Composer Package Development Setup

Leo Vujanić

Every project is a package

- Adding composer.json to directory - defines package
- Adding a “require” to project - defines package dependencies
- The only difference between your project and a library is that your project is a package without a name
- Giving a “name” property to project - makes it installable (almost)

Packagist.org

- Packagist is the main package repository for Composer, and it is enabled by default. Anything that is published on Packagist is available automatically through Composer.

- Private Packagist allows you to manage your own private Composer repository with per-user authentication, team management and integration in version control systems, but it has it’s price.

Create your first package

- create a new directory that will hold all package files
- create subdirectory “src” - it will contain package code
- create subdirectory “tests” - it will hold package tests
- create composer.json in package root or run composer init
- git init (README, LICENCE etc. or leave it for later)

```
"autoload": {
    "psr-4": {
        "vendorname\\namespace\\": "src/"
    }
}
```

composer.json

```
{
        "name": "leovujanic/server-helper",
        "description": "Server helper collection",
        "type": "library",
        "license": "BSD-3-Clause",
        "authors": [
          {
            "name": "leovujanic",
            "email": "leonard.vujanic@gmail.com"
          }
        ],
        "autoload": {
          "psr-4": {
            "leovujanic\\serverhelper\\": "src/"
          }
        }
}
```

Configure project

- add your package to some project by modifying composer.json
- add your repository as path - (symlink)

```
"repositories": [
      {
         "type": "path",
         "url": "./path/to/package-root"
      },
],
"minimum-stability": "dev"
```

Use your package

- require your package by running “composer require package-name *”
- or add it to require section and install/update
- in PhpStorm you may want to mark directory as excluded

Keep in mind

- all dependencies should be specified in package composer.json in require or require-dev section
- require-dev specifies extensions that are not needed on production (used for testing e.g.)
- if you are developing yii-framework extension, then yii is your dependency
- use only your classes and classes of vendors specified as your dependencies
- php version (syntax >= 7)
- php extensions

```
"require": {
  "php": ">=7.0.2",
  "ext-curl": "*",
}
```

When you are done

- push your code to github, bitbucket
- Create account on packagist.com
- Submit your package by pasting link to git repo
- Read https://packagist.org/about#how-to-update-packages to learn how configure webhooks to auto-update package on new pushes
- Create release respecting the rules and conventions (http://semver.org/)

1.0.0

v1.0.0

1.10.5-RC1

v4.4.4beta2

v2.0.0-alpha

v2.0.4-p1

Use your package

- Specify your package as dependency in some project (keep your development setup for future)
- If you decided not to publish package on packagist, you can configure composer to install it directly from VCS

```
"repositories": [
    {
        "type": "vcs",
        "url": "https://github.com/vendorname/repository-name/",
        "url": "git@bitbucket.org:vendorname/private-repo.git"
    }
]
```

Private Private Packagist

- Composer Satis
- Satis is open source but only a static composer repository generator. It is a bit like an ultra-lightweight, static file-based version of packagist and can be used to host the metadata of your company's private packages
- ideal for packages you want to reuse across your company but don't really want to open-source

Satis install & configure

- composer create-project composer/satis --stability=dev --keep-vcs
- create satis.json

```
{
    "name": "My Repository",
    "homepage": "http://packages.my-domain.org"
    "repositories": [
      { "type": "vcs", "url": "https://github.com/mycompany/privaterepo" },
      { "type": "vcs", "url": "https://github.com/mycompany/privaterepo2" }
    ],
   "require-all": true,
   "require": {
      "company/package": "*",
      "company/package3": "2.0.0"
   }
}
```

Build

- allow composer to access your private repositories by configuring ssh keys or auth.json
- php bin/satis build <configuration file> <build dir> and go to homepage

```
{
  "github-oauth": {
    "github.com": "kjdhsdkjhdajkshdjkashdkjashdjkhasd"
  },
  "bitbucket-oauth": {
    "bitbucket.org": {
      "consumer-key": "sdasdasdasdasdasd",
      "consumer-secret": "3434jk3j4khjghvg3434hg34vghv",
      "access-token": "420i6EUqFv_ZutbzfTErFVJ74v67HG_t48XexhdmskdnsajkndkjasndjasndjkaEGHGEJJE",
      "access-token-expiration": 1508450420
    }
  }
}
```

Use your private packagist

- in main project add repo config to composer.json
- require available packages and install/update

```
"repositories": [
        {
         "type": "composer",
         "url": "http://my-domain.loc"
        }
]
```

Links

- https://packagist.org/
- http://semver.org/
- https://semver.mwl.be/
- https://getcomposer.org/doc/articles/handling-private-packages-with-satis.md
- https://github.com/composer/satis
- https://help.github.com/articles/creating-releases/
- https://confluence.atlassian.com/
- https://getcomposer.org/doc/articles/versions.md

Thank You - Q & A

I am thankful for all those difficult questions in my life, they have shown me exactly what I want to know.