12 Factor App

Best Practices for Scala Deployment

2005	2015
WAR files	JAR files
App Servers	Microservices
Hot-Deploy	Continuous Deploy
Java	Scala

Joe Kutner

@codefinger

JVM Platform Owner
@Heroku

12 Factor App

a methodology

Scalability

Maintainability

Portability

- Immutable
- Ephemeral
- Declarative
- Automated

sbt-native-packager 1.0.0-RC2 Site Page Installation » Source

SBT Native Packager Plugin

This sbt plugin provides you with everything you need to package your application. N
want to build a simple standalone application or a server application. The JVM let's yo
anywhere. SBT Native Packager let's you deploy everywhere!

Getting Started »

https://github.com/sbt/sbt-native-packager

This repository Search Explore Gist Blog Help

sbt / sbt-native-packager Watch 44

900 commits 14 branches 45 releases 71 contributors

branch: master sbt-native-packager / +

Merge pull request #541 from dwhjames/fix/systemv-start-rpm2

muuki88 authored 11 minutes ago latest commit a243197d8b

project	FIX #82 adding the ability to specify jvm options via sbt	10 days ago
src	fix: templating for /etc/sysconfig/ path	21 hours ago
test-project-docker	Adding documentation, examples and tests.	2 months ago
test-project-jdkpackager	Fixed corrupted Windows Icon and commented maintainer setting in the ...	2 months ago

project/plugins.sbt

```
addSbtPlugin(
  "com.typesafe.sbt" % "sbt-native-packager" % "0.7.6"
)
```

```
$ sbt stage
```

without further ado…

The 12 Factors

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Dev/prod parity
- Logs
- Admin processes

Thank You!

Goodbye!

(just kidding)

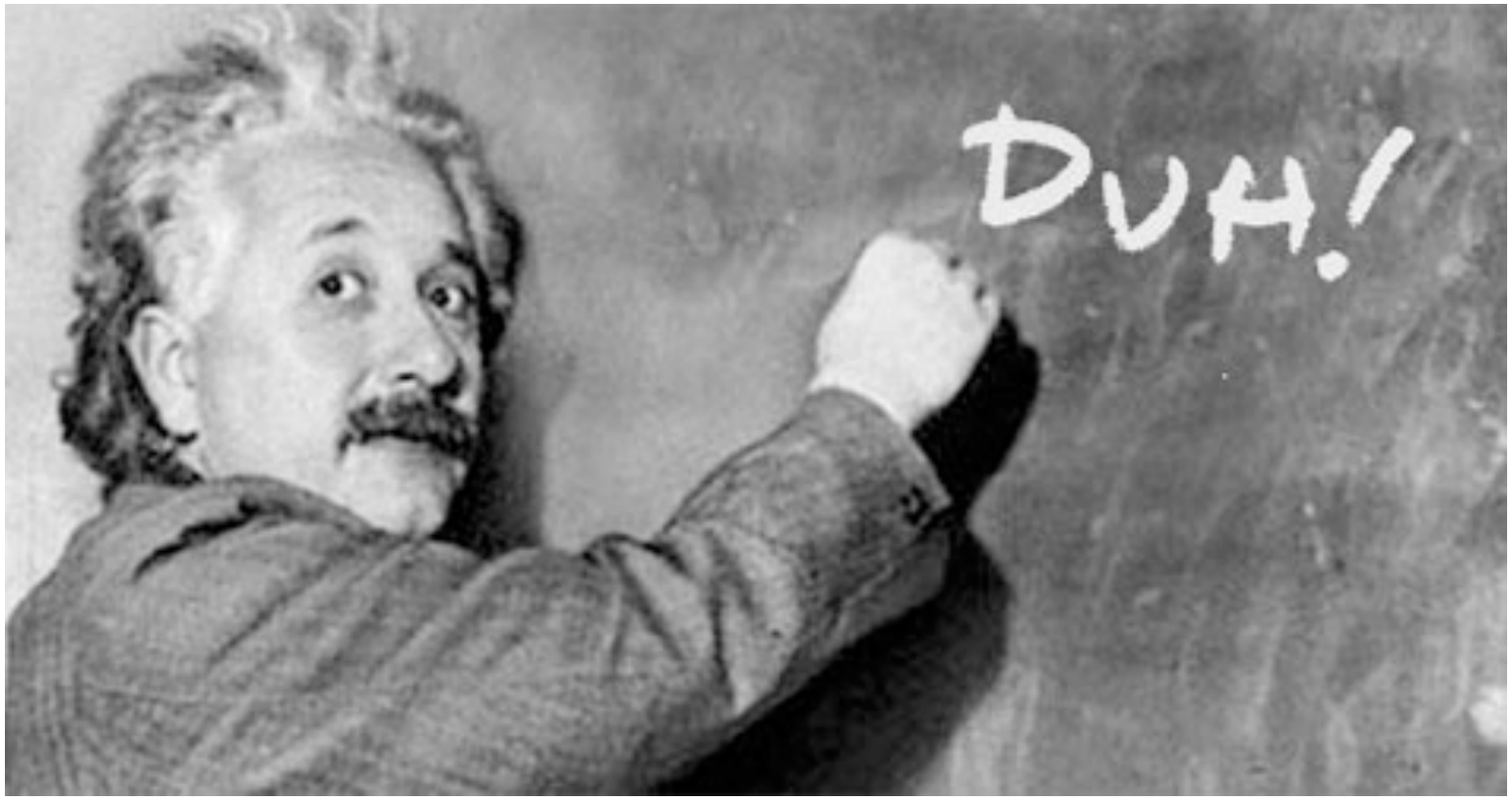
The 12 Factors

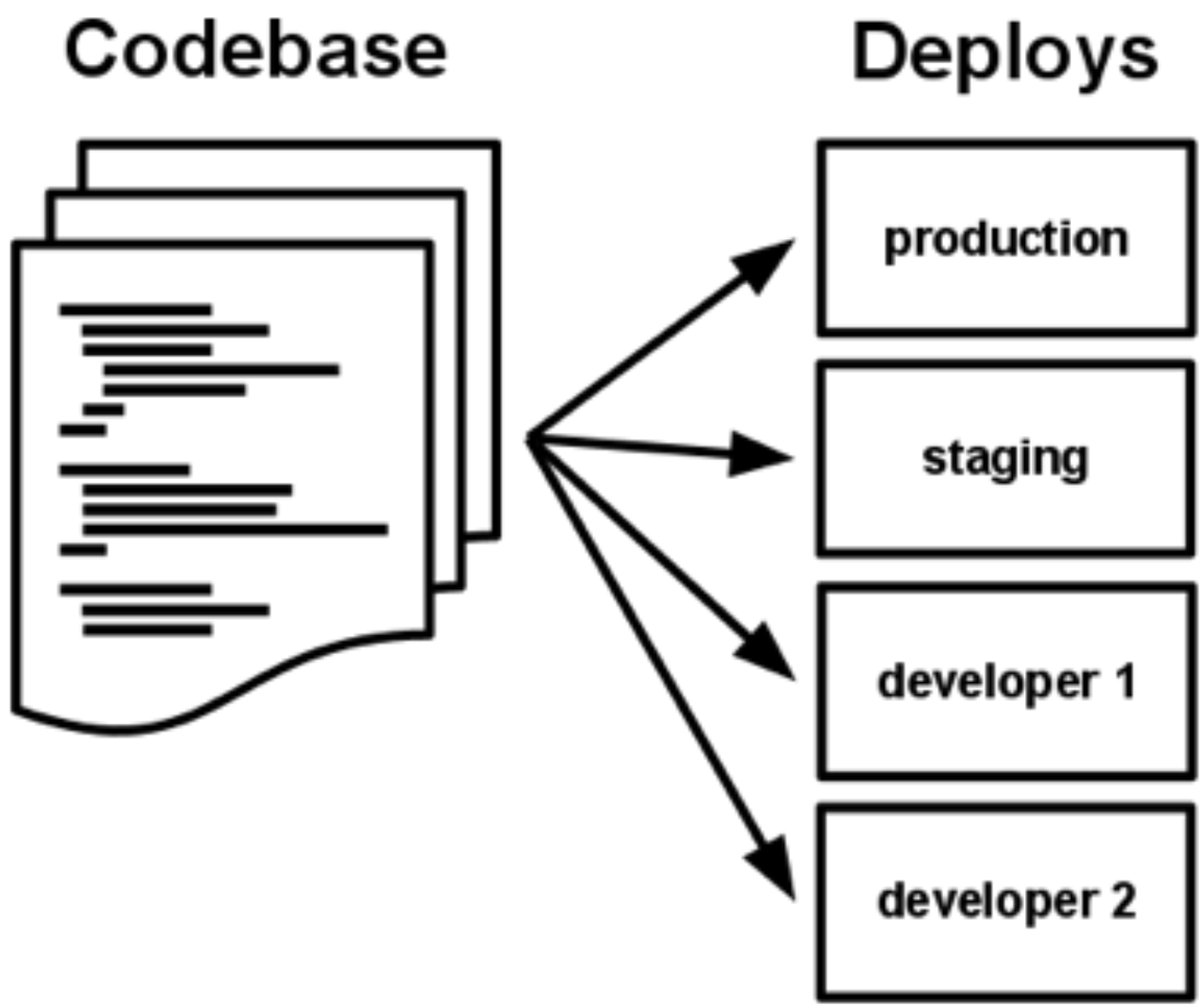
- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Dev/prod parity
- Logs
- Admin processes

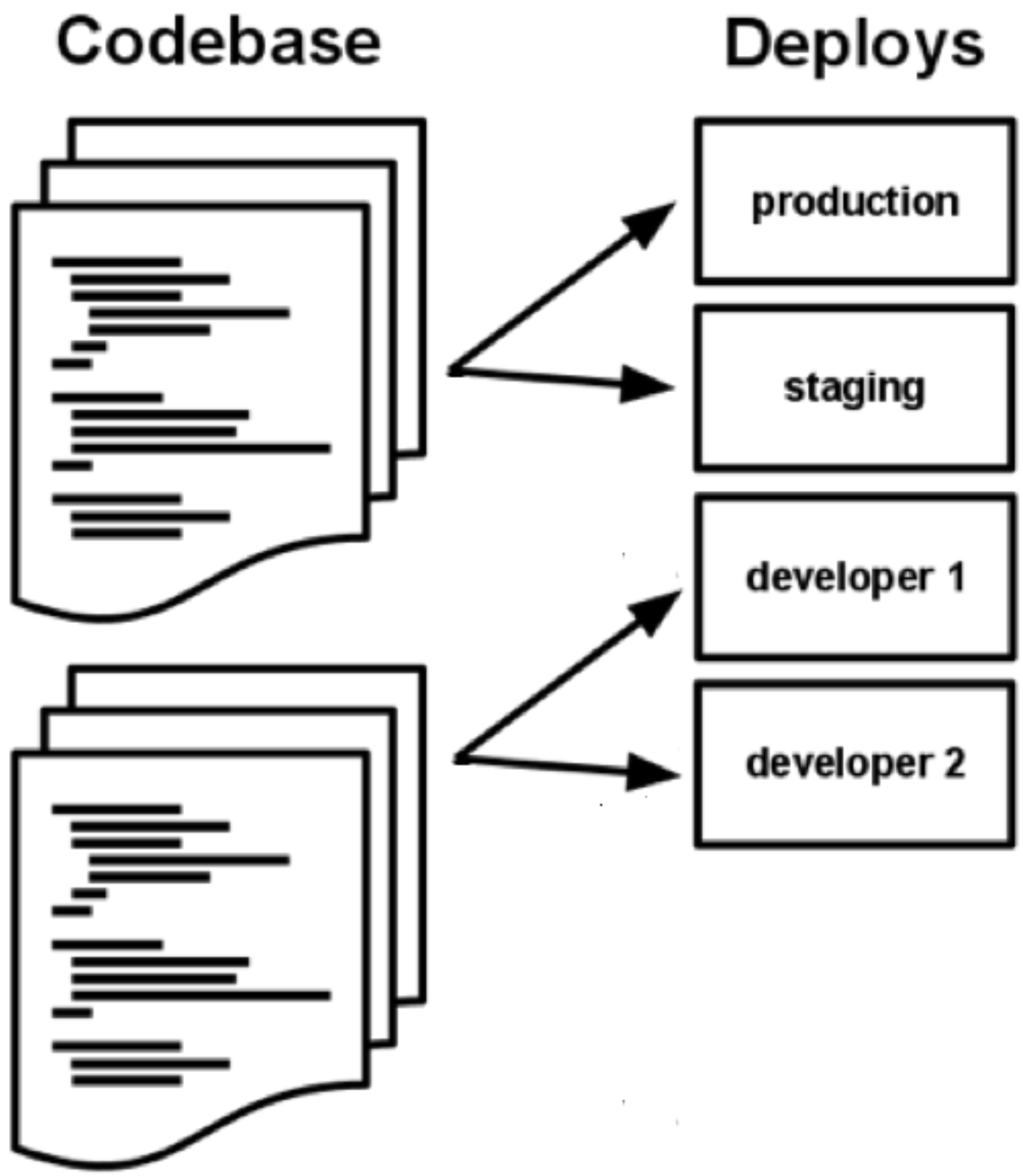
The 12 Factors

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Dev/prod parity
- Logs
- Admin processes

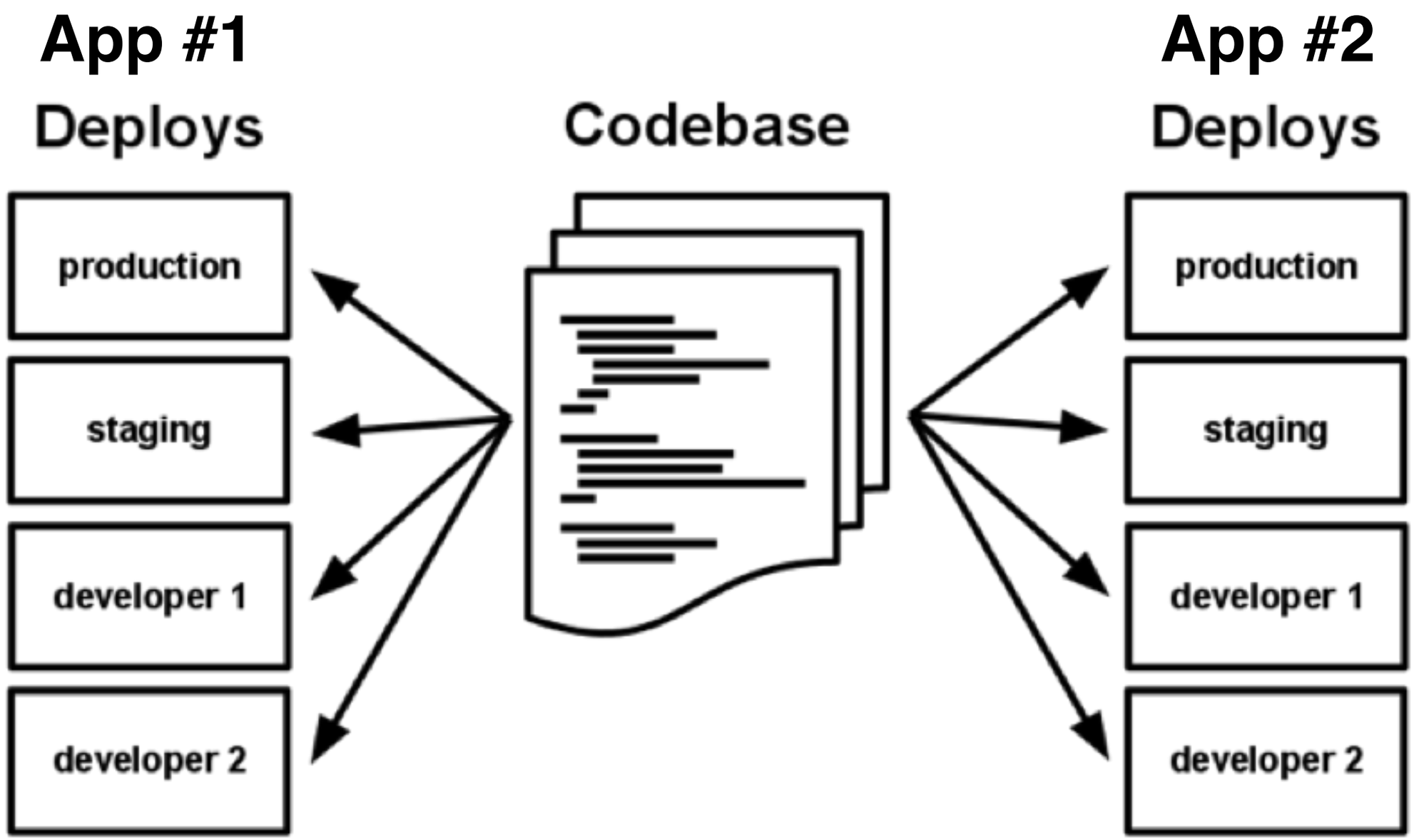
Use Version Control


DUH!


Codebase
Deploys
production
staging
developer 1
developer 2


Codebase
Deploys
production
staging
developer 1
developer 2

BAD


App #1
Deploys
production
staging
developer 1
developer 2
Codebase
App #2
Deploys
production
staging
developer 1
developer 2

BAD

```
my-project
 └ build.sbt
 └ app
 └ conf
 └ public
 └ my-library
  └ build.sbt
  └ src
   └ main
    └ scala
```

```
my-project
 └ build.sbt
 └ app
 └ conf
 └ public
 └ my-library
  └ my-sub-project
   └ build.sbt
   └ src
    └ main
     └ scala
```

BAD

Submodules

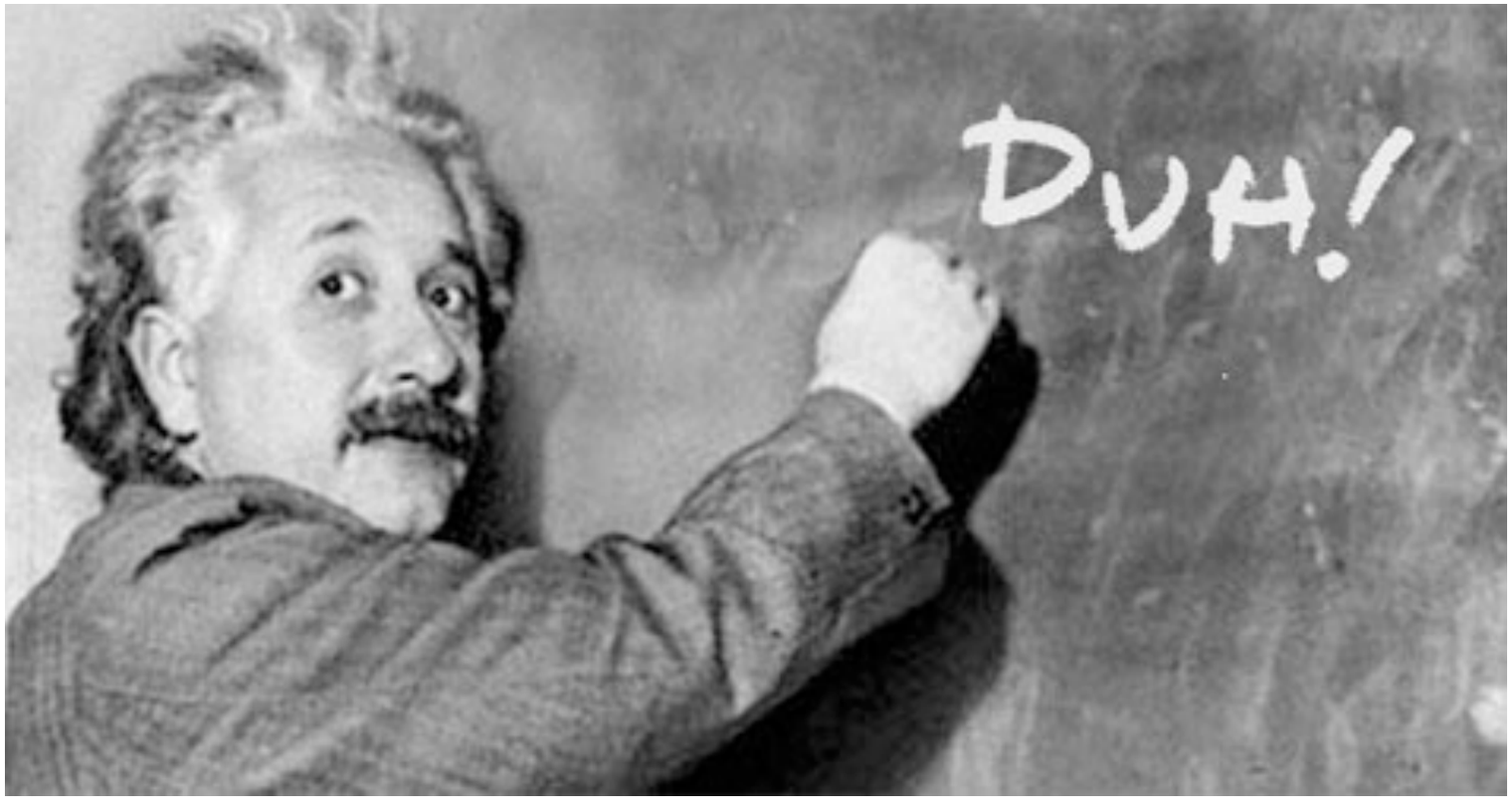
app	first commit
conf	first commit
logs	first commit
play-sub-project @ b247f72	Added a subproject
project	first commit
public	first commit
test	first commit
.buildpacks	first commit
.gitignore	first commit
.gitmodules	Added a subproject
LICENSE	first commit

```
$ git submodule add https://github.com/jkutner/play-sub-project
```

The 12 Factors

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Dev/prod parity
- Logs
- Admin processes

Don’t check JAR files into Git


DUH!

Explicitly declare and isolate dependencies

Never rely on implicit existence of system-wide packages

Dependencies

global

BAD

`~/.m2`

`~/.ivy2`

(ok in dev)

local

GOOD

`target/`

Vendoring

GOOD

jkutner — bash — 128×40

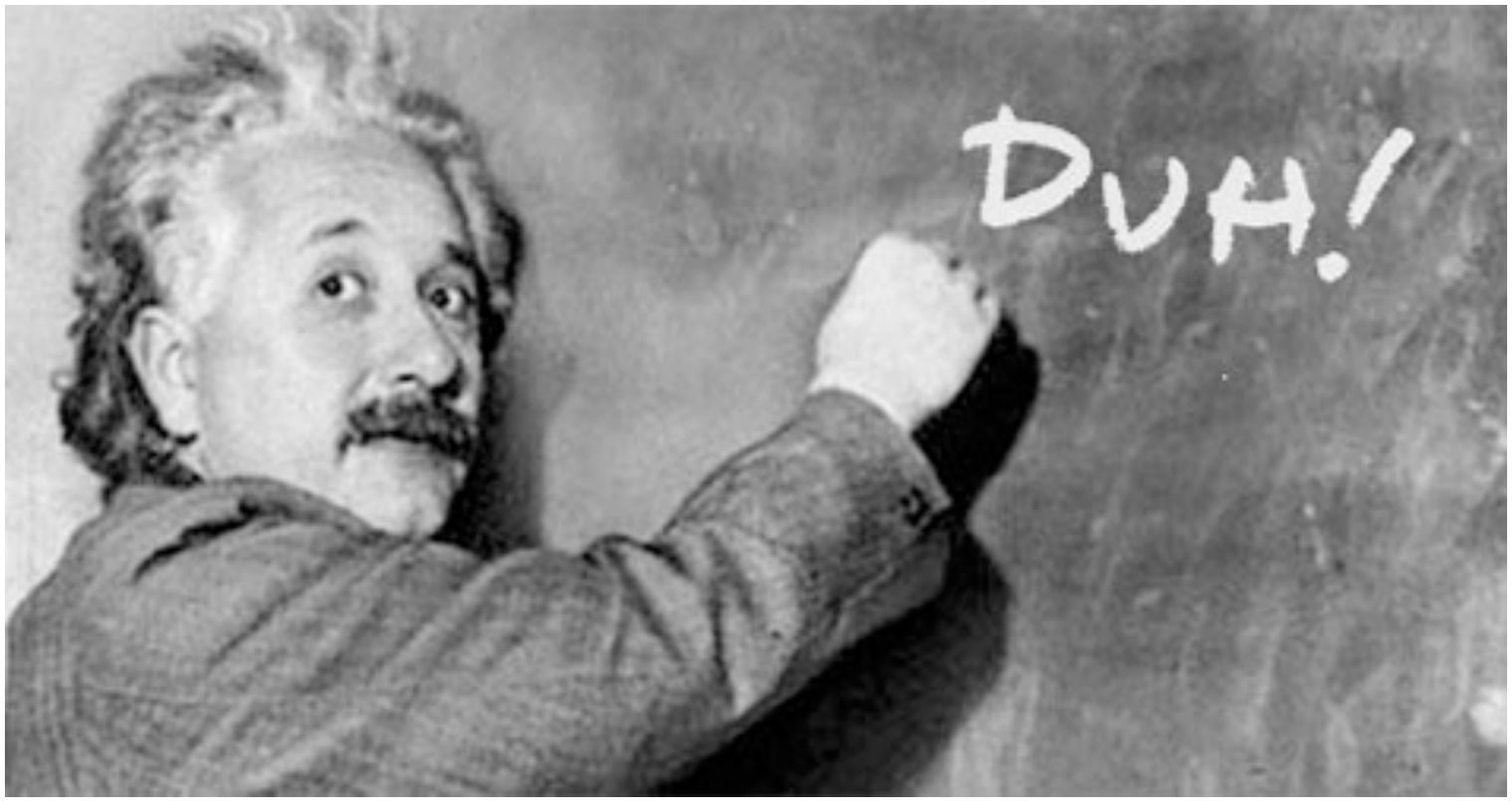
```
$ sbt stage
...

$ tree target/universal/stage/lib/
target/universal/stage/lib/
├── ch.qos.logback.logback-classic-1.1.1.jar
├── ch.qos.logback.logback-core-1.1.1.jar
├── com.fasterxml.jackson.core.jackson-annotations-2.3...
├── com.fasterxml.jackson.core.jackson-core-2.3.2.jar
├── com.fasterxml.jackson.core.jackson-databind-2.3.2.jar
├── com.google.guava.guava-16.0.1.jar
...
```

The 12 Factors

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Dev/prod parity
- Logs
- Admin processes

Don’t check passwords into Git



Or was it “DUH”?

Litmus Test

Can you make your app **open source** at any moment, without compromising any credentials?

Configuration is…

Anything that changes between deployment environments:

- 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

(does not include things like `conf/routes`)

Configuration should be strictly separated from code

Configuration belongs in the environment, not in the application

conf/application.conf

```
db.default.url=${DATABASE_URL}
```

The 12 Factors

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Dev/prod parity
- Logs
- Admin processes

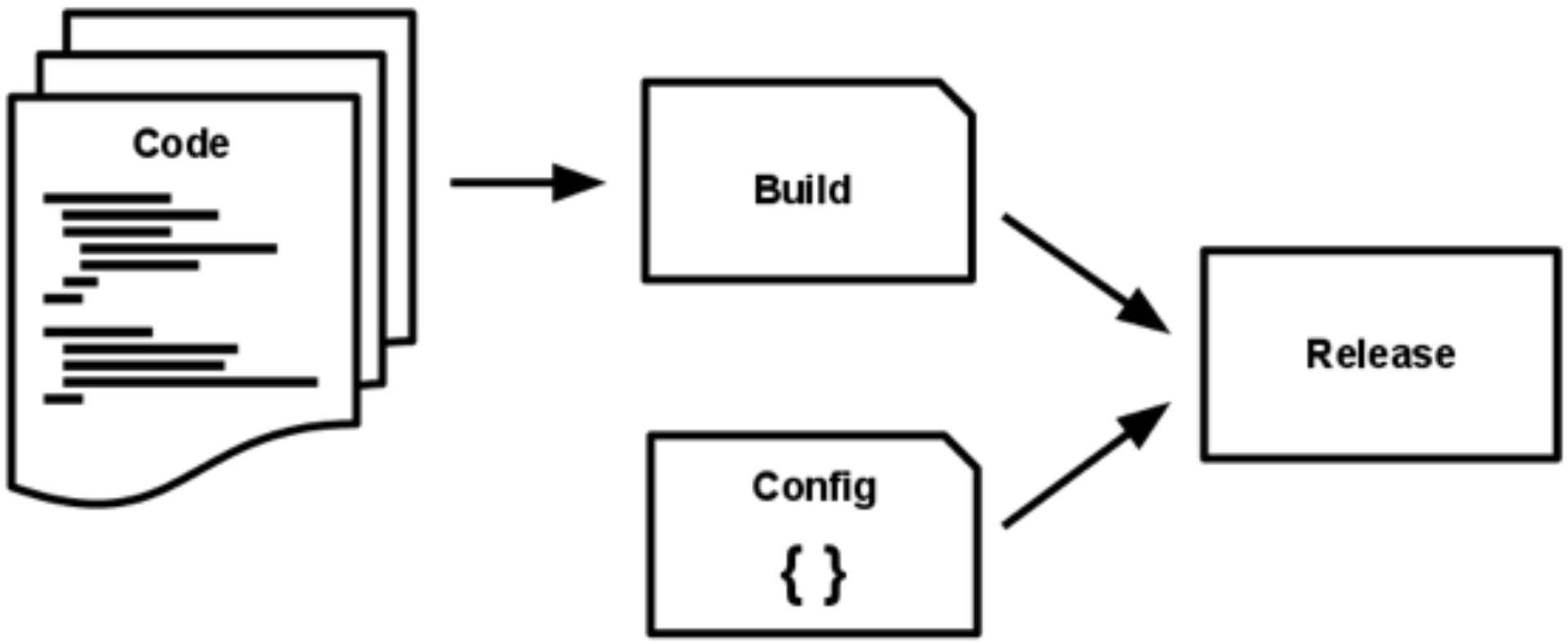
conf/application.conf

```
db.default.url=${DATABASE_URL}
```

The 12 Factors

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Dev/prod parity
- Logs
- Admin processes

build, release, run



build

release

run

jkutner — bash — 128×40

```
$ sbt stage
...

$ sbt deployHeroku
...

# in the cloud!
$ target/universal/stage/bin/my-app
```

```
$ sbt run
```

BAD

(in production)

sbt

simple **build** tool

The 12 Factors

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Dev/prod parity
- Logs
- Admin processes

DEMO!

build

release

run

jkutner — bash — 128×40

```
$ sbt stage
...

$ sbt deployHeroku
...

# in the cloud!
$ target/universal/stage/bin/scaladays
```

The 12 Factors

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Dev/prod parity
- Logs
- Admin processes

Processes should be stateless

sticky sessions

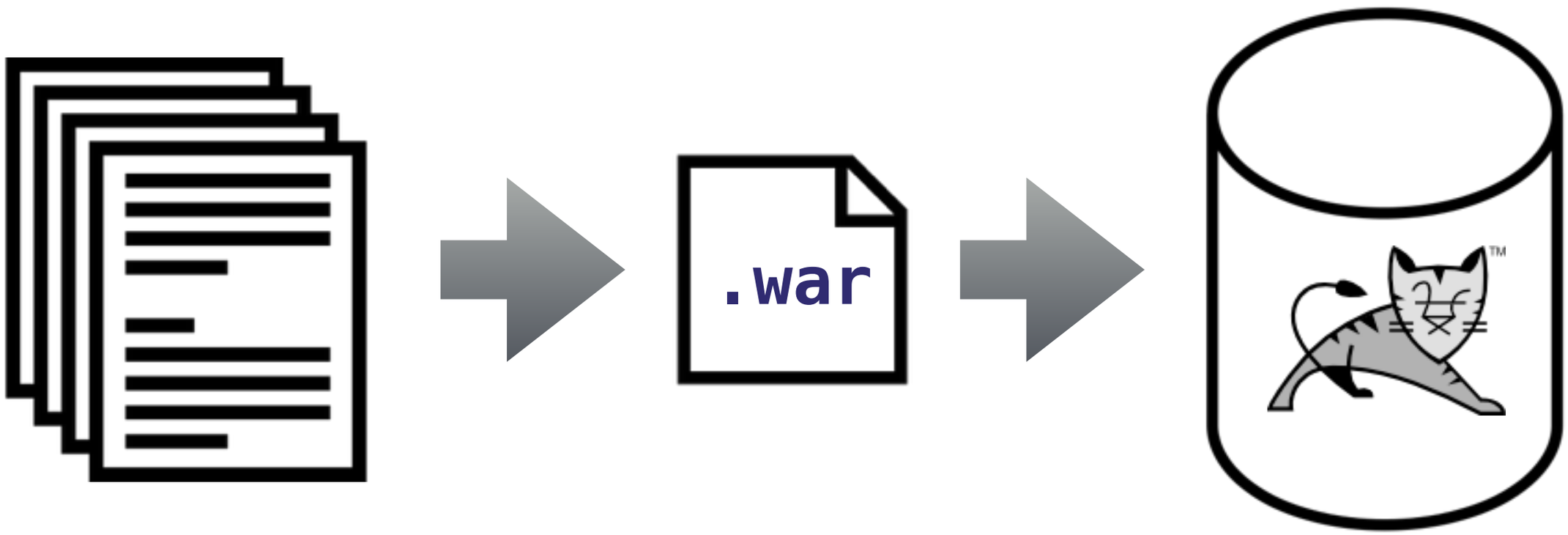
The 12 Factors

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Dev/prod parity
- Logs
- Admin processes

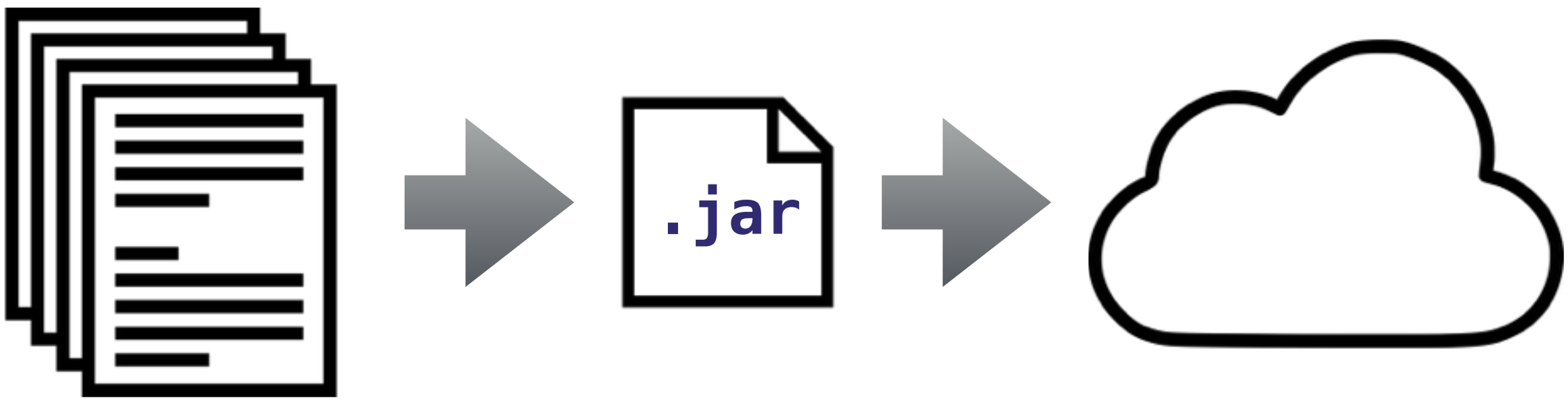
The twelve-factor app
is completely self-contained

The web app exports HTTP
as a service by binding to a port

Traditional Deployment



Modern Deployment



The 12 Factors

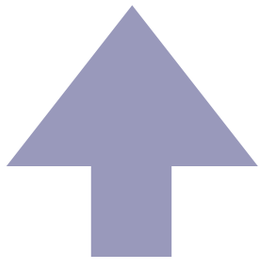
- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Dev/prod parity
- Logs
- Admin processes

Actors

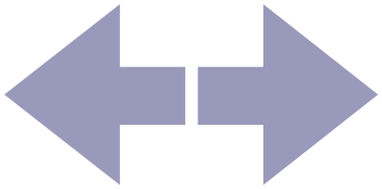
Futures

Agents

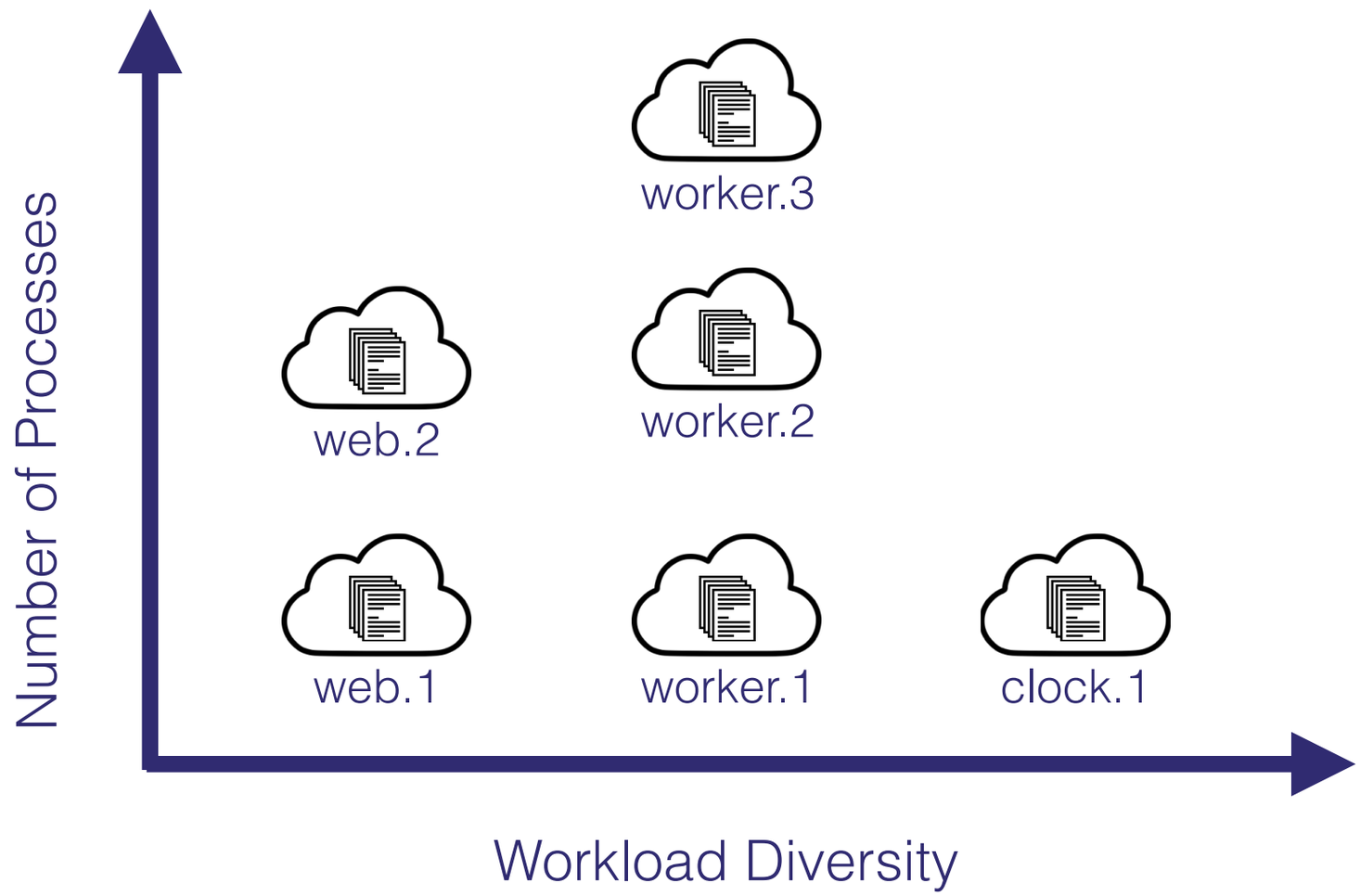
RELAX BRO, I GOT THIS…



Scale Up



Scale Out


Number of Processes
worker.3
web.2
worker.2
web.1
worker.1
clock.1
Workload Diversity

The 12 Factors

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Dev/prod parity
- Logs
- Admin processes

Quick startup

Resilience to failure

Graceful shutdown

Servers are not pets

Servers are cattle

Application Servers are
not disposable

Microservices are disposable

Easy to replace

Easy to modify

Decoupled from
external infrastructure

Microservices

The 12 Factors

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Dev/prod parity
- Logs
- Admin processes

dev = stage = prod

sqlite ≠ mysql ≠ postgres

postgres = postgres = postgres

dev = stage = prod

jetty ≠ tomcat ≠ jboss

jetty = jetty = jetty

dev	**=**	***stage***	**=**	***prod***
jetty	**≠**	***tomcat***	**≠**	***jboss***
{}	**=**	**{}**	**=**	**{}**

Dropwizard

parity $\Rightarrow$

reproducible $\Rightarrow$

disposable

The 12 Factors

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Dev/prod parity
- Logs
- Admin processes

The 12 Factors

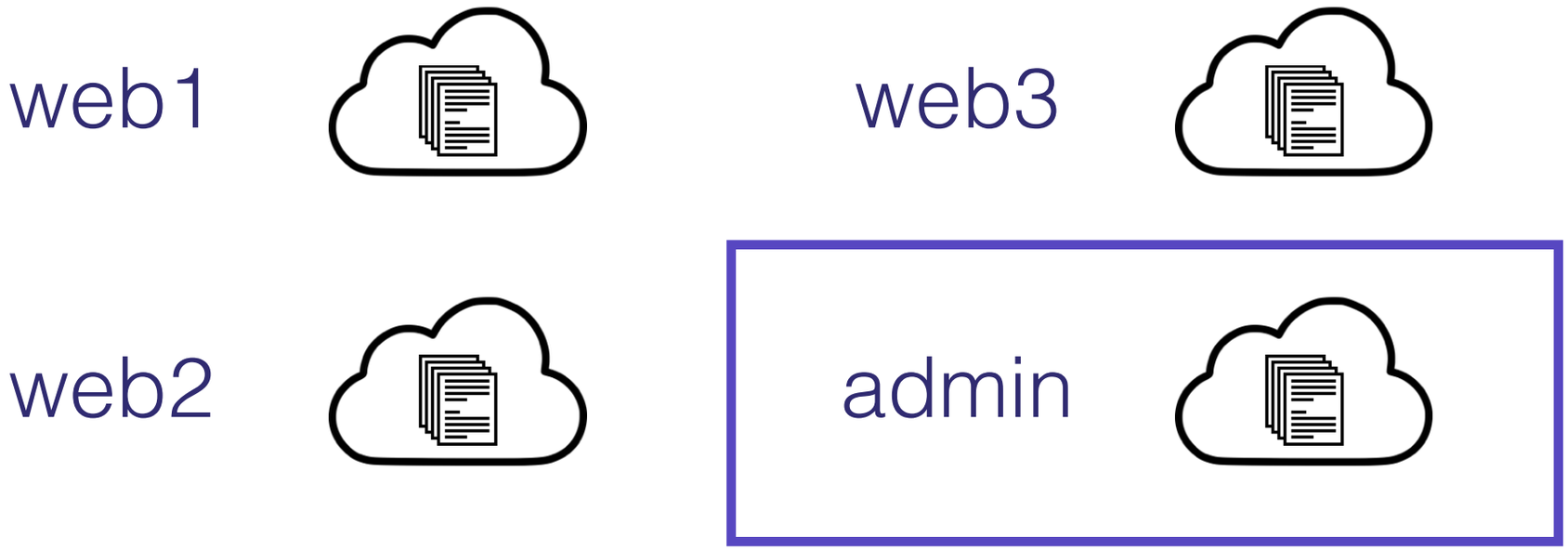
- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Dev/prod parity
- Logs
- Admin processes

Admin tasks should be
run in isolated processes

jkutner — bash — 128×40

```
$ heroku run console
Running `console` attached to terminal... up, run.2581
Picked up JAVA_TOOL_OPTIONS: -Djava.rmi.server.useCode...
Failed to created JLineReader: java.lang.NoClassDefFou...
Falling back to SimpleReader.
Welcome to Scala version 2.11.1 (OpenJDK 64-Bit Server...
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```


web1
web3
web2
admin

jkutner — bash — 128×40

```
$ heroku run sbt console
```

?

sbt

simple **build** tool

project/plugins.sbt

```
addSbtPlugin(
  "com.typesafe.sbt" % "sbt-native-packager" % "0.7.6"
)
```

Procfile

```
console: target/universal/stage/bin/my-app \
            -main scala.tools.nsc.MainGenericRunner \
            -usejavacp

worker:  target/universal/stage/bin/my-app \
            -main com.example.MyWorker
```

The 12 Factors

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Dev/prod parity
- Logs
- Admin processes

http://12factor.net

http://jkutner.github.io

What next?

1. Add sbt-native-packager
2. Run `sbt stage`
3. Deploy to Heroku?
4. Go to the sbt-native-packager talk

Joe Kutner

@codefinger

JVM Platform Owner
@Heroku

http://www.slideshare.net/jkutner/12-factor-scala