

Integration Tests with Super Powers

And even more...

Alexandre Figura · Site Reliability Engineer @ SysEleven GmbH





A Brilliant Masterpiece

The Python Magazine



Irresistibly Entertaining

Edinburgh Times



Golden Snake Awards

Nominated for

Best Speaker

Alexandre Figura



**The Most Exciting
Talk of the Year**

Scottish Insiders

Who Am I?

- Use **Python** with 🥰 since **2014**.
- 🇫🇷 Live in **Berlin** ✈️ since 2016 🇩🇪
- Always eat **2** desserts at lunch 🍦
- Work with **cool folks** at **SysEleven GmbH**,
best company in town.



Tests and Programmers

♥ A True Love Story ♥



A Well-Kept Secret

There is **one**
And **only one**
Reason **why** we write tests.

We want our **Pull-Requests** to get **accepted**.

And incidentally,
to make it **easier** to maintain our code in the **future**.

Lazy Mocking

Or how harmful mocks can be...



Mocks

Advantages

- Easy to write.

Drawbacks

- Dependant on **implementation**,
- Make **refactoring** harder,
- **Hide** errors and behavioral changes.

Conclusion

Never use them*





Please
DO NOT
Try this at home



Please **Do Not** Try Mocks at Home

Alternative Testing Strategies

...Pimp my Mock™...



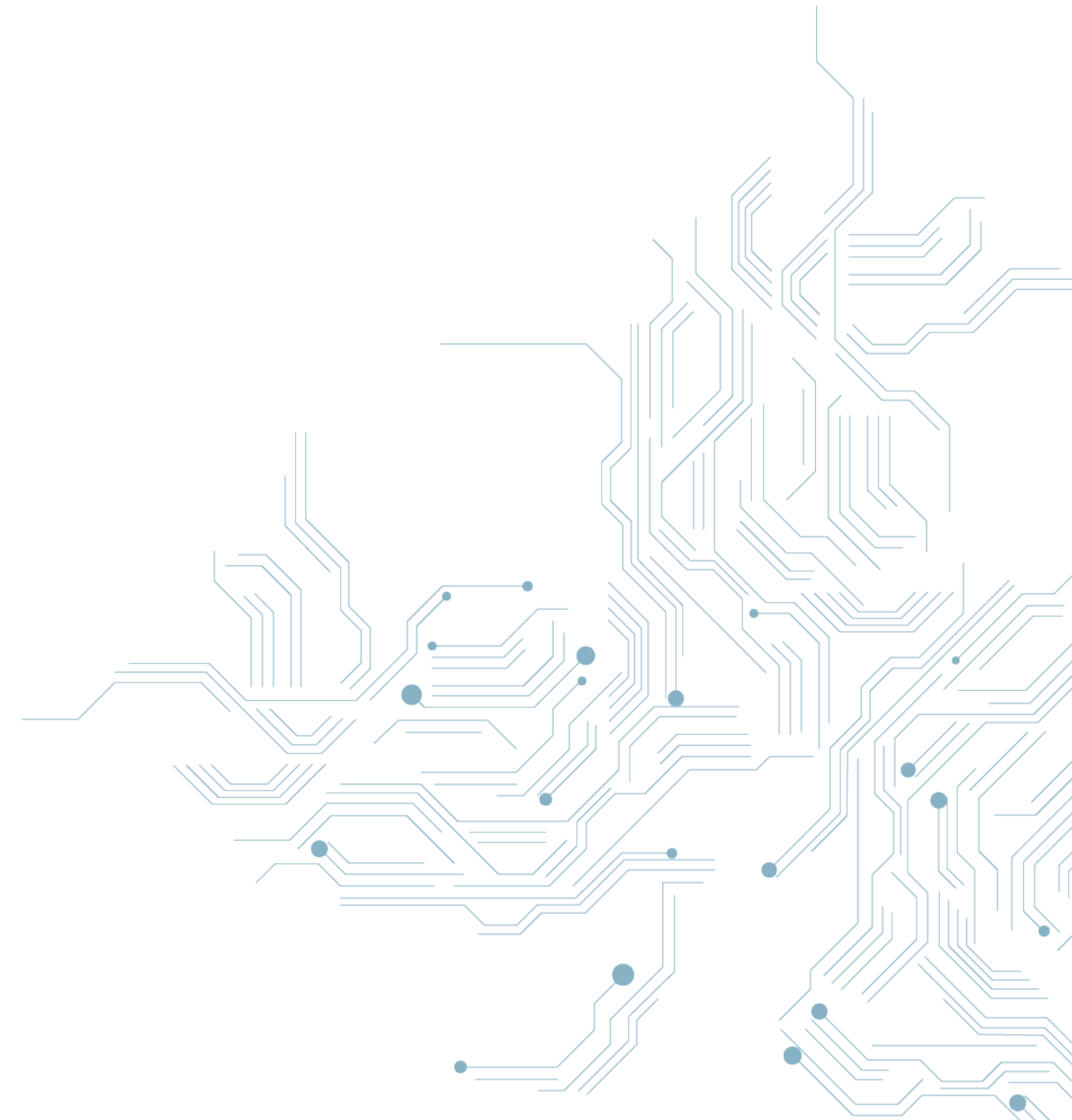
Basic Concepts

Integration Tests are built around two ideas:

1. **Dependency Injection,**
2. **Interface Testing.**

It sounds complicated.

But it isn't.



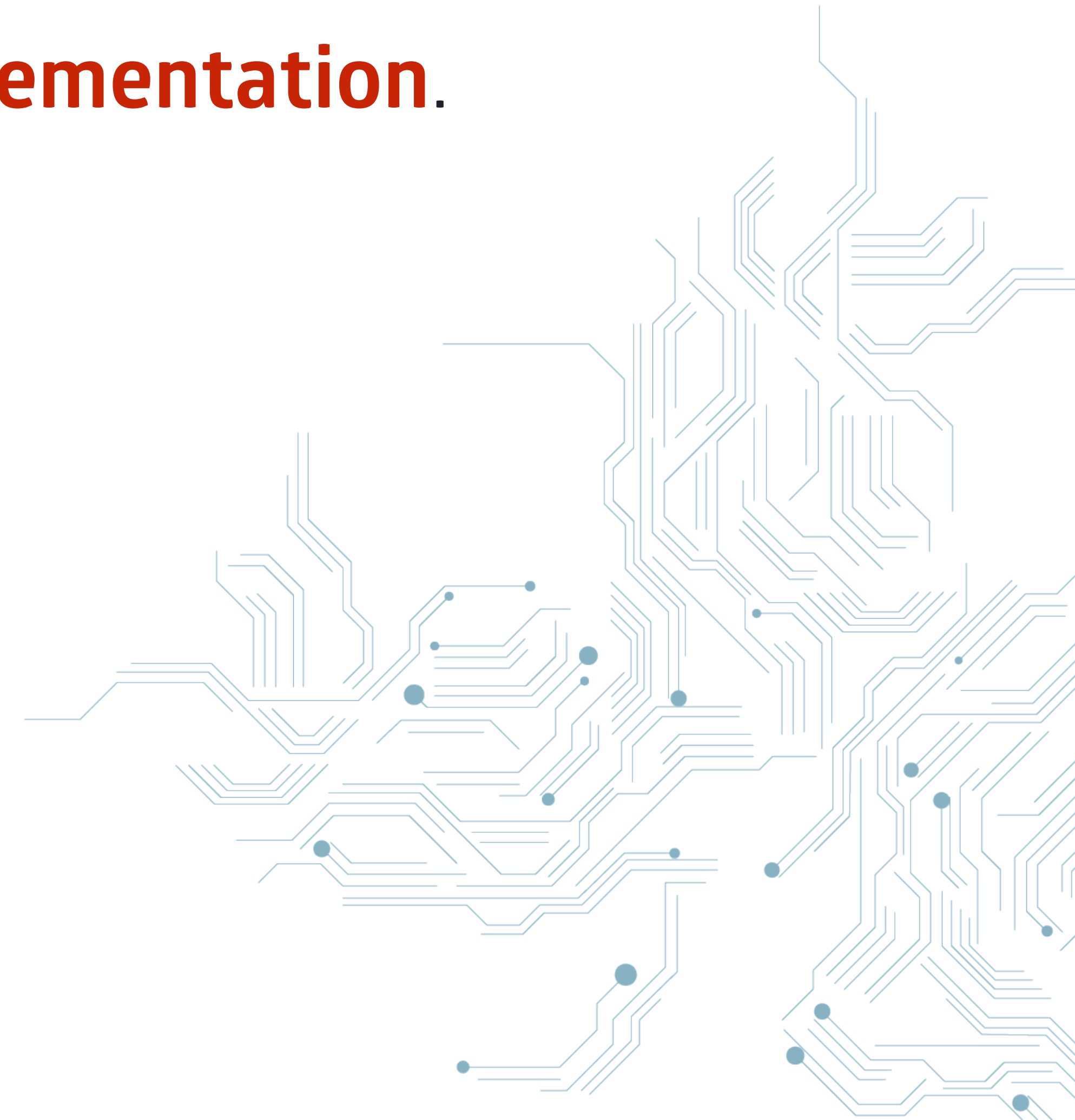
Summary

Advantages:

- Tests not dependant anymore on **implementation**.
- Easy to change **dependency** later on.
- Make **refactoring** easier.

Drawbacks:

- Require *way more* work at first.



Real World Example

When practice meets theory...



To the Cloud and Beyond

Use Case

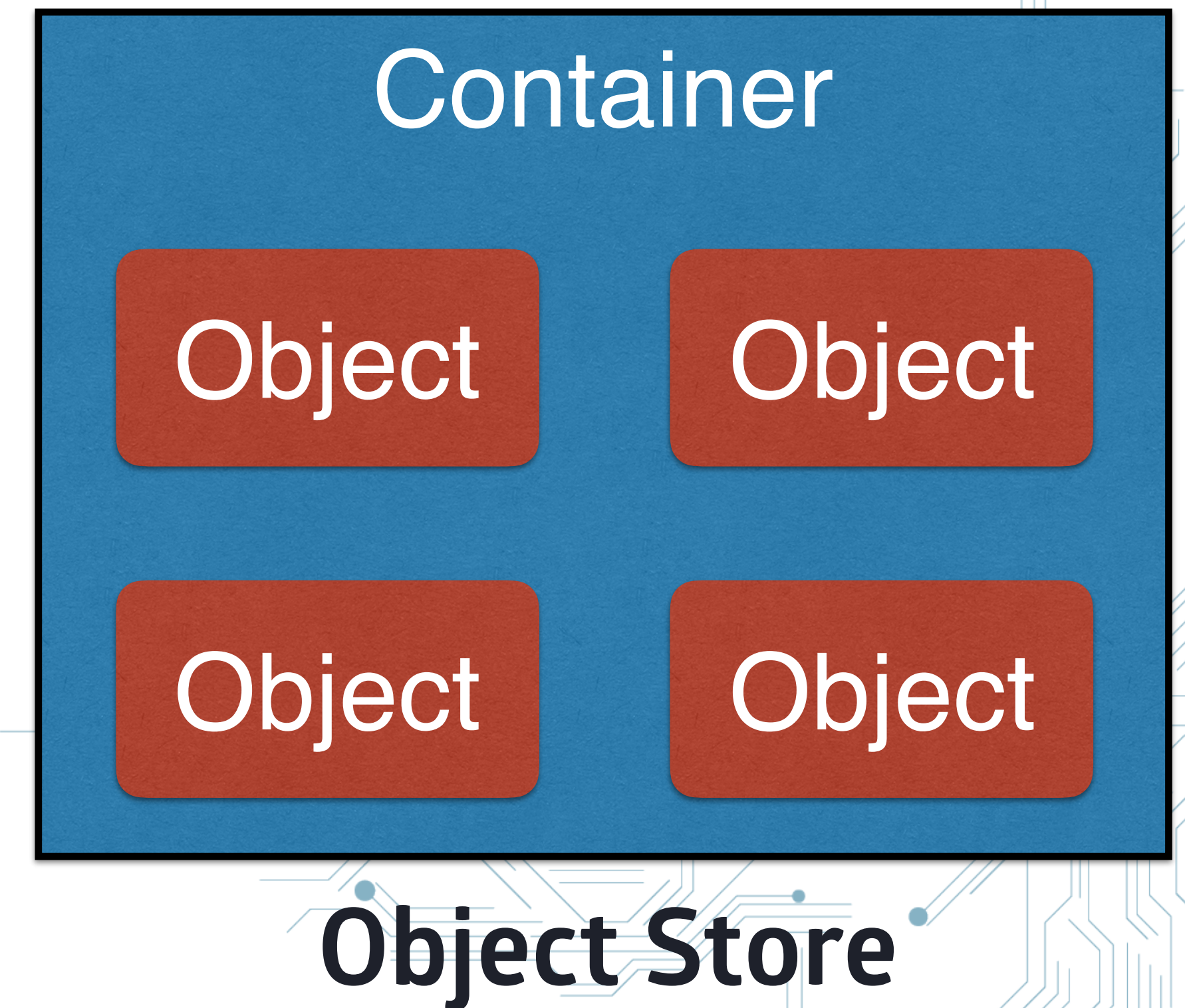
We want to upload a **static website**

To the **Cloud**

Using OpenStack **Object Store**'s API.

Links:

- Officiel API, - Fake Implementation,
- Interface Tests,
- Parametrizing with Pytest.



Additional Examples of Dependency Injection

Let's eat the snake!



Mock Injection

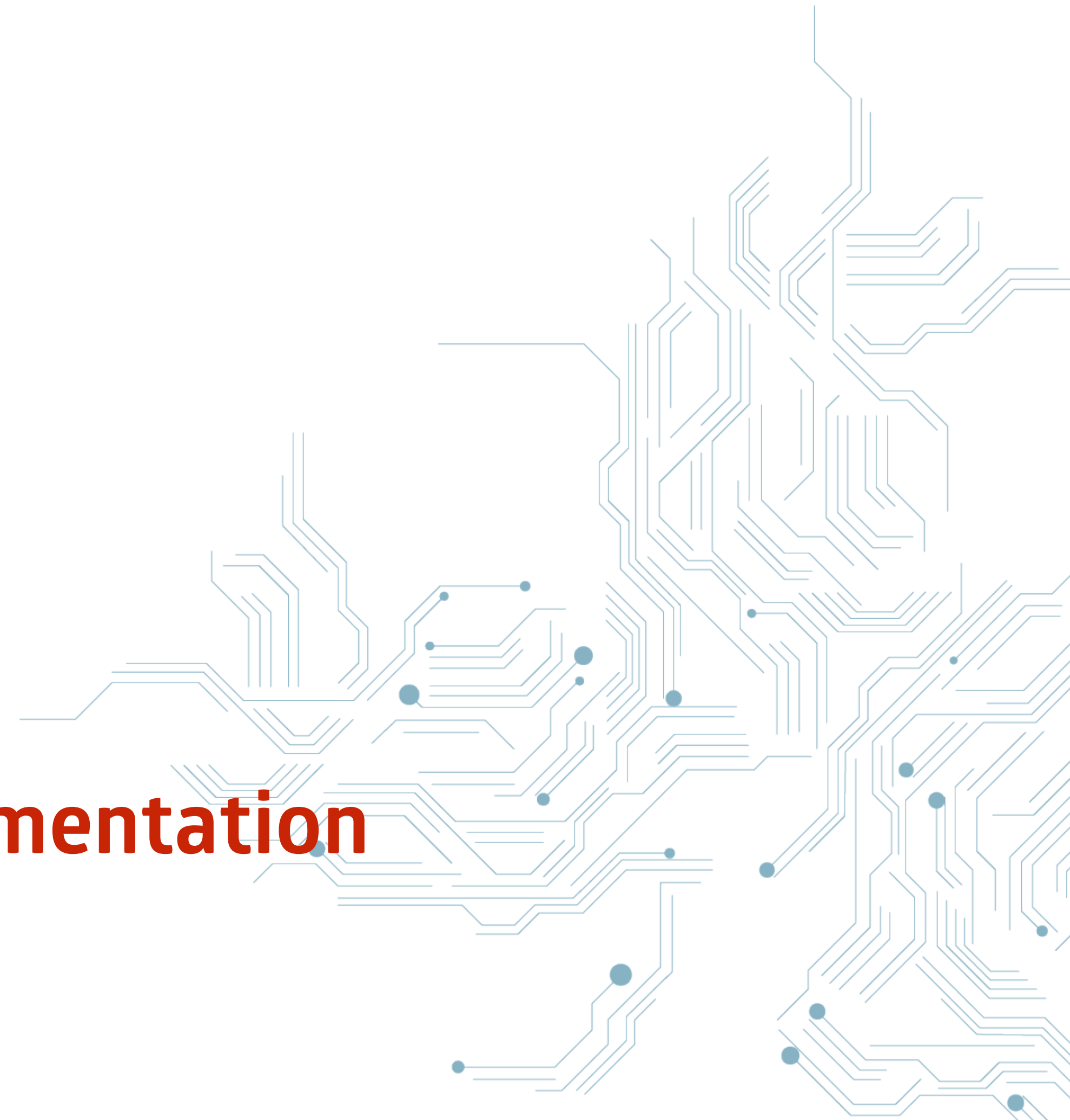
Too much work to write Fake dependencies?

- Write a **Mock-like** implementation,
- **Inject** it as a dependency.

Useful for simulating complex systems.

Advantages:

- Easy as mocks,
- Tests remaining independent of **implementation**
(makes **refactoring** easier).



Minimalist Interface Testing

Use Case:

I make Holiday movies

Recorded on **Blu-ray**

And need to convert them to **MKV**.

Problem:

A Blu-Ray is 30-50GB large.

Solution:

Only check for **syntax errors**.

Links:

- Syntax Errors Testing,
- Fake Implementations,
- Dependency Injection.

💰 Bonus 💰

Running inside Docker Compose

Because Buzzwords matter...

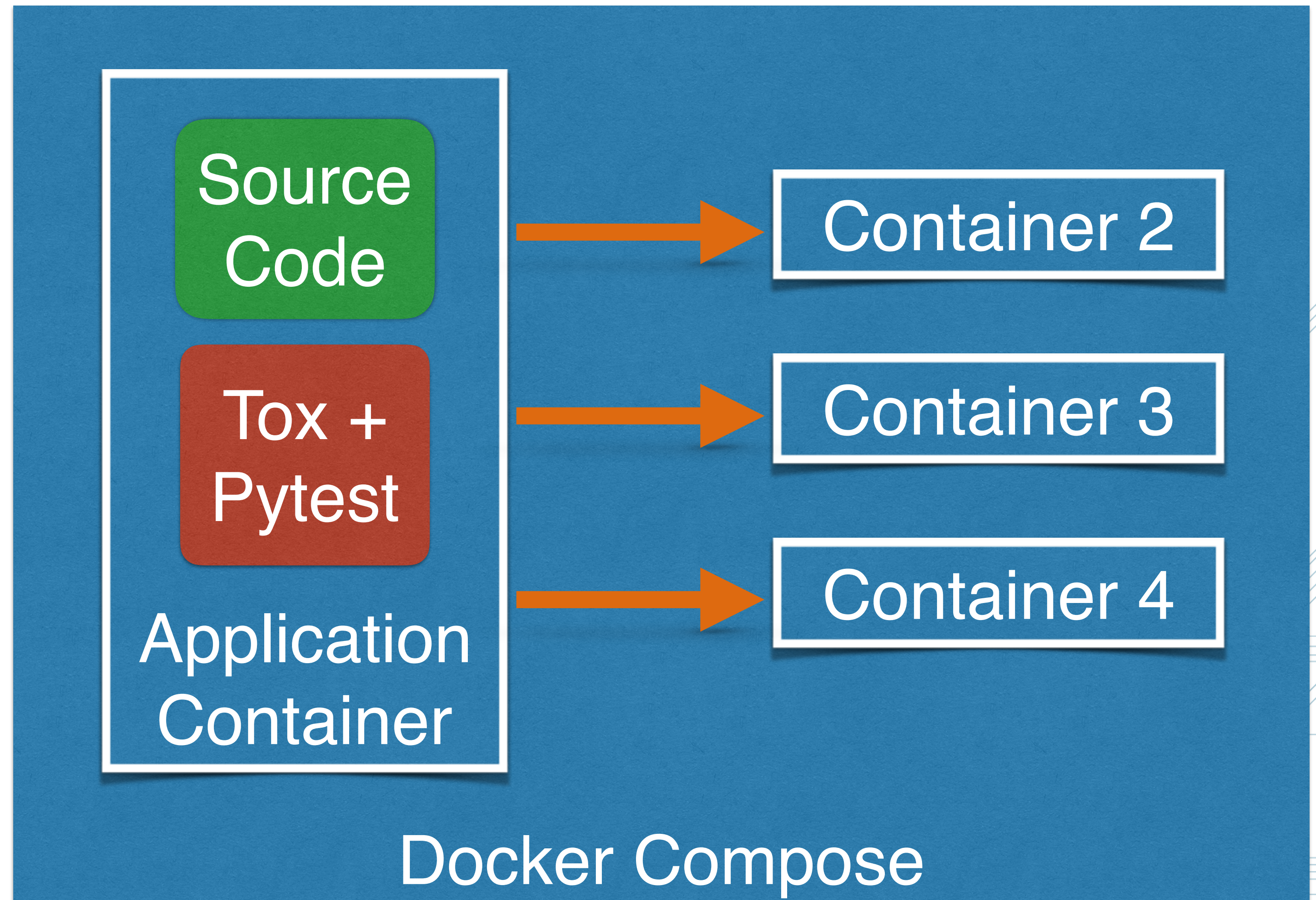


Standard Workflow

Tests independent
of running platform.

Working example
available on [Github](#).

build **passing**



Conclusion

What you should remember



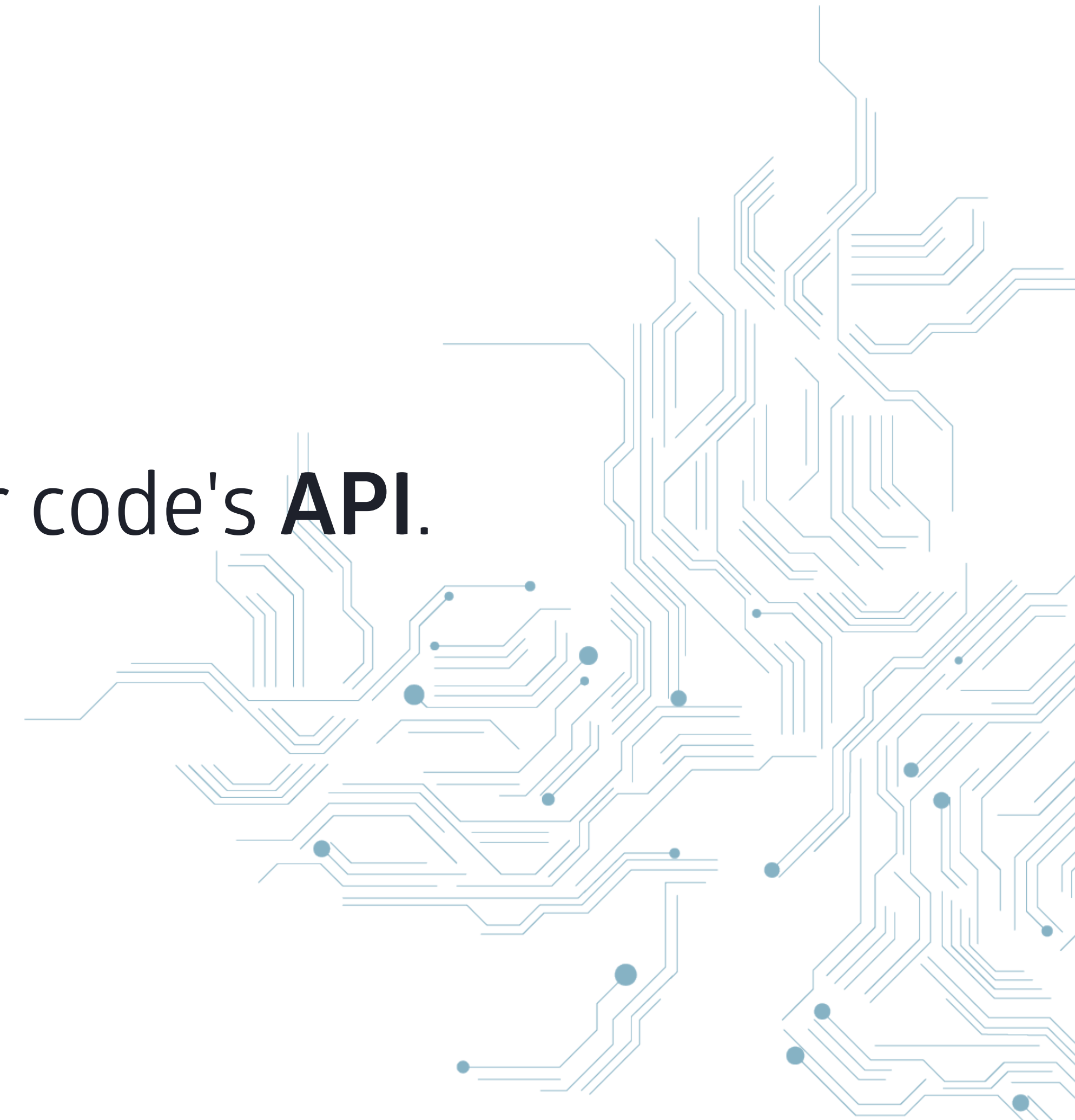
Summary

Mocks make **refactoring** harder.

Interface Testing is more flexible,
but involves **more work**.

Dependency Injection will improve your code's **API**.

Always **ask** yourself (or around)
if you **need 100%** test coverage.





Everything is about **Tradeoffs**

We're hiring! Interested? Just say: "Hello!"

Mail: jobs@syseleven.de

WhatsApp, SMS: +49 171/8934073

