LIGHTNING TALKS

ALICAN AKKUS@IYZICO

THE 12 FACTORS

1 Codebase
One codebase tracked in revision control, many deploys.

2 Dependencies
Explicitly declare and isolate dependencies.

3 Configuration
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.

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.

1

Codebase

One codebase tracked in revision control, many deploys.
This means you must not have various codebase for various versions. Branches and tags is okay, different repos are not.

2

Dependencies

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.

3

Configuration

Store config in the environment. The most important rule here is – never commit your environment-specific configuration (most importantly: password) in the source code repo. Otherwise your production system may be vulnerable.

4

Backing Services

Treat backing services as attached resources. A backing service is one that requires a network connection to run. This is a very popular paradigm found in modern application development, especially prevalent with the rise in popularity of microservice architecture. The 12-Factor App methodology advises developers to treat these services agnostically, meaning changes or modifications should occur without having to make any code changes.

5

Build, release, run

Strictly separate build and run stages. **The twelve-factor app uses strict separation between the build, release, and run stages.** For example, it is impossible to make changes to the code at runtime, since there is no way to propagate those changes back to the build stage.

6

Processes

Execute the app as one or more stateless 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.

7

Port binding

Export services via port binding. **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. The web app **exports HTTP as a service by binding to a port**, and listening to requests coming in on that port.

8

Concurrency

Scale out via the process model. **In the twelve-factor app, processes are a first class citizen.** Processes in the twelve-factor app take strong cues from the unix process model for running service daemons. Using this model, the developer can architect their app to handle diverse workloads by assigning each type of work to a *process type*.

9

Disposability

Maximize robustness with fast startup and graceful shutdown.

The twelve-factor 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.

10

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.

11

Logs

Treat logs as event streams. **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. During local development, the developer will view this stream in the foreground of their terminal to observe the app's behavior.

12

Admin processes

Run admin/management tasks as one-off 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.

THANKS

https://12factor.net
https://dzone.com/articles/the-12-factor-app-a-java-developers-perspective
https://www.sitepoint.com/12-factor-apps-methodology-implement-apps-appfog/