Spezifikation von Paketversionen mit Pipenv

Die richtige Angabe von Paketversionen ist ein zentraler Bestandteil des Abhängigkeitsmanagements in Python-Projekten. Pipenv ermöglicht es, Versionen präzise zu definieren, um sicherzustellen, dass die Pakete in der Umgebung kompatibel sind und die gewünschte Funktionalität bieten. Hier sind die wichtigsten Techniken zur Versionenspezifikation in Pipenv.

1. Einfache Installation ohne Versionsangabe

Standardmäßig installiert Pipenv die neueste verfügbare Version eines Pakets, wenn keine Version angegeben wird:

```
pipenv install <paketname>
```

Beispiel

```
pipenv install requests
```

Hierbei wird die neueste Version von `requests` installiert, und die exakte Version wird anschließend in der `Pipfile.lock` gesperrt.

2. Installation einer bestimmten Version

Um eine bestimmte Version eines Pakets zu installieren, können Sie die Version mit `==` angeben:

```
pipenv install <paketname>==<version>
```

Beispiel

```
pipenv install requests==2.25.1
```

Dies sorgt dafür, dass genau die Version `2.25.1` von `requests` installiert wird und in der `Pipfile.lock` gespeichert bleibt.

3. Angabe eines Versionsbereichs

Manchmal ist es sinnvoll, eine Version innerhalb eines bestimmten Bereichs zuzulassen, z. B. alle Patch-Updates innerhalb einer Haupt- und Nebenversionsnummer. Dies ermöglicht es, sicherheitsrelevante oder kompatible Updates automatisch zu nutzen.

- **Mindestens eine Version** (`>=`): Erlaubt alle Versionen ab einer bestimmten Version.

- **Höchstens eine Version** (`<=`): Beschränkt auf Versionen bis zu einer bestimmten Version.
- **Kombination von Mindest- und Höchstversion** (`>=` und `<`): Definiert einen Versionsbereich.

Beispiele

```
pipenv install requests>=2.25
```

Hierbei wird mindestens Version 2.25 von `requests` installiert, aber neuere kompatible Versionen werden zugelassen.

```
pipenv install requests>=2.25,<3.0
```

Dies erlaubt alle Versionen von `requests` ab 2.25, aber unter Version 3.0, was häufig bei größeren API-Änderungen nützlich ist.

4. Verwendung von „Tilde“- und „Caret“-Operatoren

Pipenv unterstützt auch spezielle Operatoren wie `~=` und `^`, die in der Python-Welt häufig verwendet werden, um kompatible Versionen zu definieren.

- `~=` **(Kompatibilität innerhalb der Minor-Version)**: Installiert die neueste Version innerhalb der gleichen Minor-Version.

  ```
  pipenv install requests~=2.25
  ```

  Dies entspricht Versionen ab 2.25 und unter 2.26.

- `^` **(Caret-Kompatibilitätsoperator)**: Installiert die neueste Version innerhalb der Major-Version, solange die API-Richtlinien kompatibel bleiben.

  ```
  pipenv install requests^2.25
  ```

  Dies schließt alle Versionen ab 2.25 und unter 3.0 ein.

5. Spezifikation in der `Pipfile`

In der `Pipfile` können Sie Versionen manuell anpassen, falls Sie spezifische Anforderungen haben. Die Syntax entspricht den oben gezeigten Beispielen und kann manuell bearbeitet werden:

```
[packages]
requests = ">=2.25,<3.0"
```

Pipenv wird diese Angabe respektieren, wenn die Umgebung neu synchronisiert wird.