i2i Systems
innovation to integration

i2i Academy

Training Document

Topic	DevOps
Document Name	DEVOPS-EX-01

Document Difficulty Level			
Beginner	**Junior**	**Senior**	**Expert**
☐	■	☐	☐

Copyright of i2i Systems Turkey 2025

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

Configuration as Code (CaC) manages system/software configurations (e.g., installing packages).
Infrastructure as Code (IaC) provisions infrastructure (e.g., creating VMs).
Example:

- CaC: Ansible to install NGINX
- IaC: Terraform to create an AWS EC2 instance

2. What are the main differences between tools like Terraform, Ansible, and CloudFormation?

- **Terraform**: IaC tool, multi-cloud, declarative, creates infrastructure.
- **Ansible**: CaC/IaC tool, configures servers/apps, procedural, agentless.
- **CloudFormation**: AWS-only IaC tool, declarative, native to AWS.

3. What is Docker Compose and what are its main use cases?

Docker Compose is a tool that defines and runs multi-container Docker apps using a docker-compose.yml file.
Use cases: local dev environments, microservices, test automation, and sharing app setups.

4. What is the default network type in Docker Compose, and how do services communicate with each other by default?

The default network type in Docker Compose is a user-defined **bridge network**.
Services communicate by default using their **service names** as hostnames (e.g., backend, db).

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
```

Completed docker-compose.yml:

- **backend** uses python:3.9, runs on port 5000.
- **frontend** uses nginx:latest, proxies requests to backend:5000.
  They are connected on the default bridge network and use service discovery.

6. What is the difference between Continuous Integration (CI) and Continuous Deployment (CD)?

- **CI (Continuous Integration)**: Automatically builds and tests code on every change.
- **CD (Continuous Deployment)**: Automatically deploys tested code to production.

CI = Code quality
CD = Code delivery

7. What is a pipeline? In a typical CI/CD pipeline, what steps would you include to ensure code quality and safe deployment?

A **pipeline** automates steps from code to deployment.
Typical steps:

1. Checkout code
2. Install dependencies
3. Run tests
4. Code linting
5. Build app
6. Deploy to staging
7. Run integration tests

8. Deploy to production

8. How would you configure a pipeline to deploy only when code is merged to the main branch, but run tests on every pull request?

Configure the pipeline to:

- Run tests on **every pull request**
- Deploy only on **merge to main branch**
  Tools like GitHub Actions or GitLab CI support conditional logic using if: or only: rules.

9. 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.

Grafana not showing Prometheus metrics? Possible reasons:

- Prometheus not reachable from Grafana
- Wrong URL (e.g., localhost instead of http://prometheus:9090)
- Prometheus not running
- Port not exposed in Compose
  Fix: Set correct data source URL, ensure both are on the same Docker network, and expose required ports.

10. List 6 commonly used DevOps tools and briefly describe them.

6 Common DevOps Tools:

1. **Git** – version control
2. **Docker** – containerization
3. **Jenkins** – CI/CD automation
4. **Terraform** – infrastructure provisioning
5. **Ansible** – configuration management
6. **Kubernetes** – container orchestration

11. Regarding to Branching strategies in Version Control Systems, which would you prefer and why?

Preferred Branching Strategy: Trunk-Based Development
Why:

- Short-lived branches
- Easier merges
- Supports CI/CD
- Fewer conflicts
  Alternative: GitFlow (for release-heavy workflows)

12. Write 10 widely used git commands and describe them

10 Git Commands:

1. git init – Initialize a repo
2. git clone – Copy a remote repo
3. git status – Check working state
4. git add . – Stage all changes
5. git commit -m – Commit changes
6. git push – Upload to remote
7. git pull – Fetch + merge remote changes
8. git branch – List/create branches
9. git checkout -b – New branch
10. git merge – Merge branches

Solution of DEVOPS-EX-01:

Please answer all question.

Create a LinkedIn post for the answer of question MODULO[calculator](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