Assignment 1: Design a 12-factor app for an e-commerce website.

# Description:

As an architect, you are tasked with designing a 12-factor app for an ecommerce website. The 12-factor app methodology is a set of principles that can be applied to develop modern, scalable, and maintainable web applications. Your task is to apply these principles to the ecommerce website and design a system that is easily deployable, scalable, and maintainable.

## Tasks:

1. Break the ecommerce website into smaller, independent components: The first step is to break the ecommerce website into smaller, independent components. Each component should have a specific responsibility, and they should be loosely coupled. This makes it easier to maintain and scale the application.
2. Use declarative formats for configuration: All configuration should be stored in declarative formats. This makes it easier to manage the configuration and ensures that the application behaves consistently across different environments.
3. Store application state in a stateless manner: Application state should be stored in a stateless manner. This makes it easier to scale the application horizontally and ensures that the application can recover quickly from failures.
4. Use backing services: All backing services should be treated as attached resources. This includes databases, message queues, and caching systems. This makes it easier to manage these resources and ensures that they can be easily swapped out or scaled independently.
5. Use a build, release, run process: All code changes should go through a standardized build, release, run process. This ensures that changes are thoroughly tested before they are deployed to production.
6. Strictly separate build and run stages: The build and run stages should be strictly separated. This makes it easier to manage the application in different environments and ensures that the same code is running in every environment.
7. Execute the application as one or more stateless processes: The application should be executed as one or more stateless processes. This makes it easier to scale the application horizontally and ensures that the application can recover quickly from failures.
8. Export services via port binding: All services should be exported via port binding. This makes it easier to manage the application and ensures that the application can be easily deployed to different environments.
9. Scale out via the process model: The application should be scaled out via the process model. This makes it easier to scale the application horizontally and ensures that the application can recover quickly from failures.
10. Maximize robustness with fast startup and graceful shutdown: The application should start up quickly and shut down gracefully. This makes it easier to manage the application and ensures that the application can recover quickly from failures.
11. Keep development, staging, and production as similar as possible: The development, staging, and production environments should be kept as similar as possible. This makes it easier to manage the application in different environments and ensures that the same code is running in every environment.
12. Treat logs as event streams: All logs should be treated as event streams. This makes it easier to manage the application and ensures that the logs can be easily searched and analyzed.

## Deliverables:

1. A document describing how the 12-factor app methodology will be applied to the ecommerce website.
2. A system architecture diagram showing how the ecommerce website will be broken into smaller, independent components.
3. A list of all backing services that will be used and how they will be managed.
4. A description of the build, release, run process that will be used to deploy the application.
5. A plan for scaling the application horizontally.
6. A plan for managing the application in different environments.
7. A plan for managing logs and using them for monitoring and debugging.
8. A plan for testing the application to ensure that it meets the

# Solution:

The 12 Factor App is a set of principles that provide a methodology for building modern, scalable, and maintainable applications. Let's see how we can apply these principles to an ecommerce website.

* Codebase: One codebase tracked in version control, many deploys
  + The code for the ecommerce website should be stored in a single codebase, such as Git or SVN, and version controlled.
  + The codebase should be deployable to multiple environments, such as development, staging, and production, with minimal configuration changes.
* Dependencies: Explicitly declare and isolate dependencies
  + The dependencies for the ecommerce website should be declared explicitly, preferably using a package manager like npm or Maven.
  + The dependencies should be isolated from the rest of the system, to avoid conflicts and ensure consistency.
* Config: Store config in the environment
  + Configuration values for the ecommerce website should be stored in the environment, preferably using environment variables.
  + This makes it easy to change configuration values without having to modify the code.
* Backing services: Treat backing services as attached resources
  + Backing services, such as databases or message queues, should be treated as attached resources.
  + They should be accessed through a URL or other endpoint, rather than being hardcoded into the application code.
* Build, release, run: Strictly separate build and run stages
  + The build, release, and run stages for the ecommerce website should be strictly separated.
  + This ensures that the code is built and tested in a consistent environment before it is released to production.
* Processes: Execute the app as one or more stateless processes
  + The ecommerce website should be designed to run as one or more stateless processes.
  + This allows the website to scale horizontally and handle traffic spikes more effectively.
* Port binding: Export services via port binding
  + The ecommerce website should export its services via port binding, using a standard port like 80 or 443.
  + This makes it easy to access the website from a web browser or API client.
* Concurrency: Scale out via the process model
  + The ecommerce website should be designed to scale out via the process model, rather than scaling up by adding more resources to a single server.
  + This allows the website to handle more traffic and provide a better user experience.
* Disposability: Maximize robustness with fast startup and graceful shutdown
  + The ecommerce website should be designed to start up quickly and shut down gracefully.
  + This ensures that the website can be restarted quickly in the event of a failure or update.
* Dev/prod parity: Keep development, staging, and production as similar as possible
  + The development, staging, and production environments for the ecommerce website should be kept as similar as possible.
  + This ensures that any issues that are discovered in one environment are more likely to be discovered in the others as well.
* Logs: Treat logs as event streams
  + The ecommerce website should treat logs as event streams, which can be easily searched and analyzed.
  + This makes it easier to troubleshoot issues and identify areas for improvement.
* Admin processes: Run admin/management tasks as one-off processes
  + Admin and management tasks for the ecommerce website should be run as one-off processes, rather than being part of the main application.
  + This ensures that the website remains focused on its primary function, and reduces the risk of issues or errors in management tasks affecting the main application.

As an assignment, an architect could be asked to review an existing ecommerce website and identify areas where the 12 Factor App principles could be applied. They could also be asked to design a new ecommerce website that follows these principles from the ground up, and ensures that each component of the application conforms to these principles. Here are some specific steps an architect can take to design a 12 Factor App for an ecommerce website:

1. Codebase: Keep the codebase in version control and have one codebase per application.
2. Dependencies: Explicitly declare and isolate dependencies.
3. Config: Store configuration in the environment and use it to configure the application.
4. Backing Services: Treat backing services, such as databases and message brokers, as attached resources and access them through URLs.
5. Build, Release, Run: Separate build, release, and run stages to enable continuous deployment.
6. Processes: Execute the application as one or more stateless processes.
7. Port Binding: Export services via port binding and use a reverse proxy to handle requests.
8. Concurrency: Scale out the application horizontally.
9. Disposability: Maximize robustness with fast startup and graceful shutdown.
10. Dev/Prod Parity: Keep development, staging, and production as similar as possible.
11. Logs: Treat logs as event streams and aggregate them in a central location.
12. Admin Processes: Run administrative tasks as one-off processes.

By following these steps, an architect can design an ecommerce website that is scalable, portable, and resilient, and that can be easily deployed and managed in a cloud environment.