The "Twelve-Factor App" Methodology

The "Twelve-Factor App" is a set of methodologies for building Software-as-a-Service (SaaS) applications that:

- Use declarative formats for setup automation, minimizing time and cost for new developers.
- Have a clean contract with the underlying operating system, offering maximum portability between execution environments.
- Are suitable for deployment on modern cloud platforms, eliminating the need for servers and system administration.
- Minimize discrepancies between development and production environments, enabling continuous deployment.
- Can scale without significant changes to tooling, architecture, or development practices.

1. Codebase

- **Essence:** One codebase, tracked in a version control system (e.g., Git), for each deployable application.
- **Description:** Multiple deployments (dev, staging, prod) can share the same codebase, but each application should have only one repository.

2. Dependencies

- **Essence:** Explicitly declare and isolate all dependencies.
- **Description:** The application should explicitly declare all its dependencies (e.g., libraries, frameworks) using package management systems (npm, pip, Maven) and use dependency isolation (virtual environments, containers) to ensure consistency.

3. Config

- **Essence:** Store configuration in environment variables.
- **Description:** Configuration (data that changes between deployments: database credentials, API keys) must be strictly separated from the code and stored in environment variables, not in code or configuration files that are part of the codebase.

4. Backing Services

- **Essence:** Treat backing services as attached resources.
- **Description:** Any services consumed by the application over the network (databases, message queues, caches, SMTP services) should be treated as "attached" resources that can be easily swapped out without changing the application's code.

5. Build, Release, Run

- **Essence:** Strictly separate build, release, and run stages.
- **Description:**
    - **Build:** Transforms the codebase into an executable bundle (artifact).
    - **Release:** Combines the build artifact with configuration for a specific environment.
    - **Run:** Launches one or more application processes in the execution environment. Each release must be immutable.

6. Processes

- **Essence:** Execute the application as one or more stateless processes.
- **Description:** The application should be entirely stateless (no state saved to the process's disk). Any data requiring persistence must be stored in backing services (e.g., a database).

7. Port Binding

- **Essence:** Export services via port binding.
- **Description:** The application should be entirely self-contained and export its services via a port bound to an IP address. This allows other applications or load balancers to access it.

8. Concurrency

- **Essence:** Scale out using a process model.
- **Description:** Scaling the application should be achieved by running more processes (instances), rather than by adding more threads within a single process.

9. Disposability

- **Essence:** Maximize robustness with fast startup and graceful shutdown.
- **Description:** Application processes should be easily startable and quickly stoppable, which facilitates rapid scaling and deployment. They should be resilient to sudden termination.

10. Dev/Prod Parity

- **Essence:** Keep development, staging, and production environments as similar as possible.
- **Description:** Strive to make the development environment as similar as possible to the production environment (identical OS versions, runtimes, dependencies, backing services).

11. Logs

- **Essence:** Treat logs as event streams.
- **Description:** The application should write its logs to standard output (stdout) and standard error (stderr). Orchestration systems or centralized logging solutions should collect, aggregate, and store these streams.

12. Admin Processes

- **Essence:** Run admin/management tasks as one-off processes.
- **Description:** One-off tasks (database migrations, script execution, console launches) should be run in the application's environment using the same codebase and configuration as the main application processes.