Portability: Able to run the same app on different environments without having to change the source code of the application

Scalability: take down your application to add additional resources to your servers.

Modern applications need to be portable, and such as must not be tightly coupled with underlying infrastructure. They should have minimal divergence when deployed between dev test and problems to enable continuous deployment and easily scalable by spinning.

**Twelve-Factor App**

Twelve-Factor App is a methodology and set of principles designed to guide the development of modern, cloud-native applications. Its purpose is to create applications that are robust, scalable, and easy to maintain in cloud environments

**I.CODE BASE:**

☺ First rule – Having a single code base for your application.

A codebase is any single repo (in a centralized revision control system like Subversion), or any set of repos who share a root commit (in a decentralized revision control system like Git).

A codebase is a collection of source code files that work together to create a software application. The way we manage and track changes to this codebase is through a version control system.

In a centralized version control system like Subversion, a codebase is usually stored in a single repository (repo). This means that all the code for a project is stored in one place, and changes are tracked there.

In a decentralized version control system like Git, a codebase can be spread across multiple repositories, but they all share a common starting point known as the "root commit." This means that there can be different copies of the codebase, but they all originate from the same initial version.

If there are, multiple codebases, it is not an app – it is a distributed system. Each component in a distributed system is an app, and each can individually comply with twelve-factor.

Multiple apps sharing the same code is a violation of twelve-factor. The solution here is to factor shared code into libraries, which can be included through the dependency manager.

**II.DEPENDENCIES:**

☺ Explicitly declare and isolate dependencies

Most programming languages offer a packaging system for distributing support libraries, such as [CPAN](http://www.cpan.org/) for Perl or [Rubygems](http://rubygems.org/) for Ruby. Libraries installed through a packaging system can be installed system-wide (known as “site packages”) or scoped into the directory containing the app (known as “vendoring” or “bundling”).

A twelve-factor app never relies on implicit existence of system-wide packages. It declares all dependencies, completely and exactly, via a dependency declaration manifest. Furthermore, it uses a dependency isolation tool during execution to ensure that no implicit dependencies “leak in” from the surrounding system. The full and explicit dependency specification is applied uniformly to both production and development.

One benefit of explicit dependency declaration is that it simplifies setup for developers new to the app. The new developer can check out the app’s codebase onto their development machine, requiring only the language runtime and dependency manager installed as prerequisites. They will be able to set up everything needed to run the app’s code with a deterministic build command.

**VIII.CONCURRENCY:**

☺ Application should be scale out horizontally and not vertically by running multiple instances of the application concurrently.

**VI.PROCESSES:**

☺ Twelve-factor processes are stateless and share-nothing. Any data that needs to persist must be stored in a stateful backing service, typically a database.

Twelve-Factor App methodology encourages you to think of your application as a collection of independent and stateless components that can be easily scaled up or down. Each part should start quickly, stop without causing problems, and be able to handle many things happening at once. Using a good process management system and managing logs as event streams helps keep everything under control and makes your application more robust.

The user may be considered logged out as the session information is not available there. Now there are load balancers that are session of air and can redirect users to the same process each time. Moreover, this is called sticky sessions. If the server crashes locally, stored data will be lost then.

Sticky sessions are violation for 12-factor

For this, we must bot store anything in these processes instead. It should all be stored in external

Run as many instances as required while ensuring we store nothing locally and enabling all instances to point to the same.

**IV.BACKING SERVICES:**

☺ Treat backing services as attached resources. A backing service is any service the app consumes over the network as part of its normal operation.

Resources can be attached to and detached from deploys at will. For example, if the app’s database is misbehaving due to a hardware issue, the app’s administrator might spin up a new database server restored from a recent backup. The current production database could be detached, and the new database attached – all without any code changes.

**III.CONFIG:**

☺Configuration is the only change for an application between different deployment environments.

Config means configuration information such as environment specific settings should be stored in environment variables (dot ENV) rather than hard coded in the applications code.

**V.BUILD, RELEASE AND RUN:**

☺The twelve-factor app uses strict separation between the build, release and run stages.

**VII.PORT BINDING:**

☺ The twelve-factor app is completely self-contained

Note also that the port-binding approach means that one app can become the backing service for another app, by providing the URL to the backing app as a resource handle in the config for the consuming app.

**IX. Disposability:**

☺ The twelve-factor app’s processes are disposable, meaning they can be started or stopped at a moment’s notice.

**X.DEV/PROD PARITY:**

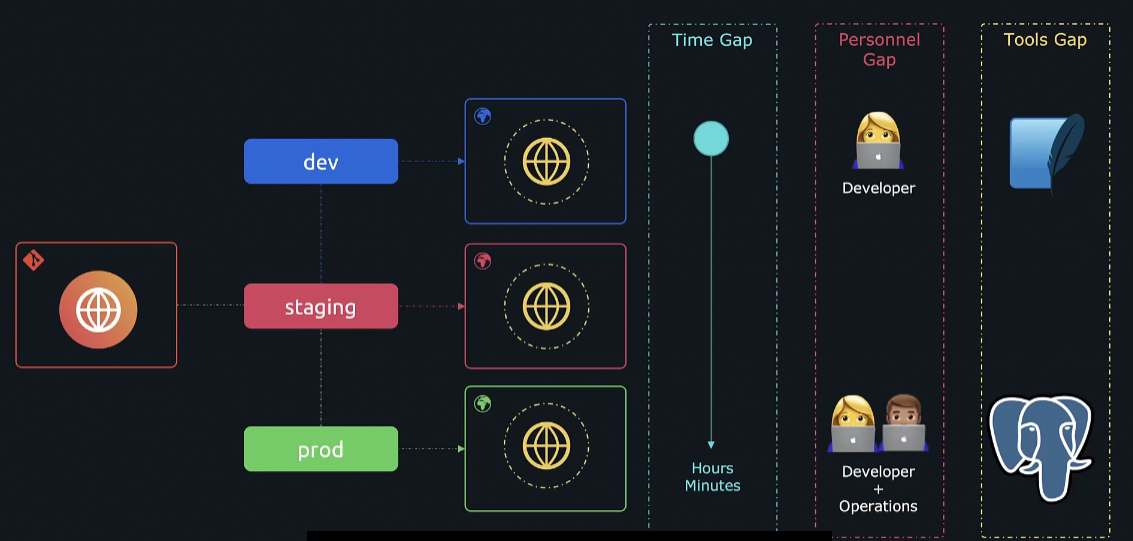
Historically, there have been substantial gaps between development (a developer making live edits to a local deploy of the app) and production (a running deploy of the app accessed by end users). These gaps manifest in three areas:

* The time gap: A developer may work on code that takes days, weeks, or even months to go into production.
* The personnel gap: Developers write code, ops engineers deploy it.
* The tools gap: Developers may be using a stack like Nginx, SQLite, and OS X, while the production deploy uses Apache, MySQL, and Linux.

☺ The twelve-factor app is designed for continuous deployment by keeping the gap between development and production small.

Looking at the three gaps described above:

* Make the time gap small: a developer may write code and have it deployed hours or even just minutes later.
* Make the personnel gap small: developers who wrote code are closely involved in deploying it and watching its behaviour in production.
* Make the tools gap small: keep development and production as similar as possible.



**XI.LOGS:**

☺ Logs are essential for monitoring and debugging issues of the deployed application.

**XII.ADMIN PROCESS:**

☺ Run admin/management tasks as one-off processess

The process formation is the array of processes that are used to do the app’s regular business (such as handling web requests) as it runs. Separately, developers will often wish to do one-off administrative or maintenance tasks for the app.