

# Paket- och beroendehantering med Composer

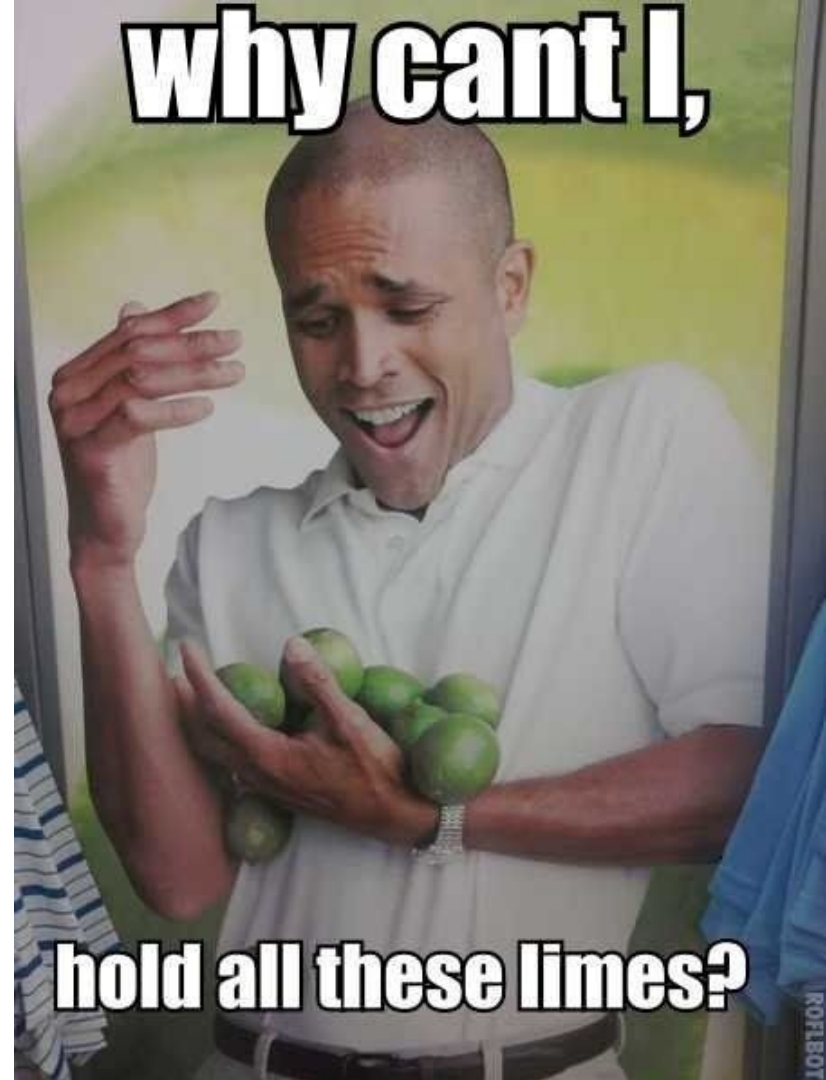
DA288A – Molnbaserade Webbapplikationer

# Dagens agenda

- Vad är paket- och beroendehantering?
  - Pakethantering i operativsystem
  - Beroendehantering i mjukvaruprojekt
- Beroendehantering i PHP
  - Introduktion till Composer
  - Packagist
  - Semantisk versionering
  - Autoloading
- Exempel

## Vårt problem

- Riktiga mjukvaruprojekt är stora
- Riktiga mjukvaruprojekt innehåller oftast externt skriven mjukvara
- Hur hanterar vi detta på ett effektivt sätt?



# Några viktiga termer

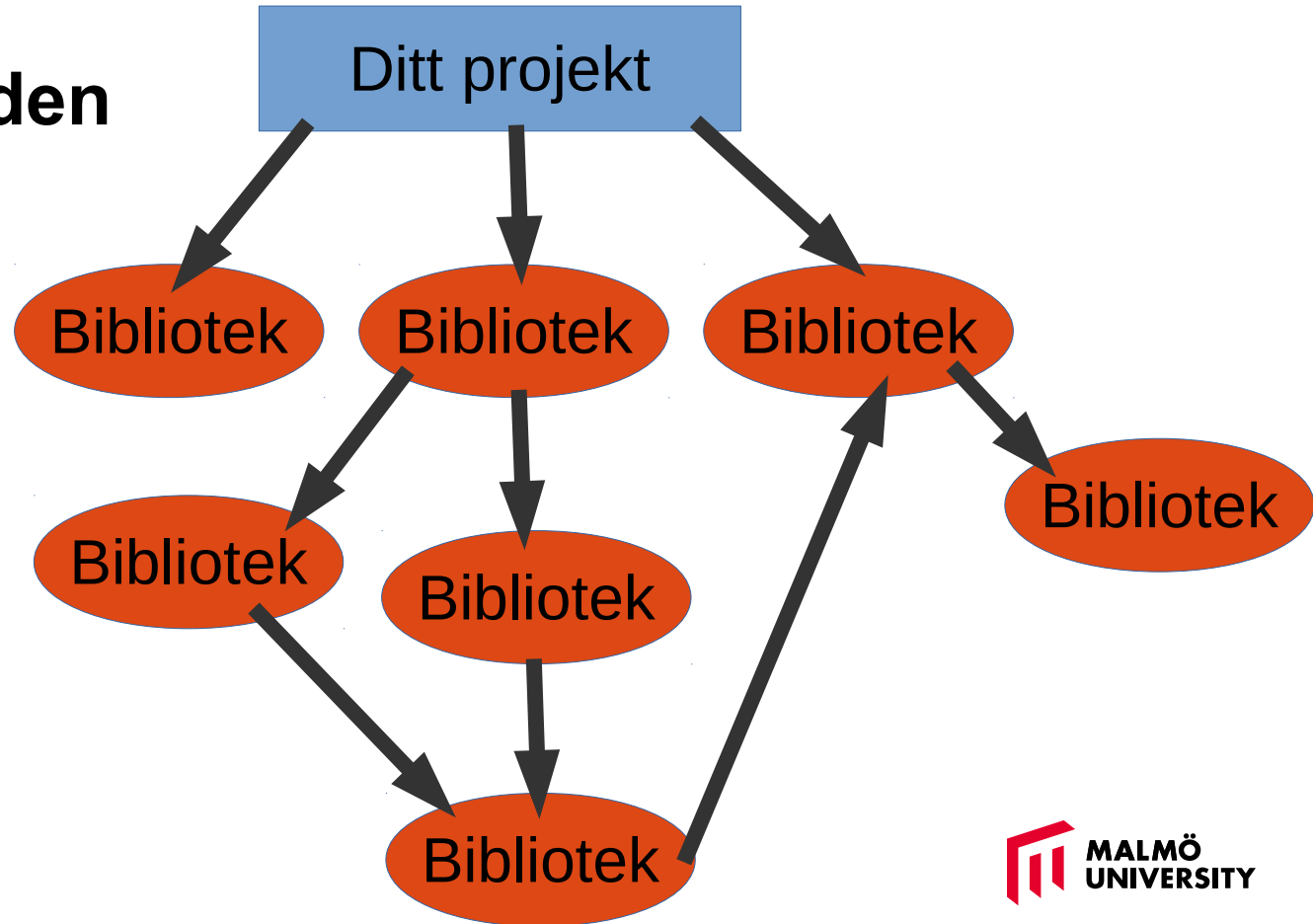
- **Paket:** En samling av filer som levereras tillsammans. Kan vara körbara filer, bibliotek eller andra resurser, så som media, typsnitt eller konfigurationsfiler.
- **Pakethanterare:** En mjukvara som hanterar paket.
- **Repository** (eller **repo**): En plats där paket kan publiceras, sökas efter och laddas ner från.
- **Beroende:** Ett eller flera paket som behövs för att ett specifikt paket ska kunna användas.
- **Beroendekonflikt:** Ett tillstånd som uppstår då två paket beror på samma paket.
- **Version:** En specifik utgåva av ett specifikt paket. Genom att känna till versionsinformation kan en pakethanterare hålla koll på flera versioner av ett paket, vilket minskar risken för beroendekonflikter.

# Beroenden

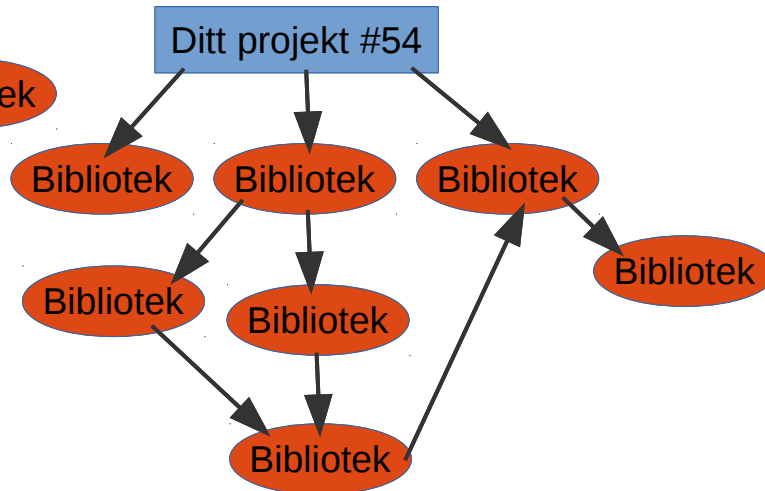
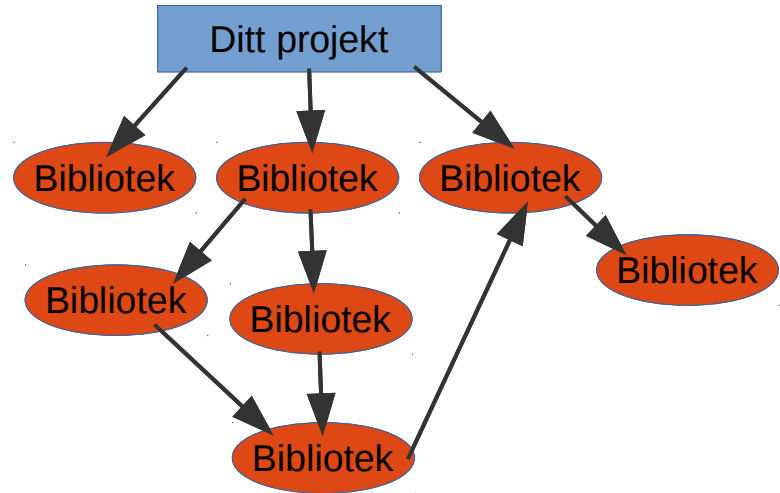
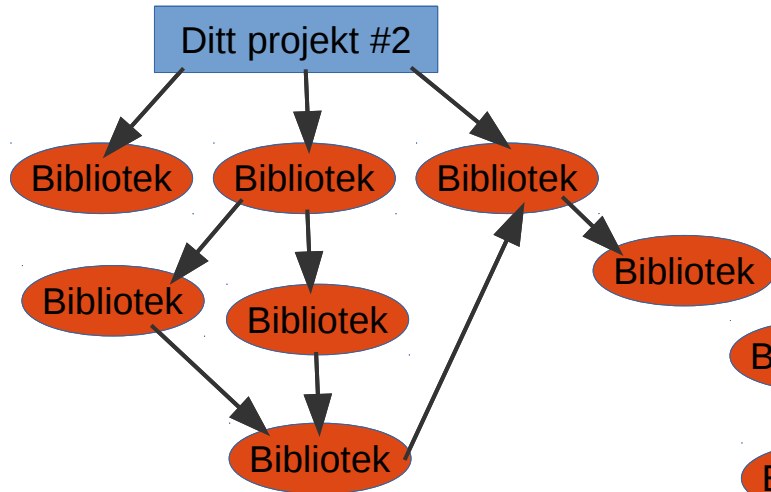
```
graph TD; A[Ditt projekt] --> B1(Bibliotek); A --> B2(Bibliotek); A --> B3(Bibliotek); B1 --> C1(Bibliotek); B2 --> C2(Bibliotek); B3 --> C3(Bibliotek); C1 --> D(Bibliotek); C2 --> D; C3 --> E(Bibliotek);
```

The diagram illustrates a dependency structure. At the top, a blue rectangle labeled "Ditt projekt" (Your project) has three arrows pointing down to three orange ovals, each labeled "Bibliotek" (Library). The middle "Bibliotek" oval has an arrow pointing down to another "Bibliotek" oval. The left "Bibliotek" oval has an arrow pointing down to a "Bibliotek" oval. The right "Bibliotek" oval has an arrow pointing down to a "Bibliotek" oval. The "Bibliotek" oval from the middle has an arrow pointing down to a "Bibliotek" oval. The "Bibliotek" oval from the left has an arrow pointing down to a "Bibliotek" oval. The "Bibliotek" oval from the right has an arrow pointing down to a "Bibliotek" oval. The "Bibliotek" oval from the middle has an arrow pointing down to a "Bibliotek" oval. The "Bibliotek" oval from the left has an arrow pointing down to a "Bibliotek" oval. The "Bibliotek" oval from the right has an arrow pointing down to a "Bibliotek" oval.

MALMÖ UNIVERSITY



# Beroenden



# Pakethantering i operativsystem

Exempel: apt-get i Ubuntu

```
johan@suell: ~  
Arkiv Redigera Visa Sök Terminal Hjälp  
johan@suell:~$ sudo apt-get install unicorn  
Läser paketlistor... Färdig  
Bygger beroendeträd  
Läser tillståndsinformation... Färdig  
The following additional packages will be installed:  
  ruby-kgio ruby-rack ruby-raindrops  
Följande NYA paket kommer att installeras:  
  ruby-kgio ruby-rack ruby-raindrops unicorn  
0 att uppgradera, 4 att nyinstallera, 0 att ta bort och 10 att inte uppgradera.  
Behöver hämta 270 kB arkiv.  
Efter denna åtgärd kommer ytterligare 892 kB utrymme användas på disken.  
Vill du fortsätta? [J/n]
```



# Beroendehantering i mjukvaruprojekt

- Python: pip
- Java: Maven/Gradle
- Javascript: npm
- C#/.NET: NuGet
- **PHP: Composer**

# Varför beroendehantera i mjukvaruprojekt?

- Robustare projekt och mjukvara
- Enklare att starta nya projekt
- Enklare för nya utvecklare att börja arbeta
- Mindre kod i era kod-repositories



Composer: <https://getcomposer.org>  
Lockart, *Modern PHP*. p. 57

# Composer

- De facto-standard för vettig beroendehantering i PHP
- Hanterar beroenden i flera led
- Kan skilja mellan olika versioner av paket
- Sköter installation av paketen åt dig


# Composer – installation

Exempel: Installation av Composer i Ubuntu

# Packagist

**Packagist** The PHP Package Repository

[Browse](#)[Submit](#)[Create account](#)[Sign in](#)



Packagist is the main Composer repository. It aggregates public PHP packages installable with Composer.

## Getting Started

### Define Your Dependencies

Put a file named `composer.json` at the root of your project, containing your project dependencies:

```
{
  "require": {
    "vendor/package1": "1.3.2",
    "vendor/package2": "1.*",
    "vendor/package3": "2.0.3"
  }
}
```

For more information about packages versions usage, see the [composer documentation](#).

### Install Composer In Your Project

Run this in your command line:

```
curl -sS https://getcomposer.org/installer | php
```

Or [download composer.phar](#) into your project root.

See the Composer documentation for complete [installation instructions](#) on various platforms.

### Install Dependencies

Execute this in your project root.

```
php composer.phar install
```

### Autoload Dependencies

## Publishing Packages

### Define Your Package

Put a file named `composer.json` at the root of your package, containing this information:

```
{
  "name": "your-vendor-name/package-name",
  "description": "A short description of what your package does",
  "require": {
    "php": "5.3.3 || ^7.0",
    "another-vendor/package": "1.*"
  }
}
```

This is the strictly minimal information you have to give.

For more details about package naming and the fields you can use to document your package better, see the [about](#) page.

### Commit The File

You surely don't need help with that.

### Publish It

[Login](#) or [register](#) on this site, then hit the [submit](#) button in the menu.

Once you entered your public repository URL in there, your package will be automatically crawled periodically. You just have to make sure you keep the `composer.json` file up to date.

# Packagist

- Composers repository för PHP-paket
- Paketen är fria att använda i dina projekt
- Använder **semantisk versionering** för att skilja mellan olika versioner av paket

# Semantisk versionering

1 . 0 . 0 . b1

major minor micro qualifier

Semantisk versionering: <http://semver.org/>

Lockart, *Modern PHP*. p. 61



# Autoloading

```
<?php
require __DIR__ . '/vendor/autoload.php';

$log = new Monolog\Logger('name');
$log->pushHandler(new Monolog\Handler\StreamHandler('app.log', Monolog\Logger::WARNING));
$log->addWarning('Foo');
```

# Composer – ett exempel

Exempel: Skapa ett Composer-projekt

```
johan@suell: ~/composer-test
Arkiv Redigera Visa Sök Terminal Hjälp

johan@suell:~/composer-test$ composer init

Welcome to the Composer config generator

This command will guide you through creating your composer.json config.

Package name (<vendor>/<name>) [johan/composer-test]:
Description []: Detta är ett Composer-exempel
Author [koddas <johan@idioti.se>, n to skip]:
Minimum Stability []:
Package Type (e.g. library, project, metapackage, composer-plugin) []: library
License []: MIT

Define your dependencies.

Would you like to define your dependencies (require) interactively [yes]?
Search for a package: slim/slim
Enter the version constraint to require (or leave blank to use the latest version):
Using version ^3.9 for slim/slim
Search for a package:
Would you like to define your dev dependencies (require-dev) interactively [yes]?
?
Search for a package: phpunit

Found 15 packages matching phpunit
[0] phpunit/phpunit-mock-objects
[1] phpunit/phpunit
[2] phpunit/php-token-stream
[3] phpunit/php-timer
[4] phpunit/php-text-template
[5] phpunit/php-file-iterator
[6] phpunit/php-code-coverage
[7] phpunit/phpunit-selenium
[8] symfony/phpunit-bridge
[9] brianium/paratest
[10] mybuilder/phpunit-accelerator
[11] johnkary/phpunit-speedtrap
```



“Coolt! I framtiden ska  
jag alltid använda mig  
av pakethantering!”

- Vettig student, 2018