01 Cloud Native Application이란?

» Cloud Native Application의 개발 원칙

대전제

무상태 (Stateless)	느슨한 결합	언어적으로 , 플랫폼적으로 독립적/자율적	CI/CD
Container 단위	http, Rest	Bind/분산 DB 지향	DevOps, 자동화

지켜야할 12가지!

12 Factors

https://12factor.net/

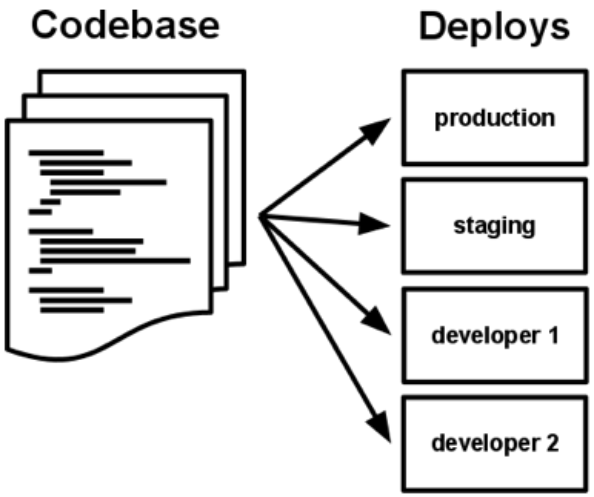
1. Codebase	7. Port binding
2. Dependencies	8. Concurrency
3. Config	9. Disposability
4. Backing Services	10. Dev/prod parity
5. Build, release, run	11. Logs
6. Sateless Processes	12. Admin processes

암기 x, 이해 기반, 주관식 X 객관식으로 O

왜 로그가 필요해, 왜 의존성이 없어야해

01 Cloud Native Application이란?

» 1. 코드 베이스



버전 관리가 가능한 하나의 코드베이스와 이를 이용한 다양한 배포

- Git, Mercurial, Subversion 같은 형상관리 시스템을 사용하여 변화를 추적하여야 한다.
- 코드베이스는 단일 저장소일 수 있고, 루트 커밋을 공유하는 여러 저장소 일 수 있다.
- 코드베이스와 앱 사이에는 1:1 관계가 성립하여야 한다. (micro service별 관리)
- Application의 코드베이스는 한 개여야 하지만 앱 배포는 여러 개가 될 수 있다. (DEV, STG, OPERATION 등)

출처: https://12factor.net/

01 Cloud Native Application이란?

» 2. 종속성

Saas에서 테넌트 레벨 3
코드를 건들지 않고 설정하는 것 처럼.

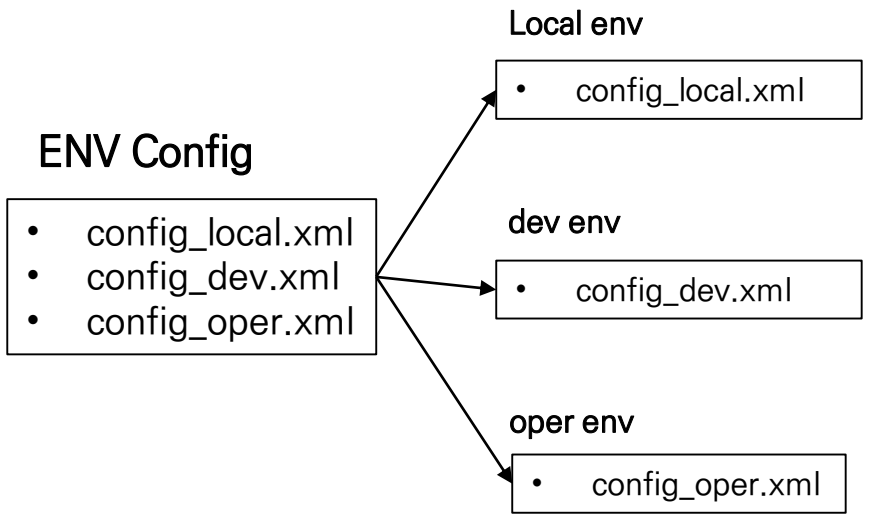
```
dependencies {
    // Spring Boot
    implementation "org.springframework.boot:spring-boot-starter-web"
    implementation "org.springframework.boot:spring-boot-starter-actuator"
    implementation "org.springframework.boot:spring-boot-starter-data-jpa"
    implementation "org.springframework.boot:spring-boot-starter-data-
mongodb"
    implementation "org.springframework.boot:spring-boot-starter-data-redis"
    implementation "org.springframework.boot:spring-boot-starter-validation"

    // Java CfEnv
    implementation "io.pivotal.cfenv:java-cfenv-boot:${javaCfEnvVersion}"

    // JPA Persistence
    runtimeOnly "org.apache.commons:commons-pool2:2.6.0"
    runtimeOnly "com.h2database:h2"
    runtimeOnly "mysql:mysql-connector-java"
    runtimeOnly "org.postgresql:postgresql"
    runtimeOnly "com.microsoft.sqlserver:mssql-jdbc"

// Webjars
    implementation "org.webjars:bootstrap:3.1.1"
    implementation "org.webjars:angularjs:1.2.16"
    implementation "org.webjars:angular-ui:0.4.0-2"
    implementation "org.webjars:angular-ui-bootstrap:0.10.0-1"
    implementation "org.webjars:jquery:2.1.0-2"

    // Oracle - uncomment one of the following after placing driver in ./libs
    // compile files('libs/ojdbc8.jar')
    // compile files('libs/ojdbc7.jar')

    // Testing
    testImplementation "junit:junit"
    testImplementation "org.springframework.boot:spring-boot-starter-test"
}
```

명시적으로 선언되고 분리 된 종속성

- Application은 특정 패키지에 암묵적으로 존재하는 것에 의존 할 수 없음.
- 모든 종속성들은 종속성 선언을 이용하여 모든 종속성을 완전하고 엄격하게 선언
- 특정 시스템에 종속되어 있는 툴(ex. Curl)의 사용을 허용하지 않음.
- 신규 참여자는 Application의 코드베이스를 개발 머신에 체크아웃 하고, 런타임과 종속성 매니저만 설치하여 빌드 할 수 있어야 함.

출처: https://12factor.net/

» 3. 설정



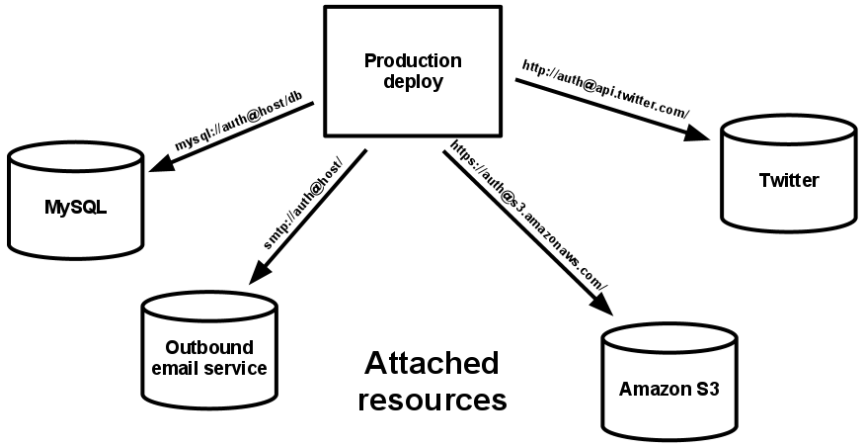
환경에 저장 된 설정의 이용

- Application 설정은 배포마다 달라질 수 있는 모든것을 의미
  (데이터베이스, Memcached 등 백엔드 서비스들의 리소스 핸들과 Amazon S3이나 트위터 등 외부서비스 인증 정보 등)
- 설정 정보는 Application 소스에 상수로 저장되지 않고 반드시 분리되어 관리
- 환경 변수에 의해 이용되는 설정정보를 변경 할 수 있음.

출처: https://12factor.net/

» 4. 백엔드 서비스

서비스에
종속성 가지면 안돼

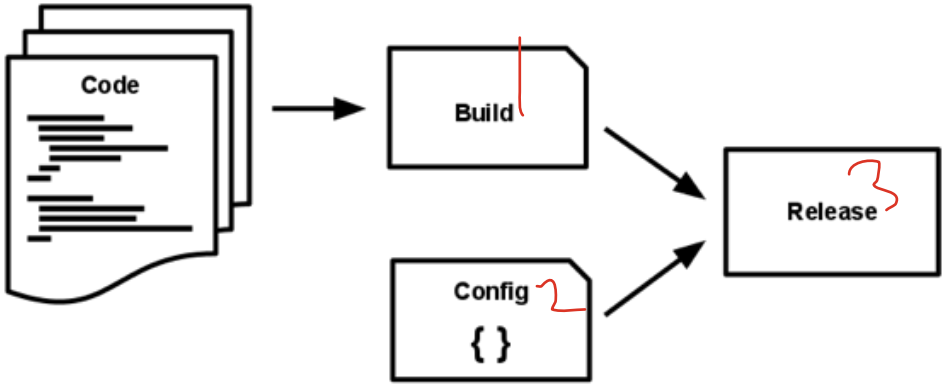


백엔드 서비스를 연결 된 리소스로 취급

- 백엔드 서비스는 Application 정상 작동 중 네트워크를 이용하여 사용되는 모든 서비스를 의미
- 로컬 서비스와 서드파티 서비스를 구별하지 않으며 설정에 있는 URL 또는 다른 로케이터와 인증정보를 사용하여 접근
- Application 코드를 수정하지 않고 백엔드 서비스 전환이 가능하여야 함

출처: https://12factor.net/

» 5. 빌드, 릴리즈, 실행

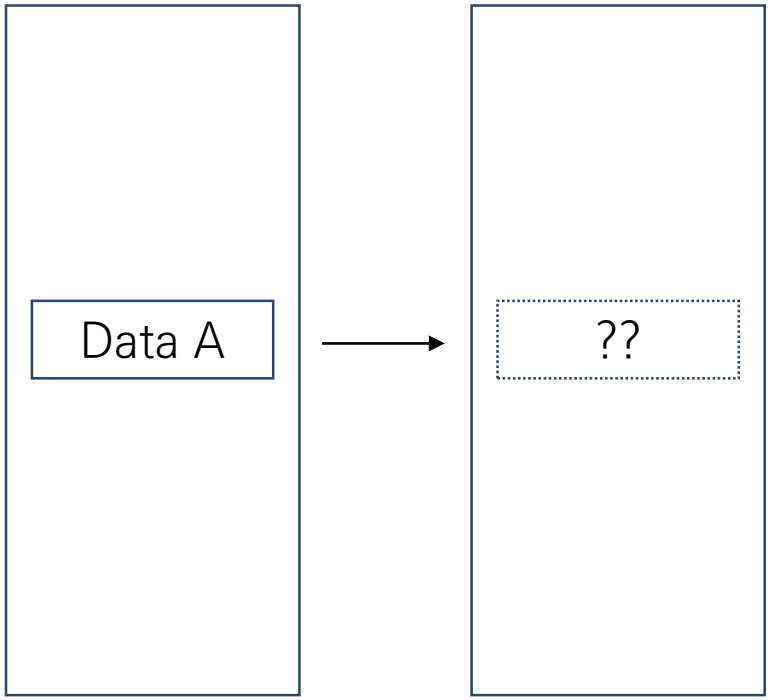


철저하게 분리된 빌드와 실행 단계

- 코드베이스는 3단계를 거쳐 배포로 변환
- 빌드 단계는 코드 저장소를 빌드라는 실행가능한 번들로 변환
- 릴리즈는 빌드 단계에서 만들어진 빌드와 배포의 현재 설정을 결합
- 실행에서는 선택된 릴리즈에 대한 애플리케이션 프로세스의 집합을 시작하여 실행

출처: https://12factor.net/

» 6. 프로세스



애플리케이션을 하나 혹은 여러 개의 무상태 프로세스로 실행

- 실행 환경은 하나 이상의 프로세스로 실행
- 캐시등을 위한 짧은 단일 프로세스에서의 데이터 저장을 제외한 저장될 필요가 있는 모든 데이터는 데이터베이스같은 백엔드 서비스에 저장
- 매 실행시마다 모든 로컬의 상태를 제거

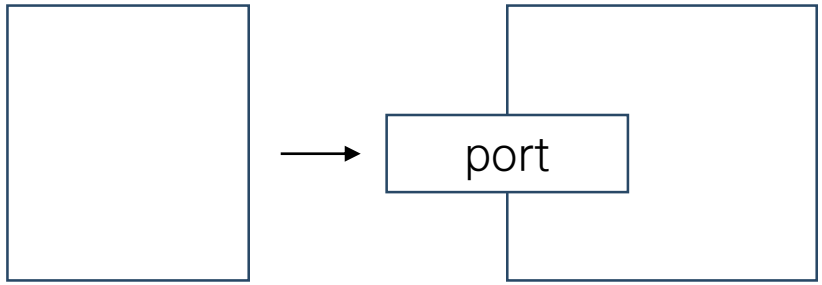
컨테이너간의 종속성이 없이
(결합도 가 떨어진)

단독으로 컨테이너 만으로도 실행 간능한 상태
==
무상태 프로세스.

각각이 데이터베이스에 연결될 수 있어야함

출처: https://12factor.net/

» 7. 포트 바인딩



포트 바인딩을 사용한 서비스 공개

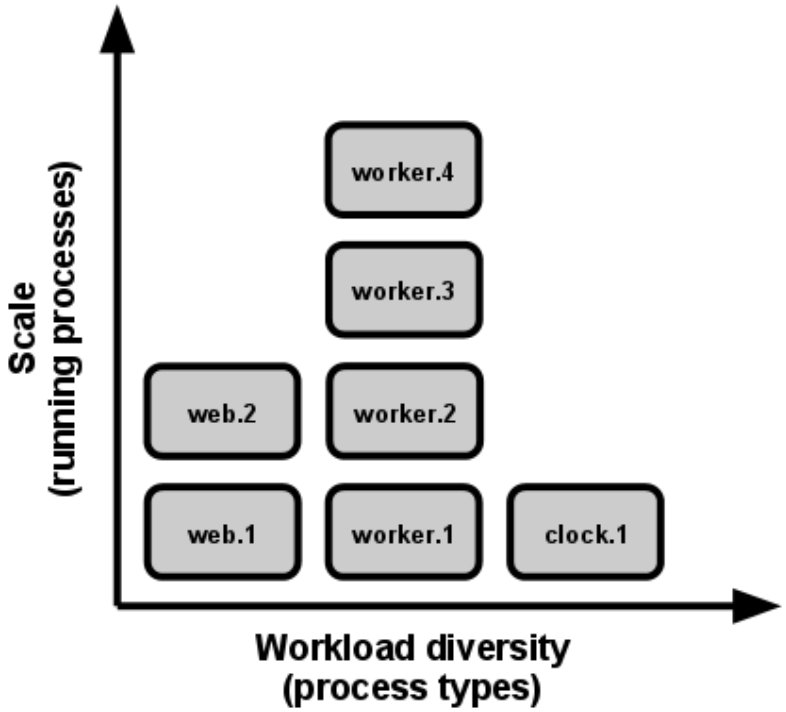
- 완전히 독립적이며 실행환경의 런타임 인젝션에 의존하지 않음
- 특정 포트를 바인딩하여 서비스를 공개하며 해당 포트로의 요청을 처리
- 다른 애플리케이션에 설정을 통해 백엔드 서비스로 등록될 수 있음

필요할 때 연결되어야하므로
각자가 포트를 갖고 있어야함

출처: https://12factor.net/

01 Cloud Native Application이란?

» 8. 동시성



확장.

부하 분산(로드밸런싱) 방법
Horizontal
vertical

프로세스 모델을 통한 확장

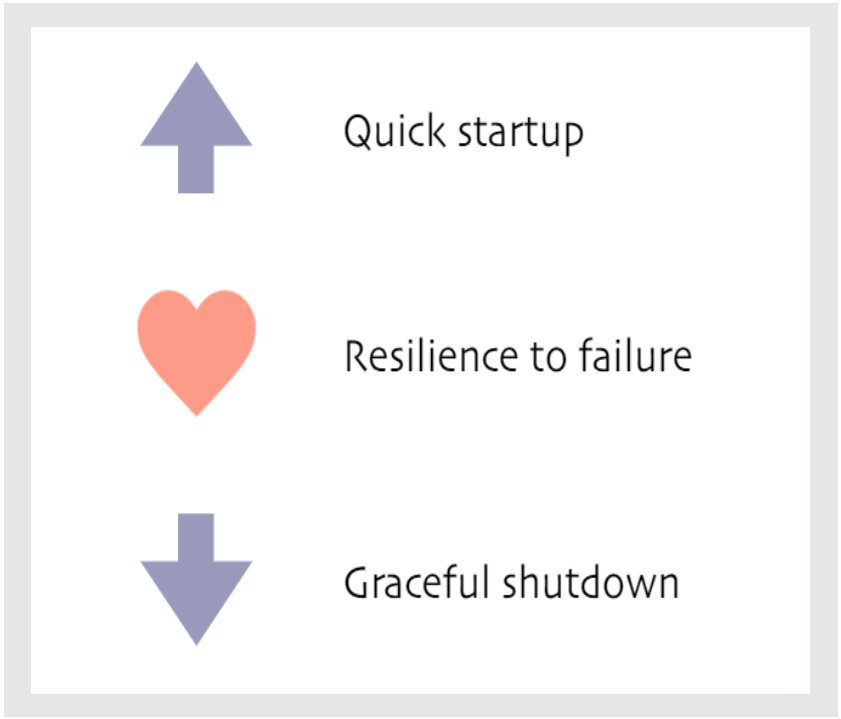
- 동시에 다양한 작업 부하를 처리 할 수 있도록 설계
- 내부 Thread나 Event를 통한 동시 처리 작업
- 아무것도 공유하지 않은 상태에서 수평적 확장이 가능해야 함.

KAFKA 오픈소스 활용

출처: https://12factor.net/

01 Cloud Native Application이란?

» 9. 폐기 가능



문제없이!

문제를 일으키지 않고 종료

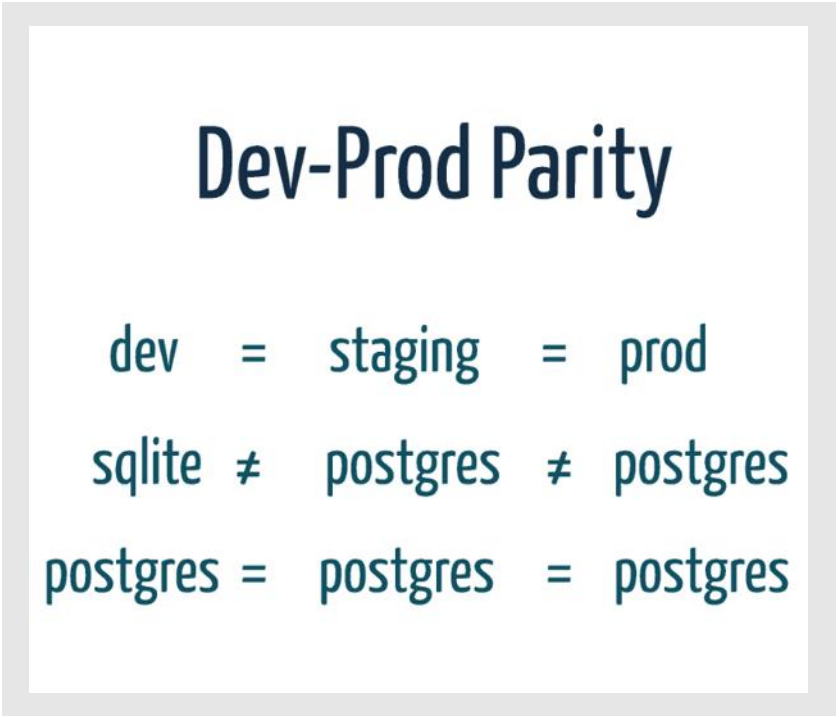
빠른 시작과 그레이스풀 셧다운을 통한 안정성 극대화

- 프로세스는 바로 시작하거나 종료 할 수 있도록 설계
- 시작 시간을 최소화하여 릴리즈 작업과 확장이 민첩하게 이루어질 수 있음
- 그레이스풀 셧다운을 통해 처리중인 작업을 큐로 되돌리는 방법으로 구현
- 갑작스러운 셧다운에도 작업을 큐로 되돌릴 수 있도록 진행

출처: https://12factor.net/

01 Cloud Native Application이란?

» 10. Dev/prod 일치



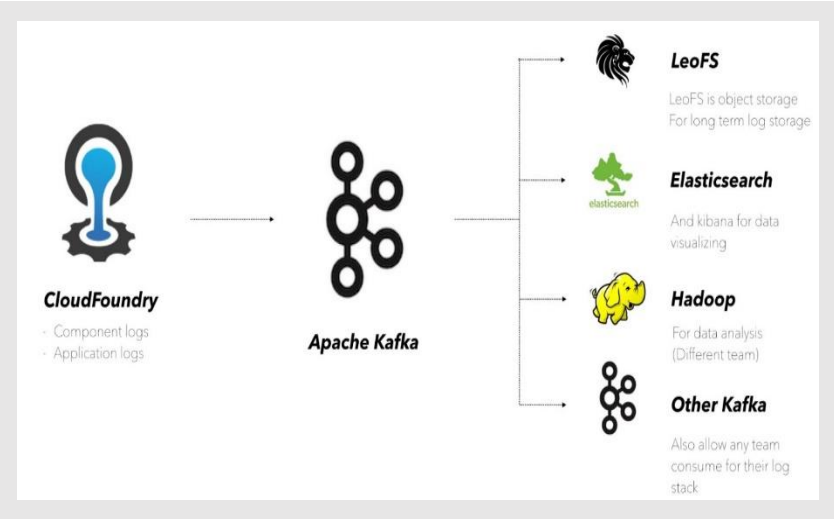
개발/운영 환경을 최대한 비슷하게 유지

- 개발환경과 운영환경의 차이를 작게 유지하여 지속적인 배포 유지
- 개발 환경과 운영 환경의 백엔드 서비스를 동일하게 유지하여 애플리케이션이 서로 다른 환경에서 백엔드 서비스의 차이로 오류가 나지 않도록 유지

출처: https://12factor.net/

Cloud Native Application이란?

» 11. 로그



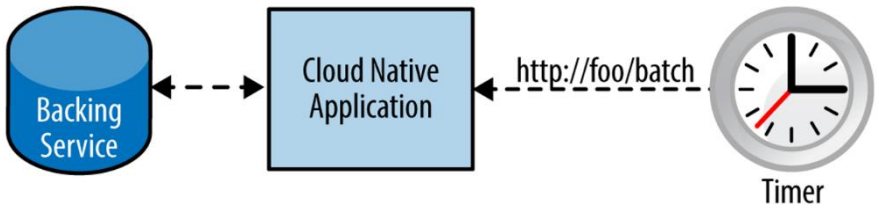
로그를 이벤트 스트림 취급

- 로그는 모든 실행중인 프로세스와 백그라운드 서비스의 아웃풋 스트림
- 애플리케이션은 아웃풋 스트림의 전달이나 저장에 관여하지 않음
- 또한 로그 스트림은 로그 분석 시스템과 범용 데이터 보관소등에 보내져 처리 가능

로그만을 저장하는 공간에
실시간으로
어떤 곳에서 어떤 일이 일어났는지
계속 쌓아야함.

출처: https://12factor.net/

» 12. Admin 프로세스



Admin/maintenance 작업을 일회성 프로세스로 실행

- 데이터베이스 마이그레이션 등 일회성 관리나 유지보수 작업을 위한 프로세스를 일회성 admin 프로세스로 수행
- 모든 프로세스와 동일한 코드베이스와 설정 사용
- 원격 수행이 가능하게끔 설정

관리자/일반 사용자 모드 나눠야함!

다 지킬 순 없다. 노력하는거지

출처: https://12factor.net/