




12 Factor Applications

Desarrollo de aplicaciones Cliente-Servidor



The Twelve Factors app

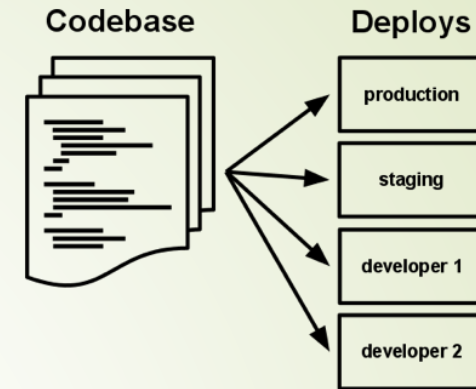
- 
1. **Codebase:** One codebase tracked in revision control, many deploys
 2. **Dependencies:** Explicitly declare and isolate dependencies
 3. **Config:** Store config in the environment
 4. **Backing services:** Treat backing services as attached resources
 5. **Build, release, run:** Strictly separate build and run stages
 6. **Processes:** Execute the app as one or more stateless processes



The Twelve Factors app (cont)

- 7. Port binding:** Export services via port binding
- 8. Concurrency:** Scale out via the process model
- 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
- 12. Admin processes:** Run admin/management tasks as one-off processes

Codebase



- A codebase is a single repo
- There is always a one-to-one correlation between the codebase and the app:
 - If there are multiple codebases, it's not an app – it's 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.



Dependencies



- Never relies on implicit existence of system-wide packages
- Declare all dependencies, completely and exactly, via a dependency declaration manifest.
- Use 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.

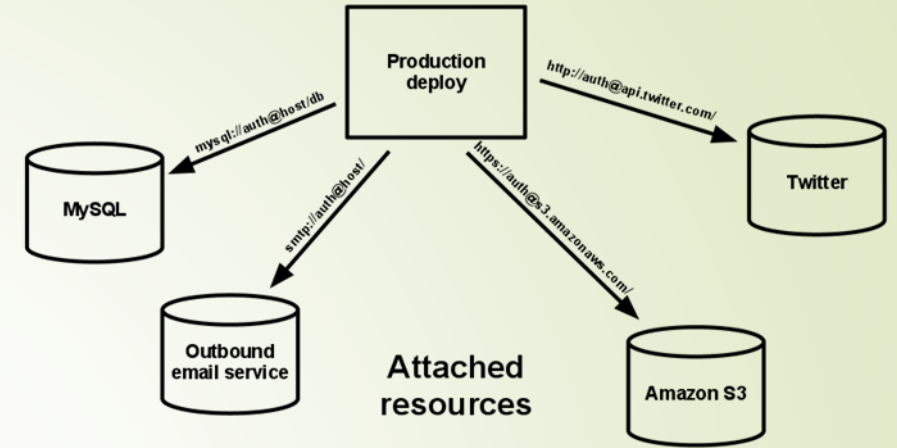


Config



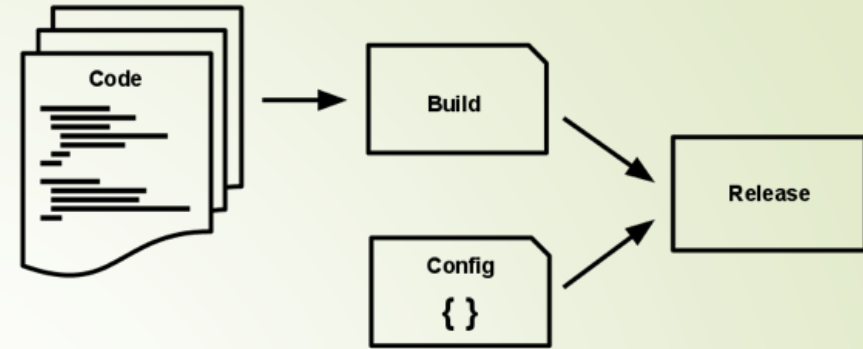
- An app's config is everything that is likely to vary between deploys
 - Resource handles to the database, Memcached, and other backing services
 - Credentials to external services such as Amazon S3 or Twitter
 - Per-deploy values such as the canonical hostname for the deploy
- Note that this definition of “config” does **not** include internal application config
- The twelve-factor app stores config in **environment variables**
- In a twelve-factor app, env vars are granular controls, each fully orthogonal to other env vars.

Backing services



- A *backing service* is any service the app consumes over the network as part of its normal operation.
- The code for a twelve-factor app makes no distinction between local and third party services.
- Each distinct backing service is a *resource*.

Build, release, run



- A codebase is transformed into a (non-development) deploy through three stages:
 - The *build stage* is a transform which converts a code repo into an executable bundle known as a build. Using a version of the code at a commit specified by the deployment process, the build stage fetches vendors dependencies and compiles binaries and assets.
 - The *release stage* takes the build produced by the build stage and combines it with the deploy's current config. The resulting release contains both the build and the config and is ready for immediate execution in the execution environment.
 - The *run stage* (also known as “runtime”) runs the app in the execution environment, by launching some set of the app's processes against a selected release.



Processes



- Processes are stateless and share-nothing. Any data that needs to persist must be stored in a stateful backing service, typically a database.
- Never assume that anything cached in memory or on disk will be available on a future request or job.
- A twelve-factor app prefers to do this compiling during the ***build stage***.
- Sticky sessions are a violation of twelve-factor and should never be used or relied upon.

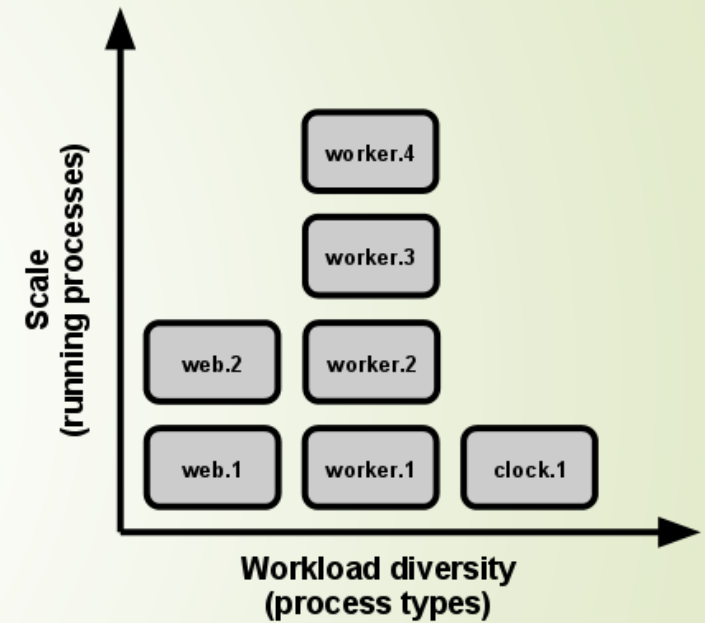


Port binding

- Web apps are sometimes executed inside a webserver container.
- **The twelve-factor app is completely self-contained** and does not rely on runtime injection of a webserver into the execution environment to create a web-facing service.

Concurrency

- Processes are a first class citizen. Processes in the twelve-factor app take strong cues from the unix process model for running service daemons.
- The share-nothing, horizontally partitionable nature of twelve-factor app processes means that adding more concurrency is a simple and reliable operation.
- Twelve-factor app processes should never daemonize or write PID files. Instead, rely on the operating system's process manager (such as systemd, a distributed process manager on a cloud platform, or a tool like Foreman in development) to manage output streams, respond to crashed processes, and handle user-initiated restarts and shutdowns.





Disposability



- app's processes are disposable, meaning they can be started or stopped at a moment's notice. This facilitates fast elastic scaling, rapid deployment of code or config changes, and robustness of production deploys.
- Processes should strive to **minimize startup time**. Ideally, a process takes a few seconds from the time the launch command is executed until the process is up and ready to receive requests or jobs.
- Processes should also be **robust against sudden death**, in the case of a failure in the underlying hardware.



Dev/prod parity

- Keep development, staging, and production as similar as possible
- **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 behavior in production.
 - Make the tools gap small: keep development and production as similar as possible.



Logs



- Logs provide visibility into the behavior of a running app. In server-based environments they are commonly written to a file on disk (a “logfile”); but this is only an output format.
- Logs are the stream of aggregated, time-ordered events collected from the output streams of all running processes and backing services.
- A twelve-factor app never concerns itself with routing or storage of its output stream. It should not attempt to write to or manage logfiles. Instead, each running process writes its event stream, unbuffered, to stdout.



Admin processes



- 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, such as:
 - Running database migrations
 - Running a console (also known as a REPL shell) to run arbitrary code or inspect the app's models against the live database.
 - Running one-time scripts committed into the app's repo.
- Twelve-factor strongly favors languages which provide a REPL shell out of the box, and which make it easy to run one-off scripts.