About FastAPI versions

FastAPI is already being used in production in many applications and systems. And the test coverage is kept at 100%. But its development is still moving quickly.

New features are added frequently, bugs are fixed regularly, and the code is still continuously improving.

That's why the current versions are still `0.x.x`, this reflects that each version could potentially have breaking changes. This follows the Semantic Versioning conventions.

You can create production applications with **FastAPI** right now (and you have probably been doing it for some time), you just have to make sure that you use a version that works correctly with the rest of your code.

Pin your `fastapi` version

The first thing you should do is to "pin" the version of **FastAPI** you are using to the specific latest version that you know works correctly for your application.

For example, let's say you are using version `0.45.0` in your app.

If you use a `requirements.txt` file you could specify the version with:

```
fastapi==0.45.0
```

that would mean that you would use exactly the version `0.45.0`.

Or you could also pin it with:

```
fastapi>=0.45.0,<0.46.0
```

that would mean that you would use the versions `0.45.0` or above, but less than `0.46.0`, for example, a version `0.45.2` would still be accepted.

If you use any other tool to manage your installations, like Poetry, Pipenv, or others, they all have a way that you can use to define specific versions for your packages.

Available versions

You can see the available versions (e.g. to check what is the current latest) in the Release Notes.

About versions

Following the Semantic Versioning conventions, any version below `1.0.0` could potentially add breaking changes.

FastAPI also follows the convention that any "PATCH" version change is for bug fixes and non-breaking changes.

!!! tip The "PATCH" is the last number, for example, in `0.2.3`, the PATCH version is `3`.

So, you should be able to pin to a version like:

```
fastapi>=0.45.0,<0.46.0
```

Breaking changes and new features are added in "MINOR" versions.

!!! tip The "MINOR" is the number in the middle, for example, in `0.2.3`, the MINOR version is `2`.

Upgrading the FastAPI versions

You should add tests for your app.

With **FastAPI** it's very easy (thanks to Starlette), check the docs: Testing

After you have tests, then you can upgrade the **FastAPI** version to a more recent one, and make sure that all your code is working correctly by running your tests.

If everything is working, or after you make the necessary changes, and all your tests are passing, then you can pin your `fastapi` to that new recent version.

About Starlette

You shouldn't pin the version of `starlette`.

Different versions of **FastAPI** will use a specific newer version of Starlette.

So, you can just let **FastAPI** use the correct Starlette version.

About Pydantic

Pydantic includes the tests for **FastAPI** with its own tests, so new versions of Pydantic (above `1.0.0`) are always compatible with FastAPI.

You can pin Pydantic to any version above `1.0.0` that works for you and below `2.0.0`.

For example:

```
pydantic>=1.2.0,<2.0.0
```