

composer.lock demystified



Nils Adermann

@naderman

Private Packagist

<https://packagist.com>

composer.lock

- Contents
 - all dependencies including transitive dependencies
 - all metadata (name, description, require, autoload, extra, ...)
 - Exact version for every package
 - Download URLs (source, dist, mirrors)
- Purpose
 - Reproducibility across teams, users and servers
 - Isolation of bug reports to code vs. potential dependency breaks
 - Transparency through explicit updating process



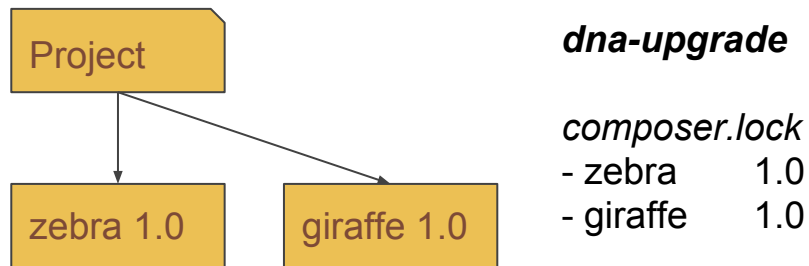
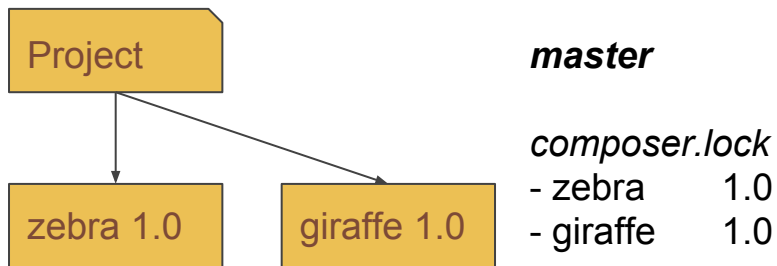
Commit The Lock File

- If you don't
 - composer install without a lock file is a composer update
 - You're not managing your dependencies, they're just doing whatever they want
 - Conflict can randomly occur on install
 - You may not get the same code
- The lock file exists to be committed!

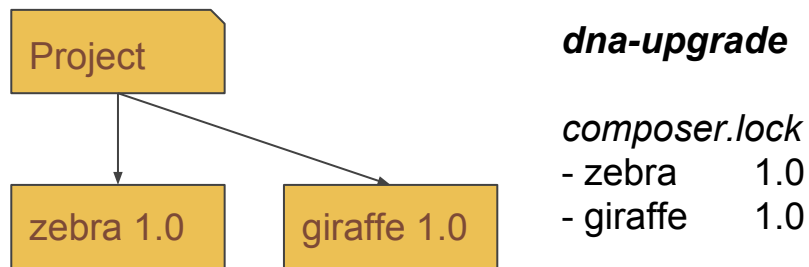
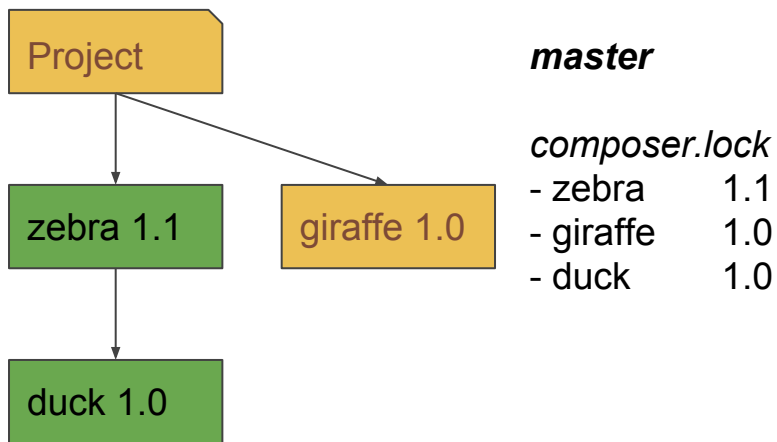


The Lock file will conflict

Day 0: “Initial Commit”



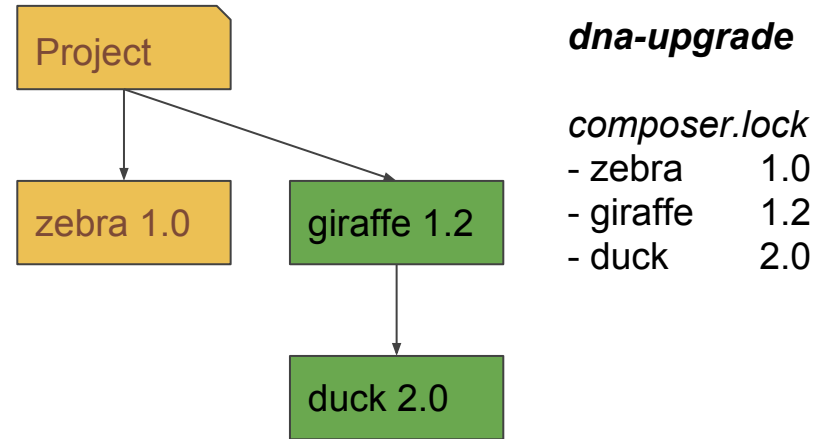
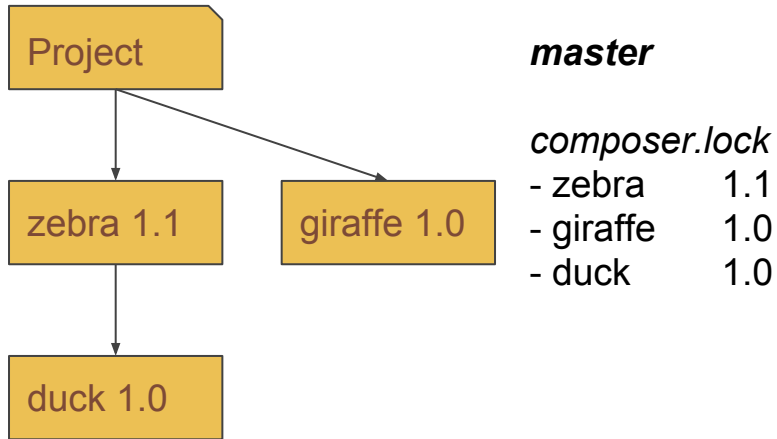
Week 2: Strange new zebras require duck



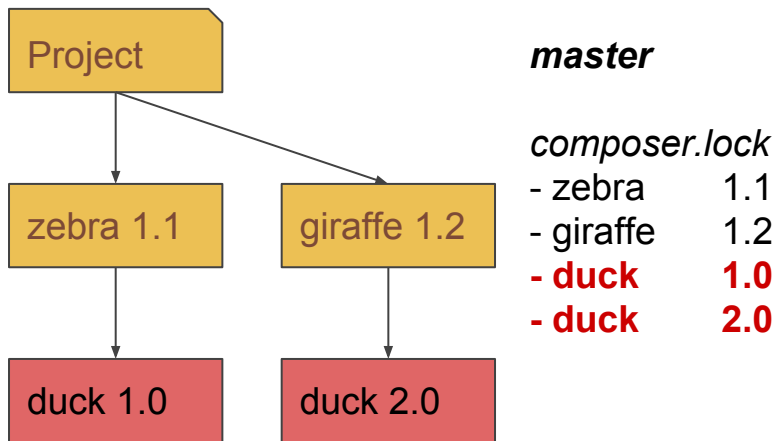


Week 3: Duck 2.0

Week 4: Giraffe evolves to require duck 2.0



Text-based Merge

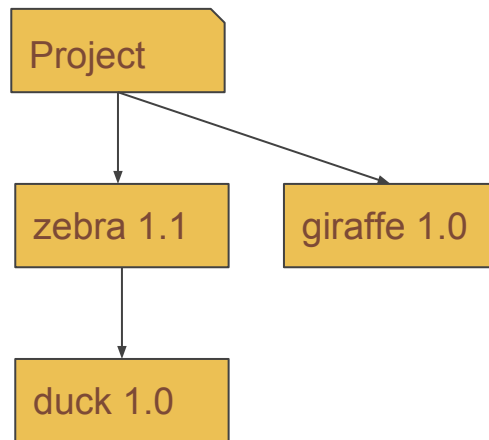


Merge results in invalid dependencies



Reset composer.lock

```
git checkout <refspec> -- composer.lock  
git checkout master -- composer.lock
```



dna-upgrade

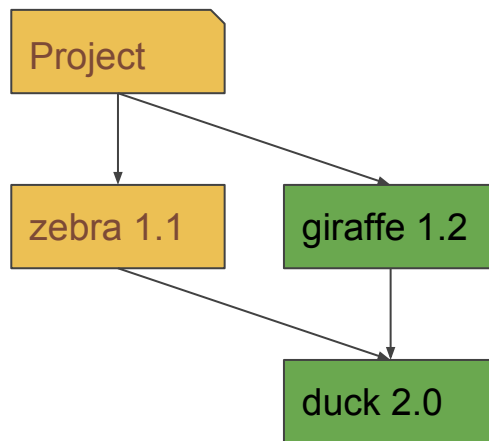
composer.lock

- zebra 1.1
- giraffe 1.0
- duck 1.0



Apply the update again

```
composer update giraffe  
--with-dependencies
```



master

composer.lock

- zebra 1.1
- giraffe 1.2
- duck 2.0



How to resolve lock merge conflicts?

- composer.lock cannot be merged without conflicts
 - contains hash over relevant composer.json values
- `git checkout <refspec> -- composer.lock`
 - `git checkout master -- composer.lock`
- Reapply changes
 - `composer update <list of deps>`