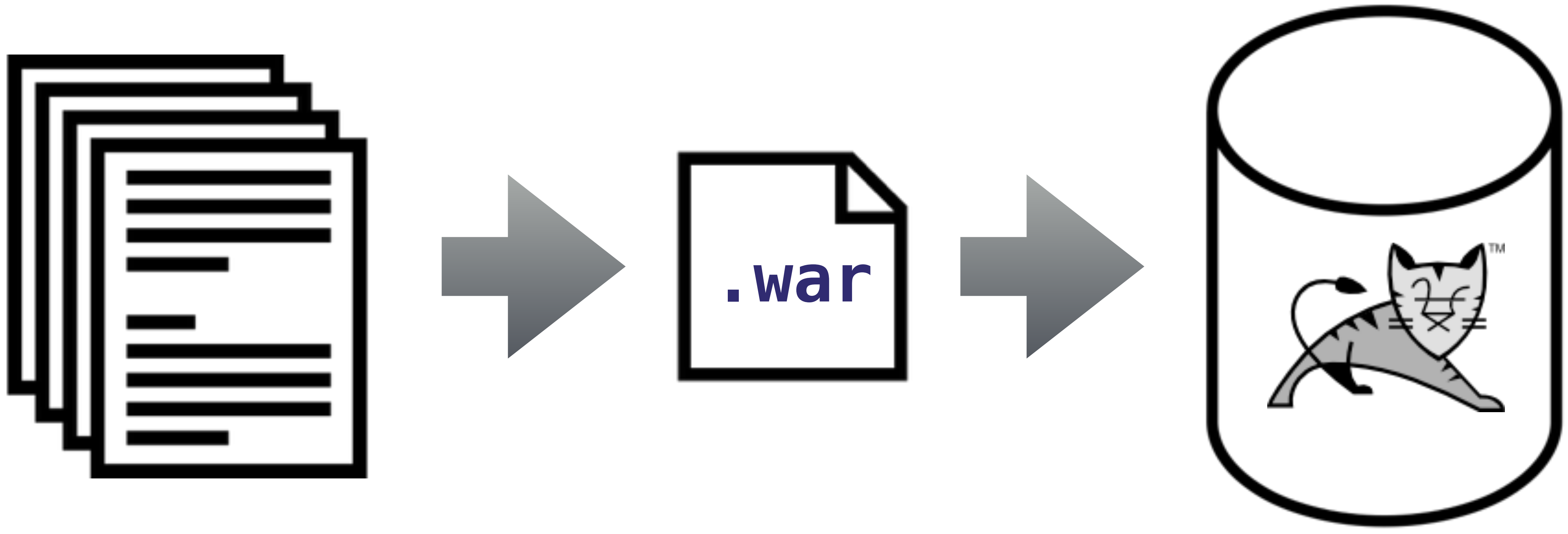
12 Factor App

Best Practices for Java Deployment

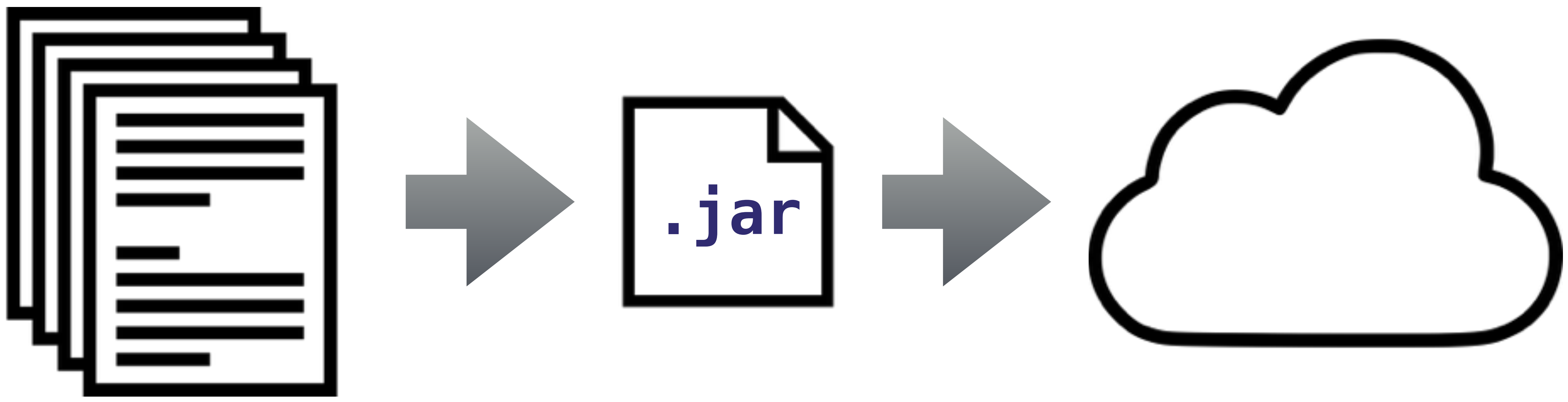
You're <u>not</u> going to like what I have to **say**

Modern Java
Deployment

Traditional Java Deployment



Modern Java Deployment



Make JAR
Not WAR
AFGHANISTAN
GOD

2005	2015
WAR files	JAR files
App Servers	Microservices
Hot Deploy	Continuous Deploy
Server in a closet	Heroku/AWS/etc

Joe Kutner

@codefinger

JVM Platform Owner

heroku

12 Factor App

a methodology

Scalability

Maintainability

Portability

- **Immutable**
- Ephemeral
- Declarative
- Automated

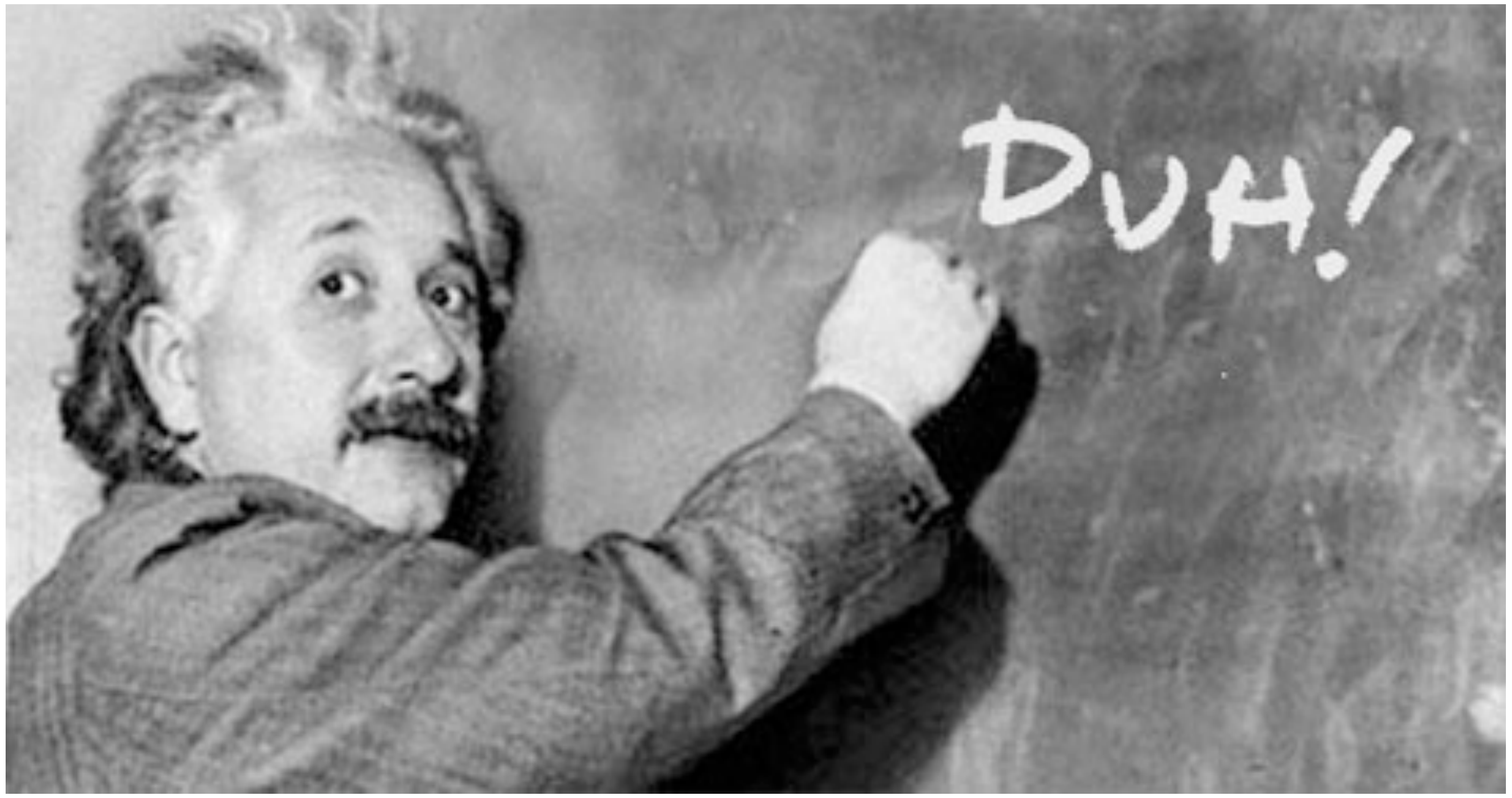
12 **Factor** App

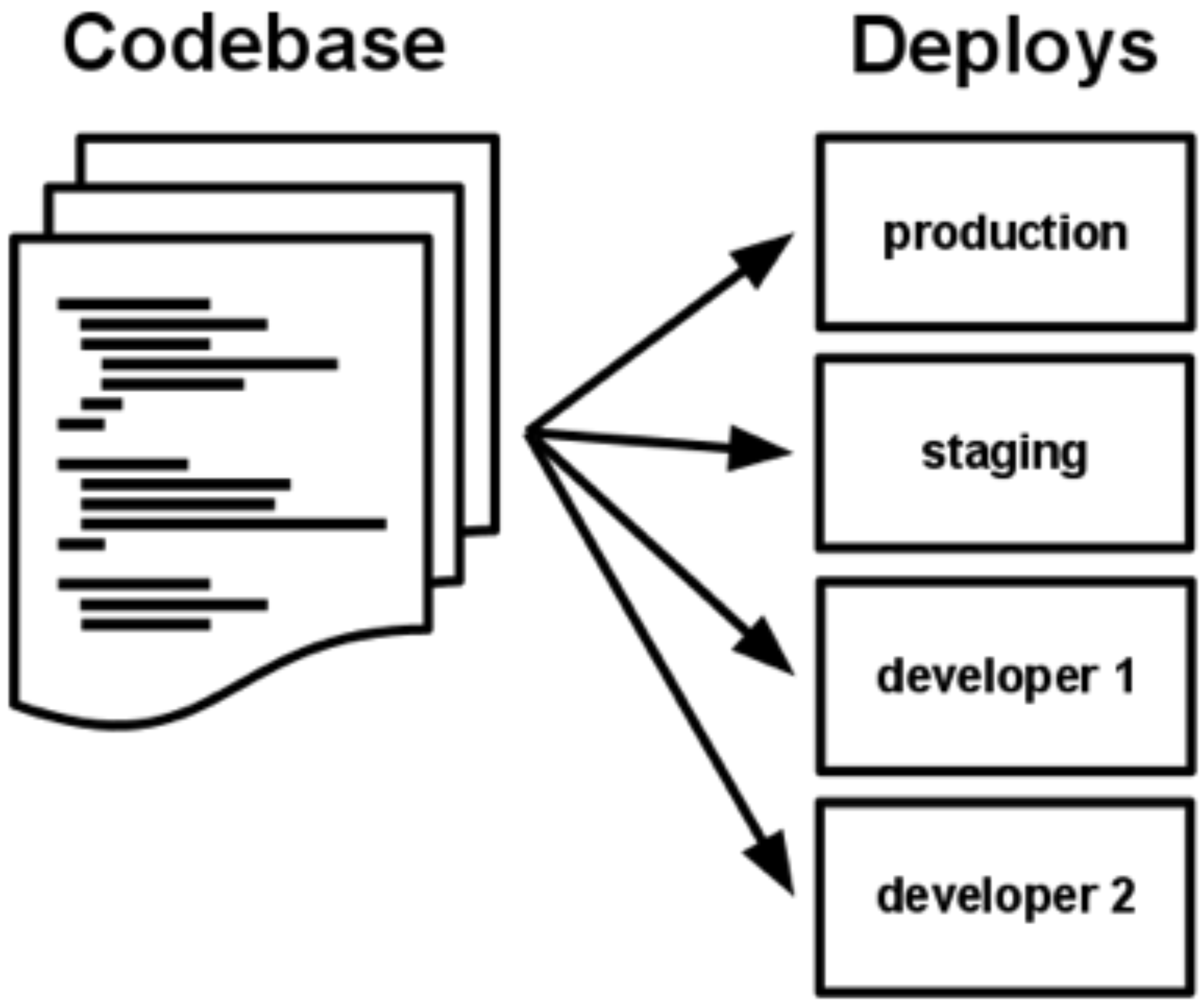
- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port Binding
- Concurrency
- Disposability
- Dev/Prod Parity
- Logs
- Admin Processes

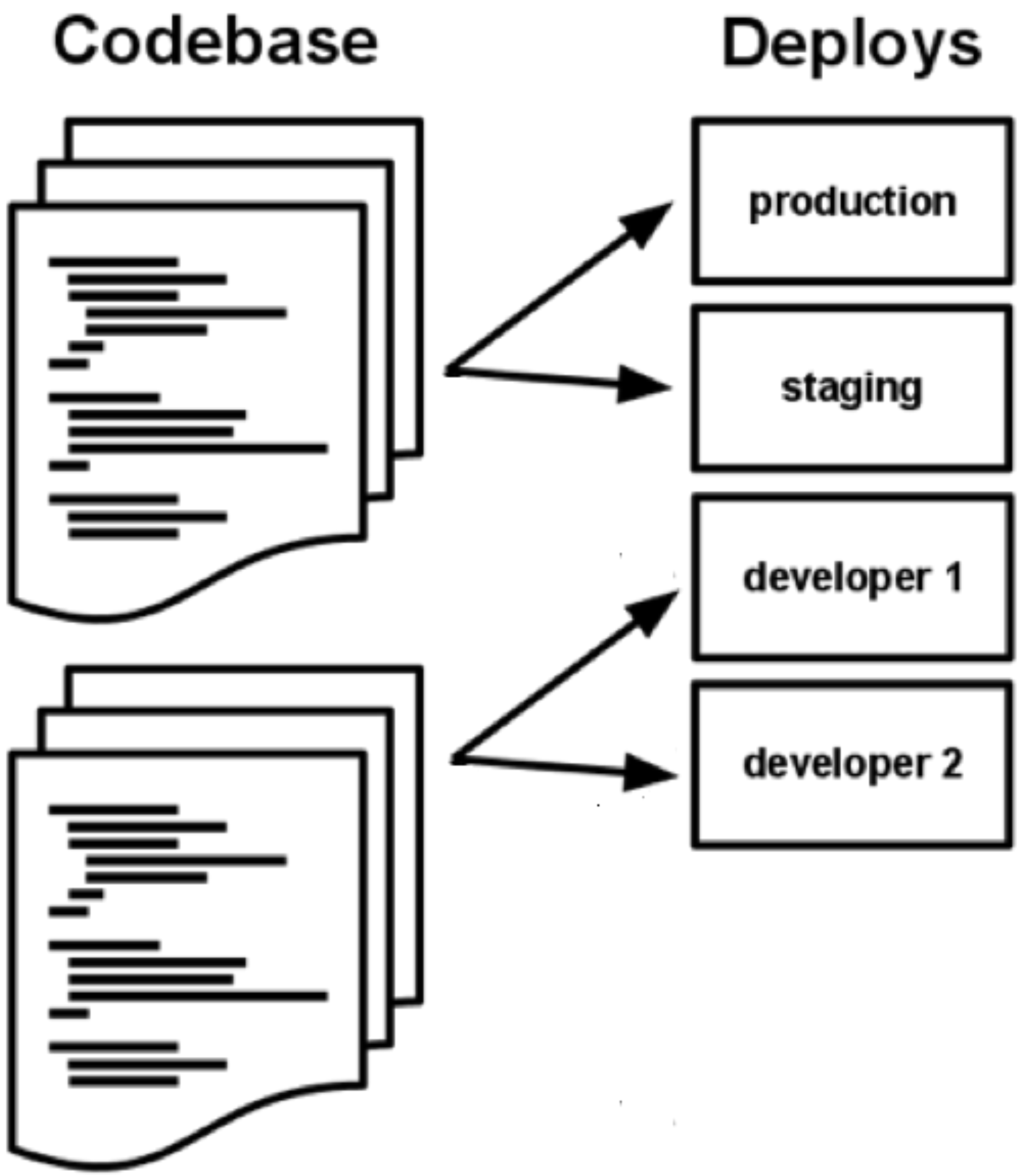
12 Factor App

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port Binding
- Concurrency
- Disposability
- Dev/Prod Parity
- Logs
- Admin Processes

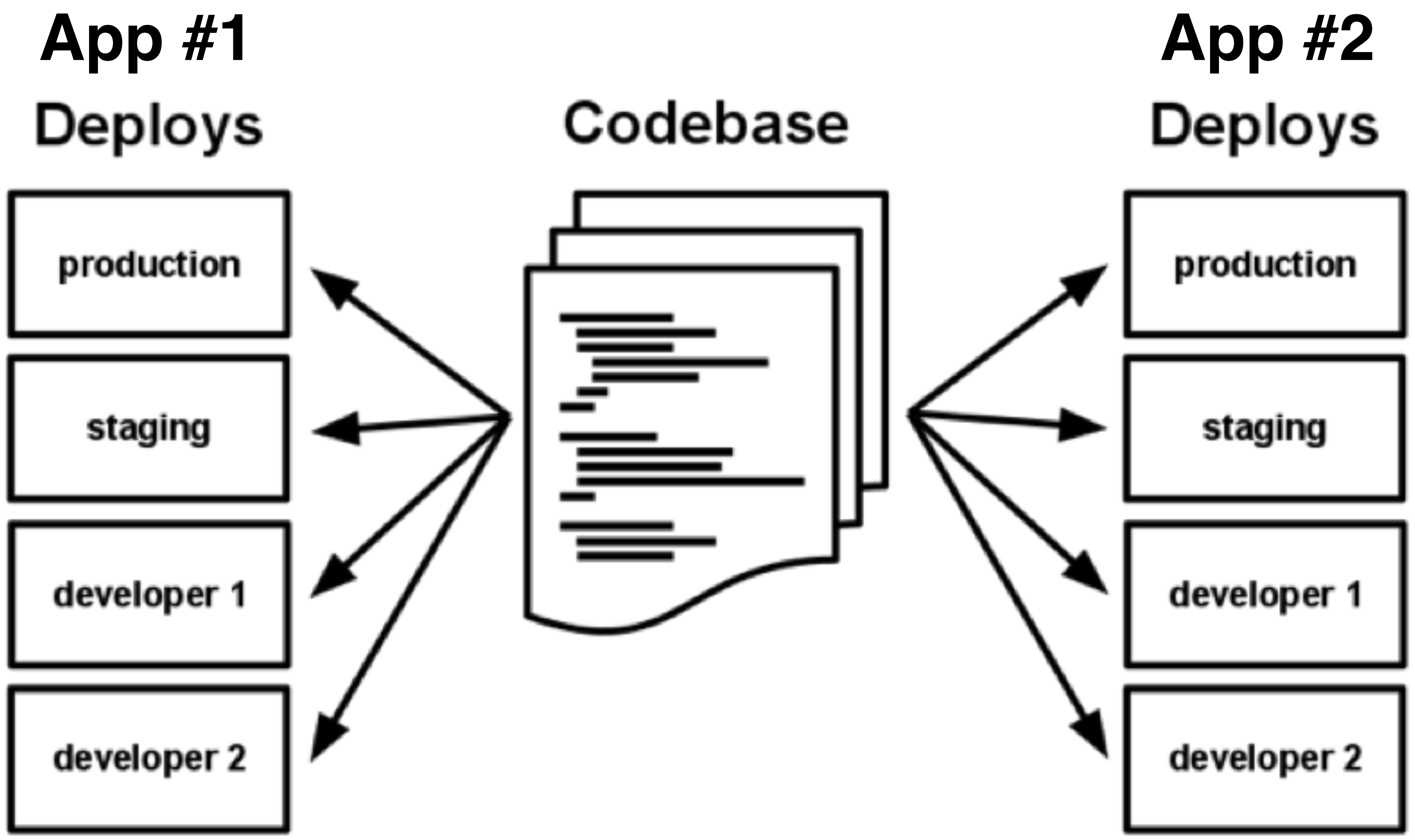
Use version control


DUH!

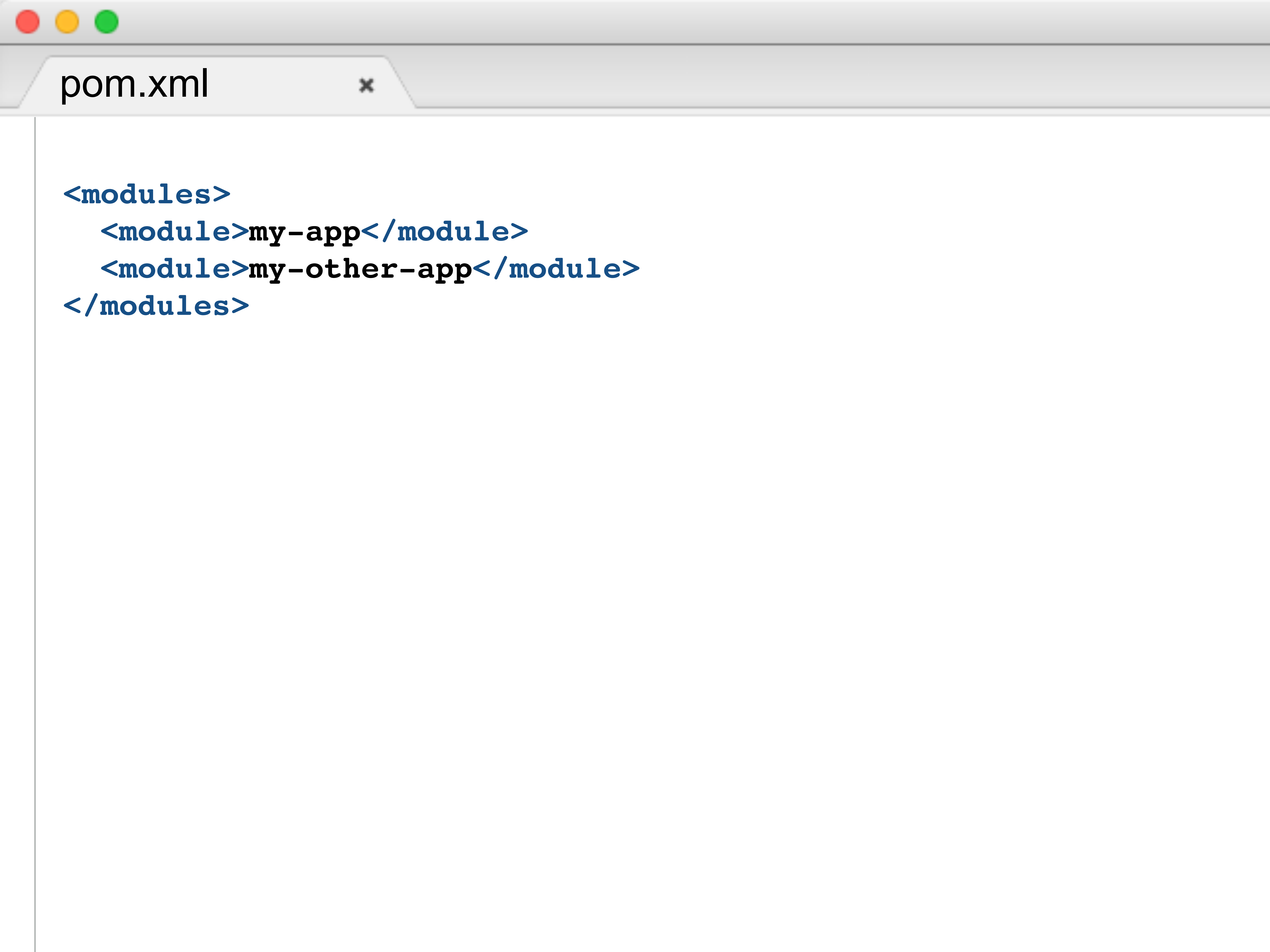

Codebase
Deploys
production
staging
developer 1
developer 2


Codebase
Deploys
production
staging
developer 1
developer 2

BAD


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

BAD


pom.xml

```xml
<modules>
  <module>my-app</module>
  <module>my-other-app</module>
</modules>
```

Submodules

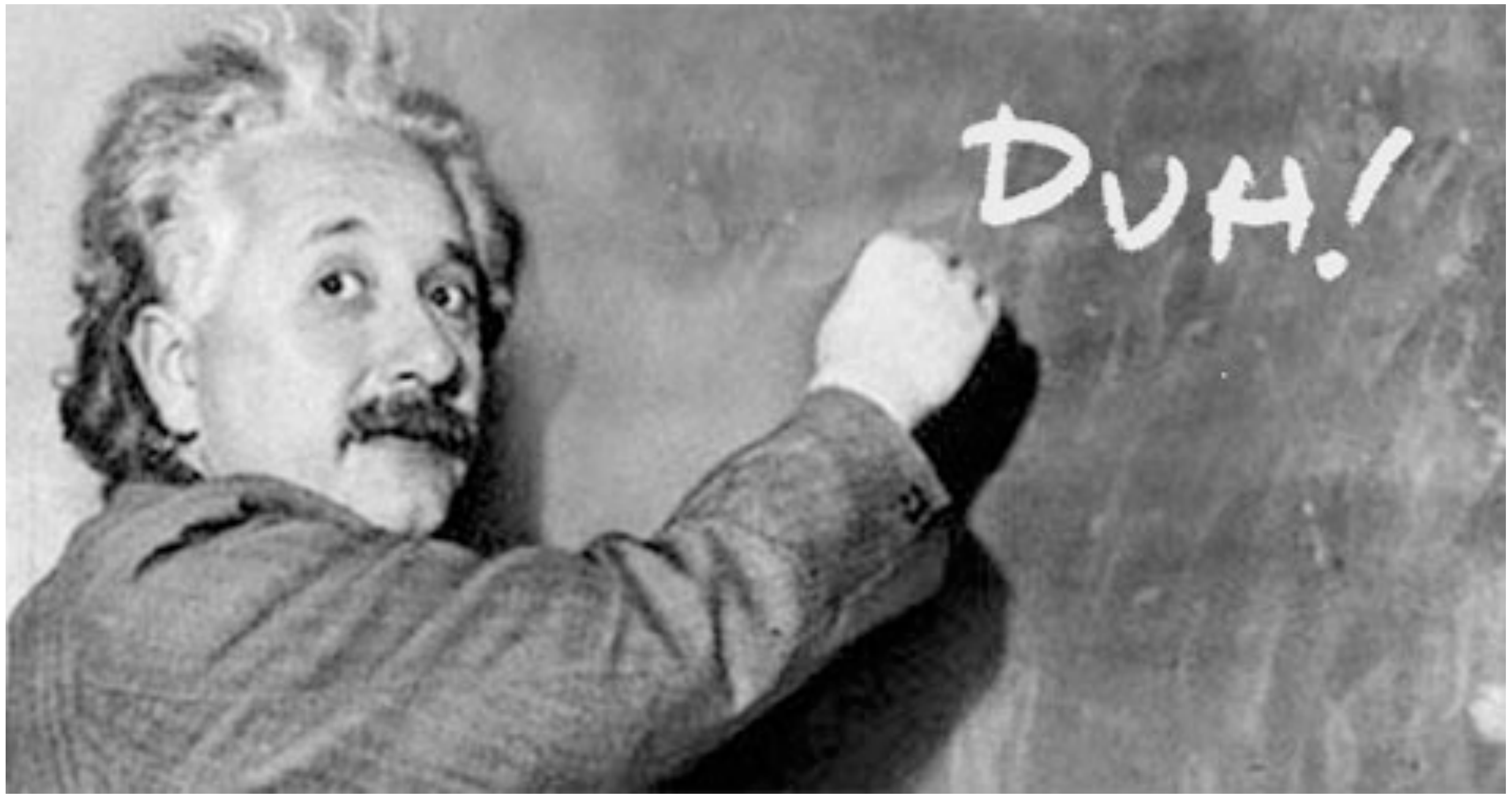
12 Factor App

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port Binding
- Concurrency
- Disposability
- Dev/Prod Parity
- Logs
- Admin Processes

Explicitly declare and isolate dependencies

Never rely on implicit existence of system-wide packages

<u>Don't</u> check JAR files into Git

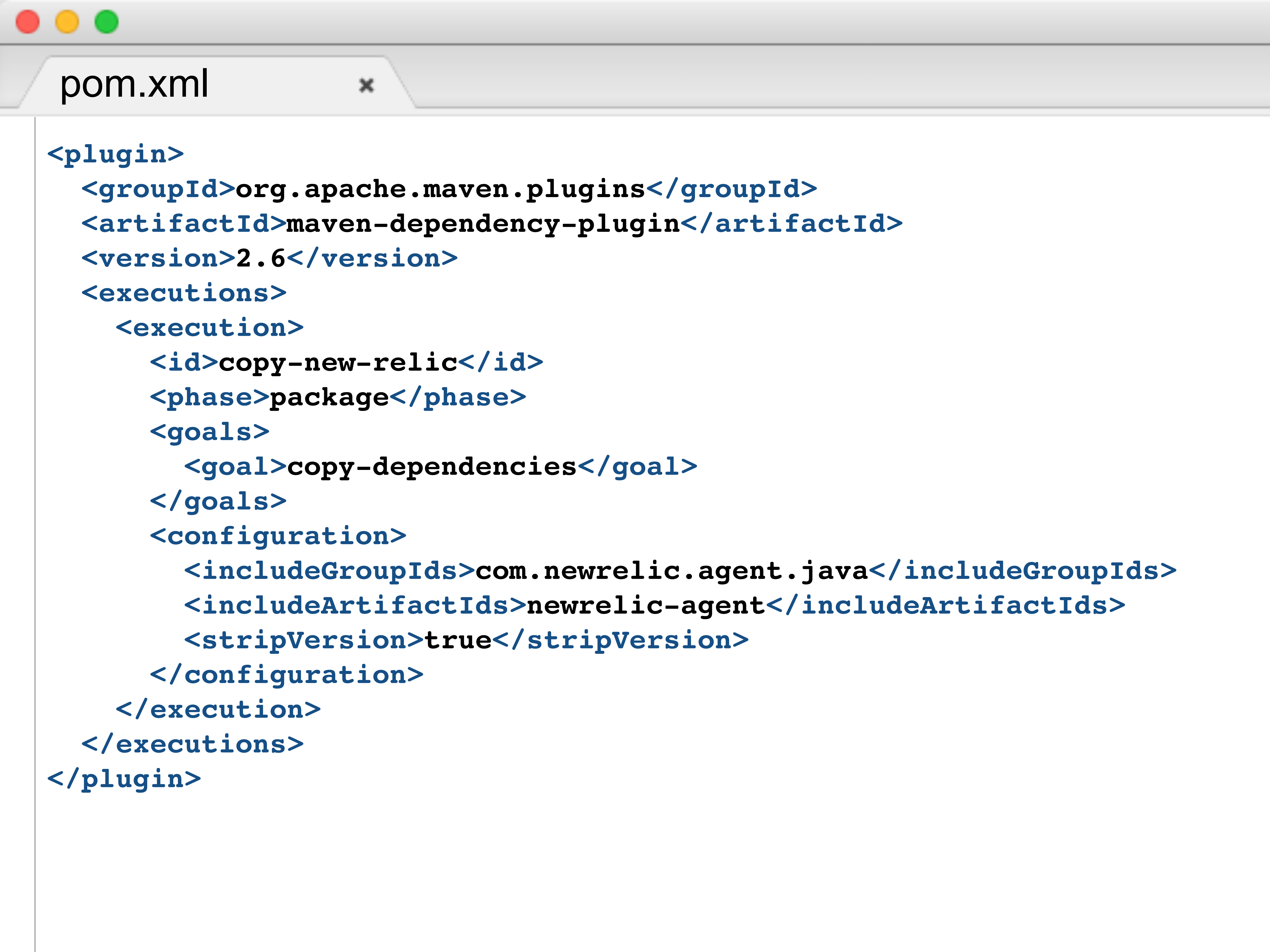

DUH!

Bintray

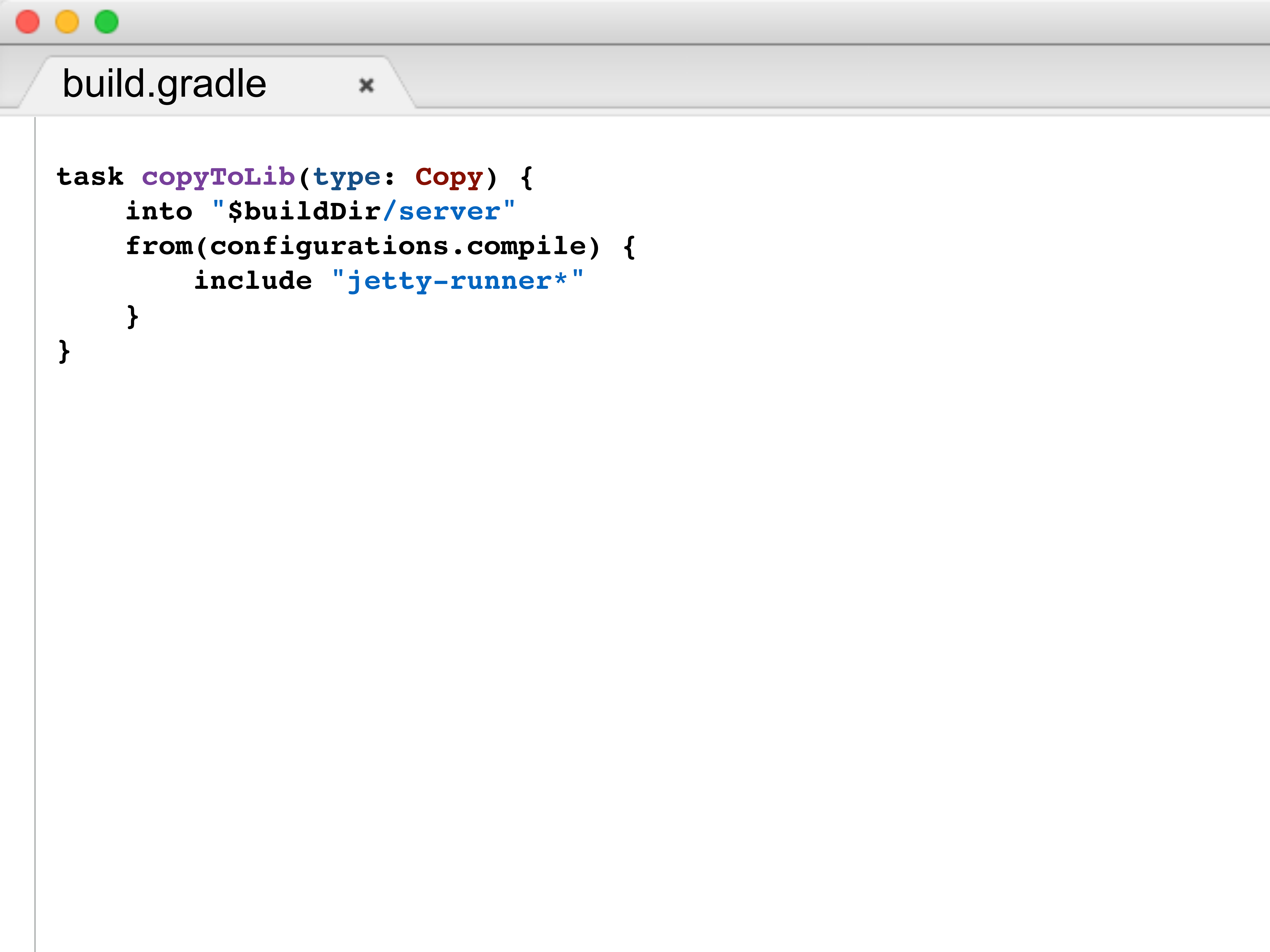
https://bintray.com/

Agent

JARS


pom.xml

```xml
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-dependency-plugin</artifactId>
  <version>2.6</version>
  <executions>
    <execution>
      <id>copy-new-relic</id>
      <phase>package</phase>
      <goals>
        <goal>copy-dependencies</goal>
      </goals>
      <configuration>
        <includeGroupIds>com.newrelic.agent.java</includeGroupIds>
        <includeArtifactIds>newrelic-agent</includeArtifactIds>
        <stripVersion>true</stripVersion>
      </configuration>
    </execution>
  </executions>
</plugin>
```



```
task copyToLib(type: Copy) {
    into "$buildDir/server"
    from(configurations.compile) {
        include "jetty-runner*"
    }
}
```

12 Factor App

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port Binding
- Concurrency
- Disposability
- Dev/Prod Parity
- Logs
- Admin Processes

Configuration is…

Anything that <u>changes</u> between deployment environments:

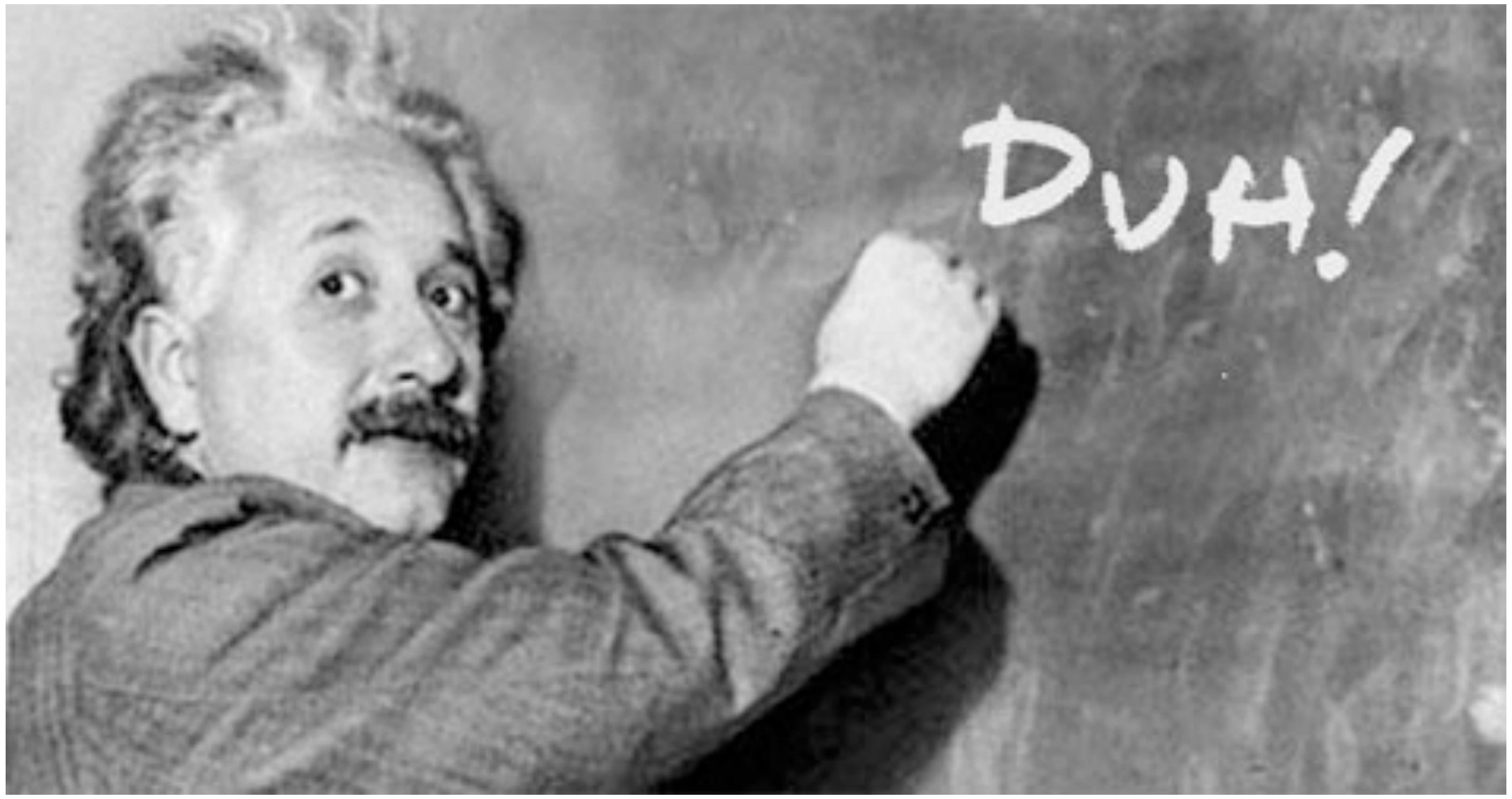
- Resource handles to the **database**, **memcached**, and other backing services
- Credentials to external services such as **Amazon S3** or **Twitter**
- Per-deploy values such as the canonical hostname for the deploy

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

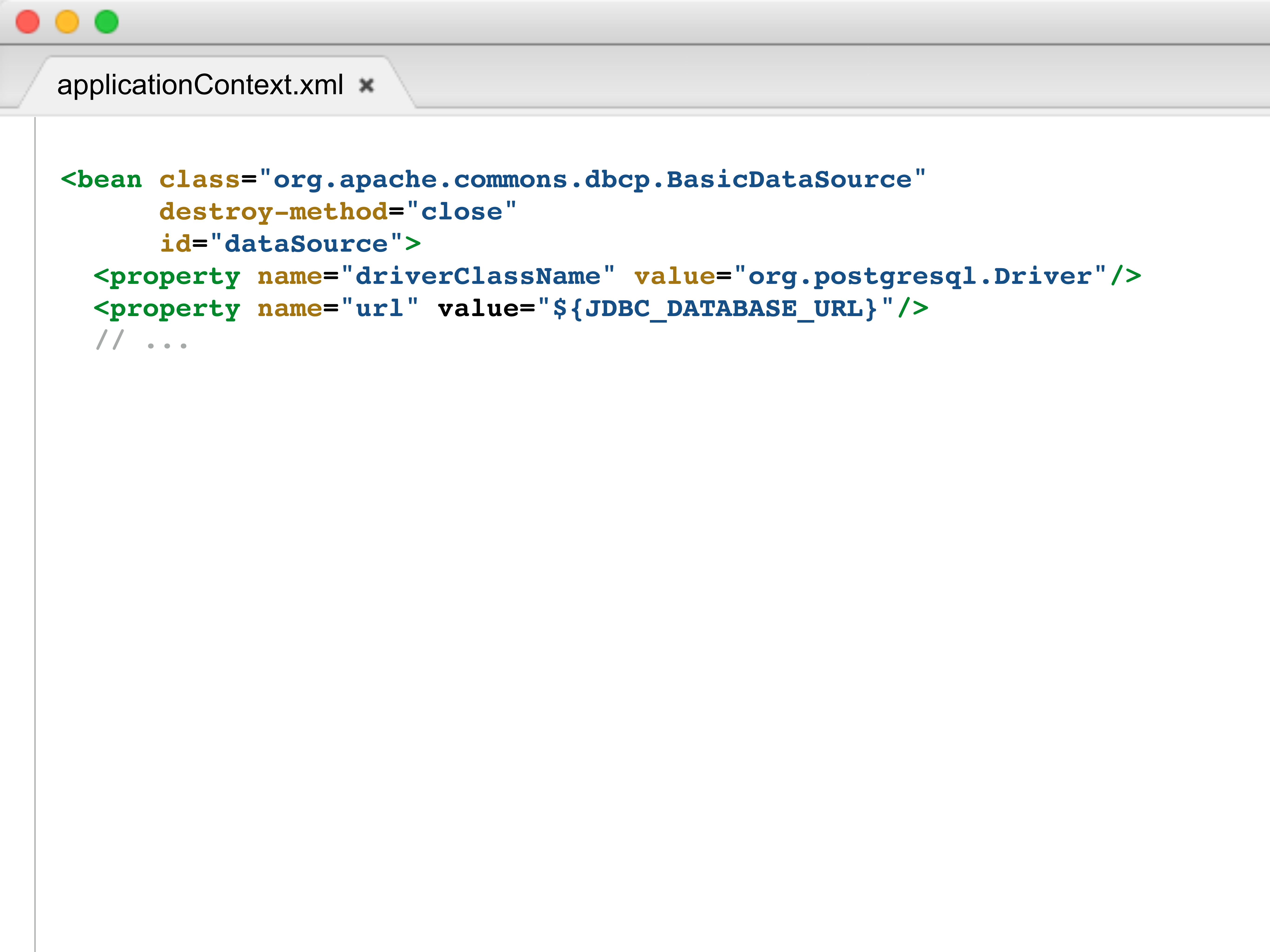
✂️

Configuration should be strictly **separated** from code

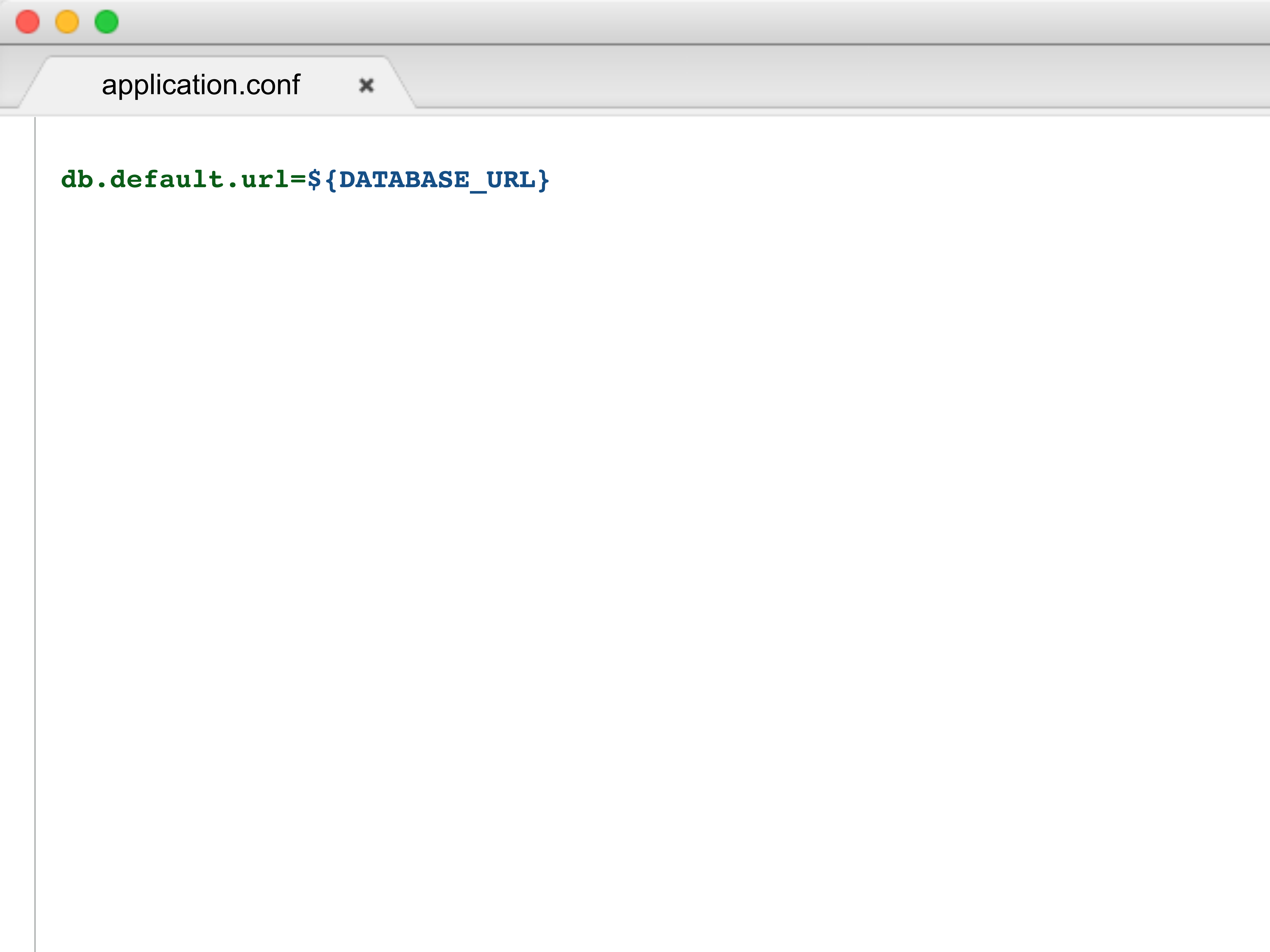
Don't check **passwords** into Git


DUH!

Configuration belongs
in the **environment**,
not in the application



```
<bean class="org.apache.commons.dbcp.BasicDataSource"
      destroy-method="close"
      id="dataSource">
  <property name="driverClassName" value="org.postgresql.Driver"/>
  <property name="url" value="${JDBC_DATABASE_URL}"/>
  // ...
```



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

Litmus Test

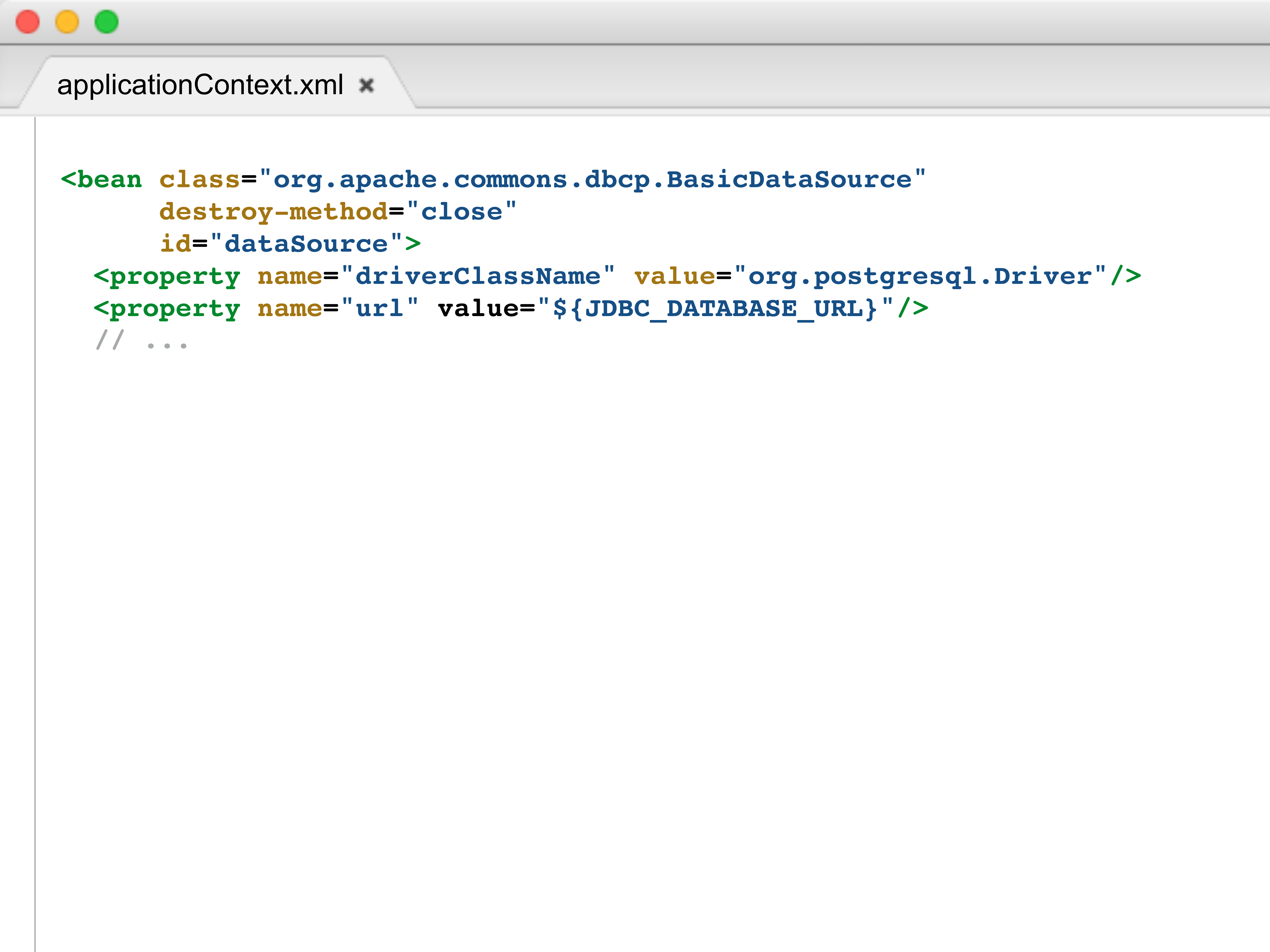
Can you make your app open source at any moment, **without** compromising any credentials?

12 Factor App

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port Binding
- Concurrency
- Disposability
- Dev/Prod Parity
- Logs
- Admin Processes

redis

MEMCACHED

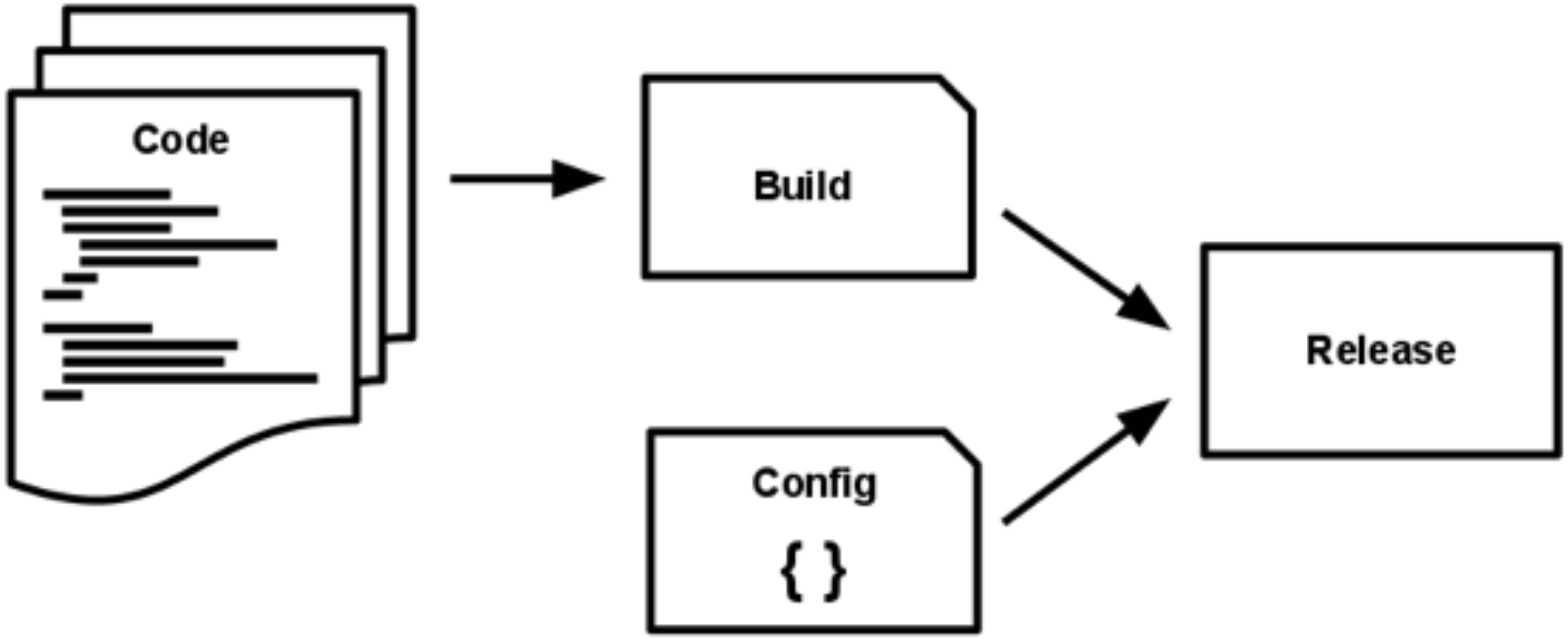


```
<bean class="org.apache.commons.dbcp.BasicDataSource"
      destroy-method="close"
      id="dataSource">
  <property name="driverClassName" value="org.postgresql.Driver"/>
  <property name="url" value="${JDBC_DATABASE_URL}"/>
  // ...
```

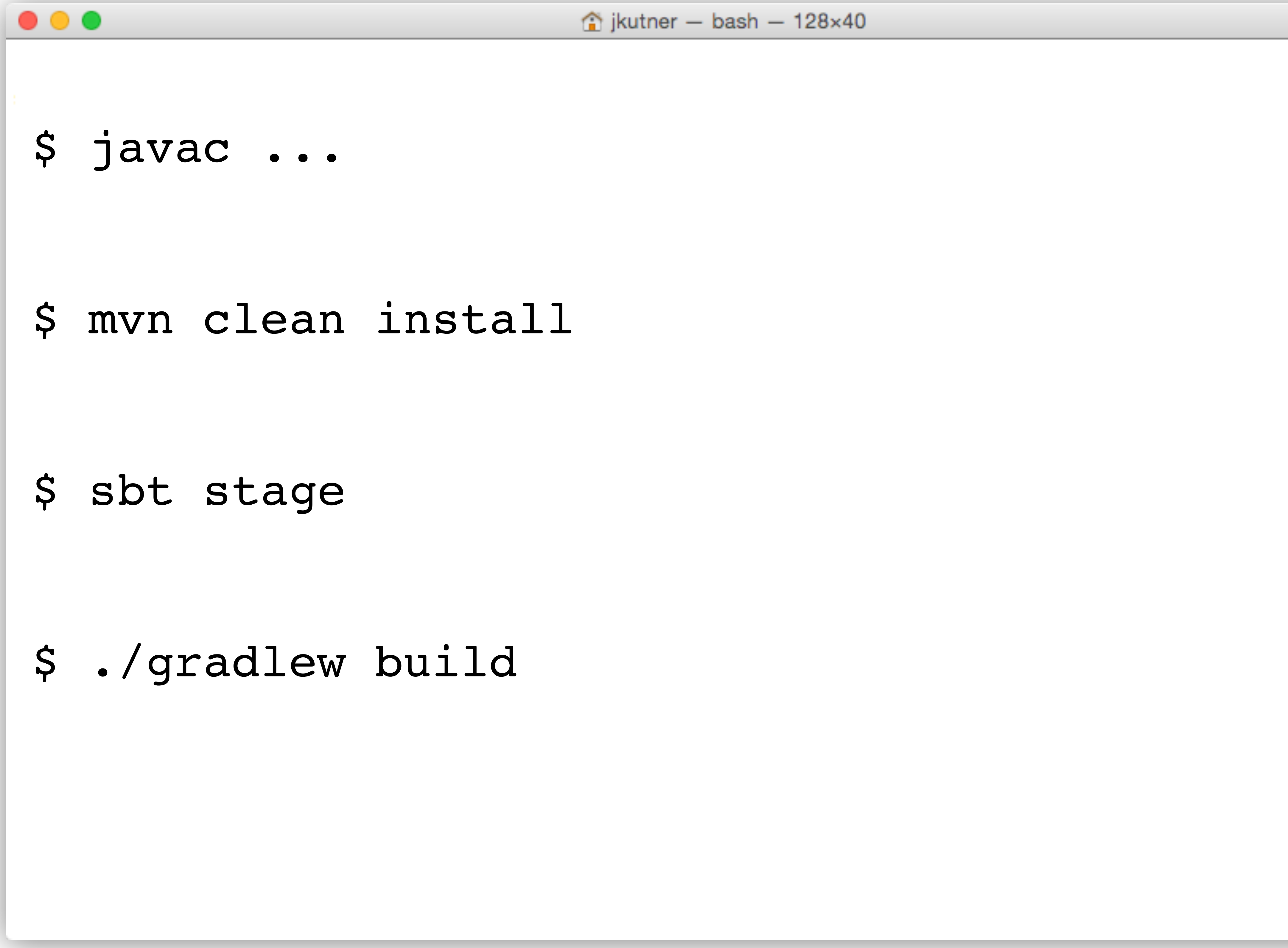
12 Factor App

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port Binding
- Concurrency
- Disposability
- Dev/Prod Parity
- Logs
- Admin Processes

build, release, run

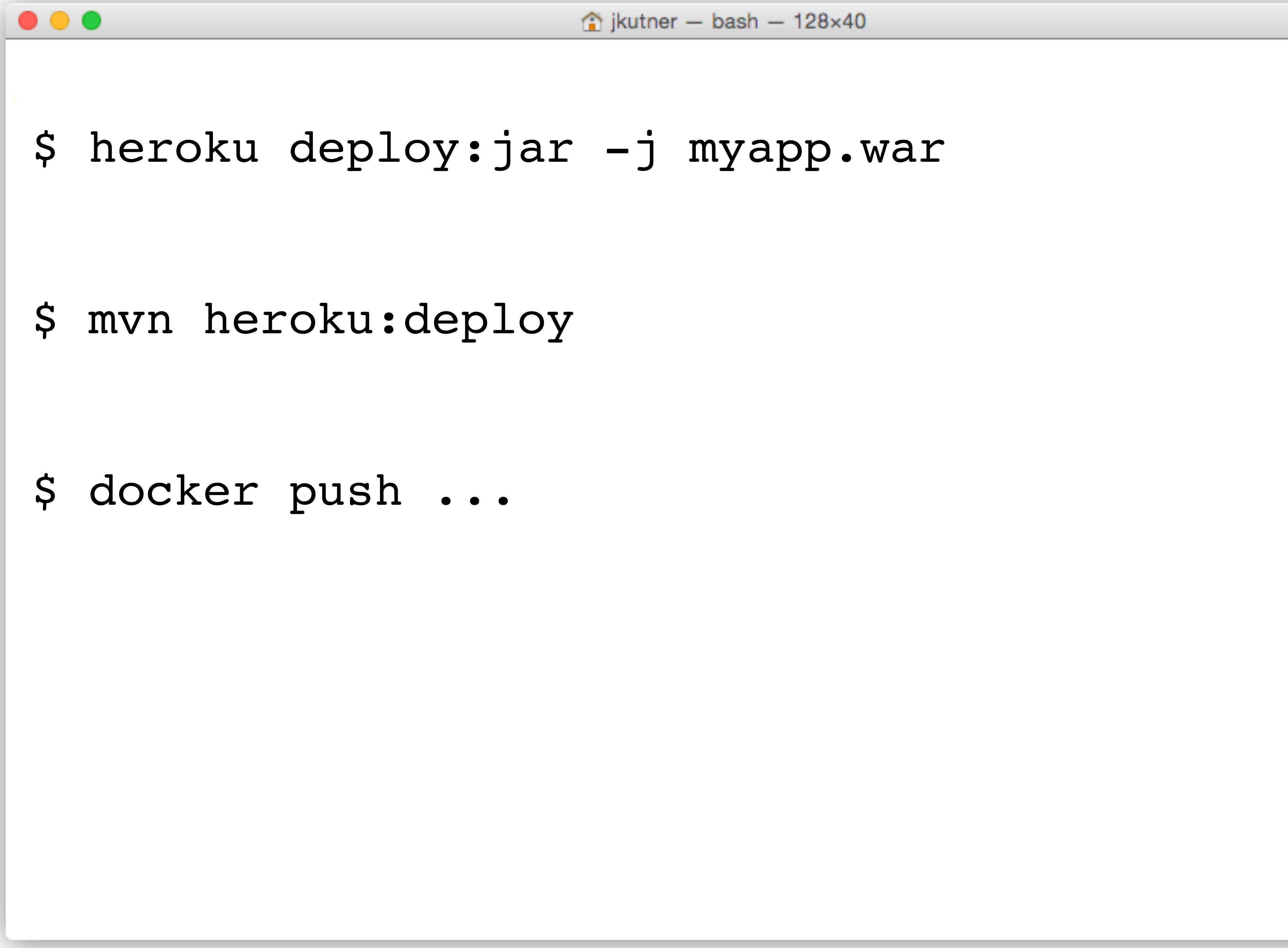


build



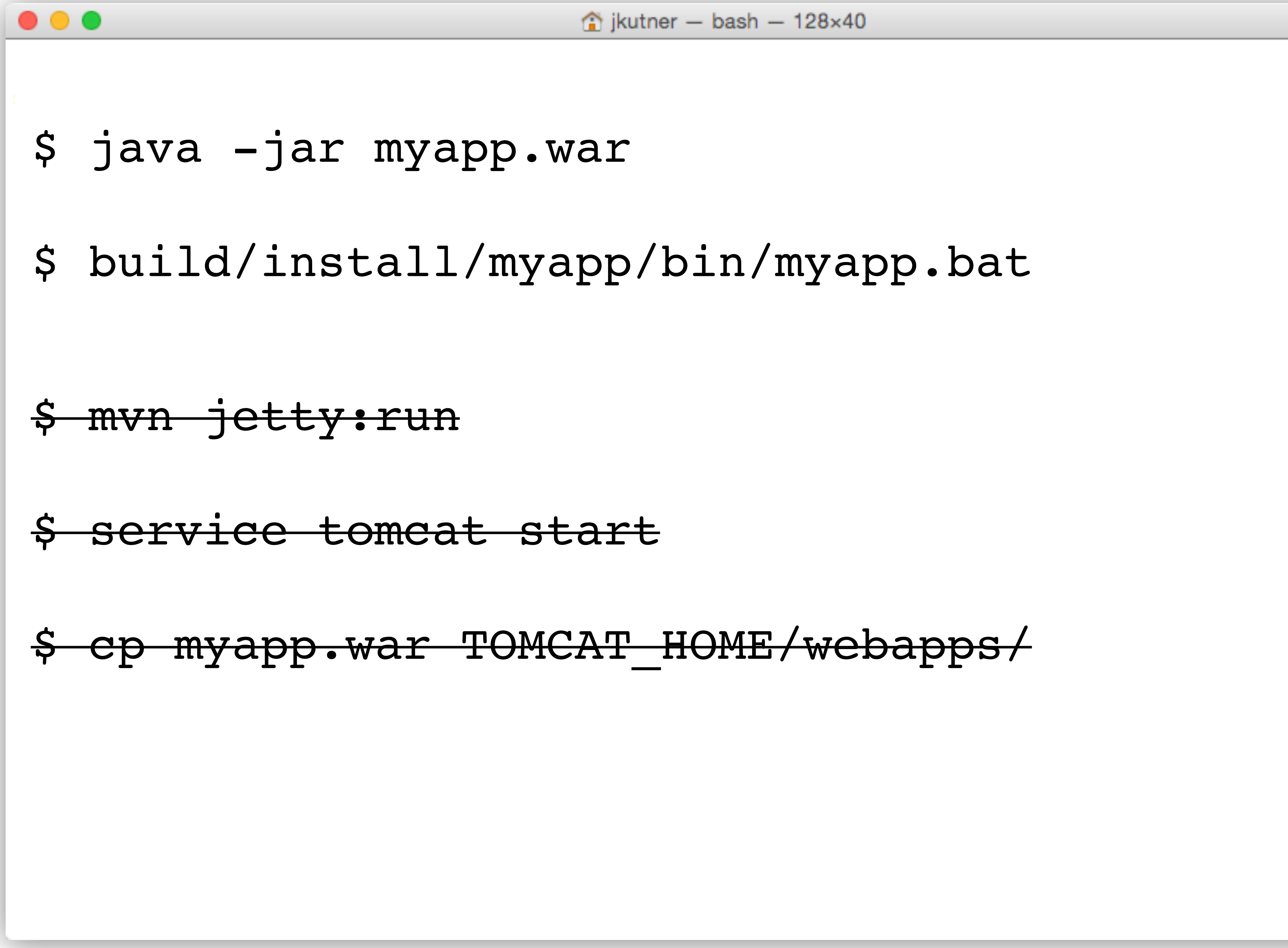
```
$ javac ...

$ mvn clean install

$ sbt stage

$ ./gradlew build
```

release



```
$ heroku deploy:jar -j myapp.war

$ mvn heroku:deploy

$ docker push ...
```

run



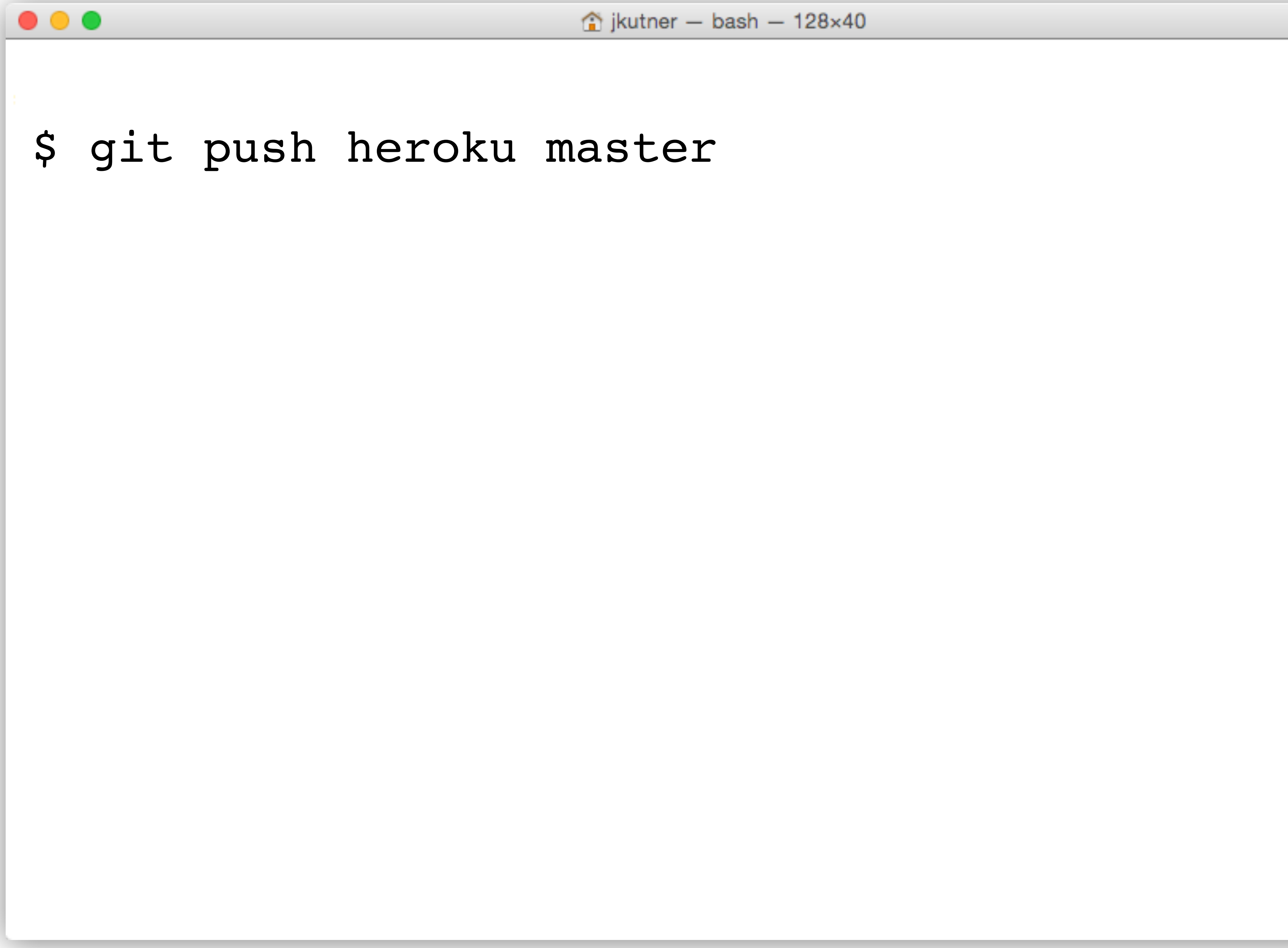
```
$ java -jar myapp.war

$ build/install/myapp/bin/myapp.bat
```

~~$ mvn jetty:run~~

~~$ service tomcat start~~

~~$ cp myapp.war TOMCAT_HOME/webapps/~~

build,
release,
& run



```
$ git push heroku master
```

12 Factor App

- 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
😞

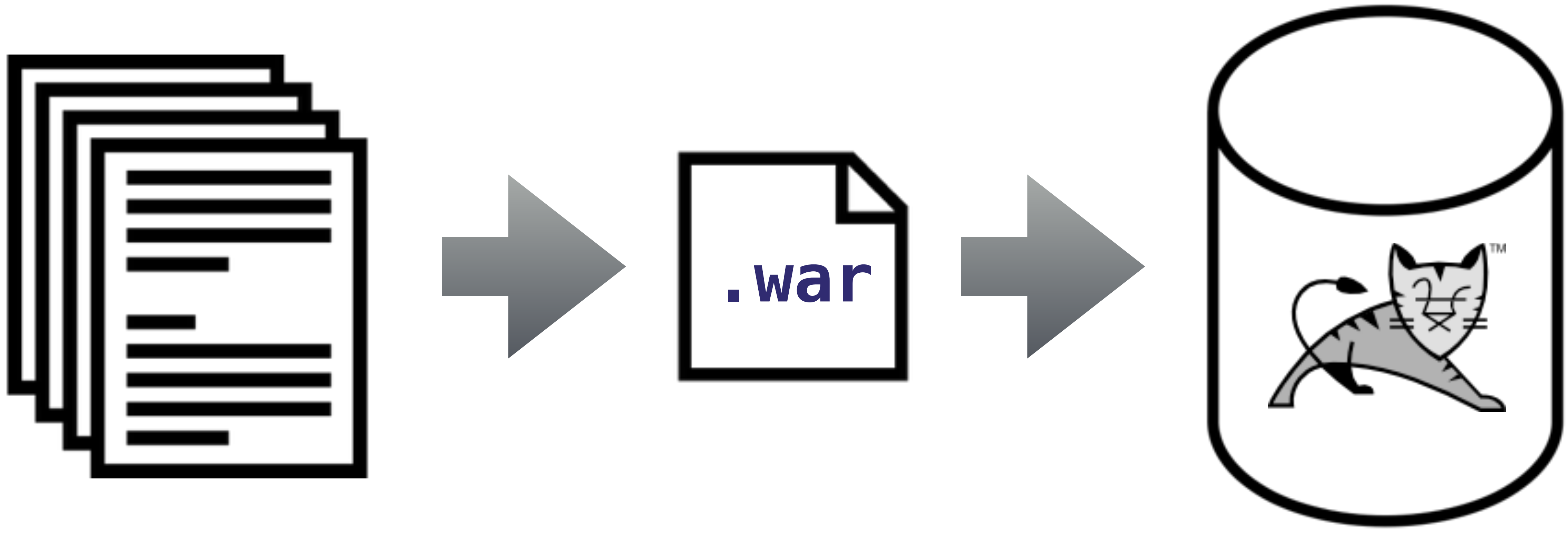
12 Factor App

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port Binding
- Concurrency
- Disposability
- Dev/Prod Parity
- Logs
- Admin Processes

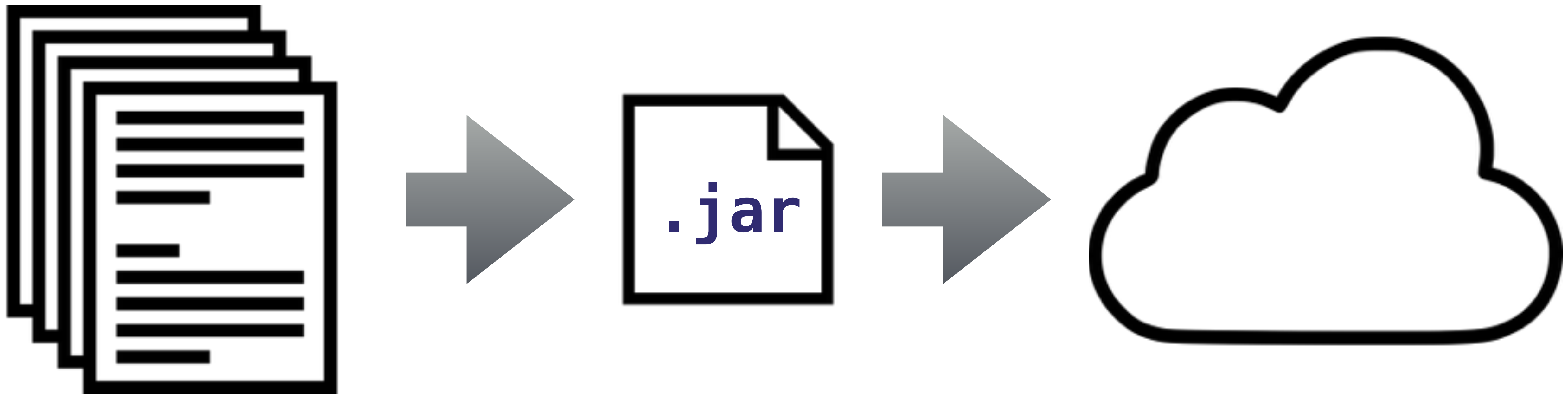
The twelve-factor app
is completely self-contained

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

Traditional Deployment



Modern Deployment



spring

Ratpack

A toolkit for JVM web applications

play

finagle

WildFly SWARM

Dropwizard

12 Factor App

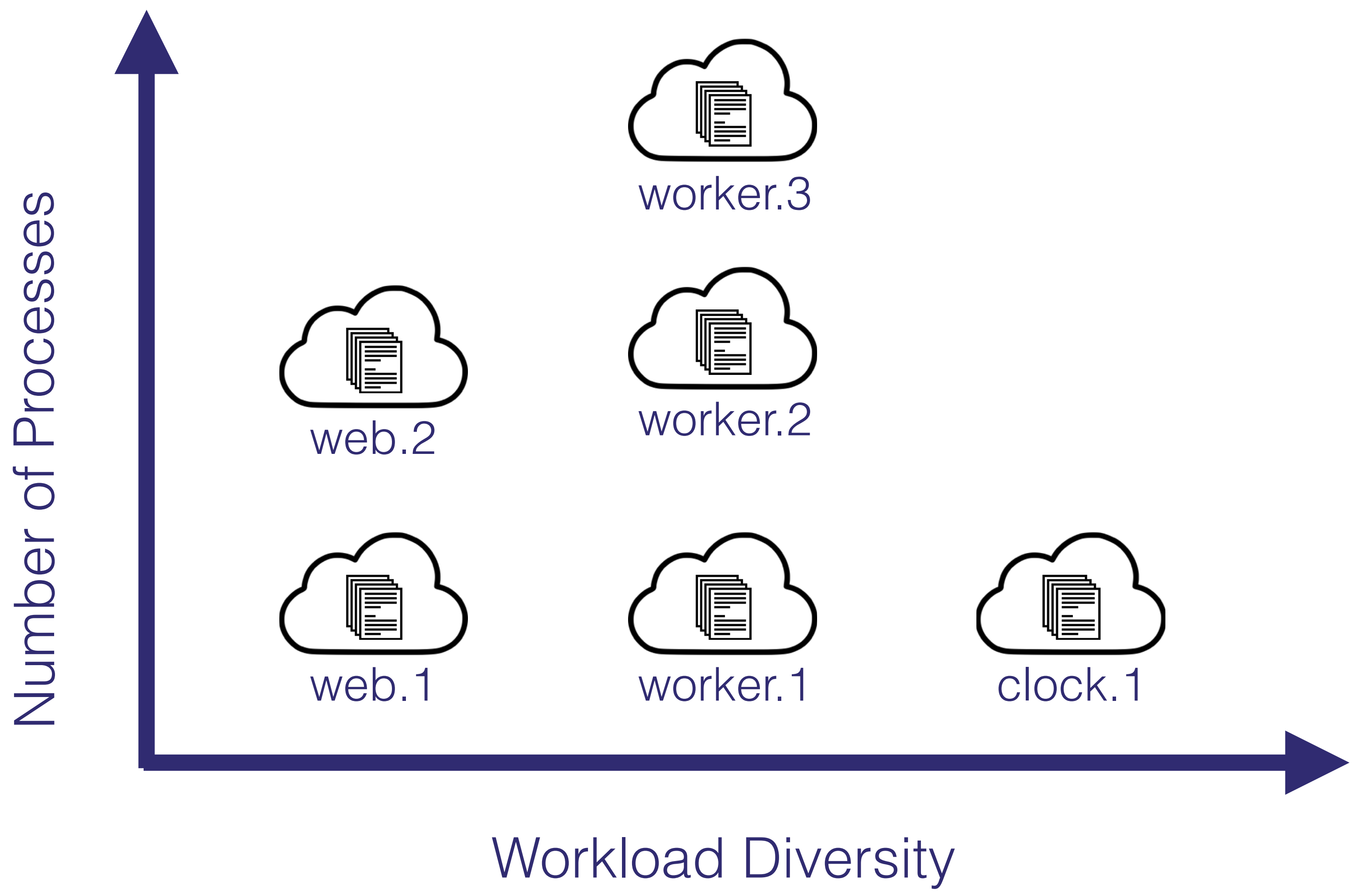
- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port Binding
- Concurrency
- Disposability
- Dev/Prod Parity
- Logs
- Admin Processes

`java.util.concurrent`

Relax bro, I got this…

Scale Up

Scale Out



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

12 Factor App

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port Binding
- Concurrency
- Disposability
- Dev/Prod Parity
- Logs
- Admin Processes

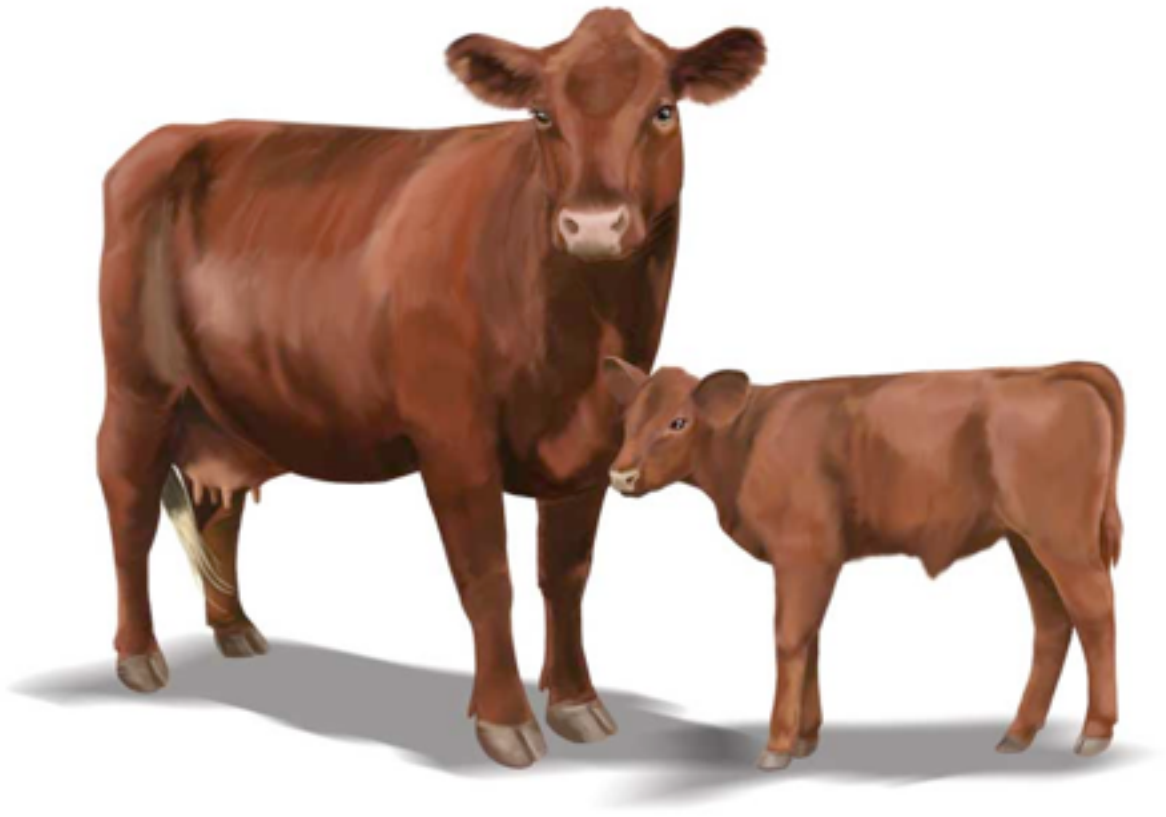
Quick startup

Resilience to failure

Graceful shutdown

Servers are <u>not</u> pets

Servers are cattle



Application Servers are <u>not</u> disposable

Microservices **are** **disposable**

Easy to replace

Easy to modify

Decoupled from
external infrastructure

Microservices

NETFLIX

GILT

12 Factor App

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port Binding
- Concurrency
- Disposability
- Dev/Prod Parity
- Logs
- Admin Processes

dev = stage = prod

dev	**=**	***stage***	**=**	***prod***
sqlite	**≠**	***mysql***	**≠**	***postgres***

dev	**=**	***stage***	**=**	***prod***
sqlite	**≠**	***mysql***	**≠**	***postgres***
postgres	**=**	***postgres***	**=**	***postgres***

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

parity ⇒

<u>reproducibility</u> ⇒

disposability

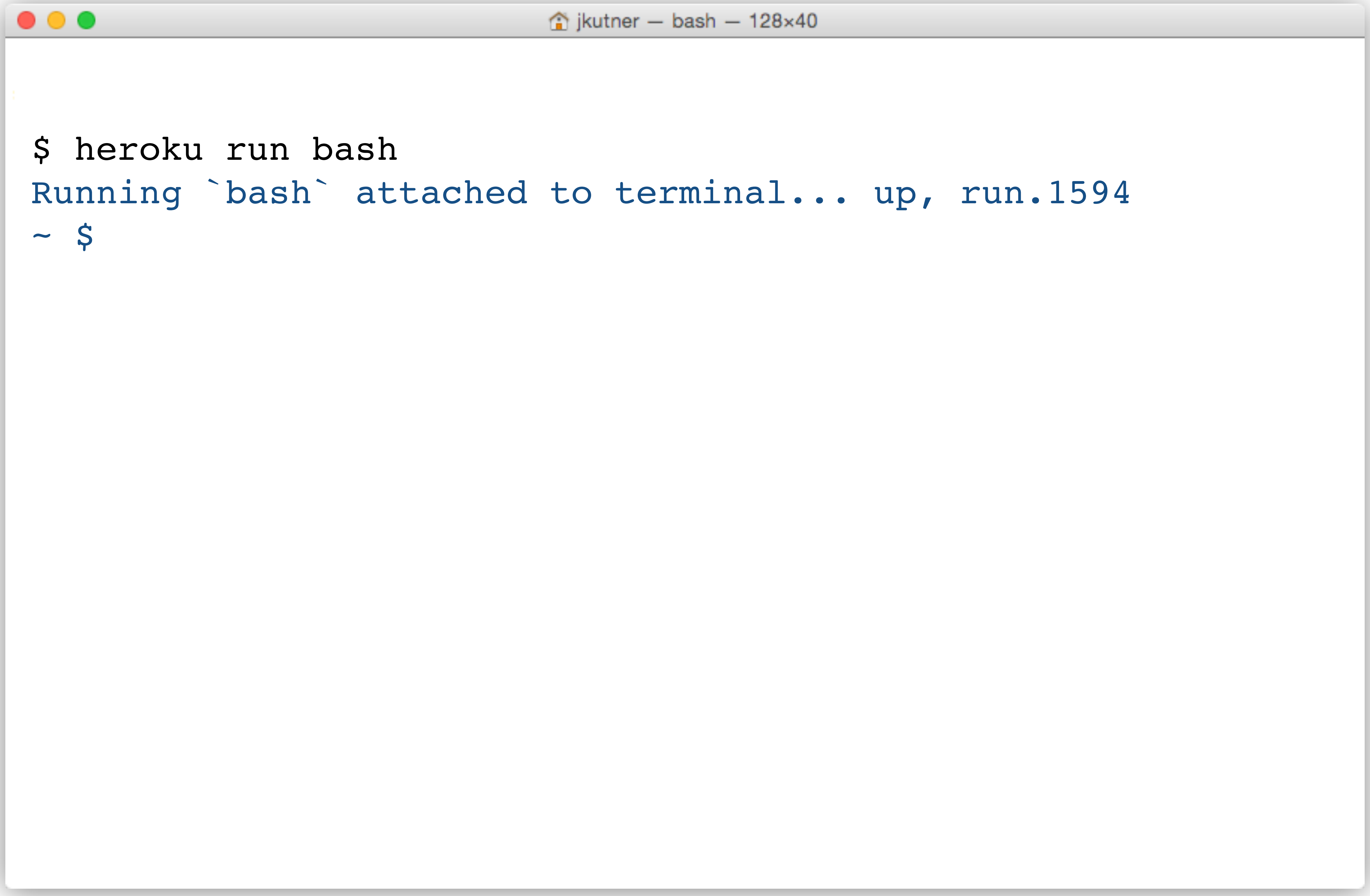
12 Factor App

- Codebase
- Dependencies
- Config
- Backing services
- Build, release, run
- Processes
- Port Binding
- Concurrency
- Disposability
- Dev/Prod Parity
- Logs
- Admin Processes

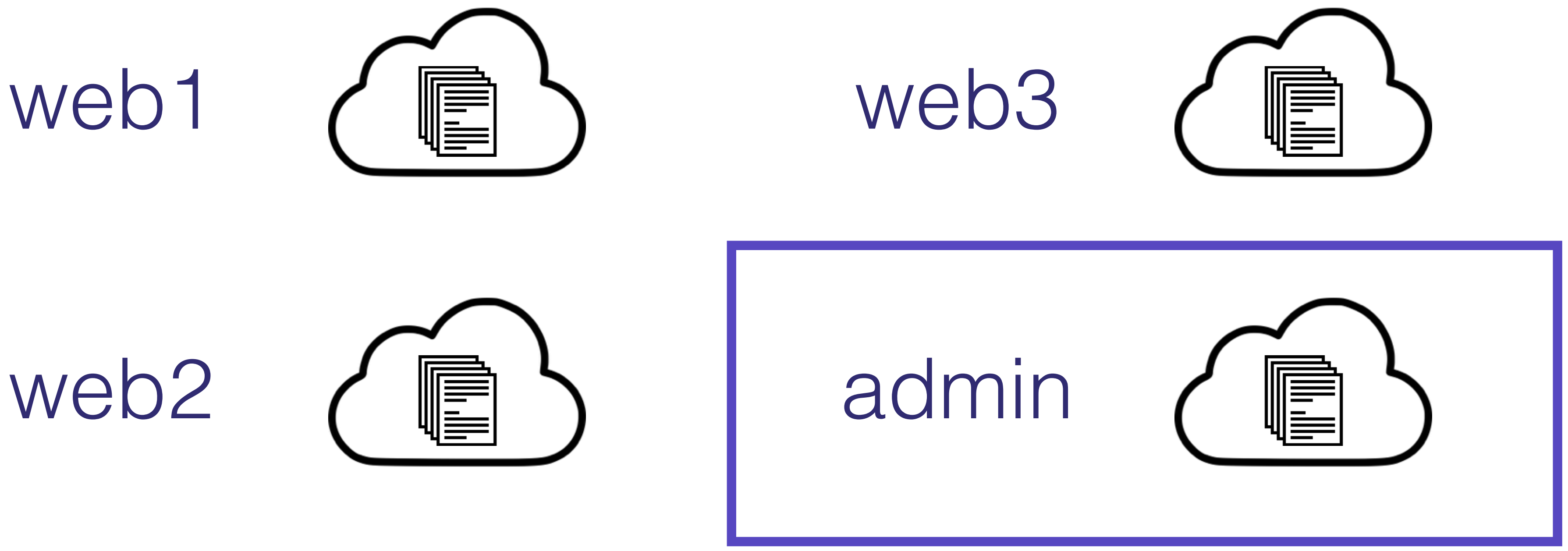
12 Factor App

- 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



```
$ heroku run bash
Running `bash` attached to terminal... up, run.1594
~ $
```


web1
web3
web2
admin

http://12factor.net

http://jkutner.github.io

What next?

1. Create a bintray.com account
2. `?`
3. Remove all passwords
4. `git push heroku master`

Joe Kutner

@codefinger

JVM Platform Owner
@Heroku

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