log | Show commit history |