4.Cloud Native and 12-Factor App

Definition:

- **Cloud Native** refers to designing, building, and operating applications to fully exploit the advantages of the cloud computing model.
- A **12-Factor App** is a methodology for building software-as-a-service apps that are scalable and maintainable.

Cloud-Ready vs. Cloud-Native:

- **Cloud-Ready**: Applications that are optimized to run on cloud infrastructure but still rely on traditional architectures.
- **Cloud-Native**: Designed to fully utilize cloud services, focused on microservices, and leverages the cloud for scalability, availability, and resiliency.

The 12-Factor App Principles:

1. **Codebase**:
   - A single codebase is tracked in version control, with multiple deployments (e.g., dev, staging, production).
2. **Dependencies**:
   - Explicitly declare and isolate dependencies in your project. Avoid relying on system-wide packages.
3. **Config**:
   - Store configuration in environment variables.
4. **Backing Services**:
   - Treat backing services (e.g., databases, message queues) as attached resources. These should be externalized.
5. **Build, Release, Run**:
   - Separate the build and run stages. Builds should be immutable and reusable across different environments.

6. **Processes**:
   ➢ Execute the app as one or more stateless processes. Any required persistent storage should be externalized.
7. **Port Binding**:
   ➢ Export services via port binding. The app should be self-contained.
8. **Concurrency**:
   ➢ Scale out via process model, splitting workloads into independent processes.
9. **Disposability**:
   ➢ Maximize robustness with fast startup and graceful shutdown to allow quick restarts and resilience.
10. **Dev/Prod Parity**:
    ➢ Keep development, staging, and production as similar as possible to reduce environmental differences.
11. **Logs**:
    ➢ Treat logs as event streams. Output them to standard output, and use a tool to aggregate and analyze them.
12. **Admin Processes**:
    ➢ Run admin/management tasks as one-off processes (e.g., database migrations).

Scaling: Horizontal vs Vertical