Gitlab CI, Verzovanie appky

Projekt 1

Čistý Git

- Máme veľký projekt
- Niekoľko devs, veľa MRs, veľa vetiev
- Tím líd to ma merdžovať
  - Najprv musí pozrieť kód
  - Čeknúť, či kvalita a čitateľnosť sú OK
  - Spustiť automatické testy
  - Niekam zavesiť "binárky"
  - ... zblázni sa

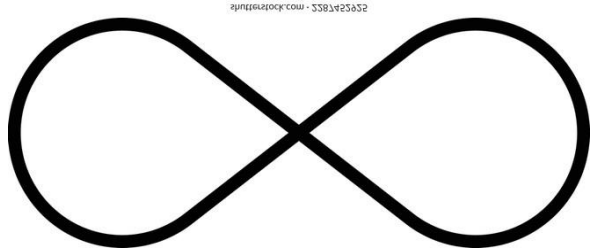
Čistý Git

- Máme veľký projekt
- Niekoľko devs, veľa MRs, veľa vetiev
- Tím líd to ma merdžovať
  - Najprv musí pozrie
  - Čeknúť, či kvalit čitateľnosť sú OK
  - Spustiť automatické
  - Niekam zavesit
  - ... zblázni sa

Linter

JUnit

Nepretržitá integrácia



- CI – Continuous Integration
- Sada úloh, ktoré sa vykonajú pri každom *push*-i kódu
  - Zkompiluj kód
  - Spusti linter
  - Spusti testy
  - Nahlás chyby
- Ak hociktorá úloha padne
  - Zakážeme mergovať, dokiaľ kód neopravíme

Nástroje

- Jenkins
- GitLab CI/CD
  - CI je natívne súčasťou GitLabu
- GitHub Actions
  - CI je natívne súčasťou GitHubu
- CircleCI
- TeamCity
- Travis CI

Súbor .gitlab-ci.yml (v koreni projektu)

```yaml
stages:
- lint
- test
- build

lint-java:
  stage: lint
  script:
    - echo "Running lint stage"

test-java:
  stage: test
  script:
    - echo "Running test stage"

build-java:
  stage: build
  script:
    - echo "Running build stage"
```

Btw... Gitlab Runner

- Gitlab sám nevie spúšťať CI súbory
- Každý projekt/groupa potrebuje tzv. Runner
- To je (typicky) Linux démon, ktorý beží na inom stroji ako Gitlab
- Registrácia runnera je triviálna
  - V grupe/projekte -> Settings -> CI/CD -> Runners -> Create project runner
  - Nainštalujte gitlab-runner na stroji kde je Docker
  - Registrujte ho pomocou tokenu

Linter

- Je skript/program, ktorý kontroluje
  - Kvalitu kódu
    - Napr. dĺžku metód
    - Cyklomatickú zložitosť (aka crapiness kódu - vnorené if-y, cykly atď.)
    - Ignorovanie výnimiek
    - …
  - Bezpečnosť kódu
    - SQL Injection
    - Path Traversal
    - Ukladanie súborov v appke s povolením 777
    - …

Poďme si vytvoriť demo appku

- V IntelliJ IDEA Ultimate, alebo cez https://start.spring.io/ si vytvorme nový *Spring Boot Project*
- ->File/New/Project/Spring Initializr
  - Language: Java
  - Type: Maven
  - JDK: cesta k JDK 25, alebo nechajte IDE stiahnuť
  - Java: 25
  - Packaging: Jar

- Klikneme na *Next*

Komponenty

- V okne *Dependencies* si vyklikáme
  - Lombok
  - Spring Web
  - Spring Data JPA
  - MySQL Driver

- Klikneme dole na *Create*

Java linter

checkstyle-suppressions.xml

```
<?xml version="1.0"?>
<!DOCTYPE suppressions PUBLIC
"-//Checkstyle//DTD SuppressionFilter Configuration 1.2//EN"
"https://checkstyle.org/dtds/suppressions_1_2.dtd">
<suppressions>
<suppress checks="FinalClass" files="EntranceApplication\.java"/>
</suppressions>
```

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-checkstyle-plugin</artifactId>
  <version>3.6.0</version>
  <dependencies>
    <dependency>
      <groupId>com.puppycrawl.tools</groupId>
      <artifactId>checkstyle</artifactId>
      <version>13.2.0</version>
    </dependency>
  </dependencies>
  <configuration>
    <consoleOutput>true</consoleOutput>
    <failsOnError>true</failsOnError>
    <violationSeverity>warning</violationSeverity>
  </configuration>
  <executions>
    <execution>
      <id>checkstyle</id>
      <phase>validate</phase>
      <goals>
        <goal>check</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

.gitlab-ci.yml

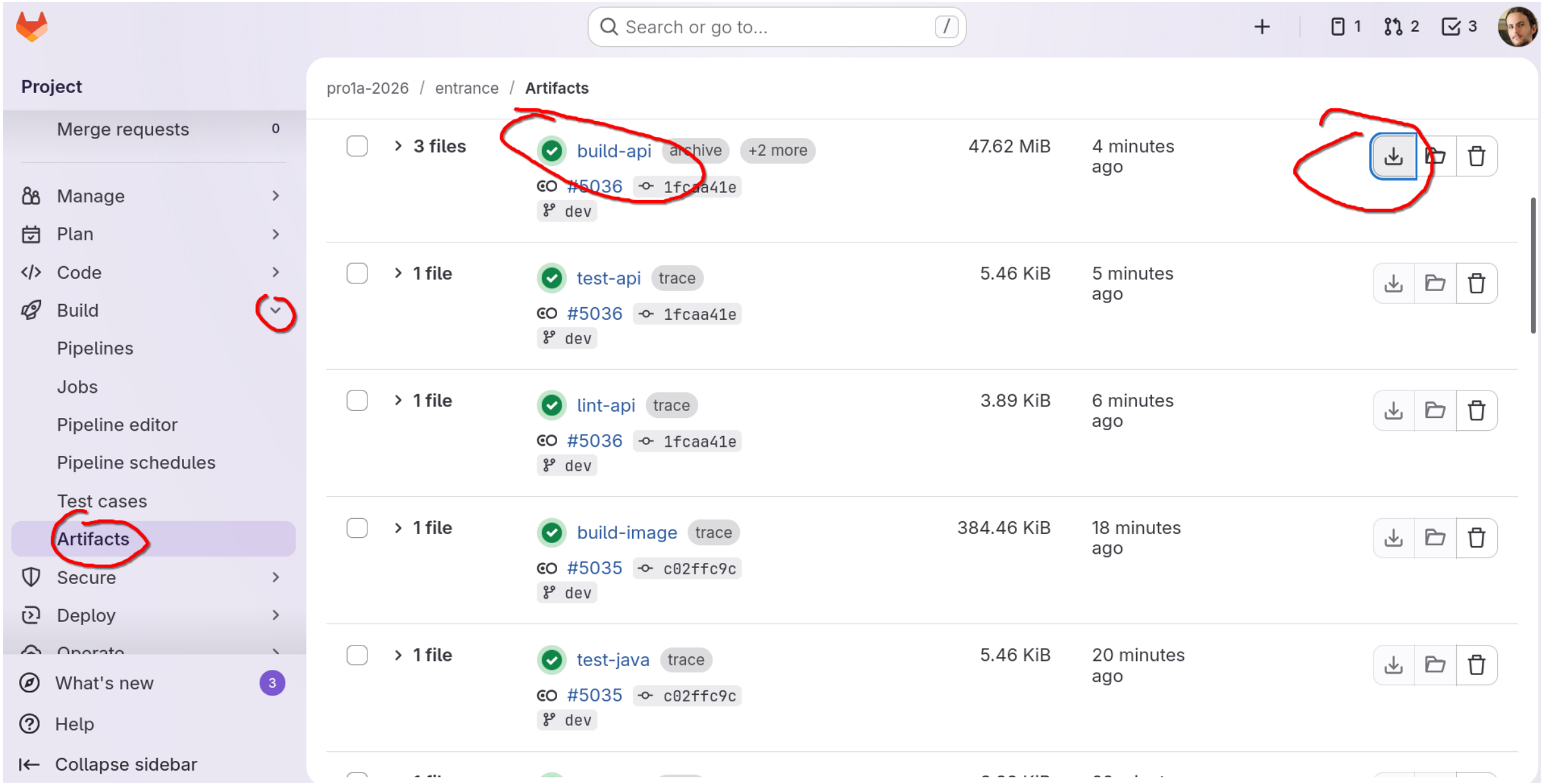
```yaml
image: maven:3.9.12-eclipse-temurin-25
...
variables:
  MAVEN_OPTS: "-Dmaven.repo.local=.m2/repository" # Use this path for m2 libs
  MAVEN_CLI_OPTS: "-B -ntp" # No interactive prompts, no progress bar

# Cache this path between pipelines
cache:
  paths:
    - .m2/repository

lint-java:
  stage: lint
  script:
    - mvn $MAVEN_CLI_OPTS checkstyle:check

test-java:
  stage: test
  script:
    - mvn $MAVEN_CLI_OPTS test
```

CI pre Java build

.gitlab-ci.yml

```
build-api:
  stage: build
  script:
    - mvn $MAVEN_CLI_OPTS -DskipTests package
  artifacts:
    paths:
      - target/entrance-0.0.1-SNAPSHOT.jar
```

Vieme si stiahnuť "binárku" a spustiť
java -jar target/entrance-0.0.1-SNAPSHOT.jar


Search or go to... /
1
2
3
Project
Merge requests 0
Manage
Plan
Code
Build
Pipelines
Jobs
Pipeline editor
Pipeline schedules
Test cases
Artifacts
Secure
Deploy
Operate
What's new 3
Help
Collapse sidebar
pro1a-2026 / entrance / Artifacts
3 files
build-api archive +2 more
#5036 1fcaa41e
dev
47.62 MiB
4 minutes ago
1 file
test-api trace
#5036 1fcaa41e
dev
5.46 KiB
5 minutes ago
1 file
lint-api trace
#5036 1fcaa41e
dev
3.89 KiB
6 minutes ago
1 file
build-image trace
#5035 c02ffc9c
dev
384.46 KiB
18 minutes ago
1 file
test-java trace
#5035 c02ffc9c
dev
5.46 KiB
20 minutes ago

Verzia Java appky

- V pom.xml

```
<version>0.0.1-SNAPSHOT</version>
```

- Pri spustení

```
~/IdeaProjects/entrance  git p02 +5 !7 ?1  java -jar target/entrance-0.0.1-SNAPSHOT.jar
                         ✓ 22s

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v4.0.2)

2026-02-17T20:29:41.856+01:00  INFO 395496 --- [entrance] [           main] s.upjs.ics.entrance.EntranceApplication   : Starting EntranceApplication v0.0.1-SNAPSHOT using Java 25.0.2 with
 PID 395496 (/home/vildibald/IdeaProjects/entrance/target/entrance-0.0.1-SNAPSHOT.jar started by vildibald in /home/vildibald/IdeaProjects/entrance)
```


so what?

Verzovanie

- Každý "artefakt" našej appky je unikátny
  - Má svoje fíčurky, svoje bugy

- Vývojári generujú strašne veľa artefaktov
- Zákazníci môžu mať rôzne verzie
- Potrebujeme ich vedieť od seba rozlíšiť

Verzovanie

```
<version>0.0.1-SNAPSHOT</version>
```

- Všetci ale majú "tú istú" verziu

Čo vlastne chceme?


Executive Order 66
Devs budú meniť verziu v pom.xml pri každom
git push
imgflip.com

Toto nebude fungovať

Verzia programu

- Musíme meniť stále ručne + potrebujeme mať unikátne verzie
  - Kto si to bude všetko pamatať?
- Verzia nám vlastne **identifikuje** zostavený program
  - Zostavený program = "binárka"/"artefakt"

Verzia je ID?!

- Verzia je vlastne forma ID-čka
  - Budeme nejak generovať umelé verzie? (autoincrement)
  - Náhodné verzie? (UUID)
- Verzia a ID sú unikátne identifikátory
  - Dve entity s rovnakým ID sú totožné
  - Dve binárky z rovnakého zdrojáku sú totožné
    - (ak sú zostavené rovnakým prekladačom s rovnakým nastavením na rovnakú platformu)

Verzia vs ID

- Verzia však nie je ako ID v databázach
  - S ID pracuje iba program
  - Pre ľudí je ID typicky nepoužiteľný údaj
  - **Z ID nevieme vyčítať nič o entite**

Verzia vs ID

- Verzia by stále mala dať napohľad užitočné informácie
  - Je nová verzia spätne kompatibilná?
  - Je upgrade triviálny?
  - Kedy bola vydaná?
  - Viem dohľadať z verzie zdroják?
  - ...

Jednoduché technické verzovanie

.gitlab-ci.yml

```
...
default:
  before_script:
    - export VERSION="$(date +%Y%m%d.%H%M%S).${CI_COMMIT_SHORT_SHA}"
    - echo "Building version ${VERSION}"
...
```

Dátum a čas v tvare YYYYMMDD.HHmmss

Heš komitu, ktorý nám dáva Gitlab ako premennú prostredia

Napr. Entrance **20260217.203345.ea27fd79**

Jednoduché technické verzovanie

.gitlab-ci.yml

Zmeň verziu v pom.xml

```
...
build-api:
  stage: build
  script:
    - mvn $MAVEN_CLI_OPTS versions:set -DnewVersion="${VERSION}" -DgenerateBackupPoms=false
    - mvn $MAVEN_CLI_OPTS -DskipTests package
  artifacts:
    paths:
      - target/*.jar
    exclude:
      - target/*.jar.original
```

Jednoduché technické verzovanie

- Presný formát samozrejme nie je pravidlom.
  - Dajte si to ako sa vám hodí/páči
- Vhodné pre začiatky vývoja
  - Triviálne na implementáciu
  - Vyrieši najpálčivejší problém s verzovaním
    - Každý artefakt je unikátny
- Veľmi vhodný pre "rolling-release" projekty
  - (Steam, hry, menšie SaaS projekty)
  - Keď v produkcii máte iba jednú verziu appky

Sémantické verzovanie

- aka Entrance v3.14.15

v **major** (hlavná verzia) . **minor** (vedľajšia verzia) . **patch** (opravná verzia)

Patch verzia

- Napr. v2.5.**3** vs v2.5.**4**
- Oprava bugov
- Zmeny v kóde, ktoré "nevidno" navonok z UI/API
- Optimalizácie
- Upgrade interných knižníc/nástrojov bez zmeny funkcionality v UI/API

Minor verzia

- Napr. v2.**5**.4 vs v2.**6**.0
- Pridanie novej funkcionality bez zmien/odstránenia starej
- Mierne vylepšenia UI, nové farebné témy

Major verzia

- Napr. v**2**.6.0 vs v**3**.0.0
- Prerobenie/odstránenie nejakej funkcionality
- Prekopanie UI/API

Výhody SV

- Vieme povedať, či upgrade je "bezpečný"
- Pri patch áno
- Pri minor takmer vždy áno
- Pri major veľmi často **nie**

Automatizácia SV

- Ak mergneme vetvu "breaking-"
  - Tak +1 major
- Ak mergneme vetvu "feature-*"
  - tak +1 minor
- Ak mergneme iné vetvy
  - Tak +1 patch
- Urobíme aj "git tag **${VERSION}** && git push --tags"

Nevýhody SV

- Náročné na mentalitu a disciplínu
- 99% projektov nerobí SV dobre
  - neboja sa priznať si to
  - ale nič s tým nerobia
- Aj Linux kernel už roky používa iba nejaké pseudo-SV

Akou verziou začať?

- Akou verziou začneme?
- Teoreticky mali by sme od v1.0.0
- Ale predsa **v1**.0.0 by mala byť prvá hotová verzia, ktorú dodáme klientom

Akou verziou začať?

- Idea: Začnime od v0.0.1
  - V0.x.y budú rané interné verzie
- Časom prejdeme na v1.0.0, aby sme odlíšili "rozpracovaný" projekt od hotového

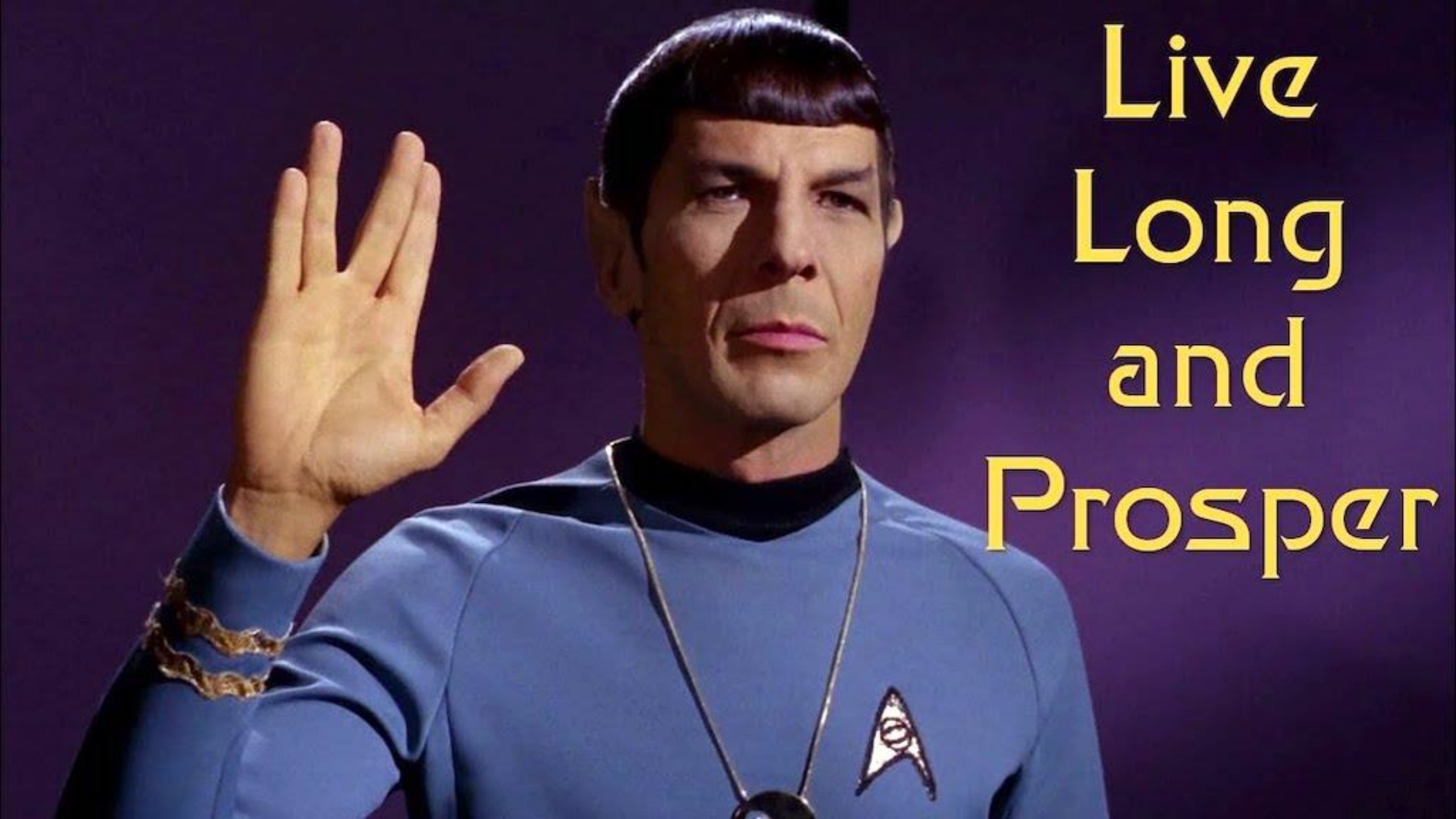

IN THE AGILE PROJECT
NOT.GONNA.HAPPEN
imgflip.com

SV a agilný vývoj

- Pri agilnom vývoji máme už skoré produkčné nasadenia
- Aplikáciu dodávame zákazníkovi už po pár týždňoch/mesiacoch
- Stiera sa hranica medzi "rozpracovanou" (v0.x.y)/"hotovou" (v1.x.y) appkou
- Bežné breaking zmeny
  - Takže teoreticky veľa v2, v3, v4, ...
- Lead devs/PM sa z nejakého dôvodu boja dvíhať major verzie v ranom vývoji
  - "Projekt je starý 2 mesiace a už má verziu v7"
  - Takže breaking zmeny bežne v minor/patch verziách
  - Skončíme s verziami typu v0.108.0, v0.109.0, v0.109.1, v0.109.1-1

Riešenie

- Začnite s jednoduchým, no jasne funkčným polo-umelým "technickým" verzovaním
  - Napr. dátum+čas buildu, názov vetvy, commit hash, ich rôzne kombinácie
  - Určené iba pre vývojárov/testerov/ops
- Sémantické verzovanie robte neskôr bokom
  - Určené pre klientov/používateľov
  - Až sa projekt trocha zastabilizuje
  - Kľudne vo forme samostatného git repozitára
    - kde bude iba .gitlab-ci.yml a pomocné skripty
  - Manuálne spustenie CI v novom repe urobí v hlavnom repe
    - Napr. git tag vX.Y.Z z hlavnej vetvy, urobí release-vX.Y vetvu, ...
    - Urobí z commit správ release notes, ...


Live
Long
and
Prosper